
Señas Chapinas: Traductor de LENSEGUA

Stefano Alberto Aragoni Maldonado, Carol Andréé Arévalo Estrada,
Jose Miguel Gonzalez y Gonzalez, Luis Diego Santos Cuéllar,
Roberto Vallecillos Chinchilla



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



Señas Chapinas: Traductor de LENSEGUA

Trabajo de graduación en modalidad de Megaproyecto Tecnológico
presentado por

Stefano Alberto Aragoni Maldonado, Carol Andréé Arévalo Estrada,
Jose Miguel Gonzalez y Gonzalez, Luis Diego Santos Cuéllar,
Roberto Vallecillos Chinchilla

para optar al grado académico de Licenciados en Ingeniería en Ciencia
de la Computación y Tecnologías de la Información

Guatemala
2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



Señas Chapinas: Traductor de LENSEGUA

Trabajo de graduación en modalidad de Megaproyecto Tecnológico
presentado por

Stefano Alberto Aragoni Maldonado, Carol Andréé Arévalo Estrada,
Jose Miguel Gonzalez y Gonzalez, Luis Diego Santos Cuéllar,
Roberto Vallecillos Chinchilla

para optar al grado académico de Licenciados en Ingeniería en Ciencia
de la Computación y Tecnologías de la Información

Guatemala
2024

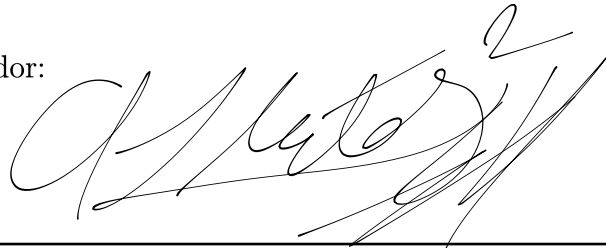
Vo.Bo.:



(f)

MSc. Luis Alberto Suriano Saravia

Tribunal Examinador:



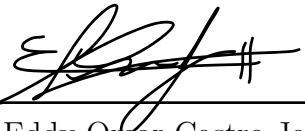
(f)

MSc. Luis Alberto Suriano Saravia



(f)

Ing. Douglas Leonel Barrios Gonzalez



(f)

Ing. Eddy Omar Castro Jauregui

Fecha de aprobación: 29 de noviembre del 2024.

El proyecto *Señas Chapinas: Traductor de LENSEGUA* emerge como una respuesta innovadora ante la necesidad crítica de desarrollar herramientas tecnológicas que faciliten la inclusión efectiva de la comunidad sorda en Guatemala. Este proyecto no solo busca abordar las barreras de comunicación existentes, sino también empoderar a las personas sordas para que puedan participar plenamente en todos los aspectos de la vida social, educativa y profesional.

La integración de tecnologías de inteligencia artificial en este contexto representa un avance significativo en la manera en que abordamos los desafíos de accesibilidad y comunicación. Al desarrollar un traductor de LENSEGUA (Lengua de Señas Guatemalteca) basado en modelos de lenguaje avanzados, este proyecto establece un precedente importante en la aplicación de soluciones tecnológicas para resolver problemáticas sociales complejas. La iniciativa no solo busca facilitar la comunicación cotidiana, sino también promover una mayor comprensión y apreciación de la riqueza lingüística y cultural de la comunidad sorda guatemalteca.

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que han contribuido a la realización de este proyecto, cada una de las cuales ha sido fundamental en su desarrollo.

Primero, expresamos nuestra gratitud a nuestros asesores, los Ingenieros Dennis Aldana, Miguel Novella, Luis Alberto Suriano y Javier Fong, profesores de la Universidad del Valle, por su invaluable guía y apoyo a lo largo de todo el proceso de investigación y redacción de este trabajo. Su experiencia y dirección experta fueron fundamentales para superar los retos académicos y prácticos de este proyecto.

Estamos profundamente agradecidos con ASEDES, especialmente con Niurka Waleska Bendfeldt Rosada y Alain de León, por proporcionarnos materiales, entrevistas y otros recursos necesarios para llevar a cabo este trabajo. Su colaboración fue indispensable para entender mejor las necesidades y desafíos de la comunidad sorda.

Nuestro reconocimiento a las alumnas practicantes de ASEDES: Evelyn Cacao, Any Max y Ruth Amézquita, quienes generosamente permitieron que las grabáramos mientras realizaban señas, contribuyendo significativamente a la autenticidad y calidad del contenido de este proyecto.

Agradecemos la colaboración de la profesora Pamela Ramírez, quien contribuyó en el diseño del logo de la aplicación. Su trabajo fue esencial, ya que el logo desempeña un papel crucial en la identidad visual y la coherencia del diseño de la aplicación.

Finalmente, un agradecimiento especial a Antonio Barrientos, Director General de En-Señas, y a Gabriela Velázquez, maestra de En-Señas, por su apertura y disposición para compartir su conocimiento y experiencia, las cuales fueron cruciales para este proyecto.

A todos ustedes, nuestro más profundo respeto y gratitud por su apoyo y contribuciones.

Prefacio	v
Agradecimientos	vii
Lista de figuras	xvii
Lista de cuadros	xx
Resumen	xxii
1. Introducción	1
2. Justificación	3
3. Objetivos	5
3.1. Objetivo general	5
3.2. Objetivos específicos	5
4. Marco teórico	7
4.1. Discapacidad auditiva	7
4.1.1. Causas de la sordera	7
4.1.2. Clasificación de la sordera	8
4.1.3. Impacto de la discapacidad auditiva en las personas	8
4.1.4. Barreras y desafíos enfrentados por la comunidad sorda en Guatemala	9
4.2. Lengua de señas	9
4.2.1. Historia de la lengua de señas	9
4.2.2. Lengua de señas en la actualidad	10
4.3. Lengua de Señas de Guatemala (LENSEGUA)	10
4.3.1. Historia	10
4.3.2. Variaciones regionales	11
4.3.3. Gramática y estructura	11
4.3.4. Aprendizaje y recursos	12
4.4. Fundamentos de visión por computadora	13
4.4.1. Procesamiento de imágenes	13
4.4.2. Técnicas básicas	13
4.4.3. Herramientas	13
4.5. Aplicaciones de visión por computadora	14
4.5.1. Medicina	14

4.5.2. Industria automotriz	14
4.5.3. Seguridad y vigilancia	14
4.5.4. Manufactura	14
4.6. Redes neuronales	14
4.6.1. Fundamentos	14
4.6.2. Tipos de redes neuronales	14
4.6.3. Aplicaciones en visión por computadora	15
4.7. Evaluación de modelos de visión por computadora	15
4.7.1. Matriz de confusión	15
4.7.2. Sensibilidad	15
4.7.3. Puntuación F1	16
4.8. Natural Language Processing (NLP)	16
4.8.1. Técnicas y fundamentos	16
4.8.2. Modelos principales	17
4.9. Arquitecturas basadas en transformadores	19
4.9.1. Generative Pre-trained Transformer (GPT)	19
4.9.2. Large Language Model Meta AI (LLaMA)	21
4.9.3. Métodos de entrenamiento y adaptación	22
4.9.4. Métricas de rendimiento y evaluación	23
4.10. Explainable Artificial Intelligence (XAI)	24
4.10.1. Local Interpretable Model-agnostic Explanations (LIME)	24
4.11. Sistema operativo: Linux server	25
4.11.1. ¿Qué es Linux?	25
4.11.2. Historia de Linux	25
4.11.3. Linux server	25
4.12. Virtualización	25
4.13. Multipass	26
4.14. VPN	26
4.14.1. ¿Qué es una VPN?	26
4.14.2. ¿Para qué sirven las VPNs?	26
4.15. Arquitectura de base de datos	26
4.15.1. Bases de datos relacionales	26
4.15.2. PostgreSQL	27
4.15.3. Arquitectura de base de datos	27
4.16. APIs	27
4.16.1. ¿Qué es una API?	27
4.16.2. ¿Cómo usar una API?	27
4.16.3. Implementar APIs a bases de datos	27
4.16.4. Implementar APIs para recepción y devolución de datos	28
4.16.5. ¿Qué es Crontab?	28
4.17. Bases de diseño y filosofía de DARPA	28
4.17.1. Costo-efectividad y control	28
4.17.2. Flexibilidad y adaptabilidad	28
4.17.3. Seguridad y privacidad	28
4.18. Pruebas de eficiencia y de extremo a extremo en el servidor mediante APIs	29
4.18.1. ¿Qué es una prueba de eficiencia?	29
4.18.2. ¿Qué es una prueba de extremo a extremo (E2E)?	29
4.18.3. Impacto de la eficiencia y de las pruebas de extremo a extremo en el servidor	29
4.19. Herramientas para pruebas de eficiencia y de extremo a extremo	30
4.19.1. Monitoreo del sistema	30
4.19.2. Pruebas de carga y end to end	30
4.20. Diseño de interfaz de usuario (UI)	31
4.20.1. Definición de UI	31
4.20.2. Elementos de la interfaz de usuario	31

4.20.3. Estándares de diseño	32
4.21. Experiencia de usuario (UX)	36
4.21.1. Definición de UX	36
4.21.2. Diferencias entre UX y UI	36
4.21.3. Tipos de experiencia de usuario	36
4.21.4. Proceso de experiencia de usuario	36
4.22. Desarrollo móvil en Android	38
4.22.1. Razones para elegir Android como plataforma de desarrollo	38
4.22.2. Arquitectura de aplicaciones Android	38
4.22.3. Buenas prácticas de desarrollo Android	39
5. Metodología	42
5.1. Visión por computadora	42
5.1.1. Delimitación del universo de palabras	42
5.1.2. Recopilación de datos	43
5.1.3. Preparación del conjunto de datos	44
5.1.4. Procesamiento de los videos	45
5.1.5. Labeling de los datos	46
5.1.6. Normalización de los datos	46
5.1.7. Oversampling	47
5.1.8. Entrenamiento del modelo	47
5.1.9. Evaluación del modelo	48
5.1.10. Evaluación del conjunto de datos	49
5.1.11. Publicación del modelo	51
5.2. Procesamiento de lenguaje natural (GPT-3.5-Turbo)	52
5.2.1. Creación de conjunto de datos	52
5.2.2. Fine-tuning	55
5.2.3. Prompt engineering	57
5.2.4. Implementación y evaluación del sistema	57
5.3. Procesamiento de lenguaje natural (LLaMA)	59
5.3.1. Creación de conjunto de datos	59
5.3.2. Configuración inicial	60
5.3.3. Fine-tuning	62
5.3.4. Comparación de modelos fine-tuneados (GPT-3.5-Turbo y LLaMA)	63
5.4. Infraestructura de red	63
5.4.1. Levantamiento de un sistema operativo y configuración de VPN	63
5.4.2. Desarrollo de la arquitectura de base de datos	65
5.4.3. Implementación de APIs	67
5.4.4. Virtualización del servidor para múltiples modelos	77
5.4.5. Implementación de pruebas de eficiencia	79
5.4.6. Pruebas de carga	79
5.4.7. Pruebas de extremo a extremo	83
5.4.8. Implementación de pruebas CVE para seguridad	83
5.4.9. Pruebas de seguridad con Lynis	84
5.4.10. Mejoras de seguridad en <code>/etc/sysctl.conf</code>	85
5.4.11. Monitoreo continuo de seguridad con ClamAV	85
5.5. Diseño y desarrollo móvil	86
5.5.1. Investigación de mercado	86
5.5.2. Desarrollo de interfaz y experiencia de usuario	100
5.5.3. Desarrollo de aplicación móvil para Android	139
5.5.4. Pruebas con usuarios finales	155

6. Resultados	162
6.1. Visión por computadora	162
6.1.1. Conjunto de datos	162
6.1.2. Proceso iterativo de desarrollo del modelo	166
6.1.3. Evaluación del modelo	182
6.1.4. Aplicaciones del modelo	183
6.2. Procesamiento de lenguaje natural (GPT-3.5-Turbo)	185
6.2.1. Fine-tuning	185
6.2.2. Prompt engineering	190
6.2.3. Retroalimentación de la comunidad sorda	193
6.3. Procesamiento de lenguaje natural (LLaMA)	196
6.3.1. Fine-tuning (versión 1)	196
6.3.2. Retroalimentación de la comunidad sorda	201
6.3.3. Fine-tuning (versión 2)	203
6.4. Infraestructura de red	206
6.4.1. Redireccionamiento de puertos para acceso SSH externo	206
6.4.2. Resultados de la prueba de carga	207
6.4.3. Resultados de la prueba E2E	210
6.4.4. Resultados de la prueba de seguridad con Lynis	211
6.5. Diseño y desarrollo móvil	213
6.5.1. Comprensión de la situación de comunidad sorda en Guatemala	213
6.5.2. Diseño centrado en el usuario	214
6.5.3. Desarrollo de aplicación móvil para Android	214
6.5.4. Pruebas con usuarios finales	214
7. Análisis de resultados	215
7.1. Visión por computadora	215
7.2. Procesamiento de lenguaje natural (GPT-3.5-Turbo)	218
7.3. Procesamiento de lenguaje natural (LLaMA)	220
7.4. Infraestructura de red	222
7.5. Diseño y desarrollo móvil	223
8. Conclusiones	226
9. Recomendaciones	228
10. Bibliografía	232
11. Anexos	240
Carta de solicitud de acceso a la VPN	242

Lista de figuras

1. Arquitectura de red neuronal recurrente.	18
2. Arquitectura de transformer	19
3. Arquitectura de Generative Pre-trained Transformer (GPT).	20
4. Arquitectura de Large Language Model Meta AI (LLaMA).	22
5. Tamaño óptimo de botones según su prioridad	33
6. Espaciado óptimo de botones según su tamaño	33
7. Proceso de diseño de experiencia de usuario	37
8. Cuota de tráfico web móvil por sistema operativo	38
9. Diagrama de funcionamiento MVVM	39
10. Ejemplos de captura de los videos para el conjunto de datos 72.	43
11. Ejemplo de reflejo de un video	44
12. Proceso de preparación del conjunto de datos.	45
13. Ejemplo de uso de MediaPipe 53.	46
14. Ejemplo de datos procesados.	47
15. Ejemplo de una red neuronal <i>feedforward</i> sencilla.	48
16. Implementación de mejoras en la red neuronal <i>feedforward</i> .	48
17. Ejemplo de visualización de datos con PCA.	50
18. Ejemplo de matriz de confusión.	51
19. Reunión de equipo de trabajo con ASEDES.	53
20. Generación recursiva de nuevas frases.	54
21. Comparación entre CPU y GPU en el procesamiento paralelo.	60
22. Diagrama de entrenamiento con LoRA.	61
23. LevantadoOpenVPN.	64
24. ConexionSSH.	64
25. Diagrama entidad relación.	67
26. Diagrama de flujo.	71
27. Flujo de trabajo completo del sistema de APIs.	72
28. Estado de las máquinas virtuales dentro del sistema	79
29. Resultados de la prueba de carga con 50 usuarios concurrentes.	81
30. Resultados de la prueba de carga con 100 usuarios concurrentes.	81
31. Resultados de la prueba de carga con 200 usuarios concurrentes.	81
32. Resultados de la prueba de carga con 300 usuarios concurrentes.	81
33. Resultados de la prueba de carga con 400 usuarios concurrentes.	81
34. Resultados de la prueba de carga con 500 usuarios concurrentes.	82
35. Resultados de la prueba de carga con 600 usuarios concurrentes.	82
36. Resultados de la prueba de carga con 700 usuarios concurrentes.	82

37. Resultados de la prueba de carga con 800 usuarios concurrentes.	82
38. Resultados de la prueba de carga con 900 usuarios concurrentes.	82
39. Resultados de la prueba de carga con 1100 usuarios concurrentes.	83
40. Muestra de aplicación “Hand Talk Translator”	87
41. Muestra de aplicación “SLAIT”	87
42. Muestra de aplicación “Lenguaje de señas IA”	88
43. Muestra de aplicación “AI Sign: Sign Language”	89
44. Muestra de aplicación “Sign Language Translator AI”	90
45. Entrevista En-Señas	94
46. Género encuesta 2	95
47. Edad encuesta 2	96
48. Conocimiento persona sorda encuesta 2	96
49. Conocimiento LENSEGUA encuesta 2	96
50. Relevancia de la aplicación encuesta 2	97
51. Relevancia de la aplicación para personas con conocidos sordos encuesta 2	97
52. Entrevista colectiva En-Señas	99
53. Lluvia de ideas para diagrama de afinidad	101
54. Agrupación de ideas para diagrama de afinidad	102
55. Diagrama de afinidad	103
56. Persona 1 - Laura	105
57. Persona 2 - Ricardo	105
58. Persona 3 - Felipe	106
59. Persona 4 - Sofía	106
60. Persona 5 - Marta	107
61. Persona 6 - Jorge	107
62. Mapa de empatía - Laura	108
63. Mapa de empatía - Ricardo	108
64. Mapa de empatía - Felipe	109
65. Mapa de empatía - Sofía	109
66. Mapa de empatía - Marta	110
67. Mapa de empatía - Jorge	110
68. Sombreros para pensar	111
69. Planteamiento del problema Señas Chapinas	111
70. Primera vez usando la aplicación	113
71. Grabación de video	113
72. Guardando video	114
73. Reporte	114
74. Diccionario de palabras	115
75. Reto diario	115
76. Mapa de sitio	116
77. Grabar video	117
78. Guardar video	118
79. Abrir video guardado	118
80. Repertir grabación de video	119
81. Completar reto	119
82. Reportar traducción	120
83. Diccionario	120
84. Wireframe bajo nivel	121
85. Wireframe nivel medio	121
86. Wireframe alto nivel	122
87. Wireframe alto nivel luego de retroalimentación	123
88. Primer logo	125
89. Logo vectorizado	125
90. Logo modernizado	126

91. Logo con colores	126
92. Logo Señas Chapinas	126
93. Paleta de colores logo	127
94. Paleta de colores aplicación	128
95. Paleta colores perfil	128
96. Contraste blanco y azul	129
97. Contraste azul y blanco	129
98. Contraste gris y blanco	129
99. Contraste blanco y rojo	130
100. Contraste verde quetzal y azul	130
101. Contraste verde claro y azul	130
102. Tipografía Nunito	131
103. Primer prototipo	133
104. Segundo prototipo	134
105. Tercer prototipo	136
106. Cuarto prototipo	138
107. Ilustraciones señas	139
108. Navegación principal	141
109. Navegación video	142
110. Navegación perfil	142
111. Diccionario	143
112. Traducción	143
113. Flujo cambio de contraseña	146
114. Deeplink página web	146
115. Plantilla correo	147
116. Ejemplo de correo enviado	147
117. Página web	149
119. Fotos de página web	150
118. Herramientas de página web	150
120. Políticas de privacidad	150
121. Ícono	151
122. Capturas de pantalla	151
123. Banner promocional	151
124. Preguntas frecuentes	152
125. Prueba cerrada Play Store	153
126. Solicitud para producción	153
127. Ejemplo historia usuario	154
128. Cronograma Kanban	154
129. Expo UVC	155
130. Cambios primera prueba con usuarios	156
131. Cambio En-Señas 1	157
132. Cambio En-Señas 2	158
133. Fragmento tutorial	158
134. Demo En-Señas	159
135. Flujo muestra de grabación	160
136. Análisis de similitud entre las clases <i>como</i> y <i>agua</i>	163
137. Análisis de similitud entre las clases <i>como</i> y <i>beber</i>	163
138. Análisis de similitud entre las clases <i>cuando</i> y <i>donde</i>	164
139. Análisis de similitud entre las clases <i>sed</i> y <i>quien</i>	164
140. Análisis de similitud entre todas las clases	165
141. Balance de clases en el conjunto de datos de entrenamiento	165
142. Modelo inicial	166
143. Historial de entrenamiento del modelo base	167

144. Matriz de confusión del modelo base	167
145. Modelo con aumento de complejidad	168
146. Historial de entrenamiento del modelo con aumento de complejidad	169
147. Matriz de confusión del modelo con aumento de complejidad	169
148. Modelo base con dropout	170
149. Historial de entrenamiento del modelo base con dropout	171
150. Matriz de confusión del modelo base con dropout	171
151. Primera iteración de fine tuning de la tasa de dropout	172
152. Historial de entrenamiento del primer modelo de fine tuning de la tasa de dropout	173
153. Matriz de confusión del primer modelo de fine tuning de la tasa de dropout	173
154. Segunda iteración de fine tuning de la tasa de dropout	174
155. Historial de entrenamiento del segundo modelo de fine tuning de la tasa de dropout	175
156. Matriz de confusión del segundo modelo de fine tuning de la tasa de dropout	175
157. Modelo con normalización por lotes	176
158. Historial de entrenamiento del modelo con normalización por lotes	177
159. Matriz de confusión del modelo con normalización por lotes	177
160. Primer modelo con dropout y normalización por lotes	178
161. Historial de entrenamiento del primer modelo con de dropout y normalización por lotes	179
162. Matriz de confusión del primer model con dropout y normalización por lotes	179
163. Segundo modelo con dropout y normalización por lotes	180
164. Historial de entrenamiento del segundo modelo con dropout y normalización por lotes	181
165. Matriz de confusión del segundo modelo con dropout y normalización por lotes	181
166. Reconocimiento de lengua de señas de Guatemala en tiempo real (una palabra)	182
167. Reconocimiento de lengua de señas de Guatemala en tiempo real (múltiples palabras)	183
168. Reconocimiento de la palabra <i>ayer</i> en tiempo real	184
169. Reconocimiento de la palabra <i>comer</i> en tiempo real	184
170. Reconocimiento de la palabra <i>universidad</i> utilizando el API	185
171. Reconocimiento de la palabra <i>tu</i> utilizando el API	185
172. Evolución de la pérdida durante el <i>fine-tuning</i> final del modelo GPT-3.5-Turbo.	186
173. Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo estándar para la frase “antes tu policía llamar pregunta”.	187
174. Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo <i>fine-tuneado</i> para la frase “antes tu policía llamar pregunta”.	187
175. Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo estándar para la frase “futuro hospital él ir”.	188
176. Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo <i>fine-tuneado</i> para la frase “futuro hospital él ir”.	188
177. Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo estándar para la frase “pasado yo ir no”.	189
178. Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo <i>fine-tuneado</i> para la frase “pasado yo ir no”.	189
179. Aplicación web desarrollada para facilitar interacción con modelo <i>fine-tuneado</i> .	193
180. Evolución de la pérdida durante el <i>fine-tuning</i> del modelo LLaMA preliminar.	196
181. Evolución de la pérdida durante el <i>fine-tuning</i> del modelo LLaMA con <i>early stopping</i> .	197
182. Evolución de la pérdida durante el <i>fine-tuning</i> del modelo LLaMA con <i>early stopping</i> y estándar LoRA.	197
183. Evolución de la pérdida durante el <i>fine-tuning</i> del modelo LLaMA con <i>early stopping</i> , estándar LoRA y <i>fine-tuning</i> supervisado.	198
184. Resultados de LIME para la interpretación generada por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para la frase “como clima hoy”.	199
185. Resultados de LIME para la interpretación generada por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para la frase “ayer viernes limpiar casa todo día”.	200
186. Resultados de LIME para la interpretación generada por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para la frase “mar gusta él mar gustar ella no”.	201

187. Resultados de LIME para la interpretación generada por el modelo LLaMA <i>fine-tuneado</i> (versión 2) para la frase “ojalá hoy carro mucho no”.	204
188. Resultados de LIME para la interpretación generada por el modelo LLaMA <i>fine-tuneado</i> (versión 2) para la frase “antes tu policía llamar pregunta”.	205
189. Resultados de LIME para la interpretación generada por el modelo LLaMA <i>fine-tuneado</i> (versión 2) para la frase “pasado yo ir no”.	205
190. Métricas de conexiones y solicitudes	207
191. Códigos de respuesta HTTP	208
192. Tiempo de respuesta y tiempo de respuesta Upstream	208
193. Métricas de CPU, memoria y carga del sistema	209
194. Tráfico de red	210
195. Resultados de la prueba E2E, mostrando un 100% de éxito en todas las operaciones realizadas.	211
196. Resultado de la primera prueba de seguridad con Lynis mostrando un índice de robustez de 58.	211
197. Resultado de la segunda prueba de seguridad con Lynis mostrando un índice de robustez de 60.	212
198. Resultado de la tercera prueba de seguridad con Lynis mostrando un índice de robustez de 62.	212
199. Resumen del estado del servidor.	213

Lista de cuadros

1. Estructura del conjunto de datos desarrollado.	52
2. Ejemplos de perturbaciones interpretadas por el modelo <i>fine-tuneado</i> junto con sus distancias de Lenneshtein normalizadas correspondientes.	57
3. Tabla: user	65
4. Tabla: video	66
5. Tabla: traducción	66
6. Tabla: dictionary	66
7. Resumen de rutas de API: <i>user routes</i>	67
8. Resumen de rutas de API: <i>video routes</i>	68
9. Resumen de rutas de API: <i>traduction routes</i>	68
10. Resumen de rutas de API: <i>dictionary routes</i>	68
11. Resumen de rutas de API: <i>profile routes</i>	68
12. Resumen de rutas de API: <i>mail routes</i>	69
13. Resumen del sistema	78
14. Distribución de recursos entre las máquinas virtuales y el host	78
15. Desempeño del modelo base	167
16. Desempeño del modelo con aumento de complejidad	169
17. Desempeño del modelo base con dropout	171
18. Desempeño del primer modelo de fine tuning de la tasa de dropout	173
19. Desempeño del segundo modelo de fine tuning de la tasa de dropout	175
20. Desempeño del modelo con normalización por lotes	177
21. Desempeño del primer modelo con dropout y normalización por lotes	179
22. Desempeño del segundo modelo con dropout y normalización por lotes	181
23. Desempeño del segundo modelo de la combinación de dropout y normalización por lotes	182
24. Valores de parámetros y pérdidas de validación en experimentos preliminares	186
25. Parámetros utilizados para el <i>fine-tuning</i> del modelo GPT-3.5-Turbo.	186
26. Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “antes tu policía llamar pregunta”.	188
27. Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “futuro hospital él ir”.	189
28. Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “pasado yo ir no”.	190

29.	Distancias de Levenshtein promedio calculadas a partir de las interpretaciones generadas por el modelo GPT-3.5-Turbo (estándar) y el modelo <i>fine-tuneado</i> , utilizando diversos <i>prompts</i> .	192
30.	Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “ayer cine tu ir pregunta”.	194
31.	Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “pasado yo medicina comprar para mamá”.	194
32.	Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “ayer abuelo llamar tu pregunta”.	194
33.	Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “antes tienda tu amiga ver pregunta”.	195
34.	Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión <i>fine-tuneada</i> para la frase “tu opinión cuál pregunta”.	195
35.	Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo <i>fine-tuneado</i> , utilizando diferentes <i>prompts</i> , para la frase “hospital yo necesitar ir ahora ojalá mucho carro no porque emergencia”.	196
36.	Valores promedio de BLEU en experimentos preliminares.	198
37.	Valores BLEU de interpretaciones generadas por el modelo LLaMA con <i>early stopping</i> , estándar LoRA y <i>fine-tuning</i> supervisado.	198
38.	Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para la frase “como clima hoy”.	199
39.	Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para la frase “ayer viernes limpiar casa todo día”.	200
40.	Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para la frase “mar gusta él mar gustar ella no”.	201
41.	Comentarios otorgados por un intérprete de LENSEGUA ante interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 1) para diversas frases.	202
42.	Distancias de Levenshtein promedio calculadas a partir de las interpretaciones generadas por el modelo GPT-3.5-Turbo <i>fine-tuneado</i> y el modelo LLaMA <i>fine-tuneado</i> (versión 2).	204
43.	Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 2) y el modelo GPT-3.5-Turbo <i>fine-tuneado</i> para la frase “ojalá hoy carro mucho no”.	204
44.	Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 2) y el modelo GPT-3.5-Turbo <i>fine-tuneado</i> para la frase “antes tu policía llamar pregunta”.	205
45.	Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA <i>fine-tuneado</i> (versión 2) y el modelo GPT-3.5-Turbo <i>fine-tuneado</i> para la frase “pasado yo ir no”.	206

Señas Chapinas: Traductor de LENSEGUA es un proyecto enfocado en el desarrollo de una aplicación móvil diseñada para interpretar la Lengua de Señas de Guatemala (LENSEGUA) y traducirla a español gramaticalmente correcto, facilitando la comunicación entre personas sordas y oyentes. La solución se basa en una arquitectura modular compuesta por tecnologías avanzadas que trabajan de forma integrada para garantizar robustez, eficiencia y accesibilidad.

El módulo de visión por computadora procesa grabaciones para identificar y reconocer las señas realizadas por los usuarios. A partir de esta información, modelos de procesamiento de lenguaje natural transforman las señas detectadas en oraciones completas y coherentes, adaptándose tanto a contextos cotidianos como a situaciones de emergencia. La infraestructura de red y el *backend* funcionan como el núcleo técnico que conecta todos los componentes, asegurando estabilidad, escalabilidad y seguridad. Finalmente, el diseño centrado en la experiencia del usuario proporciona una interfaz intuitiva y accesible, adaptada a las necesidades específicas de la comunidad sorda.

La colaboración activa con la comunidad sorda guatemalteca fue esencial en todas las etapas del proyecto, desde la definición de requerimientos hasta la validación final. Este enfoque participativo no solo asegura que *Señas Chapinas* sea una herramienta funcional, sino también un recurso profundamente conectado con las necesidades prácticas y culturales de sus usuarios.

CAPÍTULO 1

Introducción

En Guatemala, aproximadamente 300,000 personas enfrentan algún grado de discapacidad auditiva [126]. En un esfuerzo por reducir la brecha lingüística existente entre la comunidad sorda y la oyente, en el 2020, el Congreso de la República de Guatemala promulgó el Decreto 3-2020. Esta legislación reconoce oficialmente a la lengua de señas de Guatemala (LENSEGUA) como un medio de comunicación compuesto por movimientos corporales y una gramática propia de las personas sordas [28]. A pesar de este importante avance legislativo, su impacto hasta el momento ha sido limitado, lo que subraya la urgencia de implementar soluciones innovadoras para mejorar la comunicación y la inclusión de las personas sordas en la sociedad guatemalteca.

En este contexto, a través de este megaproyecto se busca crear un sistema de reconocimiento e interpretación de LENSEGUA. Para esto, se plantea el desarrollo de una aplicación móvil para Android que le permita a los usuarios grabar videos de personas utilizando la lengua de señas guatemalteca. Una vez grabados, se analizará automáticamente los videos para reconocer y, posteriormente, interpretar las señas al idioma español. Con un enfoque en vocabulario clave para situaciones cotidianas y de emergencia, esta herramienta busca facilitar las interacciones diarias y contribuir a mejorar la calidad de vida de la comunidad sorda en Guatemala.

La solución combina un diseño intuitivo y culturalmente relevante con tecnología avanzada que garantiza eficiencia, seguridad y procesamiento confiable de los datos. Este enfoque garantiza una experiencia de usuario accesible y funcional, preparada para responder a las necesidades cambiantes de sus usuarios. Mediante una colaboración activa y constante con la comunidad sorda, *Señas Chapinas: Traductor de LENSEGUA* aspira a ser más que una herramienta tecnológica; busca convertirse en un puente hacia la inclusión y el entendimiento, promoviendo una sociedad más equitativa y accesible.

La comunicación es un derecho fundamental y un elemento clave para la interacción humana, siendo indispensable en áreas como la educación, el trabajo y la participación activa en la sociedad [87]. En Guatemala, la Asociación de Sordos de Guatemala (ASEGUA) recomienda que, para garantizar una inclusión adecuada de las 250,000 personas que utilizan LENSEGUA como su principal herramienta de comunicación, deberían existir al menos 10 personas oyentes capacitadas en lengua de señas por cada una de ellas [31]. No obstante, el país está lejos de alcanzar este estándar: solo 11,500 personas han participado en cursos básicos de lengua de señas, y únicamente existen 95 intérpretes certificados en todo el territorio nacional [79].

Esta situación, según la “Encuesta Sociolingüística de la Comunidad Sorda en Guatemala”, marginaliza a la población sorda y la limita a relacionarse exclusivamente con otras personas sordas o con hijos de adultos sordos (HDAS). La escasez de personas oyentes capacitadas en LENSEGUA y de intérpretes certificados crea barreras significativas para la comunicación y la participación de personas sordas en la sociedad guatemalteca [99]. Ante esta realidad, surge la idea del proyecto *Señas Chapinas: Traductor de LENSEGUA*.

Aprovechando la amplia adopción de teléfonos inteligentes en Guatemala [119], donde la mayoría de estos dispositivos operan con el sistema Android [119], este proyecto busca desarrollar una aplicación móvil que actúe como un puente de comunicación eficiente y accesible. La aplicación pretende fortalecer la autonomía de las personas sordas en diversas situaciones cotidianas y de emergencia, promoviendo al mismo tiempo el uso de LENSEGUA. Además, se sustenta en un marco legal favorable, como el Decreto del Congreso de la República de Guatemala Número 3-2020, que reconoce a la Lengua de Señas de Guatemala como un medio de comunicación oficial [29].

Además, a través de *Señas Chapinas* se busca fomentar la inclusión laboral, facilitar el acceso a servicios esenciales y promover las interacciones sociales, contribuyendo al enriquecimiento de la comunidad guatemalteca. Es por ello que este proyecto representa un avance significativo hacia la construcción de una sociedad que valora la diversidad y garantiza igualdad de oportunidades para todos, utilizando tecnología móvil para superar las barreras de comunicación de manera eficiente y efectiva.

3.1. Objetivo general

Diseñar y desarrollar *Señas Chapinas*, una aplicación para dispositivos Android que sea capaz de traducir la lengua de señas guatemalteca (LENSEGUA) a texto gramaticalmente correcto en español, haciendo uso de modelos avanzados de visión por computadora y procesamiento de lenguaje natural, junto con una infraestructura de red segura y eficiente.

3.2. Objetivos específicos

- Desarrollar un sistema de visión por computadora destinado al reconocimiento de la lengua de señas de Guatemala. Para ello, se creará un conjunto de datos que incluya al menos veinticinco palabras de esta lengua, seleccionando aquellas que permitan formar una amplia variedad de frases de uso cotidiano.
- Adaptar un *large language model*, específicamente GPT-3.5-Turbo, para que asimile la gramática de LENSEGUA, permitiéndole interpretar oraciones que utilicen dicha gramática y las escriba correctamente en español.
- Desarrollar una herramienta basada en LLaMA que sea capaz de comprender la gramática de LENSEGUA y generar interpretaciones coherentes en español a partir de oraciones estructuradas en dicha gramática.
- Configurar un servidor seguro y eficiente que optimice el uso de recursos para administrar modelos de inteligencia artificial, procesar videos, y ofrecer un ambiente accesible para pruebas y despliegues mediante APIs.
- Realizar una investigación de mercado y entrevistas para comprender necesidades, diseñar flujos e interfaces intuitivas, y desarrollar la aplicación *Señas Chapinas* integrando servicios externos para la traducción de LENSEGUA.

4.1. Discapacidad auditiva

La discapacidad auditiva es una condición que limita la capacidad de una persona para escuchar. Esta puede manifestarse en distintos niveles de intensidad, desde una pérdida leve de audición hasta la sordera total. Además, esta condición puede tener diversas causas, como factores genéticos o exposición prolongada a ruidos fuertes [132]. Es importante destacar que, independientemente de su origen o grado, la discapacidad auditiva puede representar un desafío significativo en la vida diaria de quienes la experimentan, afectando su comunicación, educación e integración social [133].

4.1.1. Causas de la sordera

La Federación Mundial de Sordos (WFD) reporta que en todo el mundo hay 70 millones de personas sordas, de las cuales 34 millones son niños. Además, se estima que alrededor de 3 de cada 1,000 bebés nacen con sordera [133]. En este contexto, es importante destacar que la sordera en los niños puede manifestarse durante el período prenatal debido a diversos factores. Por ejemplo, hay mutaciones genéticas (hereditarias y no hereditarias) que pueden aumentar la probabilidad de sordera congénita. Asimismo, existen infecciones intrauterinas, como la rubéola o el herpes, que también pueden desencadenar la sordera en el feto. Otra causa común es la exposición a medicamentos por parte de la madre que pueden provocar diversas malformaciones [88].

La prematuridad de un bebé es también un factor que puede influir significativamente en el desarrollo de problemas auditivos. Concretamente, los bebés prematuros tienen un mayor riesgo de desarrollar sordera debido a la posibilidad de que sus oídos no hayan alcanzado su completo desarrollo al momento del nacimiento. Además, los bebés prematuros comúnmente experimentan enfermedades que aumentan la probabilidad de desarrollar sordera, como la hiperbilirrubinemia (ictericia grave). Por otro lado, durante el parto pueden surgir complicaciones que resultan en asfixia perinatal, evento que puede ocasionar daños al sistema auditivo y nervioso del recién nacido [88].

Cabe destacar que la sordera no solo aparece durante el periodo prenatal o perinatal, sino que también puede manifestarse en cualquier etapa de la vida, incluyendo la infancia, la adolescencia y la edad adulta. En niños y jóvenes, infecciones recurrentes del oído (otitis media crónica) pueden provocar pérdida de audición si no se tratan adecuadamente. A su vez, la meningitis es una infección

que también puede ocasionar daños en el nervio auditivo y, consecuentemente, resultar en pérdida auditiva. Finalmente, la exposición continua a ruidos fuertes puede dañar gradualmente la audición a lo largo del tiempo. Esto es especialmente relevante en los jóvenes y adultos que utilizan con más frecuencia audífonos a un volumen alto [133].

4.1.2. Clasificación de la sordera

Una persona sorda, según la Organización Mundial de la Salud (OMS), es aquella cuya capacidad auditiva no supera los 20 decibelios (dB) en ambas orejas [133]. Sin embargo, es importante tener en cuenta que la sordera puede manifestarse en diferentes grados, los cuales dependen de la gravedad de la pérdida auditiva y de la ubicación específica del problema en el oído.

En términos de grado de pérdida auditiva, la sordera se puede clasificar en leve, moderada, grave y profunda. La pérdida auditiva leve se refiere a una persona que puede escuchar algunos sonidos relacionados con el habla, pero no puede escuchar susurros. A una persona con pérdida auditiva moderada le cuesta entender el habla a un volumen normal. Por otro lado, una persona con pérdida auditiva grave no escucha ningún sonido a volumen normal, solo puede oír sonidos fuertes. Por último, una persona con pérdida auditiva profunda (o con sordera total) no escucha nada, exceptuando algunos sonidos extremadamente fuertes [74].

Al considerar la ubicación de la pérdida auditiva, esta puede clasificarse en tres tipos principales: conductiva, neurosensorial o mixta. La pérdida auditiva conductiva ocurre cuando hay un bloqueo que impide el paso del sonido del oído externo al medio. En cambio, la pérdida auditiva neurosensorial ocurre cuando hay daño en el oído interno o en el nervio auditivo. Esto resulta en que los sonidos no puedan ser procesados y posteriormente transmitidos al cerebro de forma correcta. Finalmente, la pérdida auditiva mixta, como su nombre sugiere, involucra una combinación de factores tanto conductivos como neurosensoriales. Esto puede ser el resultado de múltiples condiciones que afectan tanto el oído externo como el interno [74].

4.1.3. Impacto de la discapacidad auditiva en las personas

Actualmente se reporta que el 80% de personas sordas residen en países de bajos ingresos, como Guatemala. En estos países, las condiciones precarias dificultan el acceso a servicios de salud auditiva adecuados, así como a tecnologías de asistencia y educación especializada. Como resultado, es frecuente encontrar niños, jóvenes e incluso adultos con pérdida de audición no solo no tratada, sino también no diagnosticada [133].

La sordera no tratada, según Dave Cutten, puede obstaculizar el desarrollo lingüístico y comunicativo de los niños, lo que incluye impedimentos en el desarrollo del vocabulario y dificultades para comprender y utilizar el lenguaje de manera efectiva [33]. Es fundamental destacar que los niños con sordera identificada suelen aprender a comunicarse a través de la lengua de señas, lo que mitiga en gran medida estos problemas. Sin embargo, independientemente de si los niños con sordera aprenden lengua de señas o no, existe una brecha lingüística entre la comunidad sorda y la oyente que tiene implicaciones psicológicas significativas. Por ejemplo, las personas sordas tienden a sentirse excluidas, solas, frustradas, enojadas y avergonzadas debido a que no pueden socializar fácilmente con sus familiares, compañeros y miembros de sus comunidades. Además, esta situación puede afectar negativamente el bienestar emocional de las personas sordas, generando sentimientos de ansiedad, depresión y baja autoestima [23].

En el caso de los adolescentes y adultos, la discapacidad auditiva puede provocar efectos psicológicos y emocionales similares a los descritos anteriormente. Además, esta condición puede influir negativamente en las oportunidades educativas y laborales de las personas, lo que a su vez puede conducir a una mayor exclusión y discriminación en la sociedad [21]. Por ejemplo, en Guatemala, de

las 10 escuelas para personas sordas, ninguna ofrece educación secundaria. Esto, a su vez, ocasiona una brecha educativa significativa para los adolescentes sordos, limitando así su futuro académico y profesional [99].

4.1.4. Barreras y desafíos enfrentados por la comunidad sorda en Guatemala

En 2007, Elizabeth Parks y Jason Parks realizaron un estudio con el objetivo de analizar la situación sociolingüística de la comunidad sorda en Guatemala. A través de entrevistas y encuestas, identificaron que menos del 50% de los sordos guatemaltecos reciben educación especializada. Además, observaron un bajo nivel de alfabetización, así como un pobre dominio del español entre este grupo. Según los autores, esta situación se atribuye a la presencia limitada de solo diez escuelas para los veintidós departamentos del país. De estas diez, tres son instituciones privadas, lo que las vuelve inaccesibles para muchas familias debido a limitaciones monetarias [99].

En cuanto a educación superior, según Edith Paz, únicamente dos universidades en Guatemala ofrecen servicios de intérpretes para estudiantes sordos, aunque estos servicios están limitados exclusivamente a los departamentos de computación. Es importante destacar que, si bien los estudiantes tienen la opción de contratar intérpretes privados, esta posibilidad está restringida a aquellos provenientes de familias de altos recursos. Debido a esto, muchas personas sordas encuentran dificultades para acceder a empleos con un salario digno. Por ejemplo, el salario promedio mensual para un guatemalteco es de Q2,000, sin embargo, las personas sordas suelen ganar –en promedio– menos de Q600 mensuales [99].

4.2. Lengua de señas

La lengua de señas es un sistema de comunicación utilizado por personas sordas y con discapacidad auditiva. Este utiliza gestos, movimientos de las manos, expresiones faciales y posturas corporales para transmitir ideas y emociones [89]. Es importante mencionar que, al igual que los idiomas hablados, las lenguas de señas varían considerablemente entre países, existiendo más de 300 variaciones en todo el mundo [131].

4.2.1. Historia de la lengua de señas

La lengua de señas se desarrolló de manera independiente por la comunidad sorda para satisfacer sus necesidades comunicativas. Históricamente estigmatizada y mal entendida, era considerada un lenguaje de gestos simple hasta que investigaciones realizadas en 1960 por William Stokoe revelaron su capacidad para expresar ideas complejas y estructuradas [108].

En el siglo XVIII, el Abad Charles-Michel de L'Épée fundó la primera escuela pública para sordos, marcando un cambio trascendental en la educación de esta comunidad, utilizando la lengua de señas como principal medio de enseñanza. Este avance no solo facilitó la comunicación y el aprendizaje, sino que también permitió que los sordos desempeñaran roles activos como educadores. La metodología de L'Épée se expandió internacionalmente, influyendo en la creación de escuelas y en el desarrollo de nuevas lenguas de señas [108].

Durante los siglos XIX y XX, las lenguas de señas ganaron reconocimiento como sistemas lingüísticos completos y estructurados, capaces de expresar una gama completa de ideas y emociones. En el siglo XX, el reconocimiento de los derechos lingüísticos de las comunidades sordas se amplió significativamente, afirmando la importancia de las lenguas de señas como herramientas educativas y

culturales esenciales [108].

A raíz de la necesidad de comunicación en las comunidades sordas, cada país ha desarrollado su propia lengua de señas, integrando a menudo estructuras de lenguas de señas extranjeras, como el *American Sign Language (ASL)*, así como señas locales únicas. Esto ha dado lugar a que cada país, e incluso regiones dentro de los mismos, tengan su propia lengua de señas con estructuras gramaticales y léxicos distintos [110].

4.2.2. Lengua de señas en la actualidad

En la actualidad, la lengua de señas se está adaptando a un entorno globalizado y tecnológicamente avanzado, donde las necesidades comunicativas evolucionan constantemente. Estos cambios han impulsado la creación de legislaciones, políticas, formación de asociaciones y el desarrollo de nuevas tecnologías destinadas a minimizar las barreras comunicativas. Un ejemplo significativo es la iniciativa de las Naciones Unidas al proclamar el 23 de septiembre como Día Internacional de las Lenguas de Señas, enfatizando la importancia de estas lenguas [97].

Sin embargo, a pesar de estos avances, persisten desafíos significativos. La falta de estandarización de las lenguas de señas a nivel global requiere que las personas aprendan la lengua de señas específica de cada comunidad, lo cual impide la existencia de una forma de comunicación internacional uniforme [110].

Además, la lengua de señas, a menudo catalogada como una lengua minoritaria, es aprendida solamente por una pequeña fracción de la población sin discapacidades auditivas. Esta limitada difusión crea una brecha de comunicación significativa, contribuyendo a la marginación de la comunidad sorda y limitando su participación plena en actividades sociales y económicas. Esto subraya la necesidad de una mayor educación y sensibilización sobre la lengua de señas para promover una verdadera inclusión [83].

4.3. Lengua de Señas de Guatemala (LENSEGUA)

4.3.1. Historia

El primer registro de la utilización de la lengua de señas en Guatemala se remonta a la escuela Fray Pedro Ponce de León, establecida en la Ciudad de Guatemala en 1946 para la educación de niños y niñas sordas. Sin embargo, según Edith Paz, esta institución tenía una filosofía oralista que prohibía el uso de señas y gestos para la comunicación. A pesar de estas restricciones, los estudiantes desarrollaron, durante el transcurso de 20 años, un sistema de señas para comunicarse entre ellos tanto dentro de las aulas como en público. Esta lengua de señas (también conocida como GSM) fue evolucionando con el paso del tiempo, incorporando influencias de “Señas Caseras”¹ provenientes de distintos departamentos del país y de sistemas similares utilizados en España, Cuba, Costa Rica, El Salvador y Estados Unidos [99] [17].

A finales del siglo XX, se inauguraron otras diez escuelas para los jóvenes con discapacidad auditiva. A diferencia de la Escuela Fray Pedro Ponce de León, todas estas fomentaban el uso de la lengua de señas. Entre ellas, dos enseñaban *American Sign Language (ASL)*, mientras que las otras una variación de la lengua de señas formalizada a finales de la década de los sesenta (GSM) [99].

En el 2001, el Comité Pro Ciegos y Sordos de Guatemala, una institución privada no lucrativa

¹Se conoce como “Señas Caseras” al sistema de comunicación que utilizan los niños sordos, que no han sido expuestos a la lengua de señas, con padres oyentes para poder comunicarse y desenvolverse en el ámbito familiar [42].

pionera en la educación y rehabilitación de personas con discapacidad auditiva, en conjunto con otros colaboradores, publicó el primer manual oficial de la Lengua de Señas de Guatemala (LENSEGUA). Este manual representó un avance crucial en la estandarización y enseñanza de la lengua de señas, proporcionando un recurso esencial para los estudiantes, profesionales y la comunidad en general interesada en aprender este método de comunicación [76]. Sin embargo, cabe destacar que este manual no tomó relevancia hasta el 2021, cuando el Congreso de la República de Guatemala aprobó la ‘Ley que Reconoce y Aprueba la Lengua de Señas de Guatemala’ (Decreto Número 135-96). Esta ley, como indica su nombre, reconoce a LENSEGUA como un medio de comunicación compuesto por un conjunto de movimientos corporales y una gramática propia de las personas sordas. Asimismo, establece que el Ministerio de Educación debe promover la introducción de LENSEGUA al sistema educativo nacional [28].

El Decreto estableció que todas las instituciones públicas y privadas deben garantizar la inclusión de LENSEGUA como parte de su comunicación y servicios. Además, se promueve la educación bilingüe (español y LENSEGUA) en las escuelas que atienden a estudiantes sordos, asegurando así su derecho a una educación equitativa y accesible [56].

Este reconocimiento no solo valida a LENSEGUA como una lengua completa y estructurada, sino que también impulsa la creación de políticas y programas destinados a mejorar la accesibilidad en todos los aspectos de la vida pública para la comunidad sorda, desde la educación hasta el acceso a los servicios de salud y legales. El Decreto promueve la inclusión y asegura que las personas con discapacidad auditiva tengan acceso a la educación y la información en lengua de señas, libre de cualquier discriminación [56].

4.3.2. Variaciones regionales

Aunque LENSEGUA es reconocida como la forma predominante de comunicación para la comunidad sorda en Guatemala, existen variaciones regionales que reflejan la diversidad cultural y lingüística dentro del país. Estas diferencias incluyen, pero no se limitan a, cambios sutiles en el vocabulario y los gestos utilizados. Por ejemplo, según Edith Paz, mientras que la Ciudad de Guatemala, Cobán y Quetzaltenango muestran similitudes significativas, San Marcos presenta un sistema ligeramente diferente, el cual se inspira aún más en el ASL [99].

4.3.3. Gramática y estructura

La gramática y estructura de LENSEGUA reflejan una vasta complejidad lingüística que permite a los usuarios expresar una amplia gama de conceptos y emociones. Este sistema de comunicación es completo con su propia sintaxis, léxico y reglas gramaticales. Aquí se describen algunas de las características distintivas de LENSEGUA [42] [43] [44]:

- **Morfología:** La morfología en LENSEGUA utiliza modificadores manuales y no manuales para alterar el significado de los signos básicos, incluyendo modificaciones para indicar número, tiempo, aspecto, y otros atributos gramaticales.
 - El signo para “comer” podría modificarse para expresar “comer mucho” mediante la repetición del signo o cambios en la expresión facial.
- **Ausencia de género y artículos:** Como en muchas lenguas de señas, LENSEGUA no utiliza género gramatical ni artículos.
 - Español: “la casa”, “el perro”.
 - LENSEGUA signa “casa” y “perro” sin modificadores adicionales.

- **No uso de preposiciones:** LENSEGUA omite preposiciones, que en español son cruciales para las relaciones espaciales o temporales. La relación se establece a través del contexto y la configuración de los signos.
 - Español: “en la casa”.
 - LENSEGUA: se usa gesto para indicar la ubicación relativa y el signo de casa.
- **Omisión de signos de puntuación y Mayúsculas:** LENSEGUA no utiliza signos de puntuación ni mayúsculas. La escritura refleja una secuencia continua de signos, que se diferencia notablemente de la estructura del español.
 - LENSEGUA: “Disculpar mi hija no llega colegio porque muy enferma tiene tos casa tomar medicinas”.
- **Verbos no conjugados:** En LENSEGUA, los verbos no se conjugan. El tiempo y el aspecto se indican con signos específicos al principio de la frase o a través de la expresión facial.
 - Español: “Yo estoy comiendo”.
 - LENSEGUA: se signa “yo comer”.
- **Orden gramatical:** El orden gramatical típico en LENSEGUA es Tiempo, Lugar, Sujeto, Objeto, Verbo (TLSOV), diferente al orden Sujeto, Verbo, Objeto (SVO) del español. Este orden facilita que el contexto temporal y espacial quede establecido claramente al inicio.
 - Español: “Yo ayer jugué fútbol”.
 - LENSEGUA: se signa “ayer yo fútbol jugar”.

4.3.4. Aprendizaje y recursos

LENSEGUA se puede aprender en varias instituciones y a través de recursos en línea que buscan facilitar el acceso y la difusión de esta lengua. Entre las principales entidades que ofrecen cursos y formación en LENSEGUA están [5](#):

- ASEDES (Asociación Educativa para el Sordo)
- ASORGUA (Asociación de Sordos de Guatemala)
- Benemérito Comité Prociegos y Sordos de Guatemala
- En-Señas Guatemala
- CESGUA (Coordinación de Educación y Servicios en Guatemala)
- INTERGUA (Coordinación de intérpretes de lengua de señas de Guatemala)
- ANDYSISC (Servicios de interpretación profesional de Lengua de Señas)
- FUNDAL
- ONG Sordos Latinos Guatemala

Estas organizaciones no solo proporcionan educación en LENSEGUA, sino que facilitan una serie de conferencias y talleres impartidos por especialistas para personas con discapacidad auditiva [5](#).

4.4. Fundamentos de visión por computadora

4.4.1. Procesamiento de imágenes

La visión por computadora es un campo de la inteligencia artificial que permite a las computadoras interpretar y comprender el contenido de las imágenes y videos. Este campo se basa en la adquisición, procesamiento y análisis de imágenes digitales para extraer información significativa. Los objetivos de la visión por computadora incluyen la automatización de tareas que requieren procesamiento visual, como el reconocimiento de objetos, la detección de patrones y el seguimiento de movimientos [48].

4.4.2. Técnicas básicas

Los principios básicos de la visión por computadora incluyen técnicas de procesamiento de imágenes, como la mejora de imágenes, la segmentación de imágenes y la extracción de características. La mejora de imágenes puede incluir el ajuste de contraste y eliminación de ruido. La segmentación de imágenes se enfoca en la división de una imagen en partes significativas, con el objetivo de descartar información que no sea útil. Por último, la extracción de características se enfoca en la identificación de elementos clave dentro de una imagen, como bordes o puntos de interés. Estas técnicas son fundamentales para el desarrollo de aplicaciones que requieren una comprensión detallada de las imágenes [7].

4.4.3. Herramientas

En el ámbito de la visión por computadora, existen varias herramientas y bibliotecas de software que facilitan la implementación de técnicas de procesamiento de imágenes y análisis visual. Entre las más utilizadas se encuentran OpenCV y MediaPipe.

Open Source Computer Vision Library (OpenCV) es una biblioteca de software libre de visión artificial y aprendizaje automático. Esta proporciona más de 2500 algoritmos optimizados para realizar una amplia gama de tareas, como la detección y reconocimiento de rostros, la identificación de objetos, la clasificación de acciones en videos, el seguimiento de movimientos y la reconstrucción de estructuras 3D [94].

Desarrollado por Google, MediaPipe es un marco multiplataforma para la construcción de aplicaciones multimedia. MediaPipe proporciona soluciones de vanguardia para la detección y seguimiento de manos, detección de rostros, entre otras. Es particularmente útil para aplicaciones que requieren el seguimiento en tiempo real y la interacción basada en gestos [54].

Algo que estas dos herramientas tienen en común es el lenguaje de programación que se utiliza, ya que ambas son librerías de Python. Python es un lenguaje de programación ampliamente utilizado en visión por computadora, especialmente cuando se combina con bibliotecas como OpenCV y MediaPipe. Estas herramientas permiten a los desarrolladores implementar aplicaciones avanzadas de visión por computadora de manera eficiente y con buenos resultados. La principal ventaja de utilizar estas librerías es que no se necesita entrenar los modelos de visión por computadora desde cero, ya que en muchos casos se puede utilizar uno de los modelos que forman parte de estas librerías [10].

4.5. Aplicaciones de visión por computadora

4.5.1. Medicina

La visión por computadora tiene una amplia gama de aplicaciones en diversos campos. En la medicina, se utiliza para el diagnóstico asistido por computadora, ayudando a los médicos a detectar enfermedades a partir de imágenes médicas como radiografías y resonancias magnéticas. Adicionalmente, puede ayudar a que los diagnósticos sean más certeros, ya que la tecnología puede servir como una segunda opinión del diagnóstico [51].

4.5.2. Industria automotriz

En el sector automotriz, la visión por computadora es fundamental para el desarrollo de vehículos autónomos y sistemas avanzados de asistencia al conductor. Estos sistemas dependen de la capacidad de los vehículos para identificar y responder a señales de tráfico, peatones y otros obstáculos en la carretera.

4.5.3. Seguridad y vigilancia

En el ámbito de la seguridad y vigilancia, la visión por computadora se utiliza para la detección de intrusos, el reconocimiento facial y la identificación de comportamientos sospechosos. Estas tecnologías son fundamentales para la prevención de crímenes y la protección de la seguridad pública.

4.5.4. Manufactura

En el ámbito industrial, la visión por computadora se emplea para el control de calidad y la inspección automatizada de productos. Esto puede simplificar de manera drástica los procesos de manufactura, y reduce dramáticamente el tiempo requerido en procesos de control de calidad de los productos.

4.6. Redes neuronales

4.6.1. Fundamentos

Las redes neuronales son un componente esencial de la inteligencia artificial. El funcionamiento de estas se inspira en el funcionamiento del cerebro humano, con una gran cantidad de neuronas. Estas redes están compuestas por capas de nodos, que son como neuronas artificiales, que procesan la información a través de conexiones ponderadas [52].

4.6.2. Tipos de redes neuronales

Existen varios tipos de redes neuronales, cada una adecuada para diferentes tareas. Las redes neuronales feedforward (FNN) son las más simples, donde la información se propaga en una sola dirección, de la entrada a la salida. Las redes neuronales convolucionales (CNN) son especialmente efectivas para el procesamiento de imágenes debido a su capacidad para reconocer patrones espaciales

jerárquicos. Por último, las redes neuronales recurrentes (RNN) son adecuadas para el procesamiento de datos secuenciales, como el reconocimiento de voz o la traducción automática [64].

4.6.3. Aplicaciones en visión por computadora

Las aplicaciones de redes neuronales en visión por computadora incluyen la clasificación de imágenes, donde las CNN pueden identificar y categorizar objetos dentro de una imagen, y la detección de objetos, donde se localizan y etiquetan múltiples objetos dentro de una escena. Estas tecnologías son fundamentales para el desarrollo de sistemas avanzados de reconocimiento de señas, como el proyecto "Señas Chapinas: Traductor de LENSEGUA", que utiliza redes neuronales para interpretar el lenguaje de señas guatemalteco en tiempo real.

4.7. Evaluación de modelos de visión por computadora

4.7.1. Matriz de confusión

La matriz de confusión es una herramienta que permite visualizar el rendimiento de un modelo de clasificación. Se organiza en una tabla que muestra las predicciones del modelo en comparación con las verdaderas etiquetas de los datos. Esta matriz se compone de cuatro componentes principales:

- Verdaderos Positivos (TP): Casos correctamente clasificados como positivos.
- Falsos Positivos (FP): Casos incorrectamente clasificados como positivos.
- Verdaderos Negativos (TN): Casos correctamente clasificados como negativos.
- Falsos Negativos (FN): Casos incorrectamente clasificados como negativos.

La matriz de confusión es una herramienta útil para evaluar el rendimiento de un modelo de clasificación, ya que proporciona información detallada sobre los errores del modelo [65]. Adicionalmente, la matriz de confusión se puede utilizar para calcular otras métricas de rendimiento del modelo, como la sensibilidad y la puntuación F1.

4.7.2. Sensibilidad

La sensibilidad, también conocida como *recall*, es una métrica que mide la proporción de casos positivos que fueron correctamente identificados por el modelo. Un alto valor de sensibilidad indica que el modelo tiene una buena capacidad para detectar la clase positiva, lo cual es especialmente importante en aplicaciones donde se desea minimizar los falsos negativos [69]. Se calcula utilizando la siguiente fórmula:

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \quad (4.1)$$

Donde TP son los verdaderos positivos y FN son los falsos negativos.

4.7.3. Puntuación F1

La puntuación F1, más comúnmente conocida como F1-score, es una métrica que combina la precisión y la sensibilidad en un solo valor. Se utiliza para evaluar el equilibrio entre la capacidad del modelo para identificar casos positivos (sensibilidad) y la exactitud de esas identificaciones (precisión). Se calcula de la siguiente manera:

$$\text{F1-score} = 2 \times \frac{\text{Precisión} \times \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \quad (4.2)$$

Donde la precisión se calcula como:

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (4.3)$$

Donde TP son los verdaderos positivos y FP son los falsos positivos.

La puntuación F1 es especialmente útil en situaciones donde hay un desbalance en las clases, ya que proporciona una medida más completa del rendimiento del modelo que la precisión o la sensibilidad por separado [73]. Esta métrica es ampliamente utilizada en la evaluación de modelos de clasificación.

4.8. Natural Language Processing (NLP)

El *natural language processing* (NLP), o procesamiento de lenguaje natural, es un campo de la inteligencia artificial que se enfoca en la comprensión y manipulación del lenguaje humano. Este campo no se limita a un solo modelo, sino que abarca una variedad de técnicas y algoritmos diseñados para interpretar y generar texto de manera efectiva [13]. Estos modelos se pueden utilizar en diversos contextos, desde análisis de sentimientos hasta traducciones automáticas, lo que demuestra su versatilidad y aplicación práctica en diferentes áreas de la tecnología y la comunicación [34].

4.8.1. Técnicas y fundamentos

En NLP, el preprocesamiento del texto es una etapa en la cual se preparan y limpian los datos antes de aplicar técnicas de análisis más avanzadas. Este proceso incluye varias etapas, como la *tokenización*. Esto consiste en dividir el texto en unidades más pequeñas, como palabras, con el fin de facilitar su posterior análisis. Además, el preprocesamiento también suele involucrar el *stemming*, que consiste en reducir las palabras a su forma base o raíz. Esto ayuda a normalizar el texto y a eliminar las variaciones morfológicas. Por último, se suele llevar a cabo la eliminación de *stopwords*, que son palabras comunes, pero que no aportan un significado contextual importante al texto, como 'el', 'la', 'de' y 'en'. En algunos casos, cabe destacar que puede ser necesario realizar un preprocesamiento más exhaustivo. Por ejemplo, dependiendo de las necesidades, se puede considerar el manejo de sinónimos para reducir el vocabulario de un texto [34].

Posterior al preprocesamiento, se puede aplicar *text feature extraction*. Este término engloba diferentes métodos, como *bag of words* y *N-grams*. Sin embargo, todos tienen como objetivo crear estructuras de datos que permiten representar de manera adecuada la información contenida en el texto. Por ejemplo, *bag of words* genera un listado de palabras que constituyen el vocabulario de un texto, acompañado de la frecuencia con la que cada palabra aparece. Esta información puede ser útil para identificar la importancia relativa de las palabras en un documento. *N-grams*, por otro lado, es

una técnica en la cual se generan secuencias de palabras de longitud N . Estas secuencias se pueden utilizar para identificar las relaciones y estructuras entre las palabras en un texto [34].

Una vez realizado el preprocesamiento del texto y creado las estructuras de datos adecuadas, se puede introducir dicha información en un modelo (o arquitectura) para llevar a cabo una variedad de tareas. Estas pueden incluir el análisis de sentimientos y la generación de texto. El análisis de sentimientos busca comprender las emociones expresadas en el texto, determinando si son positivas, negativas o neutras. Por otro lado, la generación de texto implica crear contenido de manera automática, ya sea completando frases, resumiendo artículos o incluso generando respuestas a preguntas del usuario [37].

4.8.2. Modelos principales

Existe una amplia variedad de modelos diseñados para llevar a cabo las tareas de procesamiento de lenguaje natural mencionadas anteriormente. La elección entre estos depende de varios factores, como la complejidad de la tarea, la disponibilidad y calidad de los datos, y los recursos computacionales disponibles.

Aprendizaje automático

Entre uno de los modelos más simples está el clasificador *Naïve Bayes*. Este es un modelo probabilístico que –como indica su nombre– emplea el teorema de Bayes. En otras palabras, funciona a través de determinar la probabilidad de que un cierto evento ocurra dado que otro evento ya ha ocurrido (asumiendo la independencia condicional). Para su funcionamiento correcto, este clasificador suele recibir el vocabulario y la frecuencia con la que aparecen las respectivas palabras. A partir de esta información, por ejemplo, se puede determinar la probabilidad de que un texto pertenezca a una clase específica, como 'positivo' o 'negativo', dadas las palabras que contiene. En la detección de spam, se puede calcular la probabilidad de que un correo electrónico sea spam o no, basándose en la frecuencia de ciertas palabras o características en el mensaje [112].

La regresión logística, similarmente, es otro modelo utilizado en el aprendizaje automático para problemas de clasificación. A pesar de su nombre, la regresión logística se utiliza principalmente para problemas de clasificación binaria, donde el objetivo es predecir la pertenencia a una de dos categorías distintas. A diferencia del clasificador *Naïve Bayes*, que se basa en la probabilidad condicional, la regresión logística utiliza una función logística para modelar la relación entre las variables de entrada y la probabilidad de pertenencia a una clase específica. Esto permite su aplicación en una variedad de tareas de NLP, donde se busca clasificar el texto en categorías específicas [36].

Aprendizaje profundo

Por otro lado, los modelos de aprendizaje automático se caracterizan por la utilización de redes neuronales. Una red neuronal es un modelo, compuesto por nodos interconectados (o neuronas), capaz de aprender patrones complejos en datos. Estos modelos, debido a su arquitectura, son capaces de realizar una mayor cantidad de tareas relacionadas con NLP, como generación de texto, traducción automática, resumen de documentos, respuesta a preguntas, entre otras [59].

Es importante destacar que para entrenar y utilizar eficazmente estos modelos es necesario realizar una etapa de transformación de los datos mediante el proceso de *embedding*. El *embedding* es un proceso que convierte palabras o frases en vectores numéricos, permitiendo a los modelos de aprendizaje automático comprender su significado y relación. Además, se utiliza el *padding* para igualar la longitud de las secuencias de entrada, lo que facilita su procesamiento en los modelos [71].

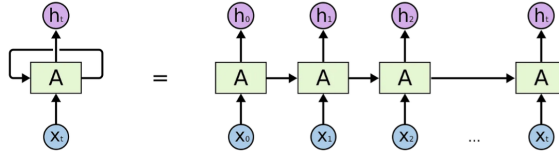


Figura 1: Arquitectura de red neuronal recurrente.

Las *recurrent neural networks* (RNN), o redes neuronales recurrentes, son comúnmente relacionadas con tareas de procesamiento de lenguaje natural. Estas redes trabajan con datos secuenciales, como series de tiempo o texto, debido a su capacidad para capturar dependencias entre los datos. Más específicamente, como se puede observar en la [1](#), las capas de estas redes toman como entrada el dato actual a procesar así como la salida de la capa recurrente en el paso anterior. Esto les permite mantener un registro que se actualiza constantemente, y que –como resultado– les permite capturar información contextual a lo largo de una secuencia [\[117\]](#). Por tal motivo, estas redes neuronales pueden ser sumamente útiles para sistemas de autocompletado de texto, por ejemplo [\[109\]](#).

Una desventaja de los RNNs es que debido a su estructura recurrente pueden tener dificultades para capturar dependencias a largo plazo. Esto se debe a que la información se propaga a través de las diferentes capas, y en cada una se realizan transformaciones que pueden ocasionar que la información relevante se diluya o se pierda. Sin embargo, cabe destacar que existen arquitecturas RNN con ciertas modificaciones capaces de mitigar –hasta cierto punto– esta problemática, como las redes *long-short term memory* (LSTM) [\[109\]](#).

En general, las redes neuronales tradicionales tienen una limitación relacionada con la cantidad de información que pueden procesar y/o recordar. Sin embargo, existen otros tipos de modelos, como los *large language models* (LLMs), o modelos de lenguaje grande, que tienen la capacidad de superar estas limitaciones. Estos modelos específicos tienen la característica de poder ser entrenados con grandes cantidades de datos (millones). Esto, obviamente, les permite poder capturar patrones más complejos en el lenguaje, así como también generar texto con mayor coherencia [\[11\]](#).

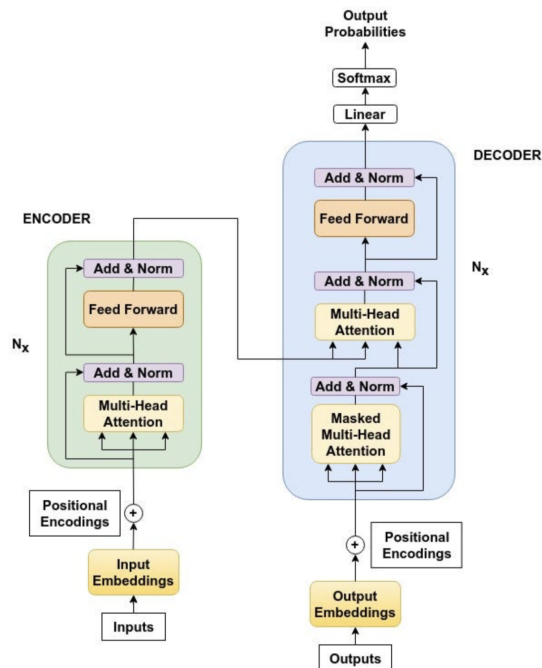


Figura 2: Arquitectura de transformer

Una de las arquitecturas más utilizadas para los LLMs es la del transformador. Un transformador es una red neuronal compuesta, comúnmente, por un *encoder* (codificador) y/o un *decoder* (decodificador). El *encoder* se encarga de procesar los *tokens* de entrada, determinando cuál es la relevancia relativa de cada uno en su contexto. A través de esto, dicho componente es capaz de crear representaciones contextuales de cada *token*. Por otro lado, el *decoder* utiliza el resultado del *encoder* para generar secuencias de salida. Sin embargo, en modelos que emplean únicamente *decoders*, estos asumen la responsabilidad de procesar la entrada y generar la salida de forma simultánea [127].

Cabe destacar que el *encoder* y *decoder* están estructurados de una forma similar. Por ejemplo, como se puede observar en la [2], ambos tienen capas llamadas *multi-head attention*, y *feed forward*. Las capas de *multi-head attention* le permiten al modelo poder enfocarse en diferentes partes de una secuencia simultáneamente, para así determinar cuál de todas es relevante. Por otro lado, las capas *feedforward* representan capas ocultas que procesan los datos y permiten al modelo aprender patrones complejos en las secuencias de entrada [46].

4.9. Arquitecturas basadas en transformadores

4.9.1. Generative Pre-trained Transformer (GPT)

Existen diversos modelos que utilizan la arquitectura de transformadores. Sin embargo, uno de los más populares es el *generative pretrained transformer* (GPT), o transformador generativo preentrenado. Este modelo fue presentado originalmente en el 2018 por OpenAI, en su artículo *Improving Language Understanding by Generative Pre-Training*. En esta publicación, los autores describen detalladamente la arquitectura del GPT. Además, resaltan que al entrenar un modelo con extensas cantidades de texto, este adquiere conocimientos sobre el mundo real y desarrolla la capacidad de procesar dependencias a largo plazo. Esto resulta en que el GPT sea capaz de responder

preguntas de cualquier índole, así como clasificar texto [70].

En la actualidad, este modelo se utiliza principalmente en un *chatbot* desarrollado por OpenAI, conocido como ChatGPT. La versión gratuita de esta herramienta utiliza la tercera versión del modelo GPT para llevar a cabo interacciones conversacionales naturales con los usuarios. Este modelo fue preentrenado con más de 45 *terabytes* (o 45,000 *gigabytes*) de texto plano [81], en comparación con la versión piloto, que fue preentrenada únicamente con 40 *gigabytes* [121].

Arquitectura

Un GPT, a diferencia de otros transformadores, se distingue por su uso exclusivo de decodificadores en su arquitectura. De hecho, emplea múltiples decodificadores apilados, los cuales dependen de la salida del decodificador anterior para generar texto coherente y relevante [57]. Esta característica le permite al modelo capturar relaciones contextuales a diferentes niveles de abstracción, lo que, a su vez, le posibilita analizar y replicar el lenguaje natural [12].

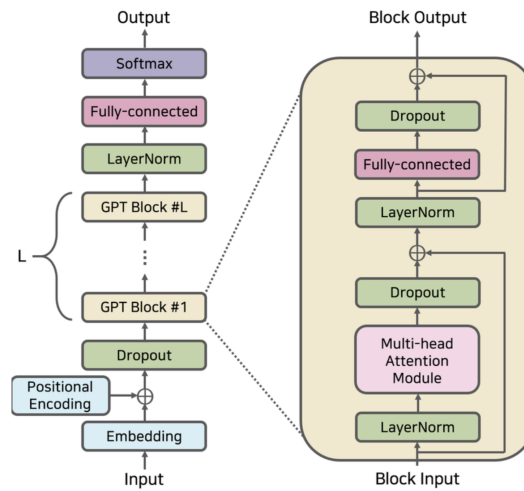


Figura 3: Arquitectura de Generative Pre-trained Transformer (GPT).

Como se puede observar en la Figura 3, la entrada de estos modelos primero pasa por una capa de *embedding*. Como se mencionó en secciones anteriores, en este proceso se convierten las palabras o frases en vectores numéricos. Cabe destacar que la entrada está limitada a 2048 palabras, por lo cual también se utiliza *padding* para ajustar las secuencias más cortas y asegurar una longitud uniforme antes de ingresarlas al modelo. Al concluir esa etapa, se continúa a la fase de *positional encoding*, en la cual se asignan vectores de posición a cada vector numérico con el propósito de poder identificar su ubicación en la secuencia [123].

Después de algunas capas variables, se procede a la utilización de los decodificadores apilados. Cabe destacar que todos están estructurados de una forma similar, conteniendo capas de normalización, *dropout*, *multi-head attention*, y *fully connected* (o *feed forward*). Posteriormente, el resultado del último *decoder* se normaliza y se pasa a una capa oculta, seguido por una función *softmax*. Esta última genera una distribución de probabilidad, la cual ayuda a identificar cuál es la próxima palabra más probable en la secuencia, según el contexto proporcionado por el modelo [123].

4.9.2. Large Language Model Meta AI (LLaMA)

Otro modelo reconocido que se basa en la arquitectura de *transformadores* es el *Large Language Model Meta AI* (LLaMA), que fue desarrollado por Meta y lanzado a principios de 2023. Lo que diferencia a LLaMA de otros modelos es su diseño *open source*, lo que permite que cualquier persona acceda a su código base y realice modificaciones o adaptaciones según sus necesidades. Además, LLaMA está optimizado para ser más eficiente en cuanto a recursos computacionales, lo que facilita su ejecución y entrenamiento en sistemas con menos capacidad. Esta eficiencia, combinada con su naturaleza de código abierto, lo convierte en una herramienta poderosa y accesible, especialmente para investigadores y desarrolladores con recursos limitados [125].

La versión más reciente de LLaMA, LLaMA 3.0, fue lanzada con significativas mejoras en su capacidad de procesamiento y versatilidad. Esta versión fue entrenada utilizando un total de 15 billones de *tokens* de datos en múltiples idiomas, lo que le permite tener un rendimiento mejorado y adaptarse a un rango más amplio de tareas y contextos. Esta versión se ofrece en dos variantes principales: una versión con 8.000 millones de parámetros (8B) y otra más avanzada con 70.000 millones de parámetros (70B). Estas variantes permiten a los usuarios elegir el modelo que mejor se ajuste a sus necesidades, equilibrando precisión, rendimiento y eficiencia computacional [125].

Arquitectura

LLaMA, al igual que GPT, utiliza exclusivamente decodificadores en su arquitectura basada en transformadores. Su diseño incluye múltiples decodificadores apilados, los cuales trabajan en conjunto para procesar y generar texto coherente y contextual [125]. Esta estructura modular le permite capturar relaciones complejas entre los *tokens* en diferentes niveles de abstracción, lo que mejora su capacidad para analizar y replicar el lenguaje natural de manera eficiente.

Como se muestra en la Figura 4, la entrada del modelo pasa inicialmente por una capa de *embedding*, donde las palabras o frases se transforman en vectores numéricos para su procesamiento. Este paso incluye también la codificación posicional, la cual asigna vectores específicos que indican la ubicación de cada palabra en la secuencia, asegurando que el modelo pueda interpretar el orden y la estructura del texto [125].

Una vez que los vectores han sido generados y enriquecidos con la información posicional, estos se introducen en las capas de decodificadores apilados. Cada decodificador contiene submódulos como normalización (*layer normalization*), *dropout*, *multi-head attention*, y redes *feed-forward*. Este diseño permite al modelo refinar progresivamente la representación de los datos en cada capa, capturando patrones contextuales complejos [125].

Finalmente, el resultado de la última capa de decodificadores pasa por una capa oculta, seguida de una función *softmax*, que genera una distribución de probabilidad para predecir la próxima palabra más probable en función del contexto [125].

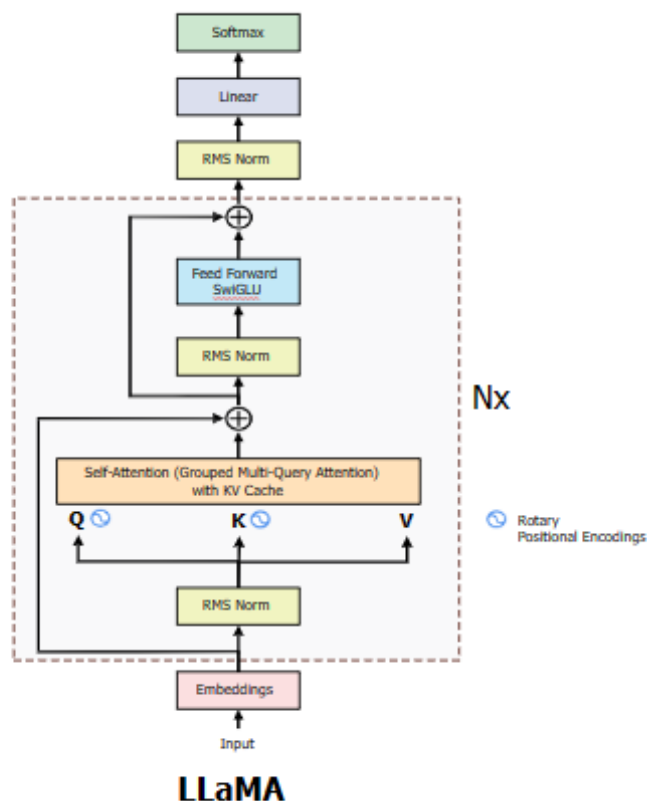


Figura 4: Arquitectura de Large Language Model Meta AI (LLaMA).

4.9.3. Métodos de entrenamiento y adaptación

Pre-entrenamiento

Una vez definidas las arquitecturas de los modelos, es posible enseñarles a comprender, generar y procesar texto mediante técnicas avanzadas de aprendizaje. Este proceso inicia con el pre-entrenamiento, una etapa en la que los modelos aprenden patrones y estructuras del lenguaje sin enfocarse en una tarea específica. Para ello, se utilizan diversos datos, como artículos de prensa, libros, transcripciones de videos y audios [121].

Durante el pre-entrenamiento, los modelos reciben frases incompletas como ejemplos y tienen la tarea de predecir la siguiente palabra o token en función del contexto. Si la predicción es incorrecta, el modelo ajusta internamente sus parámetros para priorizar las respuestas adecuadas en escenarios similares futuros. Al repetir este proceso de manera iterativa, ambos modelos adquieren la capacidad de generar texto coherente y relevante [123].

Es importante destacar que este tipo de modelos pueden ser posteriormente entrenados a través de *fine-tuning* para adaptarse a tareas específicas, ampliando su utilidad y precisión en diferentes contextos.

Fine-tuning

Fine-tuning, o ajuste fino, es un proceso a través del cual se adapta un modelo pre-entrenado para realizar una tarea nueva. Para esto, se le proporciona a un modelo ejemplos de entrada y salida relacionados a la tarea que se desea enseñar. Por ejemplo, para un traductor de idiomas, estas serían frases en el idioma de origen y la traducción correspondiente en el idioma destino. A partir de estos, el modelo ajusta los pesos de sus capas neuronales, los cuales son parámetros internos que determinan cómo se combinan las características de los datos de entrada para producir la salida. Este ajuste tiene como objetivo minimizar la diferencia entre las predicciones del modelo y los resultados reales, permitiendo que el modelo aprenda a resolver la nueva tarea de manera efectiva [93].

