

Universidad del Valle de Guatemala

Facultad de Ingeniería



Proyecto Julius, *Progressive Web App* para recopilar información sobre accidentes de tránsito y almacenarlos en una base de datos estructurada en base al estándar IRTAD

TRABAJO DE GRADUACIÓN EN MODALIDAD DE MEGAPROYECTO

presentado por:

Héctor Javier Carpio García

Gustavo Adolfo de León Tobías

Alberto Andrés Urizar Mancia

Luis Carlos Esturbán Rodríguez

Raúl Alejandro Monzón Solís

PARA OPTAR AL GRADO ACADÉMICO DE:

Licenciados en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala

2021

Universidad del Valle de Guatemala

Facultad de Ingeniería



Proyecto Julius, *Progressive Web App* para recopilar información sobre accidentes de tránsito y almacenarlos en una base de datos estructurada en base al estándar IRTAD

TRABAJO DE GRADUACIÓN EN MODALIDAD DE MEGAPROYECTO

presentado por:

Héctor Javier Carpio García

Gustavo Adolfo de León Tobías

Alberto Andrés Urizar Mancia

Luis Carlos Esturbán Rodríguez

Raúl Alejandro Monzón Solís

PARA OPTAR AL GRADO ACADÉMICO DE:

Licenciados en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

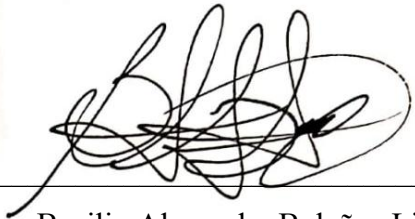
Guatemala

2021

Vo. Bo.

(f) 
Franz Josef Haidacher Avila

Tribunal Examinador:

(f) 
Bacilio Alexander Bolaños Lima

(f) 
Franz Josef Haidacher Avila

(f) 
Tomás Gálvez Peña

Fecha de aprobación: Guatemala, 13 de diciembre 2021

Índice

Índice	i
Lista de cuadros	ix
Lista de figuras	v
Resumen	x
I. Introducción	1
II. Objetivos	3
2.1 Objetivo general	3
2.2 Objetivos específicos	3
III. Justificación	4
IV: Marco teórico	9
4.1 Accidentes de tránsito	9
4.2 Estándar IRTAD	11
4.3 AppSheet	12
4.4 Progressive Web App	13
4.5 Framework	14
4.6 Dispositivos móviles	15
4.7 Frontend	16
4.8 Frameworks para frontend	16
4.8.1 Flutter	16
4.8.2 Xamarin	17
4.8.3 React Native	18
4.8.4 Angular	19
4.8.5 Vue	20
4.9 Backend	20
4.10 Bases de datos relacionales	21
4.10.1 ¿Qué es una base de datos relacional?	21
4.10.2 El modelo relacional	21
4.10.3 Ventajas de las bases de datos relacionales	22
4.11 Gestores de base de datos	22
4.11.1 Mysql	22
4.11.2 Postgresql	23
4.12 Compromiso y atomicidad	23
4.13 API	23

4.14 Servicios	24
4.15 REST	25
4.16 HTTP	26
4.16.1 Métodos HTTP	26
4.16.2 GET	26
4.16.3 POST	26
4.16.4 PUT	26
4.16.5 PATCH	26
4.16.6 DELETE	27
4.16.7 OPTIONS	27
4.17 Códigos HTTP	27
4.17.1 Respuestas informativas (100–199)	27
4.17.2 Respuestas satisfactorias (200–299)	27
4.17.3 Redirecciones (300–399)	27
4.17.4 Errores de los clientes (400–499)	28
4.17.5 Errores de los servidores (500–599)	28
4.18 Postman	28
4.19 Plataformas en la nube	28
4.19.1 Google Cloud	28
4.19.2 Amazon Web Services	29
4.20 Machine learning	29
4.20.1 Catboost	29
4.20.2 Random Forest	29
4.21 Data sintética	29
4.21.1 SMOTE	29
4.21.2 ADASYN	29
4.21.3 DBSMOTE	30
4.22 Aplicación de dispositivos inteligentes en detección de percances vehiculares	30
4.23 Leyes físicas en un accidente vehicular	31
4.24 Sonidos durante un accidente vehicular	33
V. Marco metodológico	34
5.1 Planificación	34
5.2 Investigación	34
5.3 Prototipado	34
5.4 Evaluación de tecnologías	37
Tabla 6. Comparación de aplicación nativa y PWA.	38
5.5 Diseño de bases de datos y creación de la misma	39
Figura 12. Diseño de base de datos	39
5.6 Implementación	40

5.7 Desarrollo	41
Figura 13. Inputs en pantallas 1, 2 y 3.	44
Figura 14. Inputs en pantallas 4 y 5	44
5.8 Testing	45
5.9 Desarrollo de prototipo de data sintética	46
5.10 Prototipos de implementación de algoritmos de predicción	47
5.11 Desarrollo de aplicación nativa para detección de posibles percances vehiculares	47
5.12 Prototipo de aplicación nativa para detección de posibles percances vehiculares	48
VI: Resultados	49
6.1 Sección de datos importantes	60
6.2 Sección de descripción	63
6.3 Sección de datos generales	65
6.4 Sección de tipo de vehículo	67
6.5 Sección de personas involucradas	71
6.6 Modo edición	76
6.7 Cambio de lenguaje	77
6.8 Funcionalidad en otros dispositivos	78
VII: Conclusiones	109
VIII: Recomendaciones	112
IX: Bibliografía	114
X: Anexos	122

Lista de cuadros

Tabla 1. Ventajas y desventajas de flutter.	17
Tabla 2. Ventajas y desventajas de Xamarin.	18
Tabla 3. Ventajas y desventajas de React Native.	19
Tabla 4. Ventajas y desventajas de Angular.	19
Tabla 5. Ventajas y desventajas de Vue.	20
Tabla 6. Comparación de aplicación nativa y PWA.	38
Tabla 7. Listado de endpoints	83
Tabla 8. Distribución de dataset para entrenamiento	94
Tabla 9. Tabla de lanzamientos de caída libre desde un metro de altura	106
Tabla 10. Tabla de lanzamientos de caída libre desde un metro y medio de altura.	1064
Tabla 11. Tabla de lanzamientos de caída libre desde dos metros de altura	107

Lista de figuras

Figura 1. Aumento y reducción de muertes por país.	6
Figura 2. Porcentaje por edad de muertes de peatones.	7
Figura 3. Intersección	10
Figura 4. Calle dividida	10
Figura 5. Calzada única	11
Figura 6. Diagrama de APIs.	24
Figura 7. Bosquejo uno app	35
Figura 8. Bosquejo dos app	35
Figura 9. Bosquejo tres app	36
Figura 10. Bosquejo cuatro app	36
Figura 11. Bosquejo cinco app	37
Figura 12. Diseño de base de datos	39
Figura 13. Inputs en pantallas 1, 2 y 3.	44
Figura 14. Inputs en pantallas 4 y 5	44
Figura 15. Charla con personal de Provia	46
Figura 16. Menú de accidentes vacío	51
Figura 17. Menú de accidentes	52
Figura 18. Menú de accidentes versión móvil	52
Figura 19. Slide menú celular	53
Figura 20. Filtro de menú de accidentes	54
Figura 21. Filtro de menú de accidentes	54
Figura 22. Datos del accidente: Evento	55
Figura 23. Datos del accidente: Vehículos	56
Figura 24. Datos del accidente: Personas	56

Figura 25. Datos del accidente móvil	57
Figura 26. Datos del accidente móvil: Vehículo	57
Figura 27. Datos del accidente móvil: Personas	58
Figura 28. Perfil	59
Figura 29. Pantalla de formulario en sección de Datos Importantes con vista de “stepper” en formato desktop	60
Figura 30. Pantalla de formulario en sección de Datos Importantes en formato desktop	60
Figura 35. Pantalla de formulario en sección de Datos Importantes en formato móvil	61
Figura 36. Consulta al usuario sobre dar permiso de ubicación a Julius	62
Figura 37. Pantalla de formulario en sección de Descripción, parte superior, en formato desktop	63
Figura 38. Pantalla de formulario en sección de Descripción, parte inferior, en formato desktop	63
Figura 39. Pantalla de formulario en sección de Descripción, en formato móvil	64
Figura 40. Diálogo de retroalimentación de éxito	64
Figura 41. Pantalla de formulario en sección de Datos Generales en formato desktop	65
Figura 42. Pantalla de formulario de sección de Datos Generales en formato móvil	66
Figura 43. Diálogo de retroalimentación de éxito al registrar número de vehículos y personas involucradas	66
Figura 44. Pantalla de formulario en sección de Tipo de Vehículo, con la visualización de todos los vehículos, en formato desktop	67
Figura 45. Pantalla de formulario en sección de Tipo de Vehículo, con la visualización de todos los vehículos, en formato móvil	67
Figura 46. Pantalla de formulario en sección de Tipo de Vehículo, ya con un vehículo seleccionado, parte superior en formato desktop	68
Figura 47. Pantalla de formulario en sección de Tipo de Vehículo, ya con un vehículo seleccionado, parte inferior en formato desktop	68
Figura 48. Pantalla de formulario en sección de Tipo de Vehículo, diálogo de retroalimentación de error al llenar el formulario.	69
Figura 49. Pantalla de formulario en sección de Tipo de Vehículo, diálogo de retroalimentación de éxito, al guardar un vehículo.	69

Figura 50. Pantalla de formulario en sección de Tipo de Vehículo, ya con un vehículo seleccionado, parte de en medio, en formato móvil	70
Figura 51. Pantalla de formulario en sección de Personas Involucradas, con la visualización de todas las personas involucradas, en formato desktop	71
Figura 52. Pantalla de formulario en sección de Personas Involucradas, con la visualización de todas las personas involucradas, en formato móvil	71
Figura 53. Pantalla de formulario en sección de Personas Involucradas, parte superior del formulario en formato desktop.	72
Figura 54. Pantalla de formulario en sección de Personas Involucradas, sección de asiento de persona que iba en automóvil, en formato desktop.	72
Figura 55. Pantalla de formulario en sección de Personas Involucradas, sección de asiento de persona que iba en motocicleta, en formato desktop.	73
Figura 56. Pantalla de formulario en sección de Personas Involucradas, parte inferior, en formato desktop.	73
Figura 57. Pantalla de formulario en sección de Personas Involucradas, parte inferior, en formato móvil.	74
Figura 58. Pantalla de formulario en sección de Personas Involucradas. Sección de cálculo de ISS.	74
Figura 59. Pantalla de formulario, sección de Descripción, en modo edición.	77
Figura 60. Pantalla de formulario, sección de Tipo de Vehículo, en el idioma inglés	78
Figura 61. Pantalla Formulario, sección de Descripción, en el idioma francés.	78
Figura 62. Pantalla de formulario en sección de Descripción, en dispositivo Android	79
Figura 63. Pantalla de formulario en sección de Tipo de Vehículo, en dispositivo Android	80
Figura 64. Pantalla de formulario en sección de Personas Involucradas, en dispositivo Android	81
Figura 65. Login exitoso en Postman	83
Figura 66. Login fallido en Postman	84
Figura 67. Datos generales formulario	84
Figura 68. Get en Postman	85
Figura 69. PUT en Postman	85

Figura 70. Personas involucradas	86
Figura 71. Menú personas involucradas	86
Figura 72. Get de evento en PostMan	87
Figura 73. Datos generales	87
Figura 74. Visita a PROVIAL	89
Figura 75. Gráfica de resultados de opinión sobre formulario de Datos Importantes	90
Figura 76. Gráfica de resultados de opinión sobre formulario de Descripción	90
Figura 77. Gráfica de resultados de opinión sobre formulario de Vehículos Involucrados	91
Figura 78 Gráfica de resultados de opinión sobre formulario de Personas Involucradas.	91
Figura 79. Diagrama entidad relación	92
Figura 80. Comunicación con el servidor	93
Figura 81. Resultados de Catboost sin hyperparametros	94
Figura 82. Resultados con hyperparametros	94
Figura 83. Hyperparametros de catboost	94
Figura 84. Matriz de confusión para train	95
Figura 85. Matriz de confusión para dev	95
Figura 86. Matriz de confusión para test	96
Figura 87. Resultados finales Catboost	96
Figura 88. Resultados Random Forest sin hyperparametros	96
Figura 89. Resultados del entreno de hyperparametros Random Forest	97
Figura 90. Hyperparametros Random Forest	97
Figura 91. Matriz de confusión para train	97
Figura 92. Matriz de confusión para dev	98
Figura 93. Matriz de confusión para test	98
Figura 94. Resultados finales Random Forest	99
Figura 95. Distribución de las clases.	99
Figura 96. Distribución de clases con smote	99

Figura 97. Resultados de Catboost con smote sin hyperparametros	100
Figura 98. Resultados de smote con hyperparametros	100
Figura 99. Hyperparametros de catboost con smote	100
Figura 100. Matriz de confusión para train	101
Figura 101. Matriz de confusión para dev	101
Figura 102. Matriz de confusión para test	102
Figura 103. Resultados finales Catboost smote	102
Figura 104. Resultados Random Forest con smote sin hyperparametros	102
Figura 105. Resultados Random Forest con smote sin hyperparametros	103
Figura 106. Hyperparametros Random Forest con smote	103
Figura 107. Matriz de confusión para train	103
Figura 108. Matriz de confusión para dev	104
Figura 109. Matriz de confusión para test	104
Figura 110. Resultados finales Random Forest con smote	105
Figura 111. simulación de posible percance vehicular a través de información del GPS	
Figura 112. Resultados de prueba de carga con 300 usuarios por minuto	107
Figura 113. Resultados de prueba de carga con 10,000 usuarios por minuto	108
Figura 114. Visita en las instalaciones de call center de provial.	123
Figura 115. Mapa de cobertura de patrullas de provial.	123
Figura 116. Equipo de provial y equipo de proyecto Julius.	124

Resumen

La idea del proyecto nace de la historia de un niño que falleció debido a un accidente vehicular, ocurrido en Guatemala. Este niño y muchas personas más son víctimas de un desorden vial que existe en muchos países alrededor del mundo. En honor a este niño de la historia, que fue una víctima más de lo que ocurre en Guatemala, el proyecto desarrollado adquirió su nombre: Proyecto Julius. Por lo tanto, el presente trabajo fue creado para desarrollar una plataforma tecnológica que permita almacenar la información más relevante de un accidente vehicular. La información que se pretende guardar está basada en un estándar internacional llamado IRTAD, el cual es utilizado en 63 países alrededor del mundo.

Se desarrolló una *progressive web app* con el framework Angular para el desarrollo de componentes. Se realizaron pruebas con el personal de PROVIAL para determinar que el público objetivo encuentre útil la aplicación. Los resultados obtenidos fueron positivos, pues en general, están de acuerdo en cómo está distribuida la información dentro de la aplicación. La petición de la información sobre el accidente también es de las mejores pues solamente el usuario debe seleccionar la opción correcta, y no escribirla manualmente. También, la visualización de los accidentes les pareció que fue ordenada, estructurada y fácil de entender. Se recomienda contactar más instituciones para que sean partícipes del proyecto.

También se creó un *backend* en Python utilizando el framework de flask para poder así realizarlo de una manera más fácil y rápida para la integración. Paralelamente se realizaron algoritmos de machine learning para poder predecir dependiendo de las características donde es más probable que ocurran accidentes de tránsito y bajo qué condiciones, para ello se utilizaron los algoritmos de clasificación Catboost y Random Forest, al mismo tiempo que se aplicó optimización de hypermarametros y data augmentation con la técnica de smote.

I. Introducción

Guatemala es un país que cuenta según los registros de 2020, con 4 millones 006 mil 883 vehículos según la Superintendencia de Administración Tributaria (SAT) (Aguilar, 2020), principalmente en la ciudad de Guatemala, en la que, en el año 2019 se tienen registros de que transitaban aproximadamente 1.5 millones de vehículos (García O. , 2019), por lo que es normal que ocurran accidentes, según la Policía Municipal de Tránsito de Guatemala (PMT) reporta en promedio ocurren 35 accidentes de tránsito diarios desde el 1 de octubre del 2020. (Mutz, 2020)

El Instituto Nacional de Estadística (INE) de Guatemala posee diferentes bases de datos acerca de accidentes vehiculares, entre las cuáles están los Accidentes en general, Vehículos involucrados y, Heridos y Lesionados, sin embargo, estas bases de datos presentan problemas de estructura y además, no se almacena parte de la información que es relevante.

Los principales problemas que tienen estas bases de datos son que los diccionarios que ellos mismos proveen no son congruentes entre años y carecen de información para algunas variables. Además, es imposible establecer una relación entre las tres bases de datos debido a la falta de códigos que permitan unificar las tablas. Asimismo, casi el 90% de los registros poseen variables descritas como “ignorado”, esto provocaba que se perdiera información importante. También, los códigos de colores y marcas no coinciden de un año a otro y los códigos correlativos de las tablas no coinciden entre sí provocando que crear una relación entre las tablas sea imposible.

En la actualidad, existe un estándar llamado IRTAD (International Traffic Safety Data and Analysis Group, por sus siglas en inglés), el cual cuenta con más de 40 países entre los cuales están: Estados Unidos, Argentina, Japón, Corea, Alemania, Francia, entre otros, con el propósito de convertirse en una fuerza central en la promoción de la cooperación internacional en datos sobre accidentes de tráfico y su análisis. Sin embargo, a pesar de saber de la existencia de este estándar, el INE no se apega a ninguno, lo que provoca los problemas antes mencionados.

Por lo tanto, este proyecto titulado como proyecto Julius, buscará implementar este estándar en Guatemala con el propósito de facilitar la obtención de información sobre los accidentes vehiculares y poder ser un país que sume a la recopilación de estos datos. Por medio de una aplicación web, que pueda ser accedida a través de cualquier dispositivo inteligente o computadora.

El objetivo general planteado busca poder resolver la problemática de cómo se recopila la información, cómo se analiza y en qué se puede aplicar para que estos últimos tengan valor. Por esto mismo, se relaciona con la justificación puesto que en esta, se presentan problemas en cómo el INE, Instituto Nacional de Estadística, en Guatemala, no tiene la información almacenada correctamente, y por lo mismo hace casi imposible analizar estos datos. Así pues, en este proyecto se propone una posible solución para poder recabar datos acerca de accidentes vehiculares que involucran la participación de diversas instituciones en el país.

Se realizaron pruebas de usabilidad con el equipo de provial. Se realizaron estas pruebas con el objetivo de conocer la opinión y el impacto que podría tener este proyecto en Guatemala, y a ellos como usuarios finales. Se pudo concluir de esta reunión que la aplicación aporta mucho valor a las labores de estas instituciones que velan por la seguridad vial. La razón de esto es porque facilita el trabajo, que antes se realizaba manualmente, de recopilar información y consultarla, también, que la funcionalidad de que múltiples usuarios modifiquen un accidente al mismo tiempo sin alterar el resultado de otro, puede reducir mucho el tiempo de recabado de información en el momento de un accidente vehicular.

II. Objetivos

2.1 Objetivo general

Reducir el número de muertes por accidente vehicular, determinando la causa raíz en todos los casos, con una base de datos estructurada y un proceso de captura de datos eficiente y con margen a error nulo.

2.2 Objetivos específicos

- Utilizar el estándar IRTAD en Guatemala para almacenar la información acerca de accidentes de tránsito.
- Mostrar información de forma organizada y estructurada para consultar información de accidentes de tránsito.
- Facilitar el análisis de información estructurada y previamente validada.
- Digitalizar completamente el proceso de captura de datos.
- Brindar código open source del proyecto para ofrecerlo al mundo y que cualquier país pueda utilizarlo con el propósito de salvar vidas.
- Proporcionar más herramientas al usuario para la recopilación de datos.
- Desarrollar servicios para el buen manejo y conexión con una API.
- Permitir la creación, edición o eliminación en la recopilación de datos de un accidente de tránsito.
- Garantizar que cualquier usuario con un smartphone y wifi tenga acceso a la aplicación.
- Desarrollar una aplicación que permita, por medio de formularios, almacenar la información de un accidente vial.
- Facilitar la comunicación entre el programa que procesa la información y la guarda en las bases de datos y el sitio que permite las entradas de datos.
- Aumentar la cantidad de datos sobre los accidentes vehiculares, basándonos en los datos que ya tenemos por medio de datos sintéticos, evitando así la escasez de datos.
- Facilitar la predicción de accidentes y donde es que más ocurren por medio de machine learning.
- Realizar pruebas con los sensores del dispositivo móvil, dígame acelerómetro y GPS, para definir un posible percance vehicular.
- Detectar un posible percance vehicular a través del uso del acelerómetro y GPS.
- Recolectar hora y ubicación del dispositivo móvil al momento en el que se detecte un posible percance.

- Enviar información recolectada a la base de datos Julius haciendo uso de los datos móviles del usuario.

III. Justificación

El tema de accidentes de tránsito es muy mencionado en Guatemala, sobre todo en el área metropolitana, y no hay un sistema actualmente en la región, que haga la labor de recopilar información, más que los registros que proporciona la Policía Nacional Civil (PNC), que como se ha mencionado previamente, no está completa y no tiene tanta utilidad al momento de realizar análisis, debido que para cualquier entidad o usuario que quiera realizar algún estudio orientado a seguridad vial o accidentes viales en Guatemala para un proyecto, le será un obstáculo el poder ordenar y poner de manera congruente la información, para realizar un análisis de esta.

Para manejar este problema, se utilizará el estándar IRTAD, que ha mostrado ser muy efectivo y es utilizado en gran cantidad de países como Japón, Estado Unidos, Suiza, y muchos otros más, mostrando así su validez de eficiencia, por lo cual en Guatemala podría ayudar en muchos proyectos viales en el futuro. Por experiencia, a la hora de trabajar con la información que nos proporciona el INE es muy complejo realizar análisis sobre estos datos, debido a que está muy desorganizada, es incongruente y causa más problemas que beneficios.

A nivel internacional existe el Reporte Anual de Seguridad Vial en el que participan más de 41 países, elaborado por el Grupo Internacional de Análisis y Datos de Seguridad Vial (IRTAD), el grupo de trabajo sobre seguridad vial del Foro Internacional de Transporte (ITF) de la OCDE. Se publica con motivo de la Cumbre Anual de la ITF sobre "Seguridad y Protección del Transporte".

Reducir el número de víctimas en las carreteras requiere una acción continua basada en el análisis de datos de seguridad vial de buena calidad. En varios países, las medidas fáciles de implementar ya están en vigor. Para reducir aún más el número de muertes y lesiones graves en las carreteras, se deben utilizar más datos: por ejemplo, sobre las circunstancias de los accidentes, sobre los mecanismos que conducen a los accidentes y la determinación de su gravedad, así como sobre los usuarios de la carretera involucrados. También se necesitan datos para evaluaciones proactivas de riesgos de la red de carreteras.

Por ejemplo, la siguiente gráfica muestra la reducción o aumento de las muertes causadas por accidentes de tránsito en general, donde se puede notar que la reducción más fuerte de accidentes la tiene Portugal, seguido de Lituania, Noruega y Grecia. Por el lado contrario, Estados Unidos, Jamaica y Colombia presentan un aumento de muertes por accidente vehicular.

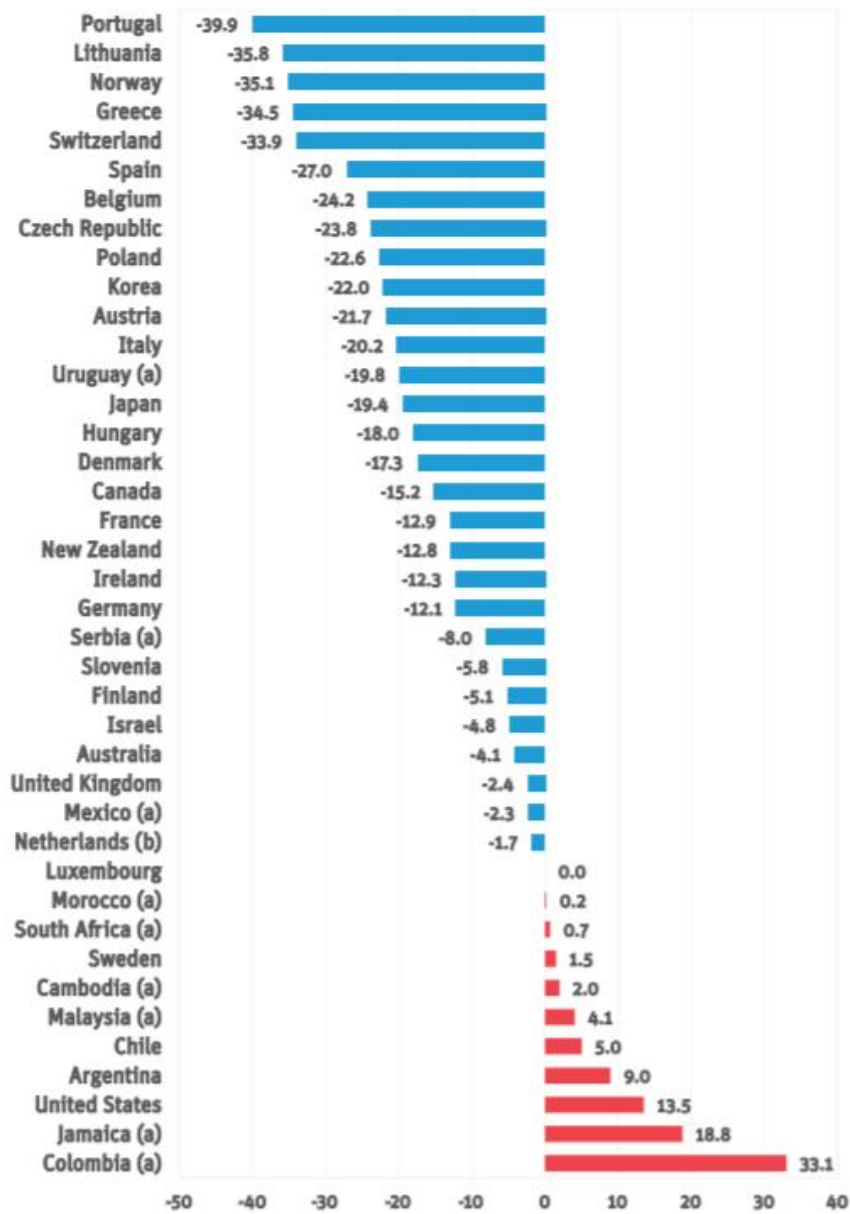


Figura 1. Aumento y reducción de muertes por país.

Por otro lado, la siguiente estadística muestra el porcentaje de muertes de peatones separados en grupos de edad, donde podemos notar que el grupo de edad con más muertes cambia mucho por país.