Es importante mencionar que la cantidad de datos para el proceso de ajuste fino depende de la complejidad de la tarea. Para tareas simples puede ser suficiente decenas o cientos de ejemplos, mientras que para tareas más complejas se pueden necesitar miles o incluso millones de ejemplos para lograr un ajuste preciso del modelo. En otras palabras, a mayor complejidad usualmente se requiere una cantidad proporcionalmente mayor de datos [93].

Prompt engineering

En ciertos casos, los modelos pueden tener dificultades para aprender nuevas tareas a través del *fine-tuning*, lo que puede resultar en respuestas incoherentes o inesperadas. Para abordar este tipo de problemas, se utiliza el *prompt engineering*, una técnica que busca mejorar la formulación de las instrucciones dadas al modelo, de manera que interprete correctamente la tarea y genere respuestas más precisas. Más específicamente, esta técnica consiste en estructurar cuidadosamente las entradas, proporcionando ejemplos claros, instrucciones detalladas o indicaciones específicas sobre el formato y el contenido esperado. Esto permite guiar al modelo para que genere respuestas más consistentes y alineadas con los objetivos deseados [129].

Una ventaja importante del *prompt engineering* es que, en algunos casos, permite que el modelo realice nuevas tareas sin necesidad de un *fine-tuning*. Sin embargo, para tareas más complejas, esta técnica por sí sola puede no ser suficiente. Por lo tanto, combinar *prompt engineering* con *fine-tuning* suele ser más efectivo. Mientras que el *fine-tuning* adapta el modelo a tareas específicas, el *prompt engineering* mejora la formulación de las instrucciones para optimizar el rendimiento. Utilizar ambas técnicas de manera complementaria maximiza las capacidades del modelo y mejora los resultados en tareas más desafiantes [129].

4.9.4. Métricas de rendimiento y evaluación

Para evaluar el rendimiento de un proceso de *fine-tuning* o *prompt engineering*, es fundamental utilizar métricas que permitan medir la similitud entre los resultados generados por un modelo y las respuestas esperadas. Estas métricas ayudan a identificar qué tan bien un modelo logra replicar o aproximarse al comportamiento deseado en tareas específicas. Algunas de las métricas más utilizadas son la distancia de Levenshtein y BLEU.

Distancia de Levenshtein

La distancia de Levenshtein mide el número mínimo de operaciones necesarias para transformar una cadena de texto en otra. Estas operaciones incluyen inserciones, eliminaciones y sustituciones de caracteres. Es una métrica útil para evaluar la similitud entre dos textos, ya que un menor valor indica una mayor cercanía entre ellos [114].

Bilingual Evaluation Understudy (BLEU)

La métrica BLEU (*Bilingual Evaluation Understudy*) evalúa la calidad de un texto generado al compararlo con un texto de referencia, basándose en la coincidencia de *n-gramas*. Estas coincidencias pueden ser unigramas, bigramas, trigramas, entre otros. BLEU también considera una penalización por brevedad, en caso de que los textos generados sean significativamente más cortos que el texto de referencia.

El valor de BLEU se expresa en una escala de 0 a 1, donde los valores más altos indican una mayor similitud entre el texto generado y el texto de referencia, reflejando una mejor calidad de generación [96].

4.10. Explainable Artificial Intelligence (XAI)

La Inteligencia Artificial Explicable (XAI), es un conjunto de técnicas que buscan hacer comprensibles las decisiones y predicciones de los modelos de inteligencia artificial. Esto resulta especialmente útil en modelos de *deep learning* (como *transformadores*) ya que estos sistemas, debido a su compleja arquitectura, funcionan como “cajas negras”. En otras palabras, el proceso mediante el cual se generan las predicciones no es fácilmente interpretable por los usuarios, lo que dificulta identificar cómo ciertas entradas llevan a un resultado específico [41].

Para abordar este desafío de interpretabilidad, las técnicas de XAI se han desarrollado siguiendo dos enfoques principales: *intrínseco* y *post-hoc*. El enfoque *intrínseco* integra la explicabilidad en el diseño del modelo desde su creación, incorporando mecanismos de transparencia durante el proceso de desarrollo. Estos modelos se construyen con arquitecturas inherentemente interpretables, como árboles de decisión o reglas lógicas. Por otro lado, el enfoque *post-hoc* ofrece interpretaciones tras el entrenamiento del modelo, sin alterar su estructura interna [41]. Entre las herramientas *post-hoc* más destacadas se encuentra LIME (*Local Interpretable Model-agnostic Explanations*), que permite visualizar la contribución relativa de cada variable de entrada en las predicciones de un modelo, ayudando así a entender qué características tienen mayor influencia en los resultados generados [39].

4.10.1. Local Interpretable Model-agnostic Explanations (LIME)

LIME es una técnica *post-hoc* introducida por Ribeiro et al. en el 2016. Su principal característica es que puede aplicarse a cualquier tipo de modelo, lo que la hace *model-agnostic*. Además, proporciona explicaciones locales, es decir, enfocadas en cada predicción individual [107].

Para generar estas explicaciones locales, LIME utiliza un proceso basado en la perturbación de los datos de entrada. La idea principal es crear múltiples versiones modificadas de los datos originales que se desean procesar para generar una predicción. Este proceso implica alterar sistemáticamente las características, ya sea variando valores numéricos o eliminando elementos. Por ejemplo, si se trabaja con características numéricas, las perturbaciones pueden consistir en variar los valores dentro de ciertos rangos o eliminarlos temporalmente. El objetivo es crear un conjunto de variaciones que permitan observar cómo el modelo responde a diferentes combinaciones de las características de entrada [128].

Una vez generadas estas perturbaciones, cada entrada modificada se procesa a través del modelo original para obtener nuevas predicciones. Al comparar cómo cambia la salida del modelo para cada perturbación respecto a la predicción original, es posible determinar la importancia relativa de cada característica en la decisión final [128]. Este análisis permite identificar qué elementos de los datos de entrada tienen mayor influencia en las predicciones del modelo, revelando patrones y dependencias

que no son evidentes a simple vista. Los resultados proporcionan una comprensión clara de cómo diferentes variables afectan el resultado final, lo que resulta fundamental para validar si el modelo está basando sus decisiones en características relevantes [107].

4.11. Sistema operativo: Linux server

4.11.1. ¿Qué es Linux?

Linux es un sistema operativo de código abierto basado en Unix que se ha establecido como una opción poderosa y versátil para una variedad de plataformas, desde servidores y supercomputadoras hasta dispositivos móviles y electrodomésticos. Fue creado inicialmente por Linus Torvalds en 1991. El núcleo de Linux, conocido como el kernel, gestiona las interacciones del software con el hardware y es vital para la regulación de los recursos del sistema. A lo largo de los años, Linux ha ganado una popularidad masiva, en parte debido a su naturaleza de código abierto, que permite a los usuarios modificar y mejorar su software según necesiten. Además, su alta configurabilidad, estabilidad y robustez en seguridad lo hacen preferido en entornos empresariales y académicos. La comunidad de Linux es una de sus mayores fortalezas, contribuyendo constantemente con una amplia gama de distribuciones adaptadas a diferentes usos y preferencias [90].

4.11.2. Historia de Linux

Linux es un sistema operativo de código abierto creado en 1991 por Linus Torvalds, un estudiante de la Universidad de Helsinki. Inspirado en Unix, Torvalds deseaba ofrecer una alternativa gratuita y accesible, por lo que liberó Linux bajo la Licencia Pública General de GNU para permitir su uso, modificación y redistribución libre. Inicialmente adoptado por entusiastas tecnológicos y desarrolladores, Linux ha evolucionado hasta ser ampliamente utilizado en servidores, dispositivos móviles y supercomputadoras. Destacado por su estabilidad, seguridad y flexibilidad, es ahora uno de los sistemas operativos más predominantes a nivel mundial, con múltiples distribuciones como Ubuntu, Fedora y Arch que facilitan su uso en una gran variedad de entornos [49].

4.11.3. Linux server

Un servidor Linux se compone fundamentalmente de Linux, una familia de sistemas operativos de software libre y código abierto que se desarrollan alrededor del kernel de Linux. Originalmente creado como una versión alternativa y libre del sistema operativo MINIX, los servidores Linux se han popularizado gracias a su estabilidad, seguridad y flexibilidad. Estas características no solo los diferencian de sus contrapartes propietarias, sino que también mantienen bajos los costos de configuración y mantenimiento. Además, ofrecen una flexibilidad aumentada en la configuración, operación y mantenimiento de un servidor. Un sistema operativo de servidor Linux proporciona la interfaz central para la gestión de usuarios e implementa diversos servicios de seguridad y administración, todos cruciales para operar en una arquitectura cliente-servidor [105].

4.12. Virtualización

La virtualización, en el contexto de la computación, implica la creación de versiones virtuales de recursos que originalmente eran físicos, como sistemas operativos y hardware de red. Este proceso permite a un solo servidor físico alojar múltiples máquinas virtuales (VMs), cada una con su propio

sistema operativo y aplicaciones, operando de manera independiente pero compartiendo los recursos subyacentes del hardware físico [124].

El propósito principal de la virtualización fue maximizar la eficiencia del uso de recursos al permitir que múltiples sistemas operativos y aplicaciones se ejecutaran en un solo servidor físico. Esto no solo reduce los costos de hardware, sino que también aumenta la flexibilidad y la escalabilidad de los sistemas de TI. Las VMs pueden migrar entre servidores con mínimo tiempo de inactividad, lo que facilita el mantenimiento y la gestión del sistema [91].

4.13. Multipass

Multipass es una herramienta ligera de virtualización diseñada para facilitar la creación y gestión de máquinas virtuales, enfocada especialmente en entornos de desarrollo y pruebas rápidas. Al integrarse de manera nativa con Ubuntu, Multipass permite desplegar y administrar múltiples instancias de Linux de manera eficiente, actuando como una mini-nube local en tu propio equipo. Multipass ha sido desarrollado con la simplicidad en mente, proporcionando una experiencia optimizada para usuarios que necesitan acceder rápidamente a entornos virtuales consistentes sin la complejidad de configuraciones avanzadas [86].

4.14. VPN

4.14.1. ¿Qué es una VPN?

Una VPN (red privada virtual) establece una conexión segura y privada entre dispositivos a través de Internet. Este tipo de red es fundamental para la transmisión de datos de manera segura y anónima a través de infraestructuras públicas. El proceso implica enmascarar las direcciones IP de los usuarios y cifrar la información transmitida, asegurando que solo las partes autorizadas puedan acceder y leer estos datos [15].

4.14.2. ¿Para qué sirven las VPNs?

Las VPNs, o redes privadas virtuales, son esenciales para permitir conexiones remotas seguras a servidores y recursos de red. OpenVPN, un destacado software de VPN de código abierto, facilitó el establecimiento de redes locales virtuales entre computadoras dispersas geográficamente, permitiendo a los usuarios acceder y operar servidores y datos como si estuvieran físicamente presentes en la misma red local. Esto es crucial para empresas y organizaciones con equipos distribuidos, asegurando que la comunicación y el acceso a recursos críticos sean seguros y eficientes [115].

4.15. Arquitectura de base de datos

4.15.1. Bases de datos relacionales

Una base de datos relacional es una estructura que organiza la información en filas y columnas dentro de tablas, facilitando el almacenamiento, la búsqueda y la gestión de datos de manera eficiente. Las tablas se enlazan entre sí mediante claves primarias y claves foráneas, creando relaciones que reflejan cómo se conectan los datos en diferentes tablas. Este sistema permite a los usuarios emplear

consultas SQL para cruzar información de diversas tablas, proporcionando una herramienta poderosa para analizar y resumir el rendimiento empresarial [67].

4.15.2. PostgreSQL

PostgreSQL es una mejora del sistema de gestión Postgres, un prototipo avanzado de DBMS de próxima generación. Mientras conserva el modelo de datos potente y los tipos de datos ricos de Postgres, PostgreSQL ha reemplazado el lenguaje de consulta POSTQUEL por un subconjunto extendido de SQL, convirtiéndolo en una opción gratuita y de código abierto para el manejo de bases de datos. Desarrollado inicialmente en la Universidad de California en Berkeley, PostgreSQL ha evolucionado significativamente gracias a la contribución de un equipo de desarrolladores a nivel mundial [85].

4.15.3. Arquitectura de base de datos

La arquitectura de datos se refiere al diseño estructural y a la metodología empleada para organizar y gestionar los recursos de datos en una organización o sistema. Este marco establece las políticas, normas y procedimientos para la recopilación, almacenamiento, manejo y uso de los datos con el fin de asegurar su accesibilidad, consistencia, integridad y seguridad. Una arquitectura de datos eficaz apoya la estrategia de información de una empresa, facilitando la integración y la operación de sistemas de TI, mejorando el soporte para las decisiones basadas en datos y optimizando el rendimiento de las aplicaciones [66].

4.16. APIs

4.16.1. ¿Qué es una API?

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre sí. Funciona como un puente que permite la interacción entre diferentes piezas de software de manera estandarizada. Por ejemplo, una aplicación de pronóstico del tiempo puede recibir datos de un servicio meteorológico utilizando una API para comunicarse con el sistema de dicho servicio [14].

4.16.2. ¿Cómo usar una API?

Usar una API a nivel básico implica interactuar con un conjunto de definiciones y protocolos que permiten la comunicación entre diferentes componentes de software. Primero, es esencial entender los métodos estándar que la API ofrece, como obtener, crear, actualizar o eliminar recursos. Cada interacción sigue un patrón predefinido, facilitando así la integración y uso de la API en proyectos [61].

4.16.3. Implementar APIs a bases de datos

Al desarrollar una aplicación que necesite interactuar con una base de datos mediante una API, puedes realizar operaciones como crear, recuperar, actualizar o eliminar registros. Estas operaciones

se definen claramente con ciertos parámetros de entrada esperados y formatos de respuesta, lo que facilita la integración de sistemas complejos [62].

4.16.4. Implementar APIs para recepción y devolución de datos

El uso de una API para enviar y recibir información implica interacciones estructuradas mediante solicitudes y respuestas definidas. Un ejemplo común es usar un método HTTP POST para enviar datos a la API, que luego devuelve una respuesta indicando éxito o fallo [63].

4.16.5. ¿Qué es Crontab?

‘Crontab’ es una herramienta en sistemas Unix/Linux que permite la programación de tareas automatizadas que se ejecutan a intervalos específicos. Estas tareas, conocidas como cron jobs, son útiles para ejecutar scripts y comandos de forma recurrente sin intervención manual, lo cual es ideal para tareas de mantenimiento o procesamiento automatizado, como el reinicio de rachas en nuestra aplicación [60].

4.17. Bases de diseño y filosofía de DARPA

Entre los conceptos fundamentales aplicados en el diseño de la arquitectura del sistema, se adoptaron varios principios expuestos en el documento “The Design Philosophy of the DARPA Internet Protocols” [27]. A continuación, se destacan los siguientes conceptos clave:

4.17.1. Costo-efectividad y control

La arquitectura de Internet, según lo planteado por DARPA, enfatiza la necesidad de manejar los recursos de manera eficaz bajo control administrativo distribuido. Esto justifica la decisión de utilizar servidores propios en lugar de servicios en la nube para este proyecto, reduciendo dependencias externas y costos operativos. Al mantener un control directo sobre la infraestructura, se puede gestionar de manera más eficiente el uso de los recursos, optimizando la relación costo-beneficio sin sacrificar el rendimiento.

4.17.2. Flexibilidad y adaptabilidad

La filosofía de diseño también subraya la importancia de adaptarse a diferentes entornos y requisitos técnicos. En este proyecto, la elección de Ubuntu como sistema operativo refleja un compromiso con un sistema flexible y robusto, ya que permite una personalización extensa. Aunque inicialmente se consideró el uso de RAID para la gestión del almacenamiento, se decidió desinstalarlo y desconfigurarlo debido a las limitaciones específicas del entorno de este proyecto. Esto asegura que la infraestructura sea adaptable a las necesidades actuales, manteniendo un enfoque simplificado en la gestión de recursos.

4.17.3. Seguridad y privacidad

Siguiendo los principios de diseño de DARPA, la seguridad es una prioridad fundamental. La integridad y seguridad de la comunicación entre los sistemas es vital para el éxito del proyecto. En

este sentido, la implementación de una VPN segura mediante OpenVPN garantiza la protección de los datos transmitidos, asegurando que el acceso remoto y la transferencia de información se realicen de manera segura y confiable, reduciendo el riesgo de vulnerabilidades y ataques.

4.18. Pruebas de eficiencia y de extremo a extremo en el servidor mediante APIs

4.18.1. ¿Qué es una prueba de eficiencia?

Las pruebas de eficiencia miden cómo el servidor responde bajo condiciones de carga habitual, evaluando aspectos como el tiempo de respuesta, la utilización de recursos (CPU, memoria y disco), y la capacidad del sistema para gestionar solicitudes simultáneas sin afectar negativamente el rendimiento. Estas pruebas permiten identificar oportunidades de optimización y ajustar la infraestructura para garantizar un rendimiento óptimo en condiciones normales de operación [98].

4.18.2. ¿Qué es una prueba de extremo a extremo (E2E)?

Las pruebas de extremo a extremo (E2E) verifican el funcionamiento integral de la aplicación, evaluando cómo interactúan todos sus componentes en un flujo completo desde el punto de vista del usuario. Estas pruebas son esenciales para asegurarse de que cada parte de la aplicación, incluidas las APIs, funcione como se espera y que todos los puntos de interacción respondan adecuadamente dentro del sistema.

- **Objetivo:** Confirmar que todas las partes de la aplicación operan de manera integrada, verificando que los flujos de trabajo completos (como registro, autenticación, envío de datos y consulta de información) se ejecuten sin errores.
- **Resultado esperado:** Que la aplicación realice todas las operaciones y flujos clave sin problemas, asegurando que cada funcionalidad principal esté correctamente integrada y funcione en conjunto.

4.18.3. Impacto de la eficiencia y de las pruebas de extremo a extremo en el servidor

- **Eficiencia:** Un servidor eficiente maximiza el uso de los recursos disponibles, lo que permite reducir los costos operativos y mantener un rendimiento constante. Esto es esencial para asegurar tiempos de respuesta rápidos y garantizar que el sistema pueda manejar múltiples solicitudes concurrentes sin una degradación del servicio.
- **Pruebas E2E:** Las pruebas de extremo a extremo permiten validar el flujo completo de operaciones, asegurando que cada componente del sistema funcione correctamente y en sincronía. Estas pruebas ayudan a identificar cualquier posible problema de integración entre APIs y otros módulos de la aplicación, lo que es esencial para una experiencia de usuario fluida y un sistema robusto.

4.19. Herramientas para pruebas de eficiencia y de extremo a extremo

Para realizar las pruebas de eficiencia y extremo a extremo, se emplearon las siguientes herramientas:

- **Python:** Se utilizó para desarrollar scripts personalizados que permiten simular múltiples usuarios y escenarios de carga en la aplicación, evaluando el tiempo de respuesta, el rendimiento y el uso de recursos en situaciones de carga típica y bajo prueba de integración completa [103].
- **NGINX Amplify:** Herramienta de monitoreo utilizada para supervisar el uso de recursos en tiempo real, como CPU, memoria y tráfico de red. NGINX Amplify proporciona métricas detalladas y facilita el análisis de rendimiento durante las pruebas, identificando posibles áreas de optimización y problemas de integración [92].

4.19.1. Monitoreo del sistema

- **htop:** Proporciona una vista en tiempo real del uso de CPU, memoria y procesos en ejecución.
- **dstat:** Genera estadísticas combinadas de CPU, disco, memoria y red.
- **iostat:** Muestra el uso de disco por cada proceso, facilitando la identificación de procesos que consumen muchos recursos.
- **Prometheus:** Sistema de monitorización para métricas a gran escala.
- **Prometheus-Exporter:** Herramienta para monitorear las solicitudes entrantes al servidor.
- **Grafana:** Plataforma de visualización de métricas, utilizada para analizar los resultados de Prometheus [40].

4.19.2. Pruebas de carga y end to end

Las pruebas de carga se centran en medir la respuesta del sistema bajo condiciones de uso previstas, simulando múltiples usuarios y solicitudes simultáneas. Estas pruebas permiten evaluar el rendimiento, tiempos de respuesta y estabilidad del sistema cuando procesa diversas acciones en paralelo [78].

- **Objetivo:** Evaluar cómo responde el sistema bajo una carga esperada de usuarios o solicitudes simultáneas, enfocándose en el rendimiento y la estabilidad mientras se procesan múltiples acciones.
- **Resultado esperado:** Que el sistema mantenga tiempos de respuesta y niveles de rendimiento aceptables bajo condiciones de uso normales (es decir, el número máximo de usuarios esperados en condiciones típicas).
- **Prueba:** La prueba de carga implementada simula a varios usuarios ejecutando operaciones simultáneas, como autenticación, envío de videos y marcación de favoritos. Se registra el tiempo de respuesta y el estado de cada acción, evaluando así la capacidad de la aplicación para manejar un volumen esperado de usuarios sin degradar su rendimiento.

Las pruebas de Extremo a Extremo (E2E) se orientan a evaluar el flujo completo de la aplicación desde la perspectiva del usuario. A diferencia de las pruebas de carga, estas pruebas verifican que todos los componentes del sistema funcionen conjuntamente como se espera en un caso de uso completo.

- **Objetivo:** Confirmar que todas las partes de la aplicación funcionan de manera integrada y que los componentes interactúan correctamente en un flujo de uso completo.
- **Resultado esperado:** Que la aplicación ejecute todas las operaciones y flujos clave sin errores (por ejemplo, registro, autenticación, envío y consulta de datos), asegurando que las funcionalidades principales se integren de manera adecuada.
- **Prueba:** La prueba E2E implementada cubre un caso de uso completo, desde la creación de un usuario hasta la interacción con las principales funciones de la aplicación, como autenticación, carga de videos, traducción y manipulación de favoritos. Esta prueba garantiza que cada parte del sistema funcione en conjunto en el flujo real de un usuario.

4.20. Diseño de interfaz de usuario (UI)

4.20.1. Definición de UI

El diseño de la interfaz de usuario (UI por sus siglas en ingles, *User Interface*) se dedica a la creación de los elementos visuales e interactivos de un producto digital. En esencia, se trata de diseñar lo que los usuarios ven en sus pantallas [95].

La interfaz de usuario determina en gran medida las primeras impresiones que tienen estos acerca de un negocio o producto. Cuando los usuarios pueden navegar fácilmente por una interfaz y completar las tareas deseadas, no solo mejora su experiencia, sino que también beneficia al negocio en general. Cuanto más visualmente atractiva y acogedora sea una interfaz, más probable es que atraiga a los usuarios y los motive a explorar más [95].

4.20.2. Elementos de la interfaz de usuario

Los elementos de la UI son los componentes básicos de las aplicaciones y sitios web con los que los usuarios interactúan. Estos elementos son esenciales para el desarrollo de funcionalidades óptimas y una experiencia de navegación agradable. Los elementos UI se agrupan en tres categorías principales [95]:

- **Elementos de entrada:** Permiten a los usuarios ingresar información en el sistema y, en ocasiones, son parte del proceso de validación de entrada. Los elementos de entrada más comunes incluyen [95]:
 - Listas desplegables
 - Campos de texto o contraseña
 - Selectores de fecha
 - Casillas de verificación
 - Diálogos de confirmación
- **Elementos de salida:** Muestran los resultados basados en las entradas del usuario y las operaciones previas. Ejemplos de elementos de salida son [95]:

- Alertas
 - Mensajes de éxito
 - Mensajes de error
- **Elementos auxiliares:** Proporcionan información adicional al usuario y mejoran la navegación. Estos elementos se dividen en tres subcategorías [95]:
 - **Elementos de navegación:** Facilitan el desplazamiento a través del producto digital [95]. Algunos ejemplos son:
 - Menús de navegación
 - Listas de enlaces
 - **Elementos informativos:** Incluyen referencias que ayudan a utilizar o comprender el producto digital [95]. Ejemplos son:
 - Iconos
 - Notificaciones
 - **Elementos de grupos o contenedores:** Ayudan a organizar y mantener los componentes del producto juntos [95]. Ejemplos incluyen:
 - Barras laterales
 - Contenedores de contenido

4.20.3. Estándares de diseño

Tamaño y posición de botones

El tamaño de los objetos táctiles en una aplicación móvil es crucial, ya que afecta directamente la facilidad con la que los usuarios pueden interactuar con la interfaz. Si los botones no tienen un tamaño y espacio óptimos, los usuarios podrían no alcanzar su objetivo o presionar el botón equivocado [24].

Investigaciones sobre el tamaño y el espaciado de los botones han establecido estándares que funcionan para la mayoría de los usuarios. Los estudios encontraron que los usuarios tienen la menor precisión táctil en botones de menos de 42 píxeles, los cuales se usan generalmente para funciones de baja prioridad. La mayor precisión se encontró en botones de entre 42 y 72 píxeles. Por ello, se utilizan botones de aproximadamente 60 píxeles para funciones de prioridad media, mientras que los de más de 72 píxeles se reservan para funciones de alta prioridad, asegurando así la accesibilidad para todos los usuarios [16].

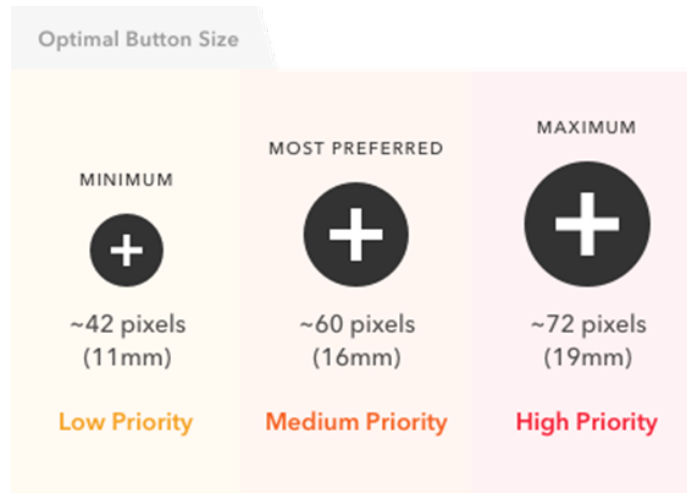


Figura 5: Tamaño óptimo de botones según su prioridad

Además del tamaño, existen ciertos márgenes y áreas táctiles recomendadas alrededor de los botones para garantizar que sean fácilmente accesibles y utilizables. Estos márgenes aseguran que haya suficiente espacio alrededor de los botones para evitar errores de pulsación. Dependiendo del tamaño del botón, se recomienda un margen de 12 a 24 píxeles para botones grandes, de 24 a 36 píxeles para botones medianos y de 36 a 48 píxeles para botones pequeños [16].

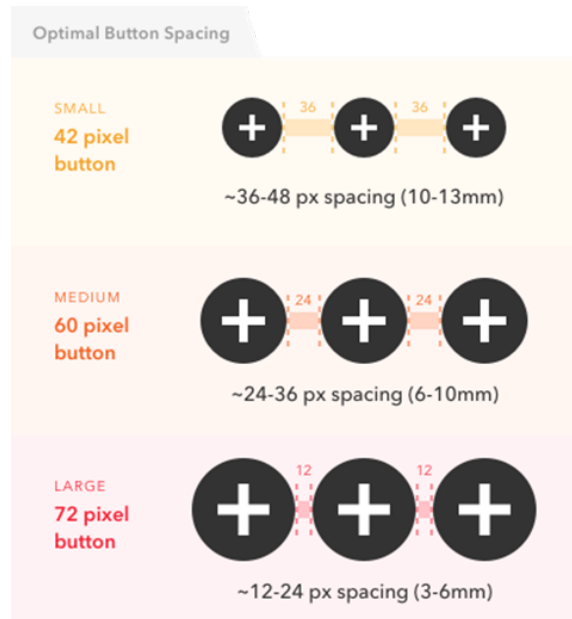


Figura 6: Espaciado óptimo de botones según su tamaño

Asimismo, la ubicación de los botones es fundamental para la usabilidad. Los usuarios están acostumbrados a interactuar con dispositivos digitales a lo largo del día y tienden a buscar ciertos elementos en ubicaciones predecibles. Por esta razón, es importante posicionar los botones en lugares donde los usuarios esperan encontrarlos, como en la parte inferior de la pantalla para acciones frecuentes o en la esquina superior derecha para opciones adicionales. Esta previsibilidad mejora la eficiencia y reduce la frustración del usuario [101].

Diseño responsivo

El diseño responsivo es crucial para aplicaciones móviles, ya que asegura que la interfaz se adapte a diferentes tamaños de pantalla y resoluciones [20].

Dado que las pantallas de los dispositivos móviles varían en densidad, con más píxeles por pulgada en pantallas de mayor resolución, es importante utilizar píxeles independientes de la densidad (dp) para definir las dimensiones de los elementos de la interfaz. Esto asegura que los elementos mantengan su tamaño y proporción adecuados en diferentes dispositivos, proporcionando uniformidad y coherencia visual [101].

Principios de diseño

Hay una serie de principios de diseño de interfaz para lograr que el producto digital satisfaga al cliente final [4].

- **Simplicidad:** Es crucial evitar elementos innecesarios que puedan causar confusión [4].
- **Consistencia:** Al emplear elementos comunes en la interfaz de usuario, los usuarios se sienten más cómodos y familiarizados con el diseño, lo que facilita el uso del producto digital [4].
- **Jerarquía y manejo de espacios:** La disposición y estructuración de los elementos según su importancia ayuda a dirigir la atención del usuario a la información más relevante. Agrupar elementos relacionados crea una relación visual y mejora la comprensión del contenido [4].
- **Color y tipografía:** El color es esencial en el diseño de interfaces. Ajustar la saturación, luz y contraste del color en el diseño destaca elementos importantes y facilita la navegación. Asimismo, se utilizan diferentes tamaños, fuentes y disposiciones del texto para mejorar la legibilidad y jerarquía visual [4].

Paleta de colores

Los colores desempeñan un papel fundamental en el diseño de interfaces de usuario, ya que no solo aportan personalidad y estilo al producto digital, sino que también influyen en la percepción y emociones del usuario [2].

Una paleta de colores bien diseñada ayuda a los usuarios a comprender rápidamente la importancia y la relación entre diferentes elementos en la interfaz, como las llamadas a la acción o la información crítica. Al asignar colores específicos a distintas categorías o secciones, los usuarios pueden identificar fácilmente su ubicación en la interfaz y cómo navegar dentro de ella [2].

La paleta de colores se elige para evocar emociones específicas que se alineen con los objetivos del diseño. Por ejemplo, los colores cálidos como el rojo y el naranja pueden provocar sensaciones de urgencia o excitación, lo que los hace ideales para botones de compra o alertas. En contraste, los colores fríos como el azul y el verde pueden inducir sensaciones de tranquilidad y confianza, siendo perfectos para páginas de inicio o secciones de información [2].

Es esencial que la paleta de colores tenga en cuenta la legibilidad y la visibilidad, cumpliendo con los estándares de contraste y asegurando que la información sea clara para todos los usuarios [2].

Selección de tipografía

Al igual que la paleta de colores, la elección de una tipografía tiene un impacto significativo en el diseño de un producto digital. La tipografía moldea la manera en que se percibe y comprende la información visual. Elementos tipográficos como el tipo de letra, su tamaño y color tienen la capacidad de transmitir diversos significados y causar distintas respuestas emocionales [116].

Una tipografía bien elegida mejora y facilita la comprensión y asimilación de la información por parte de los usuarios. Por ejemplo, una fuente en negrita y con letras mayúsculas puede transmitir fuerza y determinación, mientras que una tipografía delicada y manuscrita puede evocar elegancia y sofisticación. En contraste, una tipografía inadecuada puede dificultar la lectura, causar fatiga visual e incluso generar confusión o frustración en el usuario [116].

Las tipografías se pueden clasificar según sus características:

- **Fuentes Serif:** Estas fuentes se caracterizan por tener pequeños remates en los extremos de las letras, transmitiendo una sensación de formalidad. Ejemplos populares incluyen Times New Roman y Georgia [116].
- **Fuentes Sans Serif:** Carecen de remates, lo que les confiere una apariencia más moderna y limpia. Se utilizan comúnmente en proyectos digitales. Ejemplos comunes son Arial, Helvetica y Calibri [116].
- **Fuentes Script o Manuscritas:** Estas fuentes imitan la escritura a mano y suelen transmitir una sensación de personalización y creatividad. Ejemplos incluyen Brush Script y Pacifico [116].
- **Fuentes Decorativas:** Son variadas y altamente estilizadas, utilizadas con fines ornamentales y para llamar la atención. Pueden ser temáticas o artísticas, como las fuentes de Navidad o títulos de películas [116].
- **Fuentes Monoespaciadas:** Cada carácter ocupa el mismo espacio horizontal, lo que es útil en programación y diseño de tablas. Ejemplos son Courier New y Consolas [116].
- **Fuentes Display:** Estas fuentes son diseñadas para títulos y encabezados, siendo llamativas y de alto impacto visual. Ejemplos incluyen Impact y Lobster [116].
- **Fuentes Dingbats:** Contienen símbolos y caracteres especiales en lugar de letras y números, útiles para la creación de iconos y elementos gráficos. Wingdings y Webdings son ejemplos conocidos [116].

Seleccionar la tipografía correcta es esencial para asegurar que el mensaje se transmita de manera eficaz y atractiva para los productos digitales. Hay varios factores clave que se deben considerar para conseguirlo [116]:

- **Legibilidad:** La tipografía debe ser fácil de leer para el público objetivo. Esto incluye considerar el tamaño de la fuente, el espaciado entre letras y palabras, y la claridad de las formas de las letras [116].
- **Personalidad:** La tipografía debe reflejar la identidad y los valores del producto digital [116].
- **Consistencia:** Mantener una apariencia uniforme a lo largo del diseño refuerza la identidad visual y facilita la navegación del usuario [116].
- **Jerarquía:** Utilizar variaciones en la tipografía, como tamaños y estilos (negritas, cursivas), para establecer una jerarquía de información ayuda a los usuarios a identificar elementos clave como títulos, subtítulos y texto principal [116].

- **Combinación de fuentes:** Seleccionar dos o más fuentes que se complementen puede enriquecer el diseño [116].
- **Tamaño y espaciado:** El tamaño de la fuente y el espaciado entre líneas afectan la legibilidad y la estética [116].

4.21. Experiencia de usuario (UX)

4.21.1. Definición de UX

La experiencia de usuario (UX por sus siglas en inglés *User Experience*) se refiere a las percepciones, sentimientos y respuestas que los usuarios tienen al interactuar con un producto digital [26].

4.21.2. Diferencias entre UX y UI

El diseño de interfaz de usuario (UI) y la experiencia de usuario (UX) son conceptos estrechamente relacionados, pero tienen enfoques distintos y desempeñan roles específicos en el desarrollo de productos digitales. Mientras que UX se enfoca en el recorrido y en las interacciones de un usuario en todo el producto digital, UI se centra en cómo se ve y funciona el producto [26].

4.21.3. Tipos de experiencia de usuario

Existen diferentes tipos de experiencia que pueden influir en cómo un usuario percibe un producto digital:

- **Experiencia de navegación:** Se refiere a la forma en que un usuario se desplaza por un producto digital. Incluye la estructura de la navegación y la lógica que conecta las diferentes secciones o páginas. Una navegación intuitiva permite a los usuarios encontrar rápidamente la información que buscan. Por ejemplo, un menú de navegación claro y bien organizado facilita el movimiento entre las distintas secciones de una aplicación [26].
- **Experiencia de usabilidad:** Se enfoca en cómo los usuarios interactúan con los elementos del producto digital. La usabilidad asegura que todos los componentes, como botones, barras de desplazamiento y formularios, funcionen correctamente y de manera consistente. Por ejemplo, un botón que responde rápidamente al ser presionado y realiza la acción esperada contribuye a una experiencia de usuario positiva [26].
- **Experiencia sensorial:** Involucra los elementos que impactan sensorialmente al usuario, como colores, disposición de elementos, sonidos y animaciones. Estos factores pueden influir significativamente en la percepción emocional del usuario respecto al producto. Por ejemplo, colores suaves y animaciones fluidas pueden crear una sensación de tranquilidad y profesionalismo, mientras que colores vibrantes y sonidos dinámicos pueden generar una sensación de energía y entusiasmo [26].

4.21.4. Proceso de experiencia de usuario

La construcción de la experiencia de usuario para un producto se divide en varias etapas, cada una con su propio conjunto de actividades y herramientas [77]:

- **Investigación:** En esta etapa, se evalúan las necesidades de los clientes y se recopilan datos cruciales para entender mejor el contexto y las expectativas del usuario. Las técnicas más comunes incluyen [77]:
 - **Entrevistas:** Proveen información detallada directamente de los usuarios, ayudando a entender sus necesidades, deseos y problemas específicos [113].
 - **Encuestas y cuestionarios:** Recogen datos cuantitativos de una muestra más grande de usuarios, permitiendo identificar tendencias y patrones [113].
 - **Personas:** Creación de arquetipos basados en datos reales para representar diferentes tipos de usuarios [113].
 - **Mapas de empatía:** Herramientas visuales que ayudan a comprender mejor a los usuarios, enfocándose en lo que piensan, sienten, dicen y hacen [113].
 - **Mapa de experiencia del cliente:** Ilustra la experiencia completa de un usuario con un producto [113].
 - **Planteamiento del problema:** Define claramente los desafíos que el producto debe abordar [113].
 - **Diagramas de afinidad:** Organizan y agrupan información en categorías significativas [113].
 - **Análisis de la competencia:** Recopila información sobre tecnologías similares y reseñas de productos existentes [113].
- **Organización:** Organiza toda la información obtenida durante la etapa de investigación utilizando herramientas como [77]:
 - **Mapa de sitio:** Describe las páginas principales de un sitio y su relación, mostrando cómo se conectan [113].
 - **Flujo de usuarios:** Diagrama que muestra la ruta que tomará un usuario en una aplicación para completar una tarea [113].
 - **Estructura alámbrica/ wireframes:** Visualización 2D de un producto digital, que va desde bocetos básicos a lápiz hasta diseños digitales interactivos (de baja, media y alta fidelidad) [113].
- **Diseño:** En la etapa de creación de prototipos, los wireframes de alta fidelidad se transforman en demostraciones interactivas que simulan fielmente la apariencia y el comportamiento del producto. Aquí se integra la investigación realizada en UI, considerando colores, tipografía, tamaños y espaciado de elementos, iconografía, entre otros. Las herramientas utilizadas incluyen Figma, Sketch y Adobe XD [113].
- **Prueba:** Al finalizar la implementación, se realizan pruebas para asegurar que el producto cumple con las necesidades y expectativas de los usuarios [113].

Proceso del Diseño de Experiencia de Usuario



Figura 7: Proceso de diseño de experiencia de usuario

4.22. Desarrollo móvil en Android

4.22.1. Razones para elegir Android como plataforma de desarrollo

Una de las razones principales para elegir Android como plataforma de desarrollo es su alta popularidad en Guatemala. Según estudio la mayoría de celulares usados en el país son Samsung y Huawei, los cuales tienen sistema operativo Android [1].

Según datos recientes, Android domina el tráfico web móvil en el país, con un 82.50% de participación. Esto significa que la mayoría de los usuarios de dispositivos móviles en Guatemala utilizan este sistema operativo, lo que amplía significativamente el alcance y la accesibilidad del producto desarrollado en esta plataforma [118].

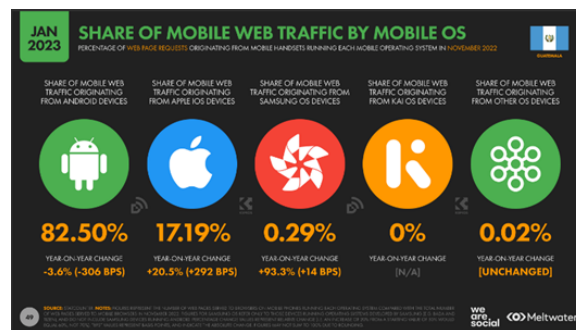


Figura 8: Cuota de tráfico web móvil por sistema operativo

Asimismo, Android es conocido por su diversidad en términos de dispositivos, desde teléfonos de alta gama hasta opciones más accesibles. También ofrece un alto nivel de flexibilidad y opciones de personalización, lo que facilita el desarrollo de aplicaciones. Finalmente, el ecosistema de desarrollo de Android está soportado con una vasta cantidad de recursos, herramientas y una comunidad activa de desarrolladores [3].

4.22.2. Arquitectura de aplicaciones Android

La arquitectura de una aplicación Android se basa en el patrón Modelo - Vista - Controlador (MVC). Aunque es muy común utilizar MVVM (Modelo - Vista - *ViewModel*), pues ofrece una separación más clara de la lógica de presentación y facilita el mantenimiento en comparación con MVC [104].

Componentes

- **Modelo:** Contiene los datos, el estado y la lógica del negocio [22].
- **Vista:** Representa la interfaz de usuario. Se comunica con el *ViewModel* a través de mecanismos de enlace de datos (*data binding*), permitiendo una actualización automática de la UI cuando cambian los datos [22].
- ***ViewModel*:** Actúa como un intermediario entre el Modelo y la Vista. Expone datos y comandos que la Vista puede consumir y ejecutar, y notifica a la Vista sobre cambios en los datos utilizando observables (como *LiveData*) [22].

Flujo de trabajo

- La Vista se enlaza automáticamente al *ViewModel* para observar los datos [22].
- El *ViewModel* obtiene los datos del Modelo y prepara la lógica de presentación [22].
- Cualquier cambio en el Modelo se refleja automáticamente en la Vista a través del *ViewModel* [22].
- La interacción del usuario en la Vista invoca métodos en el *ViewModel*, que a su vez pueden actualizar el Modelo [22].

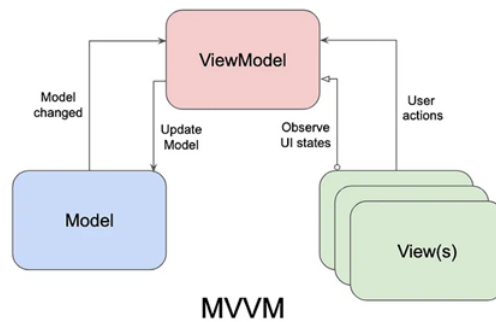


Figura 9: Diagrama de funcionamiento MVVM

4.22.3. Buenas prácticas de desarrollo Android

Al desarrollar aplicaciones para Android, es esencial seguir ciertas buenas prácticas para asegurar la calidad y la eficiencia del producto final [100]:

Uso de *Layouts* adecuados

- ***ConstraintLayout***: Es eficiente y flexible, permitiendo posicionar elementos de manera relativa a otros elementos [100].
- ***LinearLayout***: Organiza los elementos en una sola fila o columna, siendo útil para diseños sencillos y alineaciones básicas [100].
- ***RelativeLayout***: Permite posicionar los componentes en relación a otros o a sus propios padres, ofreciendo más flexibilidad pero con un mayor costo de rendimiento comparado con *ConstraintLayout* [100].

Gestión de recursos

- Utilizar unidades independientes de densidad (dp) para asegurar que los elementos de la interfaz mantengan su tamaño y proporciones adecuadas en dispositivos con diferentes tamaños de pantalla [100].
- Definir colores, estilos y dimensiones en archivos de recursos para promover la reutilización y mantener la consistencia del diseño [100].

- Utilizar *Gradle* para gestionar dependencias, lo que te permitirá mantener el código actualizado y seguro [100].

Optimización del rendimiento

- Minimizar el uso de vistas anidadas para mejorar el rendimiento de la interfaz de usuario [100].
- Evitar operaciones pesadas en el hilo principal [100].

Seguridad

- Proteger los datos del usuario mediante el uso de almacenamiento cifrado y permisos adecuados [100].
- Validar las entradas del usuario para prevenir ataques de inyección y otras vulnerabilidades de seguridad [100].
- Validar permisos de uso de almacenamiento, cámara, micrófono, etc para respetar las políticas de privacidad de Android [100].

5.1. Visión por computadora

Con el fin de cumplir con los objetivos generales y específicos del proyecto, se definió una metodología compuesta por seis etapas, cada una con objetivos claramente definidos. La primera de estas consistió en delimitar el universo de palabras que se utilizaría para entrenar el modelo, buscando palabras de uso cotidiano que permitieran crear una gran cantidad de frases con un alto nivel de utilidad. Una vez definido el universo de palabras, se procedió con la recopilación de datos, la cual incluyó crear un conjunto de datos con videos de las señas correspondientes a las diferentes palabras que se utilizarían. El siguiente paso consistió en preparar el conjunto de datos con los videos recopilados anteriormente, lo cual mayormente consistió en identificar manualmente cada señal para que estos datos pudieran ser utilizados en el proceso de entrenamiento. Al tener listo el conjunto de videos, se pudo proceder al procesamiento de las imágenes, el cual se realizó con la librería Mediapipe de Google, que creó esqueletos de los movimientos capturados en los diferentes videos. Este esqueleto se utilizó para entrenar una red neuronal que fuera capaz de identificar las diferentes señas del universo de palabras seleccionado. Como último paso, se evaluó el desempeño del modelo, buscando áreas de mejora.

5.1.1. Delimitación del universo de palabras

Dada la restricción de tiempo para la finalización del proyecto, resultó esencial establecer un conjunto definido de palabras que se emplearían en el entrenamiento y la evaluación del modelo. La selección de las palabras que formaron parte de este universo fue extremadamente importante para la utilidad del proyecto, ya que elegir las palabras equivocadas pudo haber significado que el proyecto no cumpliera con los objetivos establecidos. Para este proceso, se seleccionaron al menos veinticinco palabras que en conjunto se pudieran utilizar para crear una gran cantidad de frases de uso cotidiano. La primera fase de este proceso requirió de alguien con amplio conocimiento de la lengua de señas de Guatemala, y consistió en identificar al menos cincuenta palabras que se utilizaran con frecuencia. Al tener este dato inicial, se analizó cuáles de estas palabras se podrían utilizar para formar oraciones congruentes, haciendo énfasis en la utilidad y frecuencia de uso de estas. Adicionalmente, se tomó en cuenta el grado de similitud de la seña con las demás palabras del universo de palabras. Esto con el objetivo de reducir la probabilidad de que el modelo confundiera dos señas diferentes debido a que eran muy similares.

Al finalizar la lista original de palabras, se buscó retroalimentación de las personas que ayudaron con la grabación de los videos de entrenamiento. Este paso fue crucial para asegurar que las palabras seleccionadas fueran relevantes y útiles en el contexto de la lengua de señas de Guatemala. Las opiniones de los colaboradores permitieron identificar posibles mejoras en la selección de palabras y garantizar que el conjunto final reflejara adecuadamente las necesidades de los usuarios. Tomando en cuenta la retroalimentación recibida, se ajustó por una última vez la lista de palabras para asegurar que el universo de palabras seleccionado fuera el más adecuado para el proyecto.

5.1.2. Recopilación de datos

Al tener un universo de palabras claramente definido, se pudo proceder a recopilar los datos necesarios para representar todas las palabras elegidas. En esta fase del proyecto, nuevamente se requirió del apoyo de una persona con un profundo conocimiento de la lengua de señas de Guatemala para crear los videos que servirían como base del conjunto de datos. Para que el modelo contara con una amplia variedad de datos de entrenamiento, se realizaron quince tomas de cada seña que formara parte del universo de palabras. Se utilizaron tres ángulos de cámara diferentes, de frente y vista en diagonal de cada lado, realizando cinco tomas desde cada uno de los ángulos. El objetivo de utilizar diferentes ángulos fue mejorar la habilidad del modelo para reconocer las señas en condiciones no ideales, ya que no se podía esperar que el usuario siempre estuviera perfectamente alineado a la cámara. De igual manera, el objetivo de realizar cinco tomas desde cada uno de los ángulos fue que el modelo pudiera identificar las señas correctamente a pesar de las ligeras variaciones que se pudieran presentar en los movimientos.

Al tomar estos videos, se buscó que el usuario que realizara las señas lo hiciera de forma clara y precisa, evitando movimientos innecesarios que pudieran confundir al modelo. Adicionalmente, se tuvo cuidado de que el fondo de los videos fuera lo más simple posible, para asegurar que MediaPipe pudiera identificar correctamente las manos del usuario. Durante esta fase del proyecto se realizo el hallazgo de que las personas surdas realizan las señas de forma diferente, por lo que se tuvo que tomar en cuenta este factor al momento de grabar los videos.

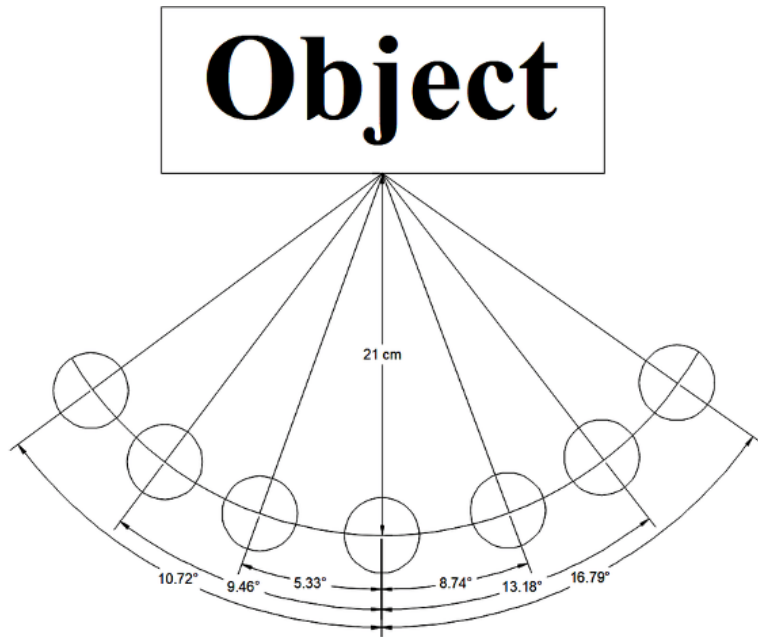


Figura 10: Ejemplos de captura de los videos para el conjunto de datos [72].

5.1.3. Preparación del conjunto de datos

Después de recopilar los datos que formaron parte del conjunto de datos, se procedió con la creación de este. El primer paso para preparar el conjunto de datos fue identificar correctamente todos los videos que se tomaron en el paso anterior. Para esto, nuevamente se requirió de la asistencia de alguien con conocimiento de la lengua de señas de Guatemala, ya que cualquier error llevaría a que el modelo identificara erróneamente las señas. La identificación de los videos se realizó por medio del nombre del archivo, el cual siguió el siguiente formato: Seña-Vista-Toma. El valor de “vista” fue *Frente*, *Der*, o *Izq*, para indicar si el video se tomó con una vista frontal, o una vista lateral desde la izquierda o la derecha, el valor de “seña” indicó la palabra a la que corresponde el video, y el valor de “toma” indicó el número de toma que se realizó. Combinado con el valor de “toma”, se pudieron distinguir claramente los videos entre sí, lo cual resultó útil si el modelo terminó teniendo problemas para identificar la seña en un video en específico.

Tomando en cuenta el hallazgo de que las personas surdas realizan las señas de forma diferente, se tuvo que tomar en cuenta este factor al momento de preparar el conjunto de datos. Para garantizar que el modelo pudiera identificar las señas correctamente, se reflejaron los videos sobre el eje vertical, de forma que las señas realizadas por una persona zurda se vieran como si fueran realizadas por una persona diestra, y viceversa. Esto permitió que el modelo pudiera identificar las señas de forma correcta sin importar si el usuario era zurdo o diestro.



Figura 11: Ejemplo de reflejo de un video

5.1.4. Procesamiento de los videos

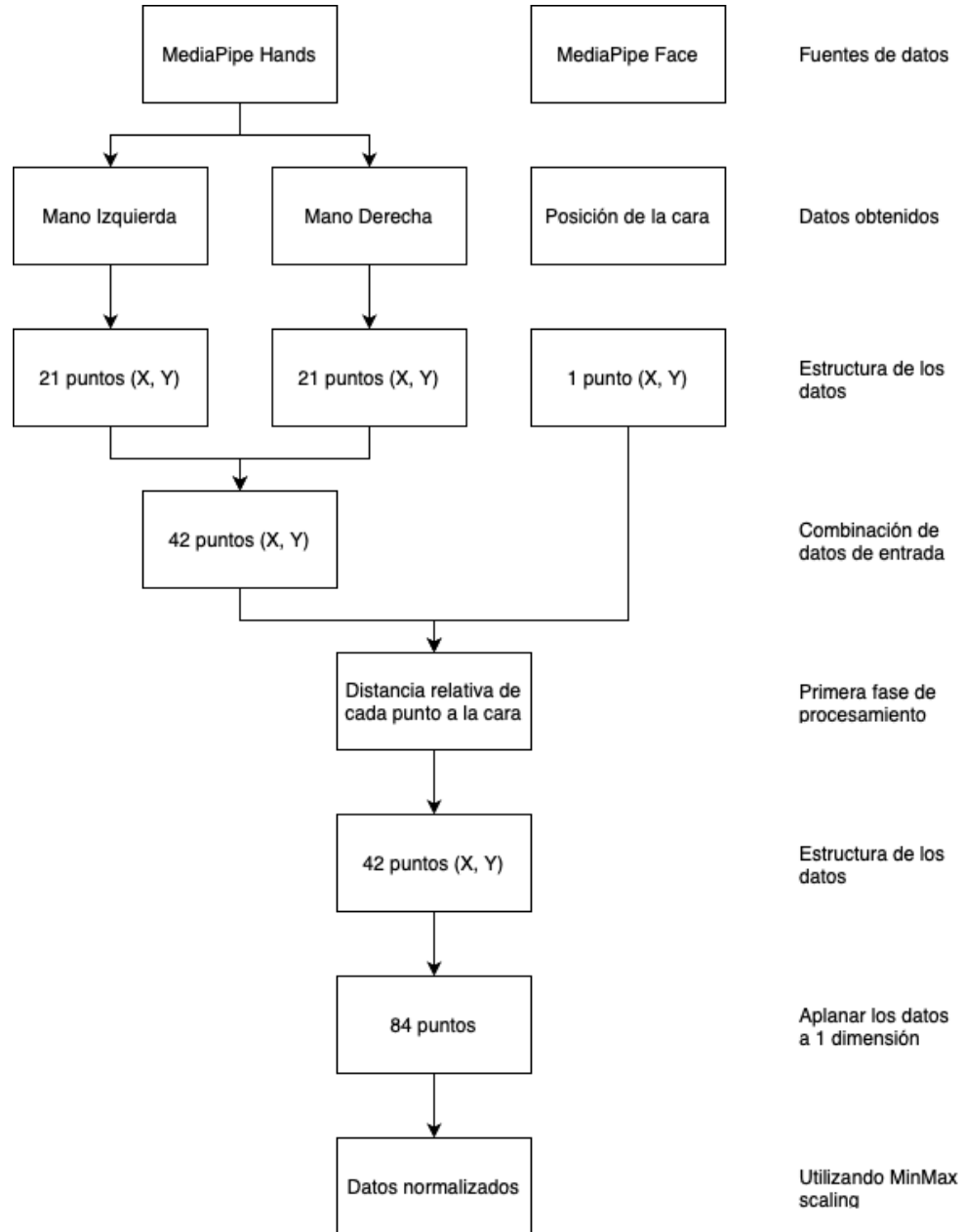


Figura 12: Proceso de preparación del conjunto de datos.

Con el conjunto de datos ya definido, se pudo procesar los videos para crear datos que resultaran útiles para entrenar y validar el modelo. Para esto, se utilizó la librería MediaPipe de Google, la cual permite realizar el reconocimiento de manos en tiempo real. Esta librería utiliza modelos de aprendizaje automático para detectar y seguir la posición de las manos en imágenes o videos, ofreciendo información detallada sobre la mano, incluyendo la detección de puntos clave y gestos. Utilizando esta librería, se crearon esqueletos que representaron los movimientos capturados en cada video.

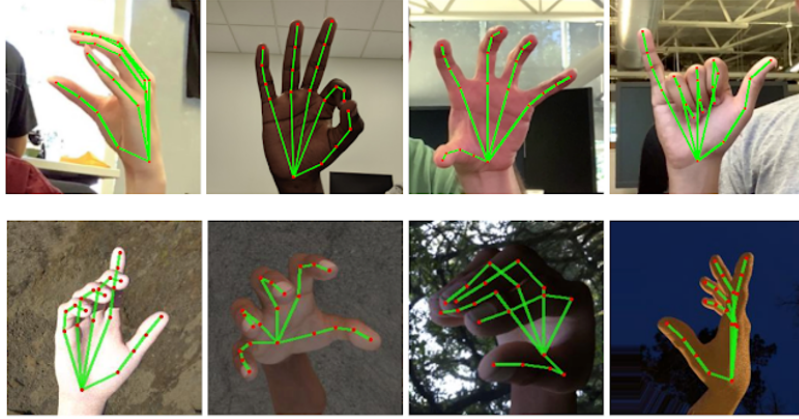


Figura 13: Ejemplo de uso de MediaPipe [53].

Utilizando estos esqueletos, se creó un conjunto de datos con todas las coordenadas relevantes para poder entrenar el modelo. La librería MediaPipe detecta un total de veintinueve puntos clave en cada mano, para un total de cuarenta y dos puntos clave por cada fotograma de video. Debido a que en LENSEGUA la posición de las manos también es relevante, se decidió calcular la posición relativa de cada punto clave con respecto a la cara, procesando las coordenadas en X y Y de cada punto clave. Cada uno de estos puntos cuenta con coordenadas en el eje x y en el eje y, lo cual resulta en un total de ochenta y cuatro características por fotograma de video.

5.1.5. Labeling de los datos

Después de procesar los videos y obtener las coordenadas de los puntos clave, se procedió a etiquetar los datos para que pudieran ser utilizados en el entrenamiento del modelo. Esto consistió en asignar una etiqueta a cada fotograma de video, indicando la seña correspondiente a la palabra que se estaba realizando en ese fotograma. Para esto, se utilizó la función `to_categorical` de la librería Keras de Python, la cual convierte un vector de etiquetas en una matriz binaria. Esta matriz binaria se utilizó para entrenar el modelo, permitiendo que este pudiera identificar las señas de la lengua de señas de Guatemala. Es importante mencionar que, para que las predicciones del modelo se puedan interpretar de forma correcta, se creó un diccionario que relaciona cada etiqueta con la palabra correspondiente.

5.1.6. Normalización de los datos

Para permitir que el modelo pudiera identificar las señas de forma en tiempo real, los datos se procesaron de forma que cada fotograma de video se convirtiera en una secuencia de datos que representara el movimiento de las manos. Tomando esto en cuenta, el conjunto de datos resultante consistió en una serie de secuencias de datos, cada una representando un fotograma de video y las coordenadas de los puntos clave detectados en ese fotograma. Después de calcular estas características, se procedió a normalizarlas para que todas las características tuvieran un valor entre menos uno y uno. Debido a que las coordenadas de los puntos clave podían ser negativas, para indicar que se encontraban a la izquierda o arriba de la cara, se utilizaron dos fórmulas diferentes para normalizar las coordenadas positivas y negativas.

Ecuación para normalizar coordenadas positivas:

$$x_{norm} = \frac{x}{|x_{min}|} \quad (5.1)$$

Ecuación para normalizar coordenadas negativas:

$$x_{norm} = \frac{x}{|x_{max}|} \quad (5.2)$$

Donde x_{norm} es el valor normalizado de la coordenada x , x es el valor de la coordenada x original, x_{min} es el valor mínimo de la coordenada en el conjunto de datos y x_{max} es el valor máximo de la coordenada en el conjunto de datos. Estas fórmulas se utilizaron para normalizar las coordenadas de los puntos clave en el eje x y en el eje y, lo cual permitió que todas las características tuvieran un valor entre -1 y 1. El resultado de este proceso fue un conjunto de datos con las características normalizadas, las cuales se utilizaron para entrenar el modelo. A continuación, se muestra un ejemplo de cómo se veían los datos después de ser procesados.

	LABEL	COORDS
0	0	[-0.14792899408284024, 0.37333333333333335, -0...
1	0	[-0.14792899408284024, 0.37333333333333335, -0...
2	0	[-0.14595660749506903, 0.37244444444444447, -0...
3	0	[-0.14595660749506903, 0.37155555555555553, -0...
4	0	[-0.14595660749506903, 0.37066666666666664, -0...

Figura 14: Ejemplo de datos procesados.

5.1.7. Oversampling

Debido al tamaño limitado del conjunto de datos, se utilizó la técnica de oversampling para aumentar la cantidad de datos disponibles para el entrenamiento del modelo. Esta técnica consistió en duplicar los datos existentes, lo cual permitió que el modelo pudiera entrenarse con una mayor cantidad de datos y mejorar su capacidad para identificar las señas de la lengua de señas de Guatemala. El oversampling se realizó copiando los datos existentes y agregando ruido a las coordenadas de los puntos clave, lo cual permitió que el modelo pudiera generalizar mejor a datos no vistos previamente. Con cada uno de los modelos entrenados, se evaluó el desempeño del modelo utilizando el conjunto de datos original y el conjunto de datos aumentado, lo cual permitió determinar si el oversampling había mejorado la capacidad del modelo para identificar las señas.

5.1.8. Entrenamiento del modelo

Para entrenar el modelo, se utilizó el 80% del conjunto de datos procesado en la fase anterior, mientras que el 20% restante se utilizó para validar el modelo. Durante este proceso, se probaron diferentes hiperparámetros y técnicas de entrenamiento para encontrar la combinación que ofreciera el mejor desempeño. Adicionalmente, una vez seleccionada una red neuronal, se evaluó el desempeño de esta de forma continua para poder hacer ajustes y optimizaciones que mejoraran sus resultados.

Debido a la naturaleza de los datos, se decidió utilizar una red neuronal *feedforward* para entrenar el modelo. Este modelo se utilizó para entrenar un clasificador que pudiera identificar las señas de la lengua de señas de Guatemala utilizando los datos procesados en la fase anterior. En cada iteración, se ajustaron los hiperparámetros de la red neuronal, como el número de capas, el número de neuronas por capa y la tasa de aprendizaje, con el objetivo de mejorar la precisión del modelo. Adicionalmente, se probaron diferentes funciones de activación y optimizadores para encontrar la combinación que resultara en el mejor desempeño.

```

model = tf.keras.Sequential()
model.add(layers.Input(shape=(data.shape[1],)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(numClasses, activation='softmax'))

```

Figura 15: Ejemplo de una red neuronal *feedforward* sencilla.

Con el objetivo de mejorar la capacidad del modelo para generalizar a datos no vistos previamente, se utilizó la técnica de *dropout* durante el entrenamiento. Esta técnica consistió en desactivar aleatoriamente un porcentaje de las neuronas en cada iteración, lo cual permitió que el modelo no dependiera de una sola característica para hacer predicciones. Adicionalmente, se probaron diferentes funciones de activación y optimizadores para encontrar la combinación que resultara en el mejor desempeño. Debido a la naturaleza de los datos, se decidió utilizar la función de activación tangente hiperbólica, ya que esta función es adecuada para datos normalizados entre -1 y 1.

```

model = tf.keras.Sequential()
model.add(layers.Input(shape=(data.shape[1],)))
model.add(layers.Dense(512, activation='tanh'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(256, activation='tanh'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(128, activation='tanh'))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(64, activation='tanh'))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(numClasses, activation='softmax'))

```

Figura 16: Implementación de mejoras en la red neuronal *feedforward*.

5.1.9. Evaluación del modelo

Como último paso, se evaluó la habilidad del modelo para reconocer diferentes señas de la lengua de señas de Guatemala. Para este proceso, se utilizó una serie de videos que no se habían utilizado en el entrenamiento del modelo, los cuales se procesaron de la misma forma que los videos de entrenamiento. La primera de estas evaluaciones se realizó con videos que contenían una sola seña, lo cual permitió evaluar la capacidad del modelo para identificar señas individuales. Posteriormente, se evaluó el modelo con videos que contenían frases completas, lo cual permitió evaluar la capacidad del modelo para identificar frases completas.

Como métrica de evaluación, se calculó el porcentaje de videos en los cuales el modelo identificó de manera correcta la palabra correspondiente a la seña. Adicionalmente, se buscaron errores comunes cometidos por el modelo para que estos pudieran ser corregidos. Un ejemplo de esto sería si el modelo constantemente confundía una seña por otra, lo cual indicaría que el modelo necesitaba más datos correspondientes a una de esas señas o que se debía modificar el modelo. En este proceso de validación manual, resultó útil la codificación con la cual se identificaron los videos en la fase de preparación del conjunto de datos.

$$\text{Precisión} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}} \quad (5.3)$$

Adicional a la precisión, se obtuvo la matriz de confusión, la cual permitió identificar si existían señas que eran constantemente confundidas por el modelo. Utilizando esta información, se ajustaron los hiperparámetros del modelo y se realizaron mejoras en el conjunto de datos para corregir los errores comunes cometidos por el modelo. La matriz de confusión también permitió calcular la sensibilidad y el puntaje F1, los cuales se utilizaron como métricas adicionales para evaluar el desempeño del modelo.

Debido a que el tiempo de procesamiento del modelo era crucial para el éxito del proyecto, se calculó la velocidad de procesamiento del modelo, en fotogramas por segundo. Los videos que se utilizaron en este proyecto fueron grabados a 30 fotogramas por segundo, por lo que se esperaba que el modelo pudiera procesar los videos a una velocidad similar. Esta métrica, de fotogramas procesados por segundo, se calculó utilizando la siguiente fórmula:

$$\text{Velocidad de procesamiento} = \frac{\text{Número de fotogramas procesados}}{\text{Tiempo de procesamiento}} \quad (5.4)$$

5.1.10. Evaluación del conjunto de datos

Para evaluar el conjunto de datos, se utilizó un analizador de componentes principales (PCA) para reducir la dimensionalidad de los datos y visualizarlos en un espacio de dos dimensiones. El PCA permitió identificar si los datos se agrupaban de forma coherente, lo cual indicaría que el modelo tendría una alta precisión al identificar las señas. Debido a que el PCA se realizó con los datos de todos los fotogramas de video, se esperaba que los datos estuvieran algo dispersos, ya que cada fotograma de video representaba un punto en el espacio de dos dimensiones. Lo que se buscaba con el PCA era identificar si existían señas que se agrupaban de forma similar, lo cual indicaría que el modelo tendría problemas para distinguir entre esas señas.

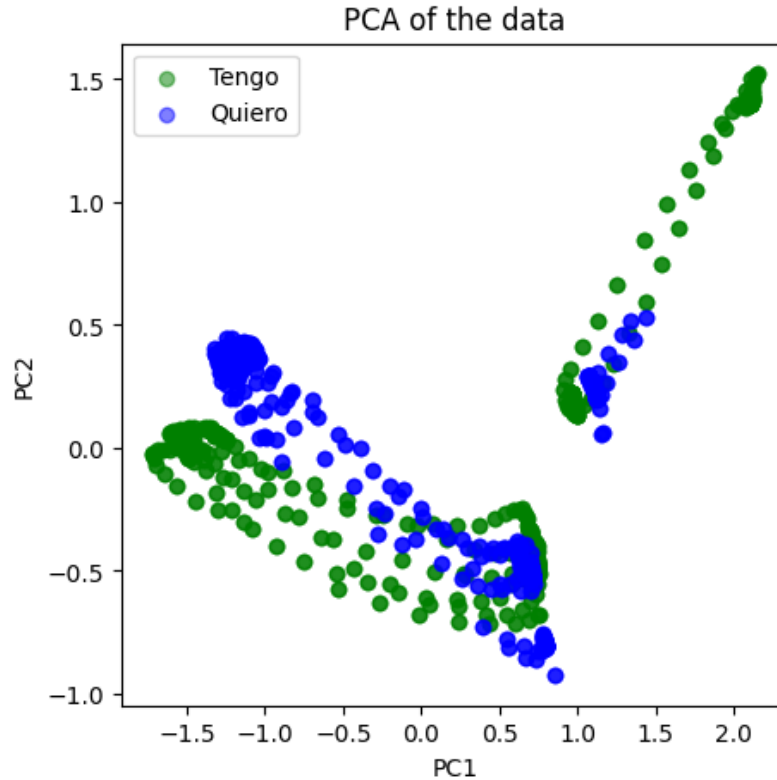


Figura 17: Ejemplo de visualización de datos con PCA.

Durante el proceso de entrenamiento del modelo, se identificaron algunas señas que eran comúnmente confundidas por el modelo. En estos casos, el análisis de los datos con PCA permitió identificar si existían diferencias significativas entre las señas, lo cual permitió identificar si el problema era causado por la similitud entre las señas o por la fallas en el modelo. En la Figura 17 se muestra un ejemplo de cómo se identificaron señas que no se distinguían claramente en el espacio de dos dimensiones.

Utilizando la matriz de confusión y el análisis de los datos con PCA, se buscaba identificar si los errores cometidos por el modelo eran causados por la similitud entre las señas o por fallas en el modelo. En el caso de que los errores fueran causados por la similitud entre las señas, se esperaba que tanto la matriz de confusión como el análisis de los datos con PCA mostraran que las señas se agrupaban de forma similar. En el caso de que los errores fueran causados por fallas en el modelo, se esperaba que la matriz de confusión mostrara que el modelo cometía errores aleatorios, mientras que el análisis de los datos con PCA mostrara que las señas se agrupaban de forma coherente. La Figura 18 muestra un ejemplo de la matriz de confusión utilizada para evaluar el desempeño del modelo.

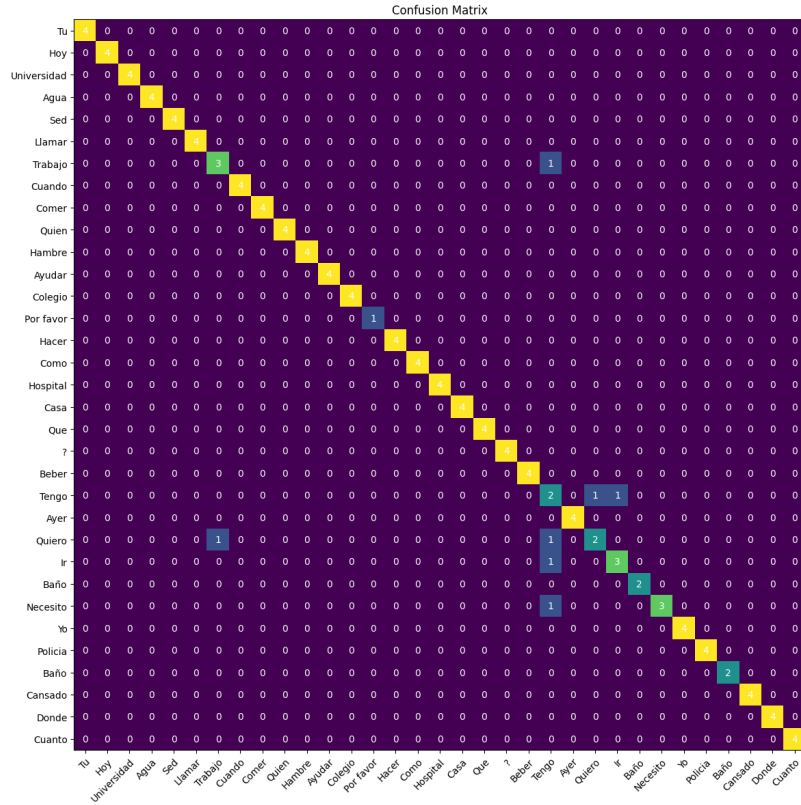


Figura 18: Ejemplo de matriz de confusión.

5.1.11. Publicación del modelo

Una vez que el modelo fue entrenado y evaluado, se procedió a publicarlo en un servidor web para que pudiera ser utilizado por los usuarios. Para esto, se utilizó la librería Flask de Python, la cual permitió crear una API que se comunicara con el modelo y que pudiera ser accedida por los usuarios a través de una interfaz web. La API permitió que los usuarios enviaran videos a través de la interfaz web, los cuales se procesaban con el modelo y se devolvía la palabra correspondiente a la seña identificada. Esto requirió de un programa adicional, diseñado para procesar los videos enviados por los usuarios y enviarlos a la API para su procesamiento. Este programa carga el modelo entrenado, al igual que algunos parámetros adicionales, y se encarga de recibir los videos de los usuarios, procesarlos con el modelo, y devolver la palabra correspondiente a la seña identificada. Para validar el funcionamiento de la Interfaz de Programación de Aplicaciones (API), se realizaron pruebas con videos de prueba utilizando Postman.

5.2. Procesamiento de lenguaje natural (GPT-3.5-Turbo)

5.2.1. Creación de conjunto de datos

Recolección de datos

El proceso de recolección de datos consistió en buscar y seleccionar diversas frases en español que aplicaran los principios gramaticales de LENSEGUA. Para ello, se exploraron recursos en línea, como grupos en redes sociales donde usuarios sordos comparten sus pensamientos y experiencias utilizando esta gramática en sus interacciones escritas. En particular, se identificó un grupo privado en la red social Facebook llamado “Lengua de Señas de Guatemala (LSG)”. Se envió una solicitud a los administradores del grupo para obtener acceso, y, una vez aprobada, se procedió a recopilar diversos comentarios que aplicaban esta gramática. No obstante, para proteger la privacidad de los usuarios, no se almacenó ninguna información personal, y se excluyeron comentarios que contuvieran información sensible o identificativa. Tras la recolección y almacenamiento de los datos en un archivo CSV, se generaron -de forma manual- las versiones gramaticalmente correctas en español de cada frase recopilada. En el Cuadro 1 se presenta la estructura utilizada para almacenar la información, mostrando ejemplos de frases en LENSEGUA y sus correspondientes interpretaciones en español.

LENSEGUA	Español
aeropuerto dónde pregunta	¿Dónde está el aeropuerto?
antes tu estar enojado pregunta	¿Estabas enojado?
haber carro mucho	Hay tráfico.
pasado yo pasaporte perder	Perdí mi pasaporte.
...	...

Tabla 1: Estructura del conjunto de datos desarrollado.

Seguidamente, se establecieron contactos con organizaciones guatemaltecas que trabajan con la comunidad sorda, entre ellas la Asociación Educativa para el Sordo (ASEDES). Esta organización colaboró activamente en la generación de más ejemplos específicos que ilustraban la aplicación de la gramática de LENSEGUA en diferentes contextos, desde situaciones cotidianas hasta emergencias. Ellos se centraron en utilizar un vocabulario que tuviera equivalentes directos en LENSEGUA, ya que no todas las palabras en español tienen una traducción literal en lengua de señas. Además, colaboraron con la creación de las versiones gramaticalmente correctas en español para cada una de estas frases, manteniendo la misma estructura descrita anteriormente.



Figura 19: Reunión de equipo de trabajo con ASEDES.

Durante esta etapa, se optó por recibir un curso virtual sobre la gramática de LENSEGUA ofrecido por la organización Sordos Latinos Guatemala. Este curso proporcionó una comprensión detallada de las reglas y estructuras gramaticales específicas de LENSEGUA, lo que permitió generar ejemplos adicionales -propios- con mayor precisión. Además, la capacitación también facilitó la validación de los textos anteriormente obtenidos, asegurando que el contenido estuviera alineado con las normas gramaticales de LENSEGUA.

Por último, se tomó como referencia en el trabajo de Fan *et al.* (2023) [45] y se procedió a utilizar la plataforma de Claude.ai para generar más datos. Más específicamente, se le proporcionó a esta herramienta de generación de texto (basada en inteligencia artificial) el archivo CSV anteriormente desarrollado y se solicitó que generara entradas adicionales para ampliar el *dataset*. Este proceso tuvo como objetivo incrementar tanto la cantidad como la diversidad de los ejemplos disponibles en el *dataset*. Posteriormente, se revisaron las frases generadas para asegurar que cumplieran con las normas gramaticales de LENSEGUA y se verificó que no hubiera duplicados en el contenido.

Cabe destacar que no se utilizó ChatGPT para generar los datos adicionales, ya que no se considera óptimo *fine-tunear* un modelo con salidas generadas por la misma herramienta. Este acercamiento no aportaría nueva información y resultaría en redundancia en el aprendizaje [45]. Por ello, se eligió Claude.ai, una plataforma independiente, para garantizar que los datos fueran verdaderamente nuevos, permitiendo así que el proceso de *fine-tuning* en GPT-3.5-Turbo se basara en entradas totalmente nuevas y variadas.

Data augmentation

Una vez generados los datos iniciales, se procedió a dividir el conjunto de datos en dos subconjuntos: un 80% se destinó al entrenamiento y un 20% a la evaluación. Esta división fue necesaria para evaluar el rendimiento del modelo de manera efectiva y garantizar que el modelo final no se sobreajustara a los datos de entrenamiento.

Después de esta separación, se desarrolló un programa en Python con el propósito de realizar *data augmentation* en el conjunto de entrenamiento. Es decir, ampliar el *dataset* existente mediante la

generación de nuevas frases. El enfoque consistió en crear variaciones de las frases originales mediante la sustitución de palabras por otras similares, manteniendo la coherencia contextual. Por ejemplo, una oración como ‘papá cirujano ser’ podría transformarse en ‘abuelo doctor ser’ y ‘tío odontólogo ser’, entre otras (ver la Figura 20).

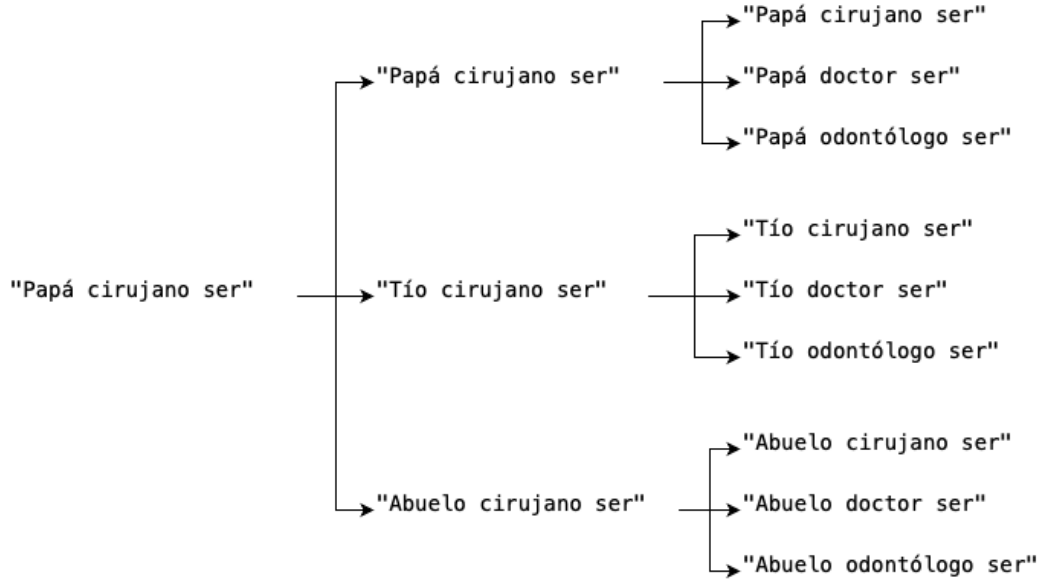


Figura 20: Generación recursiva de nuevas frases.

Para ello, se definió un conjunto de listas que agrupaban palabras intercambiables, como profesiones, lugares, objetos, entre otros. El programa analizaba cada oración del conjunto de entrenamiento y, al identificar términos presentes en estas listas, generaba todas las posibles combinaciones de oraciones mediante sustituciones recursivas. De esta manera, si una oración contenía varias palabras intercambiables, se creaban múltiples variantes de la misma, ampliando significativamente el *dataset* de entrenamiento.

Es importante mencionar que el sistema de *data augmentation* también modificaba las interpretaciones teóricas definidas en el *dataset*. Esto aseguraba que cada nueva frase en LENSEGUA tuviera su correspondiente versión en español, aumentando así la cantidad de datos disponibles para el *fine-tuning* del modelo GPT-3.5-Turbo.

Preparación del conjunto de datos

Con el *dataset* de entrenamiento ampliado y el conjunto de evaluación definido, se procedió a preparar los datos utilizando Python. Este proceso incluyó la corrección automatizada de mayúsculas y la adición de signos de puntuación, entre otros ajustes. Además, se organizaron los datos en un formato compatible con OpenAI, utilizando un formato JSON con la siguiente estructura:

```

1 'messages': [
2   {'role': 'system', 'content': DEFAULT_SYSTEM_PROMPT},
3   {'role': 'user', 'content': question + ' ->'},
4   {'role': 'assistant', 'content': answer},
5 ]
  
```

En esta estructura, el *DEFAULT SYSTEM PROMPT* constituye el mensaje inicial que establece el contexto y las directrices fundamentales para el comportamiento del modelo. El campo *question* contiene la frase que aplica la gramática de LENSEGUA, mientras que el campo *answer* proporciona la interpretación gramaticalmente correcta en español.

Al finalizar con la preparación, los datos procesados se guardaron en archivos JSONL, asegurando así su disponibilidad para el siguiente paso en el proceso de *fine-tuning* del modelo.

5.2.2. Fine-tuning

Identificación de parámetros óptimos

Para asegurar un rendimiento óptimo del modelo *fine-tuneado* GPT-3.5-Turbo, se realizaron pruebas preliminares con una porción del conjunto de datos (se excluyeron las frases generadas mediante *data augmentation*). Estas pruebas permitieron experimentar con distintas configuraciones para identificar los parámetros más adecuados. Al emplear un subconjunto del *dataset*, se optimizó el proceso sin incurrir en los altos costos asociados a múltiples entrenamientos completos, logrando además acelerar el proceso ajuste fino.

Los parámetros ajustados incluyeron:

- **Epochs:** Número de veces que el modelo recorre el conjunto de datos de entrenamiento completo. Ajustar este parámetro fue clave para evitar tanto el subentrenamiento como el sobreentrenamiento.
- **Batch size:** Cantidad de muestras procesadas antes de actualizar los parámetros del modelo. Valores pequeños favorecen la generalización, mientras que valores grandes aceleran el entrenamiento.
- **Learning Rate Multiplier:** Controla la magnitud de los ajustes que realiza el modelo en función de los errores durante el entrenamiento. Un ajuste adecuado de este parámetro fue crucial para asegurar la convergencia del modelo.

Fine-tuning del modelo GPT-3.5-Turbo

Una vez identificados los parámetros óptimos, se realizó el *fine-tuning* del modelo GPT-3.5-Turbo en la plataforma de OpenAI. Para ello, se creó una nueva tarea de *fine-tuning* y se seleccionó el modelo base correspondiente. Luego, se cargaron los archivos JSONL generados previamente. El archivo de entrenamiento incluía 4,192 entradas, combinando frases originales y aquellas generadas mediante *data augmentation*. Por su parte, el archivo de validación contenía 200 entradas para evaluar el rendimiento del modelo durante el ajuste. Finalmente, se configuraron los parámetros y se inició el proceso de *fine-tuning*.

Durante el proceso de ajuste, se monitoreó la pérdida de entrenamiento y validación para verificar que el modelo no presentara *overfitting*. El *overfitting* ocurre cuando un modelo aprende demasiado bien los datos de entrenamiento, afectando así su capacidad de generalización.

Evaluación del modelo GPT-3.5-Turbo *fine-tuneado*

Distancia de Levenshtein

Tras completar el proceso de *fine-tuning*, se procedió a evaluar el desempeño del modelo en la generación de interpretaciones utilizando la distancia de Levenshtein. Esta métrica mide el número mínimo de operaciones necesarias (inserciones, eliminaciones o sustituciones) para transformar una cadena de texto en otra [114]. Se consideró que esta métrica era particularmente adecuada para este contexto, ya que el objetivo era que el modelo generara interpretaciones sin introducir variaciones significativas en el vocabulario ni en la estructura de las frases.

Para llevar a cabo esta evaluación, se desarrolló un programa en Python que permitió al modelo *fine-tuneado* generar interpretaciones para cada frase del conjunto de validación, utilizando un *prompt* básico que definía el rol del sistema como intérprete de LENSEGUA. A partir de esto, se compararon las interpretaciones generadas con las interpretaciones teóricas predefinidas y se calculó la distancia de Levenshtein para cada par. Finalmente, se promediaron las 200 distancias obtenidas, lo que permitió determinar un valor representativo del desempeño del modelo.

Con el fin de establecer una línea de comparación, se repitió el análisis con el modelo GPT-3.5-Turbo estándar, utilizando las mismas 200 frases y el mismo *prompt*. En este caso, el modelo que presentó la menor distancia promedio fue considerado el más efectivo para la tarea de interpretación.

Local Interpretable Model-agnostic Explanations (LIME)

La técnica de *Local Interpretable Model-agnostic Explanations* (LIME) se utilizó para obtener una comprensión más profunda del impacto del *fine-tuning*. LIME permitió identificar, para frases individuales, qué palabras tuvieron mayor influencia en la interpretación generada por cada modelo. De este modo, se buscaba determinar cómo los elementos característicos de LENSEGUA influían en las respuestas de cada modelo, y si el modelo *fine-tuneado* mostraba una recepción más favorable hacia estos elementos en comparación con el modelo estándar.

El análisis se realizó utilizando Python y la librería LIME. A este programa se le proporcionaron tres frases de prueba, entre ellas “pasado yo ir no”. Cada frase fue sometida a un proceso de perturbación, que consistió en generar cien variaciones mediante la eliminación aleatoria de palabras. Estas frases modificadas se enviaron a ambos modelos (el modelo *fine-tuneado* y el modelo GPT-3.5-Turbo estándar) para que generaran sus interpretaciones utilizando el *prompt* básico previamente mencionado.

El Cuadro 2 presenta un ejemplo del resultado este proceso. En la primera fila, se muestra la frase seleccionada y su correspondiente interpretación teórica, que en este caso es “No fui”. Seguidamente, se detallan algunas perturbaciones generadas a partir de la frase original, junto con las interpretaciones producidas por el modelo *fine-tuneado*. Es importante mencionar que este mismo procedimiento se realizó utilizando el modelo GPT-3.5-Turbo estándar, generando así un conjunto comparable de interpretaciones para cada perturbación.

Para comparar las interpretaciones generadas con respecto a las interpretaciones teóricas, se utilizó la distancia de Levenshtein normalizada. Este valor, que oscila entre 0 y 1, permite determinar la similitud de entre dos frases. Un valor de 1 indica que las frases son idénticas, mientras que valores más bajos reflejan mayores diferencias.

A partir de los resultados obtenidos mediante las perturbaciones y sus correspondientes distancias de Levenshtein normalizadas, LIME permitió cuantificar la influencia de cada palabra en las interpretaciones generadas tanto por el modelo estándar como el *fine-tuneado*. Un valor positivo

Frase en LENSEGUA: pasado yo ir no		
Interpretación teórica: No fui.		
Frase / Perturbación	Interpretación	Distancia de Levenshtein (norm.)
pasado yo ir no	No fui.	1.0
yo	Yo.	0.2857
pasado yo no	No fui.	1.0
pasado	En el pasado.	0.1538
...

Tabla 2: Ejemplos de perturbaciones interpretadas por el modelo *fine-tuneado* junto con sus distancias de Levenshtein normalizadas correspondientes.

indicaba que una palabra ayudaba al modelo a generar interpretaciones más cercanas a la teórica, mientras que un valor bajo o negativo sugería que su presencia confundía al modelo, resultando en respuestas más distantes de lo esperado.

5.2.3. Prompt engineering

Durante la evaluación del modelo *fine-tuneado*, se identificaron varios aspectos que afectaron negativamente la calidad de las interpretaciones generadas. En particular, se observaron dificultades en el manejo de expresiones temporales, lo que llevaba a errores de interpretación y redundancias en las respuestas. Para abordar estas limitaciones, se implementó una fase de *prompt engineering* con el objetivo de mejorar la coherencia y precisión de las interpretaciones.

Se desarrollaron nuevas versiones del *prompt* de manera iterativa, incorporando instrucciones más precisas y delimitadores específicos. Cada nueva versión se diseñó a partir de la evaluación de la versión anterior, lo que permitió un refinamiento continuo de las instrucciones. Las modificaciones incluyeron la adición de reglas explícitas para el procesamiento de expresiones características de LENSEGUA, así como ejemplos contextuales. Estos cambios se orientaron a potenciar la capacidad del modelo para interpretar correctamente el contexto y a minimizar las ambigüedades en las respuestas.

Para evaluar cada *prompt*, se utilizó el programa de Python previamente desarrollado en la fase de *fine-tuning*. Este programa permitió generar interpretaciones para cada frase del conjunto de validación, utilizando las distintas versiones de los *prompts*. A partir de estas interpretaciones, se calculó la distancia de Levenshtein promedio para medir la similitud entre las respuestas generadas y las interpretaciones teóricas predefinidas. De esta manera, se logró determinar una distancia promedio asociada a cada *prompt*, lo que proporcionó una medida cuantitativa del rendimiento de cada versión. Cabe destacar que este proceso de evaluación se realizó tanto con el modelo *fine-tuneado* como con el modelo GPT-3.5-Turbo estándar, lo que facilitó la comparación del impacto de cada *prompt* en ambos modelos.

5.2.4. Implementación y evaluación del sistema

Uno de los objetivos principales de este proyecto fue que miembros de la comunidad sorda evaluaran la coherencia y precisión de las interpretaciones generadas por el modelo *fine-tuneado*. Para ello, se desarrolló una plataforma web que le permitió a usuarios interactuar directamente con el sistema y evaluar su rendimiento en tiempo real. Además, se diseñó una encuesta dirigida a intérpretes, estudiantes y usuarios familiarizados con LENSEGUA, con el propósito de calificar el desempeño del modelo *fine-tuneado* en comparación con la versión estándar de GPT-3.5-Turbo. Esta combinación

de interacción directa a través de la plataforma web y una evaluación estructurada proporcionó una visión completa sobre la efectividad de las interpretaciones generadas, ofreciendo una base sólida para analizar el rendimiento del sistema.

Desarrollo de plataforma web

Para facilitar la interacción con el modelo *fine-tuneado*, se desarrolló una plataforma web local (no accesible desde el internet) para que permitía a los usuarios probar el sistema de manera controlada. Esta aplicación se estructuró en dos componentes principales: el *backend*, responsable del procesamiento de las solicitudes a través de Python y Flask, y el *frontend*, encargado de la visualización de los resultados en tiempo real.

El *backend* del sistema estableció una conexión con el API de OpenAI, lo que permitió la interacción directa tanto con el modelo *fine-tuneado* como con otros modelos de OpenAI. Además, se encargó de gestionar las solicitudes de los usuarios: cuando un usuario introducía una frase, el *backend* enviaba la solicitud a la API, procesaba la respuesta generada por el modelo y devolvía la interpretación correspondiente. Se implementó también un sistema de almacenamiento de historial de solicitudes, que permitió a los usuarios visualizar el registro completo de sus consultas.

El *frontend* se desarrolló utilizando HTML y CSS, lo que permitió crear una interfaz de usuario intuitiva y eficiente. La página web adoptó un formato similar al de un *chat*, facilitando a los usuarios la entrada de frases y la recepción de respuestas en tiempo real. Además, se incluyó la opción de seleccionar entre el modelo *fine-tuneado* y el modelo GPT-3.5-Turbo estándar. La interfaz fue optimizada para dispositivos de escritorio, garantizando así una experiencia clara y amigable.

Desarrollo de encuesta

Para evaluar la coherencia de las frases generadas por el modelo *fine-tuneado*, se creó una encuesta mediante Google Forms. La encuesta comenzaba con un consentimiento informado que detallaba el propósito del estudio y el proceso de evaluación, asegurando que los participantes comprendieran plenamente la metodología antes de iniciar.

La encuesta se organizó en secciones, cada una enfocada en evaluar una frase específica mediante tres preguntas. Las dos primeras preguntas solicitaban a los participantes que calificaran, en una escala del 1 al 5 (donde 1 representaba baja coherencia y 5 alta coherencia), las interpretaciones generadas tanto por el modelo estándar GPT-3.5-Turbo como por el modelo *fine-tuneado*. La tercera pregunta requería una comparación directa entre ambas interpretaciones, pidiendo a los participantes que seleccionaran la que consideraban más coherente o adecuada para la frase en cuestión.

Además, la encuesta incluía una sección adicional dedicada a evaluar los diferentes *prompts* desarrollados. En esta sección, los participantes debían seleccionar, entre varias opciones, la interpretación que encontraban más clara y comprensible. Todas estas interpretaciones fueron generadas por el modelo *fine-tuneado*, pero utilizando diferentes *prompts*. Esta metodología permitió no solo evaluar la coherencia general del modelo *fine-tuneado*, sino también validar cuál *prompt* resultaba más efectivo para la generación de interpretaciones precisas.

Es fundamental mencionar que la encuesta se limitó a participantes que eran personas sordas, familiares de personas sordas, estudiantes de LENSEGUA o intérpretes. Esta selección garantizó que las evaluaciones provinieran de individuos con un conocimiento relevante sobre la lengua, lo que contribuyó a la validez de los resultados obtenidos.

Pruebas con usuarios

En la etapa final de evaluación del modelo, se organizó una reunión presencial con la asociación guatemalteca de sordos En-Señas. La sesión inició con una presentación detallada del modelo *fine-tuneado*, en la que se explicó su propósito y funcionamiento. Además, los participantes, en su mayoría personas sordas y un intérprete, pudieron observar el sistema en acción y hacer preguntas sobre su desarrollo. Posteriormente, se les pidió completar la encuesta previamente descrita, donde evaluaron la coherencia de las interpretaciones generadas por el modelo. Esta interacción directa fue clave para recopilar comentarios y sugerencias en tiempo real, lo que contribuyó a una evaluación más completa del sistema. Las respuestas de estos participantes, como se detallará más adelante, permitieron confirmar la efectividad del modelo desarrollado, resaltando su utilidad en la interpretación de frases que aplican la gramática de LENSEGUA.

Para ampliar el alcance de la evaluación, la encuesta se compartió también con otras organizaciones y clubes de LENSEGUA. Esto permitió obtener un conjunto más amplio de respuestas, enriqueciendo así el análisis de la coherencia y utilidad del modelo *fine-tuneado*.

5.3. Procesamiento de lenguaje natural (LLaMA)

5.3.1. Creación de conjunto de datos

Recolección de datos

El conjunto de datos utilizado para este módulo mantuvo la misma estructura que el empleado durante el entrenamiento del modelo GPT-3.5-Turbo. Sin embargo, se optó por crear datos completamente diferentes para este proceso.

Cada entrada en el conjunto de datos consistía en una frase escrita que utilizaba la gramática específica de LENSEGUA y su correspondiente versión gramaticalmente correcta en español. Este enfoque permitió representar de manera adecuada las características únicas de LENSEGUA y su relación con el español escrito. Los datos fueron diseñados por expertos con conocimiento avanzado en la gramática de LENSEGUA, asegurando la coherencia y precisión en cada ejemplo.

Los 560 pares de frases se guardaron en un archivo CSV, lo que facilitó su organización y procesamiento en las etapas posteriores del proyecto.

Preparación del conjunto de datos

Una vez recopilados los datos, estos se procesaron para cumplir con el formato requerido por el modelo seleccionado, Alpaca. Este modelo, desarrollado por la Universidad de Stanford, es una versión ajustada de LLaMA. Su formato compatible, estructurado en JSON, se compone de tres campos principales:

```
1 {  
2   "instruction": "",  
3   "input": [LENSEGUA],  
4   "output": [INTERPRETACION]  
5 }
```

El campo *instruction* especifica la tarea que se desea realizar con el modelo, es decir, la instrucción que la LLM debe completar. El *input* es un campo que proporciona contexto adicional. Este contexto puede ser información o detalles relevantes que ayudan al modelo a interpretar correctamente la instrucción, como referencias gramaticales o aclaraciones específicas. Por otro lado, el campo *output* contiene la respuesta esperada según la instrucción proporcionada.

Para organizar adecuadamente los datos, se dividieron en tres archivos JSON, siguiendo la estructura mencionada de *instruction*, *input* y *output*. Más específicamente, 70% de los datos se destinaron al entrenamiento del modelo, 15% para validación y 15% para prueba. Esta división equilibrada permitió un ajuste efectivo del modelo y una evaluación precisa de su rendimiento en las distintas etapas del proceso.

5.3.2. Configuración inicial

Para trabajar con LLaMA, se empleó la biblioteca *Transformers* de Hugging Face, la cual ofrece herramientas eficientes para cargar, entrenar y realizar *fine-tuning* en modelos de lenguaje. En particular, se utilizó la versión LLaMA 3.0 con 8 mil millones de parámetros (8B) debido a las limitaciones de recursos disponibles.

Entorno de desarrollo

La biblioteca *Transformers* ofrece compatibilidad con Unidades de Procesamiento Gráfico (GPUs) mediante la utilización de la Arquitectura Unificada de Dispositivos de Cómputo (CUDA), desarrollada por NVIDIA. CUDA es una plataforma que maximiza el poder computacional de las GPUs, permitiendo realizar procesamiento en paralelo de manera eficiente. Como se muestra en la Figura 21, las GPUs son fundamentales para manejar las exigencias de los LLMs, ya que permiten el procesamiento paralelo de millones de hilos de ejecución. Esto optimiza tanto el entrenamiento como la ejecución de los modelos, especialmente en tareas que requieren cálculos intensivos [32].

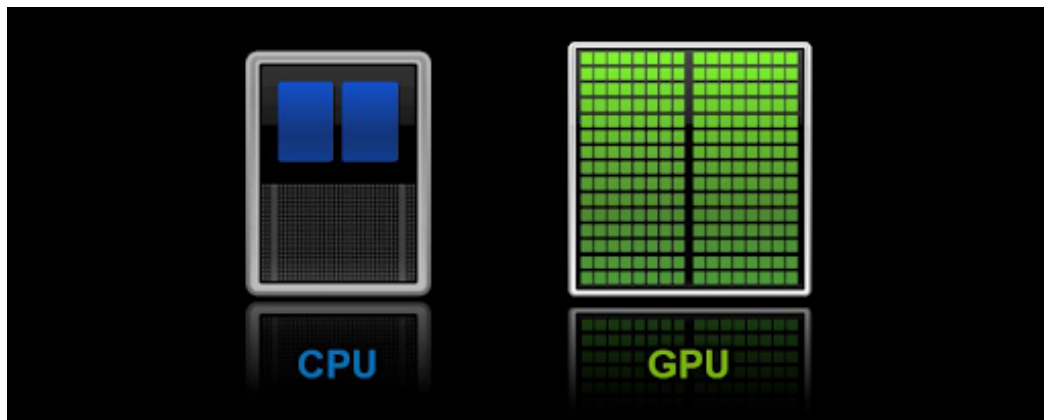


Figura 21: Comparación entre CPU y GPU en el procesamiento paralelo.

Dado que el uso de GPUs dedicadas puede ser costoso y requiere infraestructura especializada, se optó por utilizar Google Colab como entorno de desarrollo. Este servicio proporciona acceso gratuito a GPUs en la nube, lo cual permite ejecutar proyectos complejos sin depender de recursos locales. Además, su entorno basado en Python facilita la integración con bibliotecas como *Transformers*,

simplificando tanto la configuración como la experimentación en tareas de procesamiento de lenguaje natural.

Cuantización y Low-Rank Adaptation (LoRA)

Una vez configurado el ambiente en Google Colab, se procedió a cargar el modelo LLaMA 3.0 utilizando Python y la biblioteca *Transformers*. Para optimizar su uso y permitir su entrenamiento en un entorno con limitaciones de *hardware*, se aplicaron dos técnicas clave: cuantización y *Low-Rank Adaptation* (LoRA).

En primer lugar, el proceso de cuantización consistió en reducir la precisión de los pesos del modelo de 32 bits a 8 bits. Esta reducción permitió disminuir significativamente los requerimientos de memoria y el tiempo necesario para los cálculos de los pesos, haciendo el modelo más eficiente en términos computacionales. Aunque este enfoque sacrifica ligeramente la precisión del modelo, se logró un balance adecuado entre eficiencia y calidad, lo que lo hizo apto para el *fine-tuning* en un entorno con recursos limitados como Google Colab.

Dado que los LLMs contienen una gran cantidad de parámetros, el entrenamiento completo de estos modelos resulta costoso y poco práctico, ya que requiere actualizar todos los pesos de manera intensiva en recursos. *Low-Rank Adaptation* (LoRA) resuelve esta problemática al aplicar matrices de bajo rango en ciertas capas de un modelo pre-entrenado, como se ilustra en la Figura 22. Esta técnica es particularmente efectiva para ajustar LLMs a nuevas tareas sin necesidad de realizar un entrenamiento completo y sin perder el conocimiento preexistente. De esta manera, LoRA mejora la capacidad de adaptación del modelo a nuevas situaciones, manteniendo la eficiencia computacional [68].

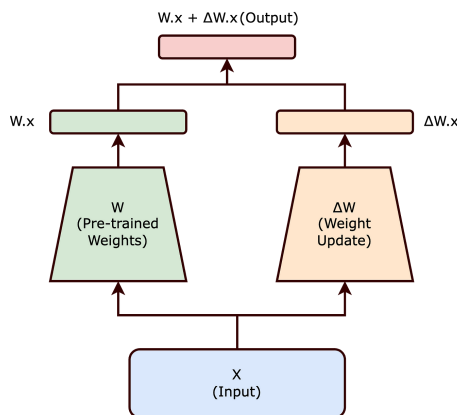


Figura 22: Diagrama de entrenamiento con LoRA.

Los dos parámetros clave en LoRA son el rango y el alpha, los cuales controlan la capacidad del modelo para adaptarse a nuevas tareas sin comprometer demasiado la eficiencia o el rendimiento.

- **Rango:** Define la dimensión del espacio de bajo rango utilizado para la adaptación. Un rango mayor permite una adaptación más precisa, pero requiere más recursos computacionales. Un rango demasiado grande puede generar *overfitting*, mientras que un rango muy pequeño puede no ser suficiente para capturar las características relevantes de los datos.
- **Alpha:** Es el factor de escala que controla la magnitud de la adaptación. Un alpha alto incrementa la magnitud de los ajustes, lo que puede llevar a *overfitting*. Un alpha bajo puede no ser suficiente para adaptar el modelo correctamente a la tarea.

5.3.3. Fine-tuning

Identificación de parámetros óptimos

Para la identificación de los parámetros óptimos para el proceso de *fine-tuning* del modelo LLaMA, se llevaron a cabo múltiples pruebas utilizando un subconjunto reducido del conjunto de entrenamiento. El objetivo principal fue encontrar las configuraciones que produjeran los mejores resultados en términos de la métrica BLEU, así como en pérdida de validación.

Durante las primeras tres pruebas, se utilizó un entrenador genérico, lo cual permitió una mayor flexibilidad en el proceso de entrenamiento. Este enfoque permitió al modelo decidir libremente si entrenar de manera supervisada o no supervisada.

En la primera prueba, se entrenó el modelo sin ningún ajuste específico, con el objetivo de verificar su funcionamiento básico. En la segunda prueba, se aplicó la técnica de *early stopping* para evitar sobreentrenar el modelo y mejorar la generalización, reduciendo el número de *batches* de entrenamiento.

Es importante mencionar que en todas las pruebas se utilizó LoRA. Sin embargo, durante el tercer experimento, se implementó una configuración estándar de LoRA, ajustando los parámetros de rango y alpha a valores de 8 y 16, respectivamente, siguiendo las recomendaciones de un estudio de traducción bilingüe [130]. Estos valores fueron seleccionados para asegurar una adaptación eficiente y controlada sin comprometer la capacidad computacional. En esta tercera prueba también se hizo uso de la técnica de *early stopping*.

En la cuarta prueba, se tomó la decisión de reducir la flexibilidad del modelo al pasar a un enfoque más estricto de *fine-tuning* supervisado. Es decir, se forzó al modelo a entrenar de manera supervisada, eliminando la capacidad de decidir si entrenar de forma supervisada o no. A pesar de este cambio, se mantuvo el estándar de LoRA con los parámetros de rango y alpha en 8 y 16, respectivamente. Además, se continuó utilizando la técnica de *early stopping* para evitar el sobreentrenamiento y mejorar la generalización del modelo.

Fine-tuning del modelo LLaMA (versión 1)

La combinación de parámetros utilizada en la cuarta prueba, que incluyó un enfoque más estricto de *fine-tuning* supervisado junto con los valores de LoRA para el rango y alpha de 8 y 16 respectivamente, presentó los mejores resultados en términos de la métrica BLEU y la pérdida de validación. Debido a estos resultados prometedores, se decidió repetir el proceso de *fine-tuning* utilizando esta configuración, pero con el conjunto de datos completo. Durante este proceso, se monitoreó cuidadosamente la pérdida de entrenamiento y validación para asegurar que el modelo no presentara *overfitting*.

Una vez completado el *fine-tuning*, se procedió a evaluar el rendimiento del modelo. Para ello, se le pidió al modelo que generara interpretaciones para todo el conjunto de prueba, calculando la métrica BLEU para cada par de interpretación generada y esperada, lo que permitió obtener un valor promedio representativo de su precisión. Adicionalmente, se utilizó la técnica LIME para analizar el impacto del *fine-tuning* en las interpretaciones del modelo. Este análisis permitió identificar la relevancia asignada por el modelo a palabras específicas, proporcionando información valiosa sobre cómo el ajuste afectó su comportamiento.

Posteriormente, se solicitó la colaboración de una intérprete experta en LENSEGUA para realizar un análisis cualitativo de las interpretaciones generadas. Este enfoque permitió evaluar la calidad de las interpretaciones desde una perspectiva lingüística y práctica, validando si cumplían con los estándares de calidad esperados. Aunque los resultados obtenidos fueron aceptables, se consideró que

las métricas, como BLEU, podrían mejorar si el *fine-tuning* se realizaba de manera más exhaustiva. Por lo tanto, se decidió repetir la tarea de *fine-tuning* utilizando un conjunto de datos más grande, pero manteniendo la configuración óptima identificada.

Fine-tuning del modelo LLaMA (versión 2)

En esta segunda versión, se decidió utilizar el conjunto de datos desarrollado específicamente para el *fine-tuning* del modelo GPT-3.5-Turbo. Este conjunto estaba compuesto por dos archivos: el archivo de entrenamiento, con 4,192 entradas, y el archivo de validación, con 200 entradas diseñadas para evaluar el rendimiento del modelo durante el ajuste.

Nuevamente, se mantuvo la configuración óptima identificada en las pruebas previas, incluyendo los valores de LoRA para rango y alpha, así como el uso de la técnica de *early stopping*, con el objetivo de evitar el *overfitting* y mejorar la generalización. Sin embargo, el uso de este conjunto de datos, más extenso y diverso, permitió al modelo aprender patrones más complejos y relevantes para las tareas de interpretación de LENSEGUA.

El proceso de *fine-tuning* incluyó un monitoreo constante de las métricas de rendimiento, como la pérdida de entrenamiento y validación. Además, se evaluó el modelo utilizando el conjunto de validación, generando interpretaciones que se compararon con las esperadas mediante dos métricas principales: BLEU, para medir la precisión en las interpretaciones, y la distancia de Levenshtein, para analizar las diferencias estructurales entre las interpretaciones generadas y las esperadas. A partir de estas comparaciones, se calcularon valores promedio que proporcionaron una evaluación detallada del desempeño del modelo.

Finalmente, para profundizar en el análisis, se utilizó LIME para evaluar el impacto del *fine-tuning* en las interpretaciones del modelo, identificando la relevancia asignada a palabras específicas en las entradas.

5.3.4. Comparación de modelos fine-tuneados (GPT-3.5-Turbo y LLaMA)

Tras completar el último proceso de *fine-tuning* del modelo LLaMA, se realizó una comparación con el modelo GPT-3.5-Turbo, también ajustado mediante *fine-tuning*. Dado que ambos modelos fueron entrenados con el mismo conjunto de datos, fue posible llevar a cabo una evaluación consistente de su desempeño. La comparación se llevó a cabo utilizando dos enfoques principales: la distancia promedio de Levenshtein y los análisis realizados mediante LIME, enfocándose en identificar las diferencias clave en la calidad y relevancia de las interpretaciones producidas por ambos modelos.

5.4. Infraestructura de red

5.4.1. Levantamiento de un sistema operativo y configuración de VPN

Para garantizar un ambiente de trabajo seguro y eficiente, se ha implementado Ubuntu 22.08 como sistema operativo en un servidor Dell R740. Se eliminó el controlador RAID para liberar un disco duro de 1.92T, adecuándolo para las necesidades del proyecto. Además, se configuró una VPN utilizando OpenVPN, lo cual permite un acceso directo y seguro a los servidores. La configuración de la VPN se complementó con ajustes en los firewalls institucionales otorgados por la universidad, facilitando el acceso remoto y el manejo del servidor desde múltiples dispositivos sin comprometer

la seguridad. Esta configuración permite una administración centralizada y segura del servidor, asegurando que solo usuarios autorizados puedan acceder a los recursos críticos del sistema.

Como se puede observar en la Figura 23, la máquina virtual fue levantada correctamente. Además, en la Figura 24 se muestra la conexión generada con la VPN mediante SSH.

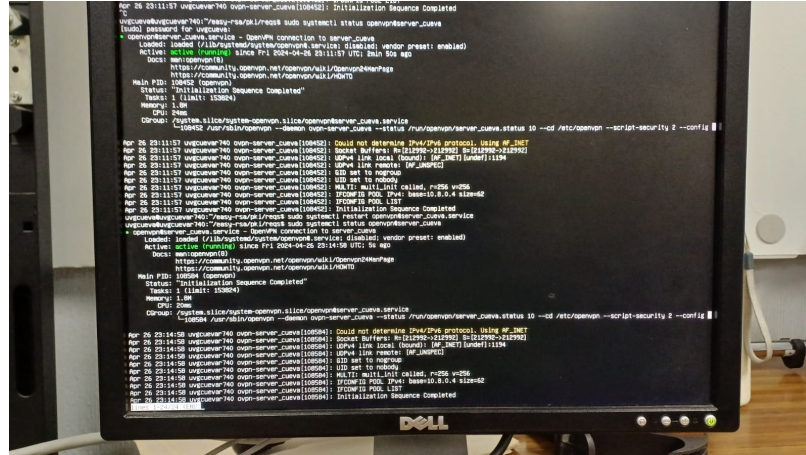


Figura 23: LevantadoOpenVPN.

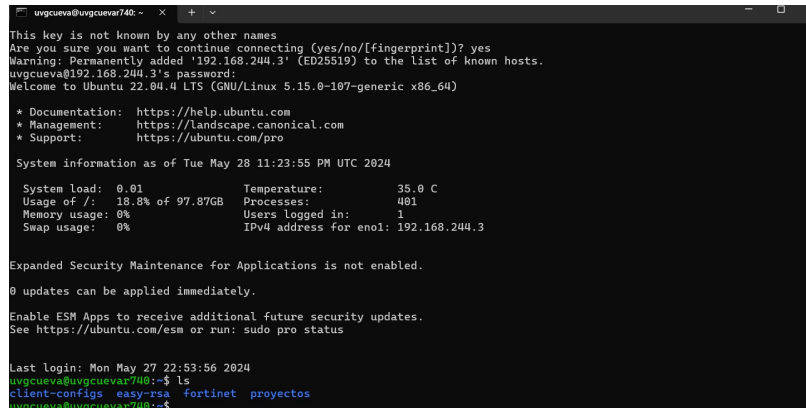


Figura 24: ConexionSSH.

El proceso requirió conectarse a través de la infraestructura de red de la universidad y la VLAN específica donde se encontraba el rack y el servidor. Para lograr esto, se utilizó FortiClient VPN, que proporcionó una conexión segura y cifrada directamente a la VLAN donde se encontraba el servidor. Para acceder a la VPN, se necesitó un archivo de configuración .ovpn que se utilizó con OpenVPN. Una vez conectados a la VPN, se pudo acceder al servidor de manera segura mediante SSH. Esto se realizó abriendo una terminal y utilizando el comando 'ssh uvgcueva@192.168.244.3' con las credenciales correspondientes. Este método aseguró que la administración remota de las máquinas virtuales se realizara de manera eficiente y segura, permitiendo a los usuarios realizar tareas administrativas y de mantenimiento sin estar físicamente presentes en el sitio.

5.4.2. Desarrollo de la arquitectura de base de datos

Se seleccionó PostgreSQL para gestionar la base de datos relacional del proyecto debido a su robustez y eficacia en el manejo de grandes volúmenes de datos, incluyendo datos de usuarios, videos y otros tipos de información. PostgreSQL ofreció avanzadas características de seguridad que fueron esenciales para proteger los datos contra accesos no autorizados. Su arquitectura permitió realizar operaciones complejas de base de datos de manera eficiente, asegurando la integridad y la privacidad de los datos gestionados. En la arquitectura se hizo especial hincapié en la importancia de diseñar con flexibilidad, utilizando estándares que permitieran la interacción entre diferentes sistemas y tecnologías.

Para implementar PostgreSQL en el proyecto de forma efectiva y segura, se siguió una metodología estructurada que abarcó desde la planificación inicial hasta la integración con otros sistemas. Comenzamos evaluando los requisitos de hardware y software para seleccionar la versión óptima de PostgreSQL y preparar el entorno del sistema operativo. Una vez instalado PostgreSQL, se procedió a configurar los parámetros de rendimiento y seguridad, como la memoria, conexiones permitidas y configuraciones de cifrado, además de establecer un protocolo regular de mantenimiento que incluyó actualizaciones y parches.

La seguridad fue una prioridad, implementando roles y permisos detallados para controlar el acceso a los datos y asegurar todas las comunicaciones mediante SSL/TLS. Paralelamente, se desarrollaron y configuraron procesos de ETL para automatizar la extracción, transformación y carga de datos, asegurando la integridad y utilidad de la información gestionada. Además, se diseñaron y desarrollaron APIs que permitieron una interacción segura y eficiente con otras aplicaciones, implementando mecanismos robustos de autenticación y autorización.

Finalmente, se establecieron sistemas de monitoreo para supervisar constantemente el rendimiento y la seguridad de la base de datos y las APIs, permitiendo ajustes continuos para optimizar la configuración según las necesidades cambiantes del proyecto. Este enfoque metodológico aseguró no solo la implementación técnica efectiva de PostgreSQL, sino también el mantenimiento de un sistema seguro, eficiente y adaptable a largo plazo.

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único del usuario.
mail	String(120)	Unique, Not Null	Correo electrónico del usuario.
password	String(120)	Not Null	Contraseña del usuario.
streak	Integer	Default = 0	Puntuación o racha del usuario.
quetzalito	String	Nullable	Color de la imagen de perfil.
confirmed	Boolean	Default=False	Indica si el usuario ha confirmado su cuenta.
last_streak_update	DateTime	Default=datetime.utcnow	Indica la última hora que se agregó un streak.

Tabla 3: Tabla: user

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único del video.
id_user	Integer	ForeignKey(user.id), Not Null	Referencia al usuario que subió el video.
traduction_esp	String(255)	Nullable	Traducción del video al español.
sentence_lensegua	String(255)	Not Null	Oración en lengua de señas guatemalteca.
video	String(255)	Not Null	Ruta del archivo de video.
prev_image	String(255)	Nullable	Imagen previa del video.
is_favorite	Boolean	Default=False	Indica si el video es favorito del usuario.

Tabla 4: Tabla: video

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único de la traducción.
id_user	Integer	ForeignKey(user.id), Not Null	Referencia al usuario que hizo la traducción.
sentence_lensegua	String(255)	Not Null	Oración en lengua de señas guatemalteca.
traduction_esp	String(255)	Nullable	Traducción al español.
is_favorite	Boolean	Default=False	Indica si la traducción es favorita.

Tabla 5: Tabla: traducción

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único del diccionario.
id_user	Integer	ForeignKey(user.id), Not Null	Referencia al usuario.
id_word	String(255)	Not Null	Palabra o identificador del término en el diccionario.

Tabla 6: Tabla: dictionary

El diagrama de entidad-relación presentado en la Figura 25 muestra la estructura y las relaciones entre las tablas diseñadas para el proyecto.

En este diagrama se observan las entidades principales del sistema: **user**, **video**, **traducción** y **dictionary**. Cada una de estas entidades incluye atributos que definen su propósito dentro del sistema. Por ejemplo, la tabla **user** almacena información esencial del usuario como su correo, contraseña y estado de confirmación, mientras que la tabla **video** registra información de los videos asociados a los usuarios, incluyendo traducciones y metadatos adicionales.

Las relaciones entre las entidades destacan cómo interactúan entre sí. Por ejemplo, la tabla `video` está vinculada a la tabla `user` mediante una relación de uno a muchos (1:N), lo que indica que cada usuario puede tener múltiples videos asociados. Asimismo, la tabla `traducción` permite almacenar traducciones específicas realizadas por los usuarios, vinculándose también a través de una relación de 1:N con `user`.

Este diseño garantizó la flexibilidad y escalabilidad del sistema, permitiendo el manejo eficiente de los datos necesarios para cumplir con los objetivos del proyecto.

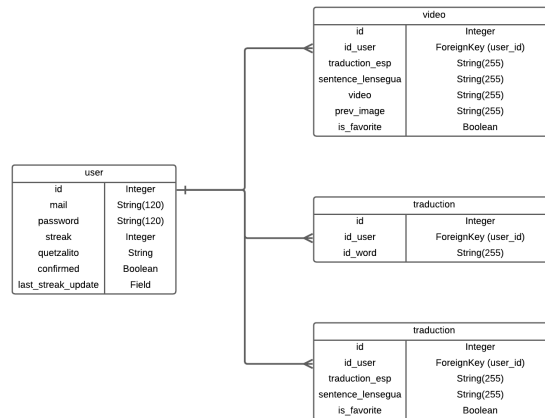


Figura 25: Diagrama entidad relación.

5.4.3. Implementación de APIs

Para facilitar las consultas y las inserciones de datos, se utilizó Flask como framework principal debido a su ligereza y capacidad para manejar peticiones de manera eficiente. Este módulo API permitió un manejo flexible de diferentes formatos de datos, incluyendo videos y texto. La estructura del módulo estuvo diseñada para optimizar el rendimiento del sistema, minimizando el uso de recursos y maximizando la velocidad de respuesta, lo que fue crucial para procesar y responder a las interacciones de los usuarios en tiempo real.

Nombre	TIPO	Entrada	Salida
login	POST	email, password	id_user
signup	POST	email, password, quetzalito	id_user
forgot_password	POST	email	200-OK
change_password	POST	id_user, new_password	id_user, 200-OK

Tabla 7: Resumen de rutas de API: `user_routes`

Nombre	TIPO	Entrada	Salida
send_video	POST	id_user, video	id_video, traduction_lensegua, traduction_esp
report_video	POST	id_user, id_video, report_message, report_img (png)	200-OK
fav_video	POST	id_user, id_video, prev_video (png)	200-OK
remove_video	DELETE	id_video	200-OK
remove_fav_video	POST	id_user, id_video	200-OK
download_video	POST	route_video	ruta_video
download_images	POST	route_image	ruta_image

Tabla 8: Resumen de rutas de API: video_routes

Nombre	TIPO	Entrada	Salida
send_traduction	POST	id_user, sentence_lensegua	id_sentence, traduction_esp
fav_traduction	POST	id_user, id_sentence	200-OK
remove_traduction	DELETE	id_sentence	200-OK
remove_fav_traduction	POST	id_user, id_sentence	200-OK

Tabla 9: Resumen de rutas de API: traduction_routes

Nombre	TIPO	Entrada	Salida
add_dictionary	POST	id_user, id_word	200-OK
remove_dictionary	DELETE	id_user, id_word	200-OK
get_dictionary	POST	id_user	palabras (json)

Tabla 10: Resumen de rutas de API: dictionary_routes

Nombre	TIPO	Entrada	Salida
get_user_info	POST	id_user	email, streak, quetzalito, videos_fav (json), traductions_fav (json)
get_video	POST	id_user, id_video	video (mp4)
get_image	POST	id_user, id_video	image (png)
delete_user	DELETE	id_user	200-OK
add_streak	POST	id_user	200-OK
remove_streak	POST	id_user	200-OK

Tabla 11: Resumen de rutas de API: profile_routes

Nombre	TIPO	Entrada	Salida
confirm	POST	email	200-OK

Tabla 12: Resumen de rutas de API: `mail_routes`

Flujo de trabajo para sistema de APIs

La infraestructura del sistema sigue un flujo bien definido en el manejo de solicitudes y respuestas, diseñado para garantizar un equilibrio entre eficiencia y robustez.

1. **Cliente envía una solicitud:** El punto de partida se da cuando un cliente realiza una solicitud HTTP al servidor, por ejemplo, `mi-ip:4242/example`. Este podría ser un navegador web, una aplicación móvil, o cualquier sistema que consuma las APIs que estamos desarrollando. En este primer punto, la solicitud se envía al servidor en la ruta definida.
2. **Nginx recibe la solicitud:** Nginx actúa como la primera línea de defensa y como un proxy inverso. Es el encargado de recibir todas las solicitudes entrantes y decidir cómo deben ser manejadas. Nginx no solo se encarga de redirigir las solicitudes, sino que también es clave en el manejo de conexiones concurrentes, balanceo de carga y la administración de recursos estáticos si es necesario. Gracias a Nginx, el sistema puede gestionar eficientemente un alto volumen de solicitudes simultáneas, *distribuyéndolas* entre los diferentes trabajadores de Gunicorn.
3. **Nginx reenvía la solicitud a Gunicorn:** Una vez que Nginx recibe la solicitud, la reenvía a Gunicorn utilizando un socket Unix o TCP, según la configuración definida. Gunicorn es un servidor de aplicaciones WSGI diseñado específicamente para ejecutar aplicaciones Flask en entornos de producción. En este paso, Gunicorn toma la solicitud y la pasa al núcleo de la aplicación Flask.
4. **Gunicorn procesa la solicitud:** Al recibir la solicitud de Nginx, Gunicorn la envía a la aplicación Flask, encargada de manejar la lógica de negocio asociada a cada *endpoint*. Por ejemplo, si la solicitud se realizó a la ruta `/example`, Flask ejecutará la función asociada a esa ruta. Esta función puede incluir lógica para consultar una base de datos, realizar cálculos, o incluso comunicarse con otros servicios.
5. **Flask genera y envía la respuesta:** Una vez que Flask ha procesado la solicitud, genera una respuesta adecuada. Esta respuesta puede estar en formato JSON, HTML, o cualquier otro formato según lo requerido por el cliente. Posteriormente, Flask devuelve la respuesta a Gunicorn, que se encargará de llevarla al siguiente paso en el flujo.
6. **Gunicorn reenvía la respuesta a Nginx:** Al recibir la respuesta de Flask, Gunicorn la reenvía a Nginx, que tiene la responsabilidad de empaquetarla y prepararla para ser entregada de vuelta al cliente.
7. **Nginx devuelve la respuesta al cliente:** Finalmente, Nginx envía la respuesta al cliente que hizo la solicitud. El cliente recibe la respuesta HTTP, que puede ser procesada y presentada de acuerdo a sus necesidades. Este paso cierra el ciclo de la solicitud, asegurando que el sistema haya gestionado de manera eficiente tanto la entrada como la salida de información.

Diagrama de flujo de la arquitectura de solicitudes

En esta sección se presenta el diagrama de flujo de la arquitectura del proyecto (Figura 26), el cual ilustra cómo se gestionan las solicitudes en el sistema. Nuestra estructura sigue el principio de "dividir

y conquistar”, separando las múltiples solicitudes en secciones que tienen objetivos específicos y bien definidos.

Las solicitudes se agrupan en diferentes módulos, cada uno manejado por un archivo dedicado:

- **profile_routes.py:** Gestiona operaciones relacionadas con la información del perfil del usuario, como obtener datos personales, videos y traducciones favoritas, y la gestión de la racha (*streak*).
- **user_routes.py:** Maneja operaciones de autenticación y registro, como iniciar sesión, registrarse, y cambiar contraseñas.
- **video_routes.py:** Se ocupa de la carga, gestión y eliminación de videos, así como de marcar o desmarcar favoritos y manejar imágenes de previsualización.
- **traduction_routes.py:** Administra las traducciones de frases en Lensegua al español, permitiendo marcar traducciones como favoritas o eliminarlas.
- **dictionary_routes.py:** Permite agregar, eliminar y consultar palabras en el diccionario de favoritos del usuario.
- **mail_routes.py:** Facilita el envío de correos electrónicos, como en el caso de restablecimiento de contraseñas.

Todos estos módulos giran en torno al uso del archivo **models.py**, el cual actúa como el eje central del sistema. Este archivo define las entidades y modelos utilizados para interactuar con la base de datos, permitiendo que todas las secciones puedan acceder y realizar operaciones en ella de manera eficiente y centralizada.

Esta división modular asegura una arquitectura organizada, fácil de mantener y escalable, donde cada módulo se encarga de tareas específicas, mientras que el modelo central proporciona una única fuente de verdad para la interacción con los datos.

Flujo de trabajo del sistema de APIs:

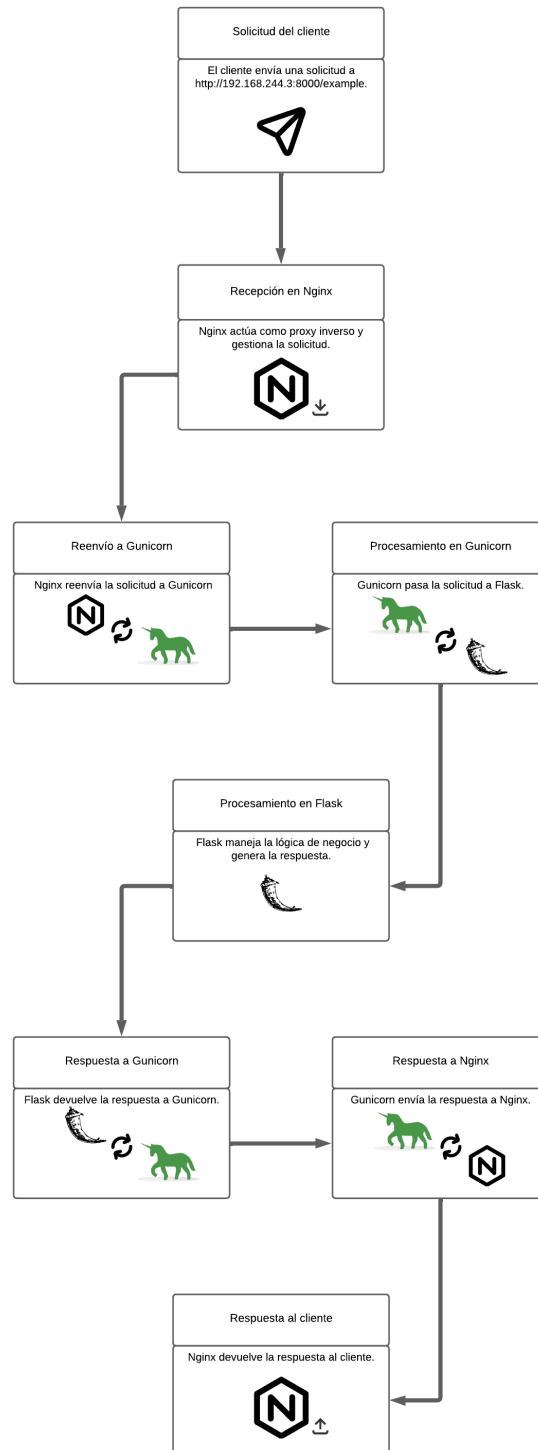


Figura 27: Flujo de trabajo completo del sistema de APIs.

Configuración general

a. Configuración de Flask

La aplicación Flask fue configurada con soporte para migraciones de bases de datos, utilizando SQLAlchemy y Flask-Migrate. Esta configuración permite la manipulación sencilla de esquemas de bases de datos a medida que se añaden nuevas funcionalidades al sistema.

b. Configuración de Gunicorn

- I. Gunicorn se seleccionó como el servidor WSGI debido a su capacidad para manejar múltiples solicitudes concurrentes mediante el uso de workers. Gunicorn se ejecuta vinculándose a todas las interfaces de red en un puerto específico, lo que permite que la aplicación esté disponible en la red local o en internet.
- II. Para gestionar múltiples procesos de Gunicorn, se configuró su ejecución como un servicio del sistema. Esto asegura que Gunicorn inicie automáticamente con el sistema y facilite la gestión del servidor en producción.

c. Configuración de Nginx

- I. Nginx se configuró como un proxy inverso para manejar la entrada de solicitudes del cliente y distribuir las a Gunicorn. Este paso es crucial para manejar eficientemente la concurrencia y asegurar que el tráfico de red no sobrecargue el servidor de aplicaciones.

d. Archivo de configuración de Nginx:

Listing 5.1: Archivo de configuración de Nginx.

```
1 server {
2     listen 4242;
3     server_name 192.168.244.3;
4
5     location / {
6         proxy_pass http://unix:/srv/web-apps/api-central/api-central
7             .sock;
8         include proxy_params;
9         proxy_redirect off;
10    }
```

Explicación del archivo de configuración de Nginx:

- I. **listen 4242**: Especifica el puerto en el que Nginx escuchará las solicitudes.
- II. **server_name 192.168.244.3**: Define la dirección IP del servidor.
- III. **proxy_pass**: Redirige las solicitudes recibidas a Gunicorn mediante un socket Unix, lo que mejora el rendimiento y la seguridad al evitar conexiones TCP adicionales.

- iv. **include proxy_params**: Incluye parámetros comunes para proxies inversos en Nginx.
- v. **proxy_redirect off**: Desactiva la redirección automática de respuestas, permitiendo un control más preciso sobre el flujo de la solicitud y respuesta.

Implementación de SQLAlchemy para manejo de base de datos y entidades

1. Definición de entidades

Las entidades en SQLAlchemy se definen como clases que heredan de `db.Model`. Cada clase representa una tabla en la base de datos, y sus atributos corresponden a las columnas de la tabla. Por ejemplo, una entidad `Usuario` se define para representar la tabla `usuarios` con columnas como `id`, `correo` y `contrasena`. Estas clases se definen en un archivo específico, como `models.py`, para mantener el código organizado y escalable.

2. Relación entre entidades

SQLAlchemy permite definir relaciones entre entidades usando claves foráneas (*ForeignKey*). Por ejemplo, en la entidad `Video`, existe una relación con la entidad `Usuario` mediante la columna `usuario_id`, que referencia al `id` de la tabla `usuarios`. Este tipo de relaciones permiten manejar estructuras de datos más complejas y realizar consultas entre tablas relacionadas de forma sencilla.

3. Migraciones de base de datos

Flask-Migrate se utiliza para aplicar cambios en el esquema de la base de datos de manera controlada. Una vez que las entidades están definidas, Flask-Migrate permite generar y aplicar migraciones para reflejar estos modelos en la base de datos. Este proceso se realiza en tres pasos:

- a. **Inicialización de las migraciones**: Se configura Flask-Migrate para gestionar la base de datos.
- b. **Creación de migraciones**: Cada vez que se añade o modifica una entidad, se genera una nueva migración.
- c. **Aplicación de migraciones**: Las migraciones se aplican a la base de datos, creando o modificando las tablas según sea necesario.

4. Operaciones CRUD

Con SQLAlchemy, las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) se manejan fácilmente mediante el uso de sesiones de base de datos. Estas sesiones permiten realizar consultas, insertar nuevos registros, actualizar datos existentes y eliminar registros, todo de una manera declarativa y fluida.

Implementación de módulo API para bases de datos

Para facilitar las consultas y las inserciones de datos, se utilizará Flask como framework principal debido a su ligereza y capacidad para manejar peticiones de manera eficiente. Este módulo API permitirá un manejo flexible de diferentes formatos de datos, incluyendo videos y texto. La estructura

del módulo está diseñada para optimizar el rendimiento del sistema, minimizando el uso de recursos y maximizando la velocidad de respuesta, lo que es crucial para procesar y responder a las interacciones de los usuarios en tiempo real.

Módulo: Procesamiento de lenguaje

- **Nombre de la API:** `API_Procesamiento_de_Lenguaje`
- **Función de la API:** Procesar texto y generar respuestas basadas en el modelo
- **Entrada:** XML con texto
- **Salida esperada:** Respuesta en XML del modelo GPT-4

Módulo: Llama

- **Nombre de la API:** `API_Llama`
- **Función de la API:** Analizar y responder consultas con el modelo Llama
- **Entrada:** XML con consulta
- **Salida esperada:** Respuesta en XML del modelo Llama

Módulo: Visión por computadora

- **Nombre de la API:** `API_Vision`
- **Función de la API:** Analizar imágenes y videos, y detectar objetos
- **Entrada:** XML con imagen/video
- **Salida esperada:** Respuesta en XML con análisis de visión por computadora

Módulo: HyperVisor

- **Nombre de la API:** `API_HyperVisor`
- **Función de la API:** Gestionar y monitorizar las máquinas virtuales
- **Entrada:** XML con comandos
- **Salida esperada:** Respuesta en XML con estado de VMs

API para integración de modelos de procesamiento

Se desarrolló una API específica para facilitar la interacción entre el módulo de visión por computadora y los modelos de lenguaje natural. Esta API aseguró que la información fluya de manera eficiente desde la recepción de videos hasta la entrega de resultados en texto o voz, pasando por la consulta de bases de datos. La arquitectura de red y del servidor fue diseñada para manejar altas demandas de solicitudes sin degradar la calidad del servicio, utilizando técnicas de balanceo de carga y optimización de recursos para garantizar un proceso robusto y confiable.

Se diseñó un entorno virtualizado utilizando Multipass sobre un servidor con Ubuntu 22.08 ya instalado, lo que facilitó la segregación de los módulos de visión por computadora, inteligencia

artificial con GPT-4 y el modelo de IA Llama. Cada uno de estos módulos operó dentro de su propia máquina virtual, configurada específicamente para satisfacer sus requisitos operativos y de procesamiento.

Para iniciar el proceso, se configuró máquinas virtuales individuales para cada módulo bajo el hypervisor Multipass en Ubuntu 22.08. Esta configuración implica la asignación de los recursos necesarios, como CPU dedicada, memoria RAM adecuada y espacio de almacenamiento suficiente, de acuerdo con las demandas específicas de cada aplicación. Además, se estableció redes virtuales que permitan una comunicación eficaz y segura entre las máquinas virtuales y los sistemas de bases de datos externos.

Implementación de Crontab

Para garantizar que los usuarios mantengan una racha activa y que se reinicie automáticamente si ha pasado más de un día sin actividad, se ha implementado un proceso automatizado utilizando 'crontab' en el servidor Linux. Esta tarea se ejecuta todos los días a las 3:00 a.m. y ejecuta un script de Python que revisa la última hora de actualización de la racha de cada usuario y la reinicia si ha pasado más de 24 horas desde la última actualización.

Configuración del Cron Job

Para configurar la tarea programada que se ejecuta a las 3:00 a.m. todos los días, se debe agregar la siguiente línea al archivo de crontab del usuario adecuado:

```
1 0 3 * * *
```

Listing 5.2: Frecuencia de ejecución

Indica la frecuencia de ejecución (todos los días a las 3:00 a.m.).

```
1 /usr/bin/python3
```

Listing 5.3: Ruta al intérprete de Python

Es la ruta al intérprete de Python.

```
1 /ruta/al/script/reset_streaks.py
```

Listing 5.4: Ruta al script de Python

Es la ruta completa al script de Python que contiene la lógica de reinicio de rachas.

```
1 >> /ruta/al/log/reset_streaks.log 2>&1
```

Listing 5.5: Redirección de salida y errores

Redirige la salida y los errores del script a un archivo de log para facilitar la revisión de su ejecución.

El script de Python que se ejecuta está diseñado para realizar las siguientes tareas:

1. Crear una aplicación Flask y un contexto de aplicación para que el script pueda interactuar con la base de datos.
2. Obtener todos los usuarios de la base de datos y revisar su última hora de actualización de racha.

3. Calcular la diferencia de tiempo entre la hora actual y la última actualización. Si ha pasado más de 24 horas, reiniciar la racha del usuario y actualizar la hora de la última modificación.
4. Registrar la acción en la base de datos y mostrar un mensaje de confirmación en la consola.

El código del script es el siguiente:

```
1 from app import create_app, db
2 from app.models import User
3 from datetime import datetime, timedelta
4
5 // Crear la aplicación Flask y contexto
6 app = create_app()
7 app.app_context().push()
8
9 def reset_streaks():
10     // Obtener todos los usuarios
11     users = User.query.all()
12     for usuario in users:
13         if usuario.last_streak_update:
14             // Calcular la diferencia de tiempo desde la última
15             // actualización
16             delta = datetime.utcnow() - usuario.last_streak_update
17             if delta > timedelta(days=1):
18                 // Ha pasado más de un día, reiniciar el streak
19                 usuario.streak = 0
20                 usuario.last_streak_update = datetime.utcnow()
21                 db.session.commit()
22                 print(f"Streak reset for user {usuario.mail}")
23
24 if __name__ == "__main__":
25     reset_streaks()
```

Listing 5.6: Script de restablecimiento de streaks en Flask

Importancia de la implementación

El uso de 'crontab' para esta tarea es fundamental para mantener la consistencia y la precisión en el control de rachas de los usuarios sin intervención manual. Esto garantiza que las rachas de los usuarios se gestionen de manera automatizada, promoviendo un comportamiento constante y ofreciendo una mejor experiencia de usuario. Además, el uso de 'crontab' permite que la tarea se ejecute en horarios de baja actividad del servidor, minimizando el impacto en el rendimiento del sistema.

5.4.4. Virtualización del servidor para múltiples modelos

Implementación de arquitectura con máquinas virtuales

Para implementar esta arquitectura, se crearon un total de tres máquinas virtuales adicionales al servidor principal que actuará como HOST. Cada máquina virtual es dedicada a uno de los módulos: visión por computadora, inteligencia artificial con GPT-4 y el modelo de IA Llama.

Utilizando Multipass sobre Ubuntu 22.08, cada máquina virtual se configuró meticulosamente para cumplir con los requisitos específicos de procesamiento, asignando CPU dedicadas, memoria

RAM adecuada y espacio de almacenamiento suficiente. Además, se establecerán redes virtuales para asegurar una comunicación eficiente y segura entre las máquinas virtuales y los sistemas de bases de datos. Esta configuración garantizará que cada módulo opere de manera independiente y óptima, permitiendo un procesamiento robusto y confiable bajo altas demandas de solicitudes sin comprometer la calidad del servicio.

Resumen del sistema:

Memoria RAM	Espacio en disco	CPU
Total: 125 GiB	/dev/mapper/ubuntu--vg-ubuntu-lv: 98G total, 19G usado, 75G disponible (20 % uso)	Arquitectura: x86_64
Usada: 50 MiB	/dev/sda2: 2.0G total, 253M usado, 1.6G disponible (14 % uso)	CPUs: 32 (Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz)
Libre: 75 GiB	/dev/sda1: 1.1G total, 6.1M usado, 1.1G disponible (1 % uso)	Núcleos por CPU: 16
Disponibles: 123 GiB		Hilos por núcleo: 2
Swap Total: 8.0 GiB		Virtualización: VT-x
Swap Usada: 0B		

Tabla 13: Resumen del sistema

El uso de técnicas de balanceo de carga y la optimización de recursos serán esenciales para garantizar que el sistema pueda manejar altos volúmenes de solicitudes sin comprometer la calidad del servicio. Cada máquina virtual se configurará con las dependencias y bibliotecas necesarias, garantizando que los módulos de visión por computadora y los modelos de IA operen de manera óptima y confiable dentro de este entorno virtualizado controlado.

Parte de este balanceo implica la distribución de recursos entre las máquinas virtuales. Para esto, se dividieron los recursos entre las máquinas de la siguiente manera:

Máquina (VM1)	Virtual	1	RAM: 32 GiB	CPUs: 8
Máquina (VM2)	Virtual	2	RAM: 32 GiB	CPUs: 8
Máquina (VM3)	Virtual	3	RAM: 32 GiB	CPUs: 8
Host (host de VM)			RAM: 30 GiB (restante)	CPUs: 8 (restante)

Tabla 14: Distribución de recursos entre las máquinas virtuales y el host

```
uvgcueva@uvgcuevar740:~$ multipass list
Name      State      IPv4          Image
VM1       Running   10.47.92.160  Ubuntu 22.04 LTS
VM2       Running   10.47.92.70   Ubuntu 22.04 LTS
VM3       Running   10.47.92.195  Ubuntu 22.04 LTS
```

Figura 28: Estado de las máquinas virtuales dentro del sistema

5.4.5. Implementación de pruebas de eficiencia

1. Instalación de herramientas de monitoreo:

- 1.a. Instalar el agente NGINX Amplify para monitorear el uso de recursos (CPU, memoria, disco y red) en tiempo real.
- 1.b. Adicionalmente, instalar herramientas complementarias como `htop`, `dstat`, `iotop`, `nload` y `sysdig` para monitoreo detallado y comparación de métricas.

2. Preparación del entorno de pruebas:

- 2.a. Desarrollar un script en Python que realice solicitudes a las APIs, utilizando librerías como `requests` para solicitudes HTTP y `psutil` para monitoreo de recursos.
- 2.b. Configurar el sistema de logging para registrar métricas en un archivo de log y enviar datos al agente NGINX Amplify.

3. Monitoreo continuo:

- 3.a. Monitorizar los recursos del sistema en tiempo real utilizando NGINX Amplify y las herramientas complementarias instaladas.
- 3.b. Registrar métricas clave como el uso de CPU, memoria, disco y red durante las pruebas.
- 3.c. Configurar alertas en NGINX Amplify para notificaciones automáticas sobre uso excesivo de recursos.

4. Análisis de resultados:

- 4.a. Revisar los logs generados durante las pruebas para evaluar el tiempo de respuesta y la utilización de recursos.
- 4.b. Utilizar el panel de NGINX Amplify para analizar métricas históricas y tendencias de rendimiento.
- 4.c. Identificar cuellos de botella y áreas de mejora mediante las métricas y gráficos proporcionados por NGINX Amplify.

5. Optimización y repetición de pruebas:

- 5.a. Implementar ajustes necesarios en la configuración del servidor y las aplicaciones basados en los resultados del análisis.
- 5.b. Repetir las pruebas para verificar las mejoras en el rendimiento del sistema.
- 5.c. Continuar el ciclo de pruebas y optimización hasta alcanzar un rendimiento óptimo, utilizando NGINX Amplify para validar las mejoras.

5.4.6. Pruebas de carga

Objetivo

El objetivo de las pruebas de carga es evaluar el rendimiento del sistema bajo condiciones de carga esperadas, simulando múltiples usuarios y solicitudes concurrentes para determinar la capacidad del sistema. Estas pruebas permiten identificar posibles áreas de mejora en la infraestructura y el manejo de recursos.

Procedimiento

1. **Simulación de usuarios:** Se desarrolló un script en Python para crear múltiples usuarios simulados. La función `create_users` crea registros y datos de autenticación de usuario, almacenándolos para reutilización durante la prueba.
2. **Acciones concurrentes:** Una vez autenticados, cada usuario simulado ejecuta una serie de acciones concurrentes usando `ThreadPoolExecutor`:
 - Obtiene su información de perfil (`get_user_info`).
 - Incrementa su racha de actividad (`add_streak`).
 - Realiza operaciones de video y traducción (subida de video, recuperación, marcación como favorito).
3. **Medición de tiempos de respuesta:** La función `send_request` mide el tiempo de respuesta de cada operación y el estado HTTP. Esto permite registrar tiempos de respuesta y determinar si el sistema mantiene la estabilidad bajo carga.

Para evaluar la capacidad de nuestro servidor y entender sus límites de rendimiento, realizamos una serie de pruebas de carga incrementales. Estas pruebas consistieron en simular un número creciente de usuarios concurrentes interactuando con el sistema, comenzando con 50 usuarios y aumentando progresivamente hasta 1100 usuarios. El objetivo de este enfoque es aplicar niveles crecientes de estrés en el servidor, analizando cómo responde bajo diferentes cargas de trabajo.

Cada prueba de carga permite observar el tiempo de respuesta y la estabilidad del sistema a medida que la cantidad de usuarios aumenta, identificando el punto en el que el rendimiento empieza a degradarse significativamente. Esta metodología nos ayuda a determinar la capacidad máxima de usuarios concurrentes que nuestro servidor puede manejar de forma estable antes de experimentar problemas críticos, como demoras excesivas o fallos en el procesamiento de las solicitudes. Además, este proceso de pruebas graduales nos permite identificar las métricas críticas que influyen en el desempeño, tales como la utilización de CPU, la memoria disponible y el tiempo de respuesta de las solicitudes.

Figura 29: Resultados de la prueba de carga con 50 usuarios concurrentes.

Figura 30: Resultados de la prueba de carga con 100 usuarios concurrentes.

Figura 31: Resultados de la prueba de carga con 200 usuarios concurrentes.

Figura 32: Resultados de la prueba de carga con 300 usuarios concurrentes.

Figura 33: Resultados de la prueba de carga con 400 usuarios concurrentes.



Figura 34: Resultados de la prueba de carga con 500 usuarios concurrentes.

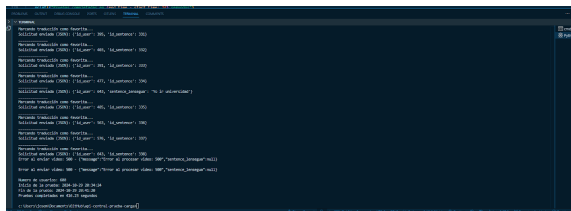


Figura 35: Resultados de la prueba de carga con 600 usuarios concurrentes.

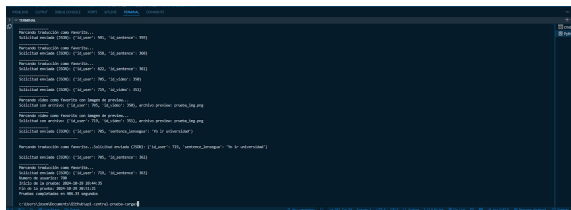


Figura 36: Resultados de la prueba de carga con 700 usuarios concurrentes.



Figura 37: Resultados de la prueba de carga con 800 usuarios concurrentes.

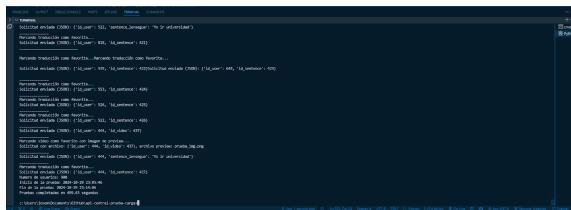


Figura 38: Resultados de la prueba de carga con 900 usuarios concurrentes.

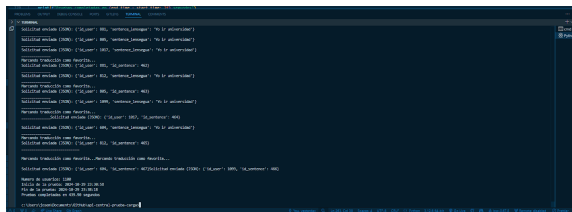


Figura 39: Resultados de la prueba de carga con 1100 usuarios concurrentes.

5.4.7. Pruebas de extremo a extremo

Objetivo

Las pruebas E2E validan el flujo completo de la aplicación desde la perspectiva del usuario, asegurando que todos los módulos y APIs se integren correctamente y que las funciones principales respondan adecuadamente en un flujo de uso continuo.

Procedimiento

1. **Flujo completo del usuario:** Un usuario simulado realiza una serie de acciones que incluyen:
 - Registro y autenticación.
 - Acceso a su perfil, subida de video, marcación y eliminación de favoritos.
 - Interacción con el sistema de traducciones y diccionario.
2. **Verificación y tiempo de respuesta:** Cada acción registra el tiempo de respuesta y verifica que el estado HTTP sea 200 (éxito). Esto garantiza que todas las funcionalidades respondan correctamente en un flujo integrado.

La prueba End-to-End (E2E) se diseñó para evaluar el flujo completo de la aplicación desde la perspectiva del usuario, verificando que todas las APIs respondieran correctamente y que las funcionalidades se integraran de manera óptima.

5.4.8. Implementación de pruebas CVE para seguridad

1. Identificación de vulnerabilidades:

- a) Utilizar herramientas de escaneo de vulnerabilidades como Nessus, OpenVAS o Qualys para identificar posibles vulnerabilidades en el sistema.
- b) Asegurarse de que todas las vulnerabilidades identificadas estén referenciadas con sus respectivos identificadores CVE.

2. Evaluación de vulnerabilidades:

- a) Realizar pruebas exhaustivas de seguridad para obtener los datos necesarios para la calculadora CVSS. Las pruebas incluyen:
 - 1) Pruebas de explotabilidad: Evaluar cuán fácilmente se puede explotar la vulnerabilidad, incluyendo factores como la complejidad del ataque y los privilegios requeridos.
 - 2) Pruebas de impacto: Medir el impacto potencial de la vulnerabilidad en la confidencialidad, integridad y disponibilidad del sistema.

- 3) Pruebas de alcance: Determinar si la vulnerabilidad afecta sólo al componente vulnerable o si se puede propagar a otros componentes.
 - b) Recopilar los datos obtenidos de las pruebas y utilizar una calculadora de puntaje CVE, como la proporcionada por el NVD (National Vulnerability Database), para calcular el puntaje de cada vulnerabilidad basada en los criterios de CVSS (Common Vulnerability Scoring System).
 - c) **Criterios de CVSS:**
 - 1) **Base Score:** Evaluación del impacto y la facilidad de explotación de la vulnerabilidad.
 - 2) **Temporal Score:** Considera factores temporales como la disponibilidad de explotaciones y parches.
 - 3) **Environmental Score:** Ajusta el puntaje base según el entorno y la configuración específicos.
3. **Validación de medidas de seguridad:**
- a) Implementar y validar las medidas de mitigación necesarias para corregir las vulnerabilidades identificadas.
 - b) Realizar pruebas de validación post-mitigación para asegurar que las vulnerabilidades hayan sido corregidas y que las medidas no afecten negativamente el rendimiento del sistema.
4. **Monitoreo continuo:**
- a) Realizar escaneos periódicos de vulnerabilidades para identificar nuevas amenazas y asegurarse de que las vulnerabilidades conocidas estén mitigadas.
 - b) Recalcular los puntajes de CVE periódicamente para asegurar que el sistema se mantenga seguro.
5. **Documentación y reportes:**
- a) Mantener un registro detallado de todas las vulnerabilidades identificadas, sus puntajes CVE y las acciones de mitigación realizadas.
 - b) Generar reportes periódicos que incluyan análisis de tendencias en las vulnerabilidades y la efectividad de las medidas de seguridad implementadas.
6. **Optimización y mejora continua:**
- a) Basado en los análisis de vulnerabilidades y puntajes CVE, implementar ajustes necesarios en la configuración del servidor y las aplicaciones.
 - b) Repetir las pruebas de seguridad para verificar las mejoras en la protección del sistema.
 - c) Continuar el ciclo de pruebas y optimización hasta alcanzar un nivel óptimo de seguridad.

5.4.9. Pruebas de seguridad con Lynis

Objetivo

En este proyecto, se utilizó Lynis para auditar la seguridad del sistema mediante su Índice de Fortalecimiento. Este índice proporcionó una evaluación cuantitativa basada en la implementación de medidas de seguridad y prácticas recomendadas. Para cumplir con los objetivos del sistema, se definió una meta inicial de un puntaje de 40, considerado como la base para garantizar un sistema con medidas de seguridad básicas adecuadas. Este enfoque siguió las recomendaciones de la

documentación de Lynis Hardening Index [84], que establecía que un puntaje de 40 era suficiente para aplicaciones estándar, mientras que resultados más altos reflejaban un sistema más robusto. Según el CIS Benchmark [25], una auditoría con un puntaje superior a 50 indicaba un fortalecimiento razonable para sistemas en entornos no críticos.

Procedimiento

1. **Instalación y ejecución:** Lynis fue instalado y ejecutado con el siguiente comando:

```
sudo lynis audit system
```

Al finalizar, Lynis generó un informe con recomendaciones específicas para mejorar la seguridad del sistema.

2. **Interpretación del Informe:** El informe proporcionado incluyó una puntuación general de seguridad, destacando las áreas de mayor riesgo y ofreciendo recomendaciones detalladas.

5.4.10. Mejoras de seguridad en `/etc/sysctl.conf`

Con base en las recomendaciones de Lynis, se realizaron ajustes en el archivo `/etc/sysctl.conf` para mejorar la seguridad de red del sistema. Estas configuraciones incluyen:

- **Deshabilitar redirecciones ICMP** para prevenir ataques de red.
- **Habilitar SYN Cookies** para mitigar ataques SYN flood.
- **Restringir el uso de `dmesg` para usuarios no privilegiados.**

5.4.11. Monitoreo continuo de seguridad con ClamAV

Instalación y configuración

ClamAV y su servicio de actualización **freshclam** fueron instalados para realizar escaneos de seguridad en el sistema de manera regular. Esto garantiza una capa adicional de protección contra amenazas de malware.

Monitoreo y ajustes adicionales

El sistema fue configurado para realizar escaneos periódicos y actualizar la base de datos de ClamAV. Esto permite mantener el sistema protegido frente a vulnerabilidades potenciales en el software o archivos nuevos.

5.5. Diseño y desarrollo móvil

5.5.1. Investigación de mercado

Investigación y revisión sobre aplicaciones y tecnologías similares

Se investigan aplicaciones con funcionalidades parecidas a la solución propuesta por Señas Chapinas.

Hand Talk Translator

Es una aplicación gratuita para dispositivos Android e iOS diseñada para mejorar la comunicación entre la comunidad sorda y las personas que pueden oír. Esta traduce texto, ya sea en texto o en audio, al lenguaje de señas americanas. Esto se realiza por medio de Hugo, un avatar tridimensional animado por IA, que facilita la traducción y el aprendizaje. Los usuarios tienen la opción de repetir las traducciones, modificar la velocidad de Hugo, guardar y calificar sus traducciones preferidas. También pueden crear mensajes en GIF para compartir y personalizar la apariencia de Hugo en su tienda [47].

■ Funcionalidades destacadas

- Traducción de frases a lengua de señas ASL mostrado a través de animación 3D.
- Opción de compartir por GIF en redes sociales.
- Opción de guardar traducción en favoritos.
- Opción de repetir la traducción y cambiar la velocidad de reproducción.

■ Características de diseño

- Incorpora elementos interactivos y de personalización para mejorar la experiencia del usuario. Esto vuelve la aplicación más atractiva y personal.
- Animación fluida de Hugo, que facilita el seguimiento visual de las señas.
- Interfaz intuitiva y simple que permite una navegación sencilla por las distintas funciones de la aplicación.
- Uso de color naranja que induce calidez, alegría, optimismo y confianza.
- Los íconos y botones son grandes y están claramente etiquetados, lo cual es útil para una rápida identificación de la funcionalidad.
- La interfaz es responsiva.

■ Comentarios de usuarios

- Los usuarios solicitan que la aplicación traduzca palabras y no letra por letra.
- Los usuarios solicitan mayor cantidad de palabras disponibles para traducción a señas.

SLAIT – Real-time Sign Language Translator with AI

Es una aplicación en fase beta que ofrece servicios de traducción de lengua de señas americanas en tiempo real para dispositivos móviles, web, entre otros. Utiliza inteligencia artificial para realizar traducciones en tiempo real, otorgando facilidad de comunicación para comunicación diaria, salud, educación y oficina. Esta aplicación tendrá modalidad de pago, aunque permite ser parte de la aplicación beta sin costo pero con previo análisis del caso por la empresa [122].

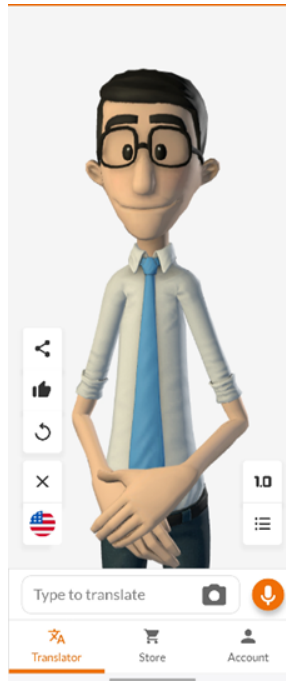


Figura 40: Muestra de aplicación “Hand Talk Translator”

■ **Funcionalidades destacadas**

- Traducción instantánea de voz a texto.
- Traducción instantánea de señas a texto.

■ **Características de diseño**

- Diseño simple y amigable con el usuario.



Figura 41: Muestra de aplicación “SLAIT”

Lenguaje de Señas IA

La aplicación ofrece una plataforma de traducción de señas ASL a texto y viceversa, con múltiples funciones como búsqueda por categoría, emoji y alfabeto, y soporte para diez idiomas. Con más de 2,600 señas reconocidas, busca facilitar la comunicación y el aprendizaje del lenguaje de señas [75].

■ Funcionalidades destacadas

- Grabación de señas ASL para su reconocimiento y traducción a 10 diferentes idiomas.
- Traducción de frases en inglés a señas ASL, seña por seña.
- Búsqueda de señas por emoji, categoría, palabra y alfabéticamente.

■ Características de diseño

- Paleta de colores e iconografía básica.
- Diseño confuso y apretado.

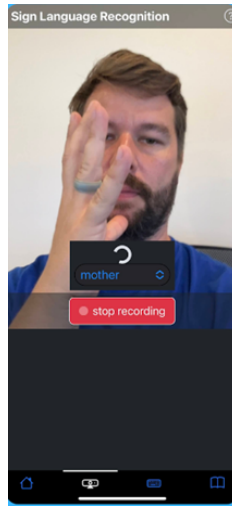


Figura 42: Muestra de aplicación “Lenguaje de señas IA”

AI Sign: Sign Language

Es una aplicación para IOS que utiliza inteligencia artificial para reconocer más de 100 señas americanas. Tiene dos modos: reconocimiento de acciones en tiempo real y captura de datos para mejorar la precisión del modelo de aprendizaje automático [9].

■ Funcionalidades destacadas

- Capacidad de reconocimiento en tiempo real.
- Contiene un modo de ayuda y ajustes de configuración para personalizar la experiencia.

■ Características de diseño

- Muestra de toma de datos de las señas, señalando puntos clave de la mano.
- No hay paleta de colores, se usan elementos gráficos básicos.
- Botones poco amigables y descriptivos.

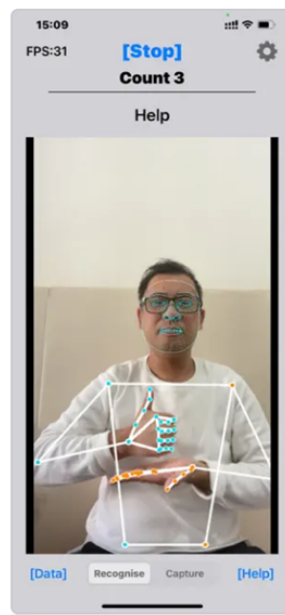


Figura 43: Muestra de aplicación “AI Sign: Sign Language”

Sign Language Translator AI

Es una aplicación móvil diseñada para reconocer y traducir el lenguaje de señas coreano. Utiliza inteligencia artificial para interpretar señas en tiempo real, buscando facilitar la comunicación para las personas sordomudas. La aplicación también fomenta la participación de los usuarios para mejorar su base de datos y aumentar la precisión del reconocimiento de gestos [120].