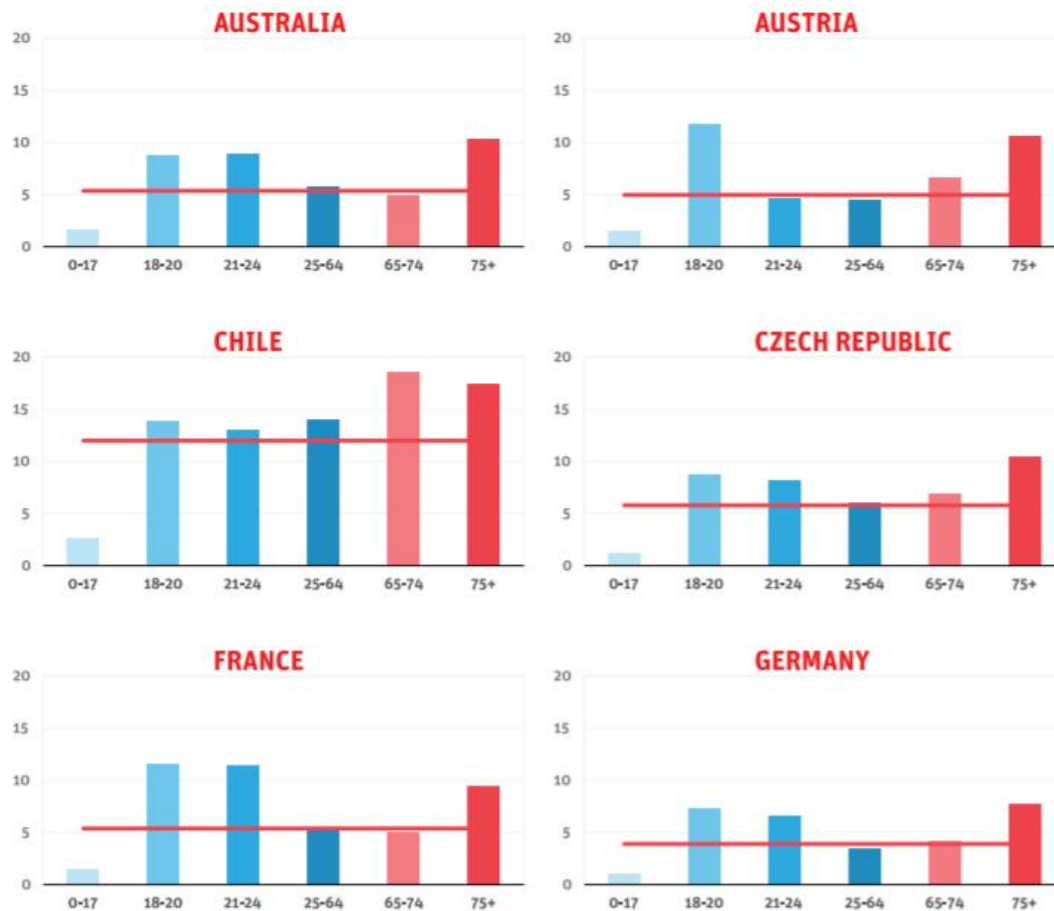


Figura 2. Porcentaje por edad de muertes de peatones.

Estos resultados y conclusiones fueron posibles de obtener gracias a la implementación del estándar IRTAD en Europa. Por lo tanto, la importancia de la realización de este proyecto es muy alta puesto que, como se mencionó anteriormente, Guatemala no cuenta con ninguna regulación para obtener datos de accidentes vehiculares. Debido a esto, implementar el estándar IRTAD o algún otro utilizado, permitiría que la recopilación de datos sea más eficiente y mejor.

El desarrollo de la aplicación para toma de datos está muy ligada con la implementación del estándar porque a través de esta aplicación, le permitiría al usuario que quiera recopilar la información, algún agente de tránsito o usuario común, que sea más amigable con el usuario. De esta forma, la información se guardará de forma ordenada, limpia y fácil. Permitiendo así, que en un futuro, algún analista de datos pueda utilizar la base de datos, que ha sido utilizada para guardar estos incidentes, para establecer causas raíz y otros factores.

Un caso en específico es Noruega, en 2019, logró determinar estadísticas muy importantes que permiten crear leyes o acciones en el país para evitar que más de estos accidentes sucedan. Entre los datos más importantes está que el 40% de accidentes fatales suceden debido a la velocidad excesiva o inapropiada del conductor. El 8% de los accidentes fatales se deben al uso de alcohol y el 12% de accidentes fatales fueron debido al consumo de drogas. La ley en Noruega estipula que los dispositivos móviles deben estar sostenidos por un objeto en el carro o bien, cercano al conductor puesto que 50 accidentes fueron causados por distracciones del conductor, y 5 de estos, específicamente por ver el teléfono. El 26% de los accidentes fueron por fatiga y por adormitación. Todos estos datos recabados están incluidos en el estándar IRTAD, por lo tanto, permiten llegar a conclusiones tangibles, algo que a Guatemala le hace falta.

IV: Marco teórico

4.1 Accidentes de tránsito

Un accidente de tránsito es definido como un hecho o evento que ocurre sobre la vía pública, y se presenta de pronto y de forma inesperada. Es causado por condiciones y acciones irresponsables que son potencialmente previsibles. También, son atribuidos a factores humanos, vehículos especialmente automotores, condiciones meteorológicas y climatológicas, mala señalización y caminos en mal estado. Un accidente de tránsito ocasiona pérdidas prematuras de vidas humanas o lesiones en los participantes del evento, así como secuelas físicas o psicológicas, pérdidas materiales y daños a terceros (*CONCEPTOS Y DEFINICIONES*, s. f.).

Existen diferentes clases de accidentes:

- **Colisión:** Se define una colisión como un choque entre uno y más vehículos en movimiento.
- **Atropello:** Es el evento en el cual uno o varios peatones son arrollados por un vehículo en movimiento.
- **Accidentes de tránsito fatales:** Se utiliza este término para indicar que hubo una o más personas fallecidas como resultado del accidente.

Dentro de un accidente vial están implicados diferentes objetos, los cuáles es importante definir para tener una mejor comprensión del tema y es importante utilizar los términos correctos para evitar confusiones.

- **Vehículo:** Es cualquier artefacto en el cual se transportan personas y/o cosas.
- **Vía (pública):** Es toda calle, avenida, camino o carreteras destinadas para el tránsito de vehículos. También están incluidos estacionamientos para vehículos.
- **Conductor implicado:** Es aquella persona que conduce un vehículo en la vía pública y está involucrado en el accidente de tránsito.
- **Víctimas:** Son todas las personas que resultaron heridas o muertas a causa de un accidente de tránsito. Una persona es considerada como herida a la persona lesionada, ya sea grave o leve, en un accidente de tránsito; y muerta, si fallece como consecuencia del accidente.
- **Intersección:** Es el área en donde se cruzan dos o más vías.



Figura 3. Intersección

- **Calle dividida o autovía:** Una carretera de doble calzada o carretera dividida es una clase de carretera con calzadas en la que el tráfico que viaja en direcciones opuestas está separadas por una reserva central.



Figura 4. Calle dividida

- **Calzada única:** Es como una autovía o calle dividida, con la única diferente que esta calle no está dividida por una reserva central entre cada una.



Figura 5. Calzada única

4.2 Estándar IRTAD

El Grupo de análisis y datos de seguridad del tráfico internacional, o International Traffic Safety Data and Analysis Group (IRTAD), es un grupo de trabajo permanente sobre seguridad vial que tienen como objetivo promover el conocimiento internacional sobre seguridad vial y contribuir a reducir el número de accidentes de tránsito. IRTAD se ha convertido en una fuerza central en la promoción de la cooperación internacional en datos de accidentes de tránsito y su análisis. Ha sido aclamado como “modelo de un esfuerzo de varios países” y sus datos de accidentes descritos como “los mejores del mundo” (IRTAD, 2021).

IRTAD a lo largo del tiempo, se ha convertido no solo en la base de datos, sino también en el grupo de análisis y datos de seguridad del tráfico internacional. Este grupo reúne expertos en seguridad vial de administraciones viales nacionales, institutos de investigación de seguridad vial, organizaciones internacionales, asociaciones de automóviles, compañías de seguros, fabricantes de automóviles y otros. IRTAD también ha facilitado la alianza entre países que se esfuerzan por mejorar su historia de seguridad vial mediante la mejora de sus datos sobre accidentes de tráfico. Este trabajo ha sido apoyado por el Banco Mundial, el Banco Interamericano de Desarrollo y la Fundación FIA (IRTAD, 2021).

La base de datos IRTAD contiene datos validados y actualizados de accidentes y exposición de 42 países. La base de datos internacional de tráfico y accidentes por carretera incluye datos de seguridad y tráfico, agregados por país y año desde 1970. Todos los datos se recopilan directamente de proveedores de datos nacionales en los países del IRTAD. Se proporciona en un formato común, basado en definiciones desarrolladas y acordadas por el grupo IRTAD. Los países que aportan datos son: Argentina, Australia, Austria, Bélgica, Canadá, Chile, República Checa, Dinamarca, Finlandia,

Francia, Alemania, Grecia, Hungría, Islandia, Irlanda, Israel, Italia, Japón, Corea, Lituania, Luxemburgo, Países Bajos, Nueva Zelanda, Noruega, Polonia, Portugal, Eslovenia, España, Suecia, Suiza, Reino Unido, Estados Unidos (IRTAD, 2021).

4.3 AppSheet

AppSheet es una plataforma de software desarrollada por Google que permite crear aplicaciones móviles y web sin necesidad de programar. Las ventajas que presenta AppSheet son:

- **Medios para creación de contenido:** Esta plataforma permite aprovechar todo el potencial de sus datos para que todos los usuarios puedan crear aplicaciones y automatizaciones personalizadas.
- **Fomenta la innovación:** Desarrolladores y personas de IT pueden colaborar para desarrollar funciones de gestión y política empresarial.
- **Optimiza el trabajo:** La creación de aplicaciones pueden ayudar a ahorrar tiempo y a integrar otras herramientas y aplicaciones de Google.

Las características que AppSheet describe son las siguientes:

Automatización de procesos y colaboración mejorada

AppSheet combina la tecnología para desarrollar sin código con la IA de Google para poder automatizar procesos empresariales más fácilmente y desde cualquier lugar, y realizado por cualquier persona. Permite desarrollar aplicaciones rápidamente con automatizaciones, para no desperdiciar tiempo ni talento.

Recolección de datos enriquecidos y sincronización sin conexión

AppSheet recoge datos de formas más inteligentes, tanto online como offline. AppSheet utiliza funcionalidades de los dispositivos para añadir datos más valiosos, como ubicaciones GPS, imágenes, dibujos, análisis de códigos de barras y reconocimiento de caracteres, entre otros.

Utiliza el aprendizaje automático y la IA de Google para crear más rápido y optimizar tus aplicaciones

Dirige las aplicaciones con el procesamiento del lenguaje natural (PLN), captura imágenes con la visión artificial y crea modelos predictivos que aprendan a generalizar a partir de aplicaciones desarrolladas previamente; todo esto sin ningún tipo de experiencia en aprendizaje automático.

Escala con control

Permite que los usuarios: desarrolladores y personas de IT puedan trabajar de forma conjunta. Define políticas y jerarquiza, y personaliza la experiencia de tu equipo mostrando solo información según el rol del usuario, el empleado u otros factores.

4.4 Progressive Web App

Una Progressive Web App (PWA) o una aplicación web progresiva que trabaja en conjunto de una aplicación web o página web como normalmente se conoce, pero incorpora particularidades y funcionalidades de un teléfono móvil que la hacen parecer como una aplicación nativa de un sistema operativo iOS o Android (Vidal, 2019). Esta solución para una aplicación tiene grandes ventajas, como las siguientes:

1. La principal ventaja es que **no requiere de una tienda de aplicaciones** como App Store o Google Play para descargarse e instalarse en el teléfono móvil. Solamente es requerido de, como requisito de instalación únicamente, conexión a internet, y un navegador web como Chrome, Safari, Firefox, Opera, etc.
2. También, cualquier usuario que cumpla con los requisitos de conexión a internet y que cuente con un navegador web, **este podrá instalar la aplicación** en el teléfono. Lo interesante de esto, es que la PWA funciona como un acceso directo hacia la aplicación web, por lo tanto no ocupa mayor espacio como lo hace una aplicación nativa de la tienda de aplicaciones, y es posible recuperar el contenido de la aplicación aún si el usuario no tiene acceso a internet, o se actualiza la información al acceder a internet.
3. Esta estrategia permite el uso de **push notifications**, que le permite saber al usuario y recordarle sobre nuestra aplicación y darle información acerca de algún contenido nuevo, ya sea en un teléfono Android o un sitio web.

Existe una configuración estándar para el correcto funcionamiento de una PWA. Esta configuración consiste en un archivo manifiesto, en formato JSON. Este manifiesto nos permite controlar configuraciones que el usuario verá al momento de utilizar la aplicación, la configuración es la siguiente:

- **Name:** Nombre de la aplicación que aparecerá en el menú del usuario.
- **Description:** La descripción de la aplicación móvil.
- **Icons:** Despliega distintos iconos, con resoluciones distintas, para que de esta manera, se vea bien en todos los dispositivos.
- **Start url:** URL donde abrirá la aplicación.
- **Display:** Se puede elegir entre standalone, fullscreen, minimal-ui, entre otros.

- **Orientation:** Se decide si la aplicación web se usará en modo retrato o en modo paisaje.
- **Theme_color:** El color que se usará para la barra superior de la aplicación.
- **Background_color:** Color para la pantalla antes de la carga completa de la aplicación.

(Vidal, 2019)

Otro elemento que es muy importante a tomar en cuenta para desarrollar una progressive web app, es el uso de un Service Worker. Un Service Worker es un servicio o proceso que se instala en el navegador Web, que se ejecuta en segundo plano y que nos permitirá ejecutar código Javascript. Sus características principales:

- Está guiado por eventos fetch que se producen en un dominio y una ruta.
- Se ejecuta en segundo plano.
- Debido a que funciona en modo offline permiten realizar acciones fuera de la aplicación y devolver elementos por caché cuando la red no está disponible.
- Permite interceptar los eventos de red y realizar los ajustes que se desea.
- Funciona solo sobre HTTPS por motivos de seguridad.

(Etxarri, 2018)

4.5 Framework

Un framework es un entorno de trabajo con la finalidad de facilitar el trabajo del programador brindando una serie de características y funciones que aceleran el proceso, reducen los errores, favorecen el trabajo colaborativo y se logra desarrollar un producto de mayor calidad. Los framework ofrecen una estructura para el desarrollo y cuentan con gran variedad para utilizar en varios lenguajes de programación.

Cabe mencionar que los desarrolladores pueden crear aplicaciones como una página web sin utilizar un framework, como en proyectos de tamaño reducido. Sin embargo, en ciertos proyectos, tienden a crecer en complejidad y tamaño, por lo que necesitan una mejor organización, seguir principios como código fácil de entender y otros aspectos que el uso de un framework lo facilita. Los beneficios principales que ofrece un framework son los siguientes:

Incremento en la velocidad de desarrollo

Utilizar un framework permite tener un aumento de la velocidad a la hora de programar. Los frameworks incluyen la opción de realizar tareas comunes en la programación de forma automatizada y fácil, como lo es crear un botón o una ventana emergente, por ejemplo. Permite también reutilizar código, que, por lo tanto, permite afrontar múltiples proyectos utilizando el mismo código optimizado, con el mismo resultado.

Reducción del número de errores de programación

A la hora de programar es habitual cometer errores de distinto tipo (por ejemplo, de sintaxis). Con el uso de un framework este tipo de errores se elimina o minimiza, consiguiendo reducir todo el tiempo necesario para encontrarlos y eliminarlos.

Facilita la colaboración

Los proyectos de desarrollo web o software usualmente están conformados por un equipo de desarrollo. Sin utilizar un framework que aportase una estructura, estándares, y normas, el esfuerzo empleado por los distintos miembros del equipo serán mucho mayores para entender los códigos y trabajar de forma conjunta. Con la ayuda de un framework, el equipo de desarrollo puede compartir código y trabajar en conjunto, lo que dará como resultado un proceso de programación de proyectos mucho más sencillo y rápido.

(Rodríguez, 2020)

4.6 Dispositivos móviles

En la actualidad la definición de dispositivo móvil puede variar según algunas características como lo son el tamaño de la pantalla y la funcionalidad de su equipo, pueden ser desde computadoras hasta teléfonos móviles. Los teléfonos inteligentes además de la comunicación por voz pueden incorporar funciones avanzadas como mensajería, acceso a Internet móvil, pantalla táctil u otros dispositivos de entrada de datos, captura de imágenes fijas y en movimiento, reproducción de documentos digitales, posicionamiento GPS, acceso a redes inalámbricas. Estos equipos avanzados incorporan sistemas operativos tales como Symbian, Android, iOS (p.e. iPhone), Windows Phone y BlackBerry.

(Cadavieco *et al.*, 2012)

Cada uno de los dispositivos cuenta con un sistema operativo el cual controla todos los recursos de los dispositivos inteligentes y que ofrece el soporte básico sobre el cual pueden escribirse los programas de aplicación (Stallings, 2004). Entre los sistemas operativos más utilizados e importantes de la actualidad se encuentran algunos de los dispositivos inteligentes como laptop, celulares, etc. son Android e iOS.

Dentro de los dispositivos móviles existen distintos tipos de sensores los cuales permiten a estas herramientas cumplir con sus tareas. Son sensores sensibles a alguna forma de energía en el ambiente que pueden ser luminosos, cinéticos o relacionados con información sobre una cantidad

física que necesita ser medida, como: temperatura, presión, velocidad, corriente, aceleración, posición, etc.

(Wendling, 2010)

Para una mejor comprensión del proyecto es necesario mencionar cómo funcionan los acelerómetros junto con los GPS. Un acelerómetro mide la tasa de cambio de la velocidad de un objeto. Es un componente mecánico muy parecido a un chip. Este es utilizado por un dispositivo móvil principalmente para saber la orientación en la que está colocado, de manera en la que pueda saber cuándo un usuario lo está viendo de manera horizontal o vertical, e incluso cuando lo coloca boca abajo. Suele estar compuesto de tres ejes para medir el movimiento en un espacio tridimensional. Por otro lado el GPS es un componente electrónico que utiliza conjuntamente una red de ordenadores y una constelación de 24 satélites para determinar por triangulación, la altitud, longitud y latitud de cualquier objeto en la superficie terrestre.

(Pozo-Ruz *et al.*, 2000)

4.7 Frontend

Un *Frontend* es el concepto que se utiliza para referirse a la parte de una aplicación que está relacionada con aquello que cualquier usuario puede ver, incluyendo aspectos de User Interface y User Experience. También representa la parte funcional, en la que el usuario puede interactuar, por lo tanto debe ser útil y con buena estética. En otras palabras, un *frontend* es la capa superior que comprende menús desplegables, imágenes, íconos, colores, elementos gráficos, animaciones y, que mejora la experiencia del usuario al utilizar una aplicación o sitio web (de Souza, 2021).

4.8 Frameworks para frontend

4.8.1 Flutter

Es un framework desarrollado por Google, y utiliza el lenguaje Dart para el desarrollo. Al trabajar con Flutter, se basará en un mismo código para aplicaciones de Android y de IOS al mismo tiempo. Tiene widgets, es decir, Material Design para Android y Cupertino para iOS, que sigue las pautas de ambas plataformas. El uso de Dart para Flutter, lo ayuda a usar la compilación Just-in-Time que mejora el flujo de trabajo de desarrollo. Dart le ofrece un rendimiento nativo para diferentes plataformas, transiciones suaves y animaciones a 60 FPS.

Pros	Contras
Recarga caliente: hacer algún cambio en el código se podrán ver los efectos reflejados inmediatamente, sin tener que compilar la aplicación de nuevo y sin perder el contexto en el que estábamos.	Dart necesario: para poder usar Flutter es necesario aprender el lenguaje de programación Dart.
Renderizado de vistas muy rápido y constante.	Framework muy joven: y aún no tiene una gran comunidad por detrás, por lo que se deberán afrontar los problemas que nos encontremos con menos ayuda que en otros frameworks.
Desarrollo multiplataforma: Flutter ya genera un código base que sirve para ambas plataformas.	Está enfocado solo a móvil: por el momento solo hay una versión oficial de Flutter y solo está enfocada para móvil. Así, si nuestra aplicación va a tener un sitio web tendremos que desarrollar paralelamente la versión de móvil.
Acceso a las funciones nativas: algunas funciones específicas de la plataforma, como la cámara y la geolocalización, requieren acceso a funciones nativas. Estas funciones deben implementarse mediante lenguajes nativos, y Flutter da la sensación de desarrollarse en la plataforma nativa.	Librerías limitadas: las bibliotecas a las que pueden acceder los desarrolladores de aplicaciones móviles están muy limitadas en Flutter.

Tabla 1. Ventajas y desventajas de flutter.

4.8.2 Xamarin

Es un framework desarrollado por Microsoft, que utiliza el lenguaje C#. Es una herramienta de .NET para construir diferentes tipos de aplicaciones, y utiliza herramientas y librerías específicas del tipo de app a construir. Es open-source y gratis. Xamarin al ser de Microsoft tiene una versatilidad para conectarse con Azure para backend. Ofrece un ecosistema con backend, API, componentes, etc.

Pros	Contras
Hardware support total.	Familiaridad con código de plataforma específico quizá sea necesaria.
Compatibilidad de MVC y MVVM arquitectura.	Soporte mínimo con la comunidad.
Sistema de desarrollo completo (.NET y Visual Studio).	No es una gran elección para aplicaciones con UI complejo.
Open-source y gratis.	El desarrollo de UI consume tiempo y no es mobile-friendly.
Tiene Xamarin Component Store que provee muchos componentes.	

Tabla 2. Ventajas y desventajas de Xamarin.

4.8.3 React Native

Es un framework desarrollado por Facebook, Inc., basado en una librería de Javascript React.

Pros	Cons
Acceso a características y funcionalidades nativas, como cámara.	Tiene paquetes y librerías que pueden estar abandonadas.
Fácil de aprender.	Requiere muchas actualizaciones.
Fácil proceso de desarrollo	No tiene la capacidad de trabajar en múltiples pantallas.
UI de alta calidad.	Problemas para trabajar con transiciones y animaciones complejas.
Open-source y free.	La navegación no es nada como una nativa.
Carga rápida.	No existe la seguridad con los plugins por terceros.

Tabla 3. Ventajas y desventajas de React Native.

4.8.4 Angular

Es un framework desarrollado por Google, Inc, que utiliza lenguaje JavaScript, en conjunto con HTML.

Pros	Cons
Soporte de Google, para paquetes y actualizaciones. Tiene la garantía que todos los paquetes tendrán mantenimiento y confianza.	Curva de aprendizaje muy alta.
Integraciones de third party que son agregadas de forma sencilla.	Es un proyecto que puede ser pesado debido a todas las herramientas que contiene.
Carga de tiempo rápido y seguridad debido que compila todo en JavaScript.	Puede ser muy rígido, a pesar de ser un orden, pero puede causar problemas
Es muy customizable, es sencillo para diseñar y hacer responsive.	
Dependency Injection, para facilitar trabajo con aplicaciones de larga escala	

Tabla 4. Ventajas y desventajas de Angular.

4.8.5 Vue

Pros	Cons
Es accesible puesto que es natural para el desarrollador, con conocimientos básicos HTML, CSS y Javascript.	Vue.js no tiene la mayoría de los complementos comunes, y esos son los inconvenientes de Vue.js. Vue.js evoluciona rápidamente. Lo que funciona hoy en Vue puede quedar obsoleto mañana.
Su desarrollo es fácil porque la producción pesa 20 KB luego de luego de minimizarlo, aún más liviano Angular, ReactJs y jQuery.	Si está utilizando la aplicación Vue.js en una versión anterior de iOS y Safari, se supone que tendrá algunos problemas, aunque se pueden solucionar.
Permite el desarrollo de aplicaciones complejas como integración de partes más pequeñas, por lo que también es escalable.	
Curva de aprendizaje muy baja, ya que hay mucha documentación y el desarrollo es intuitivo.	

Tabla 5. Ventajas y desventajas de Vue.

4.9 Backend

El *Backend* de una aplicación hace referencia a todo el código o programa al que los usuarios no tienen acceso y al cual no pueden interactuar. Este código usualmente está compuesto por múltiples lenguajes de programación, los cuáles dependen de la funcionalidad que se desea emplear. El *backend* incluye actividades como escalabilidad y disponibilidad de red, transformación de datos y gestión de bases de datos, marcos de prueba automatizados, ciberseguridad y prácticas de respaldo de datos, etc (de Souza, 2021).

Dependiendo del tipo de aplicación que se desea desarrollar y qué parte del proyecto se implementará, se pueden incluir lenguajes como Java, Ruby, Python, PHP, .Net, MySQL, Oracle, PostgreSQL, MongoDB, entre otros (de Souza, 2021). Un desarrollador backend, se encarga de diseñar la lógica y las soluciones para que todas las acciones solicitadas en una página web sean

ejecutadas de manera correcta. Trabaja del lado del servidor y procesa la información recibida a través del frontend (*¿Qué hace un Desarrollador Backend?*, s. f.).

El desarrollador backend debe estudiar los diferentes lenguajes de programación que pueden ser necesarios para desarrollar su trabajo. Debe formarse como desarrollador de aplicaciones web o como desarrollador de aplicaciones multiplataforma. Además, necesita conocer las interacciones con diferentes bases de datos (*¿Qué hace un Desarrollador Backend?*, s. f.).

4.10 Bases de datos relacionales

4.10.1 ¿Qué es una base de datos relacional?

En 1970, E. F. Codd propuso que los sistemas de bases de datos deberían presentarse a los usuarios con una visión de los datos organizados en estructuras llamadas relaciones, definidas como conjunto de tuplas (filas) y no como series o secuencias de objetos, con lo que el orden no es importante. Por lo tanto, detrás de una relación puede haber cualquier estructura de datos compleja que permita una respuesta rápida a una variedad de consultas. Entonces, Codd hizo énfasis en que el usuario de un sistema relacional sólo debía preocuparse por el qué consultar y no en cómo de las estructuras de almacenamiento (Quiroz, 2003).

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos y conjunto de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila en su respectiva tabla es un registro con un código único que sirve como identificador. Las columnas de dicha tabla poseen campos de los datos y cada registro tiene un valor para cada atributo, lo que genera una mejor estructura para crear relaciones entre diferentes tablas (Oracle, s. f.).

4.10.2 El modelo relacional

A inicios de las bases de datos, cada aplicación que se desarrollaba almacenaba datos en su propia estructura única. Por lo mismo, los desarrolladores debían conocer la estructura de datos concreta para encontrar los datos que necesitaban. Las estructuras de datos eran poco eficaces, el mantenimiento era complicado, y era muy difícil optimizar el rendimiento de las aplicaciones. El modelo de base de datos relacional se diseñó para resolver problemas causados por estructuras de datos múltiples y arbitrarias (Oracle, s. f.).

El modelo relacional brindó una forma estándar de representar y consultar datos que podía utilizarse en cualquier aplicación. Los desarrolladores se dieron cuenta que la ventaja principal era que la representación de la información podría usarse como tablas, ya que era una forma intuitiva, eficiente y flexible de almacenar y acceder a información estructurada (Oracle, s. f.).

Después de un tiempo, los desarrolladores empezaron a utilizar el lenguaje SQL, para guardar y realizar consultas a una base de datos. SQL con el tiempo, ha ganado popularidad y se han realizado mejoras pues se convirtió en el lenguaje utilizado para las bases de datos. Este lenguaje se basa en el álgebra relacional y un lenguaje matemático de uniformidad que facilita la mejora del rendimiento de todas las consultas en bases de datos (Oracle, s. f.).

4.10.3 Ventajas de las bases de datos relacionales

El modelo relacional es simple pero muy bueno y eficaz. Las bases de datos pueden ser utilizadas por cualquier tipo de organización y de tamaño. Las aplicaciones de una base de datos pueden ser rastrear inventarios, procesar transacciones de comercio electrónico, administrar cantidades enormes y esenciales de información de clientes y mucho más. Las bases de datos relaciones se pueden emplear para cualquier aplicación de datos en la que los puntos de datos se relacionan entre sí, y deban gestionarse de forma segura, conforme a normas y de un modo uniforme (Oracle, s. f.).

4.11 Gestores de base de datos

Los gestores de bases de datos son programas los cuales están encargados de poder trabajar con la información y gestionarla, por lo que ya mayoría buscan brindar una interfaz clara para poder manejarlos y poder así brindar una buena experiencia al usuario y hacer más intuitivo el manejo de las mismas, la mayoría de gestores también cuentan con versiones en consola para poder así utilizarlos en servidores o computadores que no cuentan con una interfaz gráfica en la cual desplegar el programa visual (PowerData, 2019).

4.11.1 Mysql

Es un gestor de bases de datos el cual fue creado por MySQLAB pero con el paso del tiempo fue adquirida por Oracle, Esta fue creada para poder trabajar de forma cliente servidor para poder aplicarse en software de todo tipo el cual permita la conexión con el mismo. La ventaja de MySQL es que cuenta con una versión gratis como una comercial, dependiendo de las necesidades de los usuarios.

4.11.2 Postgresql

Es un gestor de bases de datos relacional creado por la universidad de California en 1980 , es uno de los pioneros en la implementación de bases de datos comerciales, ofreciendo todas las facilidades que brindan los otros gestores basados en SQL, Uno de sus puntos fuertes es que es de código abierto por lo que permite a la comunidad buscar mejorar el mismo y así mantenerlo siempre lo más actualizado posible, también es muy fácil de usar ya que cuentan con herramientas como PgAdmin el cual hace muy fácil la navegación y control del mismo.

4.12 Compromiso y atomicidad

Las bases de datos relacionales gestionan reglas y políticas comerciales a un nivel minucioso, y tienen políticas estrictas sobre el compromiso de una transacción. Las transacciones en bases de datos relaciones están definidas por cuatro propiedades fundamentales, que estas funciones dan el nombre de ACID, su acrónimo en inglés:

- **Atomicidad:** Define todos los elementos que conforman una transacción completa de base de datos.
- **Uniformidad:** Define las reglas para mantener los puntos de datos en un estado correcto después de una transacción.
- **Aislamiento:** Impide que el efecto de una transacción sea visible a otros hasta que se establezca el compromiso, a fin de evitar confusiones.
- **Durabilidad:** Garantiza que los cambios en los datos se vuelvan permanentes cuando la transacción se haya fijado y hayamos llegado a un compromiso.

(Oracle, s. f.)

4.13 API

Una API es un conjunto de definiciones y protocolos, que se utiliza para desarrollar e integrar el software de las aplicaciones, las siglas de API son “Interfaz de Programación de Aplicaciones”, por sus siglas en inglés. Las API permiten que servicios y productos se comuniquen con otros de la misma índole, sin necesidad de conocer la implementación de estas. Esto simplifica el desarrollo de cualquier aplicación, pues nos permite ahorrar tiempo, dinero y mejora la eficiencia de la aplicación. También, otorgan flexibilidad, simplifican el diseño, la administración y el uso de las aplicaciones, además de proporcionar oportunidades de innovación (*¿Qué es una API?*, s. f.).

En ocasiones, las API son consideradas como contratos con documentación que representa un acuerdo entre las partes. Es decir, si alguna de las partes envía una solicitud con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de la otra parte. Como

consecuencia que simplifican la forma en que los desarrolladores integran nuevos elementos a las aplicaciones con su arquitectura actual, estas API permiten segmentar en partes más pequeñas el desarrollo de la aplicación (*¿Qué es una API?*, s. f.).

Las API son un medio para conectar la arquitectura del API a través del desarrollo de aplicaciones de la nube, y también, permite compartir datos con clientes o usuarios externos. Las API pueden ser utilizadas para alimentar de información la aplicación, o bien, se puede rentabilizar para utilizar esa información, como es el caso de la API de Google Maps. En otras palabras, las API permiten acceder a recursos y a información almacenada en una base de datos o calculadas en el backend (*¿Qué es una API?*, s. f.).

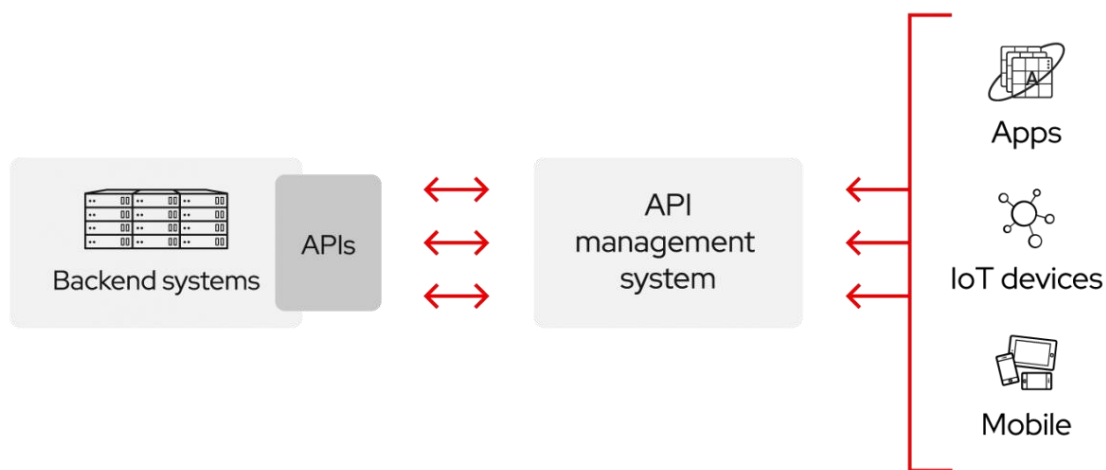


Figura 6. Diagrama de APIs.

Existen tres enfoques para las políticas del desarrollo de una API:

- **Privado:** Las API solamente son utilizadas internamente en la empresa, donde hay un mayor control sobre estas.
- **De partners:** Las API son compartidas con partners empresariales específicos, lo que permite que se tenga un mayor flujo de ingresos, sin comprometer la calidad.
- **Público:** Todo el público tiene acceso a las API, así que las compañías pueden desarrollar API que funcionen o interactúen con otras aplicaciones, y convertirse en una fuente de innovación. En este caso, terceros desarrollan aplicaciones que utilizan esta API pública.

(*¿Qué es una API?*, s. f.)

4.14 Servicios

Los servicios en Angular son clases que se encargan de acceder a los datos para entregarlos a los componentes. Su mayor ventaja es que se puede reutilizar servicios para distintos componentes.

Estos son un proveedor de datos, que mantienen lógica de acceso a ellos y operativa relacionada con el negocio y las cosas que se hacen con los datos dentro de una aplicación. Los servicios serán consumidos por los componentes, que delegaron en ellos la responsabilidad de acceder a la información y la realización de operaciones con los datos.

(CodingPotions, 2018).

4.15 REST

Es una interfaz para conectar varios sistemas basados en el protocolo HTTP, nos sirve para obtener y generar datos devolviendo estos datos en formatos como JSON o XML. Por lo general el más utilizado es el formato JSON, ya que es más fácil de entender que el XML y es más ligero que este.

(Rosa, 2018)

Ventajas de usar REST son:

- Nos permite separar el cliente del servidor, esto por ejemplo permite que nuestro “frontend” esté desarrollado en Angular usando Typescript mientras que la API esté creada en Python.
- Es totalmente independiente de la plataforma, funciona en Windows, Linux, Mac o el sistema operativo que nosotros queramos.
- Podemos hacer nuestra API pública, permitiendo darnos visibilidad si la hacemos pública.
- Nos da escalabilidad, porque podemos separar al cliente y al servidor, por tanto, podemos dedicarnos exclusivamente a la parte del servidor.

Para que una API se considere RESTful, debe cumplir con estos criterios:

- Una arquitectura cliente-servidor compuesta por clientes, servidores y recursos, con solicitudes gestionadas a través de HTTP.
- Comunicación cliente-servidor sin estado, lo que significa que no se almacena información del cliente entre las solicitudes de obtención y cada solicitud es independiente y no está conectada.
- Una interfaz uniforme entre componentes para que la información se transfiera de forma estándar. Esto requiere que:
 - Los recursos solicitados son identificables y separados de las representaciones enviadas al cliente.
 - Los recursos pueden ser manipulados por el cliente a través de la representación que reciben porque la representación contiene suficiente información para hacerlo.

- Los mensajes autodescriptivos devueltos al cliente tienen suficiente información para describir cómo el cliente debe procesarlos.
- Un sistema por capas que organiza cada tipo de servidor implica la recuperación de la información solicitada en jerarquías, invisibles para el cliente.

(Red Hat, 2020)

4.16 HTTP

Es un protocolo que nos permite enviar documentos de un lado a otro en la web. Determina qué mensajes se pueden intercambiar y qué respuestas devolver (Blancarte, 2018).

4.16.1 Métodos HTTP

Los métodos HTTP son los que definen qué acción se va a realizar, estos métodos se deben entender para comprender la forma en la que funciona la arquitectura REST previamente explicada, ya que con los datos que se le indican al servidor en una petición, una misma URL puede ser tratada de diferentes formas.

(Blancarte, 2018).

4.16.2 GET

Este método se utiliza únicamente para consultar información al servidor. Su equivalente es realizar un SELECT dentro de la base de datos.

(Blancarte, 2018)

4.16.3 POST

Este método se utiliza para solicitar la creación de un nuevo registro. Su equivalente es realizar un INSERT en la base de datos.

(Blancarte, 2018)

4.16.4 PUT

Este método se usa para actualizar por completo un registro existente. Su equivalente es realizar un UPDATE en la base de datos.

(Blancarte, 2018).

4.16.5 PATCH

Parecido al PUT, este método actualiza un registro pero permite actualizar solo un fragmento del registro. Su equivalente es realizar un UPDATE en la base de datos.

(Blancarte, 2018).

4.16.6 DELETE

Este método permite eliminar un registro existente. Su equivalente es realizar un DELETE en una base de datos.

(Blancarte, 2018).

4.16.7 OPTIONS

Este método se utiliza para describir las opciones de comunicación para el recurso de destino. Se utiliza con CORS para validar si el servidor acepta peticiones de diferentes orígenes.

(Blancarte, 2018).

4.17 Códigos HTTP

Los códigos de estado de respuesta de HTTP indican diferentes acciones, estas se pueden agrupar en 5 categorías.

(Lázaro, 2016).

4.17.1 Respuestas informativas (100–199)

Este tipo de códigos indican una respuesta provisional, el servidor ya recibió la solicitud, pero aún no ha terminado de procesarse.

4.17.2 Respuestas satisfactorias (200–299)

Este tipo de códigos indica que la acción que solicitó el cliente ha sido recibida, entendida, aceptada y procesada correctamente. El código 200 ok es el estándar para respuestas correctas.

4.17.3 Redirecciones (300–399)

Con estos códigos se le da a entender al cliente que debe de tomar una acción adicional para completar el “request”. Por lo general es un redireccionamiento a una lista de enlaces que el cliente escoge para continuar.

4.17.4 Errores de los clientes (400–499)

Estos códigos contienen todos los tipos de errores que puede ocasionar el cliente, sirven para notificar al cliente qué error cometió.

4.17.5 Errores de los servidores (500–599)

Cuando el servidor falla con una solicitud que es válida es porque se tiene un error o realmente es incapaz de procesar el “request”. Estos códigos le sirven al desarrollador para empezar a entender que problema está ocurriendo.

4.18 Postman

Esta es una herramienta que se utiliza para hacer testing de API REST sobre todo, pero también permite otras funcionalidades.

(Cuervo, 2019)

Algunas características son:

- Permite crear y enviar peticiones http a servicios REST mediante una interface gráfica. Además, que pueden ser guardadas y editadas para seguir siendo usadas más tarde.
- Genera documentación basada en las API y colecciones que hemos creado en la herramienta.
- Entorno Colaborativo, permite compartir las API para un equipo entre varias personas. Para ello se apoya en una herramienta colaborativa en Cloud.
- Establecer variables, con Postman podemos crear variables locales y globales que posteriormente utilicemos dentro de nuestras invocaciones o pruebas.
- Soporta Ciclo Vida API management, desde Postman podemos gestionar el ciclo de vida del API Management, desde la conceptualización del API, la definición del API, el desarrollo del API y la monitorización y mantenimiento del API.

4.19 Plataformas en la nube

4.19.1 Google Cloud

Google cloud es una plataforma que brinda servicios en la nube para poder ofrecer facilidades en lo que respecta a la creación de máquinas VIRTUALES, BIG Data y apps as a service, esta plataforma ofrece poder utilizar los servicios que tienen por 90 días o hasta que se acaben los \$200 que dan de saldo para la prueba de este.

4.19.2 Amazon Web Services

AWS es una plataforma la cual está completamente en la nube la cual fue de las pioneras en ofrecer toda clase de servidores y servicios dedicados para cualquier persona o público, es de las más utilizadas ya que ha logrado tener una base sólida por su gran gestión y soporte brindado.

4.20 Machine learning

4.20.1 Catboost

Es un algoritmo de aprendizaje automático el cual se basa en la potenciación del gradiente para poder llevar a cabo las predicciones, este algoritmo se puede encontrar tanto en Python como en R para poder ser llevado a cabo es necesario exportarlo, sus puntos fuertes son que puede saber solo cuando hay datos categóricos y cuando no para poder así ahorrar el trabajo de hacerlo manual.

4.20.2 Random Forest

Es un algoritmo de aprendizaje automático el cual se basa en la división de la data en distintos conjuntos en los cuales ninguna rama tiene el conocimiento de que datos tiene cada rama del árbol, esto lo que hace es que cada rama realice distintos entrenamientos para ir eliminando las distintas iteraciones que se van realizando y las que son buenas son elegidas para seguir siendo iteradas (Martínez, 2020).

4.21 Data sintética

4.21.1 SMOTE

Synthetic Minority Over-sampling es una técnica la cual se basa en generar información por medio de los elementos de la clase más pequeña para poder igualarla y crear más información útil para que los algoritmos estén balanceados, para ello utiliza los vecinos más cercanos y por ese medio es capaz de aumentar el volumen de data a utilizar. (Wong *et al*, 2016)

4.21.2 ADASYN

Oversample using Adaptive Synthetic es una técnica la cual se basa en generar la información por medio de la distribución inteligente de los datos basada en la distribución local de las clases.

4.21.3 DBSMOTE

Density Based Synthetic Minority Over-sampling, Es una técnica la cual se basa en SMOTE, pero en vez de hacerlo con la clase más pequeña, se encarga de generar la información con la información central de cada clase para así poder aumentar las dos clases de la misma manera, cabe destacar que esto no elimina el *oversampling* por completo, pero en este caso es útil ya que lo que queremos es aumentar la información. (Wong *et al*, 2016)

Como parte del proyecto, se integró un módulo el cual utiliza una aplicación nativa instalada en dispositivos móviles para poder detectar posibles percances vehiculares mientras está en segundo plano.

4.22 Aplicación de dispositivos inteligentes en detección de percances vehiculares

Con los avances tecnológicos se crean dispositivos que logran realizar tareas más complejas de manera más eficiente. Para ellos, realizar llamadas y contar con servicio de mensajería ya es algo básico que incluso la mayoría de los dispositivos de gama baja ya cuentan con ello. Una de las mejoras que más ayudaron en el avance de los dispositivos fue la inclusión de los acelerómetros y giroscopios a su sistema. Esto permitió que se pudieran crear aplicaciones para analizar la aceleración e inclinación que sufrían los objetos.

Una de las ventajas que más vale la pena resaltar de las aplicaciones en dispositivos móviles contra los posibles sistemas convencionales intrínsecos del vehículo, es que son independientes del automóvil. Y a pesar de esto, son capaces de proveer información acerca de la conducción del usuario, e incluso informar de accidentes. En los cuales, algunos casos pueden facilitar audios a los servicios de emergencia.

Como se mencionó anteriormente, la característica principal para detectar un posible percance vehicular es la aceleración experimentada durante este. En los sistemas dentro de los vehículos, los sensores están puestos en la parte delantera de este y son los mismos que se utilizan para poder activar los air-bags en caso de que sea necesario. Sin embargo, en la posición de los pasajeros la aceleración experimentada es mucho menor, pero sigue siendo lo suficiente para superar los 4G (Salim *et al.*, 2015). Lo cual puede diferenciarlo de una simple caída de celular o freno brusco.

Si bien, los choques con velocidad menor a 24 km/hr no suelen superar los 4G de aceleración y podrían no ser detectados, debido a esto algunos proyectos han optado por también utilizar el

micrófono del celular para detectar cuando una saturación podría significar un choque. Aunque también se debe tomar en cuenta, que en ese caso el teléfono no podría ser expuesto a música fuerte o ruidos de muchos decibeles dentro del vehículo. Lo que podría imposibilitar el uso del teléfono durante la conducción (Al-Alhi *et al*, 2014).

Además, otra técnica que podría ayudar a reducir la cantidad de falsos positivos podría ser la implementación de Dynamic Time Warping. El cuál es un algoritmo que permite comparar la similaridad o calcular la distancia entre dos listas o series de tiempo de diferente tamaño El cual puede ser de ayuda para comparar los decibeles que recibe el teléfono o bien para comparar las velocidades a las que viaja el usuario mientras tiene en uso la aplicación (Dougherty *et al*, 2016). También se pueden utilizar algoritmos de deep learning para poder clasificar los sonidos del entorno.

Si bien el uso del micrófono del dispositivo puede traer beneficios para la detección de posibles percances vehiculares. También existen desventajas. Ya que dependiendo del método que se utilice los resultados pueden llegar a no ser tan exactos. Debido a que si solo se mide el nivel al que llegan los sonidos, un choque puede llegar a tener el mismo nivel de otros sonidos fuertes. Mientras que, si se utiliza un algoritmo de clasificación, no importarán los decibelios del sonido captado. Ya que estos algoritmos pueden llegar a ser capaces de clasificar desde el ladrido de un perro hasta el claxon de un vehículo.

Para poder definir, clasificar y determinar un percance vehicular, es importante entender que es lo que sucede durante este evento. Por lo que el siguiente paso para la comprensión del proyecto es entender qué cálculos se realizan detrás de la pantalla táctil.

4.23 Leyes físicas en un accidente vehicular

El choque se define como una colisión en la cual pueden ser afectados de dos a más cuerpos en movimiento. El factor clave en un choque físico o mecánico es la aceleración o desaceleración repentina de un cuerpo en movimiento. Por conveniencia, suelen medirse en múltiplos de la aceleración de gravedad que tiene un valor de 9.89665 m/s^2 . Esta es la aceleración experimentada por todos los cuerpos sobre la superficie de la tierra a nivel del mar.

Para lograr medir las aceleraciones que sufre un cuerpo en el momento de un percance vehicular, es necesario rastrear su velocidad en todo momento del trayecto para poder obtener resultados más precisos. Ya que las aceleraciones pueden llegar a ocurrir en fracciones de segundo. Debido a esto, distintos dispositivos, tanto intrínsecos del vehículo como extrínsecos utilizan distintos sensores para poder determinar el momento en el que sucede un percance. Algunos aparatos

especializados utilizan acelerómetros para poder rastrear las aceleraciones del cuerpo. Al igual que utilizan giroscopios, para poder determinar la posición en la que se encuentra el vehículo (Sneha et al., 2013).

Como se mencionó anteriormente, la clave para determinar la magnitud de un choque es la aceleración que sufren los cuerpos. De esta manera podríamos calcular la magnitud de un auto que viaja a 105 km/h y frena en la mitad de un segundo. El primer paso es convertir su velocidad a m/s. Lo cual es un aproximado 29.17 m/s. El segundo paso sería calcular la desaceleración que sufrió $\frac{0\frac{m}{s} - 29.17\frac{m}{s}}{0.5s}$ dando como resultado $-59.34 \frac{m}{s^2}$. Lo cual en términos de gravedades sería un equivalente a -5.99 g. Para tener en retrospectiva de lo mucho que puede cambiar el valor de la desaceleración en caso de que el accidente suceda más rápido, considere que en caso de que el accidente hubiese ocurrido en un lapso de 0.2 segundos en lugar de 0.5 segundos, la magnitud del accidente equivaldría a un aproximado de -15 g en lugar de -5 g.

Si bien esto facilita la detección de choques a alta velocidad, existe una problemática con los accidentes a baja velocidad. Ya que muchas veces sus magnitudes no suelen superar siquiera los 4 g. Y en caso de que un dispositivo extrínseco tome estos valores como posibles percances vehiculares, puede resultar en varios falsos positivos. Ya que estas aceleraciones pueden ser causadas por muchos factores, como pueden serlo un lanzamiento del aparato, una caída de este, un deslizamiento u otras situaciones que puedan mover con fuerza el artefacto (Silva, 2017).

Es importante comprender cómo se absorbe la energía existente durante un choque vehicular. Esto ha sido estudiado por distintas compañías automotrices para poder asegurar a sus clientes un nivel de seguridad satisfactorio. Han realizado distintos estudios para poder comprender qué sucede detrás del evento donde dos vehículos chocan a una gran velocidad, y cómo estos absorben la energía de la colisión. Los vehículos comerciales actuales no son capaces de absorber toda la energía mecánica que existe en un choque a gran velocidad. Debido a esto, los crash test tienen una velocidad máxima de 64 km/h. Y han determinado que un choque entre dos objetos a gran velocidad (ya sea que los dos estén en movimiento o al menos uno de ellos esté en movimiento), la colisión empieza siendo un choque inelástico mientras el vehículo absorbe la energía mecánica creada pero casi de inmediato se transforma en un choque elástico (el objeto, en este caso el vehículo, rebota) debido a que el vehículo no logra absorber más energía. Si bien, ambos tipos de choques se pueden encontrar en una colisión, el factor clave seguirá siendo la desaceleración que sufren los objetos para poder determinar la magnitud del choque.

(ennomotive, 2019)

4.24 Sonidos durante un accidente vehicular

Algunos sistemas han tomado como variable crucial los decibeles que logran capturar de sonidos del entorno a través de micrófonos. Uno de estos fue wreckwatch el cual es un sistema que asegura resolver la problemática de falsos positivos con choques a baja velocidad. Para esto se tomó en cuenta los decibeles que pueden llegar a alcanzar algunos sucesos durante un percance vehicular, tales como el sonido de un vehículo chocando, la explosión de las air-bags o incluso el de vidrios rompiéndose.

El algoritmo que este sistema establece nos dice que una detección de accidente puede ser provocada por una de dos situaciones: (1) un evento de alta aceleración (o desaceleración) y un evento de sonido de alto decibelios se registran mientras el vehículo se mueve por encima del umbral de la velocidad mínima a la que el sujeto debe transportarse para activar el sistema cuando esta inactivo o (2) un evento de aceleración (o desaceleración) y un evento de sonido de alto decibelios se registran antes de que se desactive el sistema (Hamilton, 2011).

Para lograr que este sistema funcione, Wreckwatch decidió utilizar niveles de decibelios lo suficientemente altos para que en caso de que el usuario se encontrará escuchando música dentro de su vehículo o estuviera haciendo ruidos dentro de este, no afectará la efectividad del sistema. Tomaron en cuenta el nivel de decibelios que podrían alcanzar distintos eventos comunes que suelen ocurrir en los accidentes, tales como el sonido del impacto del vehículo, el sonido del claxon, el sonido que realizan las bolsas de aire al salir, he incluso el rechinar de las llantas. Debido a esto, establecieron que como mínimo los eventos debían alcanzar los 150 decibelios para poder tomar un evento como posible percance vehicular. Ya que el choque de un vehículo puede alcanzar como mínimo los 145 decibelios mientras que la explosión de las bolsas de aire puede alcanzar hasta los 170 decibelios (Valero, 2017).

Si bien recolectar el nivel de decibelios que alcanzan los sonidos del entorno es buen inicio para reducir el número de falsos positivos no basta con solo eso. Wreckwatch también establece que este nivel de decibelios debe mantenerse por un rango de tiempo. Además, establece que solo medir el nivel de decibelios no es suficiente por sí solo para detectar un posible percance vehicular (Hamilton, 2011).

V. Marco metodológico

5.1 Planificación

El proceso por seguir para ejecutar el proyecto planteado empezará con diseñar la base de datos, con ayuda de todo el equipo de desarrollo y el asesor Franz Haidacher, para establecer qué tablas para la base de datos son requeridas y cuáles atributos tendrá cada tabla para que se guarde información relevante, además de diseñar la aplicación que se planea realizar. Esta etapa es sumamente importante debido a que uno de los objetivos del proyecto es utilizar el estándar IRTAD, estándar internacional para recopilación de datos de hechos de tránsito, entonces la investigación de este estándar y el diseño de la base de datos es primordial en el proyecto.

Simultáneamente, a la investigación del estándar IRTAD, se investigará sobre qué tecnología es mejor para crear una aplicación en la cual sea posible captar la información que ingrese un usuario de forma fácil. Se discute si es mejor implementar una aplicación nativa para móvil o bien, utilizar un framework para implementar una PWA (Progressive Web App) para unificar la programación de todos los sistemas operativos. Para la tecnología PWA, se tienen dos alternativas, Angular y Vue.

5.2 Investigación

Se realizó una investigación exhaustiva sobre el estándar IRTAD, International Traffic Safety Data and Analysis Group, por sus siglas en inglés, con el propósito de utilizar este estándar de datos para implementarlo y mejorar la forma en que se recopilan y almacenan los datos. Se comparó con bases de datos actuales publicadas por el Instituto Nacional de Estadística (INE), y se notaron problemas causados por no realizar una planificación de la base de datos y por no crear o seguir una estructura para los datos.