■ Funcionalidades destacadas

- Guías visuales para el posicionamiento correcto ante la cámara, esenciales para el reconocimiento de gestos.
- Capacidad de reconocimiento en tiempo real.
- Invitación a los usuarios para contribuir con sus propios gestos, ayudando a mejorar la base de datos.
- Lista de palabras reconocibles que sigue expandiéndose con las contribuciones de los usuarios.

■ Características de diseño

- Botones descriptivos.
- Interfaz simple y clara.
- Diseño intuitivo.

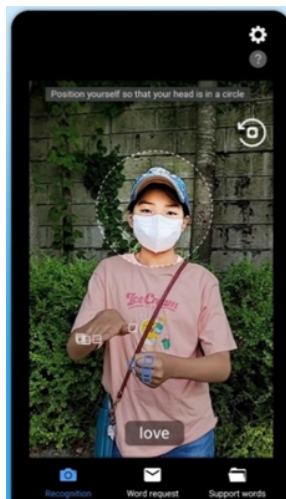


Figura 44: Muestra de aplicación “Sign Language Translator AI”

Resumen de funcionalidades y características destacadas de aplicaciones investigadas

Luego de la investigación previa, se destacan las siguientes funcionalidades:

- Espacio para grabar un video con indicadores de posición y luz adecuados para garantizar una captura óptima del vídeo.
- Modo ayuda para tutorial.
- Botón de compartir para difundir textos traducidos en redes sociales.
- Botón de agregar a favoritos la traducción realizada.
- Botón de reproducción de voz.
- Botón para calificar la traducción.
- Historial de videos grabados.
- Lista de palabras reconocibles.

Asimismo, destacan las siguientes características de diseño:

- Diseño limpio e intuitivo.
- Paleta de colores atractiva.
- Botones descriptivos para mayor comprensión de su función.

Investigación de la situación actual de los sordos en Guatemala

En Guatemala, se estima que hay aproximadamente 240,000 personas sordas, lo que representa el 3% de la población mayor de cuatro años, según datos del Instituto Nacional de Estadística (INE). Las causas de la sordera son diversas e incluyen factores genéticos, complicaciones durante el nacimiento y enfermedades infecciosas. Uno de los principales desafíos que enfrentan las personas

sordas en el país es la comunicación, lo cual limita su capacidad para participar de manera equitativa en la sociedad [30] [50].

La lengua de señas reconocida es LENSEGUA. Las personas sordas están dispersas por todo el territorio nacional, pero se estima que la mayoría que saben LENSEGUA están concentradas principalmente en la ciudad capital. En áreas rurales con acceso limitado a la educación, como el norte de Petén, las personas sordas a menudo desarrollan sus propios sistemas de señas [102].

El ámbito educativo presenta desafíos notables. A pesar de la existencia de organizaciones como la Asociación Nacional de Sordos de Guatemala, que brinda educación y otros servicios, las oportunidades educativas son escasas, especialmente en áreas rurales. Muchos estudiantes sordos deben alejarse de sus familias para acceder a opciones educativas que incluyan instrucción en lengua de señas. Un gran número de ellos ni siquiera tiene la oportunidad de acceder a una educación básica debido a la escasez de maestros capacitados en lengua de señas. Menos del 50% recibe educación formal y las escuelas rara vez ofrecen niveles de educación secundaria. Aquellos que persiguen estudios superiores a menudo lo hacen sin la ayuda de intérpretes, lo que complica su aprendizaje y progreso académico. Esto perpetúa un ciclo de desventajas educativas y económicas, con altas tasas de desempleo y dependencia económica entre la comunidad sorda [6].

El desempleo es elevado en esta comunidad debido a las barreras comunicativas y la falta de adaptaciones adecuadas en los lugares de trabajo. La mayoría de las personas sordas vive con sus padres y están relegadas a trabajos de mano de obra básica debido a las barreras para obtener empleos mejor remunerados [6].

A pesar de estos retos, ha habido iniciativas para mejorar la inclusión de las personas sordas en Guatemala. Por ejemplo, el Benemérito Comité Pro-Ciegos y Sordos de Guatemala, en colaboración con SEGEPLAN, ha trabajado en promover la educación inclusiva y el empleo equitativo, aunque la implementación efectiva de estas políticas aún enfrenta obstáculos significativos [50].

Preparación y realización de entrevistas y encuestas

Entrevista a Doctor Miguel Angel Gonzalez Palacios

El Dr. Gonzalez, médico general con más de 25 años de experiencia, compartió sus reflexiones sobre la importancia de mejorar la comunicación con pacientes sordos. Durante la entrevista, destacó las dificultades que enfrenta en su práctica diaria, particularmente en la correcta comprensión de los síntomas y necesidades de los pacientes sordos, lo cual es fundamental para proporcionar un diagnóstico preciso y un tratamiento efectivo.

El Dr. Gonzalez expresó su entusiasmo por la iniciativa de la aplicación *Señas Chapinas*, mencionando que una herramienta de este tipo podría ser revolucionaria para la práctica médica. Resaltó el potencial de la aplicación para facilitar una comunicación fluida y precisa con pacientes sordos, reduciendo los malentendidos y aumentando la calidad del cuidado médico.

Entrevista a Licenciada Claudia Barrillas

En la entrevista con la abogada Barrillas, se abordó el contexto legal de las personas sordomudas en Guatemala, destacando la importancia de proteger sus derechos en los procedimientos judiciales. Se resaltó la necesidad de contar siempre con un intérprete de lengua de señas durante las declaraciones para asegurar una comunicación efectiva. La escasez de intérpretes, sin embargo, puede provocar demoras en los procesos legales. La Licenciada Barrillas enfatizó el valor de una aplicación de traducción de lengua de señas que podría permitir una comunicación más fluida y directa, reduciendo la dependencia de intermediarios y mejorando el acceso a la justicia para las personas sordomudas,

fomentando la inclusión y la igualdad.

Se mencionaron casos donde la ausencia de intérpretes resultó en injusticias o malentendidos legales, subrayando la importancia de una comunicación clara y efectiva en el ámbito judicial. La profesional propuso que una aplicación de traducción de lengua de señas no solo facilitaría la comunicación en procesos legales, sino que también promovería una mayor autonomía para las personas sordomudas, eliminando muchas barreras que enfrentan cotidianamente.

Entrevista con Profesora Carmen Lucía Guerrero

Carmen Guerrero es profesora en la Universidad del Valle de Guatemala y forma parte del departamento de Educación para personas con necesidades especiales. Su trabajo le ha permitido adquirir conocimientos en LENSEGUA y establecer contacto directo con miembros de la comunidad sorda.

En esta entrevista se abordaron temas clave sobre la estructura y adaptación del español signado en la lengua de señas.

La profesora Guerrero explicó que las personas que nacen sordas generalmente aprenden la lengua de señas como su primer idioma, lo cual posee una gramática y estructura propias, distintas del español hablado. Por otro lado, aquellas que pierden la audición más tarde en la vida pueden intentar adaptar su forma de hablar al español, conservando características del lenguaje oral. Este contraste muestra cómo la forma de comunicación varía significativamente entre quienes han sido sordos desde el nacimiento y quienes se han vuelto sordos posteriormente.

Uno de los desafíos discutidos fue la falta de un estándar unificado en la lengua de señas en Guatemala, lo que lleva al uso de señas específicas en regiones como Quetzaltenango. Esta variabilidad regional complica la comunicación y la educación en lengua de señas. Además, la profesora subrayó la importancia de contar con intérpretes y educadores certificados por el Ministerio de Educación para asegurar utilizar LENSEGUA “oficial”.

También se mencionó que muchas de las señas utilizadas en Guatemala son adaptaciones de la lengua francesa, compartiendo similitudes gramaticales con este idioma. La licenciada recomendó la necesidad de investigar más sobre estas reglas gramaticales y consultar documentación específica que pueda profundizar el entendimiento y la correcta aplicación de la lengua de señas.

La conversación también resaltó la relevancia de la posición y el movimiento de las manos en la comunicación a través de señas, dado que pequeñas variaciones pueden alterar significativamente el significado de las palabras. Por ejemplo, las señas para “hola” y “gracias” son muy parecidas y pueden confundirse fácilmente.

Finalmente, se recalcó que no siempre es posible traducir todas las palabras directamente a señas, lo que destaca la complejidad de desarrollar recursos efectivos para la comunicación en lengua de señas y la importancia de adaptar continuamente las herramientas educativas y de comunicación para satisfacer las necesidades de la comunidad sorda.

Entrevista a intérprete de En-Señas Melany Cordero

Melany Cordero, quien ejerce como intérprete y maestra de nivel medio en En-Señas, acompaña a profesoras al impartir clases y asiste en la resolución de dudas de los alumnos. Comenzó su formación en LENSEGUA en la academia y obtuvo su diploma que la acredita como intérprete.

Durante la entrevista, Melany expresó que la propuesta de la aplicación *Señas Chapinas* le pareció tanto útil como innovadora. Sugirió agregar elementos como juegos o retos diarios, para

fomentar un aprendizaje continuo y efectivo de LENSEGUA entre los usuarios. Propuso, por ejemplo, implementar un juego de memoria o un ejercicio similar a los utilizados en exámenes, donde se presenta una palabra y los usuarios deben seleccionar la seña correcta asociada. Estas actividades no solo mantendrían el interés de los usuarios, sino que también potenciarían su capacidad de aprendizaje y retención de la lengua de señas de manera divertida y desafiante.

Entrevista a Director General de En-Señas Antonio Barrientos

El señor Barrientos desarrolló un interés por la lengua de señas inspirado por su madre, quien también la aprendió y frecuentaba a la comunidad sorda. Motivado por su deseo de entender las conversaciones de este grupo, el señor se sumergió en el estudio de la lengua de señas y actualmente es intérprete de nivel avanzado y director general de la institución “En-Señas”.

Durante la entrevista, el Director señaló que una de las principales complicaciones con la aplicación *Señas Chapinas* es la falta de un estándar uniforme para LENSEGUA. Explicó que existen variaciones significativas en el uso de señas entre los diferentes departamentos de Guatemala, e incluso entre distintas instituciones educativas, lo que puede complicar la precisión de las traducciones. Esta diversidad se debe a la ausencia de una entidad reguladora que estandarice las señas y certifique quiénes están calificados para enseñar LENSEGUA. No obstante, actualmente hay esfuerzos para lograr esta estandarización.

El Director también destacó la escasez de materiales e información en línea sobre LENSEGUA, así como las deficiencias legislativas que, aunque reconocen la lengua de señas, no establecen un marco regulatorio suficiente para su enseñanza y promoción.

En cuanto a la comunidad sorda, mencionó que existen cuatro categorías distintas: personas que utilizan señas caseras y no interactúan con LENSEGUA, aquellas que aprenden a leer los labios, los usuarios de LENSEGUA, y los bilingües, que combinan la lectura de labios con el uso de la lengua de señas.

Finalmente, el Director Barrientos abordó el alto índice de analfabetismo en la comunidad sorda, atribuyéndolo a las diferencias en el nivel de apoyo que reciben desde la infancia. Mientras algunas familias fomentan el aprendizaje de la lengua de señas y la terapia del habla desde temprana edad, otras dejan a las personas sordas sin el soporte necesario para su desarrollo educativo.

Entrevista a persona sorda hipoacúsica y maestra de En-Señas Gabriela Velázquez

La señora Velázquez, una maestra de 59 años de En-Señas y persona sorda hipoacúsica, nació en Guatemala y actualmente enseña LENSEGUA tanto a personas sordas como oyentes. Está casada con una persona sorda profunda.

Durante su infancia, la señora Gabriela relata que era común que las personas sordas fueran obligadas a aprender a vocalizar mediante terapia del habla y lectura de labios, en lugar de aprender LENSEGUA. No fue hasta la edad adulta que aprendió este medio de comunicación, convirtiéndose en bilingüe, lo cual marcó una mejora significativa en su comprensión lectora y en la ampliación de su vocabulario. Aunque la profesora puede leer labios, encuentra este método desafiante y prefiere comunicarse usando LENSEGUA, lo que le resulta más cómodo y eficaz. Ella señala que, para las personas sordas profundas, vocalizar puede ser aún más difícil, por lo que generalmente dependen más de LENSEGUA, lo que a menudo complica su comprensión del español escrito.

Sobre la aplicación *Señas Chapinas*, la profesora Gabriela considera que sería útil para usuarios de LENSEGUA y destacó la importancia de que la aplicación ofrezca tanto la traducción palabra por palabra como la oración completa en español. Esto ayudaría a los usuarios a verificar la traducción

de las señas y facilitaría el aprendizaje de la gramática en español. La profesora Velázquez y sus conocidos frecuentemente utilizan *ChatGPT*, escribiendo en gramática de LENSEGUA para que el sistema lo traduzca al español, lo que les ayuda a confirmar que están escribiendo correctamente.

Ella menciona que *Señas Chapinas* la usaría principalmente en situaciones donde no hay un intérprete presente, como visitas al médico, reuniones con abogados, testimonios en corte, emergencias y reuniones familiares. Menciona que en Estados Unidos existe un servicio de intérpretes que funciona como un *call center*, donde se ofrece traducción a lengua de señas de forma simultánea, y sugiere que la aplicación podría replicar este servicio en Guatemala, ofreciendo traducciones de LENSEGUA a texto o voz.

Además, destacó que la aplicación podría contribuir a reducir el analfabetismo entre los sordos, permitiéndoles aprender al grabar videos en LENSEGUA y ver las traducciones al español.

En el contexto de las aplicaciones móviles, se destacó que las personas sordas utilizan continuamente sus teléfonos, especialmente para acceder a redes sociales y aplicaciones de videollamadas. Estas herramientas les permiten interactuar con amigos, familiares y conocidos de manera más dinámica. Continua relatando que antes de la llegada de los teléfonos inteligentes, la comunicación era notablemente más complicada; sin embargo, hoy en día, la tecnología facilita significativamente el aprendizaje de LENSEGUA en línea y mejora la comunicación general.

La profesora Velázquez prefiere las aplicaciones que requieren poco texto y presentan interfaces de usuario simples y directas, ya que son más fáciles de utilizar. Además, comentó que las aplicaciones lentas y complejas resultan menos atractivas para los ella y sus conocidos.

Para la profesora, un aspecto crucial de la aplicación *Señas Chapinas* es que reconozca las diversas variantes y modismos presentes en LENSEGUA. Este detalle es fundamental para asegurar que la aplicación sea verdaderamente inclusiva y efectiva para todos los usuarios de la lengua de señas guatemalteca, reflejando las diferencias regionales y de estilo que caracterizan su uso cotidiano.

Además, Gabriela Velázquez espera que la aplicación le sirva como una herramienta para enriquecer su vocabulario y gramática en español. Al incorporar funciones que permitan aprender y practicar español, la aplicación no solo serviría para traducir de LENSEGUA a español, sino que también funcionaría como un recurso educativo, apoyando el desarrollo lingüístico integral de sus usuarios.



Figura 45: Entrevista En-Señas

Primera encuesta Señas Chapinas

La primera encuesta para el proyecto *Señas Chapinas* consta de varias secciones diseñadas para recopilar información y opiniones sobre la necesidad de superar las barreras de comunicación en Guatemala a través de una aplicación móvil.

La primera sección recopila datos demográficos como edad, género, lugar de nacimiento, profesión y si el encuestado presenta dificultades auditivas o del habla.

La segunda sección está bifurcada según si los encuestados son sordos o no. Para los que no son sordos, se investiga si conocen a alguien con dificultades auditivas y cómo han utilizado tecnología de asistencia para comunicarse con ellos. Además, se indaga sobre los desafíos percibidos para estas personas y en qué contextos una aplicación de traducción de señas sería beneficiosa. Para los encuestados sordos, se solicita que compartan sus experiencias con otras herramientas de asistencia, desafíos específicos enfrentados en Guatemala, situaciones de frustración al comunicarse, y cómo una aplicación podría mejorar su comunicación diaria.

La tercera sección evalúa la percepción sobre la utilidad de la aplicación para convertir la lengua de señas a texto o voz, recogiendo expectativas y características deseadas. Además, se piden sugerencias sobre funcionalidades adicionales y posibles contextos de uso.

Los resultados de esta primera encuesta no fueron los esperados. Los participantes sordos de nacimiento encontraron las preguntas demasiado complejas, atribuyéndolo a las diferencias gramaticales en su lenguaje. Esto llevó a la necesidad de desarrollar una encuesta específica para personas oyentes y planificar entrevistas con intérpretes para personas sordas.

Segunda encuesta Señas Chapinas

Como resultado de los comentarios y sugerencias recopilados en la primera encuesta, se desarrolló una segunda encuesta dirigida específicamente a personas oyentes (ver Anexo [11](#)). Esta encuesta se dirige a individuos sin conocimiento previo de LENSEGUA, a aquellos que regularmente interactúan con personas sordas, y a intérpretes.

La sección inicial de la encuesta proporciona un análisis demográfico de los participantes. Los resultados muestran que la mayoría de los encuestados son hombres. Además, el grupo de edad más representado está entre los 18 y 24 años.

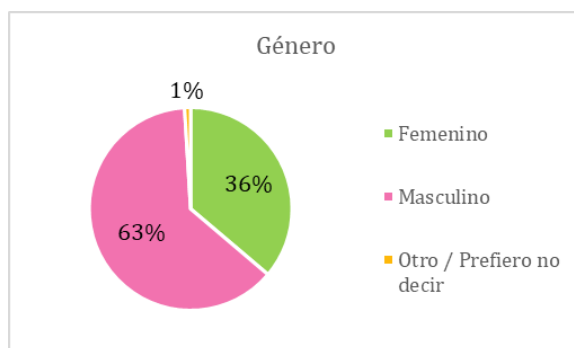


Figura 46: Género encuesta 2

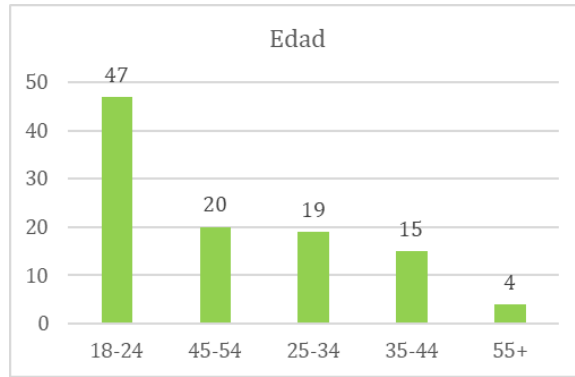


Figura 47: Edad encuesta 2

La siguiente sección de la encuesta está destinada a explorar el conocimiento y la experiencia con la lengua de señas. Alrededor del 40 % de los encuestados indica conocer a alguien sordo, lo que destaca una conexión significativa con la comunidad sorda. Adicionalmente, cerca del 70 % percibe la aplicación como relevante, mostrando un interés considerable en la herramienta propuesta. Sin embargo, es notable que aproximadamente el 70 % de los participantes no están familiarizados con LENSEGUA, lo cual es entendible considerando que su reconocimiento oficial data de hace solo cuatro años. [56].

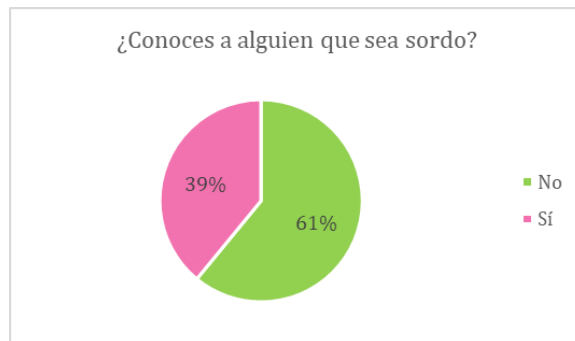


Figura 48: Conocimiento persona sorda encuesta 2

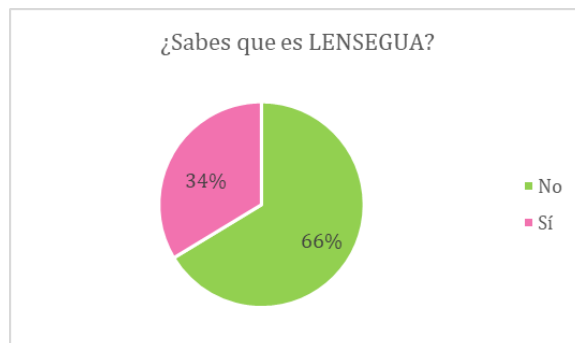


Figura 49: Conocimiento LENSEGUA encuesta 2

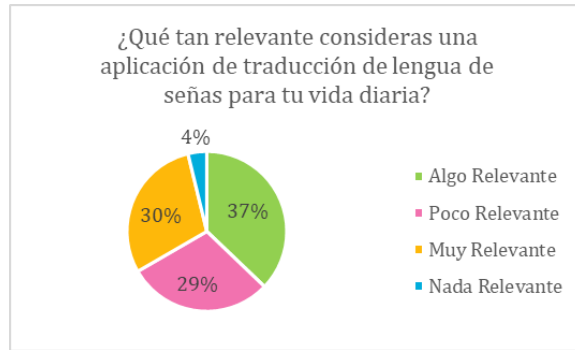


Figura 50: Relevancia de la aplicación encuesta 2

Además, se realizó un análisis comparativo entre los encuestados que conocen a personas sordas y su percepción de la relevancia de la aplicación. Los resultados muestran que aquellos familiarizados con la comunidad sorda tienden a valorar más la aplicación en comparación con quienes no tienen contacto directo con personas sordas. Este hallazgo sugiere que la experiencia personal y el conocimiento de los retos enfrentados por las personas sordas pueden influir significativamente en la percepción de la utilidad de herramientas tecnológicas como un traductor de lengua de señas.

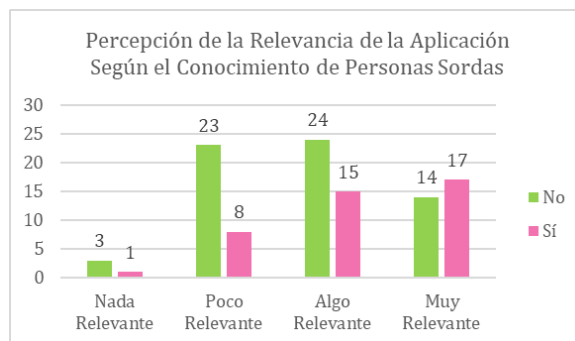


Figura 51: Relevancia de la aplicación para personas con conocidos sordos encuesta 2

Para comprender mejor las estrategias de comunicación que las personas emplearían con individuos sordos, se les preguntó a los encuestados sobre sus métodos preferidos. La mayoría, casi el 60 %, optaría por el uso de mensajes escritos, mientras que un 18 % indicó que señalarían objetos para hacerse entender. Solo un 9 % consideraría usar LENSEGUA. Es notable que solo una minoría, aproximadamente el 10 %, expresó no saber cómo comunicarse, reflejando así el interés general por explorar formas de interacción. Algunos participantes también mencionaron la lectura de labios como alternativa, aunque es importante destacar que no todos los sordos tienen la capacidad de leer los labios.

Para profundizar en el entendimiento que tienen las personas sobre los desafíos que enfrentan los sordos, se les preguntó cuáles consideraban que eran las mayores dificultades para esta comunidad. Las respuestas revelaron una conciencia sobre la falta de inclusión y herramientas adecuadas para personas sordas, destacando cómo muchas infraestructuras y servicios en Guatemala están diseñados principalmente para oyentes. Además, se mencionaron problemas como segregación y discriminación en la sociedad, así como la escasez de enseñanza de la lengua de señas.

Entre las preocupaciones más citadas también estuvieron la ausencia de sistemas de alerta para sordos en situaciones de emergencia y barreras significativas en comunicación. Los encuestados resaltaron la existencia de prejuicios y la dificultad en la realización de tareas cotidianas y trámi-

tes, lo que contribuye al aislamiento social y a la exclusión. Esta variedad de respuestas ilustra la complejidad de los desafíos a los que se enfrentan los sordos.

La siguiente sección de la encuesta se centró en la aplicación *Señas Chapinas*, explorando las motivaciones principales para usar una herramienta de traducción de lengua de señas. Destacablemente, el 60% de los participantes indicaron la curiosidad personal como su principal motivación. Casi la mitad de los encuestados mencionaron la comunicación con amigos o familiares sordos como un factor importante, subrayando la relevancia personal y social de la aplicación. Además, cerca del 40% expresaron que utilizarían la aplicación para actividades voluntarias, lo que refleja su potencial utilidad en entornos de servicio comunitario. Un pequeño porcentaje citó los requerimientos laborales como motivo, sugiriendo su aplicación en contextos profesionales donde la interacción con personas sordas es frecuente. Otras respuestas revelaron usos más específicos y personales, evidenciando la diversidad de situaciones en las que los usuarios anticipan la utilidad de la aplicación.

También se indagó en que situaciones los usuarios desean emplear la aplicación *Señas Chapinas*. Predominantemente, las actividades sociales representan el escenario más popular, con un 78.6% de los encuestados seleccionándolo. Esto es seguido por el voluntariado y la educación, con un 67.6% y un 49%, respectivamente. El trabajo también es una situación comúnmente identificada, con un 40.2% de los encuestados expresando la necesidad de utilizar la aplicación en este contexto. Estos resultados indican una fuerte preferencia por utilizar la aplicación en contextos grupales y de interacción, lo que subraya la importancia de la aplicación en facilitar la comunicación en una variedad de entornos cotidianos y profesionales.

La última sección pregunta las características más importantes en una aplicación móvil. En la última sección de la encuesta, los encuestados identificaron las características más importantes en una aplicación móvil. La facilidad de uso fue la más destacada, valorada por el 98.1% de los participantes, seguida por la velocidad y rendimiento (61.9%), funciones de accesibilidad (54.3%), y diseño atractivo (46.7%). Esto resalta la importancia de una interfaz intuitiva, un rendimiento eficiente, accesibilidad adecuada, y un diseño visualmente atractivo para los usuarios de esta aplicación.

Finalmente se dio espacio para comentarios adicionales. En la sección final de la encuesta se exploraron las características esenciales para una aplicación de traducción de lengua de señas. La facilidad de uso fue una de las características más mencionadas, destacando su importancia para una adopción rápida por parte de los usuarios. Muchos encuestados valoraron también la velocidad y la precisión de la traducción, subrayando la necesidad de interacciones fluidas y sin errores. Algunas otras sugerencias fueron mencionadas, pero serán tomadas en cuenta para futuras mejoras, pues no están dentro del alcance del presente proyecto.

Entrevista colectiva a miembros de En-Señas

Con un conocimiento más profundo de las necesidades de las personas sordas, sus familias y conocidos, así como del nivel de conciencia de la población guatemalteca sobre LENSEGUA, se llevó a cabo una última entrevista colectiva con miembros de Enseñas. Durante esta sesión, se presentó el concepto de la aplicación y se recogieron sus opiniones. Los comentarios más destacados incluyeron:

- **Uso de redes sociales:** Las personas sordas frecuentan plataformas sociales, especialmente TikTok, por su facilidad e intuitividad de uso.
- **Funcionalidades educativas:** Se sugirió que la aplicación también debería facilitar el aprendizaje de LENSEGUA en Guatemala, incorporando juegos interactivos que fomenten la participación activa.
- **Diseño atractivo y profesional:** La aplicación debe ser visualmente llamativa sin sacrificar su seriedad, generando confianza en que se trata de una herramienta fiable y efectiva.

- **Futuras Expansiones:** Se anticipa que, eventualmente, la aplicación podrá traducir del español a LENSEGUA.
- **Vocabulario Adecuado:** En las primeras fases, el vocabulario debe ser simple y cotidiano, incluyendo términos de uso frecuente y frases de emergencia.
- **Reporte de errores:** Es crucial que las traducciones incorrectas puedan ser reportadas fácilmente por los usuarios sordos para que el equipo detrás de la aplicación pueda investigar y resolver cualquier inconveniente.
- **Guardado de favoritos:** Los usuarios expresaron el deseo de poder guardar videos de frases en LENSEGUA que utilizan regularmente, para acceder a ellos de manera rápida y sencilla.
- **Flexibilidad de grabación:** La aplicación debe permitir grabaciones tanto con la cámara frontal como con la trasera, facilitando la captura de auto-grabaciones o de terceros.
- **Accesibilidad del nombre:** El nombre de la aplicación debe ser fácilmente representable en señas, asegurando su accesibilidad y reconocimiento dentro de la comunidad sorda.

Estos valiosos comentarios guiarán la próxima fase de desarrollo para asegurar que la aplicación no solo cumpla con las expectativas de la comunidad sorda sino que también sirva como un puente cultural y educativo.

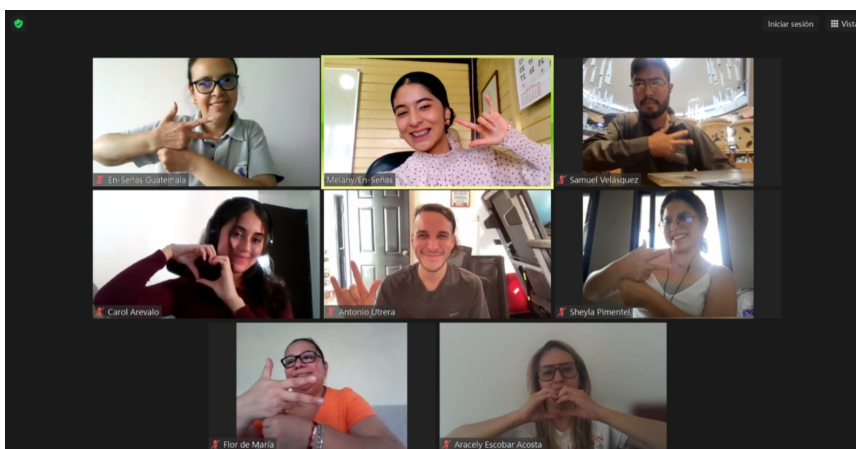


Figura 52: Entrevista colectiva En-Señas

Colaboraciones

Para profundizar en la comprensión de las necesidades tanto de las personas sordas como de aquellas que desean comunicarse con ellas, se decidió colaborar con varias entidades guatemaltecas dedicadas a mejorar la calidad de vida de este grupo.

En primer lugar, se participó en clases de Lengua de Señas Guatemalteca (LENSEGUA) ofrecidas por En-Señas. Esta formación permitió comprender mejor la gramática y el contexto cultural de esta lengua, elementos fundamentales para garantizar una comunicación efectiva y respetuosa.

Asimismo, la empresa ASEDES colaboró proporcionando apoyo en la realización de entrevistas y en otras actividades necesarias para el desarrollo de los diversos módulos del proyecto. Esta colaboración fue vital para asegurar que el diseño de la aplicación sea inclusivo y práctico para los usuarios.

Finalmente, la ONG Sordos Latinos de Guatemala estuvo brindando asistencia invaluable mediante el suministro de información y recomendaciones especializadas. Además, facilito entrevistas y otros recursos esenciales que enriquecieron el entendimiento y ayudaron a ajustar el proyecto para entender mejor las necesidades de la comunidad sorda en Guatemala.

5.5.2. Desarrollo de interfaz y experiencia de usuario

Creación de diagrama de afinidad

En el desarrollo de la experiencia del usuario (UX), los diagramas de afinidad juegan un papel crucial, ya que permiten organizar y sintetizar grandes volúmenes de datos e ideas de manera visual y estructurada. Este método es especialmente valioso durante las fases iniciales de desarrollo de un proyecto, donde el entendimiento claro y la definición del problema son esenciales [80].

El proceso comienza con una lluvia de ideas, donde se generan y recopilan múltiples puntos de vista y datos sobre las necesidades y problemas de los usuarios. Esta fase es crítica, ya que establece la base de información que influirá en todas las decisiones de diseño y desarrollo subsiguientes [80].

La lluvia de ideas para Señas Chapinas organiza conceptos clave en categorías codificadas por colores, facilitando la identificación y el análisis de diversas áreas del proyecto:

- **Naranja:** Define el público objetivo de la aplicación.
- **Amarillo:** Enumera los objetivos y metas que la aplicación pretende alcanzar.
- **Azul:** Destaca información crucial necesaria para el desarrollo de la aplicación.
- **Rojo:** Señala los problemas y desafíos que la aplicación busca resolver.
- **Verde:** Detalla las características y funcionalidades esperadas de la aplicación.

Este enfoque visual no solo ayuda a estructurar el proceso de planificación, sino que también asegura que todos los aspectos relevantes sean considerados, apoyando una toma de decisiones informada y alineada con las necesidades de los usuarios.

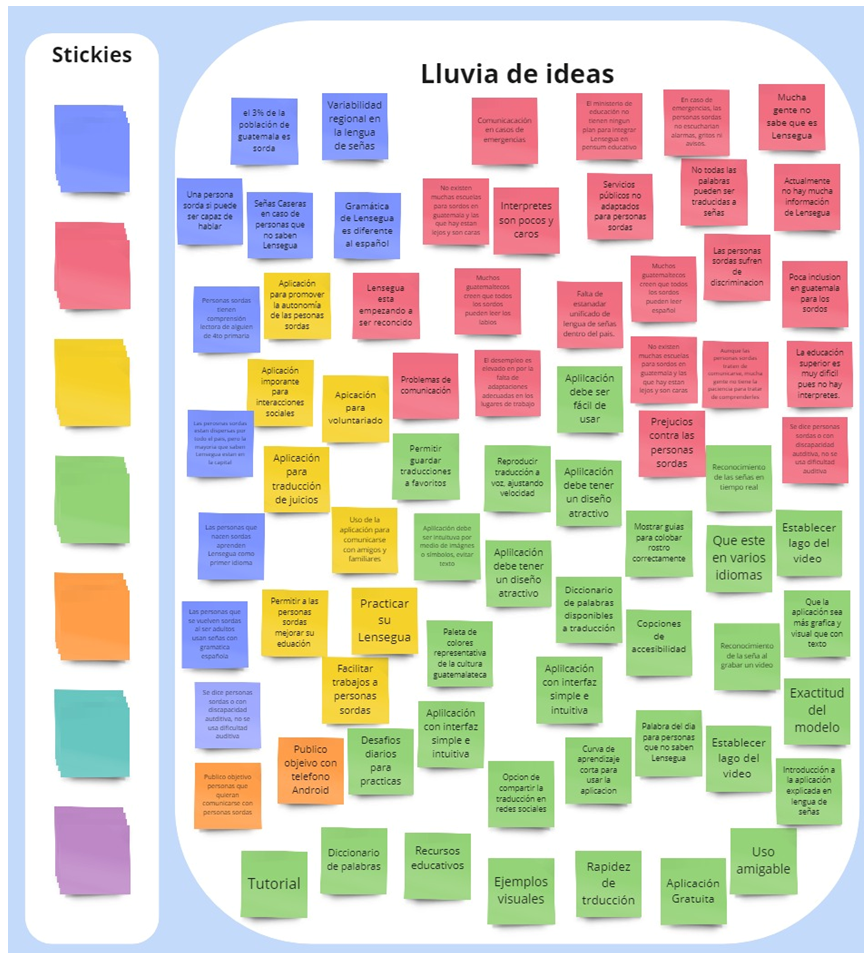


Figura 53: Lluvia de ideas para diagrama de afinidad

Posteriormente, se agrupan las ideas similares con el objetivo de identificar patrones y temas comunes. Este proceso implica organizar los datos recolectados durante la lluvia de ideas en categorías que reflejen conexiones y tendencias subyacentes. Al hacer esto, se pueden observar relaciones entre las diferentes opiniones y necesidades, facilitando la creación de soluciones más coherentes y efectivas que aborden los desafíos identificados de manera integral. Este método no solo ayuda a clarificar el alcance del proyecto, sino que también proporciona una base sólida para las decisiones de diseño y desarrollo subsiguientes, asegurando que se consideren todas las perspectivas relevantes.



Figura 54: Agrupación de ideas para diagrama de afinidad

Finalmente, se realiza el diagrama de afinidad, que sintetiza todas las ideas recolectadas y categorizadas previamente. Este diagrama visual, organizado por colores, facilita la interpretación de la información y permite una evaluación clara de cómo cada aspecto del proyecto interacciona y contribuye al objetivo global. Al agrupar las ideas en distintas categorías, como Público objetivo, Problemas a resolver, Objetivos, Escenarios de uso, Funcionalidades a desarrollar y Características de la aplicación, se destaca la interconexión entre los requisitos del usuario y las soluciones propuestas.

Diagrama de afinidad



Figura 55: Diagrama de afinidad

1. **Público objetivo** Define a las personas que se beneficiarán directamente de la aplicación, incluyendo a aquellos que interactúan regularmente con la comunidad sorda y a las personas sordas que dominan LENSEGUA.
 - Personas que interactúan regularmente con la comunidad sorda
 - Personas sordas que dominan LENSEGUA
 - Individuos residentes en la ciudad de Guatemala con dispositivos Android
2. **Problema a resolver** Identifica los principales desafíos que enfrenta la comunidad sorda y que la aplicación busca abordar, como las dificultades de comunicación entre sordos y oyentes, la discriminación y los altos costos de los servicios de interpretación.
 - Dificultades de comunicación entre individuos sordos y oyentes
 - Discriminación y falta de inclusión de la comunidad sorda
 - Costos elevados y escasez de intérpretes de LENSEGUA
3. **Funcionalidades a desarrollar** Describe las características específicas que tendrá la aplicación.
 - Diccionario de palabras disponibles en LENSEGUA
 - Función diaria de práctica de LENSEGUA para oyentes
 - Capacidad para guardar y compartir traducciones
 - Herramienta para reportar errores en traducciones
 - Opción para reproducir traducciones en español
 - Grabación de video de un largo máximo para su traducción de LENSEGUA a español (no en tiempo real)
4. **Objetivos** Detalla los objetivos principales de la aplicación, como promover la inclusión social, mejorar el acceso a la educación y empleo para personas sordas, y simplificar la comunicación entre sordos y oyentes.
 - Promover la inclusión social
 - Simplificar la comunicación entre personas sordas y oyentes
 - Mejorar el acceso a educación y empleo para personas sordas
 - Fomentar la autonomía personal de los individuos sordos
 - Incrementar la difusión y el uso de LENSEGUA
5. **Escenarios de uso** Enumera los diferentes contextos en los que la aplicación podría ser utilizada, incluyendo interacciones sociales, ambientes laborales, y servicios públicos como trámites médicos, bancarios y policiales.
 - Interacciones sociales y familiares con personas sordas
 - Ambientes laborales con empleados sordos
 - Servicios y trámites públicos como judiciales, médicos, bancarios y policiales
 - Oportunidades de voluntariado y colaboración comunitaria
6. **Características de la aplicación** Resalta aspectos del diseño y la usabilidad de la aplicación, tales como su interfaz intuitiva, el diseño visual con poca dependencia de texto y adaptaciones para accesibilidad, entre otros.
 - Diseño visualmente atractivo con poca dependencia de texto
 - Interfaz intuitiva y fácil de navegar
 - Aplicación con identidad guatemalteca
 - Guías visuales para la correcta colocación del rostro durante la grabación
 - Personalización del perfil de usuario

Creación de personas

Como parte del proceso de diseño, se han definido seis personas que representan a los usuarios finales a quienes se dirige la aplicación.



Figura 56: Persona 1 - Laura



Figura 57: Persona 2 - Ricardo



Figura 58: Persona 3 - Felipe



Figura 59: Persona 4 - Sofía



Figura 60: Persona 5 - Marta



Figura 61: Persona 6 - Jorge

Creación de mapas de empatía

Complementando las personas creadas con anterioridad, se procede a crear su Mapa de empatía respectivo para profundizar en las necesidades de los usuarios.

Mapa De Empatía

Nombre: Laura



Figura 62: Mapa de empatía - Laura

Mapa De Empatía

Nombre: Ricardo



Figura 63: Mapa de empatía - Ricardo

Mapa De Empatía

Nombre: Felipe



Figura 64: Mapa de empatía - Felipe

Mapa De Empatía

Nombre: Sofia



Figura 65: Mapa de empatía - Sofia

Mapa De Empatía

Nombre: Marta



Figura 66: Mapa de empatía - Marta

Mapa De Empatía

Nombre: Jorge



Figura 67: Mapa de empatía - Jorge

Planteamiento del problema

Inicialmente se usa los seis sombreros para pensar, una técnica de pensamiento desarrollada por Edward de Bono en los años 80, que busca facilitar de manera creativa la resolución y el análisis de problemas desde distintos puntos de vista o perspectivas. Cada “sombrero” representa una dirección diferente del pensamiento y se identifica con un color específico [11].



Figura 68: Sombreros para pensar

Posteriormente, se sigue el modelo W5H1 para realizar el planteamiento del problema, el cual busca ver las ideas desde varias perspectivas con el objetivo de comprender en profundidad una situación concreta [18].



Figura 69: Planteamiento del problema Señas Chapinas

1. ¿Quién?

a) ¿A quién afecta el problema?

Afecta principalmente a personas sordas que utilizan la Lengua de Señas Guatemalteca (LENSEGUA), quienes enfrentan barreras de comunicación cotidianas.

b) ¿Quiénes son los usuarios primarios y secundarios?

- **Usuarios primarios:** Personas sordas, quienes dependen directamente de la comunicación efectiva para su inclusión social y profesional.
- **Usuarios secundarios:** Familiares, amigos, intérpretes de LENSEGUA, y personal de servicios públicos y privados que interactúan regularmente con personas sordas.

2. ¿Qué?

a) ¿Cuáles son los límites del problema?

El problema está limitado al vocabulario esencial y de emergencia necesario para la comunicación diaria y situaciones críticas.

b) ¿Cuál es el problema que requiere nuestra atención?

La barrera de comunicación persistente entre las personas sordas y la sociedad oyente, que limita significativamente la participación de las personas sordas en la sociedad.

c) ¿Cuál es el objetivo final?

Facilitar la comunicación y mejorar la inclusión de la comunidad sorda en todos los aspectos de la vida social y profesional.

3. ¿Cuándo?

a) ¿Cuándo ocurre el problema?

El problema ocurre diariamente y se manifiesta en interacciones rutinarias, servicios de emergencia, entornos educativos y actividades sociales.

4. ¿Dónde?

a) ¿Dónde ocurre el problema?

El problema ocurre a lo largo de Guatemala, con un enfoque inicial en el departamento de Guatemala, debido a las variaciones dialectales de LENSEGUA en diferentes regiones.

b) ¿Dónde se necesita enfocar más?

El enfoque inicial será en áreas urbanas donde la densidad de población y la diversidad de servicios intensifican las necesidades de comunicación efectiva.

5. ¿Por qué?

a) ¿Por qué es importante arreglar el problema?

Es importante abordar este problema para garantizar que las personas sordas en Guatemala tengan igualdad de oportunidades en su integración y participación en todos los aspectos de la vida social y profesional. Mejorar la comunicación no solo incrementa la autonomía y el bienestar de las personas sordas, sino que también contribuye a una sociedad más inclusiva y justa, donde todos los ciudadanos pueden contribuir plenamente y sin barreras.

Creación de mapas de experiencia del cliente

El objetivo de estos mapas es entender y abordar las necesidades y los problemas del cliente en cada etapa del proceso, identificando oportunidades para mejorar la experiencia del cliente y asegurando que cada punto de contacto con el producto sea positivo y coherente. Esto permite ver dónde se encuentran los puntos de dolor y adoptar mejoras para ofrecer una experiencia más satisfactoria y efectiva [58].

Los mapas realizados incluyen los flujos principales a desarrollar dentro de la aplicación:

- Primera vez usando la aplicación como usuario, para identificar puntos de mejora con el primer contacto con los usuarios.

Primera vez usando la app

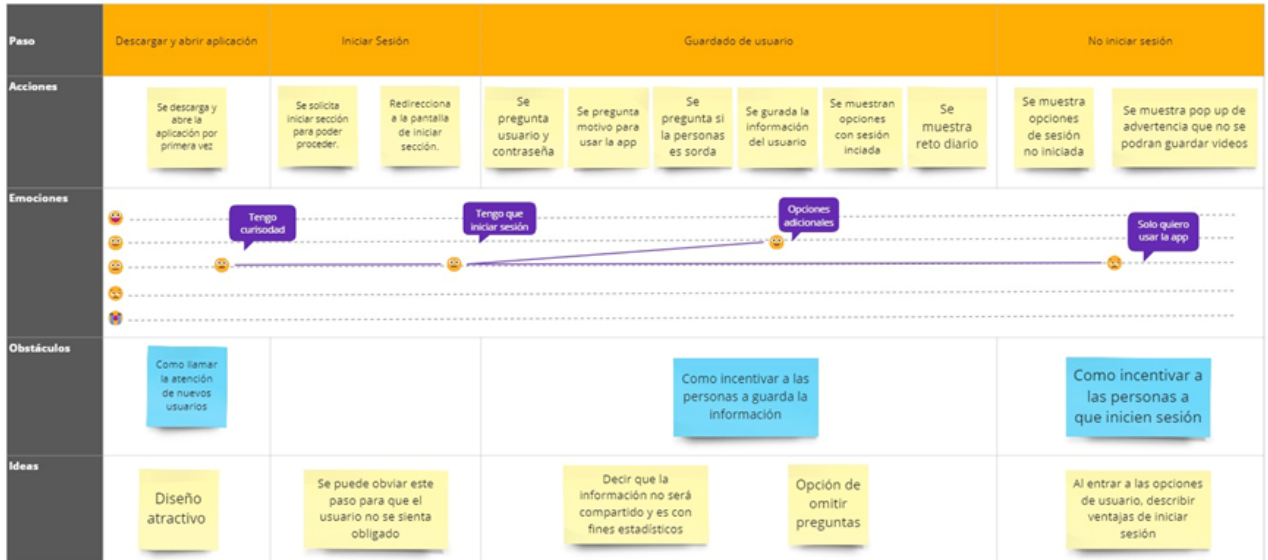


Figura 70: Primera vez usando la aplicación

- Grabación de videos, para identificar cómo debe actuar la aplicación para que la funcionalidad sea sencilla.

Grabación de Video

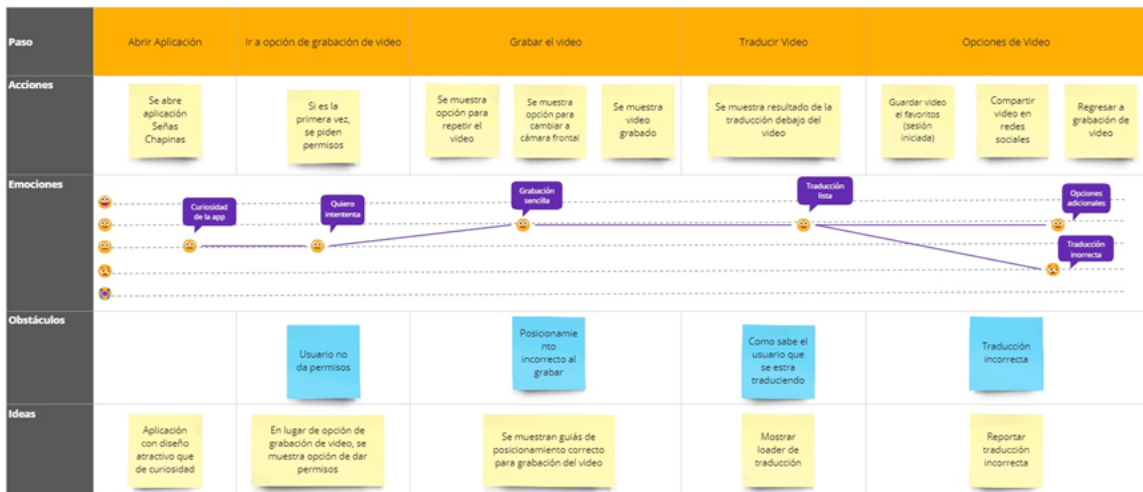


Figura 71: Grabación de video

- Guardado de video, para analizar cómo debe realizarse el proceso.

Guardado de Video

Paso	Intentar guardar video	Con sesión iniciada	Sin sesión iniciada	Guardar video
Acciones	Al finalizar la grabación del video se da la opción de guardado	Si la sección esta iniciada se procede al guardado del video.	Se solicita iniciar sesión para poder proceder.	Redirección a la pantalla de iniciar sesión. Se muestra un pop up de guardado con éxito.
Emociones	Quiero guardar mi video	Estoy guardando mi video	Tengo que iniciar sesión	Logre guardar mi video
Obstáculos			Que pasa si no quiere iniciar sesión	Que pasa si hay un error al guardar el video
Ideas			Se muestra mensaje de video no guardado	Se muestra mensaje de error

Figura 72: Guardando video

- Reporte de traducción errónea, para facilitar una manera en que los usuarios puedan ayudar a mejorar la aplicación.

Reporte

Paso	Traducción incorrecta	Pop up de enviar traducción y video como reporte	Si envía el reporte	No envía el reporte
Acciones	Se presiona el botón de queja	Se despliega un pop up para consultar si se desea enviar la queja junto a la traducción y el video	Se enviara el reporte.	Se cierra el pop up y se envía a la sección de grabar video.
Emociones	Quiero reportar la traducción	¿Quiero enviar el reporte?	Si quiero enviar el video y la traducción	No quiero enviar el video y la traducción
Obstáculos		El usuario no se sienta cómodo enviando el reporte		
Ideas		Agregar opción "saber más" para poder proporcionar información de como se maneja la información		

Figura 73: Reporte

- Diccionario de palabras, para entender de qué manera los usuarios usarían esta herramienta.

Diccionario de palabras

Paso	Abrir diccionario de palabras	Buscar	Seleccionar palabra
Acciones	Abrir el diccionario con el botón en el menú.	Escribir la palabra que se busca en la caja de búsqueda	Se muestra seña de la palabra seleccionada
Emociones			
Obstáculos		Que hacer en caso de que no este la palabra	
Ideas		Mostrar mensaje al usuario de que se esta mejorando la app para agregar más palabras cada vez	

Figura 74: Diccionario de palabras

- Reto diario, para identificar puntos de dolor en esta actividad.

Reto diario

Paso	Se muestra reto diario	Completar reto diario	Reto diario omitido	Usuario no quiere retos diarios
Acciones	Se muestra al usuario una palabra y dos posibles traducciones Se muestra opción para saltar reto diario Se muestra opción de no volver a mostrar reto diario	Reto diario completado correctamente se agrega un quezallito Reto diario completado incorrectamente se muestra mensaje para incentivar que siga jugando mañana	Usuario selecciona opción de omitir reto	Usuario selecciona opción de no volver a mostrar
Emociones				
Obstáculos		Como hacer que el usuario no se decepcione con los retos	Como evitar molestar al usuario	¿Qué pasa si el usuario quiere volver a tener retos diarios?
Ideas		Mensaje de incentivos Por cada reto diario se recolectan quezallitos	Si se obvia reto diario se puede encontrar nuevamente al dar click en el indicador de quezallitos	Se presenta opción de obviar reto para no causarle dolor al usuario al usar la app Opción para reactivar retos diarios en configuraciones

Figura 75: Reto diario

Mapa de sitio

Los mapas de sitio proporcionados visualizan la estructura y navegación de la aplicación *Señas Chapinas*, facilitando un entendimiento claro de las funcionalidades disponibles para los usuarios en diferentes estados.

- **Pantalla para grabar video:** Permite a los usuarios grabar para ser traducido a LENSEGUA.
 - **Compartir video y traducción:** Luego de traducir el video, se puede compartir el video y su traducción.
 - **Reportar traducción incorrecta:** Si la traducción es incorrecta, los usuarios pueden reportar errores.
 - **Guardar video:** Guardar traducciones realizadas para consultarlas nuevamente.
- **Diccionario de palabras:** Permite a los usuarios buscar vocabulario disponible para traducción.
- **Perfil:** Acceso a configuraciones de la cuenta.
 - **Información del usuario:** Muestra los detalles del usuario registrado.
 - **Racha de retos diarios:** Muestra el progreso del usuario en retos diarios de aprendizaje.
 - **Videos guardados:** Acceso a videos que el usuario ha decidido guardar.
 - Al seleccionar un video guardado se acceden a las mismas opciones de video traducido (compartir, reportar o grabar video).
 - **Configuración / sobre la app / ayuda:** Opciones para mejorar la experiencia del usuario en la aplicación.
 - **Cerrar sesión:** Permite al usuario salir de su cuenta.

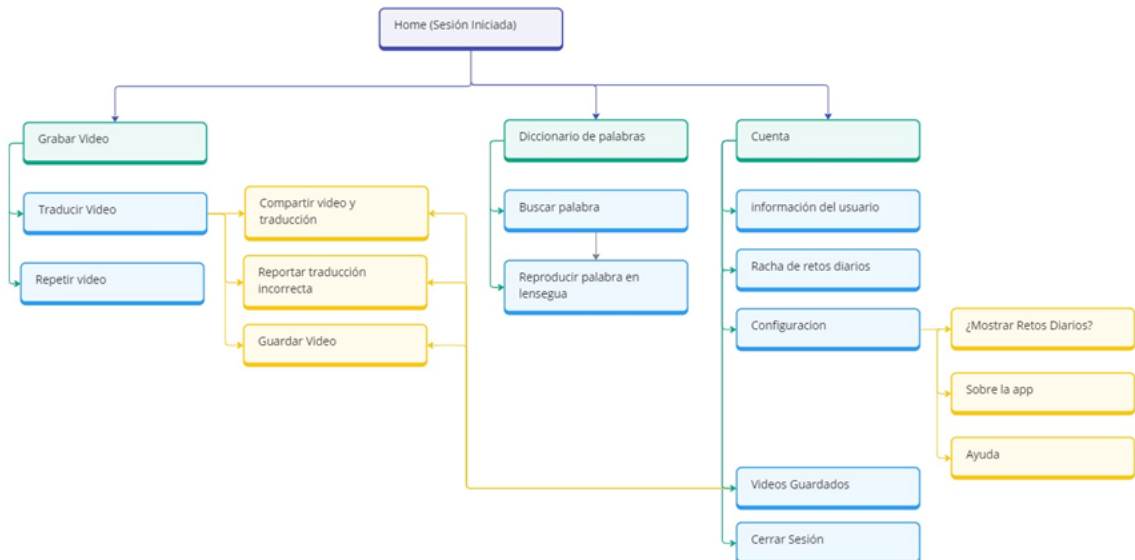


Figura 76: Mapa de sitio

Flujo de usuarios

En el diseño de UX, un flujo de usuarios es una representación visual de los pasos que un usuario sigue dentro de una aplicación para alcanzar un objetivo específico. Esto incluye todas las acciones, decisiones y procesos desde el punto de entrada hasta la salida [8].

Para crear un flujo de usuarios efectivo, es crucial seguir algunos pasos detallados:

1. **Comprender el viaje del cliente:** Con la información recopilada en las Personas, los Mapas de empatía y los Mapas de la experiencia del cliente, se identificaron las necesidades, motivaciones y comportamientos de los usuarios [8].
2. **Identificar y alinear los objetivos:** Cada sección de la aplicación debe tener un objetivo claro que puede diferir de los objetivos del usuario. Por lo tanto, es esencial identificar lo que los usuarios buscan lograr y alinear los objetivos de la aplicación con los de ellos para asegurar que el flujo de usuario los guíe efectivamente hacia acciones deseadas [8].
3. **Decidir la información que necesitan los usuarios:** Basado en las personas y los mapas del viaje del cliente, se definen los pasos necesarios que los usuarios deben seguir dentro del flujo, abordando sus puntos de dolor y proporcionando la información que buscan en cada etapa [8].
4. **Visualizar el flujo:** Finalmente, se visualiza y mapea el esquema utilizando formas para comunicar los diferentes caminos y decisiones en un flujo de usuarios [8].
 - Los óvalos representan el inicio y el final de un flujo de usuarios.
 - Los rectángulos simbolizan un paso del proceso, una página de la aplicación.
 - Las flechas conectan las formas y muestran la dirección del camino del usuario.
 - Los diamantes representan decisiones que los usuarios toman en cada paso.
 - Los paralelogramos indican dónde el usuario debe ingresar algo.
 - El rectángulo redondeado simboliza mensajes al usuario o notificaciones dentro de la aplicación.
5. **Obtener retroalimentación:** Para mejorar la experiencia en la aplicación, se comparte con usuarios finales para identificar posibles fricciones en el flujo y encontrar formas de agilizar y mejorar las funcionalidades [8].

El primer Flujo de Usuario realizado es para Marta. Marta es sorda profunda y trabaja como cajera por lo que desea ofrecer sus productos de caja para obtener comisiones. Para ello graba un video y reproduce el audio de la traducción.

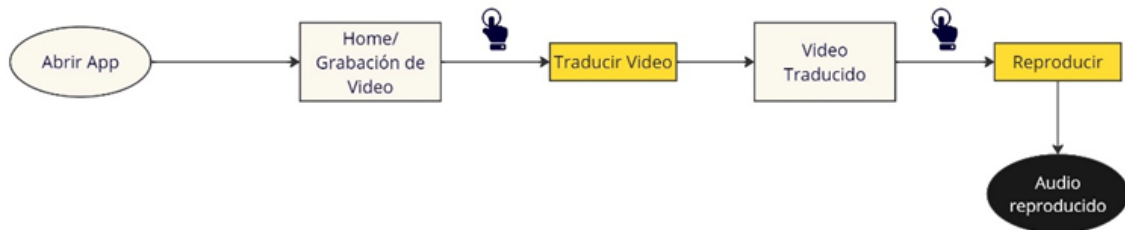


Figura 77: Grabar video

Posteriormente se da cuenta que puede guardar el video para usarlo múltiples veces.

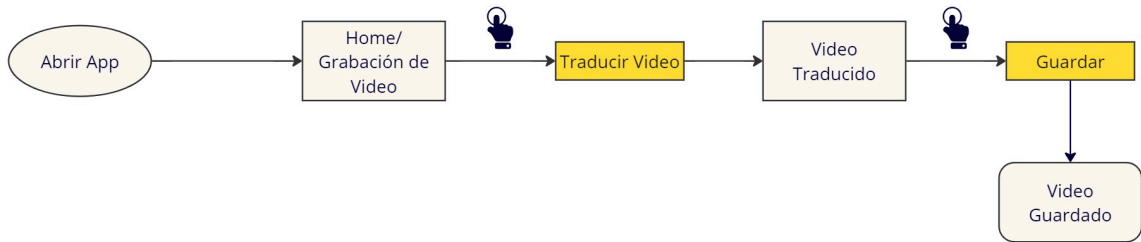


Figura 78: Guardar video

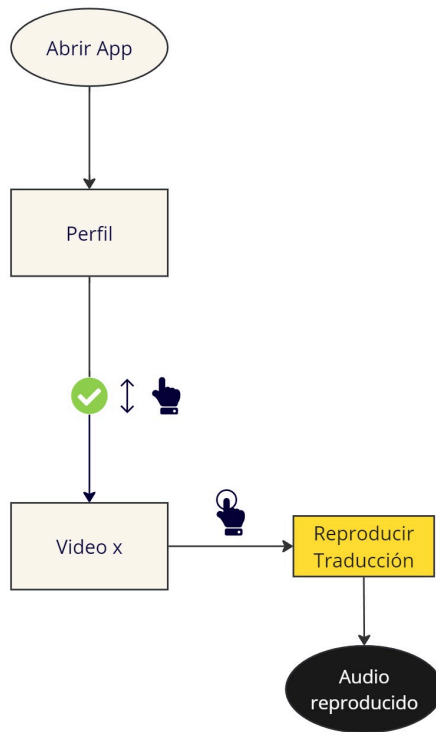


Figura 79: Abrir video guardado

Por otra parte, Ricardo tiene sesiones constantemente con sus clientes, por lo que graba y guarda videos constantemente. Adicionalmente, le gusta repetir sus videos para que queden a su gusto.

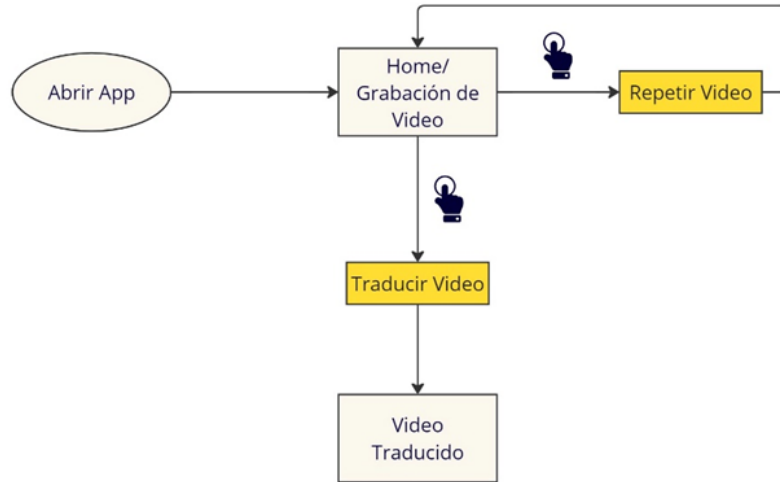


Figura 80: Repertir grabación de video

Laura a su vez, al ser madre de un hijo sordo está aprendiendo LENSEGUA. Por eso le parece importante completar los retos diarios.

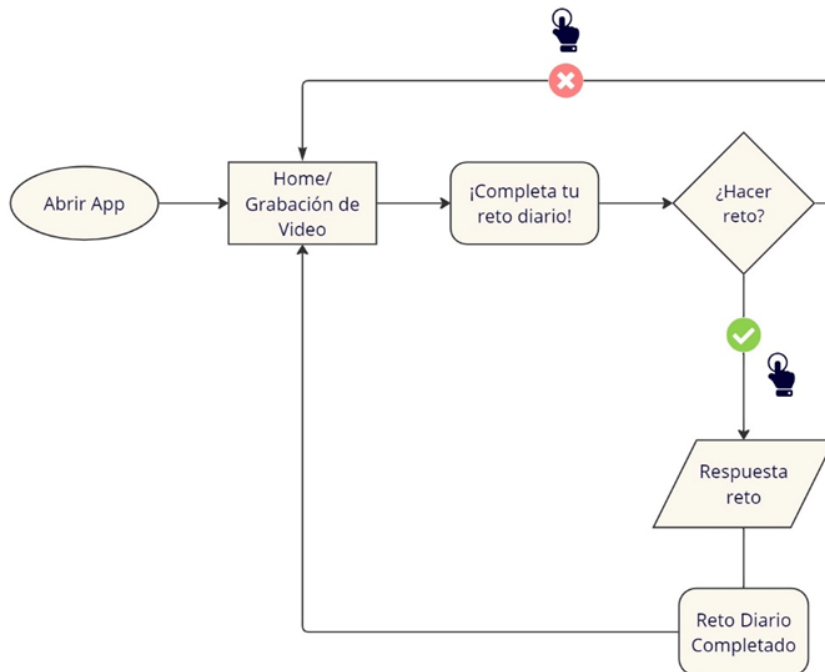


Figura 81: Completar reto

Jorge por su parte, al estar aprendiendo LENSEGUA considera muy importante tener traducciones precisas. Por ello al obtener un resultado incorrecto en su traducción, rápidamente reporta el problema.

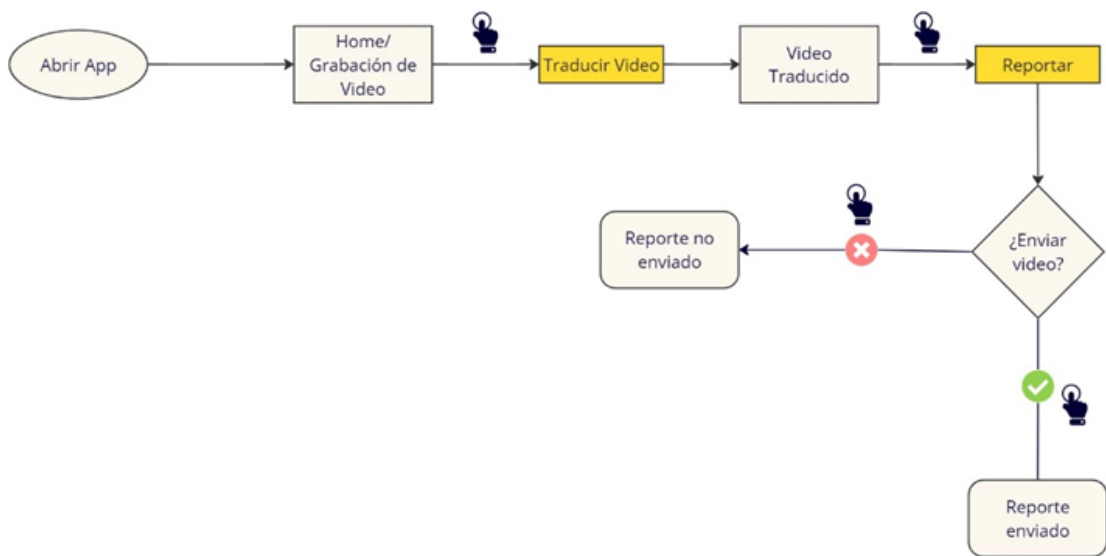


Figura 82: Reportar traducción

Sofía es voluntaria por lo que al tener contacto con la comunidad sorda, despertó su interés por aprender LENSEGUA. Por eso accede al diccionario para mejorar su vocabulario.

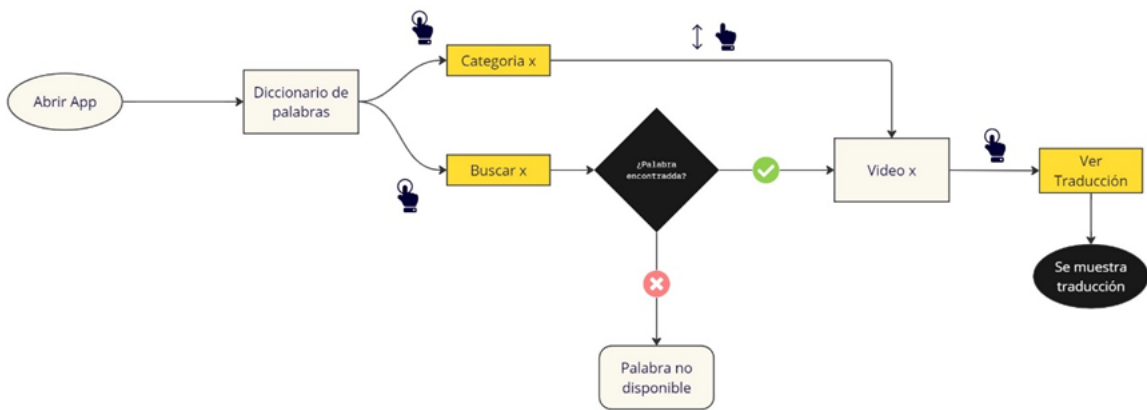


Figura 83: Diccionario

Estructura alámbrica

Un *Wireframe* o estructura alámbrica, es un esquema visual que representa la estructura básica de una aplicación, mostrando el diseño y la disposición de los elementos clave sin entrar en detalles sobre el estilo gráfico o contenido final. Los *Wireframes* utilizan formas simples como rectángulos, líneas y texto básico para indicar elementos como encabezados, párrafos, imágenes y botones [106].

■ Baja fidelidad

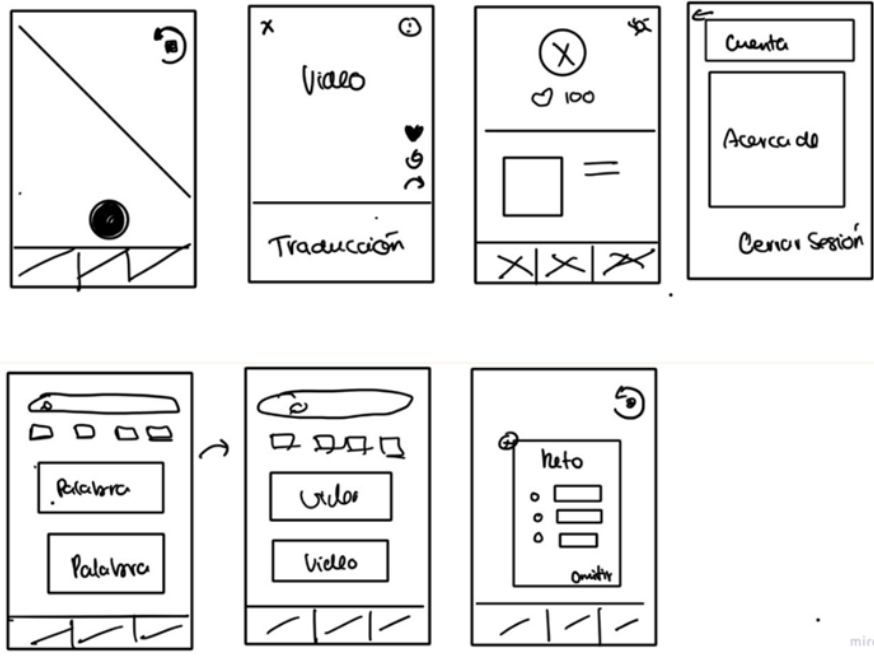


Figura 84: Wireframe bajo nivel

■ Media fidelidad



Figura 85: Wireframe nivel medio

- Alta fidelidad

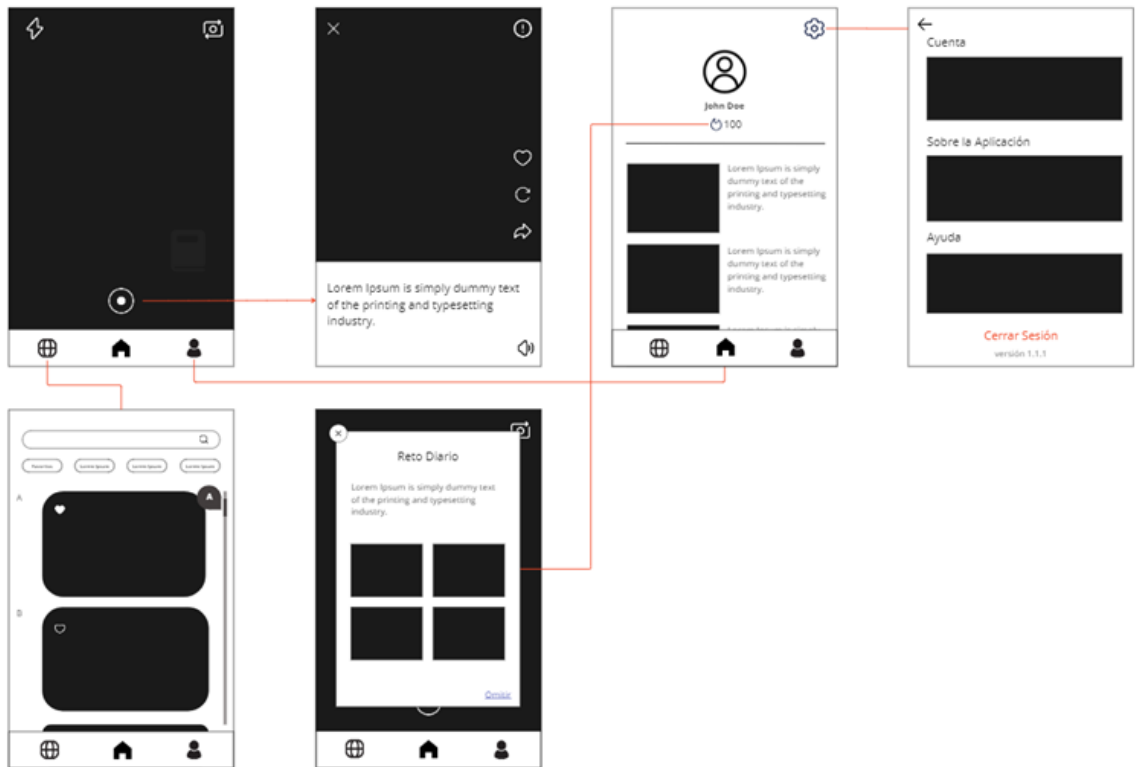


Figura 86: Wireframe alto nivel

Al llegar a este punto del desarrollo de la interfaz y experiencia de usuario se hace **una primera validación con usuarios finales**. Se tuvo una reunión con varias personas de En-Señas pertenecientes tanto a la comunidad sorda como a la comunidad oyente. Tomando en cuenta los comentarios y sugerencias de esta validación, se hicieron algunas mejoras.

- Se ha añadido una nueva pantalla denominada “Traducción” cuyo propósito es permitir que las personas escriban frases en gramática LENSEGUA, que luego son traducidas a gramática española. Esta funcionalidad fue desarrollada en respuesta a los comentarios sobre cómo las personas sordas utilizan herramientas como *ChatGPT* para mejorar su gramática escrita. Es importante destacar que, según los entrevistados, el nivel de comprensión lectora de una persona sorda promedio equivale aproximadamente al de un niño en tercer grado de primaria. Por lo tanto, esta herramienta no solo busca mejorar las habilidades de escritura de las personas sordas.
- En la pantalla del usuario, se han añadido ahora opciones para marcar como favoritos no solo vídeos sino también traducciones.
- En el diccionario, se ha implementado que las tarjetas se vuelvan interactivas, funcionando como *flashcards*. Cada tarjeta muestra, por un lado, la palabra acompañada de una imagen, y por el otro, la seña correspondiente. La incorporación de imágenes fue una recomendación de intérpretes, quienes destacaron que las personas sordas tienden a ser muy visuales y que estas representaciones facilitarían significativamente su comprensión de las palabras.

Teniendo en cuenta estos comentarios y otras sugerencias menores, se realizaron diversas mejoras que culminaron en la creación de un nuevo *Wireframe* de alto nivel.



Figura 87: Wireframe alto nivel luego de retroalimentación

Se consolida entonces la siguiente estructura para la aplicación:

1. Pantalla 1: Home

- **Función principal:** Pantalla de inicio.
- **Elementos:**
 - Botón principal para grabar video.
 - Botones adicionales para funcionalidades como cambiar cámara.
 - Barra de navegación en la parte inferior con cuatro íconos para acceder a las cuatro secciones de la aplicación.

2. Pantalla 2: Traducción de video

- **Función principal:** Mostrar el video grabado junto con su traducción.
- **Elementos:**
 - Área de reproducción del video.
 - Traducción del video en la parte inferior.
 - Botones para agregar a favoritos, compartir, repetir el video, reportar errores en la traducción y cerrar traducción.

3. Pantalla 3: Traducción de LENSEGUA a español

- **Función principal:** Traducir de gramática de LENSEGUA a gramática española.
- **Elementos:**
 - Sección de LENSEGUA para escribir o pegar texto.
 - Sección de traducción al español para copiar, reproducir o guardar la traducción.
 - Botón para traducir/nueva traducción.

4. Pantalla 4: Perfil de usuario

- **Función principal:** Mostrar la información del perfil del usuario.
- **Elementos:**
 - Ícono de perfil y nombre del usuario.
 - Contador de días de racha de desafíos completados.
 - Lista de videos favoritos con sus respectivas traducciones.
 - Lista de traducciones de LENSEGUA a Español favoritas.

5. Pantalla 5: Configuración

- **Función Principal:** Acceder a las configuraciones de la cuenta.
- **Elementos:**
 - Opción par obtener información sobre la aplicación.
 - Opción para cerrar sesión y eliminar la cuenta.
 - Opción para desactivar los retos diarios.

6. Pantalla 6: Diccionario de palabras

- **Función principal:** Mostrar palabras disponibles para traducción.
- **Elementos:**
 - Lista de palabras con sus traducciones en LENSEGUA.
 - Ordenadas alfabéticamente.
 - Botón de agregar a favoritos.
 - Sistema de búsqueda por palabra.
 - Búsqueda por categorías.

7. Pantalla 7: Desafíos diarios

- **Función principal:** Presentar desafíos diarios de traducción.
- **Elementos:**
 - Palabra en español
 - 4 imágenes de posibles traducciones en LENSEGUA
 - Opciones para omitir y cerrar el desafío.

Logo

El logo de la aplicación *Señas Chapinas* busca simbolizar la comunidad sorda y la cultura guatemalteca. Se escogió un quetzal, el ave nacional de Guatemala, para representar el país, y manos para representar a la comunidad sorda. La evolución del logo refleja una serie de mejoras y refinamientos para lograr un diseño que comunique efectivamente estos valores.

El primer logo integraba ambos componentes: el quetzal y las manos. Las alas y la cola del quetzal se diseñaron para parecerse a manos en señas, representando así tanto la identidad cultural guatemalteca como la lengua de señas. Este diseño inicial se envió a un diseñador gráfico para recibir orientación de como mejorar su forma y funcionalidad.



Figura 88: Primer logo

El segundo diseño mantuvo la misma idea pero fue vectorizado, simplificado y mejorado en términos de legibilidad y estilo. La vectorización permitió un diseño más limpio y adaptable a diferentes tamaños y medios.



Figura 89: Logo vectorizado

Para el tercer diseño, se solicitó la colaboración de una diseñadora gráfica con el fin de modernizar el logo. Como resultado, se decidió utilizar solo una mano en lugar de dos, simplificando y actualizando el concepto visual.



Figura 90: Logo modernizado

En colaboración con la diseñadora gráfica se seleccionaron los colores del logo, incorporando tonos verde, rojo y amarillo que evocan las características distintivas del quetzal y reflejan la identidad cultural guatemalteca. Estos colores no solo realzan visualmente el logo, sino que también fortalecen su vínculo con el patrimonio nacional.



Figura 91: Logo con colores

Finalmente, se logró el diseño definitivo: un logo colorido y moderno que refleja la cultura de Guatemala a través del símbolo del quetzal y representa a la comunidad sorda con la imagen de una mano utilizando LENSEGUA. Este logo combina simplicidad, modernidad y simbolismo cultural, proporcionando una representación atractiva y efectiva para la aplicación.



Figura 92: Logo Señas Chapinas

Paleta de colores

Los colores seleccionados para el logo son los siguientes:

- **Verde Quetzal (#00973A):** Este color representa al quetzal y simboliza vida y esperanza.
- **Rojo (#E20613):** Su uso en la aplicación es para indicar errores o acciones importantes, facilitando al usuario identificar problemas o acciones críticas dentro de la aplicación.
- **Verde Lima (#93C01F):** Se utiliza para dar mas detalle al quetzal y para el título de la aplicación porque tiene mayor contraste con el azul del fondo.
- **Amarillo (#FFDD00):** Usado para destacar el pico del quetzal en el logo, este amarillo no solo contrasta eficazmente con los verdes, sino que también simboliza felicidad y acción.
- **Azul (#29235C):** Aunque el azul tradicional de la bandera de Guatemala es más claro, se ha seleccionado un tono azul oscuro para conferir profundidad y seriedad al logo

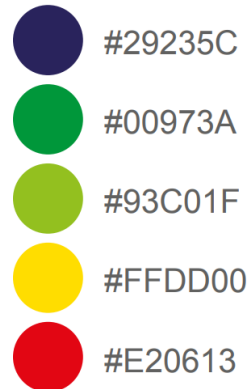


Figura 93: Paleta de colores logo

Con base en estos colores, también se seleccionaron los colores para la aplicación, buscando no solo impacto visual, sino también fortalecer la conexión con la identidad cultural guatemalteca.

- **Azul (#29235C):** Se utiliza principalmente como color para texto de la aplicación. Ofrece legibilidad y una sensación de seguridad y confianza.
- **Verde quetzal (#00973A):** Se usa para botones y títulos, aportando vitalidad y energía a la interfaz.
- **Rojo (#E20613):** Se utiliza para indicar errores o acciones importantes, facilitando al usuario identificar problemas o acciones críticas dentro de la aplicación.
- **Blanco crema (#F5F5F5):** Se utiliza como fondo para la aplicación. En lugar del blanco puro, este tono crema proporciona una suavidad que puede ser menos agresiva a la vista.
- **Gris carbón (#323232):** Se utiliza principalmente como fondo de pantallas de temporales. Es un color neutro elegido específicamente para minimizar las distracciones y evitar que los usuarios permanezcan en estas pantallas transitorias por más tiempo del necesario, a diferencia del uso de los colores azul y blanco en otras secciones de la aplicación, que buscan captar y mantener la atención del usuario.