Con ayuda del estándar IRTAD, no solo Guatemala será capaz de guardar correctamente los datos, sino también estudiar de mejor manera los accidentes vehiculares ocurridos en el país y contribuir en la captación de datos para el mundo. Puesto que actualmente, las bases de datos del INE omiten información importante que el IRTAD sí toma en cuenta.

5.3 Prototipado

Se diseñaron y elaboraron dos prototipos, un wireframe de calidad baja y un prototipo de mejor calidad para verificar la funcionalidad de la aplicación. El prototipo de baja calidad se realizó en Mobile App Wireframe, en la cual solo muestra una idea de cómo se vería el diseño de la aplicación al momento de programarla en un futuro, no posee datos ni funcionalidad.

Cabe mencionar que este prototipo fue probado con usuarios no finales, es decir, usuarios que no conocen mucho sobre el tema de accidentes de tránsito. El propósito de esto, fue obtener retroalimentación sobre cómo se ve la aplicación estéticamente, parte de cómo estarían distribuidos los campos en las distintas pantallas, versión inicial de la aplicación, entre otras cosas.

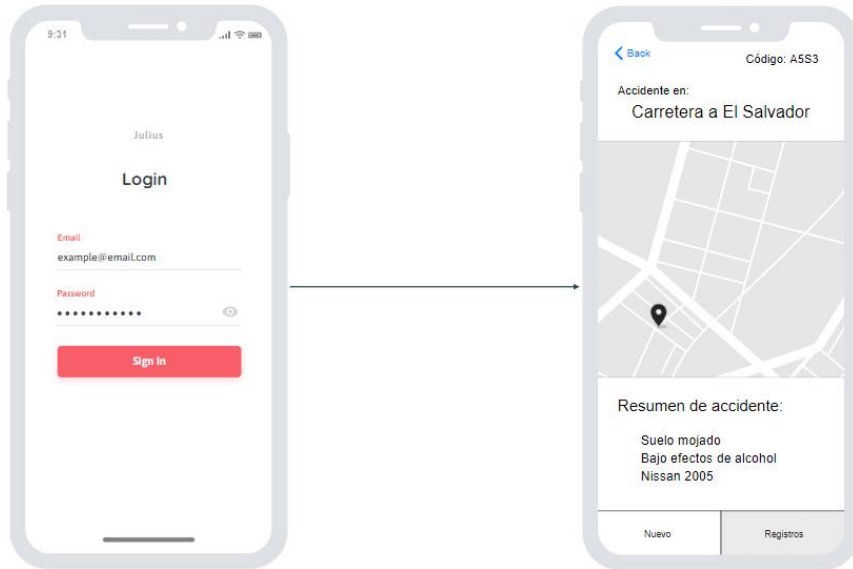


Figura 7. Bosquejo uno: app

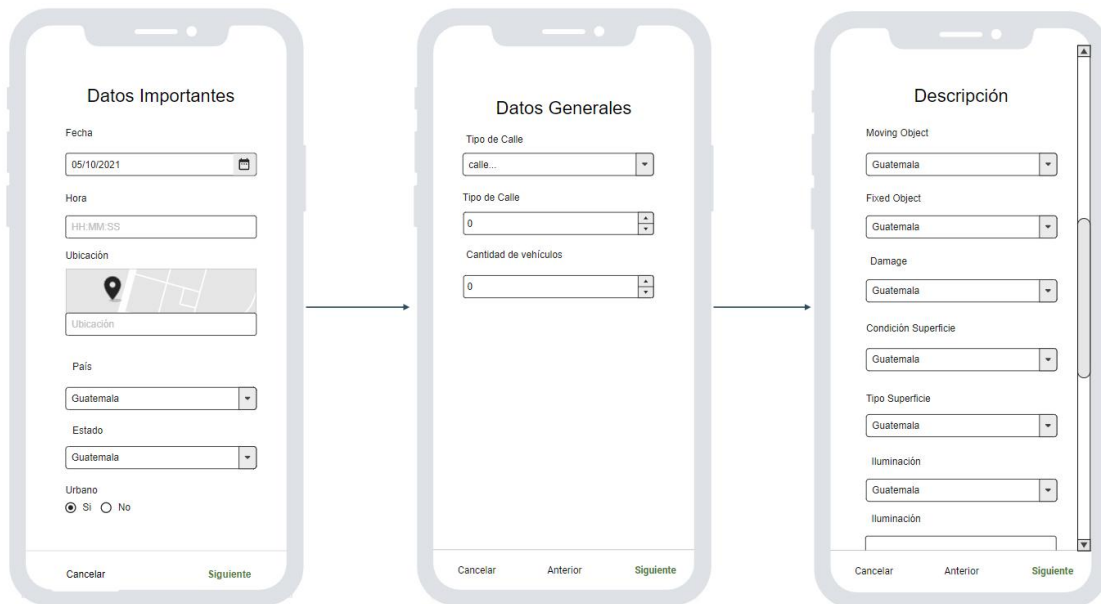


Figura 8. Bosquejo dos: app

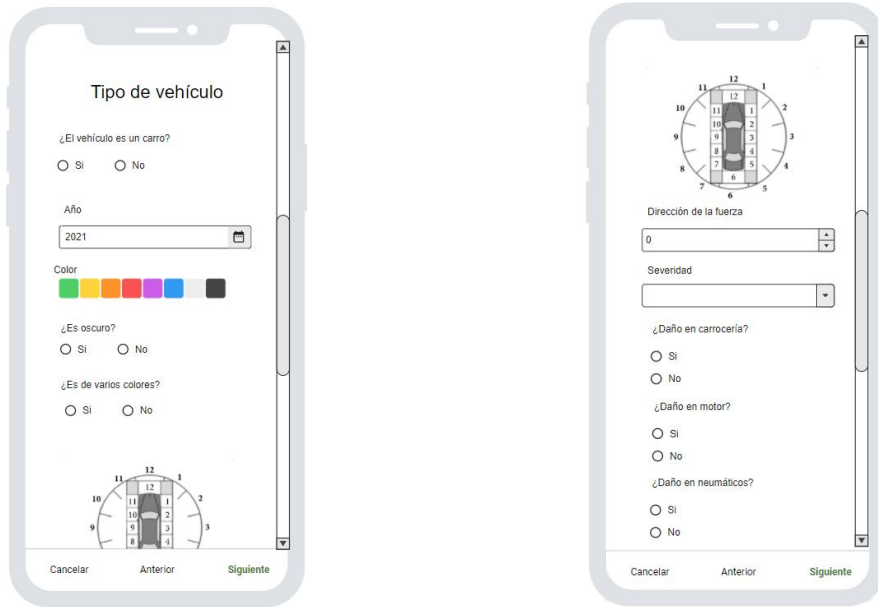


Figura 9. Bosquejo tres: app

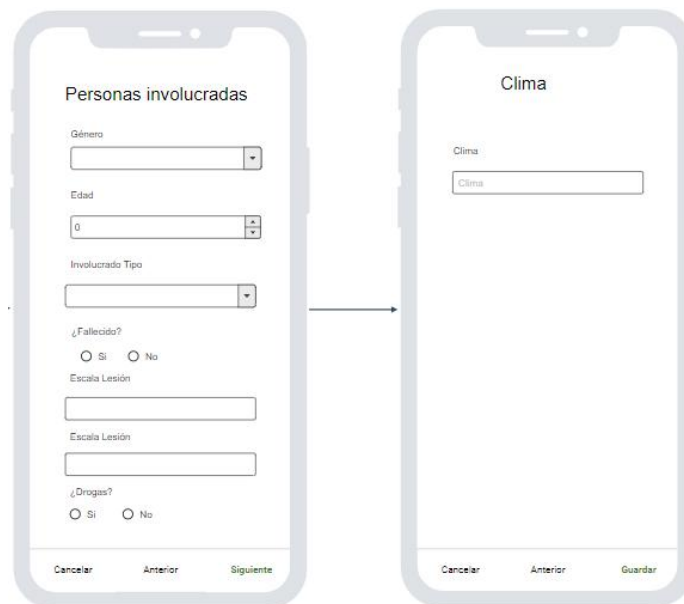


Figura 10. Bosquejo cuatro: app

Por otro lado, se elaboró un prototipo con más funcionalidad, este se realizó en AppSheet de Google. La elaboración de este prototipo permite una conexión a una base de datos de PostgreSQL en Google Cloud que es capaz de guardar información obtenida en el formulario de la aplicación, y también, obtener la información para mostrarla en pantalla, para que el usuario pueda editar registros. También, este prototipo es el que se utilizará para crear encuestas y obtener retroalimentación de usuarios.

Este segundo prototipo, es la versión “semi-programada” del prototipo número 1. En este prototipo se busca tener un poco más de funcionalidad. La funcionalidad que se implementó, fue claramente la misma organización de campos como el primer prototipo, y además, verificar la correcta funcionalidad de la base de datos para guardar información y poder consultarla en otra etapa y realizar análisis sobre esta. También, para empezar a analizar cómo se trabajará la parte de múltiples usuarios guardando información sobre el mismo accidente.

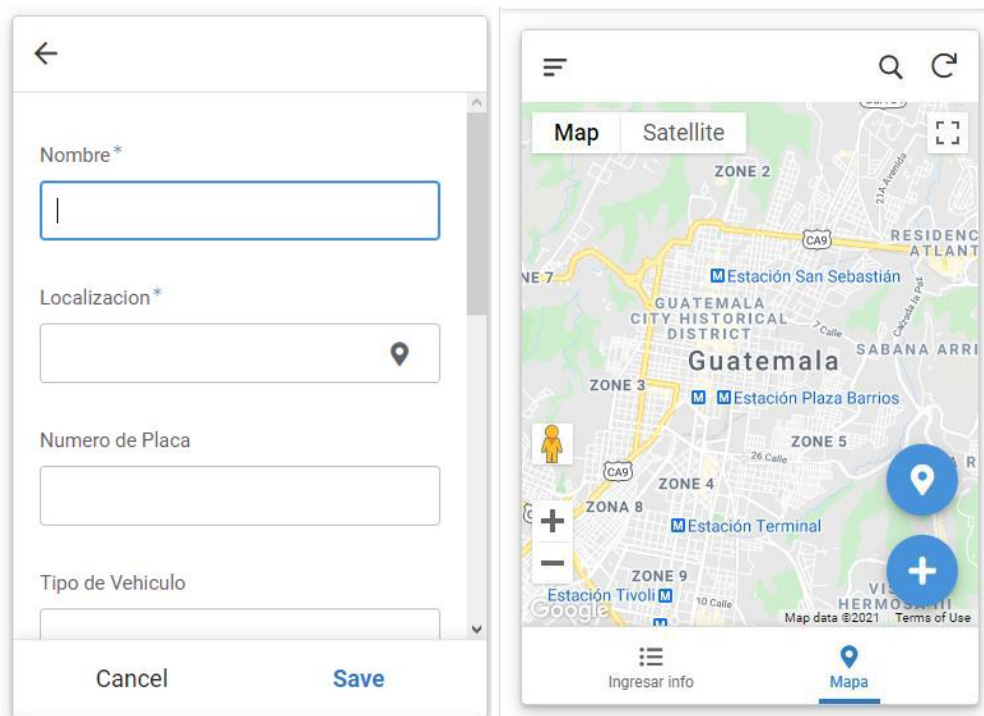


Figura 11. Bosquejo cinco app

5.4 Evaluación de tecnologías

Se realizó una investigación para determinar qué herramienta será la más útil para realizar la aplicación programada. Para la toma de decisión de qué framework utilizar, se tomó en cuenta la curva de aprendizaje, facilidad para diseño visual de la aplicación, compatibilidad con los diferentes sistemas operativos, experiencia previa en los frameworks. Primero se comparó una aplicación nativa contra una progressive web app. En la cual, se tomó la decisión de utilizar una PWA.

Native App	PWA
Subir una App al Store indica más confianza y es más sencillo utilizar una app, que una URL.	Es más barato que una Native App, debido a que no hay que hacer una aplicación para cada dispositivo. Sin importar su OS.
Uno tiene la opción de implementar muchas medidas de seguridad para la app, mientras que para PWA solo está el certificado para HTTPS	Es más fácil de construir y actualizar.
Tiene mejor desempeño y es más poderoso, aunque la PWA es más rápida para cargar.	No hay que subir la APP a ninguna Store para descargarla. Solo se necesita un browser y la URL de la aplicación.
Algunas aplicaciones nativas no necesitan conexión a internet para funcionar (aunque para el proyecto Julius necesitamos información entonces sí es requerido conexión wifi para mandar y guardar información)	PWA puede instalarse de forma sencilla y es mucho más ligera que una App normal
Para el dispositivo hay muchas más funcionalidades, como acceso a otras aplicaciones, permisos de utilizar recursos del dispositivo y demás	PWA si pueden ser indexadas y ser buscadas.

Tabla 6. Comparación de aplicación nativa y PWA.

Asimismo, se decidió sobre cuál framework para desarrollar una PWA se utilizará, se comparó entre utilizar Vue y Angular. Finalmente, se decidió usar **Angular**.

Las razones por las cuales se eligió Angular como el framework indicado para la aplicación por múltiples razones. La primera, siendo la ventaja principal, es que el desarrollo de la aplicación se realiza al mismo tiempo con diseño visual para una aplicación de computadora y una aplicación de un teléfono móvil, por lo tanto, el conocimiento de múltiples lenguajes para cada sistema operativo no es requerido porque con conocer sobre desarrollo de aplicaciones web, se está desarrollando completamente para todos los sistemas operativos.

También, se tiene experiencia con el framework de Angular, ya que ya se ha utilizado este framework en varios proyectos desarrollados previamente, y se utiliza este framework en el ámbito profesional de su servidor. Por último, permite reutilizar mucho del código que se escribe para hacerlo más legible, estandarizado y optimizado, por ejemplo, en la creación de un formulario que contiene más de un campo de selección de opciones, se puede utilizar el mismo componente, con diferentes opciones cada uno y se comporta de manera adecuada y correcta.

5.5 Diseño de bases de datos y creación de esta

Esta fase comprende el diseño, diagramas y tecnologías a utilizar con los datos obtenidos de la fase anterior para la posterior creación de las tablas y sus respectivas funciones para poder ser utilizadas por el backend, para el diseño de los diagramas se utilizará la herramienta dbdiagram la cual tiene un diseño práctico y puede generar el código para crear las tablas solo con el diseño para distintas bases de datos como MySQL y PostgreSQL.

Para la retroalimentación de esta fase se hablará con el asesor para consultar su experiencia con los gestores de bases de datos y ver si el diseño de la base de datos cumple las expectativas respecto a la magnitud del proyecto.

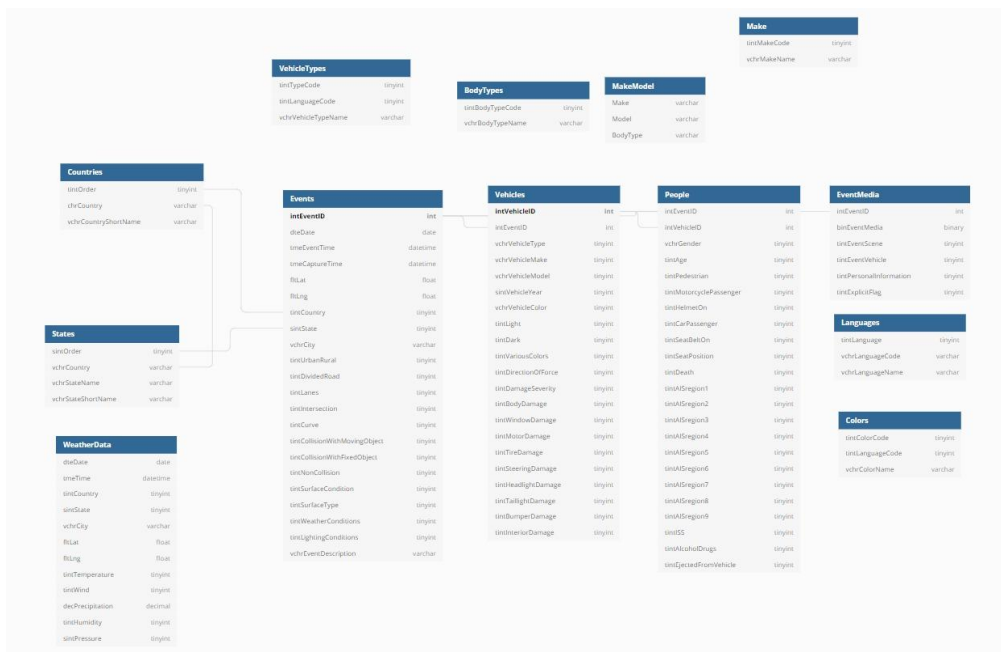


Figura 12. Diseño de base de datos

5.6 Implementación

La aplicación progressive web app, ya que utilizará un framework para desarrollo de aplicaciones web, esta deberá ser construida en HTML y CSS para todo el diseño que se construirá desde cero, con funcionalidades desarrolladas en Javascript/Typescript. La aplicación contendrá tres pantallas principales.

1. **Login:** La pantalla de inicio de sesión solamente solicitará el usuario y contraseña para registrar al usuario y saber qué usuario ha es el que va a agregar, editar y/o eliminar información.
2. **Menú principal:** En esta pantalla se mostrarán los accidentes registrados durante el día, pero también contará con un buscador que permitirá acceder a accidentes previos al día que se busca. De este modo, se podrá llenar la información por más de un usuario al mismo tiempo.
3. **Formulario para recopilación de información:** Esta pantalla es la más importante dentro de la aplicación porque es la que contendrá la implementación del estándar IRTAD mencionado anteriormente. Esta pantalla se divide en 3 partes, datos generales, descripción del accidente y datos de vehículos y personas involucradas. Cada una de estas, buscará incluir conceptos útiles de UX para mejorar la interacción del usuario con la aplicación, y permitir que los datos capturados sean estructurados y guardados correctamente en su base de datos correspondiente.
4. **Consultar un accidente:** En esta pantalla permite visualizar la información ingresada por el usuario luego de haber creado un accidente previamente. Entonces, a partir de un id de un accidente creado, se obtiene su información, y se desplegará al usuario de forma ordenada y simple por si se desea obtener algún dato del accidente.
5. **Visualizar accidentes:** Dentro de esta pantalla, se muestran los accidentes como resumen general de lo que se ha registrado en toda la vida de la aplicación. Se permite buscar accidentes en un intervalo de fechas, y a través de una palabra clave se filtran los accidentes para encontrar el que se está buscando. En esta tabla de resumen se muestra el id como código del accidente, la fecha en que se registró el accidente, número de vehículos involucrados y número de personas involucradas. Se permite también editar y consultar la información de alguno de estos accidentes desplegados.
6. **Perfil:** La pantalla de perfil nos permitirá mostrar la información del usuario autenticado, y además, cuenta con la funcionalidad de cambiar el idioma, actualmente, entre español, inglés y francés. Al seleccionar un idioma, esta información se guarda para persistir la información por si vuelve a iniciar sesión, y además, toda la aplicación cambia el idioma instantáneamente.

5.7 Desarrollo

Para el inicio del desarrollo de este proyecto, el equipo encargado de desarrollar el frontend de la aplicación se dividió en módulos para codificar el proyecto en partes más pequeñas y simples. El módulo desarrollado en este documento fue sobre visualización de la información entendible y correcta sin importar el dispositivo en el que se esté utilizando la aplicación.

Para cumplir con el desarrollo del módulo, se implementó cuatro pantallas que tienen que ver con esto. El menú principal despliega la información de todos los accidentes vehiculares registrados en el día actual. Luego, se desarrolló la visualización de un accidente específico, para este caso era necesario que un accidente ya tuviera información para decidir cómo se habría de mostrar el accidente. Para alimentar la base de datos se corrió un comando, llamado seeder, que llena la base de datos con información falsa y estática para tener registros de prueba. También, ya existía una versión inicial del formulario que nos permitió crear registros con información ingresada por el usuario. Como última opción, se utilizó Postman que llama a un endpoint al API desarrollado para almacenar algún accidente.

Seguido de esto, se desarrolló la pantalla para visualizar un histórico de accidentes, dado un intervalo de fechas. Inicialmente, este intervalo de fechas está para un rango de una semana, comprendida entre siete días antes del día actual hasta el día actual. El usuario puede modificar el intervalo de fechas para obtener los accidentes ocurridos dentro de este intervalo. Al igual que en el menú principal, esta pantalla permite filtrar los accidentes y buscarlos dado una palabra o un conjunto de palabras, que permita encontrar algún accidente.

Después, se programó el perfil. Esta pantalla es de las más sencillas pues su contenido solamente es información del usuario autenticado en la aplicación, y un selector de idiomas. Actualmente solo se cuenta con el idioma español, inglés y francés. Una vez seleccionado un idioma, el usuario navega a través de la aplicación con este idioma seleccionado.

También, para la pantalla 1 del formulario, se trabajó de dos formas: la primera fue la llenada de datos iniciales, esta fue planteada así, para que el usuario que esté creando el evento del accidente por primera vez no tenga errores en información tan importante como la fecha y hora del accidente, la ubicación exacta en la que se encuentra con coordenadas de latitud y altitud, así como el país y el estado en el que se encuentra. Estos últimos dos son recomendación de la aplicación sin embargo el usuario aún puede cambiarlo, en cambio, las coordenadas y la hora si es un dato fijo. Los datos de la fecha y hora simplemente son las actuales del dispositivo sin embargo gracias a una API de geolocalización es que se obtienen los datos del país del usuario.

Posterior a esto con nuestra base de datos y las tablas creadas y llenadas por “backend”, se tienen los departamentos o estados de diferentes países, por lo que consultamos para mostrar las opciones en pantalla. Con las coordenadas obtenidas de la geolocalización se muestra un mapa de Google para enseñar la ubicación exacta en el mapa. Por último, se tiene la pregunta que si la calle es o no urbana, solo se guarda el valor. Cabe destacar que todos los datos se van guardando en un diccionario que al finalizar se convierte en un JSON para el mejor manejo de la información. Este diccionario tiene como llave el nombre de cada componente del formulario facilitando la forma de extraer la información obtenida totalmente. Esta información aún no se envía ya que la siguiente pantalla todavía tiene información sobre la creación de un accidente entonces se discutió que era mejor esperar a que el usuario llenará las primeras dos pantallas antes que hacer el “request” POST para la creación del evento.

En la pantalla dos del formulario se tienen diferentes inputs sobre la descripción del evento desde el estado de la calle hasta los objetos involucrados. Todas las opciones que pueden ser escogidas en estos inputs son obtenidas de diferentes JSON creados previamente por “frontend” los cuales pueden ser traducidos si la aplicación se encuentra en otro idioma.

Luego de llenar todos los campos de las primeras dos pantallas ya se puede seleccionar el botón de siguiente el cual inicialmente está bloqueado hasta que se complete lo requerido. Este botón realiza el llamado al servicio encargado de hacer la conexión con el “request” de crear un evento en la base de datos, este es un POST recopilando toda la información que el usuario llenó así como el ID del usuario que lo creó. Los servicios que están empleando llamadas al API lo realizan con Promesas de JavaScript debido a que las respuestas que se necesitan en cada uno de los “endpoints” son respuestas únicas, en ninguno de los casos se estará recibiendo múltiples respuestas continuas de un solo “request” por lo que no se hará uso de Observables de JavaScript. Al crearse correctamente devuelve el ID del evento creado para que las pantallas que siguen se enlazan a este evento.

La tercera pantalla tiene 2 propósitos, obtener la cantidad de vehículos y personas involucradas en el evento para que sean creados en la base de datos y de igual forma usar esa cantidad para crear la cantidad de componentes necesarios del formulario para que se llenen en las pantallas que siguen. Al presionar el botón siguiente se llama el servicio correspondiente de la llamada el cual es un POST de la creación correspondiente de la cantidad de personas y de vehículos en la base de datos.

La cuarta pantalla está relacionada con la información de los vehículos involucrados. Esta genera la cantidad de formularios necesarios para cada vehículo dependiendo de la información

ingresada anteriormente. Este formulario tiene algunos inputs que son recolectados de la base de datos, ya que es información un poco más específica y cambiante, como lo son las marcas de los automóviles, sus modelos, sus colores, etc. Luego de esto ya solo vienen campos de sí o no, los cuales siguen siendo recolectados de la misma forma que todo lo demás, en un JSON. Cada formulario de los datos de un vehículo tiene su propio botón de guardar el cual llama el servicio de actualización de datos del vehículo. El “endpoint” que realiza el PUT lo facilita para más adelante cuando se tenga que editar los datos del vehículo, se realiza de esta forma ya que previamente se crearon los espacios correspondientes en la tabla de vehículos.

En la quinta y última pantalla del formulario se encuentran los datos de las personas involucradas en el accidente. Estas de igual forma que los vehículos tienen la cantidad de formularios correspondientes a cada persona mencionada previamente. Al inicio del formulario hay un campo para que el usuario seleccione a qué vehículo pertenece esa persona, o en el otro caso si la persona es un peatón. Estos datos se despliegan según la cantidad de vehículos reales agregados a la base de datos ya que se obtienen a través de una consulta sobre el mismo evento. Luego vienen todo tipos de datos sobre la persona, aquí es donde se aplica el Índice de Gravedad de Lesión y se calcula según los resultados de lo que se ingresó. Finalmente se selecciona el botón de guardar el cual llama el servicio de actualización de datos de la persona involucrada, el cual hace un llamado al API y de igual forma que con los vehículos realizar un POT para actualizar la información de la persona facilitando el trabajo más adelante cuando se requiera de editar.

Mientras se hacía esta toma de decisiones para el trabajo de “frontend”, el equipo de “backend” se encargó de crear la base de datos inicial en PostgreSQL con las tablas correspondientes para tener la estructura inicial de cómo iba a ser el guardado de información. Gracias a que se les dedicó tiempo a los prototipos iniciales y ya con autorización de nuestro asesor se procedió a iniciar con la aplicación.

Como la base de datos era relacional, ya se tenía la estructura de cómo se iba a obtener la información. “Backend” entregó el script de la creación y migración inicial de datos con algunos “endpoints” en su primera fase. Cómo la aplicación en general se separó en pantallas como previamente se había trabajado en el prototipo, finalmente para el formulario quedó de esta forma:

Pantalla 1 Datos importantes	Pantalla 2 Descripción	Pantalla 3 Datos generales
<ul style="list-style-type: none"> • Fecha y Hora • Ubicación • País • Estado • ¿Es Urbano? 	<ul style="list-style-type: none"> • Tipo de calle • Número de carril • Objeto en movimiento • Colisión con objeto fijo • Sin colisión • Condición de superficie • Tipo de superficie • Condición lumínica • Descripción 	<ul style="list-style-type: none"> • Cantidad de vehículos • Cantidad de personas

Figura 13. Inputs en pantallas 1, 2 y 3.