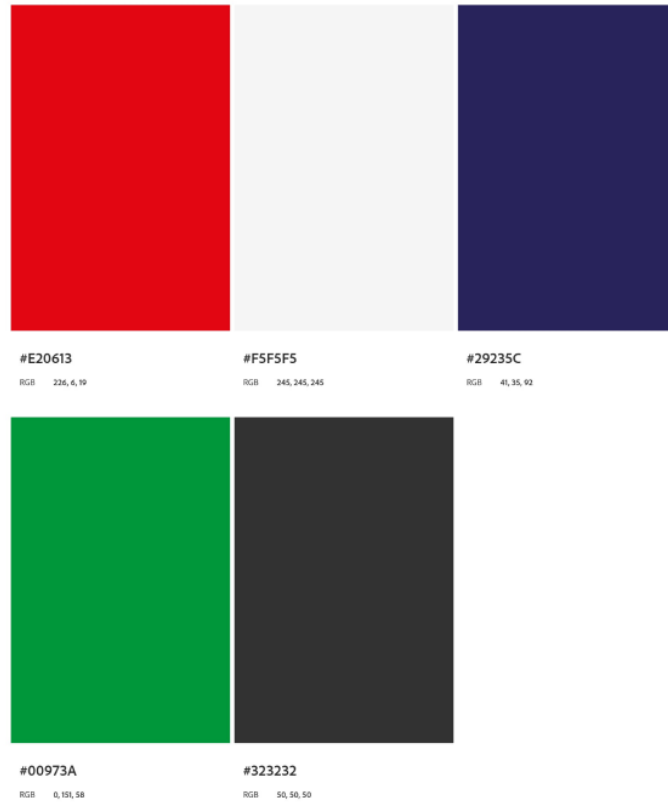


Figura 94: Paleta de colores aplicación

Con base a la paleta de colores del logo, se obtienen colores pastel utilizados para el quetzal de perfil del usuario y las opciones del reto diario:

- Aqua pastel (#37B7C3)
- Azul pastel (#83B4FF)
- Verde pastel (#8DC249)
- Amarillo pastel (#FCB424)



Figura 95: Paleta colores perfil

Asimismo también se comprueban contrastes de los colores seleccionados:

- **Azul y blanco:** Los colores azul y blanco seleccionados presentan un contraste óptimo, lo cual justifica su uso como fondo y texto principales en la aplicación.



Figura 96: Contraste blanco y azul



Figura 97: Contraste azul y blanco

- **Contraste gris y blanco:** Los colores gris y blanco seleccionados ofrecen un contraste adecuado, lo cual los hace idóneos para su uso en las pantallas temporales.



Figura 98: Contraste gris y blanco

- **Contraste blanco y rojo:** Aunque el contraste entre blanco y rojo no es tan alto como en otros casos, aún cumple con los requisitos mínimos recomendados. Este se utiliza exclusivamente en situaciones necesarias para alertar al usuario sobre una acción crítica.



Figura 99: Contraste blanco y rojo

- Contraste verde quetzal y azul:** El contraste entre el verde quetzal es ligeramente inferior al nivel recomendado. Por esta razón, se empleará únicamente para elementos destacados como títulos grandes, botones y el logo, donde la legibilidad sigue siendo efectiva a pesar del menor contraste.



Figura 100: Constraste verde quetzal y azul

- Contraste verde claro y azul:** El contraste entre el verde claro y el azul cumple con las normas recomendadas, razón por la cual se ha seleccionado para el título de la aplicación



Figura 101: Constraste verde claro y azul

Tipografía

La tipografía recomendada por la diseñadora gráfica para el logo de *Señas Chapinas* fue Adonide Bold. Esta elección se justifica por su claridad y modernidad, cualidades que reflejan la simplicidad y accesibilidad que la aplicación busca transmitir. Adonide Bold es una fuente sans-serif que ofrece un aspecto limpio y profesional, ideal para destacar en logotipos y cabeceras por su legibilidad y presencia [55].

Con base en esa tipografía y al contexto de la aplicación, se decide utilizar Nunito para el contenido de la misma. Nunito, también una sans-serif disponible gratuitamente en *Google Fonts*, es conocida por sus curvas suaves y su legibilidad en interfaces digitales, lo que la hace ideal para textos largos y elementos de interfaz de usuario en aplicaciones móviles. Su diseño amigable y accesible complementa perfectamente la estética introducida por Adonide Bold en el logo, asegurando coherencia visual y facilitando la experiencia del usuario al navegar por la aplicación. Esta elección refuerza el objetivo de la aplicación de ser accesible y fácil de usar, elementos cruciales para una herramienta destinada a mejorar la comunicación en la comunidad sorda [38].



Figura 102: Tipografía Nunito

Prototipos

Al concluir la fase de diseño UX/UI, se han definido claramente las necesidades de los usuarios, así como los flujos principales de la aplicación. Durante este proceso, también se seleccionaron cuidadosamente los colores y la tipografía que mejor se adaptan a la experiencia del usuario, garantizando así coherencia visual y funcionalidad. Con estos elementos bien establecidos, el siguiente paso ha sido la creación de prototipos detallados en Figma. Este enfoque permite simular la interacción del usuario con la aplicación.

Prototipo de bajo nivel

Este primer prototipo tiene como objetivo demostrar la funcionalidad general de la aplicación sin enfocarse en el diseño de colores, presentando una estructura básica. Para ver el diseño con más detalles consultar: [Figma](#).

- **Barra de navegación:** Contiene las opciones de “Video”, “Traductor”, “Diccionario” y “Perfil”. La opción seleccionada se destaca visualmente sobre las demás.

- **Pantalla de inicio:** Ofrece la función principal de grabar video. Al presionar el botón correspondiente, el video comienza a grabarse.
- **Pantalla de video:** Muestra el video grabado junto con opciones para reportar errores, agregar a favoritos, compartir, repetir, utilizar el altavoz y cerrar. Las traducciones en español y LENSEGUA se muestran en la parte inferior.
- **Reporte de traducción:** Un modal que permite al usuario reportar errores en la traducción. Antes de continuar, se solicita revisar el diccionario. Luego, el usuario puede proceder con el reporte.
- **Traductor:** Al ingresar, se presenta un área para escribir o pegar texto en LENSEGUA. El botón de traducción está deshabilitado hasta que el usuario ingrese el texto. Una vez ingresado, el botón se habilita, y al presionarlo, se muestra la traducción al español junto con las opciones de copiar, escuchar con el altavoz, agregar a favoritos o realizar una nueva traducción.
- **Diccionario:** Muestra tarjetas redondeadas que representan las palabras disponibles. Las tarjetas se pueden agregar a favoritos. En la parte superior, se incluye una barra de búsqueda y la opción de seleccionar por categorías. Además, cuenta con una barra de desplazamiento para navegar entre las tarjetas ordenadas alfabéticamente.
- **Perfil del usuario:** Muestra una imagen de perfil, el nombre del usuario y un contador de días consecutivos de desafíos completados. También incluye un ícono de configuración en la parte superior para acceder a ajustes. La pantalla presenta una barra de pestañas que permite alternar entre los videos favoritos y las traducciones favoritas.
- **Configuraciones:** Ofrece opciones para cambiar la contraseña, cerrar sesión, eliminar la cuenta y acceder a información sobre la aplicación. También permite activar o desactivar los retos diarios y consultar la versión actual de la aplicación.
- **Retos diarios:** Presenta cuatro imágenes como opciones de respuesta para una palabra en LENSEGUA. El usuario puede elegir una respuesta, o bien, cerrar u omitir el reto. Las opciones se presentan en un diálogo interactivo.

Cabe destacar que el diseño del primer prototipo se caracteriza por su simplicidad e intuición, con un uso mínimo de texto para facilitar la navegación. Basado en las recomendaciones de los usuarios finales, se utilizaron como referencia aplicaciones que ellos consideran fáciles de usar, adaptando y personalizando el diseño a la identidad y necesidades únicas de la aplicación 'Señas Chapinas'. Un ejemplo de ello es la disposición de las opciones en la pantalla de video, que recuerda a la aplicación *TikTok*.



Figura 103: Primer prototipo

Prototipo de nivel medio

Tras presentar el primer prototipo a los usuarios finales, se realizaron algunos ajustes visuales menores, como la modificación del tamaño del texto y de los íconos. Después de implementar estos cambios y obtener la aprobación de los usuarios, se desarrolló el segundo prototipo. Este incluyó pantallas para el inicio de sesión y la creación de cuenta, una pantalla de inicio, una pantalla para cambio de contraseña, y una pantalla con descripción de la aplicación. Además, se empezó a integrar color y mejorar la navegación. Para analizar con más detalle revisar: [Figma](#)



Figura 104: Segundo prototipo

El tercer prototipo muestra ya el diseño completo de la aplicación, para ver mas a detalle revisar [Figma](#):

- **Pantalla de *Splash*:** Fondo azul con el logo y nombre de la aplicación.
- **Pantalla de inicio:** En la parte superior se muestra el logo y el nombre, seguido por el eslogan en español y LENSEGUA. También se incluyen los botones “Empezar Ahora”, que lleva a la creación de cuenta, y “Ya tengo una cuenta”, para iniciar sesión.
- **Creación de cuenta:** El logo y el nombre se presentan en la parte superior azul, mientras que en la parte inferior blanca se muestran los campos para ingresar el correo, la contraseña y la confirmación de contraseña. Se incluyen dos botones: “Empezar Ahora”, que continúa con el flujo, y “Ya tengo una cuenta”, para redirigir al login.
- **Grabación de video:** Muestra la simulación de la toma de video, con un botón de grabación al centro y una barra de navegación inferior que incluye las opciones de video, traducción, diccionario y perfil.
- **Video:** Presenta una simulación del video grabado con opciones en la parte superior para reportar, agregar a favoritos, compartir y repetir el video. En la parte inferior se muestra la traducción en LENSEGUA y español. Se usa iconografía estándar en lugar de texto para facilitar la navegación a las personas sordas.
- **Reportar:** A diferencia del primer prototipo, ya no se utiliza un modal, sino un *bottom sheet*, de acuerdo con las recomendaciones de un experto en diseño, ya que es más dinámico, moderno y menos intrusivo. El título se presenta en rojo para resaltar que es una acción crítica. Se le pide al usuario que revise si la palabra existe en el diccionario antes de reportarla, y se muestran botones con el ícono del diccionario utilizado en la barra de navegación, cumpliendo con la solicitud de las personas sordas de utilizar iconografía y no solo texto. El botón de reportar también es rojo, ya que se trata de una acción crítica. Al finalizar el flujo de reporte, se muestra un mensaje confirmando que el reporte ha sido generado.
- **Traductor:** A diferencia del primer prototipo, se introduce un fondo azul en el área de ingreso de texto en LENSEGUA y un fondo blanco para el texto traducido en español. Se ha añadido un límite máximo de palabras para ingresar, a solicitud del equipo de inteligencia artificial. Al igual que en el primer prototipo, el botón de traducir es verde, para resaltar la acción. Este permanece deshabilitado hasta que el usuario ingrese texto en LENSEGUA. Al hacer clic en el botón de traducir, se bloquea la edición del texto en LENSEGUA, y el botón cambia a “Nueva Traducción”. Posteriormente, se muestra la traducción en español y las opciones para copiar, usar el altavoz y agregar a favoritos, manteniendo el uso de iconografía en lugar de texto.
- **Diccionario:** El diseño de las tarjetas ha cambiado ligeramente en comparación con el primer prototipo. Ahora se incluye un ícono que indica que la tarjeta se puede voltear, y se ha reducido el redondeado y sombreado de las tarjetas. La barra de búsqueda se mantiene en la parte superior. En las categorías, se ha agregado la sección de favoritos con un ícono de corazón, manteniendo la coherencia de la aplicación, donde los favoritos siempre se representan con un corazón. También se ha incorporado un estado vacío (*empty state*) para cuando no haya resultados en la búsqueda o no se hayan guardado favoritos.
- **Perfil:** A diferencia del primer prototipo, donde el usuario podía seleccionar su propia imagen de perfil, se decidió que la aplicación asignará un color al quetzal del logo, para cada usuario. Se sigue mostrando el nombre del usuario y la racha de desafíos en la parte inferior de la imagen de perfil. Además, el diseño de los favoritos, tanto de videos como de traducciones, ha sido ajustado para añadir más formalidad y dinamismo a la aplicación. También se incluyen flujos para borrar favoritos y pantallas de estado vacío cuando no haya favoritos guardados.

- Configuración:** El diseño de la sección de configuración ha cambiado respecto al primer prototipo. Ahora incluye subconjuntos de opciones e iconografía para facilitar la navegación del usuario. Además, se ha añadido el flujo de eliminación de cuenta, que se muestra mediante un *bottom sheet* con un título en rojo, dado que es una acción crítica. El diseño de la sección “Acerca de la Aplicación” también ha sido ajustado ligeramente y se han añadido los colaboradores.
- Reto diario:** El diseño de la pantalla de retos diarios fue uno de los más desafiantes, ya que no se sabía cómo proporcionar retroalimentación clara cuando el usuario seleccionaba una opción incorrecta, ni cómo manejar las rachas de forma visual. Finalmente, se decidió por una pantalla que en la parte superior muestra la racha actual, un título llamativo, y la palabra objetivo a seleccionar en las tarjetas. Las tarjetas utilizan colores pastel, y en la parte inferior se encuentra la opción de omitir. Cuando la respuesta es correcta, se añade un punto a la racha y aparece una animación de confeti. Si la respuesta es incorrecta, la racha se restablece a cero, el ícono de la racha cambia de color, y se muestra un mensaje para motivar al usuario a intentarlo de nuevo. La tarjeta correcta se muestra en un color más claro que las demás después de que el usuario haya seleccionado una opción. A diferencia del primer prototipo, esta es una pantalla completa y no un modal.
- Cambio de contraseña:** Al igual que en el primer prototipo, se muestra el flujo para cambiar la contraseña utilizando el mismo diseño empleado en las pantallas de inicio de sesión y creación de cuenta, manteniendo una coherencia visual en toda la aplicación.
- Diálogos adicionales:** Se han integrado diálogos estándar para mostrar mensajes, como alertas de error, y un *loader* para indicar la ejecución de llamadas a servicios.

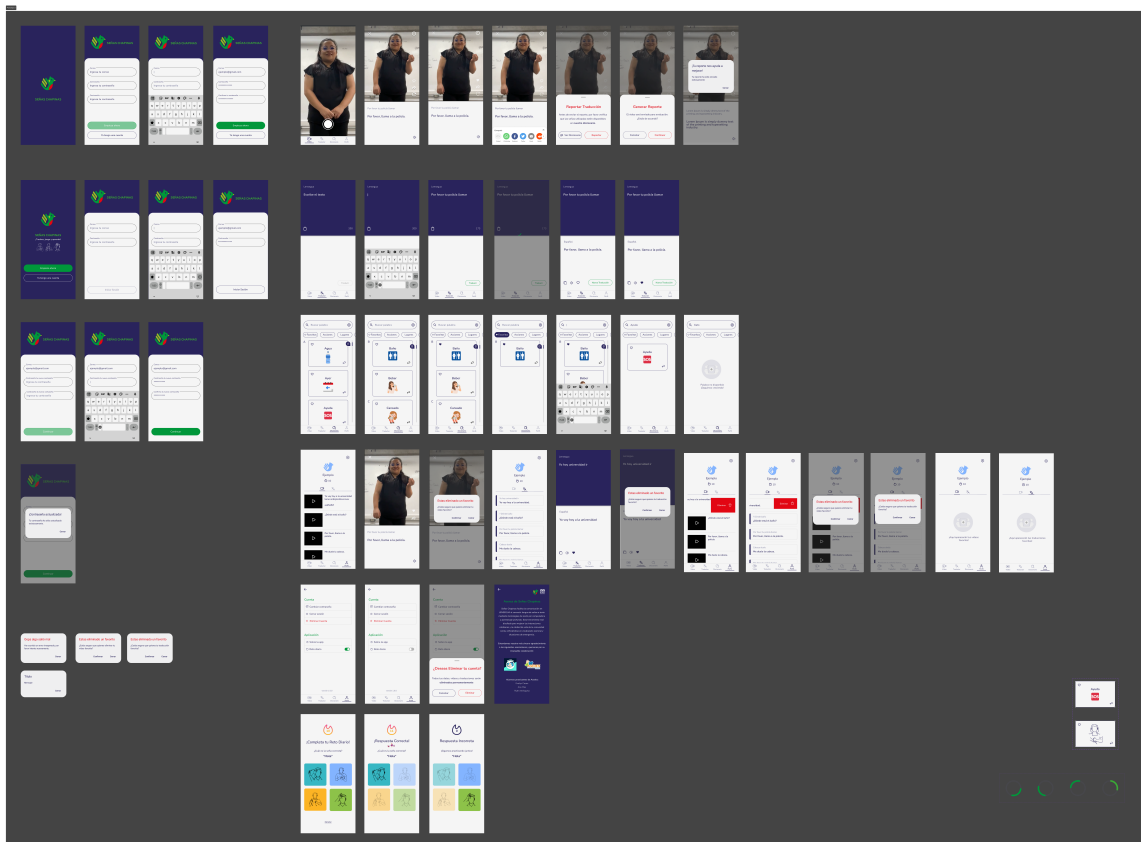


Figura 105: Tercer prototipo

Cambios prototipo de alto nivel

Después de presentar el prototipo a expertos en tecnología y usuarios finales, se realizaron varios cambios para mejorar la experiencia del usuario:

- **Pantalla de creación de cuenta:** Se eliminó el campo de confirmación de contraseña. En su lugar, se añadió un icono de ojo en el campo de contraseña para mostrar u ocultar la misma, permitiendo que el usuario confirme su contraseña sin necesidad de un campo adicional. Además, se implementaron validaciones de longitud y caracteres necesarios para cumplir con los requisitos de seguridad.
- **Pantalla de inicio de sesión:** Se agregó un campo para “Olvidé mi contraseña”, facilitando a los usuarios la recuperación de su contraseña.
- **Pantalla de inicio:** Se introdujo la opción para que los usuarios elijan si desean activar su cámara al iniciar la aplicación. Si optan por no hacerlo, aparece una pantalla gris temporal que incentiva al usuario a abrir su cámara mediante un botón en la parte inferior. Esta configuración se puede ajustar en las opciones de la aplicación. Además, se añadió una pantalla de solicitud de permisos.
- **Pantalla de grabación de video:** Se añadió un botón para cambiar la cámara y un acceso directo a videos favoritos. También se incorporó una guía de posicionamiento correcto para el rostro.
- **Pantalla de video grabado:** El diseño de esta pantalla cambió ligeramente respecto a los prototipos anteriores. Las opciones de video como reportar, altavoz, favoritos y compartir se trasladaron a la parte inferior, junto con el cuadro de traducciones en español y LENSEGUA. Esto se hizo para reflejar que estas opciones se aplican al texto traducido y no al video en sí. Además, el botón de repetir video se eliminó por ser redundante; cerrando el video grabado, es posible grabar uno nuevo.
- **Pantalla de traducción:** Se eliminó la opción de pegar directamente, ya que esta función es accesible manteniendo presionado el campo de entrada. Se añadió un acceso directo a traducciones favoritas.
- **Pantalla de diccionario:** Se sugirió cambiar el color de las tarjetas para añadir más colorido a la pantalla.
- **Pantalla de configuraciones:** Se añadió una opción para activar la cámara al iniciar la aplicación.
- **Pantalla de reto diario:** El funcionamiento de esta pantalla fue modificado respecto al prototipo anterior. Ahora, cuando el usuario selecciona una opción incorrecta, esta se torna más pálida y vibra. Cuando selecciona la opción correcta, las demás opciones se atenúan, destacando la respuesta correcta y se muestra animación de confeti. Esto garantiza que el usuario siempre aprenda cuál es la opción correcta y no se frustre al perder su racha, ya que la única forma de perder es no completar el reto diario. Se agregó un botón de cerrar para regresar a la pantalla de inicio.

Asimismo, se realizaron ajustes para adaptar la aplicación a diferentes largos de texto. Se efectuaron algunos cambios en el tamaño del texto, el reposicionamiento de iconos, y otros ajustes menores, todo ello con el fin de optimizar la legibilidad y la usabilidad en diversos dispositivos.

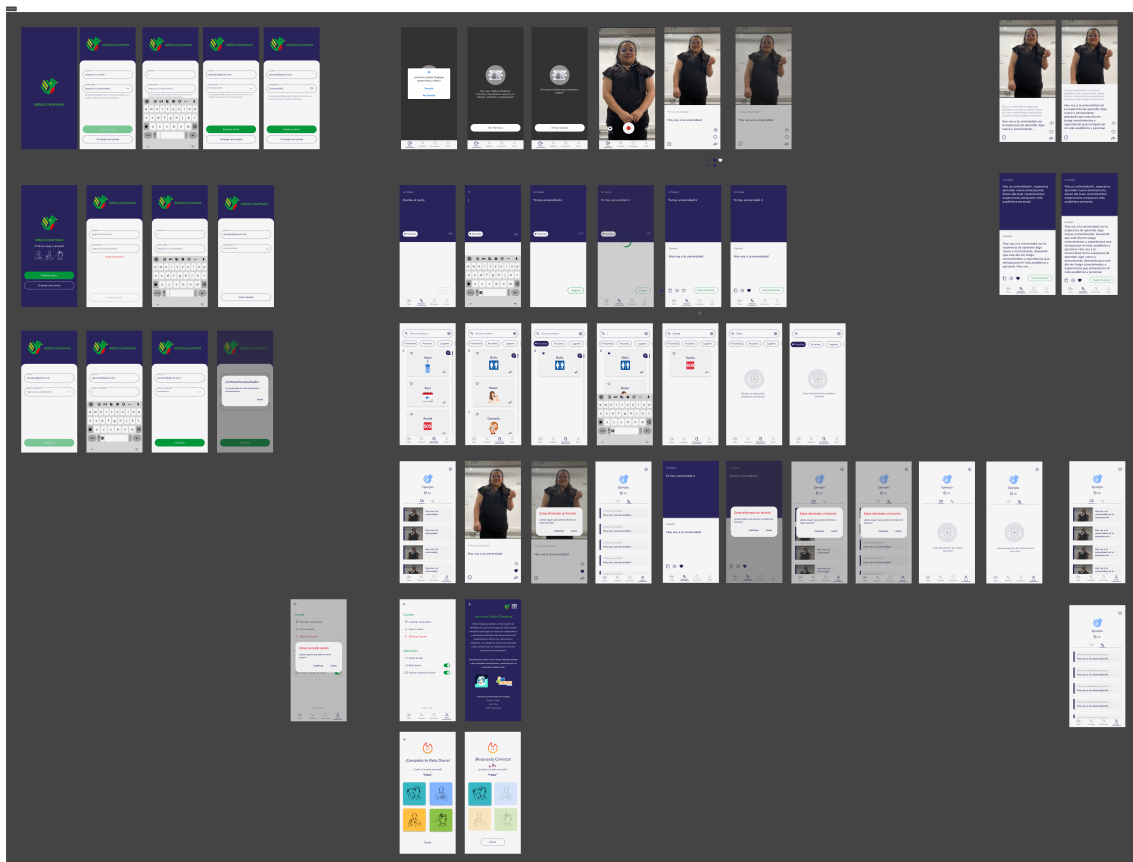


Figura 106: Cuarto prototipo

Creación de ilustraciones para módulo de diccionario

En un principio, se había decidido que la aplicación sería distribuida de manera local, compartiendo el archivo *APK* directamente con los usuarios. Sin embargo, conforme fue avanzando el desarrollo y al profundizar en el trabajo con la comunidad, se hizo evidente que la aplicación tenía un valor significativo que podría beneficiar a un público mucho más amplio. Esto llevó a la decisión de publicar la aplicación en la Play Store, lo que marcó un giro importante en el enfoque del proyecto.

Con esta decisión, surgió la necesidad de hacer ajustes importantes, especialmente en lo relacionado con los derechos de autor de las imágenes. Inicialmente, las imágenes del módulo de diccionario provenían de un libro, cuyo uso estaba permitido únicamente en un contexto académico. No obstante, al decidir que la aplicación estaría disponible en una plataforma comercial, fue indispensable crear nuevas imágenes que fueran completamente originales y que cumplieran con las normativas de propiedad intelectual.

Para ello, se emprendió un proceso creativo en el que todas las imágenes fueron redibujadas a mano utilizando Procreate, una herramienta especializada en ilustración digital. Posteriormente, cada una de estas ilustraciones fue refinada y vectorizada mediante Inkscape, un software de diseño que permite trabajar con gráficos escalables. Este proceso no solo garantizó el cumplimiento de las normativas legales, sino que también mejoró la calidad visual del módulo de diccionario, aportando un aspecto más profesional y atractivo.

El trabajo de rediseño fue exhaustivo, ya que cada imagen fue cuidadosamente elaborada para

mantener la fidelidad a los gestos y signos utilizados, asegurando que fueran fácilmente reconocibles para los usuarios. Además, se tuvo en cuenta la coherencia visual y la accesibilidad, con el objetivo de ofrecer una experiencia de usuario óptima. Este esfuerzo no solo permitió que la aplicación cumpliera con los requisitos para su publicación, sino que también elevó la calidad general del proyecto.



Figura 107: Ilustraciones señas

5.5.3. Desarrollo de aplicación móvil para Android

Descripción general del desarrollo móvil

La aplicación *Señas Chapinas* está desarrollada para dispositivos *Android* utilizando *Java* para la lógica y el desarrollo principal. Se busca ofrecer una experiencia de usuario intuitiva y accesible, utilizando arquitecturas modernas y prácticas eficientes para garantizar un rendimiento óptimo.

En cuanto al diseño de la interfaz, se optó por el uso de *XML-based UI* en lugar de *Jetpack*

Compose. Este enfoque fue elegido por su madurez, estabilidad y la familiaridad que ofrece a los desarrolladores. Además, **XML-based UI** permite un mayor control sobre el diseño de las pantallas y una fácil integración con herramientas como **View Binding** [35].

Para facilitar la vinculación entre las vistas y el código, se utilizó **View Binding**. Esta herramienta elimina la necesidad de métodos como `findViewById()`, mejorando la eficiencia y reduciendo errores al acceder a los elementos de la interfaz directamente desde el código [134].

La estructura de la aplicación está centrada en un único **MainActivity**, que gestiona la navegación entre *Home*, *Traducción*, *Diccionario*, y *Perfil* mediante **NavController** y gráficos de navegación (**NavGraphs**) independientes para cada sección. Esta decisión permite mantener un código más ordenado y modular, al mismo tiempo que se simplifica la navegación dentro de la aplicación, evitando la complejidad que supondría utilizar varios **Activities**.

La aplicación incluye un **Bottom Navigation Menu** para facilitar la navegación entre las diferentes secciones. Este menú se sincroniza con el **NavController** para permitir transiciones fluidas entre los diferentes **Fragments** de la aplicación.

Tecnologías y librerías

La aplicación utiliza una variedad de librerías modernas para optimizar su rendimiento y funcionalidad:

- **AndroidX:** Para garantizar compatibilidad y soporte con las versiones más recientes de Android, así como un diseño visual coherente y moderno.
- **View Binding:** Para gestionar de manera eficiente la vinculación de vistas.
- **SharedPreferences:** Utilizado para almacenar configuraciones del usuario, como el estado de inicio de sesión, de manera persistente entre las sesiones de uso de la aplicación.
- **LiveData** y **ViewModel:** Cada **Fragment** tiene su propio **ViewModel** para gestionar de forma eficiente los cambios de estado y lógica de la interfaz de usuario.
- **Navigation Component:** Para gestionar la navegación entre **Fragments** de manera flexible y organizada.
- **Glide:** Para la carga y manipulación de imágenes de forma eficiente.
- **CameraX:** Para implementar funcionalidades relacionadas con la captura de video.
- **ExoPlayer:** Para la reproducción de videos de alta calidad dentro de la aplicación.
- **Gson:** Para la serialización y deserialización de datos JSON.
- **Lottie:** Para integrar animaciones ligeras y dinámicas en la interfaz de usuario.
- **Retrofit:** Para la implementación de servicios.

Arquitectura del proyecto

El proyecto sigue el patrón arquitectónico *Model-View-ViewModel* (MVVM) para organizar el código de manera eficiente y mantener la separación de responsabilidades entre la lógica de negocio, la interfaz de usuario y la manipulación de los datos.

Organización de la aplicación

La aplicación se gestiona con una única `MainActivity`, que centraliza la navegación y la interacción del usuario. Cada una de las secciones principales de la aplicación (*Home*, *Traducción*, *Diccionario*, y *Perfil*) cuenta con su propio `NavGraph` para gestionar de forma modular y ordenada los flujos de navegación entre los diferentes `Fragments`. Este enfoque permite que las diferentes secciones se mantengan independientes, favoreciendo el mantenimiento y escalabilidad del código.

Gestión de estados

Cada `Fragment` extiende de un `BaseFragment`, que contiene la lógica compartida entre ellos, como la navegación, la gestión de diálogos personalizados y el manejo de la barra de navegación inferior. Además, cada `Fragment` cuenta con su propio `ViewModel`, que hereda de un `BaseViewModel`. Esto permite gestionar los cambios de estado y lógica específicos de cada pantalla de forma eficiente y organizada, manteniendo los datos y los cambios de estado en el ciclo de vida adecuado, independientemente de los cambios en la interfaz.

Navegación

La navegación en la aplicación es gestionada mediante el componente *Navigation Component*, y las transiciones entre los `Fragments` se gestionan a través de un `NavController` central, que se sincroniza con el `Bottom Navigation Menu` para ofrecer una experiencia de usuario fluida. Este enfoque también simplifica el manejo del historial de navegación, permitiendo una interacción más intuitiva para el usuario.

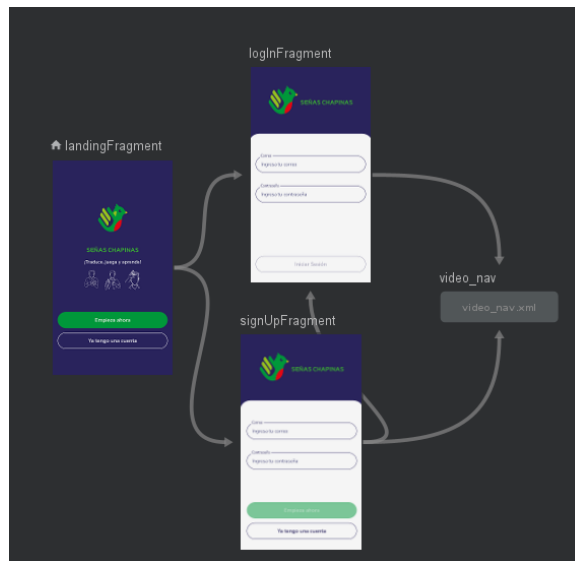


Figura 108: Navegación principal

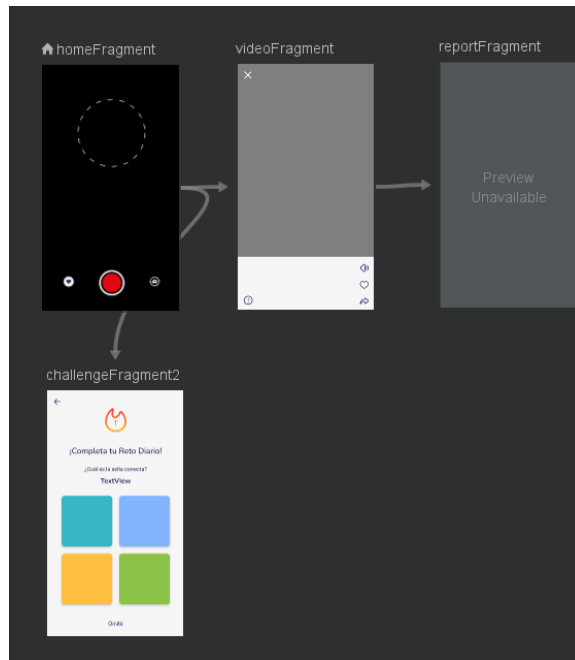


Figura 109: Navegación video

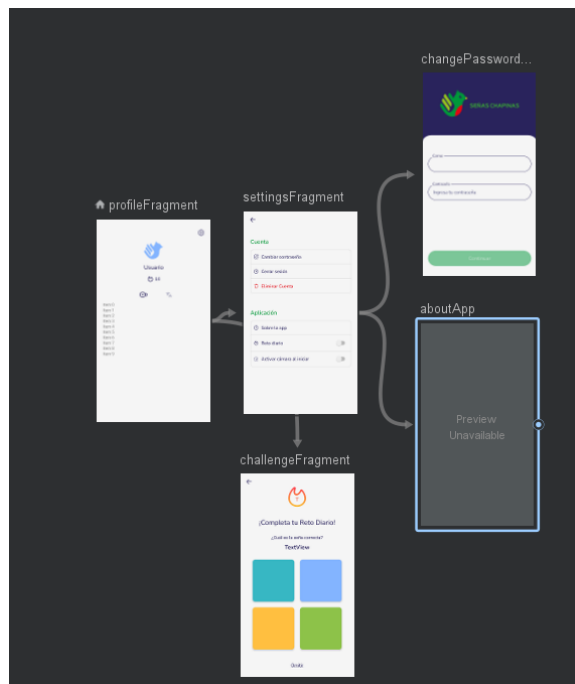


Figura 110: Navegación perfil



Figura 111: Diccionario

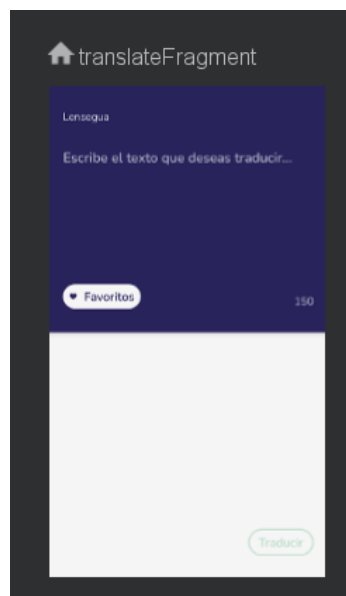


Figura 112: Traducción

Componentes reutilizables

En el desarrollo de la aplicación móvil, se ha implementado una serie de componentes reutilizables con el objetivo de mantener un código más modular, flexible y fácil de mantener. A continuación se describen algunos de los componentes más importantes:

- **DebounceClickListener:** Este componente evita múltiples clics en un corto periodo de tiempo sobre un mismo botón, lo cual es útil en situaciones donde los usuarios pueden hacer clic

repetidamente. El `DebounceClickListener` permite que solo se ejecute la acción del clic una vez dentro de un intervalo de tiempo definido, mejorando la experiencia de usuario y previniendo errores.

- **Interfaz para todos los botones:** Para garantizar consistencia y facilitar la extensión de funcionalidades, se implementó una interfaz común para todos los botones de la aplicación. Esta interfaz define métodos para habilitar o deshabilitar los botones, cambiar su estilo y gestionar eventos de clic de manera uniforme.
- **TransparentButton y MainButton:** Son dos variantes de botones reutilizables. El `TransparentButton` se utiliza principalmente para botones secundarios, donde el fondo es transparente pero el borde y el texto son visibles. Por otro lado, el `MainButton` es para botones primarios con fondo sólido y mayor visibilidad. Ambos botones mantienen un estilo consistente en la aplicación y se pueden personalizar según el contexto.
- **CustomDialogFragment:** Este componente es una implementación personalizada de un `DialogFragment`, utilizado para mostrar diálogos de manera consistente en toda la aplicación. `CustomDialogFragment` permite personalizar el contenido, la apariencia y las acciones disponibles para el usuario, como confirmaciones o advertencias.
- **Interfaz para todos los inputs:** Se creó una interfaz común para los campos de entrada (*inputs*) de la aplicación, lo que asegura un comportamiento coherente en todos los campos de texto. Esto facilita la validación de datos y el manejo de errores, reduciendo la duplicación de código y mejorando la consistencia visual y funcional.
- **InputEmail:** Componente especializado en la validación de correos electrónicos, asegurando que el usuario ingrese una dirección válida. `InputEmail` detecta errores comunes, como la falta de un símbolo “@” o un dominio incorrecto, e incluye retroalimentación visual para guiar al usuario.
- **InputPassword:** Este componente incluye validaciones centradas en la seguridad, como la longitud mínima de la contraseña, y permite mostrar u ocultar los caracteres ingresados. Esto ayuda al usuario a verificar visualmente su contraseña sin comprometer la seguridad.
- **BottomNavMenu:** Este componente gestiona la navegación entre las principales secciones de la aplicación (*Home, Traducción, Diccionario, Perfil*) mediante una barra de navegación inferior. El `BottomNavMenu` se integra con el `NavController`, permitiendo transiciones fluidas entre los diferentes `Fragments`.
- **CustomProgressBarDialog:** Diálogo personalizado utilizado para mostrar una barra de progreso durante operaciones que pueden tomar tiempo, como la carga de datos. Proporciona una retroalimentación visual clara de las acciones que se están ejecutando en segundo plano.

La creación de estos componentes reutilizables asegura que la aplicación mantenga una estructura limpia y eficiente. Además, facilita realizar modificaciones o mejoras sin tener que duplicar código, permitiendo la escalabilidad del proyecto y asegurando que los cambios se propaguen de manera consistente en toda la aplicación.

Implementación de seguridad

Autenticación: El sistema de autenticación de la aplicación se maneja a través de un servicio externo, el cual garantiza la seguridad en el manejo de credenciales y datos sensibles. Las funciones de inicio de sesión y registro están integradas de forma segura mediante *API* externas.

Cifrado: Los datos del usuario, incluyendo preferencias de configuración y estado de inicio de sesión, se almacenan de forma segura utilizando *SharedPreferences* cifrados. Esto garantiza que la

información personal del usuario se mantenga protegida incluso cuando se almacena localmente en el dispositivo.

Permisos: La aplicación solicita únicamente el permiso para acceder a la cámara, utilizado durante la grabación de videos para traducir a LENSEGUA. Este permiso se solicita solo cuando el usuario accede a la funcionalidad de grabación y se puede revocar en cualquier momento desde la configuración del dispositivo.

Acceso y control de los videos: Los videos grabados se manejan temporalmente en la *caché* de la aplicación, donde se almacenan mientras el usuario decide si desea guardarlos o eliminarlos. Estos videos solo se almacenan en el servidor si el usuario elige explícitamente la opción de *guardar*. En caso de que el usuario opte por eliminarlos, se borran tanto del dispositivo como del servidor, garantizando el control total de los videos por parte del usuario y su seguridad.

Desafíos técnicos y soluciones

Uno de los principales desafíos técnicos enfrentados durante el desarrollo de la aplicación fue la implementación de la funcionalidad de grabación de video. A continuación, se detallan los problemas específicos encontrados y las soluciones implementadas:

1. **Manejo de permisos:** Al ser una funcionalidad que requiere el uso de la cámara, fue necesario gestionar los permisos de acceso de manera adecuada. Para asegurar que el permiso de la cámara fuera solicitado de forma eficiente y solo cuando se necesitara, se implementó un sistema que solicitaba el permiso en el momento preciso y permitía al usuario revocarlo fácilmente desde la configuración del dispositivo.
2. **Corrección de la deformación del video:** Inicialmente, los videos grabados aparecían estirados o alargados en ciertas pantallas. Para resolver este problema, se integró la librería **CameraX**, que permite un mejor manejo de la cámara en dispositivos Android, asegurando que los videos mantuvieran sus proporciones correctas independientemente de la resolución del dispositivo.
3. **Botón de grabación personalizado:** Se diseñó un componente propio para el botón de grabación que muestra visualmente el progreso de tiempo durante la grabación, con un límite máximo de 15 segundos. Este botón fue implementado desde cero, utilizando animaciones que indicaban al usuario cuánto tiempo de grabación le quedaba antes de alcanzar el límite máximo.
4. **Almacenamiento temporal del video:** Para facilitar el flujo de la aplicación entre los diferentes fragmentos, fue necesario almacenar el video temporalmente. Para evitar solicitar permisos adicionales al usuario, el video se guardó en la *caché* local de la aplicación, lo que permite que el video esté disponible sin la necesidad de acceder al almacenamiento externo del dispositivo.
5. **Reproducción del video:** Al mostrar el video en la aplicación, surgieron problemas de deformación o errores al cargar los archivos de video. Para garantizar una reproducción fluida y eficiente, se decidió utilizar la librería **ExoPlayer**. Esta herramienta no solo resolvió los problemas de visualización, sino que también resultó ser más óptima en cuanto a rendimiento, asegurando que los videos se cargaran correctamente y se mostraran sin distorsiones.

Cambio de contraseña y *Deeplink*

Con el objetivo de facilitar a los usuarios la posibilidad de cambiar su contraseña de manera sencilla y directa, se implementó un flujo utilizando *deeplink* en la aplicación *Señas Chapinas*. Un

deeplink es una *URL* especial que permite abrir una aplicación móvil directamente a una pantalla específica, pasando parámetros que la aplicación puede utilizar para mostrar contenido personalizado o realizar ciertas acciones [82]. En este proyecto, el uso del *deeplink* permite que, al hacer clic en un enlace enviado por correo electrónico, el usuario sea llevado directamente a la pantalla de cambio de contraseña en la aplicación, con su correo ya preescrito para facilitar el proceso.

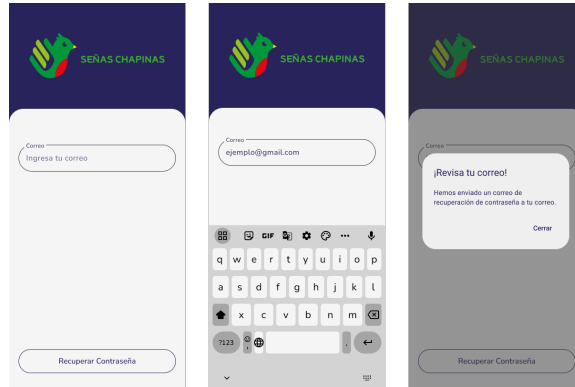


Figura 113: Flujo cambio de contraseña

Proceso de desarrollo

- **Definición del *Deeplink*:** Primero, se configuró el *deeplink* en el archivo `AndroidManifest.xml` de la aplicación. Esto incluyó definir el esquema del *deeplink* (`senaschapinas://`), el host (`journeys`), y la ruta (`/cambio-contr`). Esta configuración permite que la aplicación reconozca y maneje los *deeplinks* que coincidan con este patrón.
- **Implementación en la aplicación:** Dentro de la aplicación, se escribió la lógica necesaria para manejar el *deeplink*. Esto implica capturar el intento cuando se abre la aplicación a través del *deeplink* y extraer el parámetro del correo electrónico que se incluye en la *URL* del *deeplink*. Luego, se navega automáticamente a la pantalla de cambio de contraseña, pasando el correo electrónico como parámetro para que se muestre preescrito.
- **Creación de la página intermedia:** Se desarrolló una página web alojada en *GitHub Pages*. Esta página tiene el propósito de redirigir al *deeplink* de la aplicación. Cuando el usuario accede a esta página, se ejecuta un *script* que redirige automáticamente al *deeplink* con el correo del usuario como parámetro. Esto permite que los enlaces en el correo electrónico sean *URLs* estándar (`https://`), que son más confiables y compatibles con los clientes de correo electrónico.

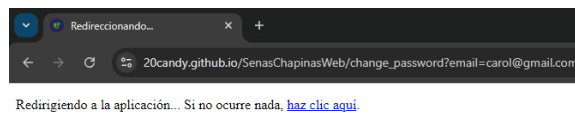


Figura 114: Deeplink página web

- **Integración con correo electrónico:** Se diseñó un correo electrónico para la recuperación de contraseñas, el cual incluye un enlace que lleva a la página intermedia de *GitHub Pages*. Este enlace contiene el correo electrónico del usuario como un parámetro en la *URL*. Al hacer clic en este enlace, el usuario es redirigido automáticamente a la aplicación móvil *Señas Chapinas* mediante el *deeplink*, lo que lleva a la pantalla de cambio de contraseña con el correo preescrito.

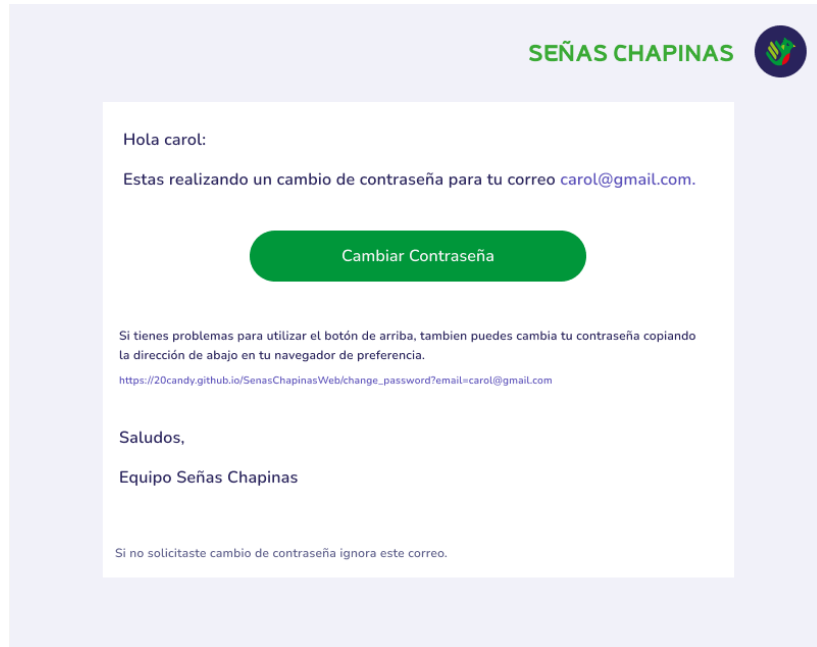


Figura 115: Plantilla correo

- **Envío de correos utilizando SMTP:** Para el envío de correos electrónicos automáticos, se implementó un proceso utilizando el servidor SMTP de Gmail. El sistema genera un correo personalizado para cada usuario, extrayendo automáticamente su nombre a partir de la dirección de correo electrónico. La implementación se realizó mediante la librería `smtplib` de Python, que establece una conexión segura con el servidor de Gmail y utiliza una contraseña de aplicación para autenticarse.

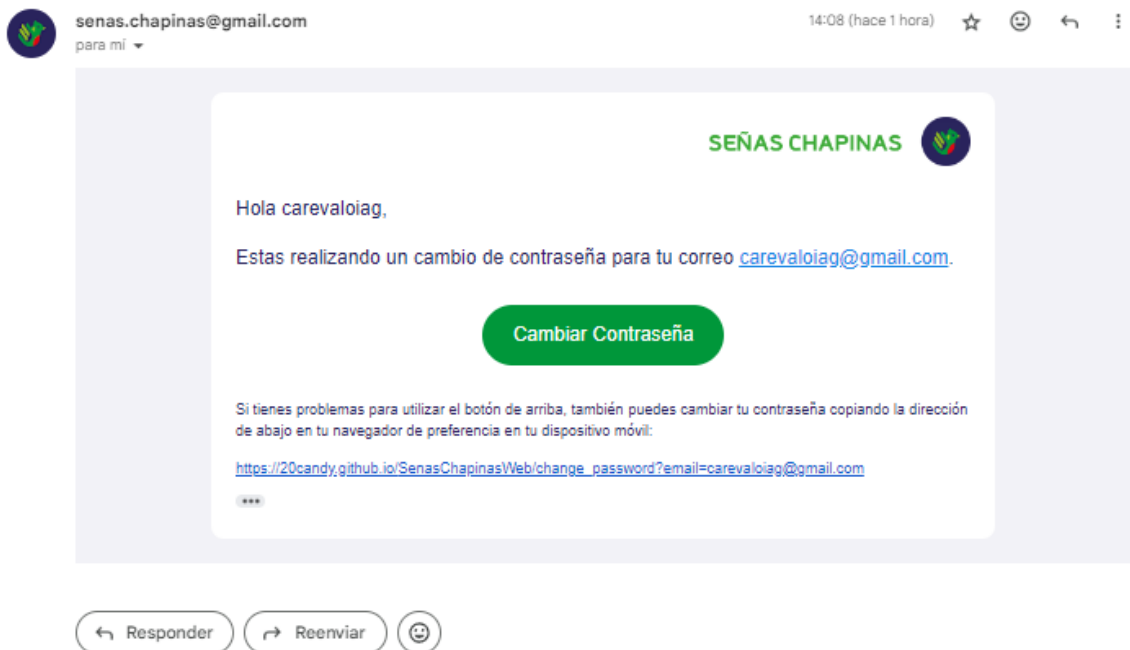


Figura 116: Ejemplo de correo enviado

La implementación de *deeplinks* mejora la experiencia del usuario al simplificar el proceso de restablecimiento de contraseñas. Los usuarios no tienen que navegar manualmente a la pantalla correcta dentro de la aplicación ni ingresar su correo electrónico. Esto hace que el proceso sea más rápido, fácil y menos propenso a errores. Además, la utilización de una página web intermedia para redirigir al *deeplink* garantiza una mayor compatibilidad con diferentes clientes de correo electrónico y navegadores.

Implementación de servicios

Para implementar los servicios en la aplicación móvil, se ha seguido un proceso estandarizado utilizando la librería *Retrofit*. Esta facilita la comunicación con el servidor, permitiendo realizar solicitudes *HTTP/HTTPS* de forma eficiente y segura, ya que cifra las comunicaciones en tránsito. A continuación, se describe el proceso general de implementación:

Proceso de implementación de servicios con *Retrofit*

1. Definición de la API (*interface*):

- Se creó una interfaz para definir los *endpoints* de la API. Aquí especificamos el tipo de solicitud (*GET*, *POST*, *DELETE*, etc.), la ruta del endpoint, y los parámetros requeridos.

2. Creación de las clases *Request* y *Response*:

- **Request:** Se definió una clase para representar el cuerpo de la solicitud. Incluye todos los parámetros necesarios que se enviarán al servidor.
- **Response:** Se creó otra clase para modelar la respuesta del servidor. Esta clase contendrá los campos que el servidor enviará de vuelta.

3. Implementación del *Repository Service*:

- Se creó un servicio de repositorio que utiliza *Retrofit* para realizar la solicitud al servidor. Aquí es donde se llama al método definido en la interfaz de la API y se maneja la respuesta.

4. Actualización del *ViewModel*:

- En el *ViewModel* correspondiente, se llama al método del *Repository Service* y se exponen los resultados para que puedan ser observados por los *fragments* o actividades.

5. Implementación en el *Fragment*:

- Finalmente, en el fragmento correspondiente, se observan los datos expuestos por el *View-Model* y se actualiza la UI según la respuesta recibida.

Publicación en *Play Store*

Una vez finalizados los procesos de diseño, implementación del diseño en Android y la integración de los servicios, se decidió proceder con la publicación de la aplicación en Play Store. Este paso es fundamental para poner la aplicación a disposición del público en general y requiere seguir un proceso estándar establecido por Google.

Creación de la cuenta de desarrollador

El primer paso para publicar en Play Store es la creación de una cuenta de desarrollador en Google Play Console. Este proceso implicó:

- **Registro de la cuenta:** Se requiere un correo electrónico que será asociado con la cuenta de desarrollador.
- **Pago de la cuota de registro:** Google cobra una tarifa única de 25 USD para establecer la cuenta de desarrollador. Este es un pago necesario para acceder a las herramientas y recursos que Google proporciona para gestionar las aplicaciones.
- **Verificación de identidad:** Para garantizar la seguridad y autenticidad, Google solicitó una verificación de identidad, lo que implica proporcionar documentación personal.

Configuración de la aplicación en Google Play Console

Una vez creada la cuenta, se procede a configurar la aplicación en la Play Console:

- **Información de la aplicación:** Se llena un formulario detallado que incluye:
 - **Título y descripción:** Un título breve y una descripción detallada de la aplicación, destacando sus principales características y funcionalidades.
 - **Clasificación de contenido:** A través de un cuestionario, Google clasifica el contenido de la aplicación según la audiencia objetivo. Esto ayuda a determinar la edad mínima para la que la aplicación es apropiada.
 - **Categoría y etiquetas:** Selección de la categoría (en este caso, comunicación, educación) y etiquetas que mejor describen la aplicación.
 - **Política de privacidad:** Google requiere una política de privacidad que informe a los usuarios sobre cómo se manejarán sus datos. Esta política debe ser clara y debe enlazarse desde la ficha de la aplicación en Play Store. Para ello fue necesario el desarrollo de una [página web](#) utilizando *Github Pages*. Adicional a las políticas se decidió incluir información básica de la aplicación.

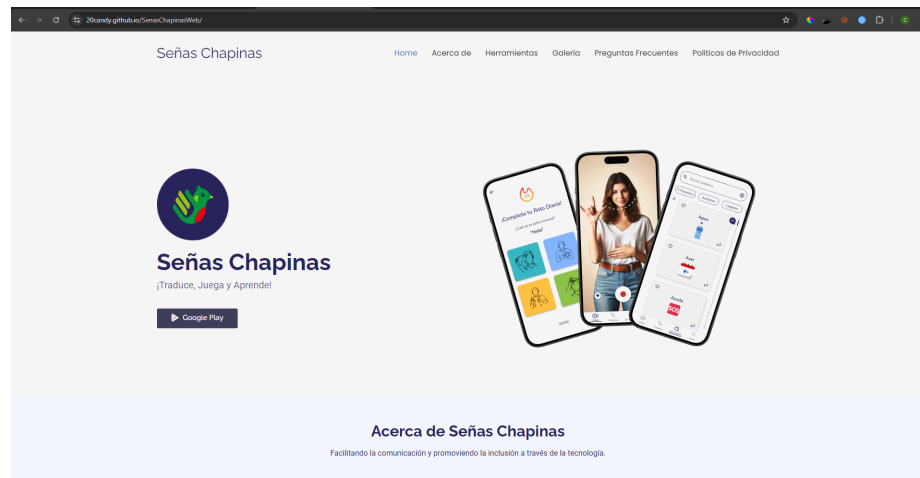


Figura 117: Página web

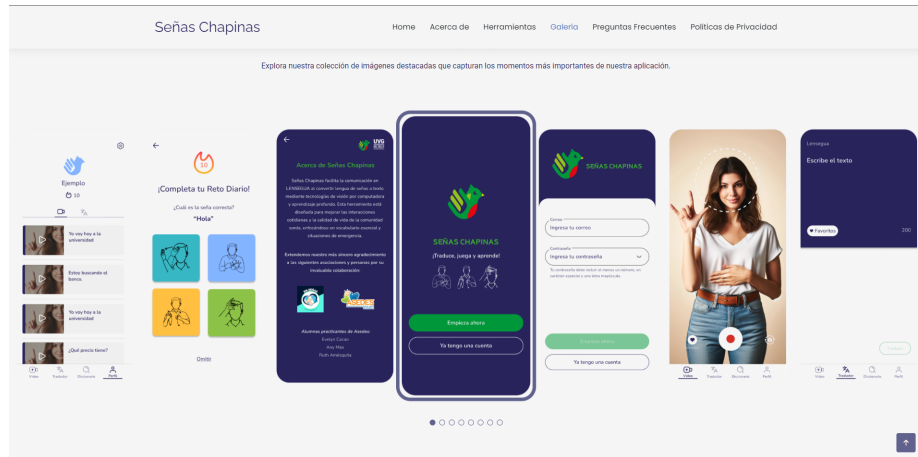


Figura 119: Fotos de página web

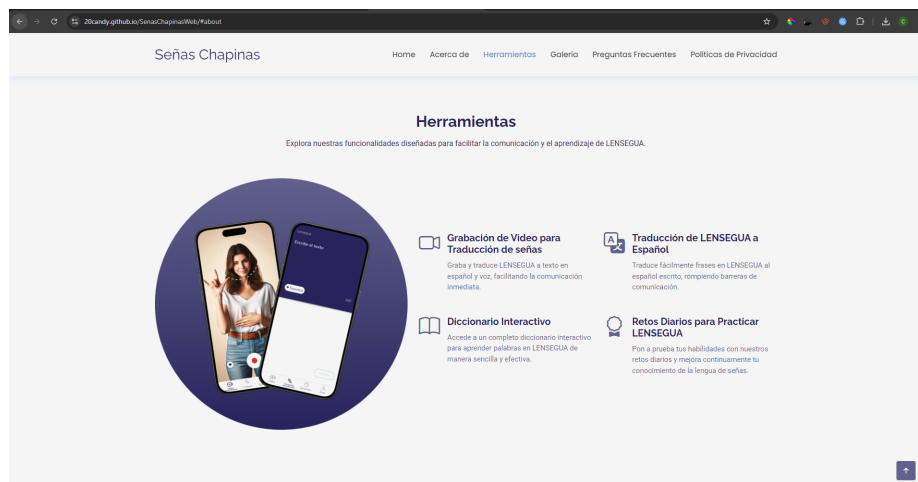


Figura 118: Herramientas de página web



Figura 120: Políticas de privacidad

- **Íconos, imágenes y vídeos:** Fue necesario diseñar elementos adicionales para la publicación de la aplicación.
 - **Ícono de la aplicación:** Un ícono de alta resolución que representará la aplicación en Play Store.



Figura 121: Ícono

- **Capturas de pantalla:** Imágenes de la aplicación en acción, mostrando sus funcionalidades y diseño.

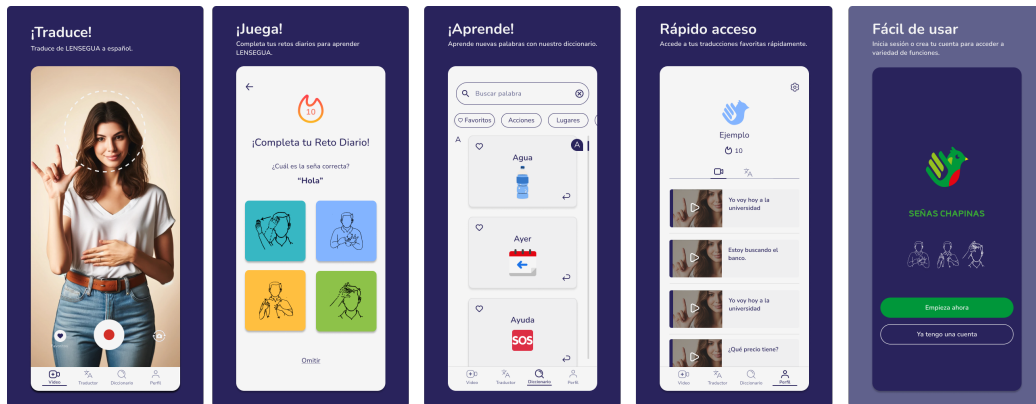


Figura 122: Capturas de pantalla

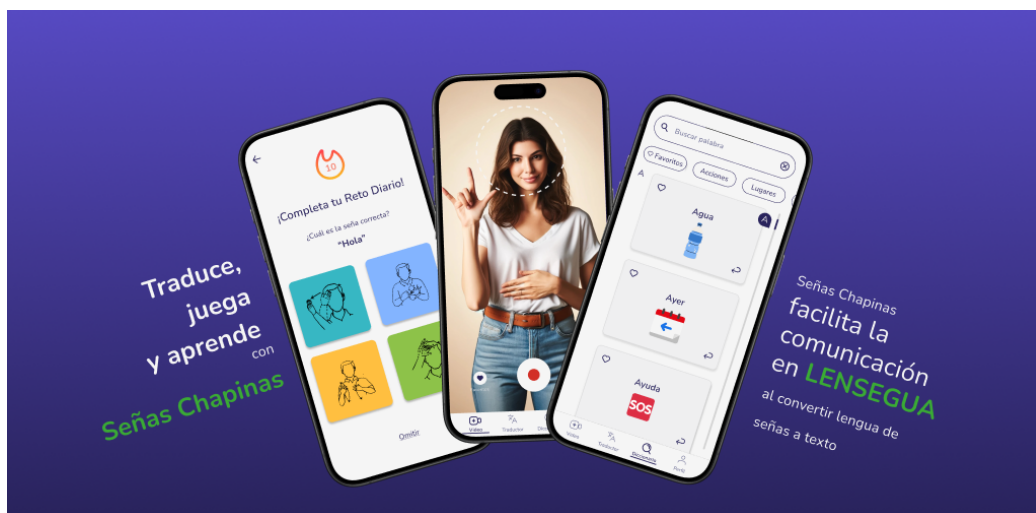


Figura 123: Banner promocional

Configuración de funcionalidades adicionales

- **Política de privacidad y eliminación de datos:** Además de la política de privacidad, Google requiere que las aplicaciones ofrezcan a los usuarios la opción de eliminar sus datos. Se desarrolló una pestaña en la página web que incluye una sección dedicada a la eliminación de datos, permitiendo a los usuarios solicitar la eliminación de su información de forma directa.



Figura 124: Preguntas frecuentes

- **Opciones de distribución:** Configuración de la distribución de la aplicación, como:
 - **Países y regiones:** Selección de los países donde se quiere que la aplicación esté disponible. En este caso se selecciona solo Guatemala.
 - **Dispositivos y programas de Google:** Configuración de si la aplicación estará disponible para dispositivos Wear OS, Android TV, etc. En este caso se escoge para dispositivos móviles únicamente.

Pruebas internas y cerradas

- **Prueba interna:** Inicialmente, se distribuyó la aplicación a un grupo selecto de miembros del equipo para pruebas internas. Esto permitió detectar problemas o errores antes de una prueba más amplia.
- **Prueba cerrada:** Luego, la aplicación fue publicada como una prueba cerrada, un paso previo a la publicación completa. Esta etapa permite la retroalimentación de un grupo más amplio de usuarios y es un requisito de Google que la aplicación esté en esta etapa al menos 15 días antes de su publicación en producción. Durante este tiempo, es necesario obtener la aprobación de al menos 20 desarrolladores que prueben la aplicación y verifiquen su funcionalidad y seguridad.

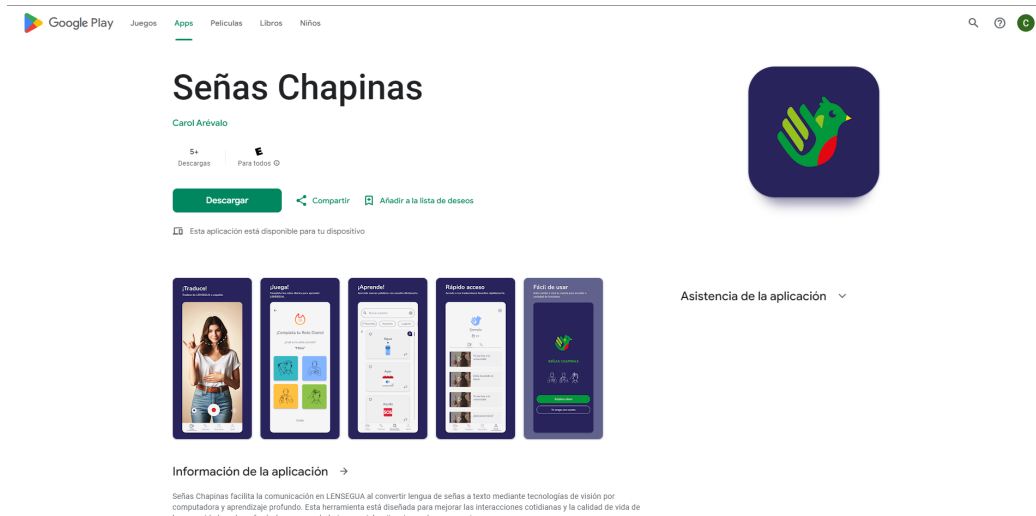


Figura 125: Prueba cerrada Play Store

- **Solicitud acceso a producción:** Luego de finalizar los 15 días de prueba se solicito acceso a liberar la aplicación en tiendas de Google a nivel productivo.

Producción



Solicitar acceso a producción

El canal de producción es donde pones tu aplicación a disposición de miles de millones de usuarios en Google Play. Para poder solicitar acceso a producción, debes realizar una prueba cerrada que cumpla nuestros criterios. Al enviar tu solicitud, también deberás responder a algunas preguntas sobre la prueba cerrada.

🕒 Hemos recibido tu solicitud de acceso a producción

Estamos revisando tu formulario de solicitud. Enviaremos un correo al propietario de la cuenta con las novedades. Este proceso suele tardar 7 días o menos, pero en ocasiones puede llevar más tiempo.

Solicitud enviada el hoy, 11:37.

Figura 126: Solicitud para producción

Uso de Kanban

En el desarrollo de la aplicación *Señas Chapinas*, se utilizó un sistema Kanban^[1] para la gestión y organización de tareas, facilitando un flujo de trabajo visual y eficiente. Aunque el desarrollo fue llevado a cabo de manera individual, este enfoque permitió organizar de forma clara las tareas y priorizar los entregables según las necesidades del proyecto.

El sistema Kanban se organizó en historias de usuario, donde cada una representaba una tarea específica del proyecto. Cada historia incluía una descripción detallada para asegurarse de que no se perdiera ningún detalle importante durante la implementación.

¹Kanban es un método visual para la gestión de proyectos que utiliza tarjetas para representar las tareas y columnas para mostrar las etapas del trabajo [19].

Proyectos / Mi proyecto de Kanba... / KAN-14 / KAN-3

Pantalla de Inicio

Adjuntar Añadir una incidencia secundaria Vincular incidencia Añadir diseño Crear

Descripción

Como (Rol)	Desarrolladora		
Necesito (Funcionalidad)	Implementar la pantalla Inicio		
Para (Beneficio)	Darle la bienvenida a la aplicación al usuario		
Criterios de aceptación / Escenarios			
Identificador	Dado	Cuando	Entonces
1	El cliente	Abra la aplicación Señas Chapinas	Debe mostrarse el la pantalla de inicio con el logo de Señas Chapinas.
2	El cliente	Muestre la pantalla de Inicio	Debe mostrar el subtítulo "Traduce, juega y aprende" es español y Lensegua
3	El cliente	Muestre la pantalla de Inicio	Debe mostrar los botones: Empieza Ahora y Ya tengo una Cuenta
4	El cliente	De tap en el botón Empieza Ahora	Debe llevarlo al flujo de Sign Up.
5	El cliente	De tap en el botón Ya tengo una cuenta.	Debe llevarlo al flujo de Login.
Figma	Señas Chapinas		
Notas:			

Figura 127: Ejemplo historia usuario

Las fases del proyecto incluyeron:

- **Arquitectura:** Se planificó y diseñó la estructura principal del proyecto, incluyendo la implementación de patrones de diseño y la creación del `MainActivity` y los `NavGraphs`.
- **Pantallas principales:** El desarrollo de pantallas se realizó de manera secuencial, comenzando por *Sign Up*, *Login*, *Home*, *Vídeo*, *Reporte*, *Traductor*, *Diccionario*, *Perfil*, *Configuración*, *Reto Diario* y *Cambio de Contraseña*. Cada pantalla fue gestionada como una tarea independiente dentro del sistema Kanban, lo que permitió un avance estructurado en el desarrollo.
- **Componentes:** Esta fase involucró la creación de componentes reutilizables que se aplicaron a lo largo de toda la aplicación, como botones y campos de texto personalizados.

El uso de Kanban permitió mantener una organización eficiente del trabajo, identificando posibles cuellos de botella y ajustando el cronograma de desarrollo de manera efectiva. De esta forma, se logró avanzar de manera rápida y adaptable a los cambios y solicitudes durante el proceso.

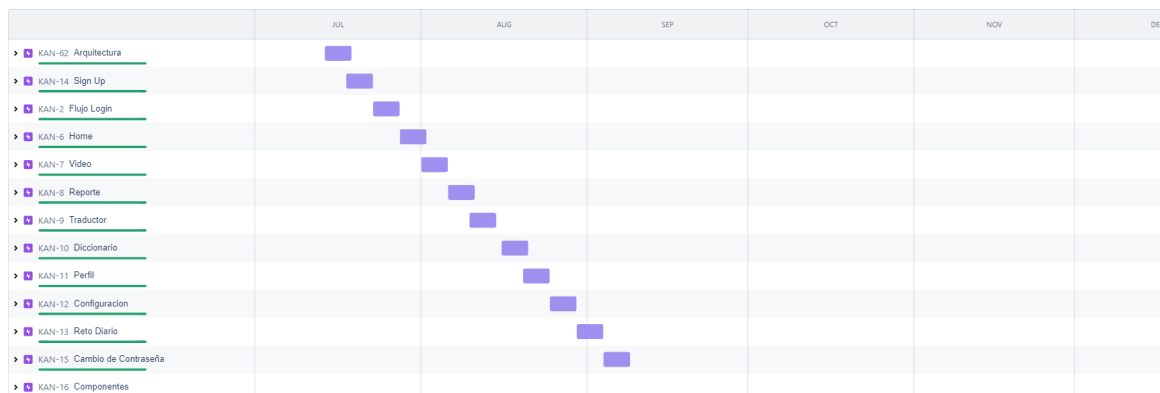


Figura 128: Cronograma Kanban

5.5.4. Pruebas con usuarios finales

Primera prueba - EXPO UVG

Después de finalizar la primera versión funcional de la aplicación *Señas Chapinas*, se presentó el proyecto en la *Expo UVG*. Este evento permitió interactuar directamente con usuarios potenciales y recibir valiosas sugerencias tanto de expertos en diseño como de personas interesadas en mejorar la accesibilidad para la comunidad sorda. Durante la exposición, la aplicación fue muy bien recibida por la mayoría de los asistentes, quienes no solo mostraron gran interés en su funcionalidad, sino que también destacaron su relevancia como una herramienta inclusiva.

Uno de los aspectos más destacados fue que muchas personas preguntaron cuándo estaría disponible en tiendas de Android, señalando su interés personal en utilizarla. Algunos asistentes mencionaron que veían la aplicación como una herramienta que podría ayudar a muchas personas en su vida cotidiana, especialmente en entornos donde la comunicación con personas sordas es limitada. Una persona en particular, que tiene familiares sordos, comentó que *Señas Chapinas* le sería de gran utilidad para aprender LENSEGUA y así poder comunicarse mejor con sus seres queridos.

Además, se mencionó que una gran parte de la población no tiene conocimiento sobre qué es LENSEGUA, por lo que los asistentes consideraron que la aplicación no solo facilitaría la comunicación, sino que también podría jugar un papel fundamental en la concientización de los guatemaltecos sobre la importancia de la lengua de señas en la sociedad. Muchos estuvieron de acuerdo en que sectores como los cajeros de supermercados, empleados de atención al cliente y otras ocupaciones en las que se interactúa directamente con el público podrían beneficiarse enormemente de esta herramienta, al mejorar la comunicación con personas sordas y fomentar un entorno más inclusivo.

Finalmente, se concluyó que *Señas Chapinas* no solo podría ser utilizada como una herramienta tecnológica de apoyo, sino también como un recurso educativo que ayude a la población guatemalteca a familiarizarse con la Lengua de Señas Guatemalteca y promover un mayor entendimiento y respeto hacia la comunidad sorda.

Las siguientes recomendaciones fueron implementadas tanto a nivel de diseño como en el funcionamiento de la aplicación:



Figura 129: Expo UVG

- **Pantalla de inicio:** Se modificó el estilo del botón de grabación para mostrar el tiempo máximo de grabación disponible, facilitando al usuario la gestión de su video.
- **Pantalla de reporte:** Se añadieron miniaturas del video grabado para que el usuario pueda

seleccionar el momento exacto en el que la seña fue mal interpretada. También se incorporó un campo de texto donde el usuario puede proporcionar una descripción detallada del error. Además, se ajustó el texto de los botones para que resulten más intuitivos y claros para el usuario.

- **Otros ajustes:** Se realizaron pequeños ajustes en el tamaño de la letra, los colores de los íconos, entre otros, siempre con el objetivo de mejorar la experiencia de usuario.

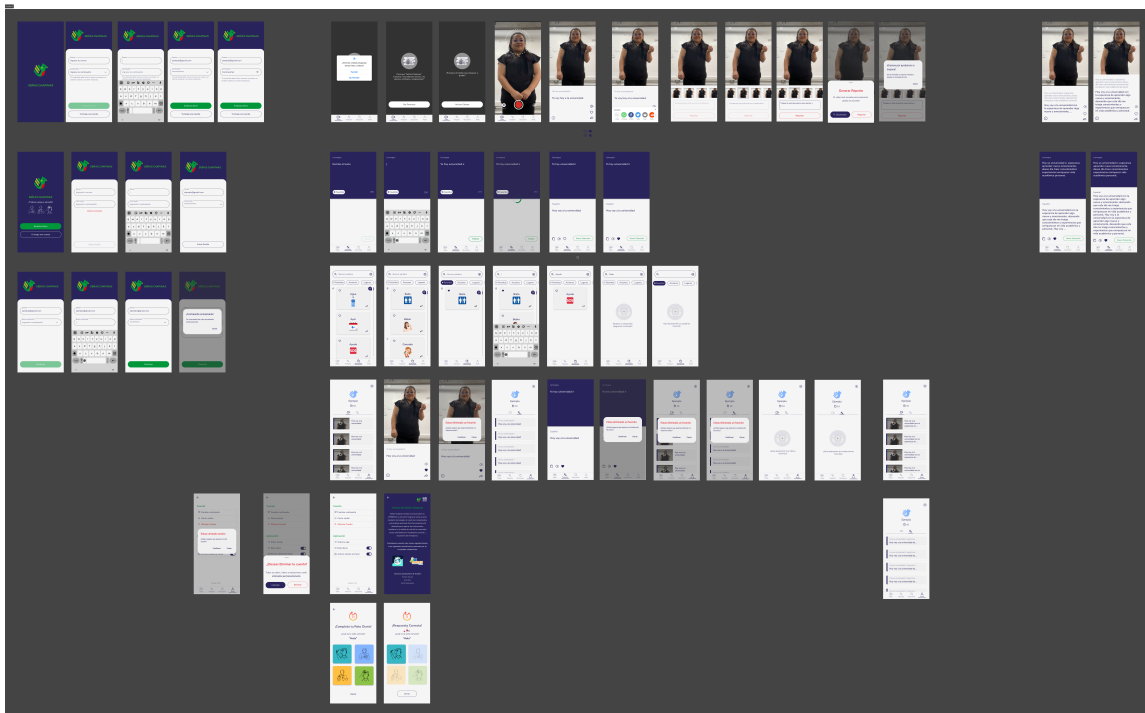


Figura 130: Cambios primera prueba con usuarios

Segunda prueba - *En-Señas*

Durante todo el proceso de creación de este proyecto, la asociación En-Señas estuvo presente de manera constante, aportando sus ideas y comentarios periódicamente para mejorar iterativamente el producto final de *Señas Chapinas*. Por ende, no podía faltar una presentación final con ellos. En esta sesión, presentamos la aplicación con todos los módulos integrados, recibiendo valiosas opiniones, sugerencias y recomendaciones finales.

Comenzamos la presentación mostrando cada parte de la aplicación: la grabación de video, la traducción de frases en LENSEGUA, el diccionario y algunas configuraciones.

En primer lugar, el logo recibió elogios, ya que fue descrito como muy atractivo y representativo del concepto de *Señas Chapinas*, al combinar de manera armoniosa un quetzal con una mano. Los colores de la aplicación también fueron apreciados, calificándolos como llamativos pero sin perder la seriedad de la propuesta. Se destacó que el uso de colores más vibrantes en el módulo del reto aportaba un aire interactivo y divertido. Además, mencionaron que el diseño de la app les recordaba aplicaciones que ya utilizan, como TikTok, ChatGPT, Duolingo y traductores, lo que facilitaba su uso.

La funcionalidad de grabación de video fue destacada por su sencillez e intuición. Además, la

opción de reportar un problema en caso de que surgiera alguna complicación fue considerada una adición importante. Como habían sugerido previamente, la aplicación ahora muestra la traducción tanto en LENSEGUA como en español, lo que no solo facilita la comunicación, sino también ayuda a mejorar el español escrito de los usuarios sordos. Nos brindaron ejemplos concretos sobre cómo la aplicación podría ser útil en contextos como bancos, tiendas, restaurantes y situaciones de emergencia, lo que resalta su relevancia.

El módulo traductor fue calificado como más fácil de usar en comparación con ChatGPT y, además, se destacó que proporcionaba mejores traducciones. La organización de las frases, con LENSEGUA arriba y español abajo, permitía un uso más práctico y una mejor comprensión. También sugirieron que se integrara la opción de reportar traducciones incorrectas, tal como en la función de video.

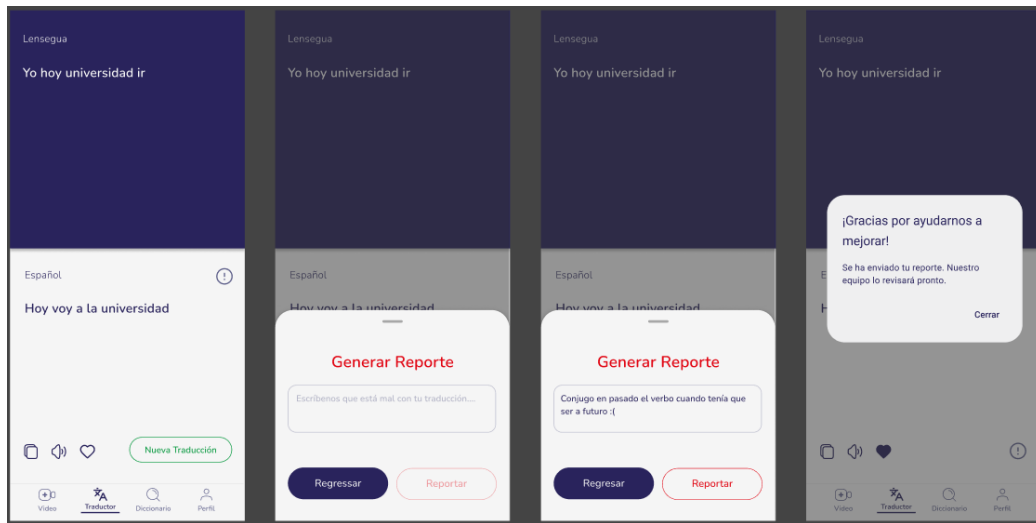


Figura 131: Cambio En-Señas 1

Al mostrar el diccionario, apreciaron que incluya las palabras más necesarias, pero recomendaron que, en futuras actualizaciones, se utilicen videos para hacer el aprendizaje de LENSEGUA más dinámico y menos tedioso.

En el apartado de perfil, les pareció ideal que se pueda acceder rápidamente a traducciones y videos favoritos para situaciones cotidianas y de emergencia, resaltando que muchas personas en Guatemala no saben LENSEGUA, pero las personas sordas necesitan comunicarse de manera rápida y efectiva.

Al mostrar el reto diario, lo calificaron como entretenido y lo relacionaron con actividades que ellos mismos realizan como profesores de señas. Sugirieron que, cuando el usuario acierte en la selección de la palabra correcta, no solo el botón se vuelva más brillante, sino que también aumente de tamaño para que la seña sea más visible. Además, mencionaron que los colores deben mantenerse consistentes en todas las pruebas, para facilitar la comparación de resultados entre usuarios sordos. Por ejemplo, sería más fácil decir “el color azul” en lugar de “el de arriba a la derecha”, ya que señalar posiciones puede ser más complicado en lengua de señas.