Pantalla 4 Tipo de vehículo	Pantalla 5 Personas involucradas
<ul style="list-style-type: none"> • Tipo de vehículo • Año • Color • ¿Es oscuro? • ¿Es de varios colores? • Dirección de la fuerza • Severidad • ¿Daño en carrocería? • ¿Daños en ventanas? • ¿Daños en el motor? • ¿Daño en las llantas? • ¿Daño en dirección? • ¿Daño en la luz delantera? • ¿Daño en la luz trasera? • ¿Daño en el bumper? • ¿Daño en el interior? 	<ul style="list-style-type: none"> • Número de vehículo • Género • Edad • Tipo de persona involucrada • ¿Con cinturón? • Asiento • ¿Falleció? • Lesión craneal • Lesión facial • Lesión de cuello • Lesión de tórax • Lesión de abdomen • Lesión de columnar vertebral • Lesión de la extremidad superior • Lesión de la extremidad inferior • Puntuación gravedad lesión • ¿Alcohol o drogas • ¿Eyectado del vehículo?

Figura 14. Inputs en pantallas 4 y 5

Una vez quedó claro qué datos serán enviados y recibidos, se plantearon diferentes “endpoints” para que la aplicación recibiera la información a través de Servicios de Angular. Cada servicio tiene una funcionalidad en específico.

En esta fase también se lleva a cabo la creación de la lógica del proyecto, lo cual comprende la conexión con base de datos y los *endpoints* para su unión con el *frontend*, para esto se utilizará la tecnología que se decida más conveniente en la primera fase, el plan a tomar será trabajar de la mano con el módulo de *frontend* para poder ir realizando los *endpoints* y las pantallas del proyecto, así se podrá tener mejor control del desarrollo y se podrá probar de una manera más eficiente el correcto funcionamiento en conjunto.

Para la retroalimentación de esta fase se trabajará en conjunto con el módulo de *frontend* para poder realizar pruebas con personas ajenas al proyecto y saber qué les parece la implementación.

5.8 Testing

Durante el desarrollo del proyecto, se realizaban avances en cada sprint semanal, el cual consistía en implementar más funcionalidades del proyecto, y en mostrar las correcciones planteadas en el sprint anterior. En cada sprint se presentaban los avances a usuario cercanos a los desarrolladores y al asesor del proyecto Franz Haidacher, para obtener retroalimentación de cómo iba el proyecto y encontrar posibles fallas para mejorarlas la siguiente semana.

En la etapa de pruebas del proyecto, se programó una visita con PROVIAL. Durante esta visita se expusieron todas las funcionalidades de la aplicación, en la cual se explicó paso por paso lo que debe hacer el usuario para utilizar llevar a cabo el correcto uso de este proyecto. En esta reunión asistió la directora de PROVIAL Jessica García, el analista de hechos de tránsito Jacob Caxaj, el vocero de PROVIAL Juan Carlos Aquino, y otras personas interesadas en el uso de la aplicación como uso para analizar información.

El propósito de la reunión era poder presentarles al equipo de PROVIAL, parte de nuestro público objetivo, para obtener retroalimentación acerca de la usabilidad, diseño, funcionalidad y utilidad de la aplicación desarrollada. Se contactó con una colaboradora de esta institución para acordar una fecha para la respectiva reunión. En la reunión, se llevó la aplicación en una computadora, y a través de esta se realizó la presentación en la cual, como se mencionó anteriormente, se hizo un recorrido a través de todo el proyecto programado. De esta forma, el equipo de PROVIAL podían conocer una posible futura herramienta para su uso como institución, compartido con otras instituciones con el mismo objetivo. Al finalizar la presentación, la directora de PROVIAL brindó un recorrido al equipo del proyecto, en las instalaciones de la institución con el objetivo de ver las áreas de trabajo y el procedimiento que se realiza en el momento de algún accidente de tránsito.



Figura 15. Charla con personal de Provial

5.9 Desarrollo de prototipo de data sintética

Esta fase comprende la creación de un prototipo de programa el cual pueda generar información para suplir las necesidades de la base de datos, esto basándose en bases de datos parecidas para poder tomar en cuenta cuando es accidente y cuando no, esto también comprende la investigación de las técnicas de generación de data sintética y *data augmentation*, las dos técnicas principales que se utilizaran son:

- SMOTE: *Synthetic Minority Over-sampling* es una técnica la cual se basa en generar información por medio de los elementos de la clase más pequeña para poder igualarla y crear más información útil para que los algoritmos estén balanceados, para ello utiliza los vecinos más cercanos y por ese medio es capaz de aumentar el volumen de data a utilizar. (Wong *et al*, 2016)
- DBSMOTE: *Density Based Synthetic Minority Over-sampling*, Es una técnica la cual se basa en SMOTE, pero en vez de hacerlo con la clase más pequeña, se encarga de generar la información con la información central de cada clase para así poder aumentar las dos clases de la misma manera, cabe destacar que esto no elimina el *oversampling* por completo, pero en este caso es útil ya que lo que queremos es aumentar la información. (Wong *et al*, 2016)

Para la retroalimentación de esta fase se llevará a cabo una reunión con el asesor del proyecto para poder evaluar los resultados de la creación de data, si está en los parámetros de la normalidad y si no se agrega algún tipo de *overfitting* al modelo.

5.10 Prototipos de implementación de algoritmos de predicción

Esta fase comprende la preparación de la información, diseño y programación de los algoritmos de *machine learning* para sus posteriores pruebas y mediciones de resultados. Los cuales se probarán con la información con y sin las técnicas de las fases anteriores para ver si existe un cambio en la eficacia de los modelos dependiendo de la información que sea utilizada para los entrenamientos.

Ya con la información para trabajar, procederemos a desarrollar algoritmos de *machine learning* para poder realizar la predicción de accidentes y en qué lugares suceden más, para ello utilizaremos redes neuronales y algoritmos de clasificación. Para ello utilizaremos algoritmos como TensorFlow-Keras, XGBoost, regresiones lineales, LightGBM y árboles de decisión para después de realizar el preprocesamiento de la información se entrenan los algoritmos para comparar sus resultados y ver cuál de todos es el óptimo para poder realizar esta predicción.

Para la retroalimentación de este módulo se llevará a cabo una reunión con el asesor para discutir los resultados obtenidos y su efectividad fue buena tomando en cuenta los datos con los que se contaron.

5.11 Desarrollo de aplicación nativa para detección de posibles percances vehiculares

Esta fase comprende la creación de un prototipo de aplicación la cual pueda obtener la ubicación aproximada del posible percance, la hora y fecha del evento. Para esto primero se definirá los eventos que serían clasificados como posibles percances vehiculares. Para lograr esto, se utilizarán los acelerómetros y GPS del dispositivo. La lógica detrás de la utilización de ambos sensores es muy similar, ambos se encargarán de medir la velocidad del usuario en tiempo real. En el momento en el cual los sensores rastreen una variación grande en la velocidad en un pequeño intervalo de tiempo se tomará como un posible percance vehicular.

Según estudios publicados en el 2021 por la OMS indica que los atropellos de peatones a una velocidad de entre 50 km/h hasta 60 km/h aumenta hasta un 4.5 las posibilidades de fallecimiento del peatón (OMS, 2021). Mientras que Insurance Institute for Highway Safety (en español, instituto de seguros para la seguridad en las carreteras) registraron que en los accidentes donde los conductores

sobrepasan los 80 km/h tendían a sufrir lesiones severas o incluso la muerte (IIHS-HLDI, 2021).

5.12 Prototipo de aplicación nativa para detección de posibles percances vehiculares

Tomando esta información como base, se realizará un algoritmo que permitirá a la aplicación detectar los cambios de velocidad en un intervalo de tiempo a partir del momento en el que la aplicación detecte que el usuario supera cierta velocidad. Existirán dos casos en el que la aplicación notificará al usuario que detectó un posible percance vehicular y enviará la información a la base de datos de Julius:

1. Posible percance detectado a través del acelerómetro: en el momento en el que el acelerómetro registre un valor muy alto en alguno de sus tres ejes, activará una rutina en la cual verificará que con el GPS que haya estado en movimiento durante los últimos segundos. Esto con el fin de reducir los posibles falsos positivos, que podrían ser desde un simple lanzamiento del dispositivo de una persona a otra hasta una caída desde una gran altura de este.
2. Posible percance detectado a través del GPS: cuando la aplicación registre un cambio de velocidad que represente una magnitud mayor a 4g, la aplicación definirá el evento como posible percance vehicular.

VI: Resultados

La aplicación finalmente fue desarrollada con la tecnología de Progressive Web App, con ayuda del framework Angular para el desarrollo de frontend. El módulo de backend, todo lo relacionado a API y consulta de información hacia la base de datos, se utilizó el lenguaje de programación Python, con apoyo del framework llamado Flask. Estos dos frameworks interactúan entre sí cuando algún componente del lado del módulo de frontend necesite algún tipo de información requerida de la base de datos, es entonces cuando este hace una solicitud de tipo http hacia el módulo de backend para que este último obtenga la información solicitada y esta es la respuesta que se envía y recibe el módulo de frontend.

Un aspecto positivo del desarrollo de la aplicación en el framework de Angular es que permite ser una aplicación reactiva al tamaño de la pantalla del dispositivo en el que se usa esta. Esto es muy importante puesto que el usuario puede decidir el sistema operativo que tenga a su disposición para utilizar la aplicación, pues el usuario puede decidir usarla en una computadora sin importar el tamaño de la pantalla, o en un dispositivo móvil iOS o Android. Esto es posible porque los componentes que se visualizan en pantalla son ordenados y adaptados en tamaño en dependencia a los píxeles de la pantalla, es decir, las dimensiones del dispositivo.

El idioma por defecto de la aplicación es el español. Sin embargo, la aplicación cuenta con dos idiomas adicionales: inglés y francés. El propósito de esto es poder internacionalizar la aplicación para que, una vez implementada en Guatemala, otros países puedan adaptarla a sus países y a los factores de sus contextos, facilitando el área del idioma, pues este es configurable por el usuario dentro de la pantalla de perfil. Esta funcionalidad permite que se cumpla el objetivo específico de permitir que otros países utilicen esta aplicación, sin tener la barrera del idioma.

La aplicación está enfocada en, principalmente, instituciones guatemaltecas encargadas de la seguridad vial, quienes buscan recopilar y almacenar información acerca de algún hecho de tránsito en el que están involucrados vehículos y personas. Estas instituciones pueden ser PROVIAL, COVIAL, PNC, Ministerio Público, bomberos, PMT, entre otros. Cada una de estas instituciones, al momento de llegar al lugar del accidente, recopilan la información que cada una necesita, por ejemplo, el cuerpo de bomberos recopila información acerca de las personas y su salud, pero PNC y PMT recopila información sobre el vehículo, mientras que PROVIAL y COVIAL, sobre el entorno donde ocurrió.

Se desarrollaron tres prototipos en total. El primero, como se mencionó en el marco metodológico, no cuenta con funcionalidad, sino que más bien sirve como una prueba de concepto y del resultado de analizar cómo se verán las pantallas en el resultado final. Como parte de la prueba de

concepto, para la parte del formulario, siendo la más relevante en el proyecto, se analizó debidamente cómo se distribuiría la petición de la información al usuario. La razón de esto es que, el estándar IRTAD, al contener múltiples datos que deben llenarse, se vuelve muy extenso, por lo tanto, la idea de este primer prototipo fue descubrir cómo organizar la información en diferentes pantallas.

El segundo prototipo contiene una funcionalidad específica, guardar información en una base de datos. En este prototipo desarrollado en AppSheet se buscó realizar una conexión del prototipo hacia la base de datos realizada por el equipo de backend. El prototipo en AppSheet se conectó a PostgreSQL, gracias a esto se pudo determinar qué campos en las tablas pueden ser llenados automáticamente con información calculada por la aplicación, como la fecha, hora, ubicación: latitud y longitud, país, estado, etc.

El prototipo final, siendo el resultado final del proyecto, es el que se espera que utilicen las instituciones guatemaltecas principalmente para recabar información acerca de los accidentes vehiculares. Como se comentó anteriormente, se desarrollaron múltiples pantallas con una funcionalidad diferente cada una que aporta para el cumplimiento de los objetivos del proyecto, específicamente para este módulo para desplegar información almacenada. Es importante mencionar que, para cada pantalla desarrollada, se implementó una versión para aplicación de escritorio y una versión para aplicación para teléfonos móviles.

En primer lugar, la implementación de la pantalla de menú tiene el propósito de brindar múltiples posibles acciones para que el usuario pueda conseguir lo que busca fácilmente. Se implementó un componente llamado *sidenav* en la cual se puede acceder a las demás pantallas: nuevo accidente, menú de accidentes, perfil. Esta barra de navegación tiene dos posibles estados: abierto y cerrado. En el caso de la versión de escritorio (desktop), la barra de navegación se mantendrá siempre abierta, pues se consideró que debido al tamaño disponible en pantalla hay espacio suficiente para mantener este componente abierto. Por el otro lado, en la versión de teléfono (móvil), hay un botón llamado “menú de hamburguesa” el cuál muestra y oculta el componente. Para este último caso, se implementó así debido a la falta de espacio en pantalla, y que, en ocasiones es más cómodo para aplicaciones de teléfono.

También en el menú, por buenas prácticas de experiencia de usuario, se desarrollaron botones que le permiten al usuario sentirse más cómodo al usar la aplicación. Hay un botón en la esquina superior derecha para refrescar la información para evitar que el usuario tenga que actualizar la página completa. También, un botón en la esquina superior derecha para cerrar sesión. En la esquina inferior derecha, hay un botón que facilita el acceso a la creación de un nuevo accidente. Finalmente, existe un

resumen de los registros que se han registrado el día actual. Es decir, aunque hayan registros de accidentes con fecha de antier, ayer y hoy, solamente se desplegarán los accidentes registrados hoy.

El resumen muestra los campos más importantes del accidente para poder identificarlo fácilmente: nombre y puesto del usuario que creó el accidente vehicular; este es importante porque muchas entidades están involucradas en la toma de datos, entre las cuáles podrían ser PNC, PMT, bomberos, Ministerio Público, etc, por lo tanto, es importante identificar quién registró el evento; el código identificador del accidente; un listado de vehículos con el tipo de vehículo (carro o motocicleta), marca y color; y, el número de personas involucradas, como se ve en las figuras 17 al 20.

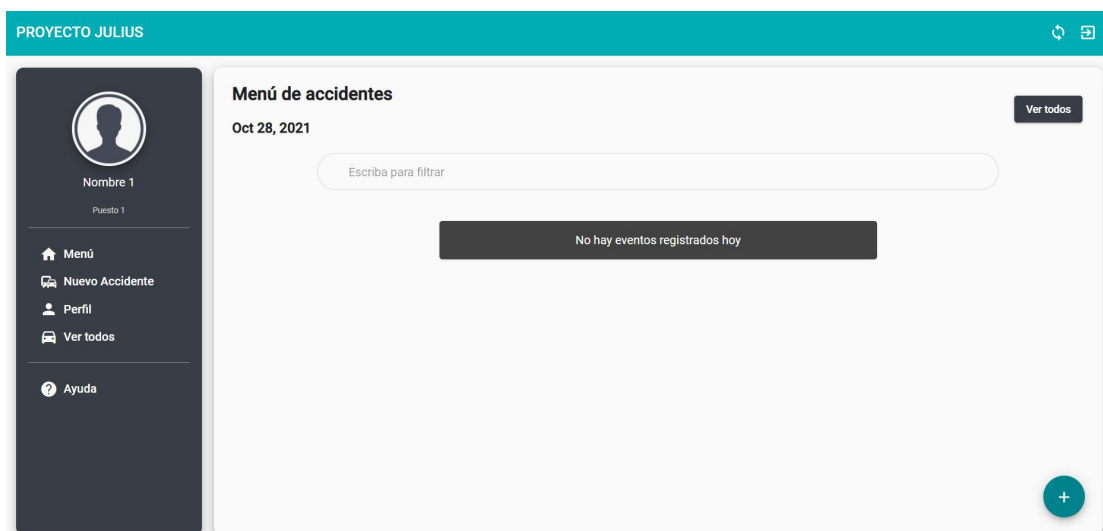


Figura 16. Menú de accidentes vacío

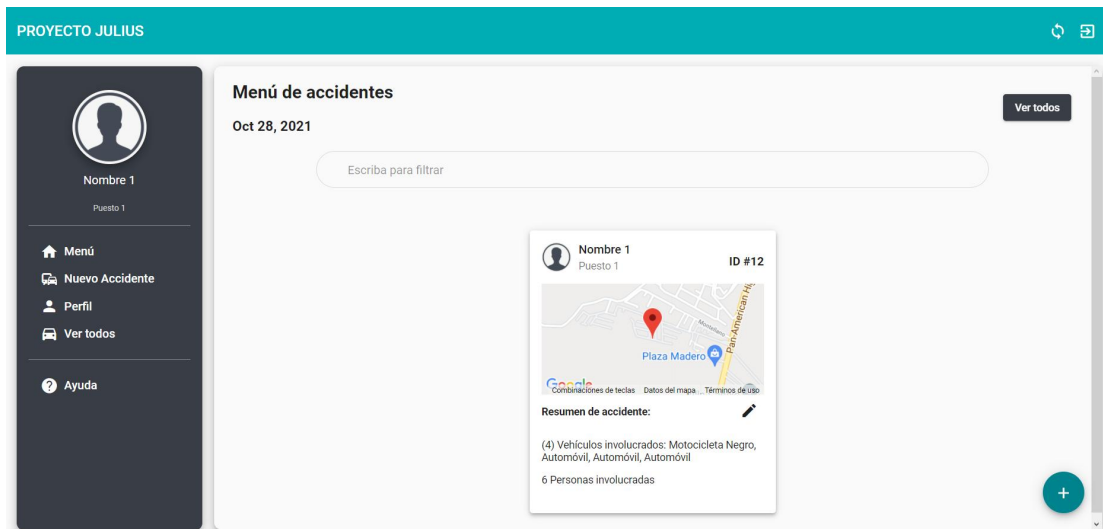


Figura 17. Menú de accidentes

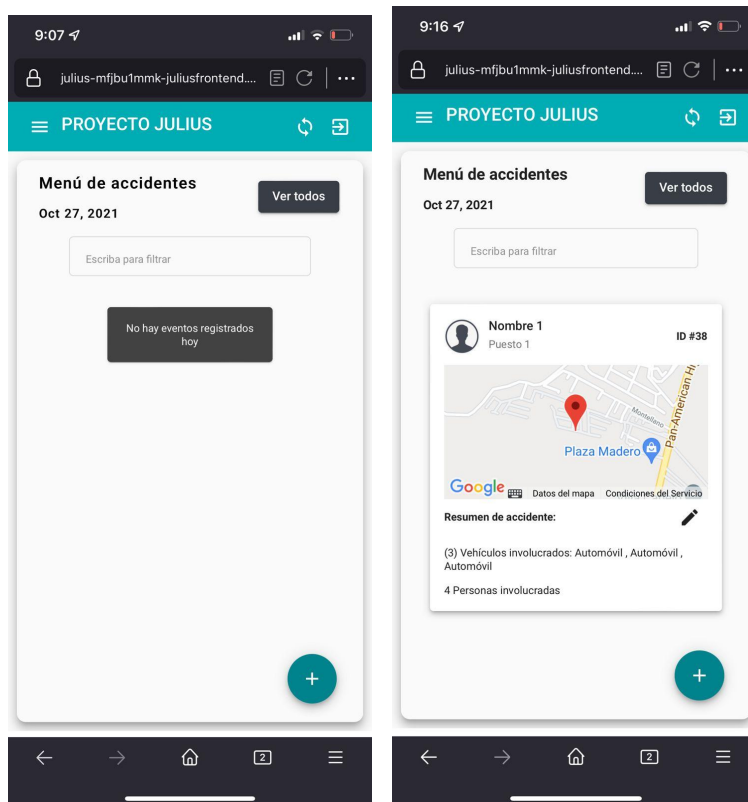


Figura 18. Menú de accidentes versión móvil

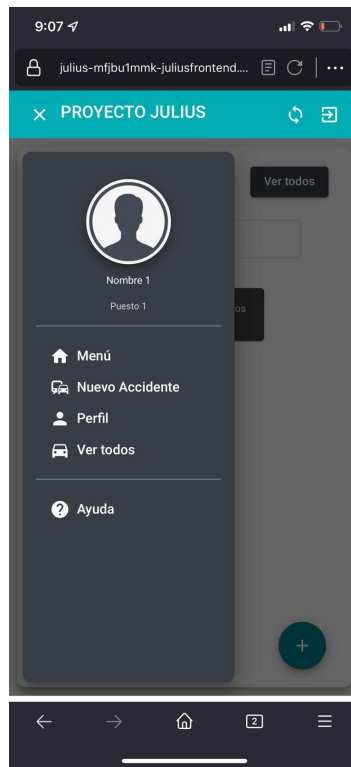


Figura 19. Slide menú celular

Seguido del menú principal, se encuentra el menú de accidentes. Esta pantalla le permite al usuario poder tener un control histórico de los accidentes que se han registrado en la aplicación. La funcionalidad más importante en este punto es poder filtrar los accidentes dado un intervalo de fechas y palabras clave. Para facilitar la usabilidad, se tiene predeterminado que el intervalo de fechas que traerá la consulta al servidor de backend sería de una semana, desde siete días antes del día actual hasta el día actual. El usuario puede aumentar este intervalo o disminuirlo según este lo desee, teniendo como restricción que no puede poner una fecha posterior a la fecha actual.

Por otro lado, hay un buscador que permite filtrar los accidentes en base a lo que el usuario escriba. Este texto escrito por el usuario se compara con la información de código de identificación del accidente, fecha, marca de algún vehículo dentro del accidente y color de algún vehículo dentro del accidente. Gracias a lo anterior, la información que cumpla con las condiciones del intervalo de fecha y el filtro ingresado se despliega en la tabla de esta pantalla. Cada elemento dentro de la tabla contiene un botón, el cuál da la opción de editar el accidente o consultar toda la información de este accidente. Este resultado puede visualizarse en las figuras 21 y 22.

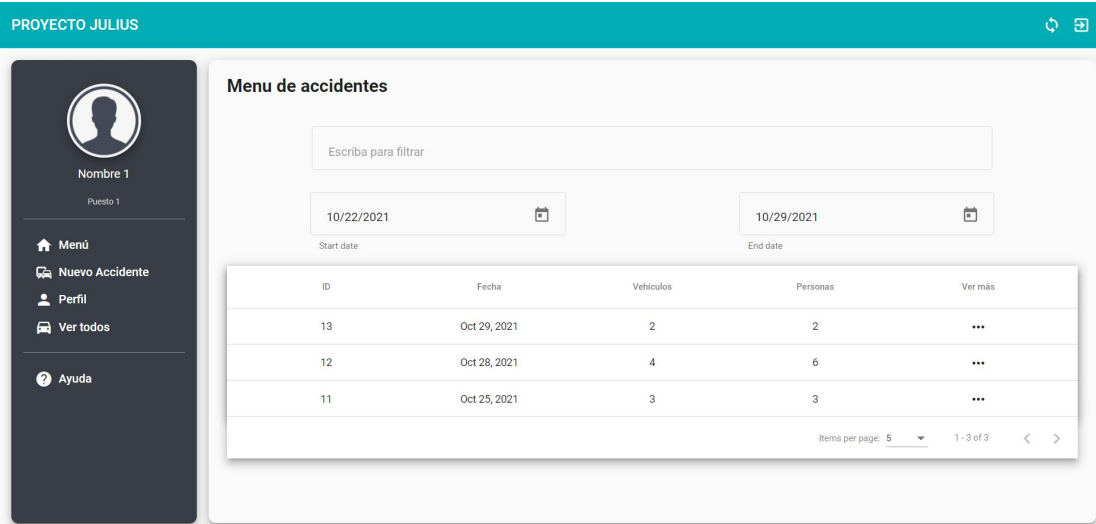


Figura 20. Filtro de menú de accidentes

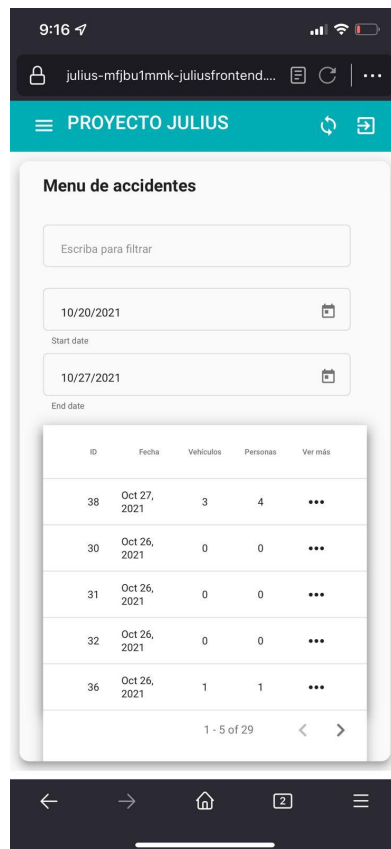


Figura 21. Filtro de menú de accidentes

Luego, el desarrollo de la pantalla de consulta de un accidente vehicular es una de las pantallas más importantes de la aplicación. En esta, el usuario consulta la información de los accidentes que ha registrado, y puede acceder a este a través del menú principal y a través del menú de accidentes. Esta pantalla tiene mucha relevancia pues en ésta, el agente que requiera revisar alguna información por algún caso no resuelto o por una investigación que alguna institución necesite. Es por esto, que la integridad de los datos debe asegurarse por la aplicación, para no perder información que el usuario ingrese.

Como consecuencia de lo anterior, se diseñaron cuatro pestañas para visualizar en grupos la información. En la primera pestaña se muestra la información del entorno en dónde ocurrió el accidente. Es decir, todo lo relacionado a la superficie, al ambiente, fecha, lugar, contra qué cosas colisionó el vehículo, etc. Luego, en la segunda pestaña se muestra la información de los vehículos: tipo de vehículo, color, marca, dirección donde ocurrió el accidente y daños en el vehículo. Después, en la tercera pestaña, se encuentra la información de todas las personas que estuvieron involucradas en el accidente, estas pueden ser peatones, conductores o pasajeros del vehículo. Por último, información del clima en el que ocurrió el accidente, para este, se calcula a través de una API la temperatura, la velocidad del viento, humedad, presión, etc. El resultado de esta pantalla puede visualizarse en las figuras 23 a la 28.