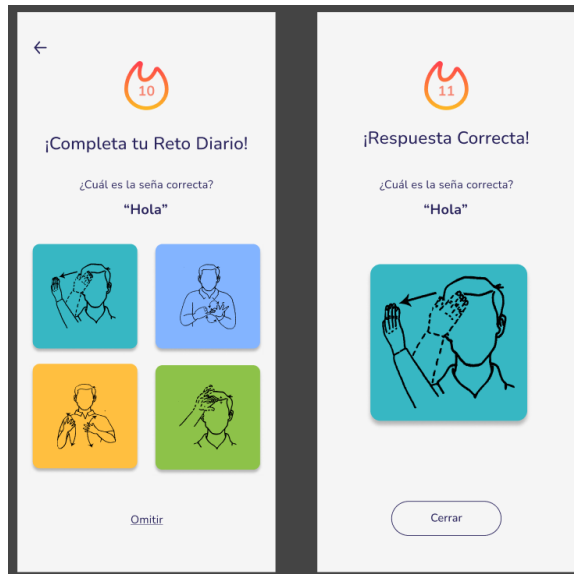


Figura 132: Cambio En-Señas 2

Además, la asociación En-Señas sugirió incluir un tutorial inicial para los usuarios, inspirado en cómo las aplicaciones bancarias presentan sus funcionalidades a través de recorridos guiados por sus distintas secciones, similar a cómo se hizo durante nuestra presentación. La implementación de este tutorial facilitaría a los usuarios comprender y aprovechar al máximo las capacidades de la aplicación desde su primer uso, mejorando significativamente la accesibilidad y la experiencia general.



Figura 133: Fragmento tutorial

Finalmente, agradecieron haber sido incluidos en los agradecimientos de la aplicación y ofrecieron algunas sugerencias para otros los módulos de este proyecto, como el reconocimiento de diferentes señas para una misma palabra y mejoras en el traductor. Señalaron que, para ser una aplicación que trabaja con frases y palabras esenciales, *Señas Chapinas* es una aplicación bastante completa, destacando que la tecnología detrás de este tipo de herramientas marca un avance importante para la inclusión en Guatemala.



Figura 134: Demo En-Señas

Por otra parte, luego de presentar la aplicación a los directivos de En-Señas, se procedió a realizar pruebas de usabilidad guiadas con 25 miembros de la asociación, incluyendo personas sordas, familiares de personas sordas, estudiantes de LENSEGUA, profesores y conocidos de la organización. Estos usuarios interactuaron con la aplicación, realizando una serie de tareas que reflejaban el uso típico en escenarios del día a día.

Se llevaron a cabo pruebas de camino crítico, evaluando las funcionalidades esenciales de la aplicación, utilizando como base los mapas de experiencia del cliente y los flujos de usuarios desarrollados en la etapa de diseño. Las pruebas incluyeron los siguientes flujos de usuario:

- Creación de usuario
- Inicio de sesión
- Grabación de video
- Guardado de video
- Apertura de un video guardado
- Traducción de una frase
- Guardado de una frase traducida
- Apertura de una frase guardada
- Búsqueda de una palabra en el diccionario

- Completar un reto diario

Los resultados de estas pruebas fueron en su mayoría exitosos, con un índice de éxito general del 90%. A continuación, se detallan los resultados obtenidos para cada flujo:

- **Creación de usuario:** un 84% de las personas completaron la tarea sin inconvenientes, reportando que la interfaz era clara y la información solicitada fácil de ingresar. No obstante, algunas personas no sabían que correo utilizar y les fue difícil idear una contraseña en el momento. Como sugerencia, varios usuarios mencionaron que les gustaría contar con la opción de iniciar sesión con Facebook o Google para agilizar el proceso.
- **Inicio de sesión:** El 80% de los participantes logró iniciar sesión correctamente en su primer intento. Algunos usuarios no recordaban sus contraseñas y, en ciertos casos, sus teléfonos no guardaban las credenciales, lo que afectó el porcentaje de éxito en esta tarea.
- **Grabación de video:** El 75% de los usuarios logró grabar un video sin dificultades. Algunos usuarios sordos no comprendieron inicialmente que debían mantener el botón presionado y mencionaron que no les era útil pues tenían que hacer señas con ambas manos. Mientras que los interpretes, quienes pueden realizar señas con una sola mano, no habían notado esa problemática. Tomando esto en cuenta se cambia el botón de grabación para que inicie a grabar con un tap y se detenga con un siguiente tap.

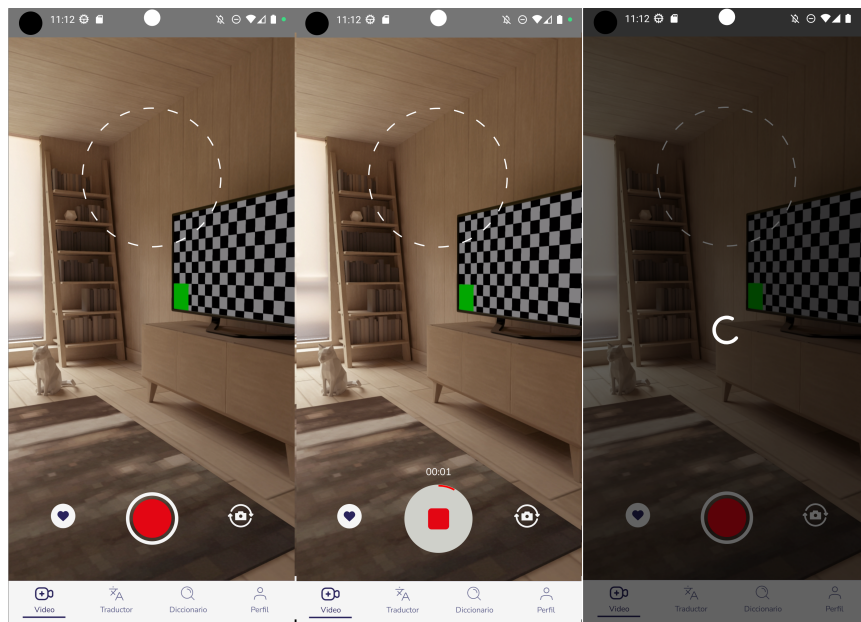


Figura 135: Flujo muestra de grabación

- **Guardado de video:** El 92% de los usuarios guardó sus videos con éxito, reportando que el proceso era claro y directo.
- **Apertura de un video guardado:** El 92% de los usuarios pudo acceder a los videos guardados fácilmente, destacando que la organización y el acceso a los archivos eran eficientes.
- **Traducción de una frase:** El 96% de los usuarios completó con éxito la traducción de frases, señalando que la disposición de la traducción en LENSEGUA y español era muy útil.

- **Guardado de una frase traducida:** El 96 % de los participantes logró guardar una frase sin problemas. Los usuarios mencionaron que el ícono de guardado era el mismo que se usa para los videos, lo que lo hacía más obvio e intuitivo, destacando la importancia de mantener una iconografía y un diseño visual coherentes en toda la aplicación.
- **Apertura de una frase guardada:** El 96 % de los usuarios accedió a las frases guardadas sin inconvenientes. Algunos usuarios no entendieron inicialmente que había dos botones separados en la interfaz de perfil: uno para los videos guardados y otro para las frases, lo que generó cierta confusión.
- **Búsqueda de una palabra en el diccionario:** El 90 % de los participantes logró realizar búsquedas exitosas, encontrando las palabras que necesitaban de manera rápida. Algunos usuarios comentaron que el desplazamiento rápido en la lista de palabras fue un poco confuso al principio, pero una vez comprendido, lo encontraron útil.
- **Completar un reto diario:** El 92 % de los usuarios completó el reto diario sin problemas, señalando que el sistema de colores y el *feedback* visual al seleccionar la respuesta correcta era efectivo. Se recomendó aumentar el tamaño de la seña correcta, como se sugirió anteriormente, para mejorar la visibilidad.

En general, las pruebas confirmaron que el diseño y flujo de la aplicaciones altamente intuitivo, lo que justifica las decisiones tomadas durante el proceso de desarrollo. La aplicación ha demostrado ser accesible y fácil de usar, permitiendo a la comunidad sorda y oyente en Guatemala interactuar de manera efectiva y eficiente en sus actividades diarias.

6.1. Visión por computadora

6.1.1. Conjunto de datos

Datos de lengua de señas de Guatemala

La recopilación de datos era un objetivo importante para este proyecto, debido a que no existen conjuntos de datos públicos de lengua de señas de Guatemala. Esto representaba un desafío importante para el desarrollo de un modelo de reconocimiento de lengua de señas, ya que no se contaba con datos para entrenar un modelo de aprendizaje. Siguiendo la metodología propuesta para el proyecto, se recopilaron datos de 32 palabras de la lengua de señas de Guatemala, las cuales son: *agua, ayer, ayudar, baño, beber, cansado, casa, colegio, comer, como, cuando, cuanto, donde, hacer, hambre, hospital, hoy, ir, llamar, necesito, policía, por favor, que, quien, quiero, sed, tengo, trabajo, tu, universidad, yo, y el signo de interrogación.*

El conjunto de datos está compuesto de 30 videos por palabra, los cuales fueron editados manualmente para incluir solo el gesto de la palabra correspondiente. En total, se recopilaron 960 videos, los cuales fueron divididos en dos conjuntos: uno de entrenamiento y otro de prueba. Utilizando una proporción de 80% para el conjunto de entrenamiento y 20% para el conjunto de prueba. Con esta proporción, el conjunto de entrenamiento está compuesto de 768 videos, mientras que el conjunto de prueba está compuesto de 192 videos. Con el objetivo de permitir que otros investigadores puedan utilizar el conjunto de datos, se ha publicado en el siguiente enlace: <https://github.com/Ldsc2002/lensegua-dataset>.

Análisis del conjunto de datos

Con base a la matriz de confusión del modelo final, se puede observar que el modelo tiene un buen desempeño en la mayoría de las clases. Sin embargo, se puede observar que el modelo ocasionalmente confunde algunas clases, como *sed* y *quien*. Esto puede deberse a que los gestos de estas palabras son similares, lo cual dificulta la distinción entre ellas. Para validar esta hipótesis, se realizó un análisis de similitud entre las clases, utilizando un análisis de componentes principales (PCA).

Se identificó que el modelo confunde las siguientes clases: *como* y *agua*, *como* y *beber*, *cuando* y *donde*, y *sed* y *quien*. A continuación se presentan los resultados del análisis de similitud entre las clases para cada par de clases.

En la Figura 136 se puede observar que las clases *como* y *agua* tienen una alta similitud en los movimientos de una mano. Aunque la otra mano tiene un movimiento distinto en cada clase, el alto grado de similitud en una mano puede dificultar la distinción entre ellas.

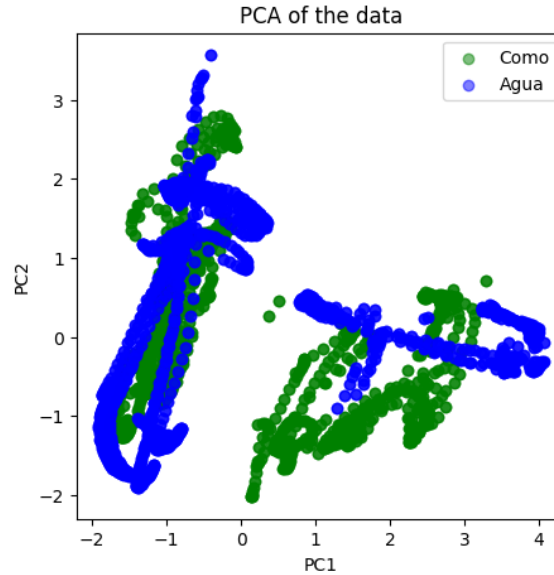


Figura 136: Análisis de similitud entre las clases *como* y *agua*

En la Figura 137 se puede observar que las clases *como* y *beber* tienen cierto grado de similitud en los movimientos de ambas manos. Esto puede dificultar la distinción entre ellas, ya que el modelo puede confundir los movimientos de ambas manos.

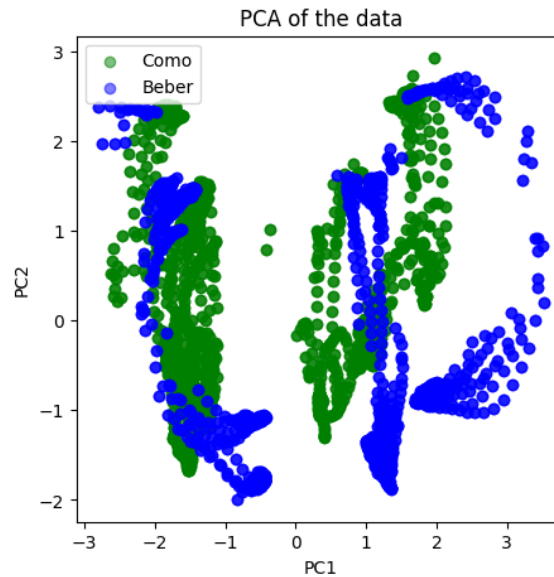


Figura 137: Análisis de similitud entre las clases *como* y *beber*

En la Figura [138](#) se puede observar que las clases *cuando* y *donde* tienen un bajo grado de similitud en los movimientos de ambas manos. Sin embargo, en el análisis de similitud se observa que las clases tienen cierto grado de superposición, lo cual puede dificultar la distinción entre ellas.

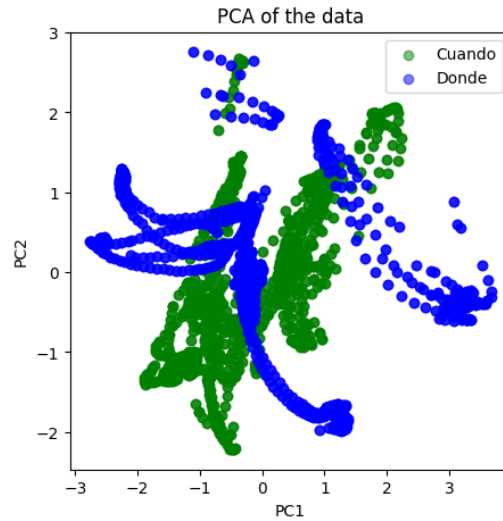


Figura 138: Análisis de similitud entre las clases *cuando* y *donde*

En la Figura [139](#) se puede observar que las clases *sed* y *quien* tienen un bajo grado de similitud en los movimientos de ambas manos. El análisis de similitud no muestra una superposición significativa entre las clases, lo cual indica que el modelo puede confundir las clases debido a otros factores.

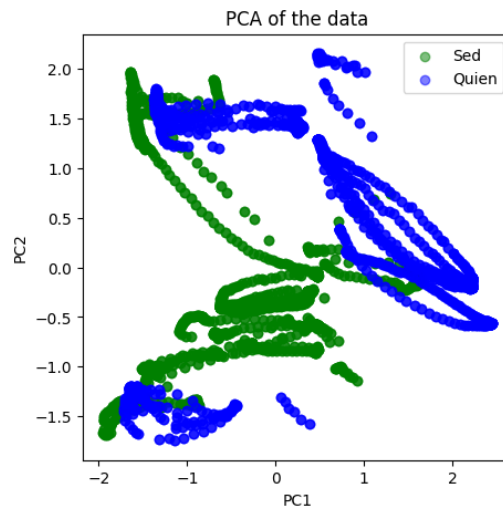


Figura 139: Análisis de similitud entre las clases *sed* y *quien*

Adicional a esto, se realizó un análisis de similitud entre todas las clases, con el objetivo de identificar otras clases que puedan tener un alto grado de similitud. En la Figura [140](#) se puede observar el análisis de similitud entre todas las clases.

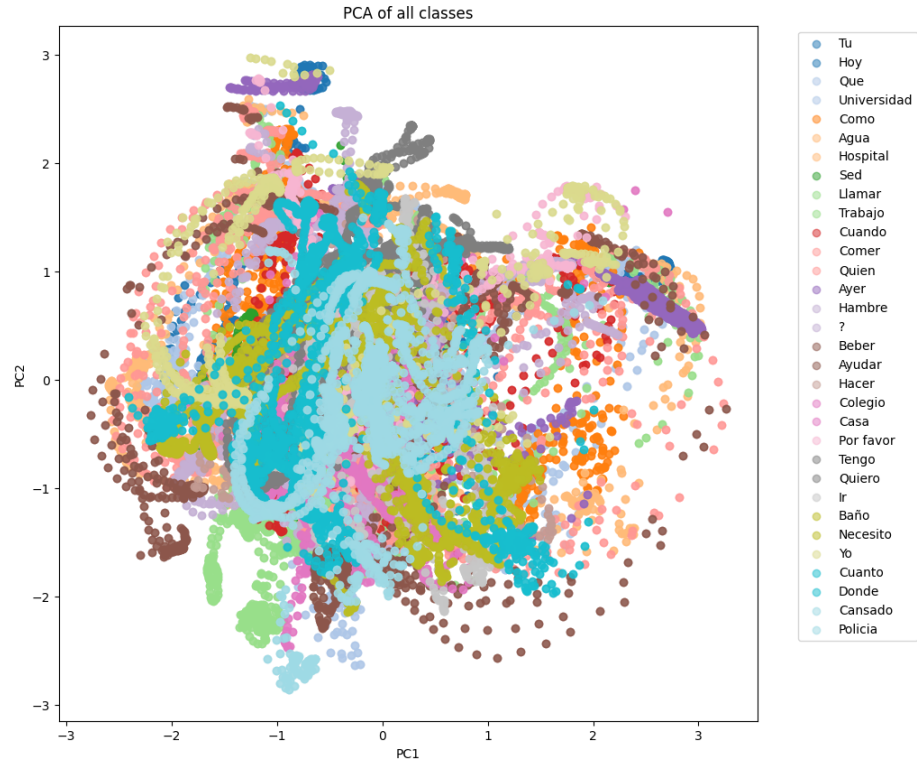


Figura 140: Análisis de similitud entre todas las clases

Por último, se validó el balance de las clases en el conjunto de datos de entrenamiento. La Figura 141 muestra el balance de las clases en el conjunto de datos de entrenamiento.

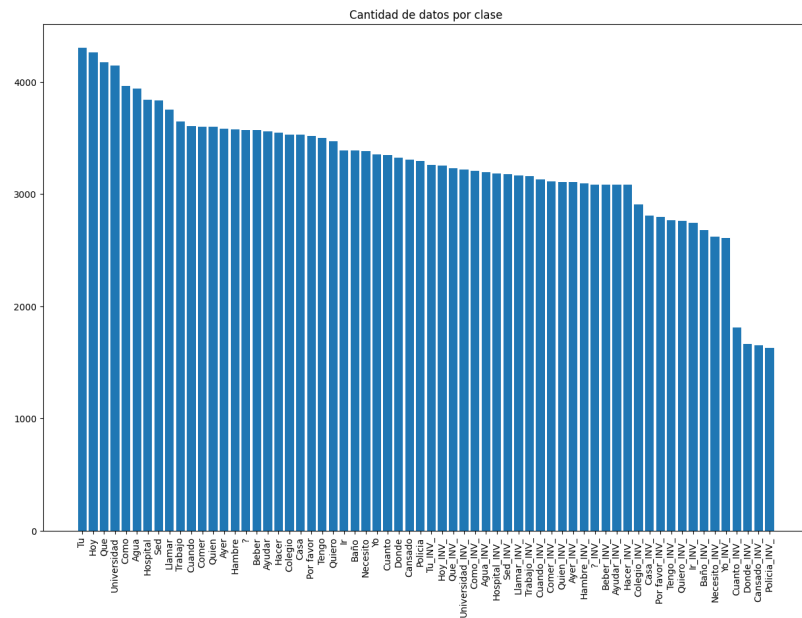


Figura 141: Balance de clases en el conjunto de datos de entrenamiento

6.1.2. Proceso iterativo de desarrollo del modelo

El proceso de desarrollo del modelo de reconocimiento de lengua de señas de Guatemala fue iterativo, siguiendo la metodología propuesta para el proyecto. En total, se realizaron 8 iteraciones, en las cuales se crearon modelos distintos y se evaluaron utilizando el conjunto de prueba. Todos los modelos fueron entrenados con 25 épocas, utilizando un tamaño de lote de 32 y un optimizador Adam. A continuación se presentan los modelos resultantes de cada iteración, junto con su desempeño en el conjunto de prueba.

Modelo base

La composición del modelo base se puede observar en la Figura 142. Este modelo tiene una capa de entrada, cuatro capas densas con activación tangente hiperbólica y una capa de salida con activación softmax.

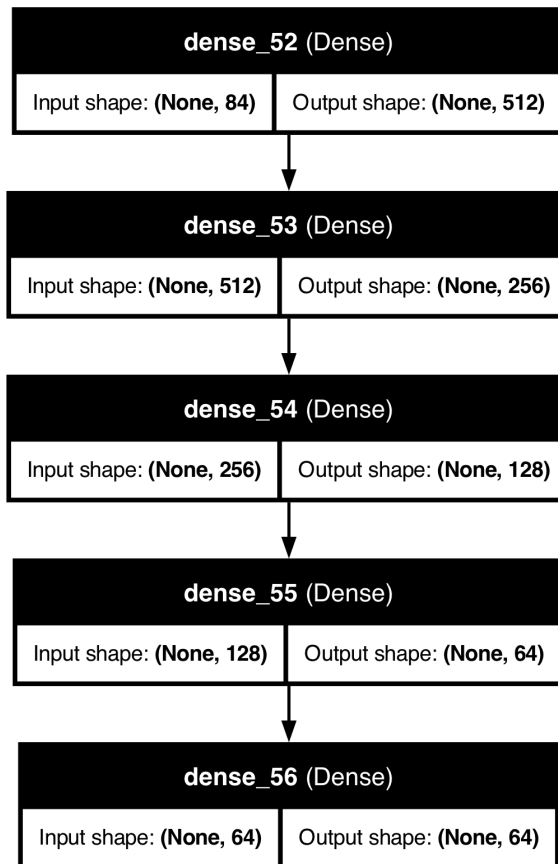


Figura 142: Modelo inicial

El desempeño del modelo inicial se puede observar en la Tabla 15. Este modelo inicial tiene una precisión de 0.7708, una sensibilidad de 0.8822 y un F1 de 0.8895. Adicional a esto, se puede observar la matriz de confusión en la Figura 144 y el historial de entrenamiento en la Figura 143.

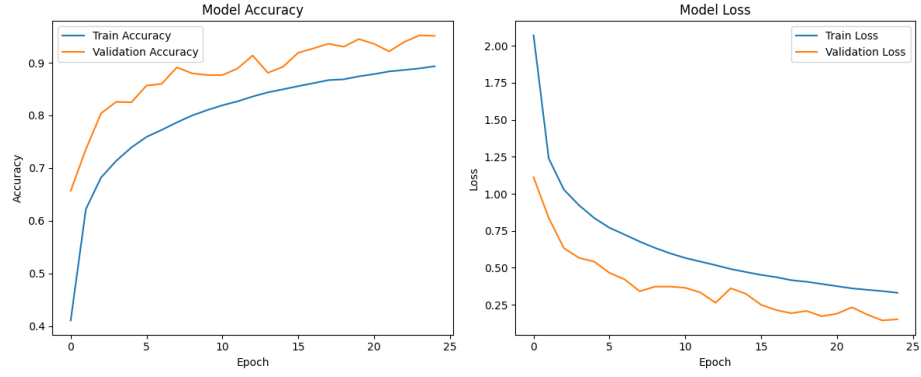


Figura 143: Historial de entrenamiento del modelo base

Métrica	Valor
Precisión	0.7708
Sensibilidad	0.8822
F1	0.8895

Tabla 15: Desempeño del modelo base

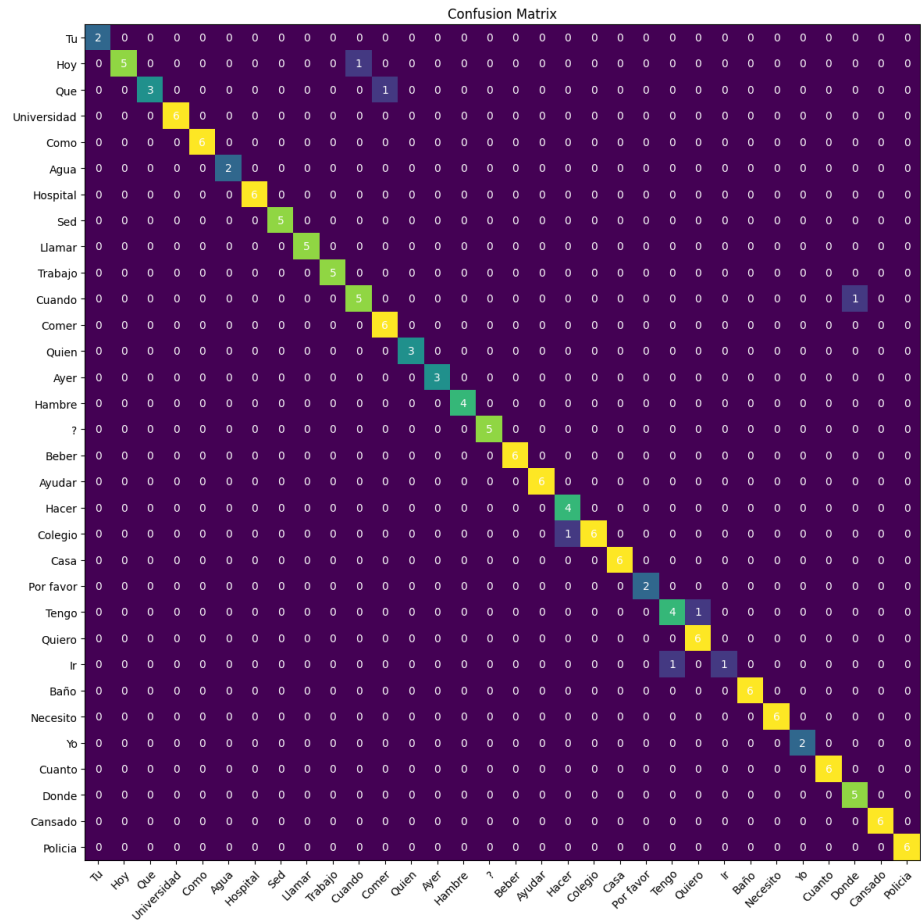


Figura 144: Matriz de confusión del modelo base

Aumento de complejidad del modelo base

En la iteración previa, se observó que el modelo base tenía un desempeño aceptable, pero se consideró que la arquitectura del modelo era muy simple. En esta iteración, se aumentó la complejidad del modelo base, añadiendo dos capas densas adicionales con activación tangente hiperbólica. Estas capas adicionales tienen 2048 y 1024 neuronas, respectivamente.

La composición del modelo con aumento de complejidad se puede observar en la Figura 145. Este modelo tiene una capa de entrada, seis capas densas con activación tangente hiperbólica y una capa de salida con activación softmax.

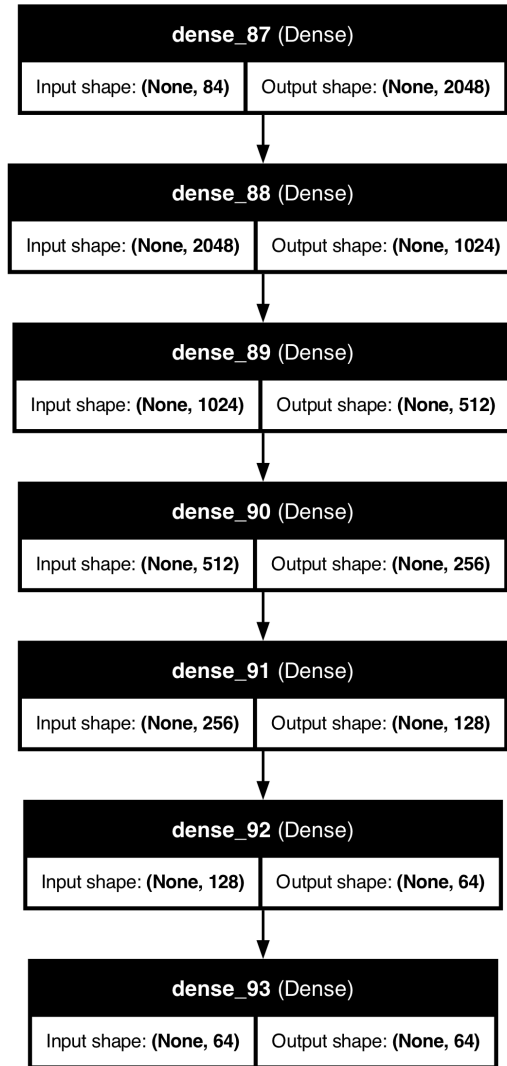


Figura 145: Modelo con aumento de complejidad

El desempeño de este modelo se puede observar en la Tabla 16. Este modelo tiene una precisión de 0.5937, una sensibilidad de 0.8957 y un F1 de 0.9026. Adicional a esto, se puede observar la matriz de confusión en la Figura 147 y el historial de entrenamiento en la Figura 146.

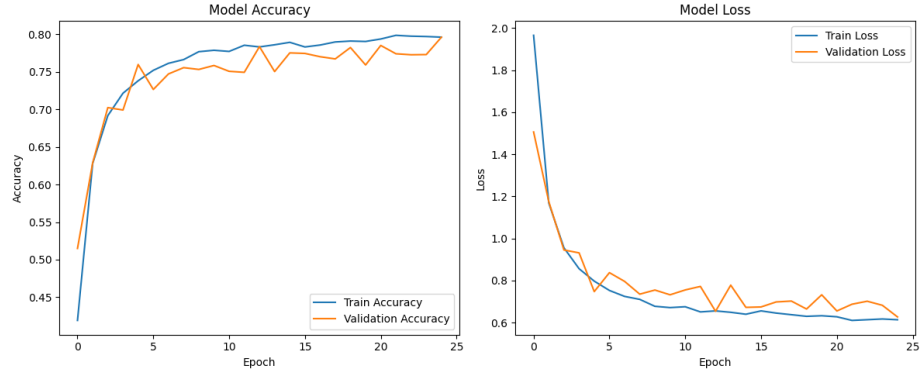


Figura 146: Historial de entrenamiento del modelo con aumento de complejidad

Métrica	Valor
Precisión	0.5937
Sensibilidad	0.8957
F1	0.9026

Tabla 16: Desempeño del modelo con aumento de complejidad

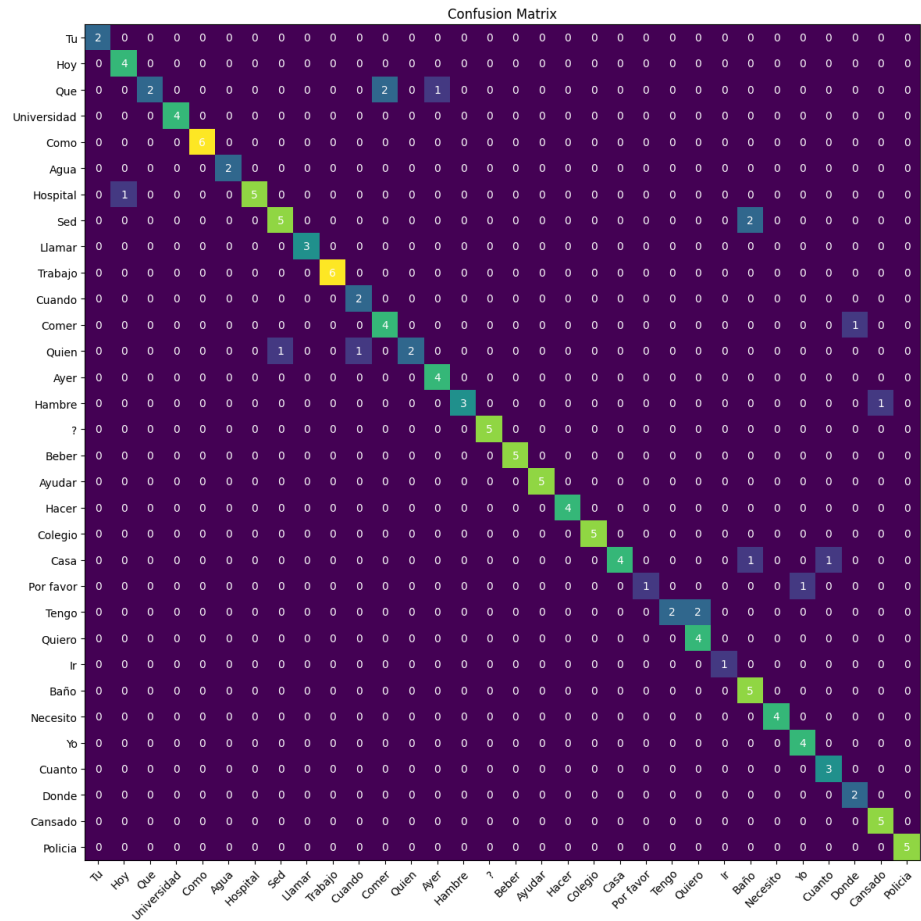


Figura 147: Matriz de confusión del modelo con aumento de complejidad

Adición de dropout al modelo base

La composición del modelo base con dropout se puede observar en la Figura 148. Este modelo tiene una capa de entrada, cuatro capas densas con activación tangente hiperbólica y una capa de salida con activación softmax. Adicional a esto, se ha añadido una capa de dropout con una tasa de 0.1 a cada capa densa.

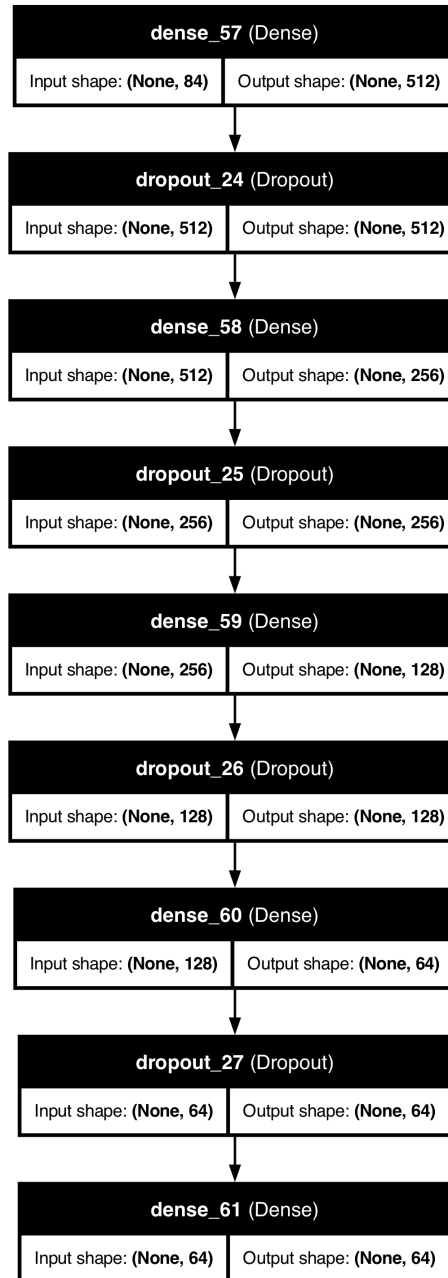


Figura 148: Modelo base con dropout

El desempeño de este modelo se puede observar en la Tabla 17. Este modelo tiene una precisión de 0.8072, una sensibilidad de 0.9247 y un F1 de 0.9274. Adicional a esto, se puede observar la matriz de confusión en la Figura 150 y el historial de entrenamiento en la Figura 149.

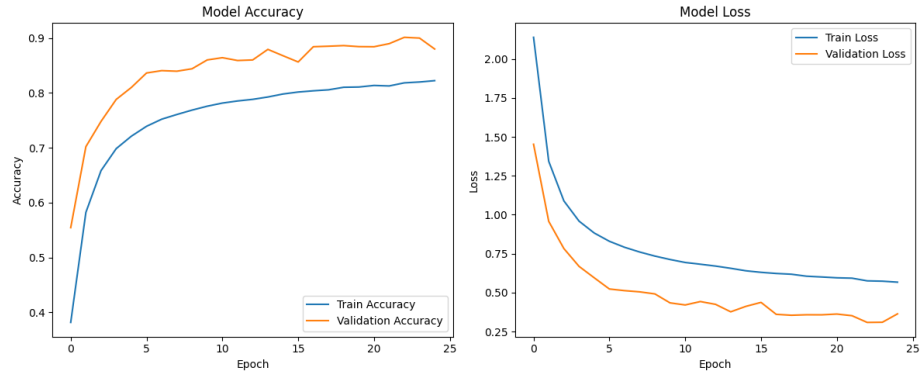


Figura 149: Historial de entrenamiento del modelo base con dropout

Métrica	Valor
Precisión	0.8072
Sensibilidad	0.9247
F1	0.9274

Tabla 17: Desempeño del modelo base con dropout

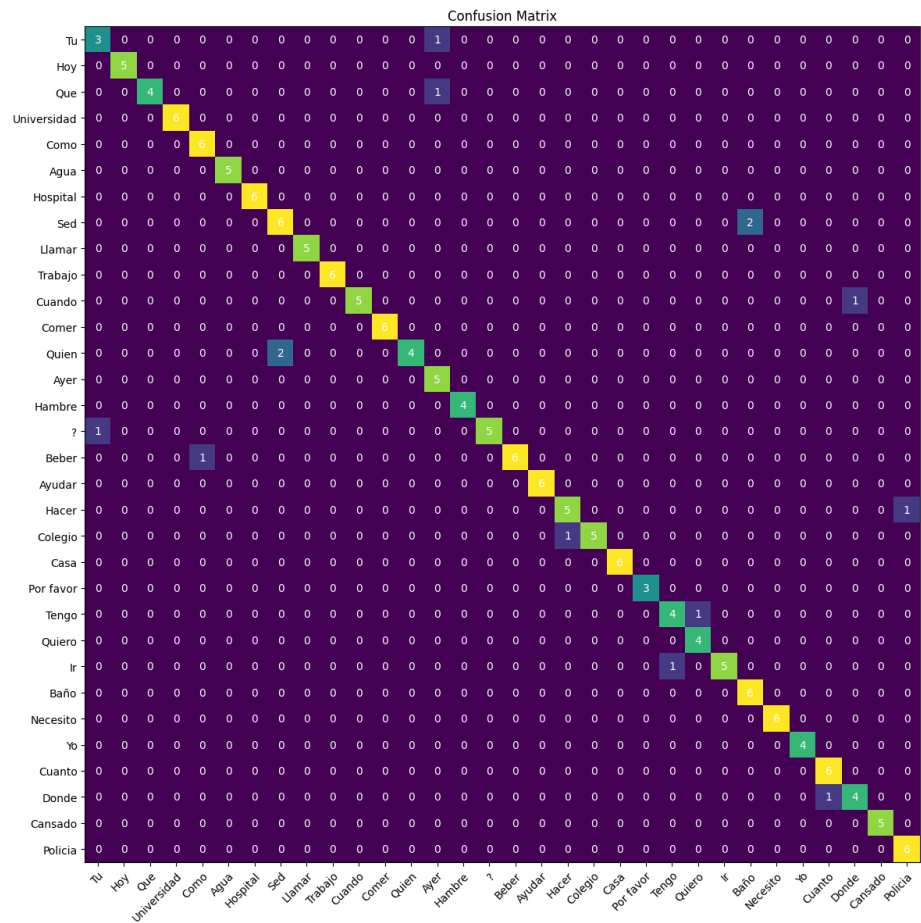


Figura 150: Matriz de confusión del modelo base con dropout

Fine-tuning de las capas dropout

En la iteración previa, se observó que la adición de dropout al modelo base mejoró el desempeño del modelo. Esto se debe a que el dropout ayuda a prevenir el sobreajuste del modelo, al desactivar aleatoriamente un porcentaje de las neuronas en cada época, lo cual permite que el modelo generalice mejor. En esta iteración, se realizó un fine-tuning de la tasa de dropout, con el objetivo de encontrar la tasa de dropout que maximizara el desempeño del modelo. Se realizan pruebas con dos modelos distintos, con el objetivo de encontrar la tasa de dropout que maximice el desempeño del modelo.

La composición del primero de los modelos se puede observar en la Figura 151. Este modelo tiene una capa de entrada, cuatro capas densas con activación tangente hiperbólica y una capa de salida con activación softmax. A diferencia del modelo anterior, este modelo tiene una capa de dropout con una tasa de 0.2 a cada capa densa.

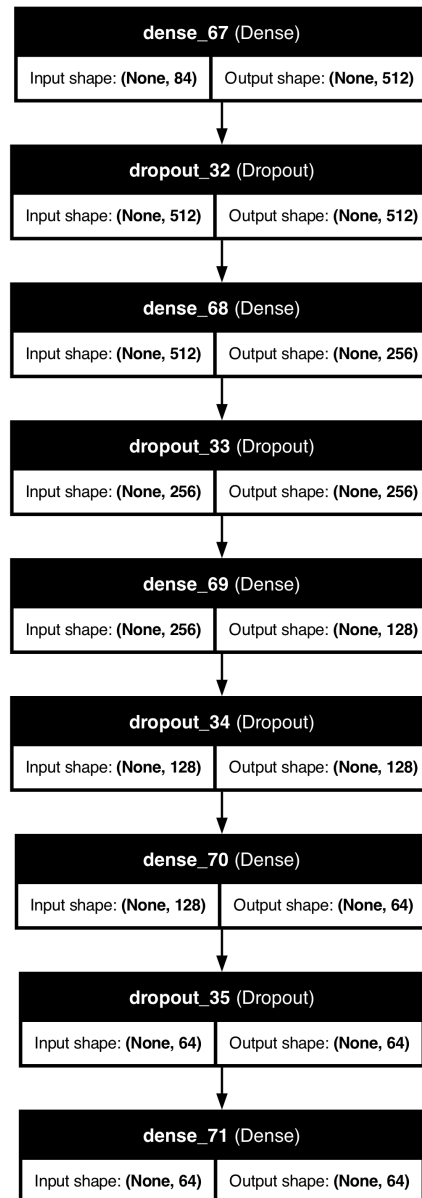


Figura 151: Primera iteración de fine tuning de la tasa de dropout

El desempeño de este modelo se puede observar en la Tabla 18. Este modelo tiene una precisión de 0.7656, una sensibilidad de 0.9681 y un F1 de 0.9617. Adicional a esto, se puede observar la matriz de confusión en la Figura 153 y el historial de entrenamiento en la Figura 152.

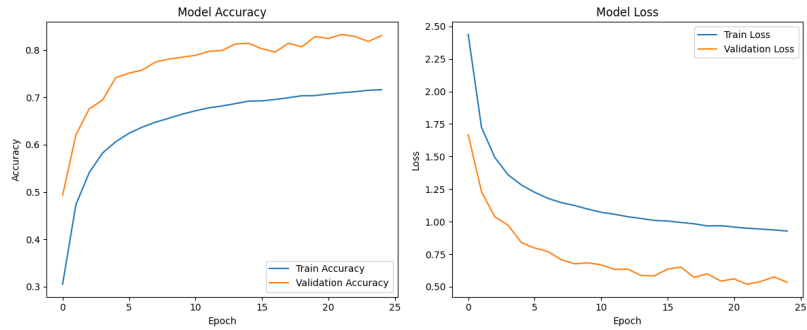


Figura 152: Historial de entrenamiento del primer modelo de fine tuning de la tasa de dropout

Métrica	Valor
Precisión	0.7656
Sensibilidad	0.9681
F1	0.9617

Tabla 18: Desempeño del primer modelo de fine tuning de la tasa de dropout

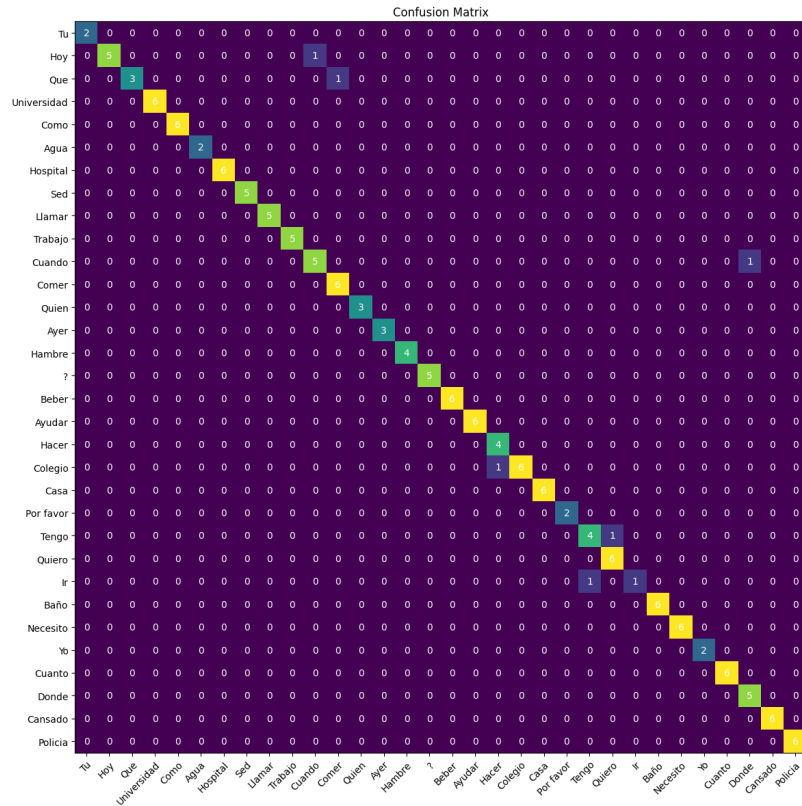


Figura 153: Matriz de confusión del primer modelo de fine tuning de la tasa de dropout

La composición del segundo de los modelos se puede observar en la Figura 154. Este modelo tiene una capa de entrada, cuatro capas densas con activación tangente hiperbólica y una capa de salida con activación softmax. Debido a que el modelo anterior tuvo un desempeño inferior al modelo base con dropout, se decidió probar con dos capas de dropout con una tasa de 0.2 y dos capas de dropout con una tasa de 0.1.

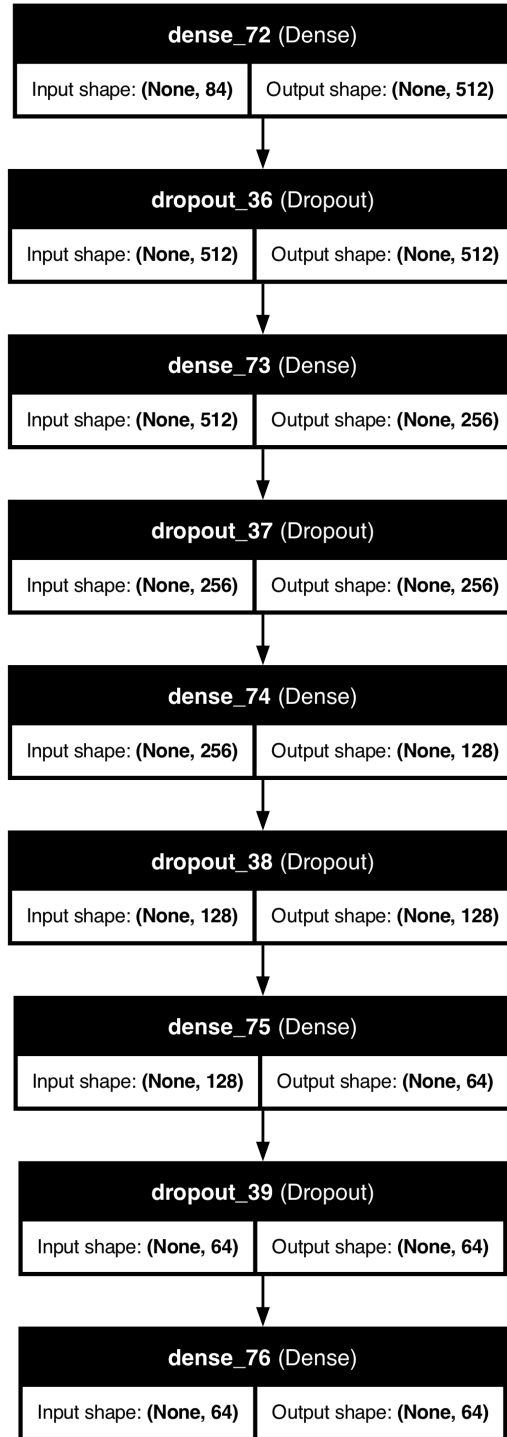


Figura 154: Segunda iteración de fine tuning de la tasa de dropout

El desempeño de este modelo se puede observar en la Tabla 19. Este modelo tiene una precisión de 0.7708, una sensibilidad de 0.9541 y un F1 de 0.9543. Adicional a esto, se puede observar la matriz de confusión en la Figura 156 y el historial de entrenamiento en la Figura 155.

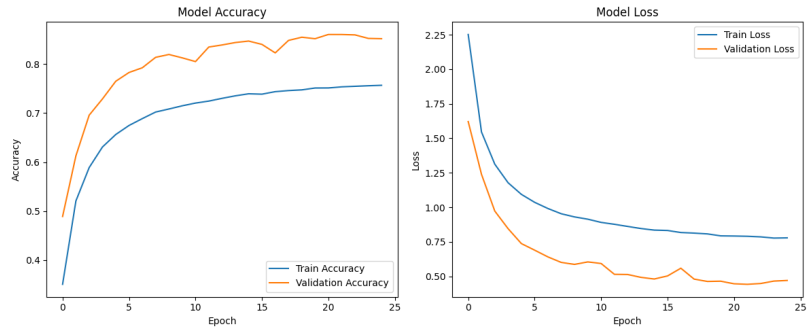


Figura 155: Historial de entrenamiento del segundo modelo de fine tuning de la tasa de dropout

Métrica	Valor
Precisión	0.7708
Sensibilidad	0.9541
F1	0.9543

Tabla 19: Desempeño del segundo modelo de fine tuning de la tasa de dropout

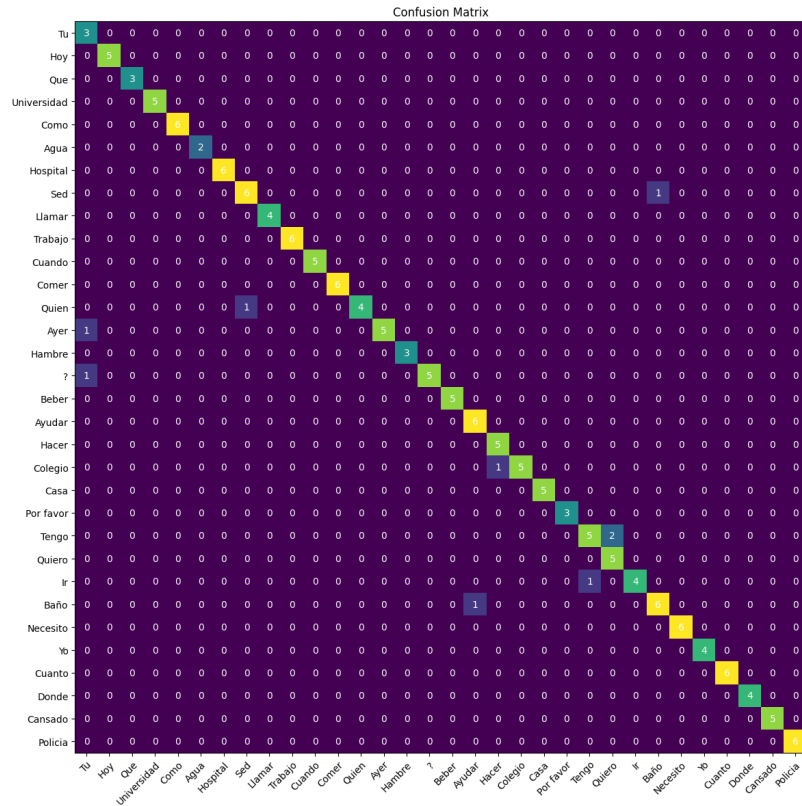


Figura 156: Matriz de confusión del segundo modelo de fine tuning de la tasa de dropout

Adición de normalización por lotes al modelo base

La normalización por lotes es una técnica que se utiliza para normalizar las entradas de una red neuronal, con el objetivo de acelerar el proceso de entrenamiento y mejorar la precisión del modelo. En esta iteración, se añadió una capa de normalización por lotes por cada capa densa del modelo base, con el objetivo de evaluar si la normalización por lotes mejoraba el desempeño del modelo. La composición del modelo con normalización por lotes se puede observar en la Figura [157](#).

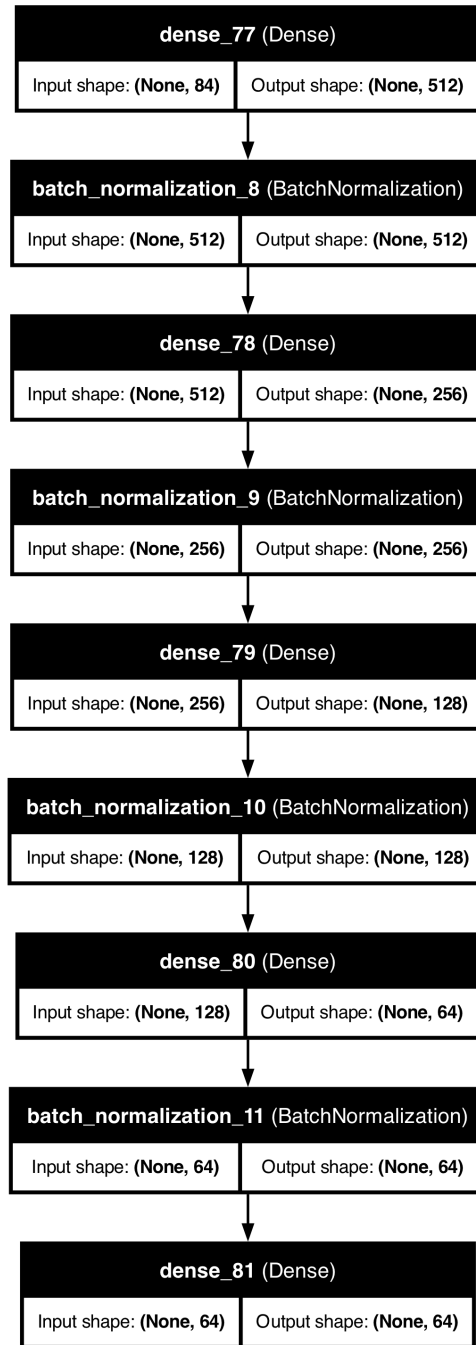


Figura 157: Modelo con normalización por lotes

El desempeño de este modelo se puede observar en la Tabla 20. Este modelo tiene una precisión de 0.8593, una sensibilidad de 0.9136 y un F1 de 0.9155. Adicional a esto, se puede observar la matriz de confusión en la Figura 159 y el historial de entrenamiento en la Figura 158.

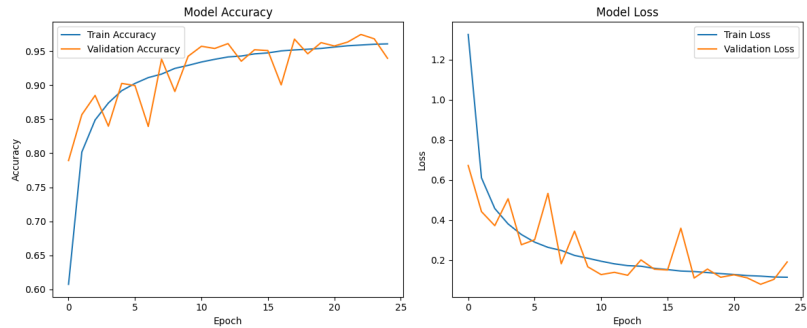


Figura 158: Historial de entrenamiento del modelo con normalización por lotes

Métrica	Valor
Precisión	0.8593
Sensibilidad	0.9136
F1	0.9155

Tabla 20: Desempeño del modelo con normalización por lotes

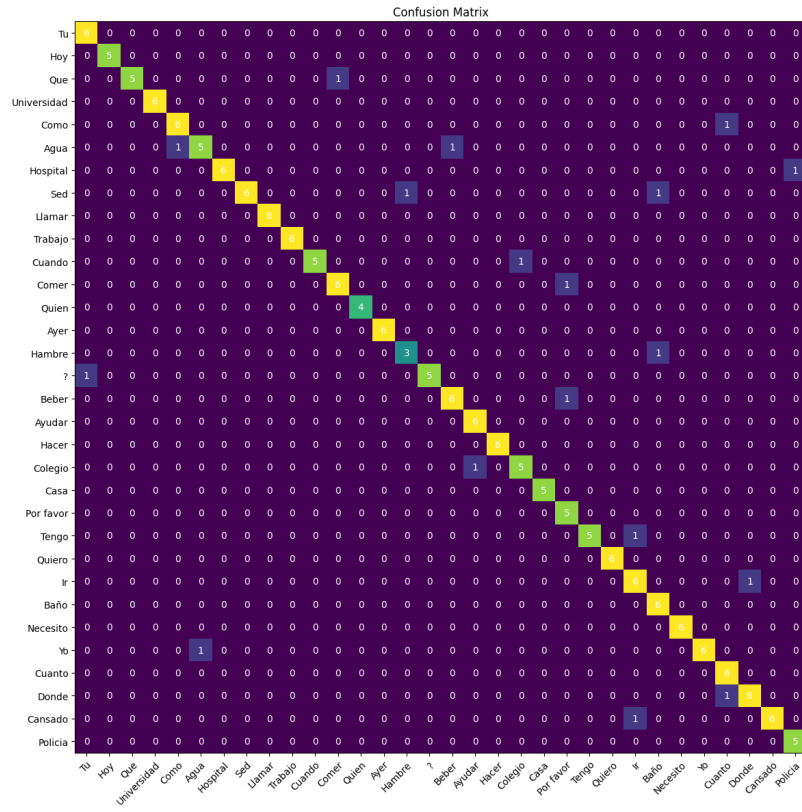


Figura 159: Matriz de confusión del modelo con normalización por lotes

Combinación de dropout y normalización por lotes

En esta iteración, se combinaron las técnicas de dropout y normalización por lotes, con el objetivo de evaluar si la combinación de ambas técnicas mejoraba el desempeño del modelo. Se realizan pruebas con dos modelos distintos, con el objetivo de encontrar la combinación de dropout y normalización por lotes que maximizara el desempeño del modelo. La composición del primero de los modelos se puede observar en la Figura [160](#).

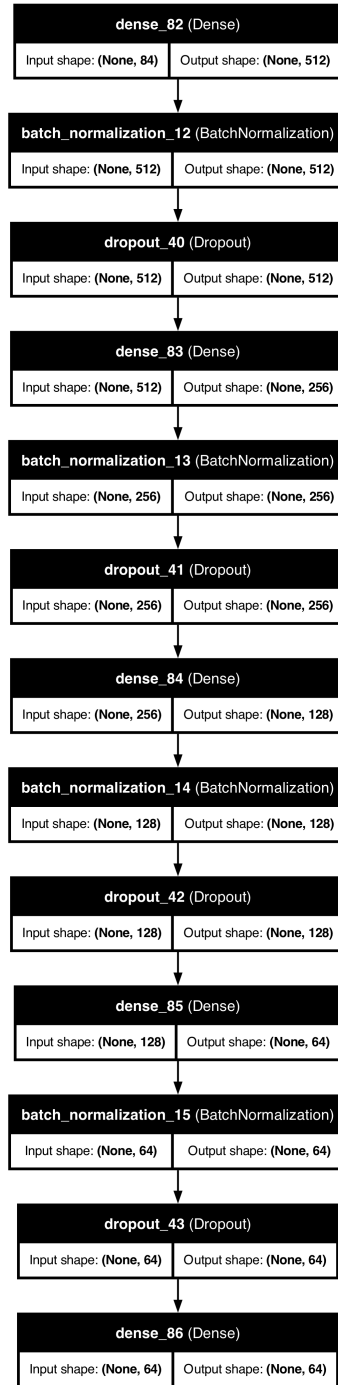


Figura 160: Primer modelo con dropout y normalización por lotes

El desempeño de este modelo se puede observar en la Tabla 21. Este modelo tiene una precisión de 0.8593, una sensibilidad de 0.9461 y un F1 de 0.9441. Adicional a esto, se puede observar la matriz de confusión en la Figura 162 y el historial de entrenamiento en la Figura 161.

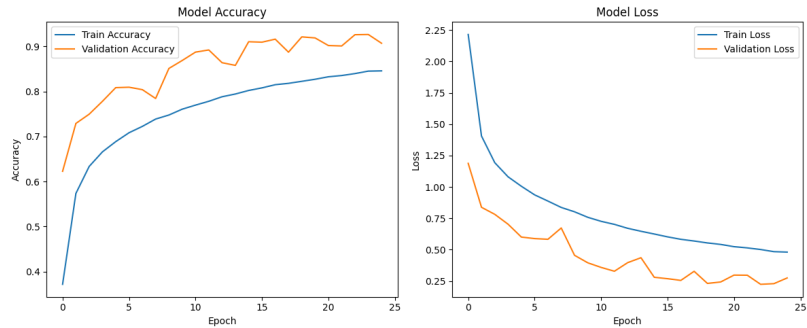


Figura 161: Historial de entrenamiento del primer modelo con de dropout y normalización por lotes

Métrica	Valor
Precisión	0.8593
Sensibilidad	0.9461
F1	0.9441

Tabla 21: Desempeño del primer modelo con dropout y normalización por lotes

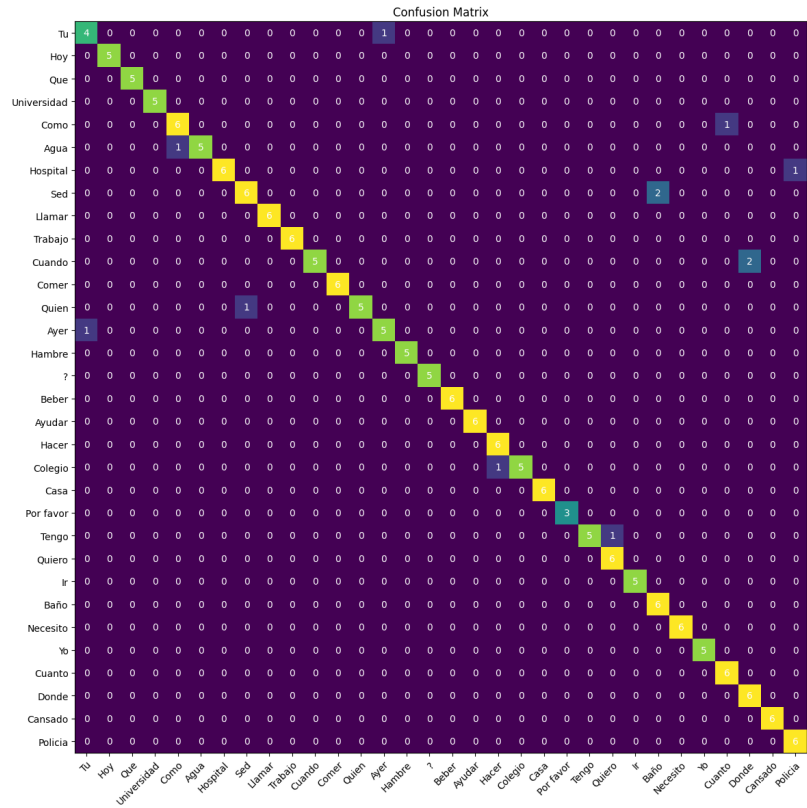


Figura 162: Matriz de confusión del primer model con dropout y normalización por lotes

La composición del segundo de los modelos se puede observar en la Figura [163](#).

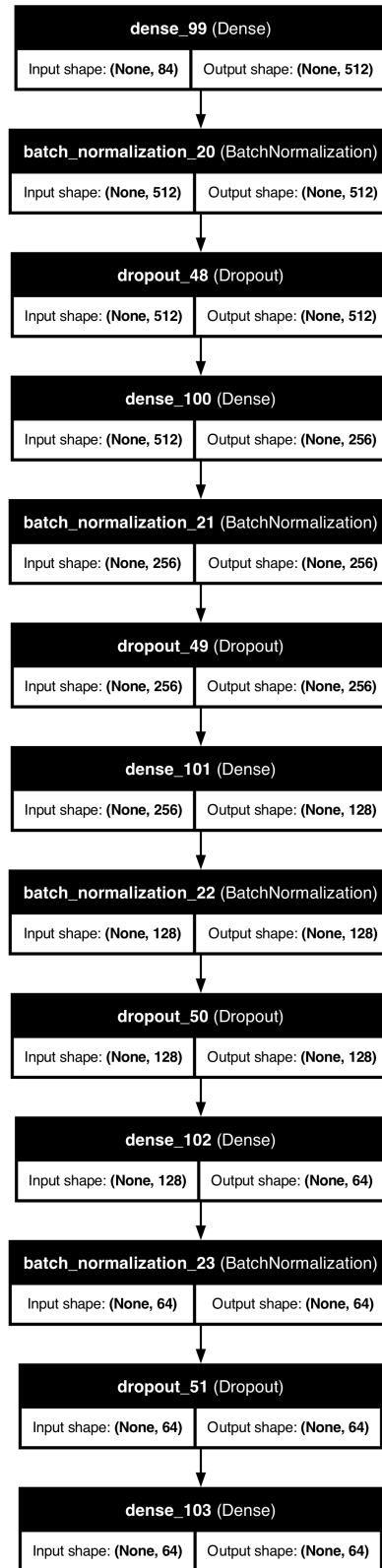


Figura 163: Segundo modelo con dropout y normalización por lotes

El desempeño de este modelo se puede observar en la Tabla 22. Este modelo tiene una precisión de 0.8593, una sensibilidad de 0.9461 y un F1 de 0.9441. Adicional a esto, se puede observar la matriz de confusión en la Figura 165 y el historial de entrenamiento en la Figura 164.

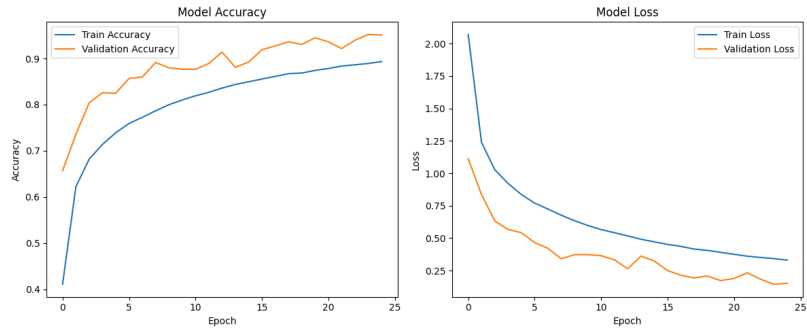


Figura 164: Historial de entrenamiento del segundo modelo con dropout y normalización por lotes

Métrica	Valor
Precisión	0.8802
Sensibilidad	0.9685
F1	0.9706

Tabla 22: Desempeño del segundo modelo con dropout y normalización por lotes

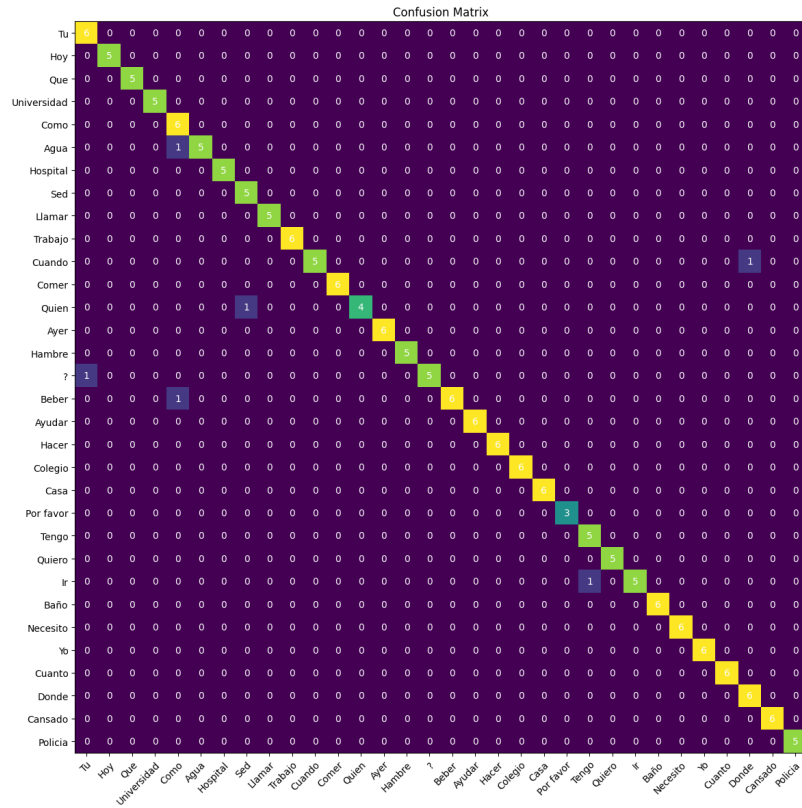


Figura 165: Matriz de confusión del segundo modelo con dropout y normalización por lotes

Elección del modelo final

El modelo final seleccionado es el modelo con dropout y normalización por lotes, con una precisión de 0.8802, una sensibilidad de 0.9685 y un F1 de 0.9706. Este modelo tiene un desempeño superior a los modelos anteriores, lo cual indica que la combinación de dropout y normalización por lotes es efectiva para mejorar el desempeño del modelo. Al observar la matriz de confusión en la Figura 165, se puede observar que el modelo tiene un buen desempeño en la mayoría de las clases.

6.1.3. Evaluación del modelo

Reconocimiento de una palabra

El modelo final fue implementado en una aplicación de reconocimiento de lengua de señas de Guatemala en tiempo real, la cual se probó con un conjunto de datos de prueba. Este conjunto de datos de prueba está compuesto de 192 videos, los cuales contienen gestos de 32 palabras de la lengua de señas de Guatemala. Al utilizar el modelo final, se obtuvieron resultados satisfactorios, ya que el modelo fue capaz de reconocer correctamente los gestos de lengua de señas en tiempo real. En la Figura 166 se puede observar el desempeño del modelo en tiempo real. Adicionalmente, se puede observar el desempeño del modelo en la Tabla 23.

Métrica	Valor
Precisión	0.8802
Sensibilidad	0.9685
F1	0.9706
FPS promedio	19.38

Tabla 23: Desempeño del segundo modelo de la combinación de dropout y normalización por lotes

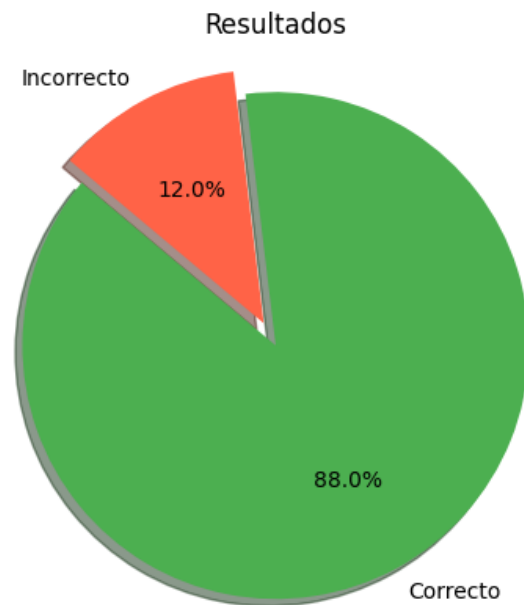


Figura 166: Reconocimiento de lengua de señas de Guatemala en tiempo real (una palabra)

Reconocimiento de múltiples palabras

Para evaluar el desempeño del modelo en la detección de múltiples palabras, se utilizó un conjunto de datos de prueba con videos de varias señas. Este conjunto de datos de prueba está compuesto de 27 videos, los cuales contienen gestos de 2 a 4 palabras de la lengua de señas de Guatemala. Al utilizar el modelo final, no se obtuvieron resultados satisfactorios, ya que el modelo no fue capaz de reconocer correctamente los gestos de lengua de señas en tiempo real. En la Figura 167 se puede observar el desempeño del modelo en tiempo real.



Figura 167: Reconocimiento de lengua de señas de Guatemala en tiempo real (múltiples palabras)

6.1.4. Aplicaciones del modelo

Pruebas en tiempo real

En las pruebas realizadas en tiempo real, el modelo final fue capaz de reconocer correctamente los gestos de lengua de señas de Guatemala en la mayoría de los casos. Estas pruebas permitieron observar como el modelo se comporta en un entorno real, y si es capaz de reconocer todos los puntos clave de las señas. A continuación se muestran algunas capturas de pantalla de la aplicación de reconocimiento de lengua de señas de Guatemala en tiempo real. Cabe mencionar que cada uno de estos dos videos fueron grabados en diferentes angulos, con diferentes dispositivos, y con diferentes personas realizando los gestos. Las figuras 168, y 169 muestran el funcionamiento de la aplicación en tiempo real, con las palabras *ayer* y *comer*, respectivamente.



Figura 168: Reconocimiento de la palabra *ayer* en tiempo real

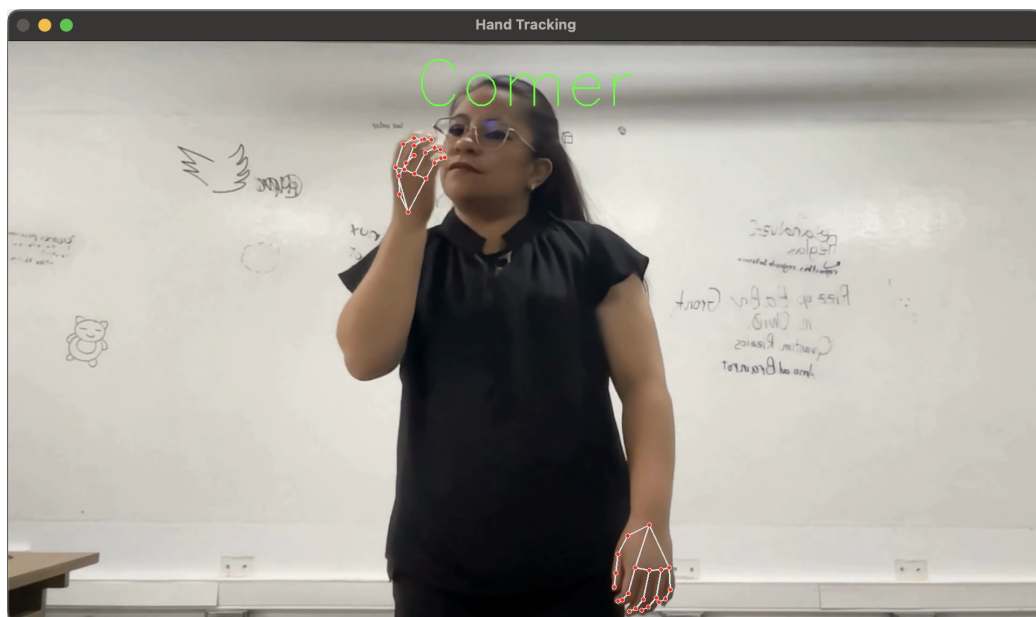


Figura 169: Reconocimiento de la palabra *comer* en tiempo real

Pruebas del API

El modelo final fue implementado en una Interfaz de Programación de Aplicaciones (API), la cual fue probada utilizando Postman y permitió acceder al modelo de reconocimiento de lengua de señas de forma remota. A continuación se muestran algunas capturas de pantalla de las pruebas realizadas con el API. Las figuras [170](#), y [171](#) muestran el funcionamiento del API, con las palabras *universidad* y *tu*, respectivamente.

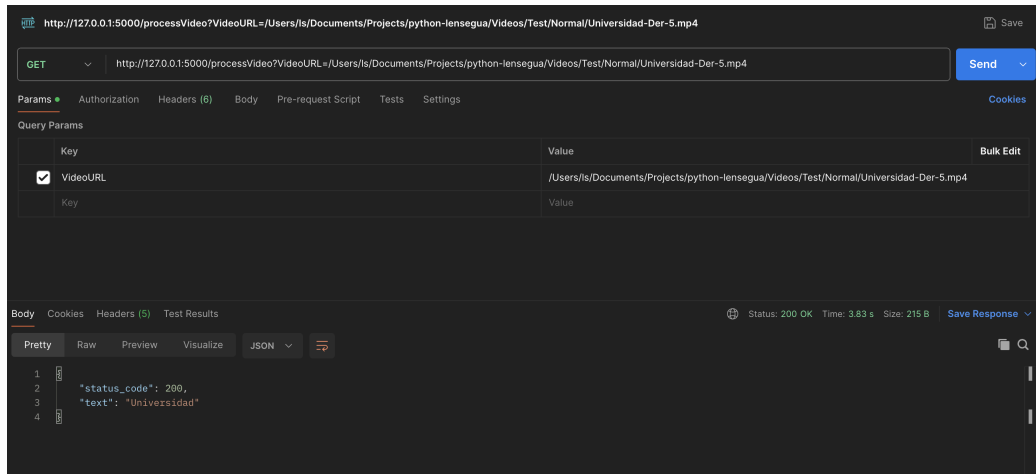


Figura 170: Reconocimiento de la palabra *universidad* utilizando el API

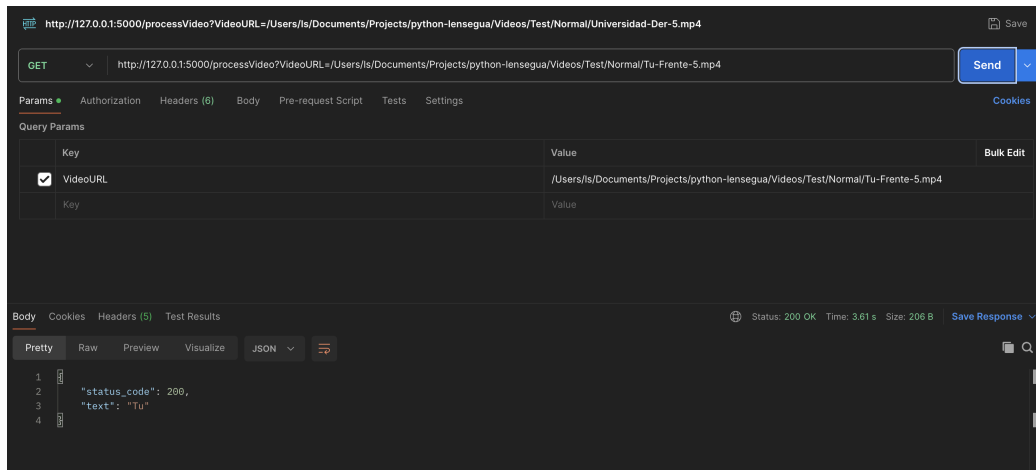


Figura 171: Reconocimiento de la palabra *tu* utilizando el API

6.2. Procesamiento de lenguaje natural (GPT-3.5-Turbo)

6.2.1. Fine-tuning

Antes de proceder con el *fine-tuning* del modelo GPT-3.5-Turbo, se llevaron a cabo experimentos preliminares para identificar los parámetros que generaban la menor pérdida de validación. Esta métrica indica la eficacia del modelo para predecir correctamente las salidas a partir de las entradas proporcionadas. Una menor pérdida de validación sugiere que el modelo está aprendiendo de manera efectiva y puede generalizar mejor a datos no vistos. En la Tabla 24, se presentan los valores de los parámetros ajustados durante estas pruebas, junto con la pérdida de validación correspondiente a cada configuración.

Epochs	Batch Size	Learning Rate Multiplier	Pérdida de validación
2	16	0.2	0.1267
1	16	0.3	0.1334
1	8	0.5	0.2383
2	1	2	0.3163
3	1	2	0.4418
6	1	2	0.5966

Tabla 24: Valores de parámetros y pérdidas de validación en experimentos preliminares

Tras analizar los resultados de estas pruebas, se decidió llevar a cabo el *fine-tuning* final utilizando el conjunto de datos de entrenamiento completo y los parámetros que demostraron generar la menor pérdida de validación. Los parámetros seleccionados fueron: 2 *epochs*, un *batch size* de 16 y un *learning rate multiplier* de 0.2.

Parámetro	Valor
Epochs	2
Batch size	16
Learning Rate Multiplier	0.2

Tabla 25: Parámetros utilizados para el *fine-tuning* del modelo GPT-3.5-Turbo.

Durante el proceso de *fine-tuning*, la plataforma de OpenAI generó una gráfica que se actualizaba con cada paso del entrenamiento, lo que facilitó el monitoreo de la evolución de la pérdida en tiempo real. La gráfica final, presentada en la Figura 172, confirmó que el modelo no presentaba signos de *overfitting*, es decir que el modelo resultante tenía la capacidad para adaptarse a nuevos datos. Esta observación fue fundamental para garantizar que el modelo pudiera aplicarse efectivamente en situaciones prácticas.



Figura 172: Evolución de la pérdida durante el *fine-tuning* final del modelo GPT-3.5-Turbo.

Una vez completado el *fine-tuning* del modelo GPT-3.5-Turbo, se utilizó la técnica de *Local Interpretable Model-agnostic Explanations* (LIME) para evaluar el impacto de este proceso en las interpretaciones generadas por el modelo. LIME permitió identificar, para frases específicas, qué palabras influían más en las respuestas de ambos modelos, facilitando el análisis de cómo cada uno priorizaba ciertos elementos del texto al generar sus respuestas. Para ello, se seleccionaron tres frases

representativas de la gramática de LENSEGUA, diseñadas para evaluar la capacidad de los modelos en interpretar estructuras lingüísticas clave, como marcadores temporales, términos interrogativos y otras palabras funcionales.

Las figuras 173 y 174 presentan los resultados de los análisis LIME realizados para la frase “antes tu policía llamar pregunta”. Cada visualización proporciona información detallada sobre el impacto de cada palabra en las respuestas generadas por ambos modelos, codificándolas por colores y asociándolas a un valor numérico específico. En ambas figuras, las palabras resaltadas en azul (con valores cercanos a -1) fueron aquellas que generaron confusión, resultando en respuestas menos alineadas con la interpretación teórica establecida. En contraste, las palabras resaltadas en naranja (con valores cercanos a 1) contribuyeron de manera positiva, ayudando a formular interpretaciones más precisas.

Estos resultados se resumen en la Tabla 26, donde destaca un cambio significativo en la palabra “pregunta”, que pasó de tener una influencia negativa (-0.09) en el modelo estándar a una positiva (0.09) en el modelo adaptado. Esta transformación sugiere que el *fine-tuning* mejoró la capacidad del modelo para reconocer y procesar marcadores interrogativos. Adicionalmente, las palabras “policía” y “llamar” también mostraron un aumento en su relevancia en el modelo adaptado (0.43 y 0.20, respectivamente), en comparación con el modelo estándar (0.13 y 0.07).

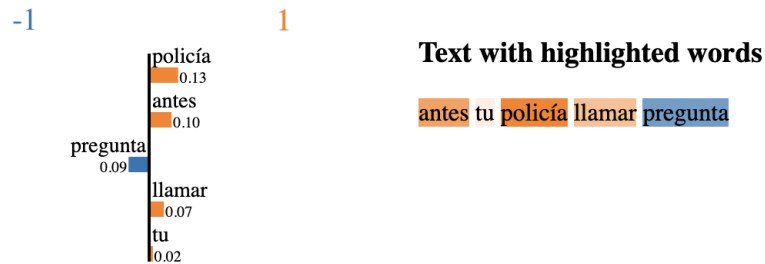


Figura 173: Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo estándar para la frase “antes tu policía llamar pregunta”.

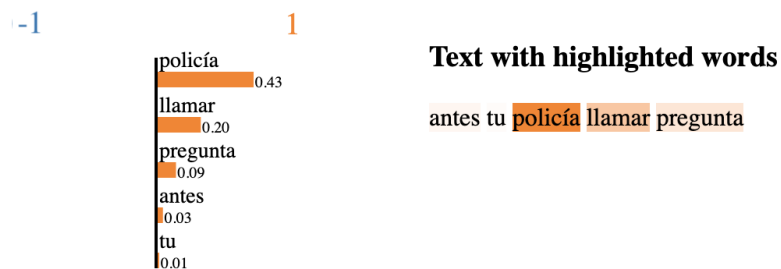


Figura 174: Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo *fine-tuneado* para la frase “antes tu policía llamar pregunta”.

Frase en LENSEGUA: antes tu policía llamar pregunta		
Interpretación teórica: ¿Llamaste a la policía?		
Palabra	Valores de influencia (LIME)	
	Modelo GPT-3.5-Turbo	Modelo <i>fine-tuneado</i>
antes	0.10	0.03
tu	0.02	0.01
policía	0.13	0.43
llamar	0.07	0.20
pregunta	-0.09	0.09

Tabla 26: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “antes tu policía llamar pregunta”.

En la Tabla 27 se presentan los resultados para la frase “futuro hospital él ir”. El marcador temporal “futuro” mostró el cambio más significativo, pasando de una influencia negativa de -0.04 en el modelo estándar (ver Figura 175) a una positiva de 0.09 en el modelo adaptado (ver Figura 176). Esto indica que el modelo ajustado reconoció mejor la importancia de “futuro” como un elemento clave para la interpretación de la frase. Por otro lado, la palabra “hospital” aumentó su relevancia (de 0.34 a 0.51), mientras que las influencias de “él” e “ir” experimentaron una ligera reducción.

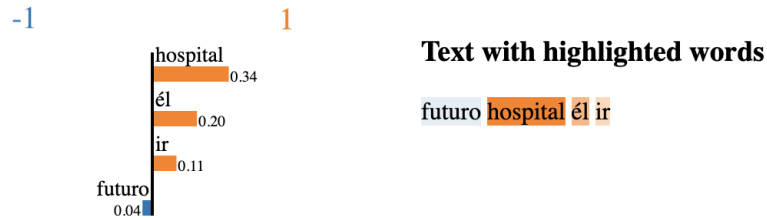


Figura 175: Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo estándar para la frase “futuro hospital él ir”.

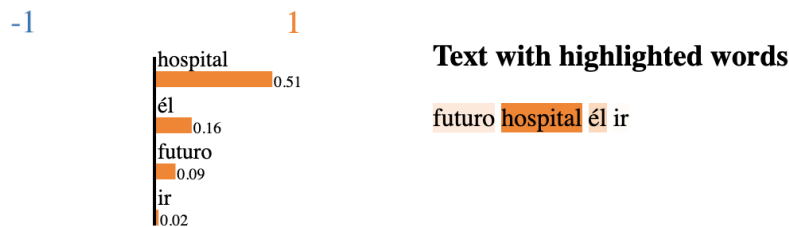


Figura 176: Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo *fine-tuneado* para la frase “futuro hospital él ir”.

Frase en LENSEGUA: futuro hospital él ir		
Interpretación teórica: Él irá al hospital.		
Palabra	Valores de influencia (LIME)	
	Modelo GPT-3.5-Turbo	Modelo <i>fine-tuneado</i>
futuro	-0.04	0.09
hospital	0.34	0.51
él	0.20	0.16
ir	0.11	0.02

Tabla 27: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “futuro hospital él ir”.

La Tabla 28 muestra los resultados de los análisis LIME para la frase “pasado yo ir no”, combinando la información de las Figuras 177 y 178. En este caso, se destaca un aumento significativo en el impacto del término “pasado”, que subió de 0.08 en el modelo estándar a 0.24 en el modelo *fine-tuneado*. Además, la palabra “no” mostró un notable incremento en su relevancia, pasando de 0.12 a 0.44. Finalmente, la influencia de “ir” se redujo de 0.37 a 0.17, mientras que “yo” mostró una leve mejora, pasando de -0.03 a 0.06.

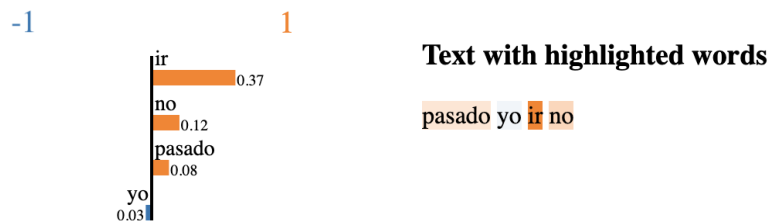


Figura 177: Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo estándar para la frase “pasado yo ir no”.

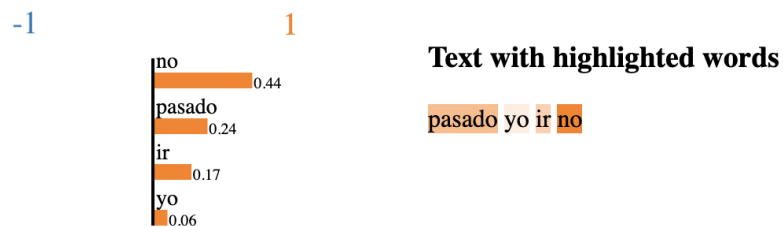


Figura 178: Resultados de LIME para la interpretación generada por el modelo GPT-3.5-Turbo *fine-tuneado* para la frase “pasado yo ir no”.

Frase en LENSEGUA: pasado yo ir no		
Interpretación teórica: No fui.		
Palabra	Valores de influencia (LIME)	
	Modelo GPT-3.5-Turbo	Modelo <i>fine-tuneado</i>
pasado	0.08	0.24
yo	-0.03	0.06
ir	0.37	0.17
no	0.12	0.44

Tabla 28: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “pasado yo ir no”.

6.2.2. Prompt engineering

Durante las fases iniciales, se utilizó el siguiente *prompt* para guiar al modelo *fine-tuneado* en la generación de interpretaciones. Este *prompt* establecía las instrucciones necesarias para que el modelo actuara como un intérprete en LENSEGUA. Sin embargo, durante las pruebas realizadas, se identificaron varios aspectos que limitaron la calidad de las respuestas. En particular, se presentaron dificultades al procesar expresiones temporales, lo que resultó en errores y redundancias.

Prompt (versión 1)

Eres un intérprete experto en Lengua de Señas de Guatemala (LENSEGUA). El usuario te proporcionará frases en español escritas utilizando la gramática de LENSEGUA. Tu tarea es interpretarlas y convertirlas en español gramaticalmente correcto.

Con el objetivo de mejorar la calidad de las interpretaciones, se llevó a cabo una fase de *prompt engineering*. En esta etapa, se realizaron ajustes iterativos sobre el *prompt* original, incorporando instrucciones más precisas para corregir errores recurrentes. Además, para evaluar la efectividad de cada variante, como se mencionó en la sección de metodología, se calculó la distancia de Levenshtein promedio utilizando el conjunto de validación, el modelo *fine-tuneado* y las diferentes versiones de *prompts*. En este contexto, se propuso un segundo *prompt*.

Prompt (versión 2)

Soy un intérprete experto en Lengua de Señas de Guatemala (LENSEGUA). El usuario me proporcionará frases en español escritas utilizando la gramática de LENSEGUA. Mi tarea es interpretarlas y convertirlas en español gramaticalmente correcto.

Para esta segunda versión, se decidió formular las instrucciones en primera persona en lugar de tercera. Esto se hizo para que el modelo asumiera un rol más activo en el proceso de interpretación. Sin embargo, a pesar de este cambio, la distancia de Levenshtein promedio obtenida con las interpretaciones generadas por el modelo *fine-tuneado* y este nuevo *prompt* fue de 5.84, ligeramente superior al valor de 5.815 obtenido con la primera versión. Además, tras una revisión de las interpretaciones, se identificó que los errores anteriormente mencionados aún persistían. Ante esto, se retomó el *prompt* original y se elaboró una tercera versión del *prompt* que especificaba claramente la exclusión de ciertos términos.

Prompt (versión 3)

Eres un intérprete experto en Lengua de Señas de Guatemala (LENSEGUA). El usuario te proporcionará frases en español escritas utilizando la gramática de LENSEGUA. Tu tarea es interpretarlas y convertirlas en español gramaticalmente correcto, teniendo en cuenta la siguiente regla específica:

- *Las palabras “pasado”, “futuro” y “antes” indican la conjugación verbal. Debes conjugar correctamente los verbos en tu respuesta sin incluir las palabras “pasado”, “futuro” o “antes”.*

Esta tercera versión mostró una mejora, con una distancia de Levenshtein promedio de 5.37. No obstante, se identificaron problemas en la interpretación de expresiones comunes en LENSEGUA que no tienen un equivalente directo en español. Aunque el modelo *fine-tuneado* había sido expuesto a expresiones como ‘carro mucho’ (interpretado correctamente como ‘tráfico’) y ‘bien mucho’ (como ‘muy bien’) durante su entrenamiento, las respuestas que generaba eran inconsistentes; como si en ocasiones el modelo “olvidara” estas interpretaciones o no las aplicara de manera uniforme. Para asegurar respuestas más coherentes, se realizaron ajustes adicionales en el *prompt*, añadiendo recordatorios específicos sobre cómo manejar dichas expresiones. Además, se incluyó una instrucción que prevenía al modelo de interpretar frases como preguntas, a menos que incluyeran términos como ‘pregunta’ o ‘sí o no’. Estos cambios contribuyeron significativamente a mejorar la estabilidad y la precisión en las respuestas del modelo.

Prompt (versión 4)

Eres un intérprete experto en Lengua de Señas de Guatemala (LENSEGUA). El usuario te proporcionará frases en español escritas utilizando la gramática de LENSEGUA. Tu tarea es interpretarlas y convertirlas en español gramaticalmente correcto, teniendo en cuenta las siguientes reglas específicas:

- *No debes interpretar las frases como preguntas, ni agregar una entonación interrogativa, a menos que incluyan las palabras “pregunta” o “sí o no”.*
- *Las palabras “pasado”, “futuro” y “antes” indican la conjugación verbal. Debes conjugar correctamente los verbos en tu respuesta sin incluir las palabras “pasado”, “futuro” o “antes”.*
- *La frase “carro mucho” debe interpretarse como “tráfico”.*
- *La frase “bien mucho” debe interpretarse como “muy bien”.*

La cuarta versión del *prompt* alcanzó una distancia de Levenshtein promedio de 4.30, calculada a partir de las interpretaciones generadas por el modelo *fine-tuneado*. Al analizar detenidamente las interpretaciones individuales, se observó que la mayoría eran correctas. Sin embargo, en algunos casos, el modelo alteraba el orden de los elementos dentro de las oraciones, lo que, en comparación con lo esperado, afectaba negativamente la métrica de desempeño. Para intentar solventar esta situación, se incorporó una instrucción adicional que enfatizaba la necesidad de mantener una estructura y secuencia específica en las interpretaciones finales.

Prompt (versión 5)

Eres un intérprete experto en Lengua de Señas de Guatemala (LENSEGUA). El usuario te proporcionará frases en español escritas utilizando la gramática de LENSEGUA. Tu tarea es interpretarlas y convertirlas en español gramaticalmente correcto, teniendo en cuenta las siguientes reglas específicas:

- No debes interpretar las frases como preguntas, ni agregar una entonación interrogativa, a menos que incluyan las palabras “pregunta” o “sí o no”.
- Las palabras “pasado”, “futuro” y “antes” indican la conjugación verbal. Debes conjugar correctamente los verbos en tu respuesta sin incluir las palabras “pasado”, “futuro” o “antes”.
- La frase “carro mucho” debe interpretarse como “tráfico”.
- La frase “bien mucho” debe interpretarse como “muy bien”.
- Mantén el orden de los elementos de la interpretación final lo más similar posible al texto original cuando sea gramaticalmente válido en español. Por ejemplo, si la frase en LENSEGUA es “Por favor cuando tu salir luz apagar”, la interpretación debe ser “Por favor cuando salgas apaga la luz” en lugar de “Por favor apaga la luz cuando salgas”.

Esta última versión del *prompt* obtuvo una distancia de Levenshtein promedio de 3.38. Este resultado indica una notable mejora en la coherencia y precisión de las interpretaciones, demostrando la efectividad del proceso iterativo de *prompt engineering*.

Es importante señalar que también se calculó la distancia de Levenshtein promedio para cada versión del *prompt* utilizando el modelo GPT-3.5-Turbo estándar y el *dataset* de validación. Esto permitió comparar el impacto de las modificaciones en cada versión del *prompt* sobre ambos modelos. Los valores obtenidos se detallan en la Tabla 29.

Prompt	Distancia de Levenshtein promedio	
	Modelo GPT-3.5-Turbo	Modelo <i>fine-tuneado</i>
Versión 1	10.065	5.815
Versión 2	10.755	5.840
Versión 3	8.660	5.370
Versión 4	8.635	4.300
Versión 5	8.680	3.375

Tabla 29: Distancias de Levenshtein promedio calculadas a partir de las interpretaciones generadas por el modelo GPT-3.5-Turbo (estándar) y el modelo *fine-tuneado*, utilizando diversos *prompts*.

Los resultados indican que el modelo *fine-tuneado* superó consistentemente al modelo estándar, logrando menores distancias promedio para todos los *prompts* evaluados. Por ejemplo, con la versión 1 del *prompt*, el modelo estándar alcanzó una distancia de 10.065, mientras que el modelo *fine-tuneado* logró una distancia significativamente menor de 5.815. Asimismo, la menor distancia del modelo *fine-tuneado* se registró con la versión 5 del *prompt* (3.375), mientras que el mejor desempeño del modelo original fue con la versión 3, con una distancia de 8.660.

Aunque el *prompt* se fue mejorando de forma iterativa, el impacto de estos cambios se pudo observar únicamente en las interpretaciones generadas por el modelo *fine-tuneado*. Esto debido a que con cada versión nueva del *prompt*, se logró obtener una distancia de Levenshtein promedio menor.

En cambio, las interpretaciones del modelo estándar no mostraron mejoras significativas, manteniendo distancias relativamente altas a pesar de los cambios. Esto sugiere que, si bien los *prompts* se optimizaron progresivamente, el modelo estándar no logró beneficiarse de estas actualizaciones. Esto posiblemente debido a que la tarea era demasiado compleja para resolverse únicamente con *prompt engineering*.

6.2.3. Retroalimentación de la comunidad sorda

En la última fase del proyecto, se recopiló retroalimentación sobre el desempeño del modelo *fine-tuneado*. Esta evaluación se llevó a cabo durante una reunión presencial en las oficinas de En-Señas, tal como se planteó en la metodología. Los participantes interactuaron con el sistema a través de la aplicación web local desarrollada, que se muestra en la Figura 179. Ellos destacaron la utilidad de la herramienta, especialmente porque la comunidad sorda utiliza frecuentemente ChatGPT para mejorar la redacción de sus comentarios escritos. Sin embargo, señalaron una limitación importante: el modelo estándar a menudo genera respuestas imprecisas, ya que no conoce la estructura gramatical y los matices lingüísticos de LENSEGUA. En contraste, los usuarios afirmaron que la versión *fine-tuneada* demostró una mejor comprensión de las características propias de LENSEGUA, lo cual se vio reflejado en las interpretaciones generadas.

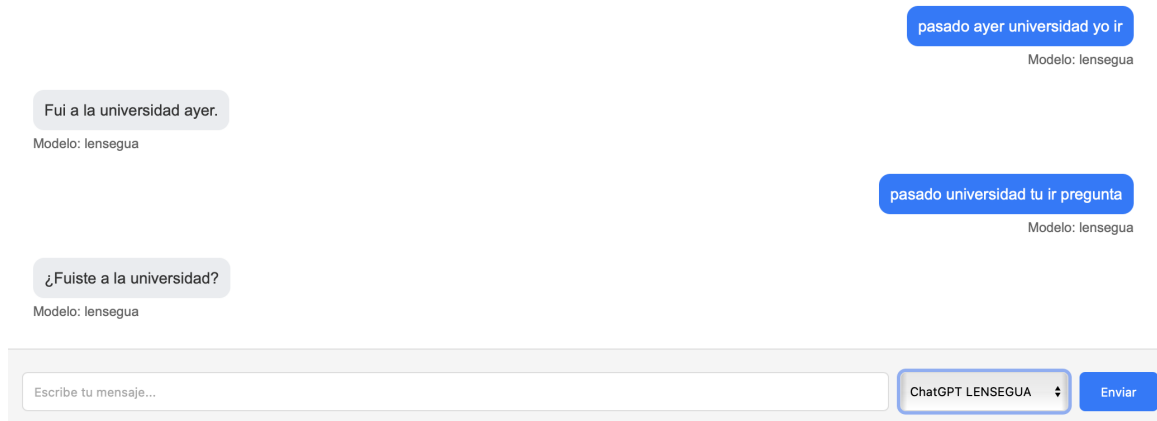


Figura 179: Aplicación web desarrollada para facilitar interacción con modelo *fine-tuneado*.

Para complementar la evaluación, se llevó a cabo una encuesta en Google Forms, que fue respondida no solo por los participantes de la reunión en En-Señas, sino también por otras personas sordas, intérpretes, estudiantes y usuarios familiarizados con LENSEGUA. En esta encuesta, un total de 19 personas evaluaron las interpretaciones generadas por ambos modelos en una escala del 1 al 5, donde 1 correspondía a menor coherencia y 5 a mayor coherencia o claridad. Cabe resaltar que dichas interpretaciones se generaron utilizando la quinta versión del *prompt* desarrollado.

En este contexto, la Tabla 30 presenta un ejemplo comparativo de las interpretaciones de la frase “ayer cine tu ir pregunta”. El modelo original generó la interpretación “Ayer fuiste al cine, ¿verdad?”, y obtuvo una puntuación promedio de 3.4. En cambio, el modelo *fine-tuneado* interpretó la frase como “¿Fuiste al cine ayer?”, alcanzando una puntuación de 4.8. Los 19 participantes estuvieron de acuerdo en que la interpretación del modelo *fine-tuneado* fue más precisa y adecuada.

Frase en LENSEGUA: ayer cine tu ir pregunta		
Modelo	Interpretación	Puntuación promedio
GPT-3.5-Turbo	Ayer fuiste al cine, ¿verdad?	3.4
<i>Fine-Tuneado</i>	¿Fuiste al cine ayer?	4.8

Tabla 30: Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “ayer cine tu ir pregunta”.

En la Tabla 31 se analiza la frase “pasado yo medicina comprar para mamá”. El modelo original generó la interpretación “En el pasado compré medicina para mi mamá”, obteniendo una puntuación promedio de 3.8. Por su parte, el modelo *fine-tuneado* simplificó la interpretación a “Compré medicina para mi mamá”, logrando una puntuación de 4.9. Si bien ambas interpretaciones son correctas, todos los participantes coincidieron en que la del modelo *fine-tuneado* fue más apropiada. Esto probablemente debido a que el modelo original añadió información -“en el pasado”- que suele omitirse.

Frase en LENSEGUA: pasado yo medicina comprar para mamá		
Modelo	Interpretación	Puntuación promedio
GPT-3.5-Turbo	En el pasado compré medicina para mi mamá.	3.8
<i>Fine-Tuneado</i>	Compré medicina para mi mamá.	4.9

Tabla 31: Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “pasado yo medicina comprar para mamá”.

La Tabla 32 presenta los resultados para la frase “ayer abuelo llamar tu pregunta”. El modelo original generó “Ayer tu abuelo te llamó para preguntarte algo”, y obtuvo una puntuación de 1.8. En contraste, el modelo *fine-tuneado* generó la frase “¿Ayer te llamó tu abuelo?”, obteniendo 4.9 puntos. Los participantes seleccionaron esta última interpretación como la más clara y correcta, posiblemente ya que capturaba de manera más fiel la idea que se buscaba transmitir.

Frase en LENSEGUA: ayer abuelo llamar tu pregunta		
Modelo	Interpretación	Puntuación promedio
GPT-3.5-Turbo	Ayer tu abuelo te llamó para preguntarte algo.	1.8
<i>Fine-Tuneado</i>	¿Ayer te llamó tu abuelo?	4.9

Tabla 32: Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “ayer abuelo llamar tu pregunta”.

En la Tabla 33 se muestra la evaluación de la frase “antes tienda tu amiga ver pregunta”. El modelo original generó “Tu amiga vio la tienda y te preguntó algo”, y obtuvo una puntuación de 1.4. El modelo *fine-tuneado* produjo “¿Viste a tu amiga en la tienda?”, alcanzando 4.1 puntos. Nuevamente, todos los participantes coincidieron en que la interpretación generada por el modelo adaptado era la más clara y coherente.

Frase en LENSEGUA: antes tienda tu amiga ver pregunta		
Modelo	Interpretación	Puntuación promedio
GPT-3.5-Turbo	Tu amiga vio la tienda y te preguntó algo.	1.4
<i>Fine-Tuneado</i>	¿Viste a tu amiga en la tienda?	4.1

Tabla 33: Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “antes tienda tu amiga ver pregunta”.

Finalmente, en la Tabla 34 se analiza la frase “tu opinión cuál pregunta”. El modelo original la interpretó como “Tu opinión sobre cuál es la pregunta”, obteniendo 1.4 puntos. Por otro lado, el modelo *fine-tuneado* generó “¿Cuál es tu opinión?”, logrando la máxima puntuación de 5.0 puntos. Todos los participantes estuvieron de acuerdo en que esta última interpretación fue la más adecuada.

Frase en LENSEGUA: tu opinión cuál pregunta		
Modelo	Interpretación	Puntuación promedio
GPT-3.5-Turbo	Tu opinión sobre cuál es la pregunta.	1.4
<i>Fine-Tuneado</i>	¿Cuál es tu opinión?	5.0

Tabla 34: Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo estándar y su versión *fine-tuneada* para la frase “tu opinión cuál pregunta”.

En la última fase de la encuesta, se solicitó a los participantes evaluar cinco interpretaciones de una misma frase. Cada una de las interpretaciones fue generada utilizando un *prompt* distinto, en conjunto con el modelo *fine-tuneado*. El objetivo fue identificar cuál de los cinco *prompts* producía la interpretación más adecuada y coherente según personas conocedoras de la gramática de LENSEGUA.

Como se puede observar en la Tabla 35, las primeras tres interpretaciones obtuvieron calificaciones promedio bajas (1.4, 1.1 y 1.4, respectivamente), ya que no lograron transmitir correctamente la idea original y contenían errores. En lugar de indicar que se esperaba que no hubiera tráfico, estas interpretaciones mencionaron lo opuesto, lo que distorsionó el mensaje. Por otro lado, la cuarta y quinta versión generaron interpretaciones mucho más adecuadas y claras, logrando transmitir el mensaje de manera efectiva. Es importante mencionar que estas últimas dos interpretaciones eran muy similares; sin embargo, debido a ligeras variaciones en la redacción, los usuarios indicaron que la interpretación generada por el quinto *prompt* fue superior, obteniendo una puntuación promedio de 5.0.

Finalmente, de los 19 participantes, 18 eligieron la interpretación generada con el quinto *prompt* como su favorita, mientras que solo uno optó por la interpretación generada con el cuarto *prompt*. Este resultado no solo sugiere que el quinto *prompt* fue el más efectivo, sino que también coincide con las puntuaciones de distancia de Levenshtein promedio mencionadas anteriormente.

Frase en LENSEGUA: hospital yo necesitar ir ahora ojalá mucho carro no porque emergencia		
Prompt	Interpretación	Puntuación promedio
Versión 1	Necesito ir al hospital ahora espero que haya muchos carros porque es una emergencia.	1.4
Versión 2	Necesito ir al hospital ahora espero que haya muchos carros disponibles no tengo carro porque es una emergencia.	1.1
Versión 3	Necesito ir al hospital ahora porque es una emergencia y ojalá haya mucho carro.	1.4
Versión 4	Necesito ir al hospital ahora espero que no haya mucho tráfico porque es una emergencia.	4.6
Versión 5	Necesito ir al hospital ahora mismo. Espero que no haya mucho tráfico porque es una emergencia.	5.0

Tabla 35: Puntuación promedio otorgada por usuarios finales a las interpretaciones generadas por el modelo GPT-3.5-Turbo *fine-tuneado*, utilizando diferentes *prompts*, para la frase “hospital yo necesitar ir ahora ojalá mucho carro no porque emergencia”.

6.3. Procesamiento de lenguaje natural (LLaMA)

6.3.1. Fine-tuning (versión 1)

Para encontrar la configuración óptima del modelo LLaMA en la tarea de interpretación de LENSEGUA, se realizaron varios experimentos preliminares utilizando una porción del *dataset*. Estos experimentos se centraron en identificar las configuraciones que generaban el mejor modelo en términos de pérdida de validación y de la métrica BLEU promedio.

En el primer experimento, se entrenó un modelo sin ajustes específicos para verificar su funcionamiento básico. Como se observa en la Figura 180, existía una tendencia al sobreentrenamiento, resultando en un valor BLEU promedio de 0.125, lo cual indicaba interpretaciones deficientes.

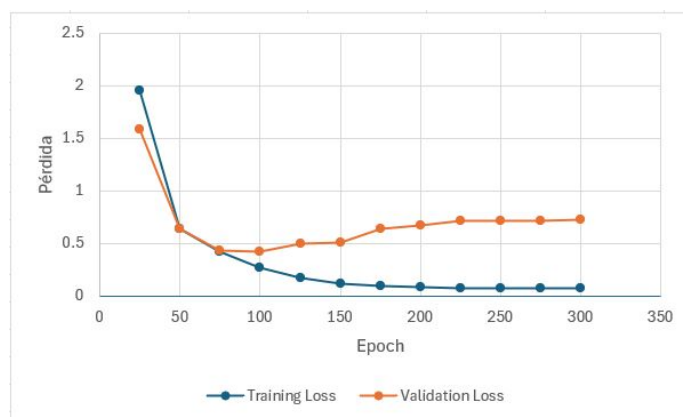


Figura 180: Evolución de la pérdida durante el *fine-tuning* del modelo LLaMA preliminar.

Para abordar este problema, se implementó la técnica de *early stopping* en el segundo experimento, reduciendo los *batches* de entrenamiento de 300 a 100. La Figura 181 muestra la evolución

de la pérdida durante este experimento, evidenciando una mejora significativa en la generalización del modelo. Se alcanzó un valor BLEU promedio de 0.212, lo cual refleja una mejora en la calidad de las predicciones generadas por el modelo.

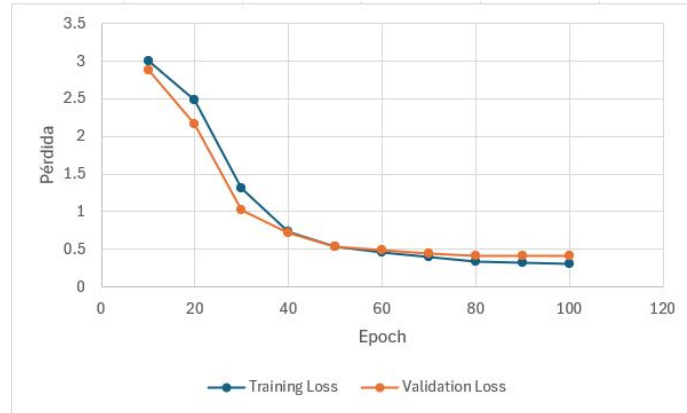


Figura 181: Evolución de la pérdida durante el *fine-tuning* del modelo LLaMA con *early stopping*.

Es importante mencionar que en todas las pruebas se utilizó LoRA. Sin embargo, durante el tercer experimento, se implementó una configuración estándar de LoRA, ajustando los parámetros de rango y alpha a valores de 8 y 16, respectivamente, siguiendo las recomendaciones de un estudio de traducción bilingüe [130]. La Figura 182 muestra cómo esta configuración, junto con el *early stopping*, resultó en una mejora en la pérdida del modelo. Este ajuste contribuyó a obtener un valor BLEU promedio de 0.239, representando un avance significativo respecto a los experimentos anteriores.

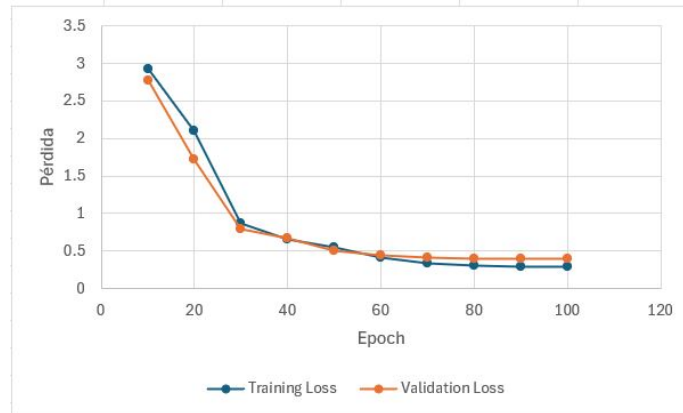


Figura 182: Evolución de la pérdida durante el *fine-tuning* del modelo LLaMA con *early stopping* y estándar LoRA.

Finalmente, en el cuarto experimento, se tomó la decisión de reducir la flexibilidad del modelo al pasar a un enfoque más estricto de *fine-tuning* supervisado. Es decir, se forzó al modelo a entrenar completamente bajo supervisión, eliminando la capacidad de decidir si entrenar de forma supervisada o no. A pesar de este cambio, se mantuvo el estándar de LoRA y se continuó utilizando la técnica de *early stopping*. Este ajuste alcanzó un valor BLEU promedio de 0.263, estableciendo el mejor rendimiento observado hasta el momento.

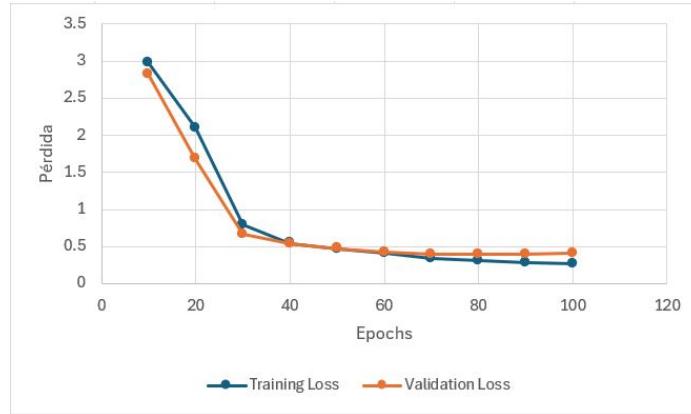


Figura 183: Evolución de la pérdida durante el *fine-tuning* del modelo LLaMA con *early stopping*, estándar LoRA y *fine-tuning* supervisado.

La Tabla 36 resume los resultados de estos experimentos preliminares, mostrando la progresión en el rendimiento del modelo con cada modificación implementada. Basándose en estos resultados, se seleccionó la configuración del cuarto experimento para realizar un entrenamiento final utilizando el conjunto completo de datos. Este entrenamiento permitió aumentar el valor BLEU promedio a 0.2894, lo cual confirma que la combinación de *early stopping*, estándar LoRA y un enfoque de *fine-tuning* supervisado estricto era la más adecuada para esta tarea.

Modelo	BLEU promedio
Modelo predeterminado	0.125
Modelo con <i>early stopping</i>	0.212
Modelo con <i>early stopping</i> y estándar LoRA	0.239
Modelo con <i>early stopping</i> , estándar LoRA y <i>fine-tuning</i> supervisado	0.263

Tabla 36: Valores promedio de BLEU en experimentos preliminares.

Para evaluar el rendimiento específico del modelo, se analizaron tres oraciones de prueba, cuyos resultados se presentan en la Tabla 37. Los valores BLEU variaron significativamente entre 1.0, 0.0 y 0.5, proporcionando información valiosa sobre la capacidad del modelo para manejar diferentes tipos de oraciones.

Interpretación esperada	Interpretación real	BLEU
¿Como esta el día hoy?	¿Como esta el día hoy?	1.0
Miguel gana la carrera.	Miguel gano la carrera.	0.0
Pon la flor azul sobre la mesa.	Pon la flor azul en la mesa	0.5

Tabla 37: Valores BLEU de interpretaciones generadas por el modelo LLaMA con *early stopping*, estándar LoRA y *fine-tuning* supervisado.

Tras completar el proceso de *fine-tuning* final, se aplicó la técnica de *Local Interpretable Model-agnostic Explanations* (LIME) para evaluar el impacto de este proceso en las interpretaciones generadas por el modelo. Esta técnica permitió identificar las palabras clave que ejercían mayor influencia en las interpretaciones generadas para frases específicas, proporcionando una visión más clara y detallada sobre cómo el modelo procesa y utiliza la información para generar sus predicciones. Para este

análisis, se seleccionaron tres frases representativas, lo que permitió explorar diferentes escenarios lingüísticos y evaluar el desempeño del modelo en contextos variados.

Dado que se empleó la librería de transformadores para este análisis, fue necesario trabajar con valores de logits y atenciones del transformador. Estos valores, aunque pequeños en magnitud, representan la importancia relativa de cada palabra en la interpretación final. Los logits y las atenciones reflejan la medida en que el modelo se enfoca en ciertos términos y su relevancia en la interpretación.

En la Figura 184, el análisis LIME para la frase “como clima hoy” muestra el impacto de las palabras y cómo estas influyen en la interpretación final generada por el modelo. Todos los valores son positivos, lo que indica que las palabras fueron entendidas y contribuyeron significativamente a la construcción de la interpretación, permitiendo resultados más precisos.

En esta figura, se observa que la palabra “como” obtuvo el mayor impacto, posiblemente debido a que el modelo reconoció que esta palabra representaba una pregunta. La referencia temporal con “hoy” y el término “clima” complementaron la construcción de la interpretación. En la Tabla 38 se presentan los resultados numéricos del análisis LIME.

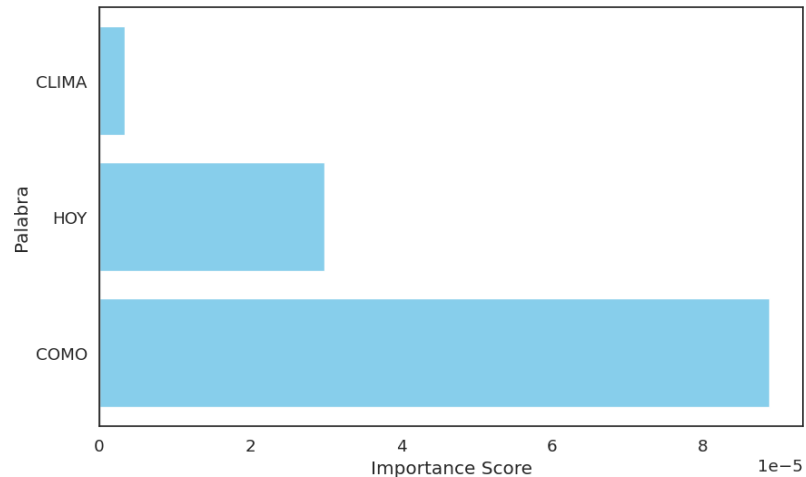


Figura 184: Resultados de LIME para la interpretación generada por el modelo LLaMA *fine-tuneado* (versión 1) para la frase “como clima hoy”.

Frase en LENSEGUA: como clima hoy	
Interpretación teórica: ¿Cómo está el clima hoy?	
Palabra	Valores de influencia (LIME) Modelo <i>fine-tuneado</i>
COMO	9.246e-05
HOY	3.395e-06
CLIMA	7.904e-07

Tabla 38: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 1) para la frase “como clima hoy”.

En la Figura 185, se analiza la frase “ayer viernes limpiar casa todo día”. Similarmente a la Figura anterior, todas las palabras tienen valores positivos, destacando términos como “día”, “todo” y “ayer” por su mayor relevancia. En este caso, las referencias temporales fueron mejor interpretadas, mientras que la acción “limpiar” presentó un menor valor de importancia.

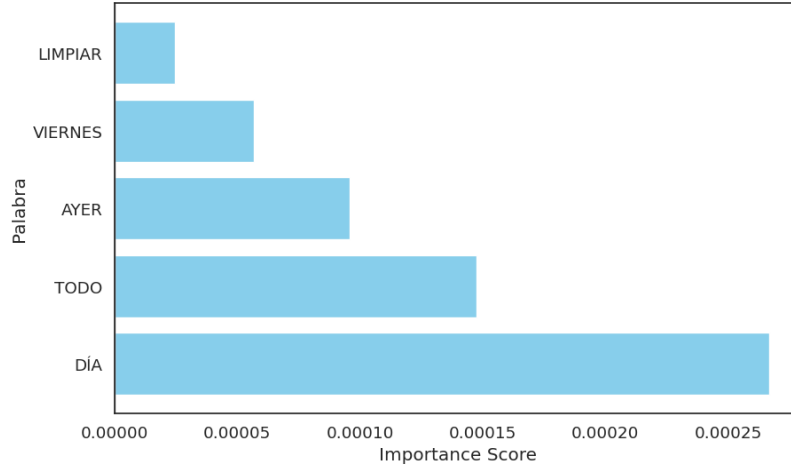


Figura 185: Resultados de LIME para la interpretación generada por el modelo LLaMA *fine-tuneado* (versión 1) para la frase “ayer viernes limpiar casa todo día”.

Frase en LENSEGUA: ayer viernes limpiar casa todo día	
Interpretación teórica: Ayer viernes limpié la casa todo el día.	
Palabra	Valores de influencia (LIME) Modelo <i>fine-tuneado</i>
DÍA	2.666e-4
TODO	1.475e-4
AYER	9.583e-05
VIERNES	5.679e-05
LIMPIAR	2.4804e-05

Tabla 39: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 1) para la frase “ayer viernes limpiar casa todo día”.

Finalmente, en la Figura 186 se analiza la frase “mar gustar él mar gustar ella no”. Este caso incluye una oración más larga y contiene una palabra repetida. El análisis muestra que la palabra repetida tiene menor importancia, mientras que el término “no” es crucial para establecer la distinción entre las dos entidades mencionadas. Sin embargo, se evidencia cierta confusión por parte del modelo con la palabra “ella”, que representa una entidad diferente.

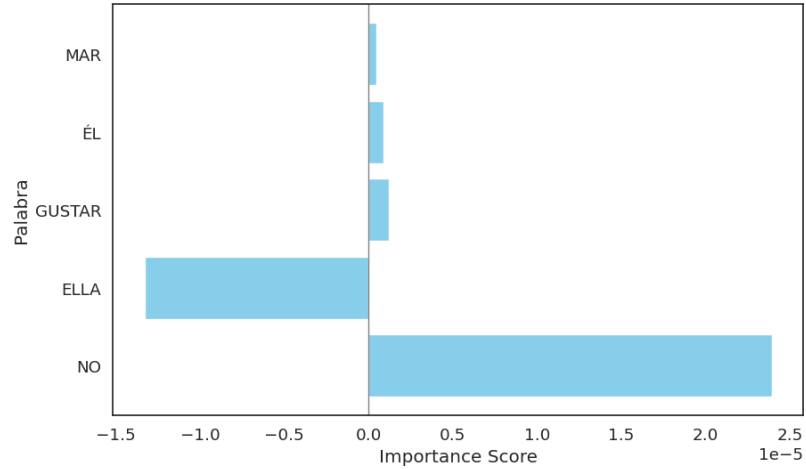


Figura 186: Resultados de LIME para la interpretación generada por el modelo LLaMA *fine-tuneado* (versión 1) para la frase “mar gusta él mar gustar ella no”.

Frase en LENSEGUA: mar gusta él mar gustar ella no	
Interpretación teórica: A él le gusta el mar, pero a ella no.	
Palabra	Valores de influencia (LIME) Modelo <i>fine-tuneado</i>
NO	2.393e-05
ELLA	-1.329e-05
GUSTAR	1.181e-06
ÉL	8.746e-07
MAR	4.894e-08

Tabla 40: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 1) para la frase “mar gusta él mar gustar ella no”.

6.3.2. Retroalimentación de la comunidad sorda

Al finalizar el entrenamiento, los resultados finales del *fine-tuning* se presentaron a una experta en LENSEGUA, quien evaluó las interpretaciones generadas por el modelo. Durante esta entrevista se analizaron diversas traducciones, destacando aspectos positivos y áreas de mejora. La Tabla 41 resume los comentarios proporcionados.

Interpretación teórica	Interpretación real	Comentario
¿Ustedes comen pastel?	Ustedes comieron pastel	La oración en sí esta bien. Sin embargo, esta en forma de interrogación, lo que no es según los signos específicas. Se necesita especificar con signos de puntuación.
El eligió entre dos casas.	El elige entre dos casas.	Buy buena traducción. Se guarda la presencia de la información. Lo que sí, es que esta en pasado la oración traducción. En las frases, no hay ninga referencia de la acción esta en pasado o presente.
A Juan no le gusta leer.	A Juan no le gusta leer.	Una traducción bastante perfecta. Guarda la información y tiene sentido con respecto a las señas utilizadas.
Pon la flor azul sobre la mesa.	Pon la flor azul en la mesa.	La traducción esta muy buena y es de alta calidad. Lo que hay que tener en mente es que en LENSEGUA, no existén los arículos o preposiciones, estos hay que inferirlos basados en las señas, y estas cambian basado en la interpretación de la persona.
SACAGRAPAS	Sacagawea	Esta traducción esta bastante mal. Se tiene la idea, pero las letras no coinciden con lo que se quiere decir exactamente.
La pizza de queso es mi favorita.	Pizzo es mi queso favorite	Hay una idea general de lo que se quiera hablar, que es el queso favorito. Sin embargo, esta a media la oración, y a pesar que la esencia básica esta ahí, no es aceptable de la misma manera y hay que mejorar en la base de datos para poder mejorar el resultado final.

Tabla 41: Comentarios otorgados por un intérprete de LENSEGUA ante interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 1) para diversas frases.

Uno de los puntos clave mencionados fue la dificultad del modelo para diferenciar entre preguntas y declaraciones. Por ejemplo, en la oración “¿Ustedes comen pastel?”, la interpretación generada fue “Ustedes comieron pastel”, lo cual no respeta el formato interrogativo. La experta recomendó incorporar marcadores específicos en el conjunto de datos para que el modelo pueda identificar adecuadamente el tipo de enunciado, evitando así confusiones.

Otro aspecto señalado fue la inconsistencia en los tiempos verbales. En la frase “Él eligió entre dos casas”, la traducción generada estuvo en presente, “Él elige entre dos casas”, en lugar de pasado. Esto refleja la necesidad de enriquecer el conjunto de datos con información que especifique el tiempo verbal, lo que permitiría obtener interpretaciones más precisas y acordes al contexto original.

Asimismo, se identificaron errores en algunas traducciones, como el caso de “SACAGRAPAS”, donde la interpretación fue “Sacagawea”, lo que muestra una falta de coincidencia entre las letras esperadas y las generadas. La experta destacó la importancia de mejorar la calidad y diversidad del conjunto de datos para abordar estos problemas y reducir la incidencia de errores. A pesar de estas limitaciones, ella reconoció que la mayoría de las interpretaciones eran comprensibles y lograban transmitir el significado general esperado. En palabras de la experta: “ambas dicen lo mismo, solo que de maneras diferentes”. Esto refleja que, aunque las traducciones podían diferir en estructura o formato, conservaban la esencia básica de LENSEGUA.

Finalmente, la experta otorgó una calificación de 9 sobre 10 a las traducciones generadas por el modelo, considerando que el contenido reflejaba de manera adecuada el contexto de la lengua de señas. Este puntaje positivo resalta la calidad general de las interpretaciones y su coherencia con las señas utilizadas, aunque subraya la importancia de seguir mejorando aspectos como la diferenciación de enunciados, la consistencia en tiempos verbales y la precisión en palabras específicas.

6.3.3. Fine-tuning (versión 2)

Los resultados del modelo LLaMA *fine-tuneado* (versión 1) mostraron un desempeño aceptable; sin embargo, con base en las métricas y comentarios obtenidos en la entrevista con la experta en LENSEGUA, se identificó la oportunidad de mejorar las métricas mediante un proceso de *fine-tuning* más exhaustivo. Manteniendo la misma configuración óptima previamente identificada, se llevó a cabo una nueva tarea de *fine-tuning* utilizando un conjunto de datos específicamente desarrollado para el modelo GPT-3.5-Turbo. Este conjunto de datos constaba de 4,192 entradas para entrenamiento y 200 entradas de validación.

A partir de este proceso, se observó una mejora sustancial en la métrica BLEU promedio del modelo, que pasó de 0.2894, obtenida con el conjunto de datos anterior, a 0.632 con el nuevo conjunto de datos. Esta mejora sustancial indica que el modelo es capaz de generar interpretaciones más cercanas a las referencias teóricas establecidas.

Dado que los modelos LLaMA (versión 2) y GPT-3.5-Turbo fueron entrenados con el mismo conjunto de datos, se decidió comparar su rendimiento utilizando la distancia de Levenshtein y el largo promedio de las interpretaciones. Como se observa en la Tabla [42](#), LLaMA generó interpretaciones con un promedio de 28.65 palabras y una distancia de Levenshtein de 6.185. Por otro lado, GPT-3.5-Turbo produjo interpretaciones más cercanas a las teóricas, con un promedio de 26.635 palabras y una distancia promedio menor de 3.375.

Modelo	Largo promedio de interpretaciones	Distancia de Levenshtein promedio	Porcentaje de diferencia
LLaMa 3.0 (versión 2)	28.65	6.185	15.27 %
GPT-3.5-Turbo	26.635	3.375	11.98 %

Tabla 42: Distancias de Levenshtein promedio calculadas a partir de las interpretaciones generadas por el modelo GPT-3.5-Turbo *fine-tuneado* y el modelo LLaMA *fine-tuneado* (versión 2)

Para profundizar en el análisis del comportamiento del modelo, se aplicó la técnica LIME al modelo LLaMA (versión 2), cuyos resultados se visualizan en las Figuras 187, 188 y 189. Además, se realizó una comparación con los valores de influencia LIME obtenidos con el modelo GPT-3.5-Turbo, presentados en las Tablas 43, 44 y 45. Esta comparación permitió entender cómo cada modelo asignaba diferentes niveles de relevancia a las palabras durante el proceso de interpretación.

En el análisis de la frase “ojalá hoy carro mucho no”, como se muestra en la Figura 187, el modelo LLaMA identificó “ojalá” como la palabra más influyente con un valor de 0.12, seguida por “hoy” con 0.05. La Tabla 43 revela que GPT-3.5-Turbo asignó la mayor relevancia a “hoy” con 0.34, seguido por “ojalá” con 0.19 y “carro” con 0.13, mostrando una distribución diferente de la importancia entre los elementos temporales y contextuales de la frase.

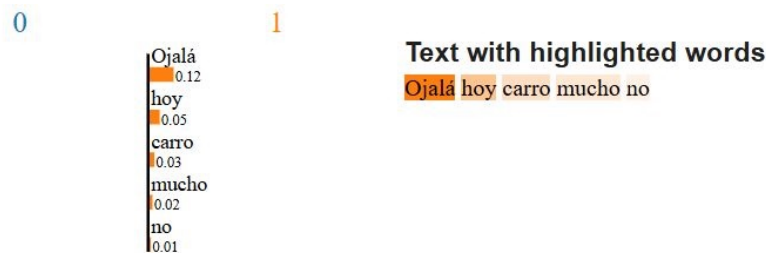


Figura 187: Resultados de LIME para la interpretación generada por el modelo LLaMA *fine-tuneado* (versión 2) para la frase “ojalá hoy carro mucho no”.

Frase en LENSEGUA: ojalá hoy carro mucho no		
Interpretación teórica: Espero que no haya tráfico hoy.		
Palabra	Valores de influencia (LIME)	
	LLaMA <i>fine-tuneado</i> (versión 2)	GPT-3.5-Turbo <i>fine-tuneado</i>
ojalá	0.12	0.19
hoy	0.05	0.34
carro	0.03	0.13
mucho	0.02	0.09
no	0.01	0.02

Tabla 43: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 2) y el modelo GPT-3.5-Turbo *fine-tuneado* para la frase “ojalá hoy carro mucho no”.

Para la segunda frase analizada, “antes tu policía llamar pregunta”, se observa una distribución uniforme de la influencia entre todas las palabras, con valores entre 0.00 y 0.02. Al examinar la Tabla 44, se evidencia que GPT-3.5-Turbo asignó valores de influencia significativamente más altos a palabras específicas: “policía” con 0.43 y “llamar” con 0.20, mientras que las demás palabras

mantuvieron valores por debajo de 0.10. Esto último sugiere que el modelo GPT-3.5-Turbo tiene una mejor comprensión de los elementos clave en la frase.

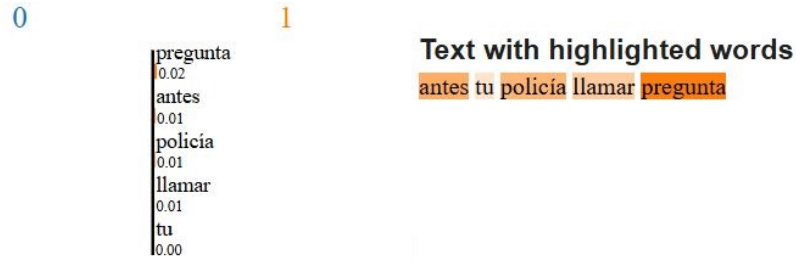


Figura 188: Resultados de LIME para la interpretación generada por el modelo LLaMA *fine-tuneado* (versión 2) para la frase “antes tu policía llamar pregunta”.

Frase en LENSEGUA: antes tu policía llamar pregunta		
Interpretación teórica: ¿Llamaste a la policía?		
Palabra	Valores de influencia (LIME)	
	LLaMA <i>fine-tuneado</i> (versión 2)	GPT-3.5-Turbo <i>fine-tuneado</i>
antes	0.01	0.03
tu	0.00	0.01
policía	0.01	0.43
llamar	0.01	0.20
pregunta	0.02	0.09

Tabla 44: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 2) y el modelo GPT-3.5-Turbo *fine-tuneado* para la frase “antes tu policía llamar pregunta”.

Finalmente, el análisis de la tercera frase, “pasado yo ir no”, se presenta en la Figura 189. El modelo LLaMA identificó el verbo “ir” como el elemento más influyente con 0.06, mientras que “yo” y “no” obtuvieron 0.02 cada uno, y “pasado” mostró una influencia mínima de 0.00. La Tabla 45 muestra que GPT-3.5-Turbo asignó la mayor relevancia a elementos diferentes: la palabra “no” alcanzó 0.44 de influencia, seguida por “pasado” con 0.24 e “ir” con 0.17, evidenciando una mayor sensibilidad a los modificadores temporales y la negación.

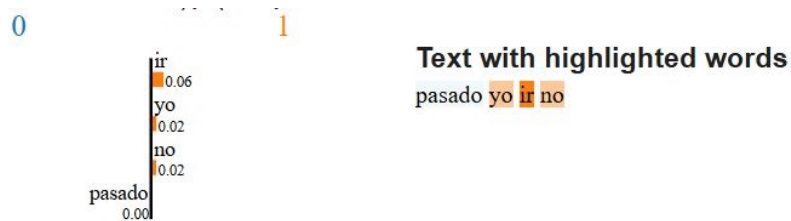


Figura 189: Resultados de LIME para la interpretación generada por el modelo LLaMA *fine-tuneado* (versión 2) para la frase “pasado yo ir no”.

Frase en LENSEGUA: pasado yo ir no		
Interpretación teórica: No fui.		
Palabra	Valores de influencia (LIME)	
	LLaMA <i>fine-tuneado</i> (versión 2)	GPT-3.5-Turbo <i>fine-tuneado</i>
pasado	0.00	0.24
yo	0.02	0.06
ir	0.06	0.17
no	0.02	0.44

Tabla 45: Contribución de cada palabra según los valores de influencia (LIME) en las interpretaciones generadas por el modelo LLaMA *fine-tuneado* (versión 2) y el modelo GPT-3.5-Turbo *fine-tuneado* para la frase “pasado yo ir no”.

Esta comparación entre ambos modelos revela patrones distintos en la asignación de relevancia a las palabras. LLaMA tiende a mantener una distribución más uniforme de la influencia, con valores relativamente bajos para la mayoría de las palabras y pequeñas diferencias entre ellas. En contraste, GPT-3.5-Turbo muestra una mayor variabilidad en los valores de influencia, asignando pesos significativamente más altos a ciertas palabras que considera clave para la interpretación de cada frase.

6.4. Infraestructura de red

6.4.1. Redireccionamiento de puertos para acceso SSH externo

Una vez asignados los recursos, se procede a la configuración de la red y acceso remoto para cada máquina virtual. Mediante la implementación de reglas de redireccionamiento de puertos **con iptables**, se permite el acceso a las VMs desde el exterior a través de SSH. Cada máquina virtual se ha configurado para ser accesible mediante un puerto específico: **2222 para VM1**, **2223 para VM2** y **2224 para VM3**. Este esquema de puertos permite una administración centralizada y segura desde cualquier ubicación externa, sin necesidad de modificar la configuración interna de las VMs.

- Para VM1 (puerto 2222)
 - `sudo iptables -t nat -A PREROUTING -p tcp --dport 2222 -j DNAT --to-destination 10.47.92.160:22`
 - `sudo iptables -t nat -A POSTROUTING -p tcp -d 10.47.92.160 --dport 22 -j MASQUERADE`
- Para VM2 (puerto 2223)
 - `sudo iptables -t nat -A PREROUTING -p tcp --dport 2223 -j DNAT --to-destination 10.47.92.70:22`
 - `sudo iptables -t nat -A POSTROUTING -p tcp -d 10.47.92.70 --dport 22 -j MASQUERADE`
- Para VM3 (puerto 2224)
 - `sudo iptables -t nat -A PREROUTING -p tcp --dport 2224 -j DNAT --to-destination 10.47.92.195:22`

- `sudo iptables -t nat -A POSTROUTING -p tcp -d 10.47.92.195 --dport 22 -j MASQUERADE`

Estas reglas fueron guardadas y aplicadas de manera persistente para asegurar su disponibilidad tras cualquier reinicio. El acceso SSH se configuró adicionalmente para utilizar claves públicas generadas desde sistemas Windows, facilitando una autenticación segura y sin contraseña.

En cuanto a la seguridad de las bases de datos, se decidió utilizar MySQL Server, gestionando la contraseña del usuario root para fortalecer la protección. El comando `ALTER USER` fue utilizado para actualizar la contraseña y asegurar que el acceso sea exclusivo desde el host local, minimizando riesgos.

Esta metodología garantiza una infraestructura virtualizada sólida, segura y eficiente, diseñada para soportar altas demandas y mantener la estabilidad bajo cargas intensivas.

6.4.2. Resultados de la prueba de carga

Para evaluar el rendimiento del servidor, se realizaron pruebas de carga con un número creciente de usuarios concurrentes. A continuación, se presentan los resultados obtenidos y las métricas de rendimiento observadas.

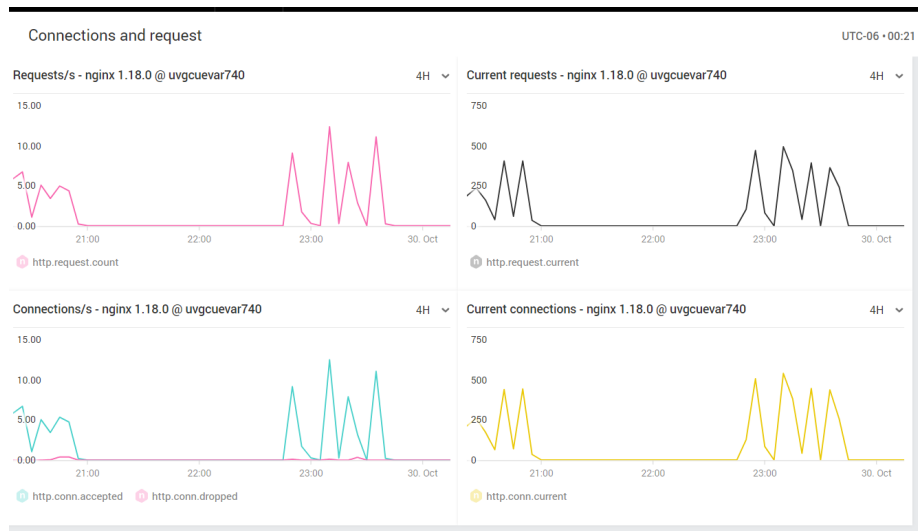


Figura 190: Métricas de conexiones y solicitudes

Requests/s y conexiones actuales

El servidor mostró un buen manejo de solicitudes por segundo (requests/s), manteniendo una cantidad de peticiones estable hasta las pruebas con una carga elevada. Esto demuestra que la configuración de NGINX y Gunicorn está bien ajustada para recibir y gestionar una alta cantidad de solicitudes concurrentes. Los picos de conexiones actuales también son gestionados adecuadamente, lo que indica que NGINX puede aceptar múltiples conexiones sin saturarse. La configuración con Gunicorn, que permite manejar conexiones asíncronas y de múltiples hilos, ayuda a procesar solicitudes de manera eficiente y a mantener baja la latencia.

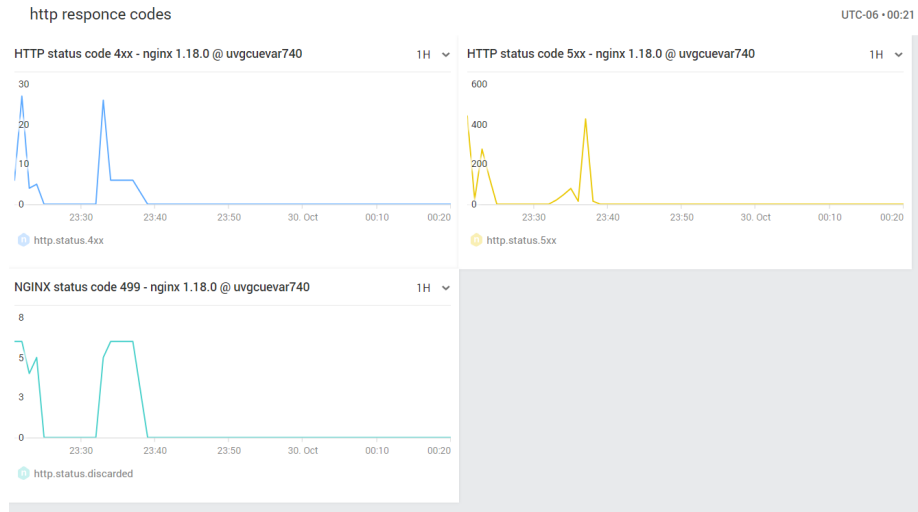


Figura 191: Códigos de respuesta HTTP

Códigos de respuesta HTTP

A medida que la carga se incrementó, comenzaron a aparecer errores 4xx y 5xx, particularmente en las pruebas de 500 usuarios en adelante. Sin embargo, los errores no fueron críticos y no afectaron significativamente el desempeño general del sistema. Esto sugiere que el sistema maneja bien la mayoría de las solicitudes válidas, y los errores observados son principalmente resultado de una sobrecarga extrema y no de una falla en la configuración de NGINX o Gunicorn. Los errores 499 indican que algunos clientes abandonaron la conexión antes de recibir respuesta, lo cual puede deberse a tiempos de respuesta más largos bajo carga alta.

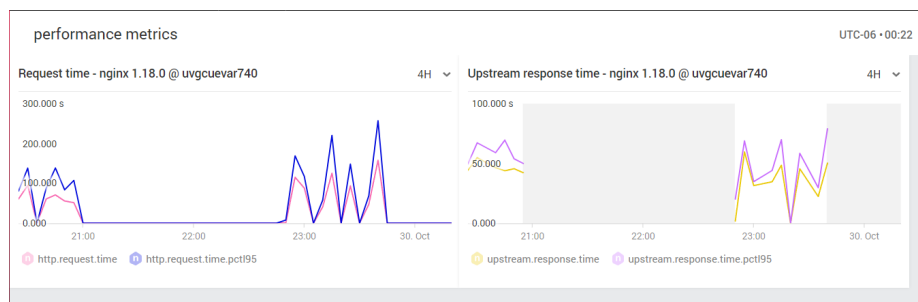


Figura 192: Tiempo de respuesta y tiempo de respuesta Upstream

Tiempo de respuesta y tiempo de respuesta Upstream

Los tiempos de respuesta aumentaron bajo cargas intensas, pero se mantuvieron razonablemente estables hasta los niveles de carga de 400-500 usuarios. Esto demuestra que la arquitectura NGINX-Gunicorn-Flask es capaz de manejar eficientemente el procesamiento de las solicitudes hasta un nivel considerable de concurrencia. A pesar de algunos picos, los tiempos de respuesta hacia los servicios backend (upstream) fueron manejados adecuadamente, lo que muestra que la configuración de Gunicorn en conjunto con Flask está optimizada para responder rápidamente.

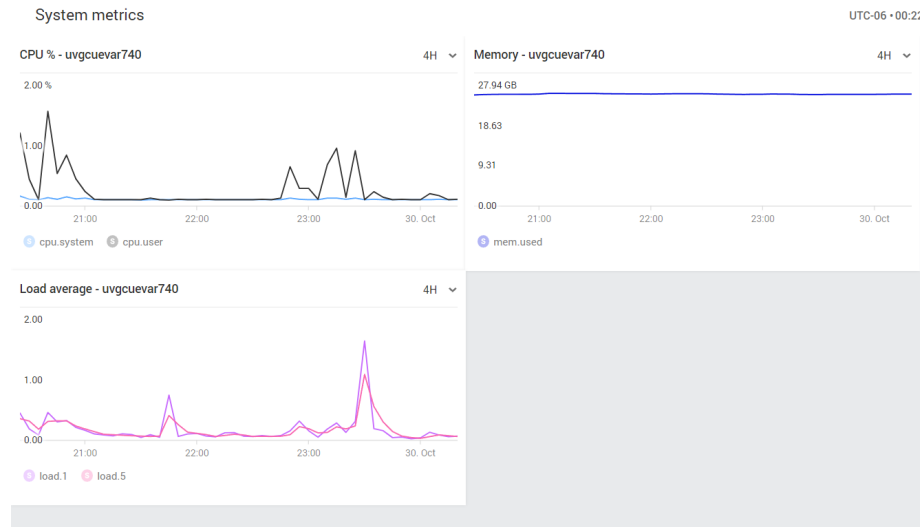


Figura 193: Métricas de CPU, memoria y carga del sistema

CPU, memoria y carga del sistema

Aunque se observan picos en el uso de CPU bajo carga intensa, el sistema utilizó eficientemente los recursos de CPU proporcionados (32 CPUs en total), manteniendo un uso moderado incluso en las pruebas con cargas más altas. Esto muestra que el sistema distribuye eficientemente las solicitudes entre los hilos de Gunicorn, sin sobrecargar los recursos de CPU. El uso de memoria se mantuvo bajo, lo cual es un indicativo de una configuración optimizada para el consumo de memoria. El load average del sistema permaneció dentro de límites aceptables, indicando capacidad para gestionar volúmenes de carga sin requerir ajustes adicionales.

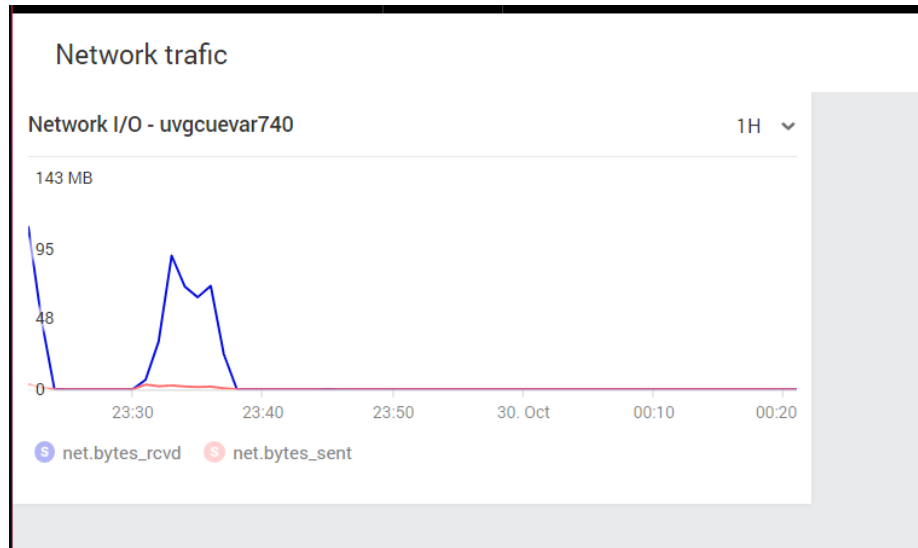


Figura 194: Tráfico de red

Tráfico de red

El tráfico de red aumentó durante las pruebas de carga, particularmente en el tráfico recibido (`net.bytes_rcvd`), lo cual es esperado debido al incremento de solicitudes. NGINX manejó eficientemente el enrutamiento de solicitudes y respuestas a través de la red, incluso a través de la VPN de la universidad, lo cual muestra que la configuración es robusta en entornos con restricciones de red.

Resumen de desempeño y justificación

En general, el sistema basado en NGINX, Gunicorn y Flask ha demostrado ser eficiente y efectivo para manejar solicitudes concurrentes bajo una variedad de cargas. Las métricas muestran que el sistema puede manejar hasta aproximadamente 400-500 usuarios concurrentes sin problemas mayores, mientras que las pruebas de carga más altas empiezan a mostrar signos de saturación. La configuración actual es adecuada para el entorno de red restringido de la universidad y demuestra estabilidad en condiciones de carga media-alta.

6.4.3. Resultados de la prueba E2E

Los resultados de esta prueba demostraron un éxito del 100%, ya que cada API involucrada respondió adecuadamente en el flujo esperado. La prueba incluyó las siguientes operaciones:

- Registro de usuario.
- Inicio de sesión.
- Consulta de información de perfil.
- Envío de video.
- Marcado y desmarcado del video como favorito.

- Envío de una traducción.
- Marcado y desmarcado de la traducción como favorita.
- Incremento de la racha de actividad (streak).

Cada una de estas operaciones se ejecutó sin errores, y los tiempos de respuesta se mantuvieron dentro de los niveles aceptables, garantizando la correcta integración de los diferentes módulos del sistema.

```

Lynis 3.0.7
=====
Lynis security scan details:
Hardening index : 58 [##### ]
Tests performed : 271
Plugins enabled : 1

Components:
- Firewall [V]
- Malware scanner [X]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status [?]
- Security audit [V]
- Vulnerability scan [V]

Files:
- Test and debug information : /var/log/lynis.log
- Report data : /var/log/lynis-report.dat

=====

Lynis 3.0.7
Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOFy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====
[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for all settings)
uvgcueva@uvgcueva740:~$ nano /etc/lynis/default.prf
  
```

Figura 195: Resultados de la prueba E2E, mostrando un 100% de éxito en todas las operaciones realizadas.

6.4.4. Resultados de la prueba de seguridad con Lynis

Para evaluar la seguridad de nuestro sistema, utilizamos Lynis, una herramienta de auditoría que examina la configuración del sistema, busca vulnerabilidades y proporciona un índice de robustez conocido como el "Hardening Index". Lynis es ampliamente utilizado en sistemas UNIX y Linux para evaluar y mejorar la seguridad del sistema. Los resultados de la prueba realizada en nuestro servidor se muestran en la Figura 196.

```

=====
Lynis security scan details:
Hardening index : 58 [##### ]
Tests performed : 271
Plugins enabled : 1

Components:
- Firewall [V]
- Malware scanner [X]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status [?]
- Security audit [V]
- Vulnerability scan [V]

Files:
- Test and debug information : /var/log/lynis.log
- Report data : /var/log/lynis-report.dat

=====

Lynis 3.0.7
Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOFy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====
[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for all settings)
uvgcueva@uvgcueva740:~$ nano /etc/lynis/default.prf
  
```

Figura 196: Resultado de la primera prueba de seguridad con Lynis mostrando un índice de robustez de 58.

Durante esta primera ejecución, obtuvimos un "Hardening Index" de 58. Según la documentación de Lynis y opiniones de expertos en foros de seguridad, un índice de robustez de 50 o superior indica que el sistema es seguro, aunque siempre hay áreas que podrían beneficiarse de mejoras adicionales. Este resultado nos da confianza en la configuración actual del sistema, aunque decidimos realizar ajustes adicionales para elevar este puntaje en pruebas posteriores.

```
=====
Lynis security scan details:
Hardening index : 60 [#####          ]
Tests performed : 277
Plugins enabled : 1

Components:
- Firewall           [V]
- Malware scanner    [V]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status [?]
- Security audit    [V]
- Vulnerability scan [V]

Files:
- Test and debug information : /var/log/lynis.log
- Report data                : /var/log/lynis-report.dat

=====

Lynis 3.0.7

Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOFy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====

[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for al
l settings)

uvgcueva@uvgcuevar748:~$
```

Figura 197: Resultado de la segunda prueba de seguridad con Lynis mostrando un índice de robustez de 60.

La Figura 197 muestra el resultado de nuestra segunda ejecución de Lynis después de aplicar mejoras recomendadas en la configuración del sistema. En esta segunda prueba, el índice de robustez aumentó a 60, lo que demuestra que las optimizaciones realizadas han mejorado la seguridad del servidor. En general, se considera que Lynis ofrece mejores resultados en sistemas basados en Fedora, donde las configuraciones de seguridad predeterminadas están más alineadas con las recomendaciones de la herramienta.

```
=====
Lynis security scan details:
Hardening index : 62 [#####          ]
Tests performed : 278
Plugins enabled : 1

Components:
- Firewall           [V]
- Malware scanner    [V]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status [?]
- Security audit    [V]
- Vulnerability scan [V]

Files:
- Test and debug information : /var/log/lynis.log
- Report data                : /var/log/lynis-report.dat

=====

Lynis 3.0.7

Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOFy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====

uvgcueva@uvgcuevar748:~$
```

Figura 198: Resultado de la tercera prueba de seguridad con Lynis mostrando un índice de robustez de 62.

En la Figura 198, se observa el resultado de la tercera y última prueba, donde auditoría realizada con Lynis arrojó un puntaje de 62, superando significativamente la meta establecida de 40. Este

resultado demostró que el sistema implementó medidas de seguridad suficientes para su propósito. Lynis, en su configuración básica y sin personalización, evaluó rigurosamente las configuraciones del sistema y detectó áreas críticas que necesitaron atención. Es importante destacar que este puntaje reflejó un sistema seguro que no había sido ajustado para ignorar reglas no aplicables a nuestro entorno, lo cual reforzó la validez del resultado obtenido. Según el artículo [Lynis hardening index \[84\]](#) publicado en Linux Audit, un puntaje entre 50 y 70 indica un sistema razonablemente fortalecido, ideal para aplicaciones. Este puntaje, sin configuraciones personalizadas, representó un sistema confiable y demostró que las medidas implementadas fueron más que adecuadas para el contexto en el que se desarrolló el proyecto.

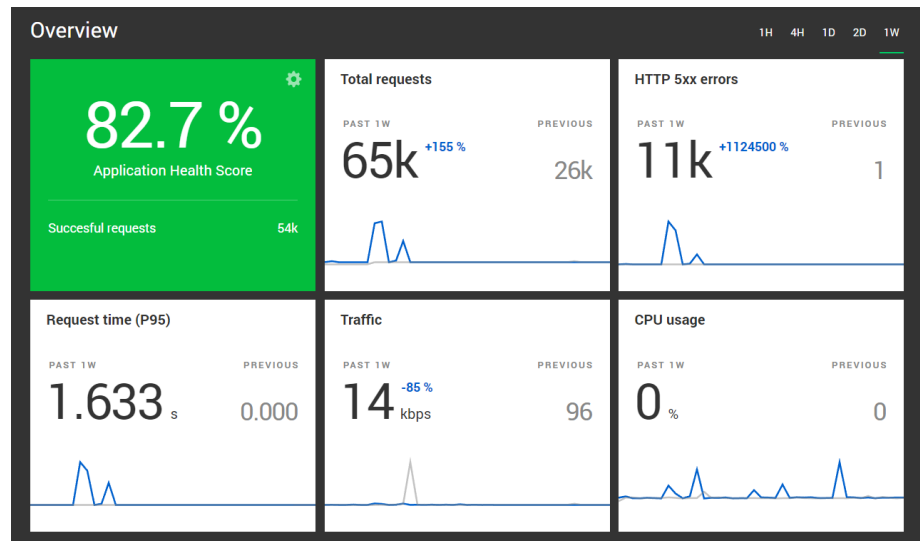


Figura 199: Resumen del estado del servidor.

Con base en las evaluaciones de seguridad, carga y pruebas de extremo a extremo (E2E), el sistema muestra un buen estado general (Figura [199](#)). El *Application Health Score* es de 82.7%, con un tiempo de respuesta promedio de 1.633 segundos, indicando una respuesta rápida y estable en la mayoría de las solicitudes. Aunque se registraron algunos errores HTTP 5xx, el uso de CPU se mantuvo bajo, reflejando una buena eficiencia en el manejo de recursos.

6.5. Diseño y desarrollo móvil

6.5.1. Comprensión de la situación de comunidad sorda en Guatemala

El análisis de mercado y el contexto guatemalteco revelaron que, aunque existen aplicaciones internacionales como Hand Talk Translator y SLAIT, ninguna se adapta a las necesidades específicas de la comunidad sorda en Guatemala, ni incluye LENSEGUA. Como resultado, se determinó la necesidad de desarrollar una solución local, específicamente diseñada para las barreras lingüísticas y culturales del país. Además, la revisión del Decreto 3-2020 confirmó la importancia de una herramienta que no solo facilite la comunicación entre sordos y oyentes, sino que también promueva el aprendizaje de LENSEGUA. Las entrevistas realizadas a personas sordas y a individuos en constante contacto con la comunidad sorda, junto con encuestas dirigidas a personas oyentes, subrayaron de manera contundente la necesidad de esta aplicación. Se evidenció que las barreras para las personas sordas en Guatemala son significativas y que existe un considerable desconocimiento sobre LENSEGUA, dado que el 70 % de los encuestados reconoció no estar familiarizado con esta lengua.

Esta información recalca la urgencia de desarrollar una herramienta diseñada específicamente para atender estas deficiencias.

6.5.2. Diseño centrado en el usuario

El diseño de la aplicación fue un proceso iterativo basado en varias herramientas de diseño, como mapas de empatía, personas, diagramas de afinidad y flujos de usuario. Este proceso dio como resultado un prototipo interactivo de alto nivel desarrollado en Figma. Dicho prototipo permitió definir de manera clara la navegación y la experiencia del usuario. El prototipo integró las recomendaciones de asociaciones como En-Señas y expertos en diseño, lo que aseguró que la aplicación no solo fuera funcional, sino también accesible y fácil de usar para la comunidad sorda.

6.5.3. Desarrollo de aplicación móvil para Android

El desarrollo de la aplicación fue llevado a cabo utilizando una arquitectura modular basada en el patrón MVVM, lo que permitió que la aplicación fuera fácilmente escalable y mantenible. Se implementaron componentes reutilizables, lo que aumentó la eficiencia en el desarrollo, reduciendo la redundancia en el código. Además, se integraron diversas librerías que optimizaron el rendimiento general de la aplicación. El uso de Kanban durante el desarrollo facilitó una gestión eficaz de las tareas, asegurando que se cumplieran los plazos y que las funcionalidades clave se implementaran correctamente. El desarrollo resultó en una versión funcional de la aplicación lanzada en fase de prueba cerrada en *Play Store*, lo que permitió a un grupo selecto de usuarios interactuar con ella antes de su lanzamiento oficial.

6.5.4. Pruebas con usuarios finales

Durante las pruebas, se presentaron los flujos críticos de la aplicación en eventos clave como la Expo UVG y una presentación especial con la asociación En-Señas. El resultado de estas pruebas fue altamente positivo, ya que los usuarios destacaron la intuitividad y facilidad de uso de la aplicación. Se registró un 90% de éxito en las pruebas de usabilidad con En-Señas, en las cuales los usuarios pudieron completar tareas clave como la creación de cuentas, grabación de videos y traducción de señas sin dificultades importantes. Estas pruebas confirmaron que la aplicación era funcional, accesible y cumplía con las expectativas de los usuarios, demostrando cómo la investigación, el diseño y el desarrollo se integraron armoniosamente para formar un producto final exitoso. Esto reafirma que cada fase fue ejecutada correctamente, contribuyendo al logro de un resultado sólido y coherente.

A lo largo del proyecto, se abordaron las necesidades específicas de la comunidad sorda en Guatemala, prestando especial atención a la adaptación cultural y a la integración de LENSEGUA, la lengua de señas guatemalteca. En esta sección, se analizarán los hallazgos clave y las lecciones aprendidas en cada fase del proyecto, destacando los desafíos y las soluciones implementadas para crear una herramienta que responda a las demandas comunicativas y educativas de sus usuarios.