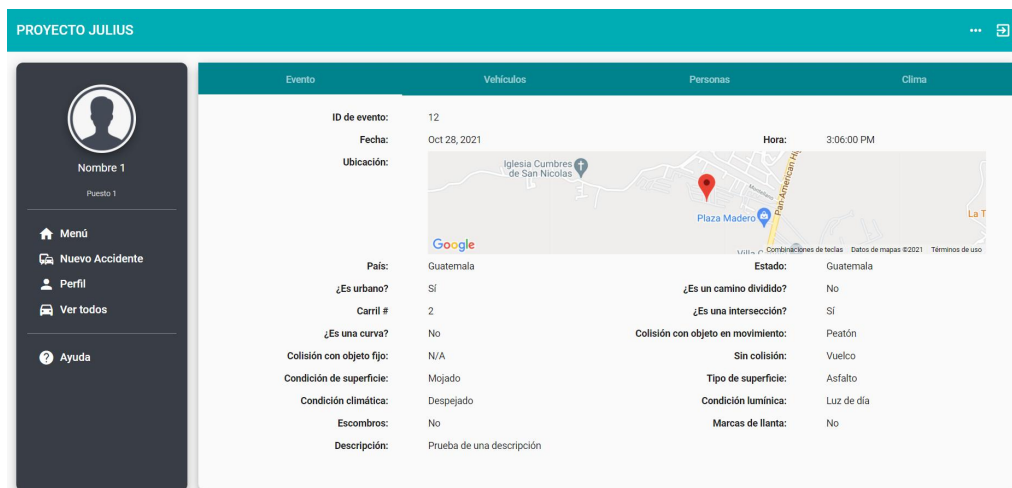


Figura 22. Datos del accidente: Evento

PROYECTO JULIUS

Evento Vehículos Personas Clima

Vehículo 1 Automóvil Acura 1

Tipo de vehículo: Automóvil Marca: Acura
 Modelo: ILX Color: Blanco
 Año: 2002 ¿Es claro?: No
 ¿Es oscuro?: Sí ¿Es de varios colores?: No

Dirección de fuerza:

¿Daños en vehículo?

Severidad: Daños cosméticos En carrocería: No
 En ventanas: No En motor: No

Figura 23. Datos del accidente: Vehículos

PROYECTO JULIUS

Evento Vehículos Personas Clima

Persona 1 Femenino 25años

Género: Femenino Edad: 25
 ¿Pasajero de carro?: Sí ¿Con cinturón?: Sí

Asiento:

¿Falleció?: No

Escala de lesiones por región:

Cabeza: Sin lesión Cara: Sin lesión
 Cuello: Sin lesión Tórax: Sin lesión
 Abdomen: Sin lesión Columna vertebral: Sin lesión
 Extremidades superiores: Sin lesión Extremidades inferiores: Sin lesión
 Externo u otro: Sin lesión ¿Alcohol o drogas?: No

Figura 24. Datos del accidente: Personas

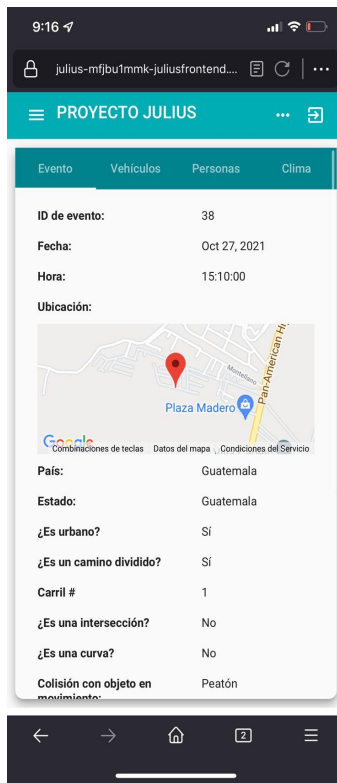


Figura 25. Datos del accidente móvil

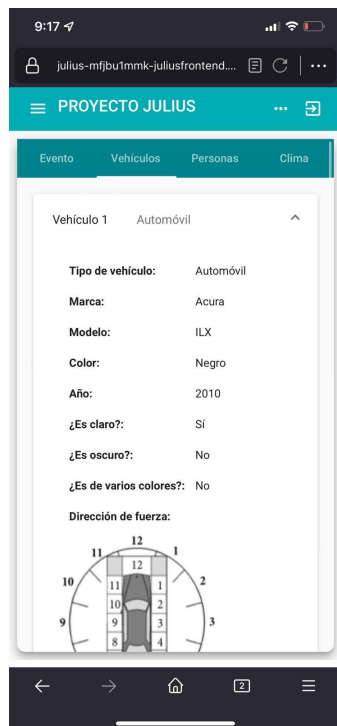


Figura 26. Datos del accidente móvil: Vehículo



Figura 27. Datos del accidente móvil: Personas

Además, se implementó la pantalla de perfil del usuario. Esta pantalla contiene dos funcionalidades importantes. La primera funcionalidad es que se muestra información del usuario: el nombre, puesto y su correo electrónico. La segunda funcionalidad permite configurar el idioma en toda la aplicación entre los idiomas español, inglés o francés. El idioma por defecto dentro de la aplicación es el español debido a que su principal funcionalidad es para Guatemala, sin embargo, el usuario es capaz de guardar el idioma que desea que su aplicación funcione en el futuro pues esta configuración es persistente con el tiempo.

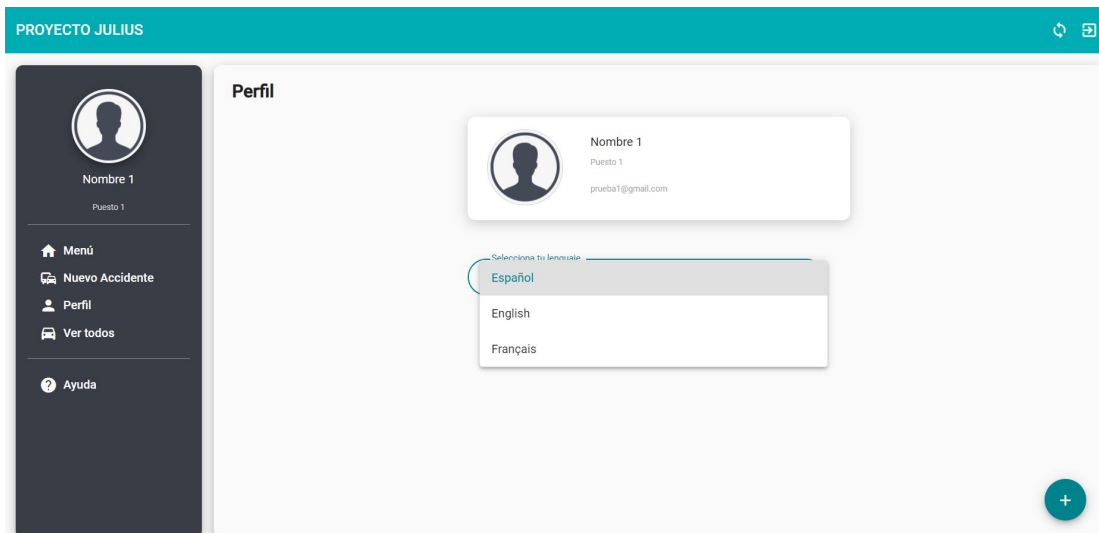


Figura 28. Perfil

Debido a que Julius es una PWA, debe tener un diseño adaptable para cualquier dispositivo, y debe ser sencilla de utilizar. En la versión final de la aplicación, se encuentran las siguientes pantallas:

- Login.
- Menú de accidentes.
- Nuevo accidente (Formulario).
- Perfil.
- Ver todos los accidentes.

La página con la que se tiene más relación respecto del formulario es la pantalla de ver todos los accidentes, debido que aquí es donde se despliega la información que ya ha sido ingresada de un accidente, y también aquí es donde se encuentra la opción de editar el accidente para redirigir al usuario al formulario, y llenar los campos que ya han sido ingresados de vehículos y personas involucradas, y la información general del accidente.

Para la versión final del formulario de Julius, hay dos formatos de formularios, uno para cada tipo de dispositivo. El formulario, en general, utiliza los mismos componentes y mantiene la misma estructura. Siempre se mantiene el formato de cinco subformularios:

1. Datos Importantes
2. Descripción
3. Datos Generales
4. Tipo de Vehículo
5. Personas Involucradas

Las siguientes imágenes de las pantallas finalizadas se realizaron en los siguientes dispositivos:

- Desktop: Laptop 15”
- Móvil: iPhone 11

6.1 Sección de datos importantes

Para la sección de Datos Importantes, el resultado de la pantalla se ve así:

PROYECTO JULIUS

Nuevo accidente

1 2 3 4 5

Datos importantes

Fecha y hora del accidente
10/27/2021 09:05 PM

Ubicación

Google Maps showing San Angel, Guatemala. Landmarks include San Angel II y III and Condominio Alamedas de San Gabriel.

Pais
Guatemala

Figura 29. Pantalla de formulario en sección de Datos Importantes con vista de “stepper” en formato desktop

PROYECTO JULIUS

Datos importantes

Fecha y hora del accidente
10/27/2021 09:05 PM

Ubicación

Google Maps showing San Angel, Guatemala. Landmarks include San Angel II y III and Condominio Alamedas de San Gabriel.

Pais
Guatemala

Estado
Guatemala

¿Es urbano?
 No Sí

Siguiente

Figura 30. Pantalla de formulario en sección de Datos Importantes en formato desktop

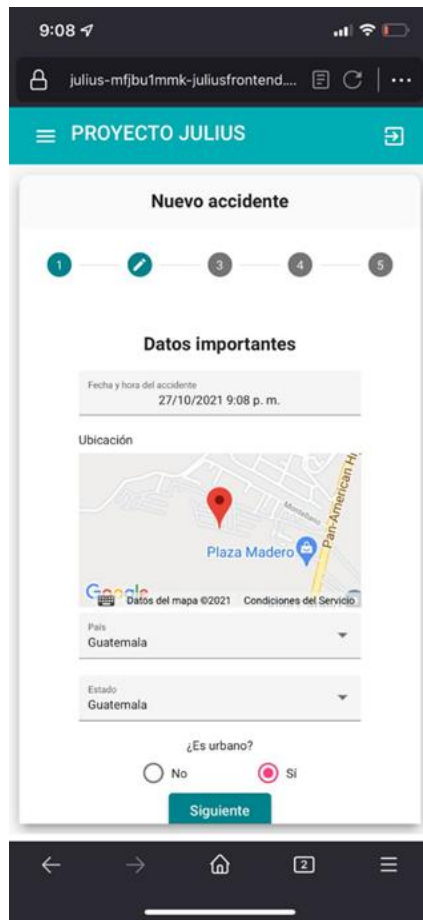


Figura 35. Pantalla de formulario en sección de Datos Importantes en formato móvil



Figura 36. Consulta al usuario sobre dar permiso de ubicación a Julius

Para este subformulario, la información es predeterminada, con excepción del campo ¿Es Urbano?, no obstante, todos los campos son editables, y variables, como el campo de estados que es dependiente del campo de País. Es importante que el usuario otorgue permiso de localización al dispositivo, como se puede visualizar en la Figura 36, para el correcto funcionamiento de la aplicación, para obtener su localización (latitud y longitud) para que se registre y se vea la posición en el mapa, como se visualiza en las figuras 34 y 35, y consultar la información meteorológica por medio de un API. El componente del mapa se puso como retroalimentación al usuario de en donde está ubicado en el accidente.

6.2 Sección de descripción

Para la sección de Descripción, el resultado de la pantalla se ve así:

The screenshot shows the 'Nuevo accidente' form in the 'Descripción' section. The form is titled 'Nuevo accidente' and has a progress indicator with five steps. The first step is active, and the second step is highlighted. The form contains the following fields:

- Tipo de calle
- Número de carril: 1
- Objeto en movimiento
- Colisión con objeto fijo

Figura 37. Pantalla de formulario en sección de Descripción, parte superior, en formato desktop

The screenshot shows the 'Nuevo accidente' form in the 'Descripción' section, focusing on the bottom part. The form contains the following fields:

- Condición de superficie: Mojado
- Tipo de superficie
- Condición lumínica
- Descripción
- Marca de neumático: No Sí
- Escombros: No Sí
- Anterior (button)
- Siguiente (button)

Figura 38. Pantalla de formulario en sección de Descripción, parte inferior, en formato desktop

Figura 39. Pantalla de formulario en sección de Descripción, en formato móvil

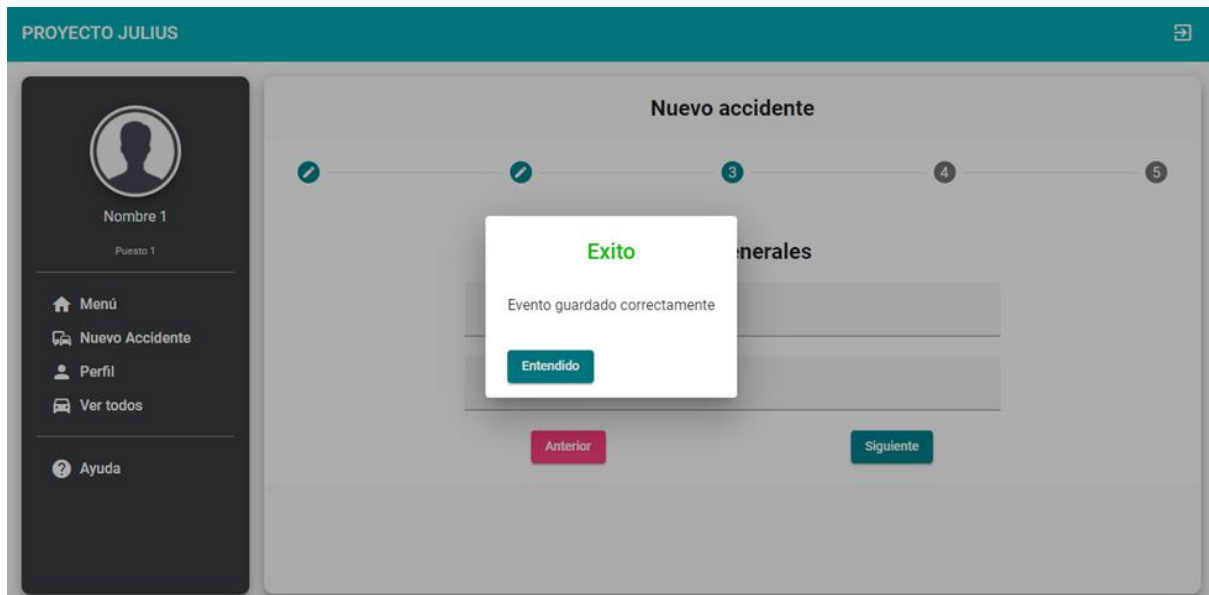


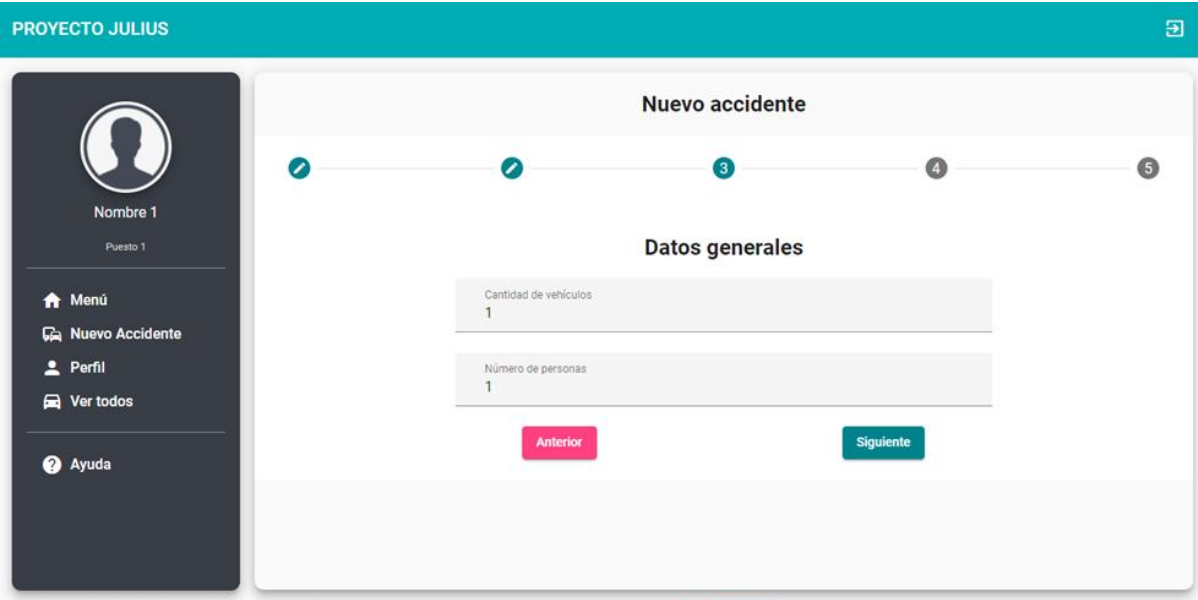
Figura 40. Diálogo de retroalimentación de éxito

Para este subformulario, como se puede ver en las figuras 37 y 38, es mucho más extenso que el anterior, ya con información más específica sobre detalles sobre el accidente, como el entorno, la iluminación del entorno, la condición en la cual se produjo el accidente, y que elementos del entorno se vieron afectados. En la figura 38, se puede ver el botón de siguiente que aparece inhabilitado, el cual se habilita solamente si el usuario llena ambos formularios (Datos Importantes y Descripción), debido que esta es la información clave del accidente. Los campos como el número de carril o

Descripción cuentan con sus respectivas validaciones, ya que estos son campos que no tienen opciones fijas, como el resto de los campos del formulario que poseen su lista de valores con las opciones disponibles, y al presionar el botón de Siguiente, se envía la información al formulario, se dirige al usuario a la siguiente fase del formulario y se despliega el diálogo de retroalimentación de si la consulta fue un éxito o un error, como se aprecia en la figura 40.

6.3 Sección de datos generales

Para la sección de Datos Generales, el resultado de la pantalla se ve así:



The screenshot displays a web interface for 'PROYECTO JULIUS'. On the left is a dark sidebar with a user profile (Nombre 1, Puesto 1) and navigation links: Menú, Nuevo Accidente, Perfil, Ver todos, and Ayuda. The main content area is titled 'Nuevo accidente' and features a progress indicator with five steps, where step 3 is active. Below the progress bar is the 'Datos generales' section, containing two input fields: 'Cantidad de vehículos' with the value '1' and 'Número de personas' with the value '1'. At the bottom of this section are two buttons: 'Anterior' (pink) and 'Siguiente' (teal).

Figura 41. Pantalla de formulario en sección de Datos Generales en formato desktop



Figura 42. Pantalla de formulario de sección de Datos Generales en formato móvil

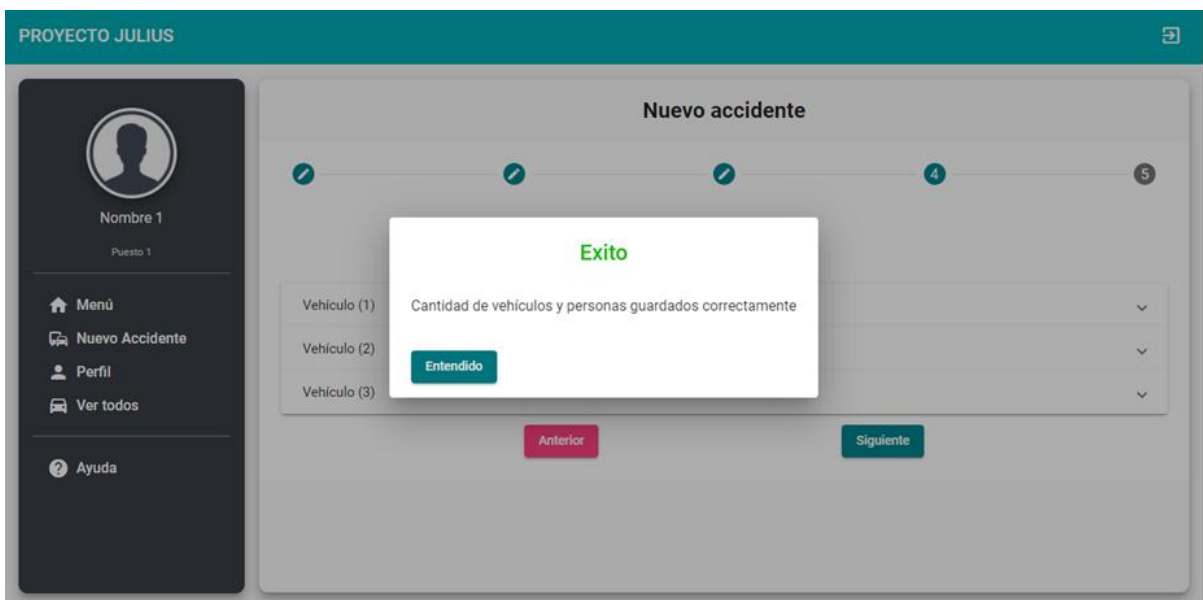


Figura 43. Diálogo de retroalimentación de éxito al registrar número de vehículos y personas involucradas

Esta pantalla, es muy simple en cuestión de diseño, pero es la que más validaciones involucró al momento de si realizar la consulta o no, o si cambia el valor de los campos, debido que esto afecta los siguientes subformularios, aun así, el diseño es el mismo como se visualiza en las figura 41 y 42, y no continua al siguiente formulario si no se despliega el diálogo de éxito como se ve en la Figura 43.

6.4 Sección de tipo de vehículo

Para la sección Tipo de vehículo, el resultado de la pantalla se ve así:

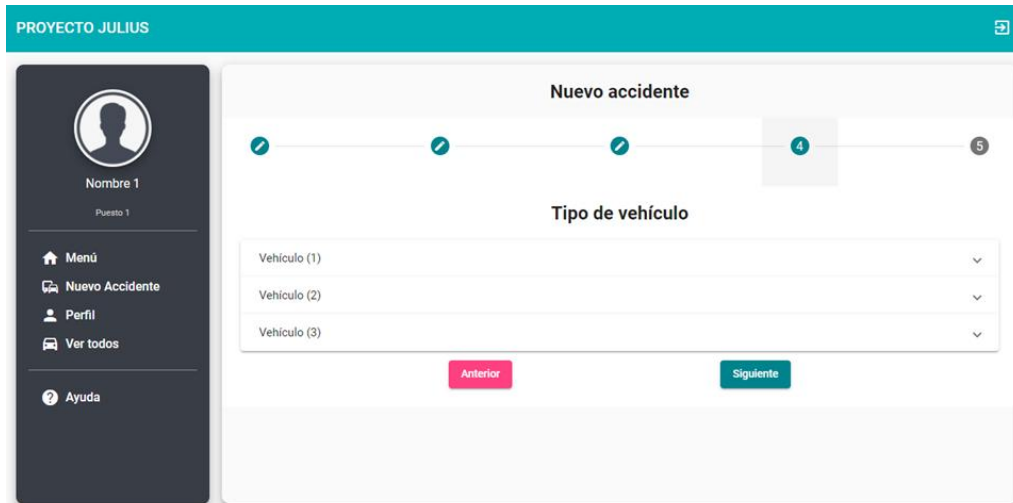


Figura 44. Pantalla de formulario en sección de Tipo de Vehículo, con la visualización de todos los vehículos, en formato desktop

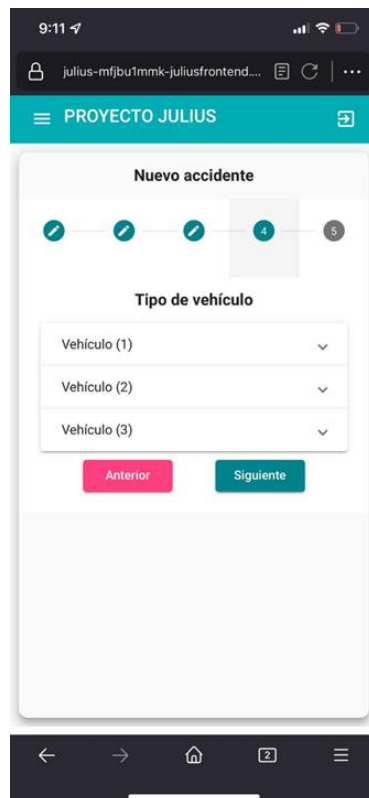


Figura 45. Pantalla de formulario en sección de Tipo de Vehículo, con la visualización de todos los vehículos, en formato móvil

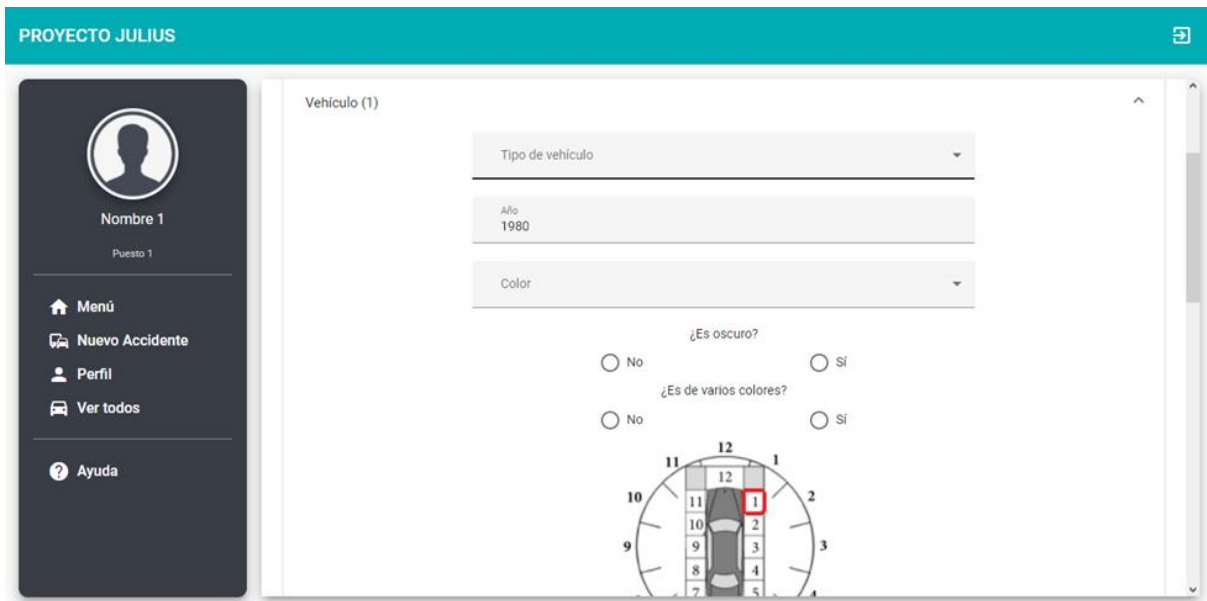


Figura 46. Pantalla de formulario en sección de Tipo de Vehículo, ya con un vehículo seleccionado, parte superior en formato desktop