7.1. Visión por computadora

Desempeño del modelo

El modelo final tiene una precisión de 0.8802, una sensibilidad de 0.9685 y una puntuación F1 de 0.9706. Esto indica que el modelo tiene un buen desempeño en la clasificación de los gestos de lengua de señas de Guatemala. Adicional a esto, el modelo tiene la capacidad de procesar 19.38 fotogramas por segundo, lo cual indica que el modelo es capaz de reconocer los gestos en tiempo real con una latencia baja. En otras palabras, para un video grabado a 30 fotogramas por segundo, la velocidad más comúnmente utilizada en dispositivos móviles, el modelo es capaz de procesar los fotogramas a un 64.6 % de la velocidad de grabación. Al utilizar un video grabado a 24 fotogramas por segundo, la velocidad relativa del modelo aumenta a un 80.75 % de la velocidad de grabación.

Para la elección del modelo final, se consideraron varios factores, como la precisión, la sensibilidad, y la puntuación F1. No se consideró la velocidad de procesamiento del modelo, ya que en general los modelos de reconocimiento de lengua de señas presentaron velocidades de procesamiento muy similares. La precisión, la cual indica la proporción de predicciones correctas, fue el factor más importante en la elección del modelo final, ya que se consideró que era importante que el modelo fuera capaz de reconocer correctamente los gestos de lengua de señas. Sin embargo, la sensibilidad y la puntuación F1 también fueron factores importantes, ya que indican la proporción de verdaderos positivos y la precisión del modelo, respectivamente.

Un modelo con una alta precisión, pero que produjera muchos falsos positivos o tuviera una baja sensibilidad, no sería útil en la práctica, ya que no sería capaz de reconocer correctamente los gestos de lengua de señas. En el caso de un modelo con baja sensibilidad, este modelo no sería capaz de predecir en la mayoría de los casos, lo cual resultaría en un modelo poco útil. Por otro lado, un

modelo con una alta sensibilidad, pero que produjera muchos falsos positivos, tampoco sería útil en la práctica, ya que produciría mucho ruido en las predicciones. Balancear estos dos factores es importante para obtener un modelo con un buen desempeño en la clasificación de los gestos de lengua de señas, lo cual se logró con el modelo final.

En las pruebas realizadas en tiempo real, el modelo final fue capaz de reconocer correctamente los gestos de lengua de señas de Guatemala en la mayoría de los casos. De los 192 videos de prueba, el modelo fue capaz de reconocer correctamente los gestos de lengua de señas en 169, lo cual corresponde a un 88 % de precisión. En la Figura 166 se puede observar el desempeño del modelo en tiempo real. Sin embargo, en las pruebas de reconocimiento de múltiples palabras, el modelo no fue capaz de reconocer correctamente los gestos de lengua de señas. De los 27 videos de prueba, el modelo no fue capaz de reconocer correctamente los gestos de lengua de señas en ninguno de los casos.

Limitaciones del modelo

A pesar de que el modelo final tiene un buen desempeño en la clasificación de los gestos de lengua de señas de Guatemala, existen algunas limitaciones que deben ser consideradas. Una de las limitaciones del modelo es que no es capaz de reconocer correctamente los gestos de lengua de señas en tiempo real en todos los casos. En las pruebas realizadas con un conjunto de datos de una sola palabra por video, el modelo fue capaz de reconocer la seña correctamente en un 88 % de los casos. Sin embargo, en las pruebas de reconocimiento de múltiples palabras, el modelo no fue capaz de reconocer correctamente los gestos de lengua de señas en ninguno de los casos. Esto indica que el modelo tiene dificultades para reconocer los gestos de lengua de señas en tiempo real cuando se presentan múltiples palabras en un video.

Esta limitación se debe a varios factores, entre los cuales esta el conjunto de datos de entrenamiento y la forma en la que se identifica el inicio y el fin de una seña. Al realizar el análisis de similitud entre las clases, se observó que algunas clases tenían un alto grado de similitud en los movimientos de las manos, lo cual puede dificultar la distinción entre ellas. Estas clases que tienen un alto grado de similitud son las que presentan mayores dificultades para el modelo, lo cual lleva a que se produzcan errores en la clasificación de los gestos de lengua de señas. Adicional a esto, el análisis de similitud entre todas las clases mostró que la mayoría de las clases presentan cierto grado de superposición, lo cual puede dificultar la distinción entre ellas.

Esta limitación proveniente del conjunto de datos esta directamente relacionada con la forma en la que se identifica el inicio y el fin de una seña, la cual constituye otra limitación del modelo. Debido a que el modelo distingue el inicio y el fin de una seña utilizando un umbral de confianza y un tiempo mínimo de duración, es posible que el modelo no sea capaz de identificar correctamente el inicio y el fin de una seña en todos los casos. La superposición entre las clases dificulta la identificación del inicio de una seña, ya que durante la transición entre una seña y otra, el modelo no puede predecir con certeza cuál es la seña que se está realizando. Al no ser capaz de identificar correctamente la seña durante estos periodos de transición, el modelo puede producir errores en la clasificación de los gestos de lengua de señas. Esto lleva a un efecto de acumulación de errores, y hace que cada seña consecutiva sea más difícil de reconocer correctamente.

Como se mencionó anteriormente, el conjunto de datos de entrenamiento fue un factor limitante en el desempeño del modelo. Adicional a la superposición entre las clases, el conjunto de datos de entrenamiento también presentó desafíos en el balance de las clases. Debido a que el conjunto de datos está compuesto por los fotogramas de los videos, y no por los videos completos, es posible que algunas clases tengan más fotogramas que otras. Esto sucede con las señas que tienen duración más larga, ya que estas señas tienen más fotogramas en el conjunto de datos. Al tener más fotogramas, estas clases tienen una mayor probabilidad de ser seleccionadas durante el entrenamiento, lo cual puede llevar a un desbalance en el conjunto de datos.

El desbalance en el conjunto de datos puede llevar a que el modelo tenga dificultades para reconocer las clases minoritarias, ya que estas clases tienen menos ejemplos en el conjunto de datos. Esto se podría solucionar utilizando técnicas de disminución de la clase mayoritaria, como la de submuestreo, pero esto podría llevar a una pérdida de información en el conjunto de datos. Debido a que no se puede seleccionar los fotogramas a eliminar del conjunto de datos para el submuestreo, es posible que se eliminen fotogramas importantes para la clasificación de los gestos de lengua de señas. La eliminación de estos fotogramas podría llevar a una pérdida de información en el conjunto de datos, lo cual podría llevar a un desempeño inferior del modelo.

La forma correcta de lidiar con el desbalance en el conjunto de datos es desde la recolección de los datos, ya que no se puede solucionar completamente en la etapa de entrenamiento. Al recolectar los datos, es importante asegurarse de que todas las clases tengan una cantidad similar total de fotogramas en el conjunto de datos. Para este proyecto, el enfoque de la recolección de datos fue recolectar la misma cantidad de videos para cada clase, pero no se consideró la duración de los videos. Al considerar la duración de los videos, se podrían haber recolectado más videos para las clases que tienen una duración más corta, y menos videos para las clases que tienen una duración más larga. Esto habría permitido tener un conjunto de datos más balanceado, y habría mejorado el desempeño del modelo en la clasificación de las señas.

Cumplimiento de los objetivos

El objetivo principal de este proyecto era desarrollar un modelo de reconocimiento de lengua de señas de Guatemala en tiempo real, el cual fuera capaz de reconocer un conjunto establecido de palabras. Este objetivo se logró, ya que el modelo final fue capaz de reconocer correctamente las palabras de lengua de señas de Guatemala en las pruebas realizadas en tiempo real, con una precisión de 0.8802, una sensibilidad de 0.9685 y una puntuación F1 de 0.9706. Adicional a esto, se creó un proyecto de código abierto en GitHub, el cual promueve el desarrollo de futuras investigaciones en el campo del reconocimiento de lengua de señas.

Adicional al objetivo principal, se plantearon varios objetivos específicos, la mayoría de los cuales se lograron. El primer objetivo específico era recolectar un conjunto de datos de lengua de señas de Guatemala, el cual se logró con la recolección de 960 videos de 32 palabras de la lengua de señas de Guatemala. El segundo objetivo específico era preprocesar el conjunto de datos, el cual se logró con la edición, el etiquetado manual de los videos y la extracción de los fotogramas de los videos. El tercer objetivo específico era crear un sistema con la capacidad de reconocer los puntos clave de las señas, el cual se logró con la implementación de aplicaciones de reconocimiento de lengua de señas en tiempo real. El cuarto objetivo específico era entrenar un modelo de reconocimiento de lengua de señas de Guatemala, el cual se logró con la implementación de un modelo de redes neuronales, el cual fue entrenado con el conjunto de datos recolectado.

El quinto objetivo específico era diseñar un sistema de reconocimiento de lengua de señas de Guatemala en tiempo real, el cual fuera capaz de identificar el inicio y el fin de una seña, con el objetivo de reconocer múltiples palabras en un video. Este objetivo no se logró completamente, ya que el modelo no fue capaz de reconocer correctamente los gestos de lengua de señas en tiempo real cuando se presentaban múltiples palabras en un video. Debido a las limitaciones del modelo, el sistema no fue capaz de identificar correctamente el inicio y el fin de las señas, lo cual llevó a errores acumulativos en la clasificación de las señas. A pesar de esto, el proyecto logró cumplir con la mayoría de los objetivos planteados, y logró desarrollar un modelo de reconocimiento de lengua de señas de Guatemala en tiempo real.

7.2. Procesamiento de lenguaje natural (GPT-3.5-Turbo)

El proceso de *fine-tuning* del modelo GPT-3.5-Turbo funcionó por dos razones principales. En primer lugar, se contó con el apoyo de intérpretes de LENSEGUA para la generación del *dataset*, lo que garantizó que los datos reflejaran fielmente la gramática y las estructuras de LENSEGUA. La calidad y autenticidad de estos datos fue esencial para que el modelo capturara los matices del lenguaje, evitando interpretaciones inconsistentes o erróneas. Además, la cantidad de datos recopilados y la variación dentro del *dataset* fue fundamental. Sin una muestra representativa y suficientemente diversa, el modelo no habría podido desarrollar la capacidad de interpretar tanto frases simples como complejas, ni hubiera aprendido a generalizar de forma efectiva en escenarios no vistos durante el entrenamiento.

La optimización de hiperparámetros también fue importante para el éxito del *fine-tuning*. A través de las diversas pruebas, se identificó que un entrenamiento con 2 *epochs* proporcionaba el balance óptimo entre aprendizaje y generalización, evitando el sobreajuste que se observó con *epochs* adicionales. Por ejemplo, con 3 *epochs* se observó que el modelo comenzaba a memorizar patrones específicos en lugar de aprender reglas generales de la lengua. De manera similar, el *learning rate multiplier* de 0.2 demostró ser ideal para permitir ajustes graduales en los pesos del modelo, garantizando un aprendizaje estable y progresivo. Este valor contrastó significativamente con pruebas realizadas con valores más altos (como 2.0), los cuales generaban actualizaciones demasiado bruscas que impedían una convergencia adecuada del modelo.

La efectividad del *fine-tuning* se pudo validar mediante el análisis LIME, el cual reveló cambios significativos en la forma en que el modelo procesaba elementos lingüísticos clave de LENSEGUA. Más específicamente, este análisis demostró que el modelo *fine-tuneado* logró aprender a interpretar palabras contextuales específicas que, aunque no deben aparecer explícitamente en las interpretaciones finales, son esenciales para la comprensión del mensaje en LENSEGUA, como “pregunta”, “futuro” y “pasado”. Estas palabras cumplen un rol único, porque representan conceptos que en el español estándar se comunican mediante el uso de signos de interrogación o conjugaciones verbales.

El análisis mostró que el modelo estándar asignaba puntuaciones de importancia negativas o muy bajas a estas palabras. Por ejemplo, “pregunta” recibió un valor de -0.09, “futuro” -0.04 y “pasado” 0.08. Estos resultados sugieren que el modelo percibía estas palabras como elementos disruptivos para la interpretación. Esta percepción errónea puede atribuirse a que el modelo GPT-3.5 original no fue entrenado específicamente para trabajar con las estructuras lingüísticas de LENSEGUA. Por lo tanto, el modelo estándar simplemente intentaba acoplar todas las palabras disponibles para generar una oración coherente en español, añadiendo conectores, preposiciones, artículos y otros elementos típicos de la gramática estándar. Sin embargo, al operar de esta manera, el modelo no lograba capturar el significado implícito que estas palabras tienen en LENSEGUA.

En contraste, el modelo adaptado mediante *fine-tuning* demostró un cambio significativo en la forma de interpretar estas palabras, asignando valores positivos a “pregunta” (0.09), “futuro” (0.09) y “pasado” (0.24). Esto evidencia que el modelo aprendió a reconocer la relevancia de estos términos en la construcción de interpretaciones coherentes en el contexto de LENSEGUA, lo cual fue fundamental para garantizar interpretaciones alineadas con el significado original.

El refinamiento iterativo de los *prompts* también fue fundamental para optimizar el desempeño del modelo *fine-tuneado*. Esto debido a que, en primer lugar, se desarrollaron instrucciones más específicas y detalladas que explicaban claramente al modelo cómo debía procesar las estructuras gramaticales de LENSEGUA. Se establecieron formatos precisos para las respuestas esperadas, definiendo exactamente cómo debían estructurarse las interpretaciones y qué elementos debían incluirse o excluirse. Por ejemplo, se especificó que ciertos marcadores gramaticales, como “pregunta”, no debían aparecer en la interpretación final, sino que debían usarse como guías para ajustar la estructura de la oración resultante.

La efectividad de estas mejoras en los *prompts* se comprobó mediante el análisis de las distancias de Levenshtein, una métrica que mide cuántas ediciones se necesitan para convertir una interpretación generada por el modelo en una interpretación de referencia establecida. En este caso, un valor menor indica mayor similitud y, por lo tanto, mejor rendimiento del modelo. Con la primera versión del *prompt*, el modelo *fine-tuneado* ya mostraba un rendimiento superior al modelo estándar, con una distancia promedio de 5.815 frente a 10.065, lo que representa una mejora inicial del 42.23%. Esta diferencia significativa en el rendimiento base confirma que el proceso de *fine-tuning* estableció una base sólida para la comprensión de las estructuras lingüísticas de LENSEGUA.

Las iteraciones sucesivas en el diseño de los *prompts* permitieron mejorar progresivamente el desempeño del modelo *fine-tuneado*. En realidad, la versión final del *prompt* logró reducir la distancia de Levenshtein promedio del modelo *fine-tuneado* a 3.375, lo que representa una mejora del 41.96% respecto a la primera versión. Esta reducción significativa demuestra que la especificación más precisa de las instrucciones permitió al modelo generar interpretaciones más cercanas a las referencias definidas. Cada iteración en el diseño de los *prompts* aportó mayor claridad sobre cómo manejar casos específicos de la gramática de LENSEGUA, lo que se reflejó directamente en la calidad de las interpretaciones.

Por otro lado, los resultados del modelo estándar no siguieron esta tendencia de mejora. De hecho, las métricas de este modelo se mantuvieron consistentemente elevadas con todos los *prompts*. Entre la primera y la tercera versión del *prompt*, la distancia promedio calculada solo disminuyó un 13.96%. Más aún, de la tercera versión a la última, se observó un ligero aumento del 0.23% en la distancia de Levenshtein promedio. Estos resultados indican que incluso las instrucciones más detalladas y estructuradas no pudieron compensar la falta de un entendimiento básico de las estructuras lingüísticas de LENSEGUA que proporciona el *fine-tuning*. El *prompt engineering* funciona mejor cuando el modelo ya tiene una comprensión fundamental del lenguaje, permitiendo que las instrucciones refinadas guíen y optimicen este conocimiento base.

Los resultados de las encuestas realizadas con la comunidad sorda reafirmaron esta diferencia en el rendimiento. Los participantes mostraron una clara preferencia por las interpretaciones del modelo *fine-tuneado*, destacando su coherencia y claridad. Este modelo logró captar matices específicas de la lengua, generando interpretaciones más acordes con las expectativas de los usuarios. En contraste, el modelo estándar se limitó a agregar conectores y preposiciones a las frases originales, produciendo oraciones gramaticalmente correctas pero que a menudo no transmitían adecuadamente la idea principal.

Por ejemplo, la frase en LENSEGUA “ayer abuelo llamar tu pregunta” fue interpretada de manera muy diferente por ambos modelos. El modelo estándar de GPT-3.5-Turbo generó la interpretación “Ayer tu abuelo te llamó para preguntarte algo”, que recibió una puntuación promedio de 1.8 por parte de los usuarios. Aunque es gramaticalmente correcta, esta respuesta realmente no captó la esencia de la frase original. Por otro lado, el modelo *fine-tuneado* interpretó la misma frase como “¿Ayer te llamó tu abuelo?”, obteniendo una puntuación promedio de 4.9. Esta interpretación no solo era más concisa, sino que también mantuvo la idea que se buscaba transmitir.

La Tabla 35 muestra evidencia adicional del impacto positivo del refinamiento de los *prompts* en la efectividad del modelo *fine-tuneado*. A medida que se ajustaron los *prompts*, los participantes de la encuesta evaluaron las interpretaciones generadas para la frase “hospital yo necesitar ir ahora ojalá mucho carro no porque emergencia”. La primera versión del *prompt* produjo una interpretación confusa, “Necesito ir al hospital ahora espero que haya muchos carros porque es una emergencia”, recibiendo una puntuación promedio de 1.4. En lugar de expresar que se esperaba que no hubiera tráfico, la interpretación resultante indicó lo opuesto, lo que distorsionó completamente el mensaje. Sin embargo, la versión final logró una interpretación precisa y contextualmente adecuada, alcanzando una puntuación máxima de 5.0.

La interpretación final, “Necesito ir al hospital ahora mismo. Espero que no haya mucho tráfico porque es una emergencia”, no solo reflejó correctamente el significado original, sino que también

transmitió la urgencia del mensaje. Esta mejora sustancial se debe a que los últimos *prompts* eran más completos e introdujeron aclaraciones específicas para abordar problemas detectados en las interpretaciones anteriores. Nuevamente, se incluyeron instrucciones detalladas sobre cómo manejar términos temporales, cómo interpretar correctamente las palabras clave, y cómo estructurar las interpretaciones finales de manera que transmitieran el tono y el contexto adecuados.

En conclusión, este análisis demuestra que la combinación de *fine-tuning* y *prompt engineering* fue fundamental para enseñarle al modelo GPT-3.5-Turbo a interpretar este tipo de frases. Los resultados obtenidos confirman que la tarea mencionada era demasiado compleja para ser resuelta únicamente mediante *prompt engineering*. Nuevamente, el modelo estándar, incluso con *prompts* refinados, no logró generar interpretaciones satisfactorias consistentemente. Sin embargo, es importante destacar que el *fine-tuning* por sí solo, sin un *prompt* óptimo, aunque genera buenos resultados, tampoco alcanza el nivel máximo de rendimiento.

7.3. Procesamiento de lenguaje natural (LLaMA)

La implementación de LLaMa como intérprete de LENSEGUA fue efectiva, gracias al proceso iterativo de experimentación para encontrar la configuración óptima. Este proceso involucró tres mejoras: la implementación de *early stopping*, la aplicación del estándar LoRA, y la decisión de asegurar que el proceso de *fine-tuning* se realizara únicamente de manera supervisada.

El *early stopping* ayudó a prevenir el sobreajuste (*overfitting*). En las primeras iteraciones, se observó que el modelo alcanzaba una métrica BLEU promedio de apenas 0.125, indicando interpretaciones deficientes. Este bajo rendimiento se debía a que el modelo, al continuar entrenando sin restricciones, comenzaba a memorizar en lugar de aprender patrones generalizables. La implementación de *early stopping* permitió detener el entrenamiento en el punto óptimo, mejorando la métrica BLEU promedio a 0.21.

Es importante señalar que, aunque la métrica BLEU proporciona un indicador útil del progreso del entrenamiento, presenta limitaciones significativas en su evaluación de las traducciones. BLEU es particularmente sensible al orden de las palabras y trata todos los *tokens* con igual importancia, incluyendo elementos como signos de puntuación y capitalización. Esto significa que diferencias aparentemente menores pueden afectar significativamente la puntuación, aunque no alteren sustancialmente el significado de la traducción. Esta característica es especialmente relevante en el contexto de LENSEGUA, donde pueden existir múltiples formas válidas de expresar el mismo concepto.

La aplicación del estándar LoRA, con un rango de 8 y un *alpha* de 16, fue efectiva para la interpretación de frases que aplicaran la gramática de LENSEGUA. El rango de 8 resultó apropiado porque permite que el modelo se enfoque específicamente en la tarea de traducción sin necesidad de aprender conceptos adicionales. El *alpha* de 16 proporcionó la escalabilidad necesaria para manejar los pesos de manera efectiva, permitiendo que el modelo identifique y utilice los *tokens* apropiados sin cambios drásticos en presencia de *tokens* únicos o diferentes.

Un cambio significativo fue optar por realizar el proceso de *fine-tuning* exclusivamente de manera supervisada. En las versiones anteriores, el modelo tenía la flexibilidad de elegir entre entrenamiento supervisado y no supervisado, lo cual resultaba problemático porque en el entrenamiento no supervisado, el modelo carecía de retroalimentación directa sobre las interpretaciones correctas esperadas. Al forzar el enfoque supervisado, se aseguró que cada ejemplo de entrenamiento tuviera una referencia clara, permitiendo que el modelo aprendiera directamente de ejemplos correctos y mejorando su capacidad de generar traducciones precisas.

El análisis LIME del modelo LLaMa (versión 1) reveló una distribución notablemente uniforme de la importancia entre las palabras, con valores de influencia relativamente bajos y similares entre sí. Esta uniformidad sugería que el modelo inicial trataba todas las palabras con similar relevancia,

sin distinguir efectivamente elementos como marcadores temporales o modificadores en las oraciones. Este comportamiento podría atribuirse a la forma en que los logits y la atención se distribuyen en el sistema de transformadores, especialmente en un modelo *fine-tuneado* con LoRA. Aunque los valores de atención y los logits eran bajos, estos datos siguen siendo útiles para analizar cómo el modelo procesa el contexto y asigna importancia a las palabras. Sin embargo, en esta versión inicial, el modelo no lograba identificar con precisión las características lingüísticas clave necesarias para generar traducciones más precisas.

La entrevista con la experta en LENSEGUA, quien otorgó una calificación de 9 sobre 10 a las traducciones, reveló que uno de los problemas más notables del primer modelo era su incapacidad para diferenciar correctamente entre oraciones en tiempo presente y pasado, así como para identificar cuándo una oración era una pregunta. Estas deficiencias resultaban en traducciones imprecisas que afectaban negativamente las métricas BLEU y evidenciaban una comprensión incompleta de las estructuras temporales y sintácticas fundamentales en LENSEGUA. Dichos errores se explican, en gran medida, por el tamaño limitado del *dataset* original. Al contar con un conjunto de datos reducido, el modelo no tuvo acceso a suficientes ejemplos que le permitieran aprender las particularidades del lenguaje, lo que contribuyó a generar interpretaciones incorrectas.

Esta situación llevó a la decisión de utilizar el *dataset* desarrollado específicamente para el módulo de GPT-3.5-Turbo en un nuevo proceso de *fine-tuning* del modelo LLaMa. Este *dataset*, siendo ocho veces más grande y diseñado específicamente para traducción a LENSEGUA, resultó en una mejora significativa. La métrica BLEU se duplicó al comparar las versiones 1 y 2 del modelo LLaMa, evidenciando cómo un *dataset* más comprehensivo y específico puede mejorar significativamente el rendimiento del modelo tras diferentes procesos de *fine-tuning*.

Para evaluar aún más el impacto del nuevo *dataset*, se realizó un análisis comparativo entre LLaMa (versión 2) y GPT-3.5-Turbo, ambos entrenados con los mismos datos. Este análisis reveló diferencias notables en cómo cada modelo procesa y prioriza la información. GPT-3.5-Turbo logró una distancia de Levenshtein significativamente menor (3.375 comparado con 6.185 de LLaMa), sugiriendo una mejor capacidad para generar traducciones más precisas.

Además, el análisis LIME de ambos modelos reveló una diferencia en el procesamiento lingüístico: mientras que LLaMa mantuvo una distribución uniforme de valores de influencia (generalmente entre 0.00 y 0.06), GPT-3.5-Turbo mostró una clara diferenciación en la importancia asignada a diferentes elementos lingüísticos. Por ejemplo, en la frase "pasado yo ir no", GPT-3.5-Turbo asignó una alta relevancia a "no" (0.44) y "pasado" (0.24), reconociendo la importancia de los marcadores temporales y la negación. En contraste, LLaMa asignó valores mucho más bajos y uniformes (0.00 para "pasado" y 0.02 para "no"), sugiriendo una menor capacidad para identificar la importancia relativa de estos elementos gramaticales.

Esta diferencia en el procesamiento de elementos temporales y modificadores es relevante en el contexto de LENSEGUA, donde la temporalidad y los modificadores son necesarios para la correcta interpretación del mensaje. La capacidad de GPT-3.5-Turbo para asignar mayor relevancia a estos elementos explica en parte su mejor desempeño en las métricas de distancia de Levenshtein, ya que puede generar traducciones que capturan mejor estos aspectos del lenguaje de señas.

A pesar de estas diferencias, LLaMa representa una alternativa viable, especialmente en situaciones donde los recursos computacionales son limitados. Los resultados sugieren que futuras mejoras podrían enfocarse en desarrollar mecanismos que permitan al modelo identificar y priorizar mejor los elementos lingüísticos, particularmente los marcadores temporales y modificadores contextuales que son necesarios en la lengua de señas.

7.4. Infraestructura de red

Durante el desarrollo del proyecto, se buscó crear una infraestructura robusta y segura para la gestión de datos y la implementación de modelos de inteligencia artificial. En esta sección, se presentan los resultados clave y los aprendizajes derivados de las distintas fases del proyecto, enfocándonos en la eficiencia, seguridad y rendimiento del servidor, así como en la accesibilidad y la virtualización.

Durante la fase de implementación inicial, se configuraron tres máquinas virtuales, cada una diseñada para ejecutar modelos de inteligencia artificial y pruebas de carga. Este enfoque de virtualización demostró ser una solución efectiva para segmentar los entornos de desarrollo y producción, permitiendo un manejo más eficiente de los recursos computacionales. La utilización de herramientas como NGINX y Gunicorn en combinación con Flask para el backend resultó en una arquitectura capaz de soportar un alto volumen de solicitudes concurrentes, lo cual fue validado mediante pruebas de carga exhaustivas.

Los resultados obtenidos en las pruebas de carga reflejaron un manejo eficiente de las solicitudes hasta un nivel de 400-500 usuarios concurrentes sin mayores problemas de rendimiento. Estas pruebas, ejecutadas a través de un entorno controlado, permitieron observar métricas clave como el tiempo de respuesta, la utilización de CPU y la carga del sistema. El uso de SQLAlchemy para la gestión de bases de datos optimizó el acceso y manejo de datos, asegurando una interacción fluida entre las APIs y el servidor. Las métricas de rendimiento mostraron un incremento progresivo en el uso de recursos bajo cargas más altas, lo que resalta la necesidad de futuras optimizaciones para escenarios de carga extrema.

La prueba end-to-end (E2E) fue otra parte integral del proyecto, garantizando que el flujo completo de la aplicación funcionara como se esperaba. La prueba abarcó desde el registro de usuarios y la autenticación, hasta la interacción con funcionalidades avanzadas como el envío de videos y la gestión de traducciones. Los resultados indicaron un éxito del 100 %, demostrando que todos los módulos interactuaban correctamente y que la integridad de la aplicación estaba asegurada. Este éxito se tradujo en una validación de la funcionalidad y operatividad del sistema en condiciones reales de uso.

En cuanto a la seguridad, se realizaron múltiples auditorías usando Lynis, una herramienta reconocida en el análisis de vulnerabilidades y configuración de sistemas Linux. Los resultados iniciales arrojaron un Hardening Index de 58, que fue mejorado a 62 tras realizar ajustes y optimizaciones en la configuración del servidor. Este índice demuestra que el servidor es seguro, aunque siempre hay margen para futuras mejoras. Es importante destacar que los resultados obtenidos fueron superiores a la meta establecida de un índice de 4.0 en la escala de CVE, lo que subraya el éxito en la implementación de medidas de seguridad sólidas.

El proyecto enfrentó desafíos notables, especialmente en la obtención de accesos y permisos necesarios para la integración de la infraestructura con la red universitaria. La colaboración con el equipo de TI y la gestión de permisos resultaron ser procesos complejos y burocráticos, que demandaron una planificación y coordinación significativa. Estos obstáculos, sin embargo, resaltaron la importancia de prever estas gestiones con anticipación para futuros proyectos.

En resumen, el desarrollo del servidor y su infraestructura demostró ser un éxito al cumplir con los objetivos de eficiencia, seguridad y accesibilidad. Las pruebas realizadas evidenciaron que el sistema puede soportar un alto volumen de carga y que sus componentes interactúan de manera efectiva.

7.5. Diseño y desarrollo móvil

Durante la fase de investigación de mercado, se analizaron aplicaciones con funcionalidades similares a las que se querían implementar en “Señas Chapinas”. No se encontró ninguna aplicación que estuviera enfocada en las necesidades y la cultura guatemalteca ni que utilizara LENSEGUA, lo que convierte a “Señas Chapinas” en una propuesta pionera en este ámbito. Esta investigación también permitió identificar funcionalidades clave a partir de los comentarios de usuarios, tales como la grabación de videos con indicadores de posición, el uso de tutoriales, la posibilidad de agregar a favoritos y la calificación de traducciones. Estas funcionalidades fueron adaptadas e implementadas en la aplicación, ajustándose al contexto cultural y comunicativo de los usuarios guatemaltecos. Además, se destacaron elementos de diseño relevantes, como la simplicidad de la interfaz, una paleta de colores atractiva y botones descriptivos sin exceso de texto. Estas características fueron aplicadas cuidadosamente en la fase de diseño y se pueden observar en la versión final de la aplicación.

Paralelamente, se llevó a cabo una investigación exhaustiva sobre la situación actual de las personas sordas en Guatemala. Se descubrió que representan solo el 3 % de la población mayor de 4 años, lo que evidencia su condición de minoría. LENSEGUA fue reconocida oficialmente hace poco tiempo y su uso es más común en la capital. Sin embargo, actualmente se está llevando a cabo un proceso de estandarización y difusión de la lengua de señas en todo el país, lo que subraya la necesidad de herramientas como “Señas Chapinas” para apoyar este esfuerzo. La aplicación facilita la expansión del conocimiento y el uso de LENSEGUA a nivel nacional, a través del desarrollo de un diccionario de señas y retos diarios, estrategias clave para promover su difusión en todas las regiones.

Las encuestas realizadas a personas oyentes revelaron que el 70 % desconocía la existencia de LENSEGUA, y solo consideraban su aprendizaje relevante en caso de tener contacto directo con personas sordas. Esto pone en evidencia la necesidad de “Señas Chapinas” no solo como una herramienta de comunicación, sino también como un vehículo educativo para incrementar la visibilidad de LENSEGUA en Guatemala.

Asimismo, se descubrió una alta tasa de analfabetismo y desempleo en la comunidad sorda, reflejando las limitadas oportunidades a las que se enfrentan. Entrevistas a personas sordas y a quienes interactúan frecuentemente con ellas confirmaron las dificultades diarias en áreas como la salud, la educación y la representación legal. Las funcionalidades de “Señas Chapinas” están diseñadas específicamente para abordar estas barreras. Por ejemplo, la integración de un módulo de traducción que acepta vocabulario para situaciones médicas, como emergencias, se desarrolló en respuesta a necesidades identificadas en estas entrevistas, demostrando que la aplicación no solo facilita la comunicación sino que también ofrece soluciones concretas a problemas cotidianos.

Con una comprensión clara de las necesidades y del contexto, se inició el diseño de la aplicación. Se definieron los objetivos y funcionalidades clave a través de la creación de diagramas de afinidad, mapas de empatía y la identificación de escenarios de uso. Esto permitió desarrollar un prototipo que integraba las necesidades específicas identificadas en la fase de investigación. La colaboración con expertos y la retroalimentación constante permitieron iterar y mejorar los prototipos hasta obtener un diseño final intuitivo y sencillo.

Las pruebas de usabilidad fueron exitosas, con más del 75 % de éxito en la realización de cada uno de los flujos de la aplicación por parte de los usuarios. Durante la EXPO UVG y en reuniones con directivos de En-Señas, se confirmó que el diseño del logo, la elección de colores y tipografía lograban representar la identidad guatemalteca así como a la comunidad sorda. Este enfoque en la usabilidad y el diseño confirma la sinergia entre la estética y la funcionalidad de la aplicación, lo que permitió un producto final que satisface tanto las expectativas de los usuarios como los estándares de diseño.

El proceso no estuvo exento de desafíos, especialmente en lo que respecta a las expectativas y la coherencia funcional. Alcanzar un prototipo que cumpliera con las expectativas de todos los

usuarios finales sin comprometer las funcionalidades clave fue complejo debido a la diversidad de opiniones de los colaboradores. Se buscó un equilibrio cuidadoso entre las sugerencias para mantener la dirección del proyecto clara. Asimismo, hubo dificultades con el contenido visual de la aplicación. Originalmente, se utilizaron imágenes de un libro de señas proporcionado por colaboradores, pero estas tuvieron que ser reemplazadas por ilustraciones hechas a mano debido a cuestiones de derechos de autor. A pesar de estos retos, las adaptaciones logradas fortalecieron la identidad visual del proyecto.

Con un diseño validado, la fase de desarrollo móvil avanzó de manera fluida, siguiendo un enfoque estructurado y sistemático. Se adoptaron estándares arquitectónicos para Android, específicamente utilizando la arquitectura MVVM (Model-View-ViewModel), que facilitó la separación de lógica de negocio y la interfaz de usuario, garantizando una mayor mantenibilidad y escalabilidad del proyecto. Se hizo un uso extensivo de componentes reutilizables definidos durante la etapa de prototipado, lo que permitió mantener la consistencia visual y funcional a lo largo de la aplicación.

La experiencia de usuario se optimizó mediante la implementación de flujos alternativos, como el uso de *deeplinks* para simplificar tareas complejas, como la recuperación y cambio de contraseñas. Además, se integraron herramientas de seguridad, incluyendo el cifrado de datos sensibles y un manejo seguro de la información del usuario, siguiendo las mejores prácticas recomendadas para aplicaciones móviles. Esto no solo mejoró la experiencia del usuario, sino que también aumentó la confianza en la aplicación, asegurando la protección de sus datos.

El proceso de desarrollo se gestionó mediante la metodología Kanban, lo que facilitó una organización clara y eficiente de las tareas. Las funcionalidades se dividieron en entregas incrementales, lo que permitió revisiones y ajustes constantes. Esto dio lugar a un desarrollo ágil, flexible y centrado en los objetivos, garantizando que cada fase del proyecto se completara a tiempo y con altos estándares de calidad.

Un elemento no planificado, pero que aportó un valor significativo, fue la publicación de la aplicación en tiendas de Google. A pesar de enfrentar retos administrativos y legales, como la integración de términos y condiciones y el diseño de pantallas de promoción, este proceso fue un hito importante. Además, llevó a la creación de una página web para respaldar la aplicación, lo que ayudó a consolidar la presencia digital del proyecto y a generar mayor confianza entre los usuarios.

La aprobación de Google fue un momento crucial, ya que permitió llevar a cabo una prueba cerrada con un grupo selecto de colaboradores de En-Señas. Esta fase de pruebas fue esencial para validar la funcionalidad de la aplicación en condiciones reales, permitiendo recibir retroalimentación directa y realizar ajustes finales antes del lanzamiento oficial. La aplicación no solo cumplió con las expectativas del público objetivo, sino que también mostró un alto nivel de rendimiento y estabilidad, confirmando la efectividad del enfoque de desarrollo adoptado.

Conclusiones

- El objetivo principal del proyecto, que consistía en diseñar y desarrollar la aplicación *Señas Chapinas*, fue alcanzado con éxito. La aplicación para dispositivos Android ha demostrado ser capaz de traducir la lengua de señas guatemalteca (LENSEGUA) a texto gramaticalmente correcto en español, utilizando modelos avanzados de visión por computadora y procesamiento de lenguaje natural. Además, se implementó una infraestructura de red segura y eficiente, lo que garantiza la correcta operación de la aplicación y su capacidad para ofrecer traducciones precisas.
- A través de un proceso iterativo de recolección de datos, modelado y evaluación, se logró un modelo final que presenta una precisión de 0.8802, una sensibilidad de 0.9685 y un F1 de 0.9706, lo cual indica un desempeño aceptable en la clasificación de gestos de lengua de señas en tiempo real. A pesar de estos logros, el modelo enfrenta limitaciones significativas, especialmente en su capacidad para reconocer correctamente múltiples palabras en un solo video. Estas limitaciones se derivan en gran medida del desbalance en el conjunto de datos y de la dificultad en la identificación del inicio y fin de las señas. La implementación exitosa del modelo en una aplicación de reconocimiento de lengua de señas de Guatemala y en un API muestra el potencial de esta tecnología para facilitar la comunicación y promover la inclusión de las personas sordas en diferentes contextos.
- El objetivo de adaptar el modelo GPT-3.5-Turbo para asimilar la gramática de LENSEGUA se cumplió satisfactoriamente. El modelo *fine-tuneado* demostró una capacidad significativamente mejorada para analizar oraciones en LENSEGUA e interpretarlas correctamente al español. Por ejemplo, mientras que las interpretaciones del modelo estándar presentaron una distancia de Levenshtein promedio de 10.065, las del modelo *fine-tuneado* lograron obtener una distancia promedio de 5.815, lo que refleja una mejora del 42.23 %. Además, los resultados de la encuesta, realizados con la participación de personas sordas e intérpretes, confirmaron el éxito de esta adaptación, ya que las interpretaciones del modelo *fine-tuneado* obtuvieron una calificación promedio de 4.74 (en una escala de 1 a 5) en comparación con las del modelo estándar, que solo alcanzaron un 2.36.
- Se desarrolló una herramienta basada en LLaMA que era capaz de comprender la gramática de LENSEGUA y generar interpretaciones coherentes en español a partir de oraciones estructuradas en dicha gramática. Esta herramienta fue inicialmente entrenada utilizando un conjunto de datos reducido, el cual, debido a su tamaño limitado, no consideraba muchas de las características y características propias de LENSEGUA. Como resultado, la primera versión del modelo presentó un rendimiento modesto, con un BLEU promedio de 0.2894. No obstante, al

ampliar el conjunto de datos con ejemplos más representativos de LENSEGUA, la segunda versión del modelo mostró una mejora significativa, logrando un valor BLEU promedio de 0.632. Este cambio reflejó una mayor capacidad del modelo para captar las sutilezas y estructuras propias de LENSEGUA, permitiendo obtener interpretaciones más precisas.

- En este proyecto se implementó un servidor robusto y eficiente que administra datos de manera segura, con configuraciones optimizadas para la ejecución de modelos de inteligencia artificial y visión por computadora en tres máquinas virtuales. Herramientas como Lynis garantizaron un nivel de seguridad elevado, alcanzando un puntaje de 6.2 en la escala CVE. Las configuraciones incluyeron apertura de puertos y scripts automatizados para el despliegue de modelos de IA y APIs, facilitando pruebas y actualizaciones. Las pruebas de carga demostraron un rendimiento sólido incluso bajo alta demanda, gracias a la integración de Gunicorn, Nginx y SQLAlchemy, asegurando estabilidad y eficiencia. Este entorno virtual controlado y seguro permite a los usuarios desarrollar y desplegar aplicaciones avanzadas con confianza.
- Se realizaron investigaciones de mercado e entrevistas con usuarios finales para identificar las necesidades específicas de la comunidad sorda en Guatemala, lo que permitió desarrollar perfiles detallados e integrar estas necesidades en flujos de usuario intuitivos. Con un enfoque estándar de UX/UI y retroalimentación constante, se diseñó una interfaz atractiva, accesible e intuitiva, adaptada a las expectativas de los usuarios. En cuanto al desarrollo técnico, se implementó exitosamente una aplicación Android con arquitectura escalable, integrando servicios externos para traducir LENSEGUA a texto gramaticalmente correcto en español, cumpliendo el objetivo de facilitar la interacción entre personas sordas y oyentes.

Recomendaciones

Tras la culminación del desarrollo de la aplicación *Señas Chapinas*, se proponen una serie de recomendaciones para continuar optimizando su funcionalidad y aumentar su alcance. Estas recomendaciones están diseñadas para mejorar la experiencia del usuario, expandir las capacidades de la aplicación y asegurar su sostenibilidad a largo plazo.

1. Para mejorar el desempeño del modelo de visión por computadora, se recomienda realizar una recolección de datos más equilibrada, teniendo en cuenta la duración de las señas. Además, asegurarse de que cada clase tenga un número similar de fotogramas en el conjunto de datos puede ayudar a mitigar el problema del desbalance y mejorar la capacidad del modelo para reconocer gestos de clases minoritarias.
2. Se sugiere ampliar el conjunto de videos con más ejemplos de cada palabra, especialmente para las que presentan una mayor confusión en el modelo. Esto puede incluir la recopilación de más videos de señas en diferentes contextos y variaciones para aumentar la diversidad y robustez del conjunto de datos.
3. Se recomienda explorar técnicas alternativas para identificar el inicio y fin de las señas en los videos, como el uso de una segunda red neuronal para detectar estos puntos clave.
4. Implementar técnicas de aumento de datos, como la rotación, la variación en la iluminación y la manipulación de la velocidad de los videos, puede ayudar a mejorar la generalización del modelo al introducir variaciones en los gestos que el modelo debe aprender a reconocer.
5. Crear videos en diferentes entornos y condiciones, como diferentes ángulos, iluminación y antecedentes, puede proporcionar un desempeño más robusto del modelo y permitir su adaptación a situaciones de la vida real.
6. Se recomienda buscar el apoyo de diversas asociaciones que trabajen con la comunidad sorda en Guatemala, además de ASEDES, para la elaboración de más frases representativas que apliquen la gramática de LENSEGUA. Dado que existen variaciones regionales de LENSEGUA, la inclusión de múltiples perspectivas enriquecería el conjunto de datos usado por los modelos de procesamiento de lenguaje natural, y garantizaría una representación más completa de la gramática y vocabulario utilizado.
7. Se recomienda ampliar el conjunto de datos para incluir frases más complejas y largas, así como párrafos completos. Esto permitiría que los modelos de procesamiento de lenguaje natural estén

expuestos a una mayor diversidad de estructuras lingüísticas, asegurando que sean capaces de interpretar frases más elaboradas.

8. Se recomienda evaluar el uso de modelos más actualizados, como GPT-4o, para realizar el proceso de *fine-tuning*. Este modelo podría ofrecer mejoras significativas en la calidad de las interpretaciones debido a su entrenamiento con un *corpus* más amplio, así como optimizaciones en su arquitectura que le permiten gestionar de manera más eficiente tareas complejas.
9. Se recomienda continuar explorando técnicas de *prompt engineering* para mejorar aún más las interpretaciones del modelo GPT-3.5-Turbo. Si bien esta técnica ya permitió desarrollar *prompts* más avanzados que incrementaron la precisión del modelo *fine-tuneado*, es posible que ajustes adicionales optimicen aún más los resultados. Además, se sugiere probar el uso exclusivo de *prompt engineering* con modelos más recientes, como GPT-4o, para evaluar si esta combinación permite generar interpretaciones correctas sin necesidad de un *fine-tuning* específico.
10. Se recomienda implementar prácticas de MLOps que permitan evaluar periódicamente el rendimiento de los modelos de procesamiento de lenguaje natural. Esto incluiría procesos automatizados para monitorear la calidad de las predicciones generadas y, en caso de que el rendimiento de un modelo disminuya, activar un *pipeline* de re-entrenamiento o ajuste del modelo para mantener su efectividad a lo largo del tiempo.
11. Se han tomado medidas significativas para asegurar y optimizar el sistema mediante configuraciones avanzadas de NGINX y Unicorn. No obstante, la habilitación de HTTP/2 para mejorar el rendimiento de la transferencia de datos y la integración de autenticación mTLS para comunicaciones internas son pasos adicionales que pueden aumentar el nivel de seguridad y rendimiento. Además, la implementación de pruebas de carga automatizadas con herramientas como Locust puede ayudar a evaluar la robustez del sistema frente a escenarios extremos.
12. Aunque Lynix ya ha demostrado ser una herramienta confiable para la auditoría de seguridad del sistema, se recomienda explorar configuraciones avanzadas que permitan adaptar su funcionamiento a las necesidades específicas del proyecto. Esto incluye la posibilidad de personalizar reglas para ignorar aspectos no relevantes en el entorno evaluado y priorizar áreas críticas mediante configuraciones específicas. Además, se sugiere establecer un ciclo de auditorías periódicas para identificar y mitigar nuevas vulnerabilidades, acompañado de un proceso de actualización continua de las configuraciones de seguridad. La flexibilidad de Lynix también permite ajustar el índice de fortalecimiento para optimizar su utilidad, alineándolo con los objetivos y prioridades del sistema. Según CIS Benchmarks, personalizar herramientas de auditoría es una práctica estándar que asegura la relevancia y precisión de las evaluaciones, permitiendo consolidar medidas robustas y prácticas de seguridad eficientes para entornos en evolución.
13. Actualmente, el proyecto utiliza un entorno virtualizado mediante contenedores para aislar y gestionar cada módulo o modelo de inteligencia artificial. Sin embargo, se pueden realizar mejoras adicionales, como la configuración avanzada de redes virtuales para facilitar la comunicación segura entre contenedores y el uso de herramientas como Terraform para la gestión de infraestructura como código, permitiendo una replicación más eficiente del entorno.
14. El proyecto integra un sistema de mapeo objeto-relacional (ORM) con SQLAlchemy, permitiendo una interacción estructurada y eficiente con la base de datos. No obstante, sería valioso explorar configuraciones avanzadas, como la implementación de estrategias de particionamiento horizontal en bases de datos para manejar grandes volúmenes de datos, y el uso de migraciones automatizadas con Alembic para garantizar la coherencia de esquemas durante actualizaciones.
15. Actualmente, el sistema cuenta con un monitoreo continuo utilizando herramientas como NGINX Amplify. A pesar de ello, podrían implementarse mejoras como la integración con Prometheus y Grafana para un análisis más profundo de métricas personalizadas y alertas

basadas en patrones de uso. También es recomendable configurar un sistema de balanceo de carga dinámico que permita escalar en función de las métricas recolectadas.

16. Para incrementar la utilidad de la aplicación, es esencial expandir el número de palabras disponibles en el diccionario, lo cual permitirá a los usuarios acceder a un repertorio más amplio para la traducción y el aprendizaje. Esto también mejoraría la funcionalidad del reto diario, proporcionando una mayor variedad de términos y enriqueciendo la experiencia de los usuarios al interactuar constantemente con nuevas palabras.
17. Se recomienda contratar a un diseñador gráfico que desarrolle las imágenes del diccionario, asegurando que sigan la misma línea estética que el resto de la aplicación. Esto no solo mejorará la funcionalidad del diccionario, sino que también elevará la experiencia visual de los usuarios, presentando imágenes claras, atractivas y alineadas con la identidad visual de la aplicación.
18. La incorporación de opciones de inicio de sesión a través de plataformas populares como Google o Facebook facilitará el acceso de los usuarios, simplificando el proceso de registro y reduciendo las barreras para nuevos usuarios. Esta funcionalidad no solo mejorará la usabilidad de la aplicación, sino que también podría incrementar la tasa de adopción de la aplicación.
19. Para hacer la aplicación más completa, se recomienda implementar la traducción inversa, permitiendo a los usuarios traducir texto de español a LENSEGUA. Esta funcionalidad expandiría significativamente las posibilidades de la aplicación, facilitando que las personas oyentes aprendan y utilicen LENSEGUA de manera más práctica y efectiva.
20. Para aumentar el alcance de la aplicación, sería recomendable desarrollar una versión para dispositivos iOS. Esta ampliación permitiría a usuarios de la plataforma de Apple beneficiarse de las funcionalidades de la aplicación, logrando un mayor impacto y accesibilidad para personas sordas y oyentes en un rango más amplio de dispositivos.

-
- [1] *¿Qué marcas de celulares prefieren en Centroamérica?*, marzo 11 2019. https://m.centralamericadata.com/es/article/home/Qu_marcas_de_celulares_prefieren_en_Centroamerica, Visitado 2024.
 - [2] *Cómo crear paletas de colores para un UI fabuloso*, N.A. <https://www.espacioux.com/blog/paletas-de-colores-ui>, Visitado 2024.
 - [3] *Android ofrece oportunidades*, s.f. https://www.android.com/intl/es_es/everyone/enabling-opportunity/, Visitado 2024.
 - [4] *Diseño UX en Español*, s.f. <https://uxenespanol.com/articulo/basicos-de-ui>, Visitado 2024.
 - [5] *Información LENSEGUA*, s.f. <https://LENSEGUA.com/>, Visitado 2024.
 - [6] *Lengua de señas guatemalteca [LENSEGUA]*, s.f. <https://www.endangeredlanguages.com/lang/7110?hl=en>, Visitado 2024.
 - [7] Aakanksha, N.: *Image Processing and Data Augmentation Techniques for Computer Vision*, 2020. Recuperado el 30 de mayo de 2024, de <https://towardsdatascience.com/image-processing-techniques-for-computer-vision-11f92f511e21>.
 - [8] Adobe: *User flow diagram — what it is, why it's important, and how to create one*, noviembre 14 2022. <https://business.adobe.com/blog/basics/how-to-make-a-user-flow-diagram>, (visitado 2024).
 - [9] AI Sign: *Sign language: App Store*, enero 7 2023. <https://apps.apple.com/us/app/ai-sign-sign-language/id1663187476>, Visitado 2024.
 - [10] Altunay, N.: *The basics of image processing and OpenCV*, 2019. Recuperado el 30 de mayo de 2024, de <https://developer.ibm.com/articles/learn-the-basics-of-computer-vision-and-object-detection/#before-everything-what-is-opencv-10>.
 - [11] Amazon Web Services: *¿Qué son los modelos de lenguaje de gran tamaño? - Explicación sobre los LLM de IA - AWS*, 2023. <https://aws.amazon.com/es/what-is/large-language-model/>.

- [12] Amazon Web Services: *What is GPT AI? - Generative Pre-Trained Transformers Explained - AWS*, 2024. <https://aws.amazon.com/what-is/gpt/>
- [13] Amazon Web Services: *What is Natural Language Processing? - NLP - AWS*, 2024. <https://aws.amazon.com/what-is/nlp/>
- [14] Amazon Web Services: *¿Qué es una API? - Explicación de interfaz de programación de aplicaciones*, s.f. <https://aws.amazon.com/es/what-is/api/>, visitado el 2024.
- [15] Amazon Web Services: *¿Qué es una VPN? - Explicación de las redes privadas virtuales*, s.f. <https://aws.amazon.com/es/what-is/vpn/>, visitado el 2024.
- [16] Anthony, B.: *Optimal size and spacing for mobile buttons*, febrero 28 2019. <https://uxmovement.com/mobile/optimal-size-and-spacing-for-mobile-buttons/>, Visitado 2024.
- [17] Aroche, K.: *El lenguaje de señas guatemalteco*, septiembre 22 2022. <https://aprende.guatemala.com/cultura-guatemalteca/general/el-lenguaje-de-senas-guatemalteco/>, Visitado 2024.
- [18] Artigas, S. G.: *Exponer problemas como habilidad esencial de UX*, junio 28 2017. <https://torresburriel.com/weblog/exponer-problemas-una-habilidad-esencial-ux/>, (visitado 2024).
- [19] Asana: *¿Qué es Kanban?*, 2024. <https://asana.com/es/resources/what-is-kanban>, Accedido: 2024-09-12.
- [20] Bescós, A.: *La Importancia Vital del Diseño Responsivo en Aplicaciones Móviles*, febrero 10 2024. <https://www.overant.com/blog/la-importancia-vital-del-diseno-responsivo-en-aplicaciones-moviles/>, Visitado 2024.
- [21] Better Health Australia: *Hearing loss - how it affects people*, 2024. <https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/hearing-loss-how-it-affects-people#:~:text=Hearing%20loss%20can%20affect%20a>
- [22] Bhadoria, D. S.: *Android MVVM, how to use MVVM in android?*, enero 12 2013. <https://medium.com/@dheerubhadoria/android-mvvm-how-to-use-mvvm-in-android-example-7dec84a1fb73>, Visitado 2024.
- [23] Brentwood Hearing Center: *How Does Hearing Loss Affect Communication?*, 2020. <https://brentwoodhearingcenter.com/how-does-hearing-loss-affect-communication/>
- [24] Bustos, J. L.: *Tamaños de los objetos táctiles en una aplicación móvil*, marzo 25 2022. <https://keepcoding.io/blog/tamanos-objetos-tactiles-aplicacion-movil/>, Visitado 2024.
- [25] Center for Internet Security: *CIS Benchmarks: Configuration Guidelines for Security*. Informe técnico, Center for Internet Security, 2024. <https://www.cisecurity.org/cis-benchmarks>, visitado el 2024, Acceso en línea.
- [26] Chacón, N.: *UX: guía completa sobre la experiencia de usuario*, enero 17 2024. <https://blog.hubspot.es/website/experiencia-de-usuario-ux#que-es>, Visitado 2024.
- [27] Clark, David D.: *The design philosophy of the DARPA internet protocols*. Computer Communication Review, 18(4):106–114, 1988.
- [28] Congreso de Guatemala: *LENSEGUA, ley que fomenta la inclusión social*, 2022. https://www.congreso.gob.gt/noticias_congreso/9131/2022/4#gsc.tab=0.

- [29] Congreso de la República de Guatemala: *LENSEGUA, Ley que Fomenta la Inclusión Social*, 2022. https://www.congreso.gob.gt/noticias_congreso/9131/2022/4#gsc.tab=0, visitado el 2024.
- [30] Congreso de la República de Guatemala: *Día para conmemorar la inclusión social para las personas con discapacidad auditiva*, s.f. https://www.congreso.gob.gt/noticias_congreso/7190/2021/4#gsc.tab=0, Visitado 2024.
- [31] Consejo Nacional para la Atención de las Personas con Discapacidad: *Informe sobre la población sorda en Guatemala*, 2021. <https://ejemplo.com/informe-conadi>, visitado el 2024.
- [32] Corporation, NVIDIA: *CUDA C++ Programming Guide (12.6)*, 2024. https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.
- [33] Cutten, D.: *How can hearing impairment affect language development?*, 2018. <https://www.hiddenhearing.co.uk/hearing-blog/hearing-aids/how-can-a-hearing-impairment-affect-language-development>.
- [34] Data Kwery: *The Fundamentals of Natural Language Processing: A Beginner's Guide - DataKwery*, 2023. <https://www.datakwery.com/post/fundamentals-of-nlp-guide/>.
- [35] De La Grana, A.: *Jetpack Compose vs XML: Android UI development compared*, agosto 28 2024. <https://qubika.com/blog/jetpack-compose-vs-xml-android-u/>, (visitado 2024).
- [36] DeepLearningAI: *A Basic Introduction to Logistic Regression for Machine Learning*, 2022. https://www.deeplearning.ai/the-batch/logistic-regression-follow-the-curve/?_gl=1.
- [37] DeepLearningAI: *Natural Language Processing (NLP) - A Complete Guide*, 2023. <https://www.deeplearning.ai/resources/natural-language-processing/>.
- [38] Design: *Best fonts for mobile app design*, junio 4 2024. <https://www.justinmind.com/ui-design/best-font-mobile-app>, (visitado 2024).
- [39] Dieber, Jürgen y Sabrina Kirrane: *Why model why? Assessing the strengths and limitations of LIME*. arXiv preprint arXiv:2012.00093, 2020.
- [40] DigitalOcean: *System Monitoring Tools: A Comprehensive Guide*, 2021. <https://www.digitalocean.com/monitoring-tools/>, visitado el 2024.
- [41] Doshi-Velez, Finale y Been Kim: *Towards a rigorous science of interpretable machine learning*. arXiv preprint arXiv:1702.08608, 2017.
- [42] En-Señas: *Cultura sorda*, s.f. Visitado 2024.
- [43] En-Señas: *Gramática de las personas sordas*, s.f. Visitado 2024.
- [44] En-Señas: *Técnicas y reglas para una buena interpretación*, s.f. Visitado 2024.
- [45] Fan, Yaxin, Feng Jiang, Peifeng Li y Haizhou Li: *Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning*. En *CCF International Conference on Natural Language Processing and Chinese Computing*, páginas 69–80. Springer, 2023.
- [46] Ferrer, J.: *How Transformers Work: A Detailed Exploration of Transformer Architecture*, 2024. <https://www.datacamp.com/tutorial/how-transformers-work>.
- [47] Foggetti, F. y F. Foggetti: *Meet the Hand Talk Sign Language Translator App*, abril 11 2023. <https://www.handtalk.me/en/blog/meet-the-hand-talk-sign-language-translator-app/>, Visitado 2024.

- [48] Forsyth, D. y J. Ponce: *Computer vision: A modern approach*, 2012. Recuperado el 30 de mayo de 2024, de <https://archive.org/details/computervisionmo0000fors/page/696/mode/2up>.
- [49] GeeksforGeeks: *History of Linux*, 2024. <https://www.geeksforgeeks.org/linux-history/>, visitado el 2024.
- [50] Gobierno de Guatemala, SEGEPLAN y Comité Pro Ciegos y Sordos de Guatemala: *SEGEPLAN y el Comité Pro Ciegos y Sordos de Guatemala comprometidos con la inclusión social de las personas con discapacidad*, octubre 19 2022. <https://prensa.gob.gt/comunicado/segeplan-y-el-comite-pro-ciegos-y-sordos-de-guatemala-comprometidos-con-la-inclusion>, Visitado 2024.
- [51] Goldsmith, J.: *The Value of Computer Vision in Healthcare*, 2021. Recuperado el 30 de mayo de 2024, de <https://www.himss.org/resources/value-computer-vision-healthcare>.
- [52] Goodfellow, I., Y. Bengio y A. Courville: *Deep learning*. MIT Press, 2016. Recuperado el 31 de mayo de 2024, de <https://link.springer.com/article/10.1007/s10710-017-9314-z>.
- [53] Google: *On-device machine learning for everyone*, n.d. Recuperado el 21 de abril de 2024, de <https://developers.google.com/mediapipe>.
- [54] Google: *On-device machine learning for everyone*, n.d. Recuperado el 21 de abril de 2024, de <https://developers.google.com/mediapipe>.
- [55] Graphique, La Boîte: *Adonide*, s.f. <https://www.myfonts.com/es/collections/adonide-font-la-boite-graphique>, Accedido: 2024-09-12.
- [56] Guatemala, Congreso de la República de: *Decreto número 3-2020: Ley que reconoce y aprueba la Lengua de Señas de Guatemala, LENSEGUA*, febrero 18 2020. <https://conadi.gob.gt/web/wp-content/uploads/2021/04/Decreto-3-2020-editable.docx>, Visitado 2024.
- [57] Gupta, Y.: *Chat GPT and GPT 3 Detailed Architecture Study-Deep NLP Horse*, 03 2023. <https://medium.com/nerd-for-tech/gpt3-and-chat-gpt-detailed-architecture-study-deep-nlp-horse-db3af9de8a5d>.
- [58] Hamond, M.: *Customer Journey Map: qué es, cómo crearlo y ejemplos (con plantilla)*, febrero 8 2024. <https://blog.hubspot.es/service/customer-journey-map>, (visitado 2024).
- [59] Hardesty, L.: *Explained: Neural networks*, 2017. <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>.
- [60] Hostinger: *Sintaxis Crontab: Tutorial Completo*. Informe técnico, Hostinger, 2024. <https://www.hostinger.es/tutoriales/sintaxis-crontab>, visitado el 2024, Acceso en línea.
- [61] HowToGeek: *How to Use APIs: A Guide for Beginners*, 2021. <https://www.howtogeek.com/123466/how-to-use-apis/>, visitado el 2024.
- [62] HowToGeek: *Implementing APIs with Databases*, s.f. <https://www.howtogeek.com/implementing-apis-databases/>, visitado el 2024.
- [63] HowToGeek: *Sending and Receiving Data with APIs*, s.f. <https://www.howtogeek.com/sending-receiving-data-api/>, visitado el 2024.
- [64] IBM: *What is a neural network?*, n.d. Recuperado el 31 de mayo de 2024, de <https://www.ibm.com/topics/neural-networks>.
- [65] IBM: *¿Qué es una matriz de confusión?*, n.d. Recuperado el 20 de octubre de 2024, de <https://www.ibm.com/mx-es/topics/confusion-matrix>.

- [66] IBM: *Data Architecture Overview*, s.f. <https://www.ibm.com/cloud/learn/data-architecture>, visitado el 2024.
- [67] IBM: *¿Qué es una base de datos relacional?*, s.f. <https://www.ibm.com/mx-es/topics/relational-databases>, visitado el 2024.
- [68] Jawade, B.: *Understanding LORA — low rank adaptation for finetuning large models*. Medium, December 2023. <https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6>.
- [69] Journey, ML: *What is Recall in Machine Learning?*, 2024. Recuperado el 20 de octubre de 2024, de <https://mljourney.com/what-is-recall-in-machine-learning/>.
- [70] Karthik, A. Ilya, T.: *Improving Language Understanding by Generative Pre-Training*, 2018. <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf>.
- [71] Koehrsen, W.: *Neural Network Embeddings Explained*, 2018. <https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526?gi=5bc1084d4154>.
- [72] Kucsván, Z.: *Comparing two- and three-view computer vision*, 2019. Recuperado el 23 de abril de 2024, de https://www.researchgate.net/publication/335322802_Comparing_two_and_three-view_computer_vision.
- [73] Kundu, R.: *F1 Score in Machine Learning: Intro and Calculation*, 2022. Recuperado el 20 de octubre de 2024, de <https://www.v7labs.com/blog/f1-score-guide>.
- [74] La Asociación Americana del Habla, Lenguaje y Audición: *Tipo, grado y configuración de la pérdida de audición SERIE INFORMATIVA DE AUDIOLOGÍA*, 2020. <https://www.asha.org/siteassets/ais/ais-type-degree-and-configuration-of-hearing-loss-spanish.pdf>.
- [75] Lenguaje de señas IA: *Aplicaciones en Google Play*, s.f. <https://play.google.com/store/apps/details?id=ai.terp.www.twa>, Visitado 2024.
- [76] León, R. I. de, L. Reyes de Ramos, J. R. Bámaca y E. Méndez: *Lenguaje de señas guatemalteco (LENSEGUA)*, julio 2021. <https://es.scribd.com/document/413170416/Manual-Lenguaje-de-Senas>, Visitado 2024.
- [77] León, R. R.: *Diseño de Experiencia de Usuario: etapas, actividades, técnicas y herramientas*, junio 6 2013. <https://www.nosolousabilidad.com/articulos/uxd.htm>, Visitado 2024.
- [78] LoadView: *¿Qué es la prueba de carga? — Mejores prácticas en 2024*. Informe técnico, LoadView, 2024. <https://www.loadview-testing.com/es/mas-informacion-sobre-las-pruebas-de-carga/pruebas-de-carga/>, visitado el 2024, Acceso en línea.
- [79] Martínez, B.: *Aprendamos lengua de señas para una verdadera inclusión*, 2021. <https://www.prensalibre.com/guatemala/comunitario/aprendamos-lengua-de-senas-para-una-verdadera-inclusion/>.
- [80] Maze: *Diagramas de afinidad en UX: Cómo recolectar, organizar y agrupar datos*, febrero 1 2024. <https://maze.co/blog/affinity-diagrams/>, Visitado en 2024.
- [81] McKinsey Company: *What is generative AI?*, 2024. <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai>.
- [82] Meijomil, Sergio: *Qué es un deeplink, para qué sirve y ejemplos*, February 2024. <https://www.inboundcycle.com/diccionario-marketing-online/deeplink>, Consultado el: 6 de Febrero de 2024.

- [83] Meléndez-Labrador, S.: *El lugar de la lengua de señas como lengua minoritaria en la accesibilidad comunicativa universal*, septiembre 7 2021. <https://www.redalyc.org/journal/5115/511569019004/html/>, Visitado 2024.
- [84] Michael Boelen: *Lynis Hardening Index*, s.f. <https://linux-audit.com/lynis/lynis-hardening-index/>, visitado el 2024.
- [85] Momjian, Bruce: *PostgreSQL: Introduction and Concepts*. Addison-Wesley, 2000.
- [86] Multipass: *Multipass Documentation*, s.f. <https://multipass.run/docs>, visitado el 2024.
- [87] Naciones Unidas: *Declaración Universal de los Derechos Humanos de las Naciones Unidas*, s.f. <https://www.un.org/en/about-us/universal-declaration-of-human-rights>, visitado el 2024.
- [88] National Deaf Children's Society: *Deafness causes before birth — Deafness in childhood*, 2023. <https://www.ndcs.org.uk/information-and-support/childhood-deafness/causes-of-deafness/>
- [89] National Deaf Children's Society: *What is sign language? — Communication*, 2023. <https://www.ndcs.org.uk/information-and-support/language-and-communication/sign-language/what-is-sign-language/>.
- [90] Negus, Christopher: *Linux Bible*. John Wiley & Sons, 2020.
- [91] Negus, Christopher: *Linux Bible*. John Wiley & Sons, 2020.
- [92] NGINX, Inc.: *NGINX Amplify: Monitoriza tus servidores web NGINX*, 2024. <https://www.xn--linuxenespaol-skb.com/herramientas/nginx-amplify-monitoriza-tus-servidores-web-nginx/>, visitado el 2024-12-28, Acceso en línea.
- [93] OpenAI: *Fine-tuning*, 2024. <https://platform.openai.com/docs/guides/fine-tuning>.
- [94] OpenCV: *OpenCV: Introduction*, n.d.
- [95] Ortega, K.: *¿Qué es el diseño UI y cuáles son sus componentes?*, s.f. <https://worldcampus.saintleo.edu/noticias/cuales-son-los-componentes-de-la-interfaz-de-usuario-elementos-del-diseno-ui>, Visitado 2024.
- [96] Papineni, K., Roukos S. Ward T. Zhu W.: *BLEU: a Method for Automatic Evaluation of Machine Translation*. Association for Computational Linguistics, 2002. <https://doi.org/10.3115/1073083.1073135>.
- [97] Parada, D.: *La lengua de señas: puente entre dos mundos*, septiembre 24 2022. <https://dca.gob.gt/noticias-guatemala-diario-centro-america/testimonial/la-lengua-de-senas-puente-entre-dos-mundos/>, Visitado 2024.
- [98] Parasoft: *¿Qué son las pruebas de un extremo a otro? Una descripción general completa*. Informe técnico, Parasoft, 2024. <https://es.parasoft.com/learning-center/end-to-end-testing-guide/>, visitado el 2024, Acceso en línea.
- [99] Parks, E. Parks, J.: *Encuesta sociolingüística de la comunidad sorda en Guatemala*. . 2015. https://www.academia.edu/download/60283266/Encuesta_sociolingüística_de_la_comunidad_sorda_en_Guatemala20190813-53108-c6xgg6.pdf
- [100] Phillips, B. y C. Stewart: *Android Programming: The Big Nerd Ranch Guide*. Big Nerd Ranch, segunda edición edición, 2022.

- [101] Pikaso, Team: *Diseño de Apps: 7 Consejos Para Diseñar Botones UX/UI*, noviembre 22 2022. <https://pickaso.com/2021/7-consejos-disenar-botones-ux-ui>, Visitado 2024.
- [102] Project, J.: *Deaf in Guatemala*, s.f. https://joshuaproject.net/people_groups/19007/GT, Visitado 2024.
- [103] Python Software Foundation: *Pruebas automatizadas con scripts de Python*, 2024. <https://peerdh.com/es/blogs/programming-insights/automated-testing-with-python-scripts-3>, visitado el 2024-12-28, Acceso en línea.
- [104] Ramos, J.: *¿Qué es MVC, MVP y MVVM?*, s.f. <https://programacionymas.com/blog/android-mvc-mvp-mvvm>, Visitado 2024.
- [105] Red Hat: *What is a Linux server?*, s.f. <https://www.redhat.com/en/topics/linux/linux-server>, visitado el 2024.
- [106] Rees, D.: *What is wireframing — Experience UX*, febrero 27 2024. <https://www.experienceux.co.uk/faqs/what-is-wireframing/>, (visitado 2024).
- [107] Ribeiro, Marco Tulio, Sameer Singh y Carlos Guestrin: *”Why should i trust you?”. Explaining the predictions of any classifier*. En *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, páginas 1135–1144, 2016.
- [108] Rodríguez S., M. I. y R. del P. Velásquez G.: *Historia y Gramática de la Lengua de Señas*, s.f. <https://revistas.pedagogica.edu.co/index.php/PYS/article/view/6242/5695>, Visitado 2024.
- [109] Rokon, O.: *RNN vs. LSTM vs. Transformers: Unraveling the Secrets of Sequential Data Processing*, 2023. <https://medium.com/@mroko001/rnn-vs-lstm-vs-transformers-unraveling-the-secrets-of-sequential-data-processing-c4541c4b0>
- [110] Ruíz Villa, A.: *La lengua de señas en un mundo globalizado*, 2022. https://edu.ijd.org.mx/data/files/La-lengua-de-se-as-en-un-mundo-globalizado-Alejandra-Ruiz-Villa_VBLOG_vf_3.pdf, Visitado 2024.
- [111] Santos, D.: *Qué es la técnica de los seis sombreros para pensar y cómo aplicarla*, junio 3 2024. https://blog.hubspot.es/?hubs_content=blog.hubspot.es%2Fmarketing%2Fseis-sombreros-para-pensar&hubs_content-cta=null, (visitado 2024).
- [112] Scikit-learn: *1.9. Naive Bayes — scikit-learn 0.21.3 documentation*, 2019. https://scikit-learn.org/stable/modules/naive_bayes.html.
- [113] Semi, N.: *10 steps of the UI/UX design process every expert does!*, mayo 6 2022. <https://bootcamp.uxdesign.cc/10-steps-of-the-ui-ux-design-process-every-expert-does-254e2a17ac34>, Visitado 2024.
- [114] Serva, Maurizio y Filippo Petroni: *Indo-European languages tree by Levenshtein distance*. *Europhysics letters*, 81(6):68005, 2008.
- [115] Servicio al Cliente de NordVPN: *¿Qué es OpenVPN?*, s.f. <https://support.nordvpn.com/hc/es/articles/19683394518161--Qu%C3%A9-es-OpenVPN#:~:text=OpenVPN%20es%20un%20protocolo%20VPN,%20de%20punto%20a%20punto>, visitado el 2024.
- [116] Sevilla, Cámara de Comercio de: *El papel de la tipografía en el diseño web. ¿Cómo elegir la adecuada?*, octubre 16 2023. <https://en.camaradesevilla.com/tipografia/>, Visitado 2024.
- [117] Sherstinsky, A.: *Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network*, 2018. <https://arxiv.org/abs/1808.03314>.

- [118] Shum, Y. M.: *Estadísticas de la Situación digital, Internet y redes sociales Guatemala 2023*, abril 27 2023. <https://yiminshum.com/estadisticas-digital-internet-redes-sociales-guatemala-2023/>, Visitado 2024.
- [119] Shum Xie, Y. M.: *Estadísticas de la Situación digital, Internet y redes sociales en Guatemala 2023*, abril 27 2023. <https://yiminshum.com/estadisticas-digital-internet-redes-sociales-guatemala-2023/>, visitado el 2024.
- [120] Sign language translator AI: *Aplicaciones en Google Play*, s.f. <https://play.google.com/store/apps/details?id=com.siyaroll.sltranslator>, Visitado 2024.
- [121] Sinha, S.: *From GPT-1 to GPT-4: A Look at the Evolution of Generative AI*, 2023. <https://hgs.cx/blog/from-gpt-1-to-gpt-4-a-look-at-the-evolution-of-generative-ai/>.
- [122] SLAIT: *Real-time Sign Language Translator with AI*, s.f. <https://slait.ai/>, Visitado 2024.
- [123] Taneja, A.: *Deep Dive into the Positional Encodings of the Transformer Neural Network Architecture: With Code!*, 2023. <https://www.linkedin.com/pulse/deep-dive-positional-encodings-transformer-neural-network-ajay-taneja#:~:text=Positional%20Encodings%20can%20be%20looked>.
- [124] Techopedia: *Virtualization: How it works*, 2022. <https://www.techopedia.com/definition/654/virtualization>, visitado el 2024.
- [125] Touvron, H., T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave y G. Lample: *LLaMA: Open and Efficient Foundation Language Models*. arXiv, 2023. <https://doi.org/10.48550/arxiv.2302.13971>
- [126] UNICEF: *II Encuesta Nacional de Discapacidad en Guatemala*, 2016. <https://www.unicef.org/guatemala/informes/ii-encuesta-nacional-de-discapacidad-en-guatemala>.
- [127] Vaswani, A., Vaswani, B., Shazeer, N., Niki J., Uszkoreit, J.: *Attention Is All You Need*, 2017. <https://arxiv.org/pdf/1706.03762>.
- [128] Vu, Minh N, Truc D Nguyen, NhatHai Phan, Raluca Gera y My T Thai: *Evaluating explainers via perturbation*. 2019.
- [129] Weiss, A.: *Prompt engineering: definición, ejemplos y buenas prácticas*, 2023. <https://www.ionos.mx/digitalguide/paginas-web/creacion-de-paginas-web/prompt-engineering/>.
- [130] Weller, O., M. Marone, V. Braverman, D. Lawrie y B. Van Durme: *Pretrained Models for Multilingual Federated Learning*. En *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2022.
- [131] Woll, B.: *The History of Sign Language Linguistics*. Oxford, 2013. <https://doi.org/10.1093/oxfordhb/9780199585847.013.0005>.
- [132] World Federation of the Deaf: *International Advocacy*, 2023. <https://wfdeaf.org/our-work/international-advocacy/>.
- [133] World Health Organization: *Deafness and hearing loss*, 02 2023. <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss#:~:text=A%20person%20who%20is%20not>.
- [134] Özaltun, S.: *ViewBinding vs DataBinding — Diferencias y Detalles*, Noviembre 18 2022. <https://medium.com/huawei-developers/viewbinding-vs-databinding-differences-and-detailed-8bfd4d4be290f>, (visitado 2024).

1. Encuesta para personas oyentes

1. Información básica

- Edad
 - 18-24
 - 25-34
 - 35-44
 - 45-54
 - 55+
- Género
 - Masculino
 - Femenino
 - Otro / Prefiero no decir
- Lugar de nacimiento
 - Guatemala
 - Extranjero
- Profesión
 - (Espacio para respuesta abierta)

2. Conocimiento y experiencia con la Lengua de señas

- ¿Sabes qué es Lensegua?
 - Sí
 - No
- ¿Tienes algún conocimiento de la lengua de señas?
 - Básico
 - Medio
 - Avanzado

- Ninguno
- ¿Conoces a alguien que sea sordo?
 - Sí
 - No
- ¿Cómo te comunicarías con una persona sorda?
 - Por mensajes escritos
 - Señalando lo que quiero decir
 - Usando Lensegua
 - No sabría cómo hacerlo
- ¿Cuáles consideras que son los principales desafíos que enfrentan las personas con discapacidad auditiva diariamente?
 - (Espacio para respuesta abierta)

3. **Señas Chapinas** Es una aplicación para traducción de LENSEGUA (Lengua de Señas Guatemalteco) a texto o voz.

- ¿Qué tan relevante consideras una aplicación de traducción de lengua de señas para tu vida diaria?
 - Muy relevante
 - Algo relevante
 - Poco relevante
 - Nada relevante
- ¿Cuáles serían tus principales motivaciones para usar una aplicación de traducción de lengua de señas? (Selecciona todas las que apliquen)
 - Comunicación con amigos/familiares sordos
 - Curiosidad personal
 - Requerimientos laborales
 - Actividades voluntarias
 - Otras: _____
- ¿En qué situaciones te gustaría usar la aplicación? (Selecciona todas las que apliquen)
 - Trabajo
 - Educación
 - Actividades sociales
 - Voluntariado
 - Otras: _____

4. **Aplicaciones móviles**

- ¿Qué características consideras más importantes en una aplicación móvil? (Selecciona todas las que apliquen)
 - Facilidad de uso
 - Velocidad y rendimiento
 - Diseño atractivo
 - Funciones de accesibilidad
 - Otras: _____

5. **Comentarios adicionales**

- ¿Cuáles características consideras esenciales para una aplicación de traducción de lengua de señas y por qué?
 - (Espacio para respuesta abierta)

2. Preguntas para entrevistas a personas sordas

1. Información básica

- Edad
- Lugar de nacimiento
- Profesión

2. Comunicación

- ¿Qué tipo de sordera tienes?
- ¿Cómo hablas con una persona oyente que no sabe LENSEGUA?
- ¿Utilizas tu teléfono para comunicarte con personas oyentes? ¿Cómo?

3. Señas Chapinas es una aplicación para traducción de LENSEGUA a texto o voz.

- ¿Cuándo usarías la aplicación?

4. Aplicaciones móviles

- ¿Usas mucho las aplicaciones en tu teléfono?
- ¿Cuáles son tus aplicaciones más utilizadas? ¿Por qué?
- ¿Qué hace que una aplicación sea fácil de usar para ti?
- ¿Hay algo que no te guste o te sea difícil en las aplicaciones?

5. Comentarios adicionales

- ¿Qué es lo más importante que esperarías de la aplicación Señas Chapinas?

3. Carta de solicitud de acceso a la VPN

Guatemala, Lunes 6 de Mayo del 2024

SDT-12100 Solicitud de acceso a VPN, para realización de mega proyecto “*Señas chapinas: Módulo infraestructura de red*”

Estimados responsables de la gestión de redes, Nery Antonio Alvizures Melendez, Hernandez Sican Ludwig Werner y Angel Andres Rodas Lopez

Mediante la presente, me dirijo a ustedes con el propósito de solicitar acceso para hacer uso de la VPN que se ha implementado en el rack de la cueva. La razón principal de esta solicitud es poder realizar trabajo remoto en el marco de un proyecto importante en el cual estoy involucrado, así como también para optimizar la gestión de mis recursos y completar eficientemente dicho proyecto.

Quisiera destacar que este acceso a la VPN se solicita para las siguientes personas:

- **José Miguel González y González - 20335:** Miembro del equipo encargado del módulo de infraestructura de red.
- **Carol Andree Arevalo Estrada - 20461:** Miembro del equipo encargada del módulo de aplicación móvil.
- **Luis Diego Santos Cuellar - 20226:** Miembro del equipo encargado del modulo de visión por computadora.
- **Stefano Alberto Aragoni Maldonado - 20261:** Miembro del equipo encargado del modulo de procesamiento de lenguaje natural.
- **Roberto Vallecillos Chinchilla - 20441:** Miembro del equipo encargado de del modulo de inteligencia artificial.
- **Miguel Novella Linares - Docente:** Supervisor del proyecto. Su acceso a la VPN facilita la supervisión y el apoyo continuo que requiere este proyecto.

Entendemos que, hasta el momento, los accesos están limitados a estas dos personas debido a los requerimientos específicos del proyecto. Sin embargo, también consideramos que esta solicitud marca el inicio de futuras implementaciones de VPN en los servidores de la universidad, lo cual podría abrir nuevas oportunidades para la enseñanza y el aprendizaje a través de modalidades innovadoras, como laboratorios en clase y actividades colaborativas en línea.

José Miguel González y González

Carnet: 20335

gon20335@uvg.edu.gt

+(502) 5460-2815



Ing. Douglas Barrios
Director Ciencia de la Computación