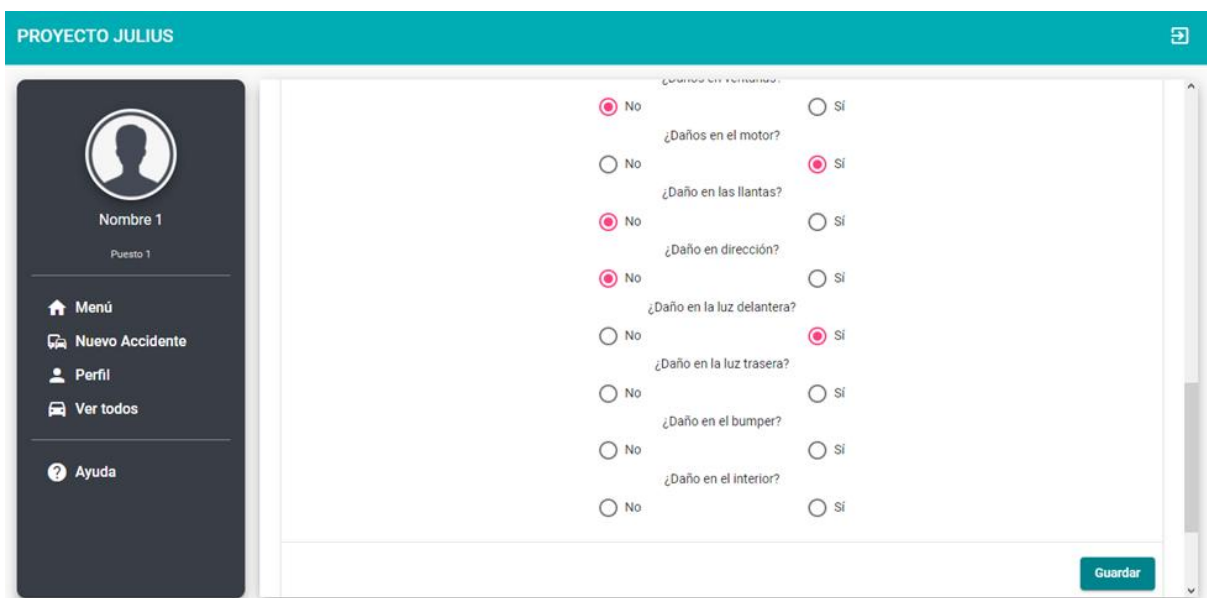


Figura 47. Pantalla de formulario en sección de Tipo de Vehículo, ya con un vehículo seleccionado, parte inferior en formato desktop

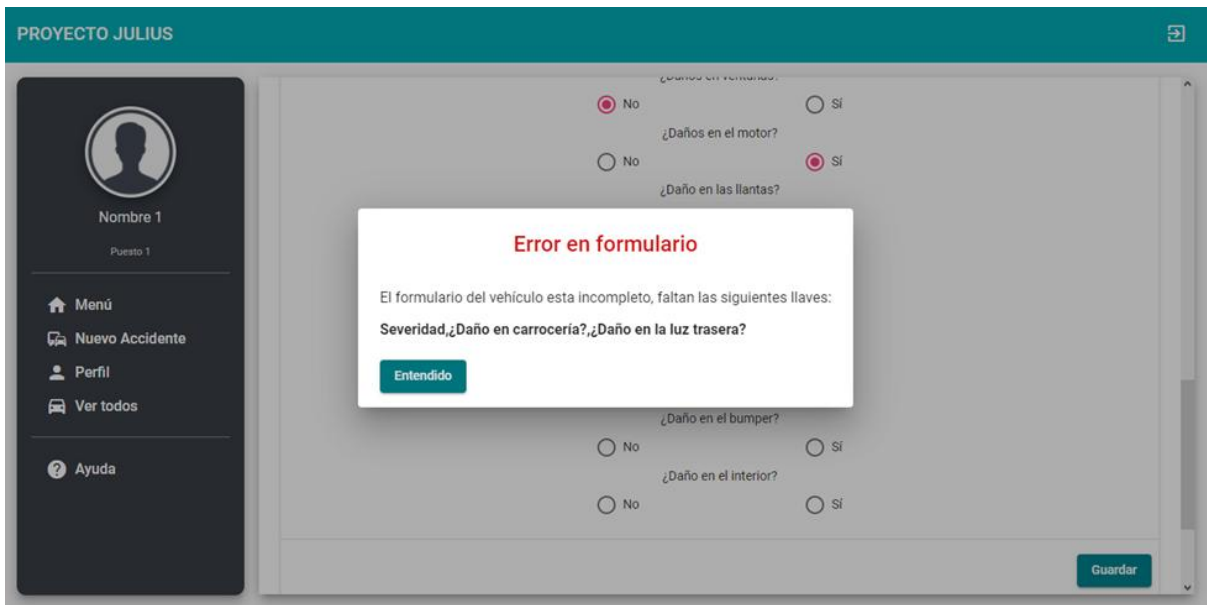


Figura 48. Pantalla de formulario en sección de Tipo de Vehículo, diálogo de retroalimentación de error al llenar el formulario.

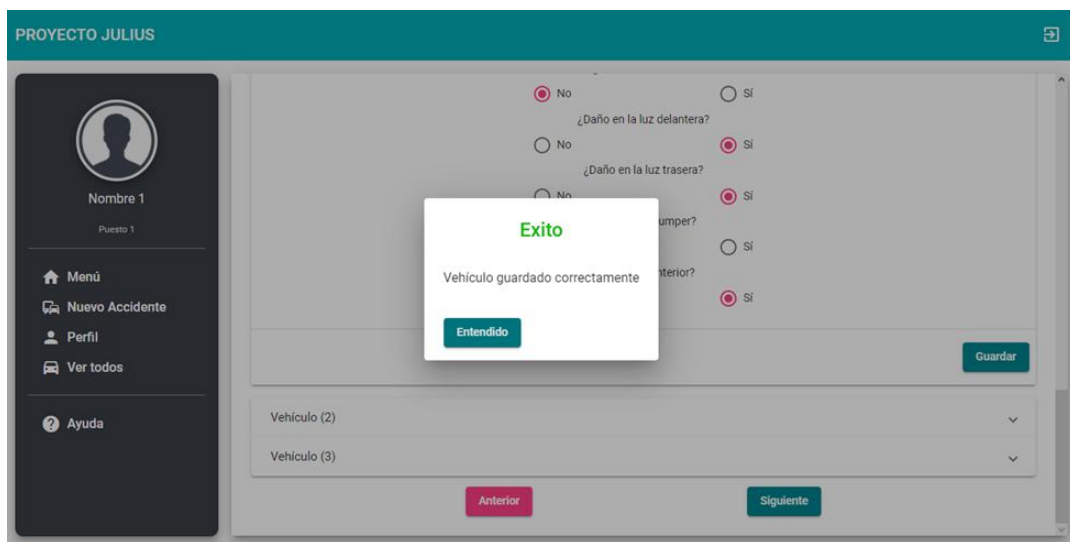


Figura 49. Pantalla de formulario en sección de Tipo de Vehículo, diálogo de retroalimentación de éxito, al guardar un vehículo.

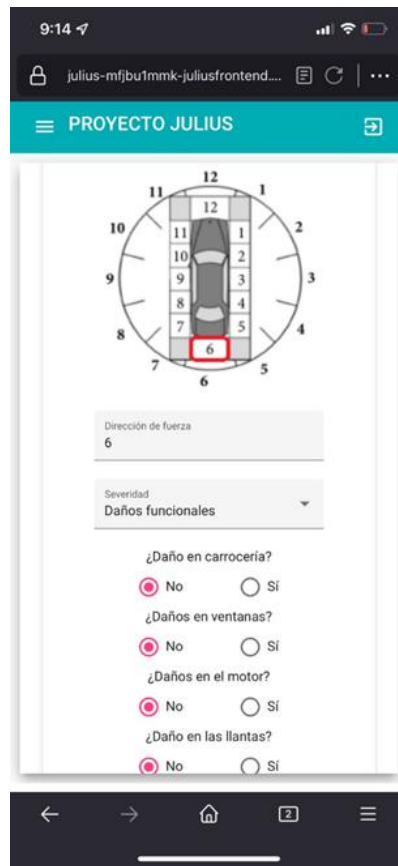


Figura 50. Pantalla de formulario en sección de Tipo de Vehículo, ya con un vehículo seleccionado, parte de en medio, en formato móvil

Como se aprecia en las figuras 44 y 45, se crean la cantidad de subformularios de vehículos dependiendo la cantidad ingresada en el formulario de Datos Generales, y cada formulario es independiente de cada vehículo, o sea que se pueden llenar en diferentes momentos y por diferentes usuarios si se requiere. En la Figura 47, se visualiza el botón enviar del subformulario de ese vehículo, esto para que sea más sencillo para el usuario el llenar un vehículo y enviar la información al terminar de llenar la información de este y al presionarlo depende si está completo o no el formulario, u ocurre un error es que tipo de diálogo se despliega al usuario. En la Figura 48, se visualiza el diálogo cuando un usuario no ha ingresado todos los campos del formulario, y se le indica los primeros 3 campos que se encontraron que no han sido llenados, y una vez finalizado se despliega el diálogo de éxito como se ve en la Figura 49. El único componente es el del mapa de dirección de la fuerza del impacto, el que parece un reloj en la Figura 50, el cual cambia el cuadro rojo, dependiendo del valor de la dirección de la fuerza que indique el usuario. El formulario en su esencia no es extenso, sin embargo, la habilitación de subformularios a causa de la participación o involucramiento de más vehículos y/o personas producen que la información requerida aumente, por lo que se hace necesaria la opción para el registro de forma asíncrona.

6.5 Sección de personas involucradas

Para la sección de Personas Involucradas, el resultado de la pantalla se ve así:

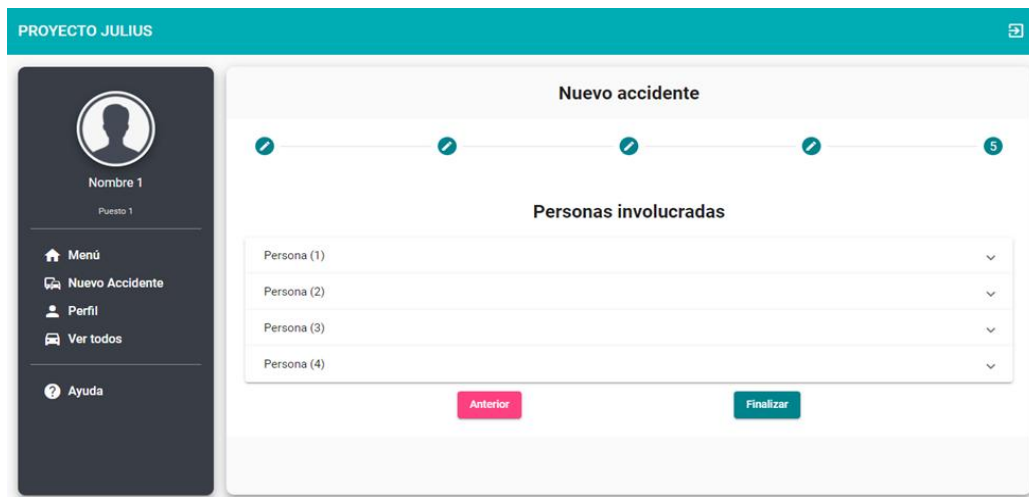


Figura 51. Pantalla de formulario en sección de Personas Involucradas, con la visualización de todas las personas involucradas, en formato desktop

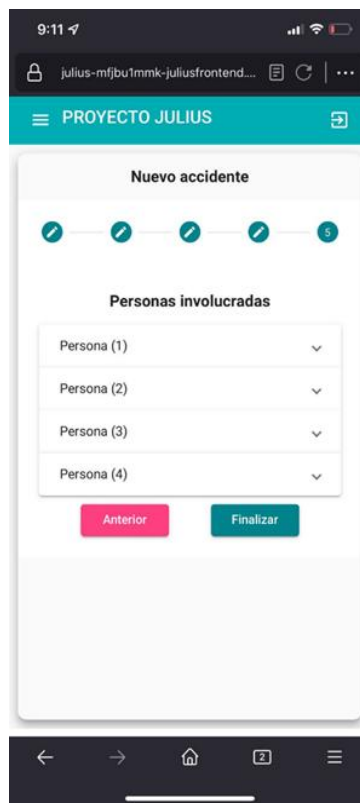


Figura 52. Pantalla de formulario en sección de Personas Involucradas, con la visualización de todas las personas involucradas, en formato móvil

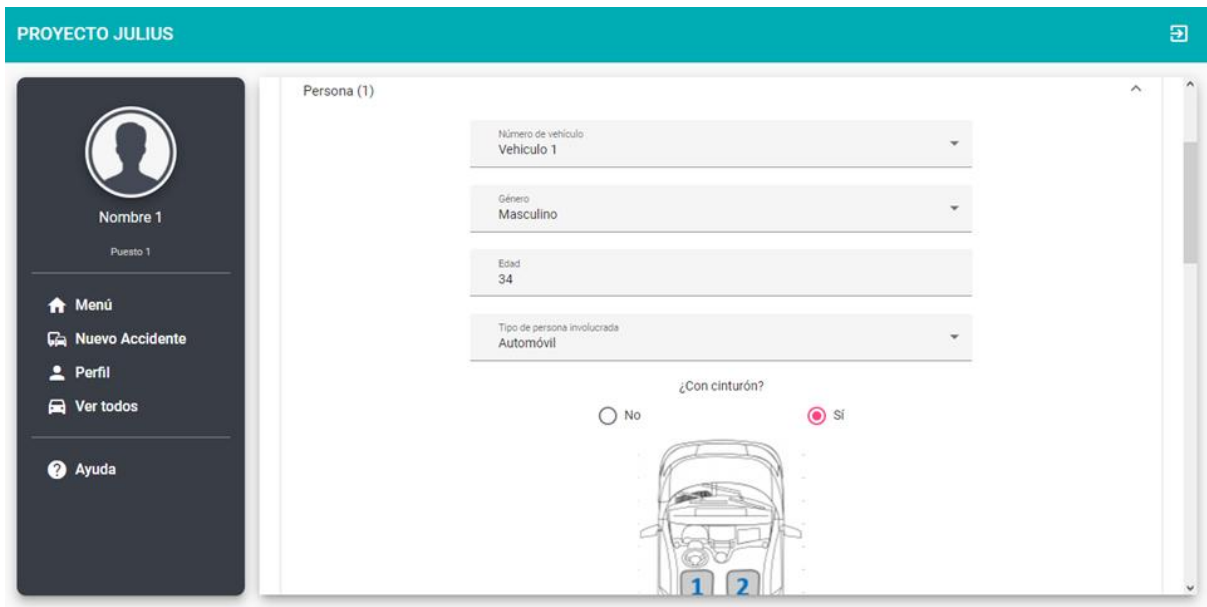



Figura 53. Pantalla de formulario en sección de Personas Involucradas, parte superior del formulario en formato desktop.



Figura 54. Pantalla de formulario en sección de Personas Involucradas, sección de asiento de persona que iba en automóvil, en formato desktop.

PROYECTO JULIUS



Nombre 1
Puesto 1

- [Menú](#)
- [Nuevo Accidente](#)
- [Perfil](#)
- [Ver todos](#)

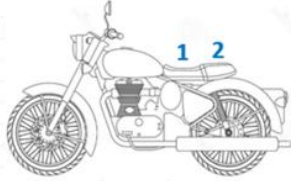
- [Ayuda](#)

Edad
34

Tipo de persona involucrada
Motocicleta

¿Con casco?

No Si




Asiento
2

¿Falleció?

No Si

Figura 55. Pantalla de formulario en sección de Personas Involucradas, sección de asiento de persona que iba en motocicleta, en formato desktop.

PROYECTO JULIUS



Nombre 1
Puesto 1

- [Menú](#)
- [Nuevo Accidente](#)
- [Perfil](#)
- [Ver todos](#)

- [Ayuda](#)

Lesión de columna vertebral
Sin lesión

Lesión de la extremidad superior
Sin lesión

Lesión de la extremidad inferior
Sin lesión

Lesión externa u otra
Sin lesión

Puntuación gravedad lesión: 0/75

¿Alcohol o drogas?

No Si

¿Eyectado del vehículo?

No Si

[Guardar](#)

Figura 56. Pantalla de formulario en sección de Personas Involucradas, parte inferior, en formato desktop.

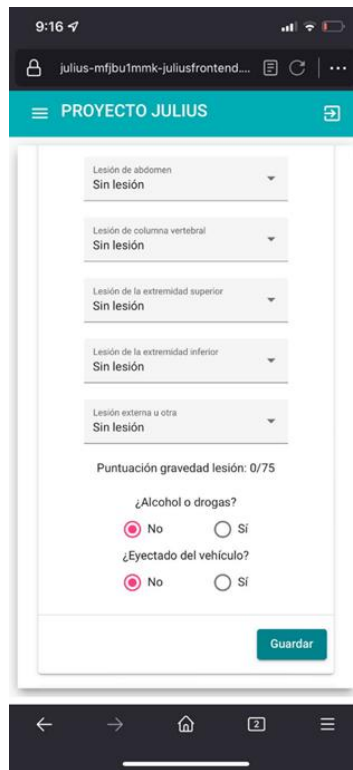


Figura 57. Pantalla de formulario en sección de Personas Involucradas, parte inferior, en formato móvil.

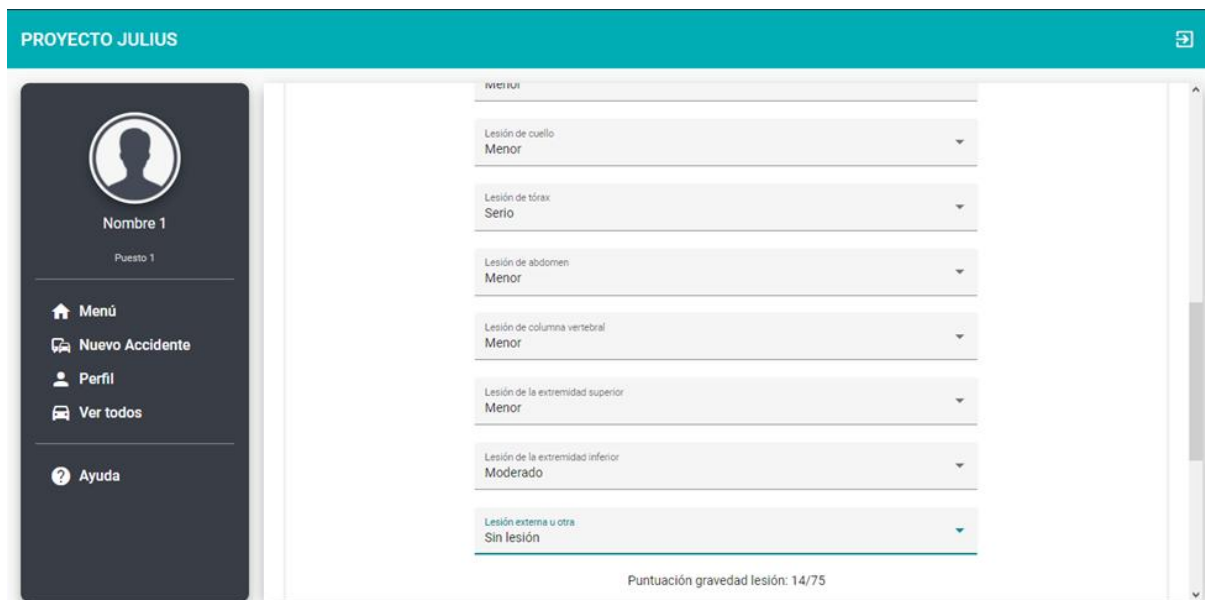


Figura 58. Pantalla de formulario en sección de Personas Involucradas. Sección de cálculo de ISS.

Como se puede observar en la Figura 51 y la Figura 52, de la misma manera en que se presentan los vehículos involucrados, se presenta en el componente de “acordeón”, la cantidad de personas involucradas en el accidente, y al seleccionar una persona, se muestra el subformulario de la persona involucrada, donde primero se pregunta información de la persona, respecto al accidente, después la información personal como el género o la edad, y dependiendo del lugar de ubicación de la persona durante el accidente, como pasajero en un vehículo o era peatón, se despliegan los campos del formulario, en el caso si eligió automóvil:

- ¿Con cinturón?
- No. de asiento, con un mapa de un automóvil con numeración de 1 a 8 para indicar donde se encontraba el pasajero, si aplica o se conoce la información.

En la Figura 54, se pueden observar los campos al seleccionar automóvil, en el campo de Tipo de persona involucrada

En el caso que se eligió motocicleta:

- ¿Con casco?
- No. de asiento con un mapa de una motocicleta con numeración de 1 a 2 para indicar donde se encontraba el pasajero, si aplica o se conoce la información.

En la Figura 55, se pueden observar los campos al seleccionar una motocicleta, en el campo de Tipo de persona involucrada.

Estas imágenes de mapas de número de asiento fueron dadas por el asesor y stakeholder de la aplicación Franz Haidacher, para una mejor retroalimentación del usuario.

En la Figura 58 se muestran los campos respecto al ISS (Injury Severity Score), que al ingresar información en ellos se realiza el cálculo de la gravedad de la lesión de la persona involucrada, y hay 9 campos involucrados en el cálculo los cuales son:

1. Lesión craneal.
2. Lesión facial.
3. Lesión de cuello.
4. Lesión de tórax.
5. Lesión de abdomen.
6. Lesión de columna vertebral.
7. Lesión de la extremidad superior.
8. Lesión de la extremidad inferior.
9. Lesión externa u otra.

Y cada uno de los campos posee las siguientes opciones:

1. Sin lesión.
2. Menor.
3. Moderado.
4. Serio.
5. Severo.
6. Crítico.
7. Máxima (Si se selecciona esta opción en cualquier campo el resultado del ISS es 75).

Al igual que en la pantalla de Tipo de Vehículo, y el formulario de cada persona involucrada se puede enviar de forma independiente, de otras personas y también si no se ha llenado la información del vehículo en el que iba la persona, si es el caso.

6.6 Modo edición

Las pantallas de Datos Generales, Tipo de Vehículo y Personas Involucradas son las únicas pantallas que tienen la posibilidad de ser modificadas, y su diseño se mantiene igual en modo edición, esto porque la información de estos campos puede llegar a cambiar en el futuro, como la condición de una persona que pase su estado a fallecido, o en el caso haya ocurrido un error en el registro de vehículos, por ejemplo, y la información de las pantallas de Datos Importantes y Descripción, no son modificables, debido que la información general del accidente no va a cambiar, así como su ubicación, el tipo de la calle o demás, esto como medida de seguridad de que la información principal del accidente no se verá modificada ya sea por accidente o un usuario con una mala intención en el futuro. En la Figura 59, se observa el cambio del diseño en los campos sin la opción de modificar, en el cual el texto sale de manera clara, pero el color gris en los títulos de campos o en los radiobutton permiten evidenciar, que el campo no está activado.

The screenshot shows a mobile application interface for 'PROYECTO JULIUS'. On the left is a dark sidebar menu with a profile icon, 'Nombre 1', 'Puntaje 1', and options: 'Menú', 'Nuevo Accidente', 'Perfil', 'Ver todos', and 'Ayuda'. The main area is a form with the following fields:

- 'Condición de superficie' with a dropdown menu showing 'Mojado'.
- 'Tipo de superficie' with a dropdown menu showing 'Concreto'.
- 'Condición lumínica' with a dropdown menu showing 'Amanecer o atardecer'.
- 'Descripción' with a text input field containing 'Motivo de prueba'.
- 'Marca de neumático' with two radio buttons: 'No' (selected) and 'Sí'.
- 'Escombros' with two radio buttons: 'No' and 'Sí' (selected).

 At the bottom of the form are two buttons: 'Anterior' (red) and 'Siguiente' (teal).

Figura 59. Pantalla de formulario, sección de Descripción, en modo edición.

6.7 Cambio de lenguaje

El formulario, así como toda la aplicación, cuentan con la opción para traducir la información completa, los títulos de los formularios, de los campos y las listas de opciones a partir del idioma seleccionado por el usuario. Como se observa en la Figura 60, las opciones también están en inglés, como en lugar de motocicleta, aparece motorcycle. Esto es debido que, en los componentes del formulario, este posee un input o entrada en el que ingresa una lista de valores, que provienen de la base de datos, y todo lo que son campos, títulos u opciones como en el radiobutton, si se maneja de manera local la traducción, por medio de una librería de traducción de JavaScript. Actualmente la aplicación cuenta con 3 lenguajes:

- Español (lenguaje predeterminado)
- Inglés
- Francés

Se realizó la traducción de todo, primero debido que se espera la aplicación pueda ser utilizada no solamente en Guatemala, sino que, en un futuro, pueda llegar a ser utilizada en otros países, que no tendrá problema en ser utilizada, debido a que ya utiliza un estándar internacional.

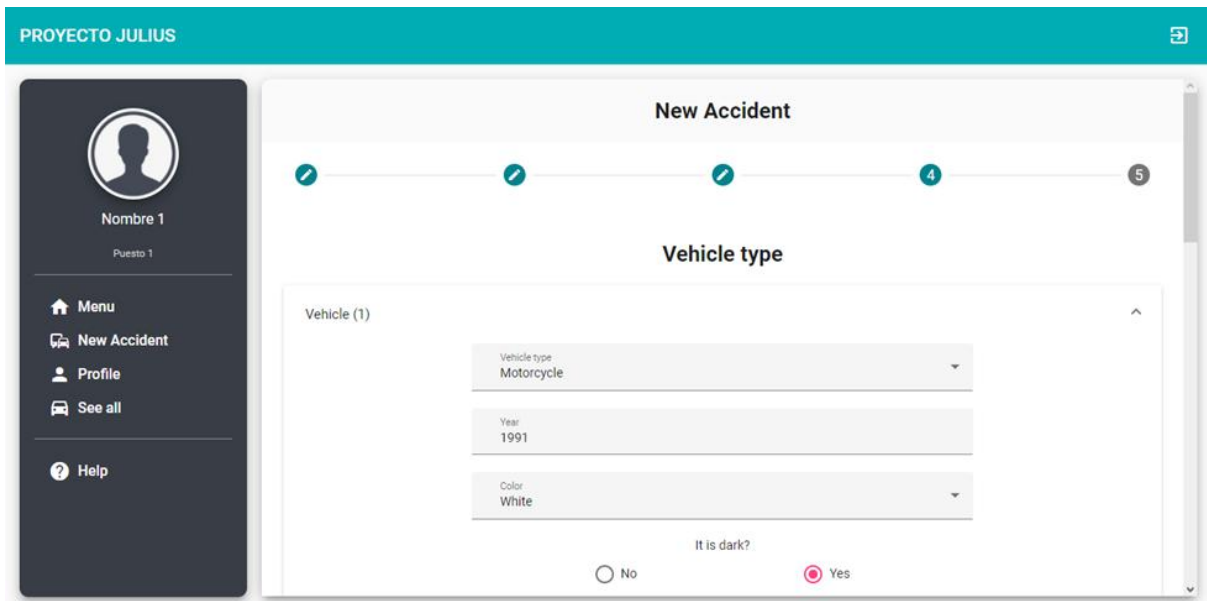


Figura 60. Pantalla de formulario, sección de Tipo de Vehículo, en el idioma inglés

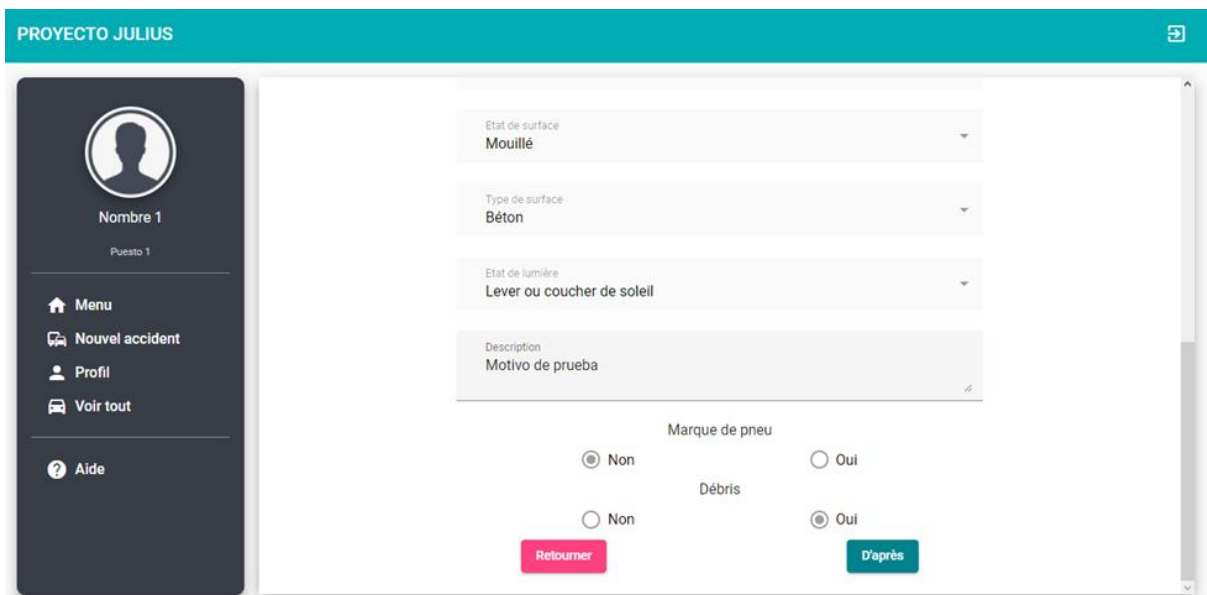


Figura 61. Pantalla Formulario, sección de Descripción, en el idioma francés.

6.8 Funcionalidad en otros dispositivos

Anteriormente se mostró en los resultados de las pantallas, como se visualizaban las pantallas del formulario en dispositivos, como una laptop y un iPhone 11. También una característica importante de Julius, es que al ser PWA puede instalarse en cualquier dispositivo, y funcionar

independientemente del tipo del mismo. Se hicieron pruebas también en otros dispositivos, como los que se presentarán en las siguientes figuras, en donde se puede visualizar parte de la creación del registro de un accidente en un dispositivo Samsung A50, sistema operativo Android, en el cual la aplicación funciona de manera normal.

The image shows a mobile application interface on an Android device. At the top, the status bar shows the time 4:54, signal strength, Wi-Fi, and 33% battery. Below the status bar is a teal header with a hamburger menu icon, the text 'PROYECTO JULIUS', and a share icon. The main content area is titled 'Nuevo accidente' and features a progress indicator with five steps, where the first step is completed. Below the progress indicator is a section titled 'Descripción' containing several form fields, all of which are dropdown menus. The fields are: 'Tipo de calle', 'Número de carril' (with the value '1' displayed), 'Objeto en movimiento', 'Colisión con objeto fijo', 'Sin colisión', 'Condición de superficie', 'Tipo de superficie', and 'Condición luminica'. The bottom of the screen shows the start of a 'Resolución' field.

Figura 62. Pantalla de formulario en sección de Descripción, en dispositivo Android

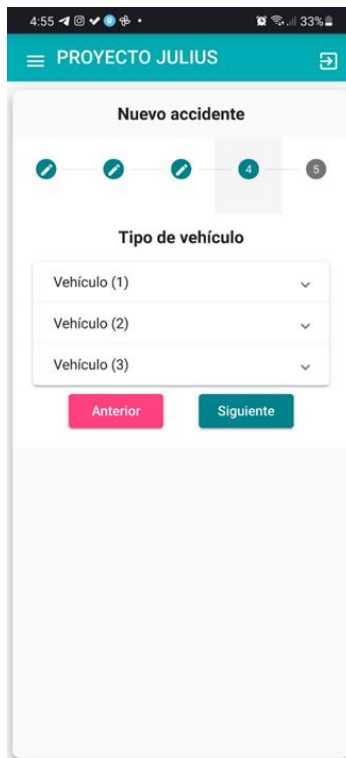


Figura 63. Pantalla de formulario en sección de Tipo de Vehículo, en dispositivo Android

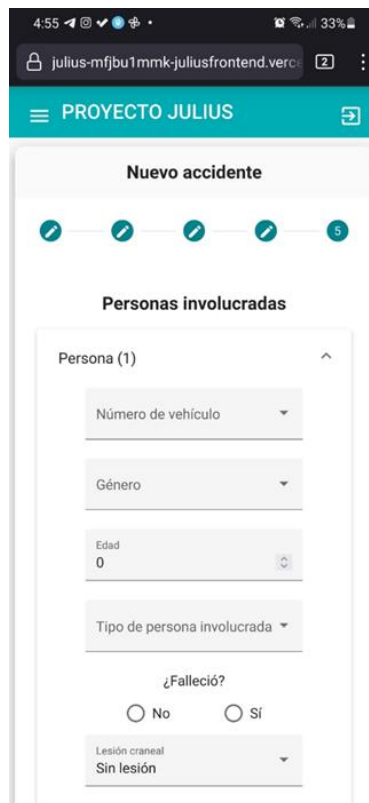


Figura 64. Pantalla de formulario en sección de Personas Involucradas, en dispositivo Android

Por otro lado, la aplicación cuenta con una configuración adicional que se realiza desde base datos y con una configuración en el API. Esta configuración es sobre las listas de valores y sus respectivas traducciones. Las listas de valores dentro del proyecto son muy importantes porque, como un aspecto de User Experience desarrollado, los componentes del formulario son solamente para seleccionar entre diferentes opciones y no permite que el usuario escriba texto.

Entonces, para el componente que es la selección de una opción, su información es solicitada a un endpoint del API, que realiza un query para generar la lista de opciones según el idioma solicitado. Por ejemplo, la aplicación le pide al endpoint que desea la lista de valores con llave “colores” con el código de lenguaje en el que la aplicación está configurada, y este devuelve el contenido de los colores que están almacenados en la base de datos para ser desplegados en la selección de colores dentro del formulario.

Nombre	Método	Función	Servicio
/api/login	POST	Realizar login a la aplicación con un usuario y contraseña.	Account.Service
/api/languages	PUT	Actualiza el valor del idioma del usuario para que cuando ingrese de nuevo se preserve el valor.	Account.Service
/api/events	POST	Crea un evento en su tabla, llenando todos los campos correspondientes	Form1.Service
/api/people	POST	Crea el espacio correspondiente para las n personas y las relaciona con el accidente.	People.Service

Nombre	Método	Función	Servicio
/api/people	DELETE	Elimina los registros de las personas y sus relaciones	People.Service
/api/cars	POST	Crea el espacio correspondiente para los n vehículos y los relaciona con el accidente.	Car.Service
/api/cars	DELETE	Elimina los registros de los vehículos y sus relaciones	Car.Service
/api/form3	PUT	Actualiza los datos del vehículo	Car.Service
/api/form4	PUT	Actualiza los datos de la persona	People.Service
/api/models	GET	Obtiene la información de modelos de los carros	ValueList.Service
/api/auxinfo	GET	Obtiene información auxiliar como colores, tipos de vehículo, marcas, países	ValueList.Service
/api/states	GET	Obtiene información de estados por país	ValueList.Service
/api/description	GET	Obtiene toda la información de un accidente	Consulta.Service

Nombre	Método	Función	Servicio
/api/descriptionByDate	GET	Obtiene toda la información de los accidentes entre 2 fechas	Menu.Service

Tabla 7. Listado de endpoints

Ya al implementar y unir el trabajo de los servicios con los componentes que los llaman y manejan, se tienen los siguientes resultados:

Para el Login se tienen los dos escenarios habituales, primero que el usuario y la contraseña sean correctas, esto devuelve un código 200 de Ok y los datos del usuario. Esto nos permite ingresar a la siguiente pantalla la cual es el menú principal y ya se colocan los datos obtenidos como el nombre y puesto además que la aplicación estará en el lenguaje guardado en su usuario.

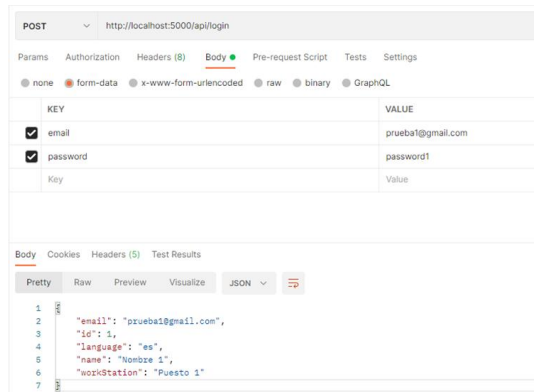


Figura 65. Login exitoso en Postman

En cuanto a las pruebas de usabilidad, se realizó una prueba importante en una reunión con el equipo de Provia, en la cuál asistieron directores, analistas de datos y personas encargadas de dirigir las patrullas para socorrer y atender accidentes de tránsito dentro del país. Dentro de esta reunión, se realizó un demo completo de toda la aplicación, se les explicó en su totalidad las funcionalidades de la aplicación y cómo se supone que debería usarse. También, se permitió que ellos pudieran navegar alrededor de la aplicación para que pudieran darnos retroalimentación.

Ahora, si los datos son incorrectos se devolverá un mensaje de error y el código 400 referenciando que fue una Bad Request por parte del usuario. El mensaje de error se recibe en la app y ya se despliega de forma informativa.

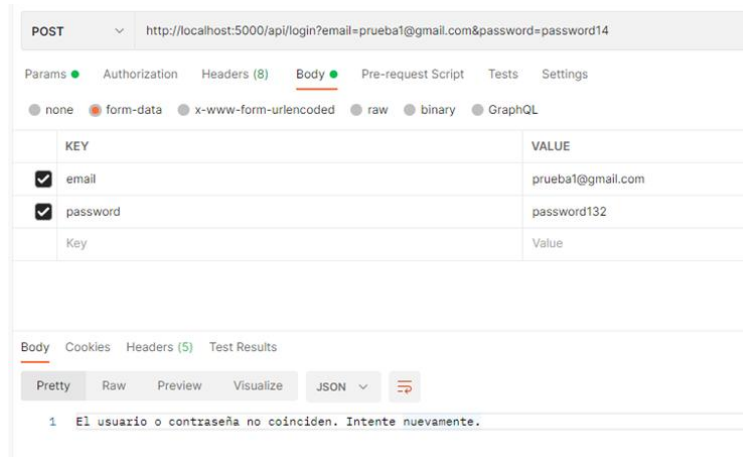


Figura 66. Login fallido en Postman

De la misma forma para la pantalla de datos generales, “frontend” se aseguró que no se pueda seleccionar 0 vehículos o 0 personas por lo que el Api y el servicio no deben de preocuparse de algún llamado incorrecto.

Datos generales

Cantidad de vehículos	2
Número de personas	2

[Anterior](#) [Siguiente](#)

Figura 67. Datos generales formulario

Algunas opciones de los dropdown se obtienen a través del “endpoint” de información auxiliar extraídos de la base de datos como los colores, las marcas, los modelos, etc...

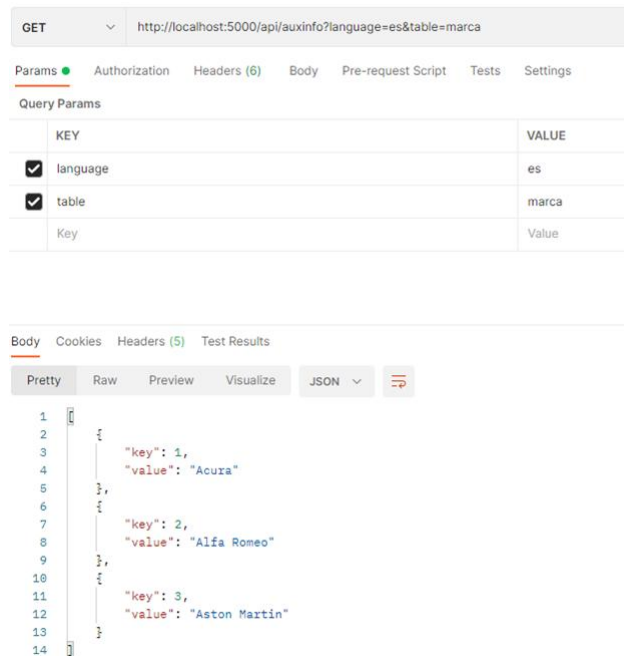


Figura 68. Get en Postman

Para guardar los datos de un vehículo realizar un PUT facilita que en el modo edición del formulario se pueda reutilizar esta función debido a la idempotencia de este.

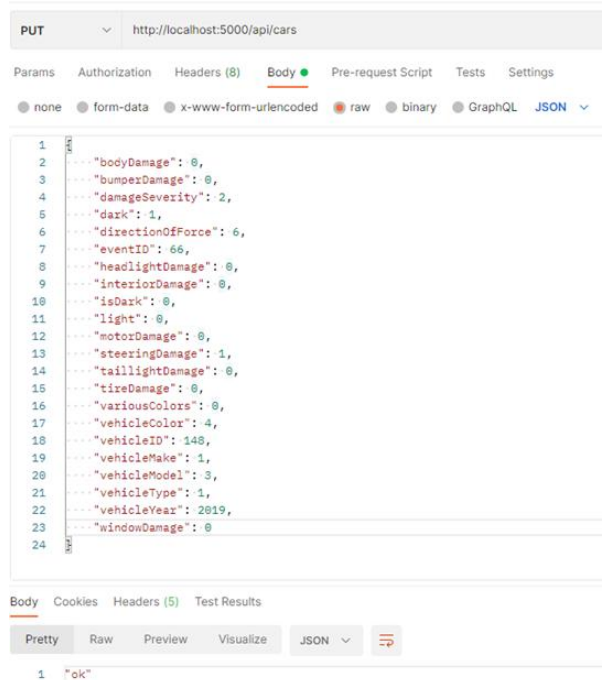


Figura 69. PUT en Postman

De la misma manera que los vehículos, las personas tienen un formulario individual.

Personas involucradas

Persona (1) ▼
Persona (2) ▼

Anterior Finalizar

Figura 70. Personas involucradas

Al inicio del formulario, se indica a qué vehículo se unirá la persona afectada o en su defecto si es un peatón. Obteniendo esta información de la cantidad de vehículos registrados en el evento anteriormente, con la consulta de descripción del API.

Persona (1) ^

Peatón

Vehículo 1

Vehículo 2

Personas involucradas

Figura 71. Menú personas involucradas

Cuando el usuario entra al evento, el servicio de descripción obtiene toda la información relacionada a ese accidente, sirviéndose en la pantalla para tener más información del mismo.

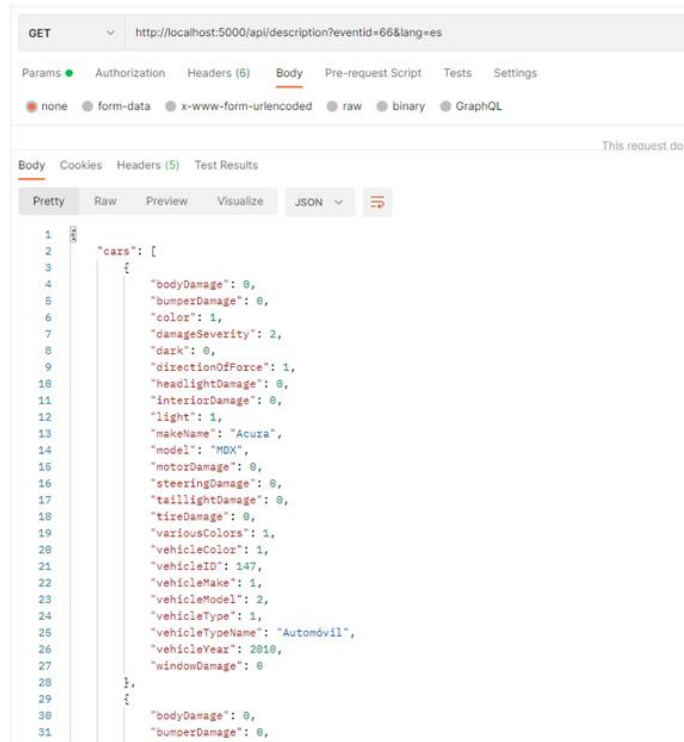


Figura 72. Get de evento en PostMan

Un pequeño reto de la edición fue el agregado o la removiada de vehículos y/o personas. Debido a que inicialmente se pensó para crear ambos objetos en una sola llamada al API, sin embargo a la hora de editar pueden existir casos donde se quiera remover un vehículo pero agregar personas o a la inversa. Por lo que se decidió separar esta funcionalidad en un “endpoint” para cada objeto así en la aplicación solo se valida que acción está realizando el usuario, ya que pueden ser cuatro: agregar vehículos y personas, agregar vehículos y remover personas, remover vehículos y agregar personas o remover vehículos y personas. Por lo que la separación de estos concluyó en diferentes llamadas de POST o DELETE.



Figura 73. Datos generales

Luego de la presentación de la aplicación, Jacob Caxaj resaltó los aspectos útiles y positivos dentro de la aplicación. Estos aspectos positivos fueron que pueda ser utilizado en multiplataforma: aplicación desktop y móvil para iOS y Android; también, la posibilidad de poder agregar diversa cantidad de personas y vehículos involucrados pues la mayor cantidad de personas en un accidente ha sido de 60 personas, mencionó Jacob. Por este último punto mencionado, se resaltó la alta utilidad de que múltiples colaboradores puedan ingresar simultáneamente información de diferentes personas y vehículos sin alterar lo que algún otro usuario esté llenando. Por esto mismo, se afirma que es una funcionalidad muy útil pues Jacob mencionó que Proviaal no es la única institución que asiste a un hecho vehicular, sino que también asisten bomberos, MP, PMT, PNC, entre otros, por lo cuál, que múltiples usuarios, o instituciones, puedan llenar la información que le interesa a cada uno es muy bueno.

Jacob habló sobre aspectos adicionales que le gustaría que pudieran estar dentro de la aplicación porque sirve como información que puede servir de referencia para resolver casos inconclusos en el pasado, o que permiten anclar a una persona y a un vehículo a algún hecho de tránsito. Se solicitó agregar como campos adicionales en la base de datos, información importante como el número de chasis, número de placa, número de dpi. La razón de esto es porque es necesario conocer quién estuvo involucrado en el accidente ya que, por ejemplo, pudo haber sido trasladado al hospital luego del accidente y para modificar el estado de salud de esta persona involucrada solamente podría hacerse por el número de dpi. O bien, en caso de un caso policial, se desee buscar evidencia en un vehículo, esta información solamente puede ser recabada por el número de placa y el número de chasis. Por lo mencionado anteriormente, la implementación realizada sobre buscar el historial de un accidente, filtrado por diferentes aspectos del accidente, resuelve el problema mencionado por Jacob de poder obtener la información de un accidente que pasó hace semanas, o meses, para resolver cualquier caso legal.



Figura 74. Visita a PROVIAL

Para pruebas con usuarios, se realizaron reuniones con personas para que pudieran dar su opinión acerca de la aplicación en general, como miraban la paleta de colores, el tipo de letra, como sentían la experiencia de utilizar los distintos subformularios para registrar un accidente, que escribieran comentarios y si consideraban alguna funcionalidad extra que debía agregarse a la aplicación. Lamentablemente se realizaron pruebas con pocos usuarios, debido a problemas con el backend que no permitía el uso de la aplicación de forma externa, por problemas con el certificado de seguridad y también por la falta de tiempo, solo se lograron obtener cinco respuestas, por lo que la muestra no es representativa a nivel nacional, sin embargo para efectos del presente informe estos resultados representan un marco de referencia para la identificación de puntos fuertes y vulnerables de la aplicación.

De los resultados obtenidos de la encuesta, se muestran los relacionados al formulario de registro de accidentes:

Evalúa el diseño del formulario de Datos Importantes
5 respuestas

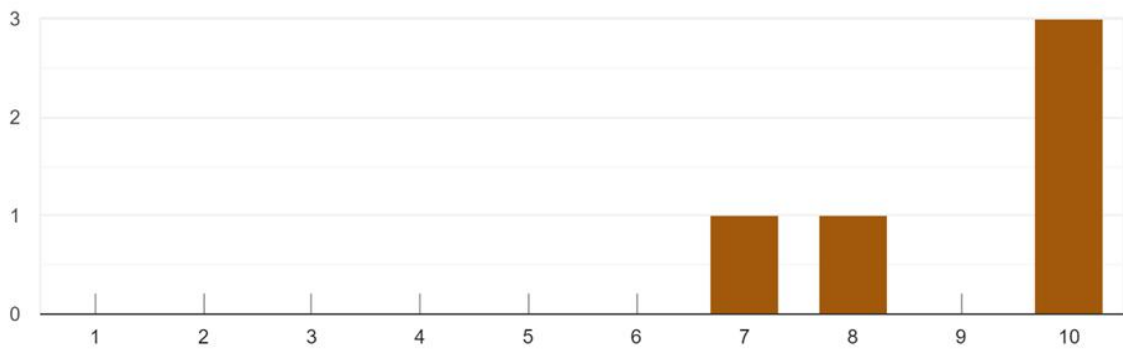


Figura 75. Gráfica de resultados de opinión sobre formulario de Datos Importantes

Evalúa el diseño del formulario de Descripción
5 respuestas

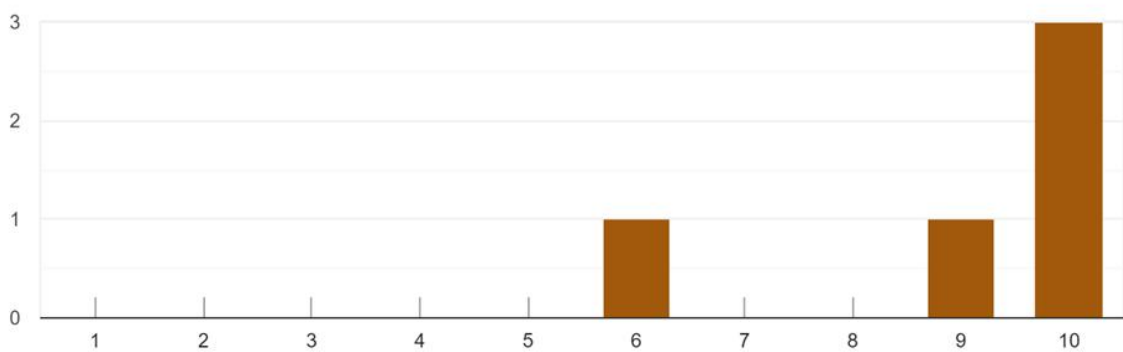


Figura 76. Gráfica de resultados de opinión sobre formulario de Descripción

Evalúa el diseño del formulario de Vehículos Involucrados

5 respuestas

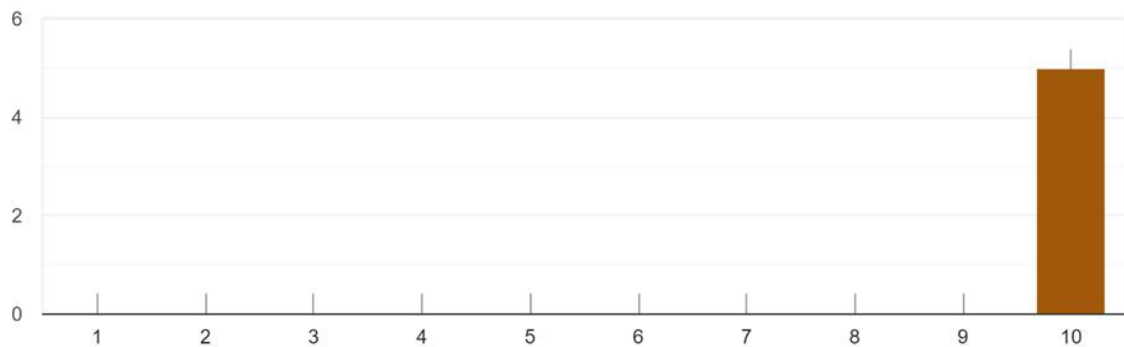


Figura 77. Gráfica de resultados de opinión sobre formulario de Vehículos Involucrados

Evalúa el diseño del formulario de Personas Involucradas

5 respuestas

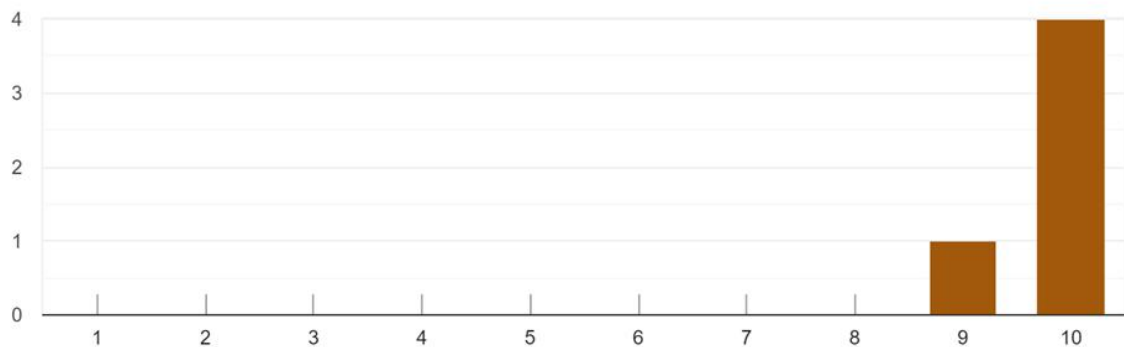


Figura 78 Gráfica de resultados de opinión sobre formulario de Personas Involucradas.

Resultados de encuesta:

https://docs.google.com/spreadsheets/d/1M5HYwUIYH21IOkU7gp86DUdk9_jkeNaGbp8hN7OBoaU/edit?usp=sharing

Para poder almacenar toda la información de los accidentes que pueden llegar a suceder, se creó una base de datos basada en los datos y recomendaciones dadas por la IRTAD por lo que se pasó

a crear una base de datos en postgreSQL la cual llevaba el orden de los elementos y contenían las tablas auxiliares las cuales ayudaban al frontend para la carga de los datos.

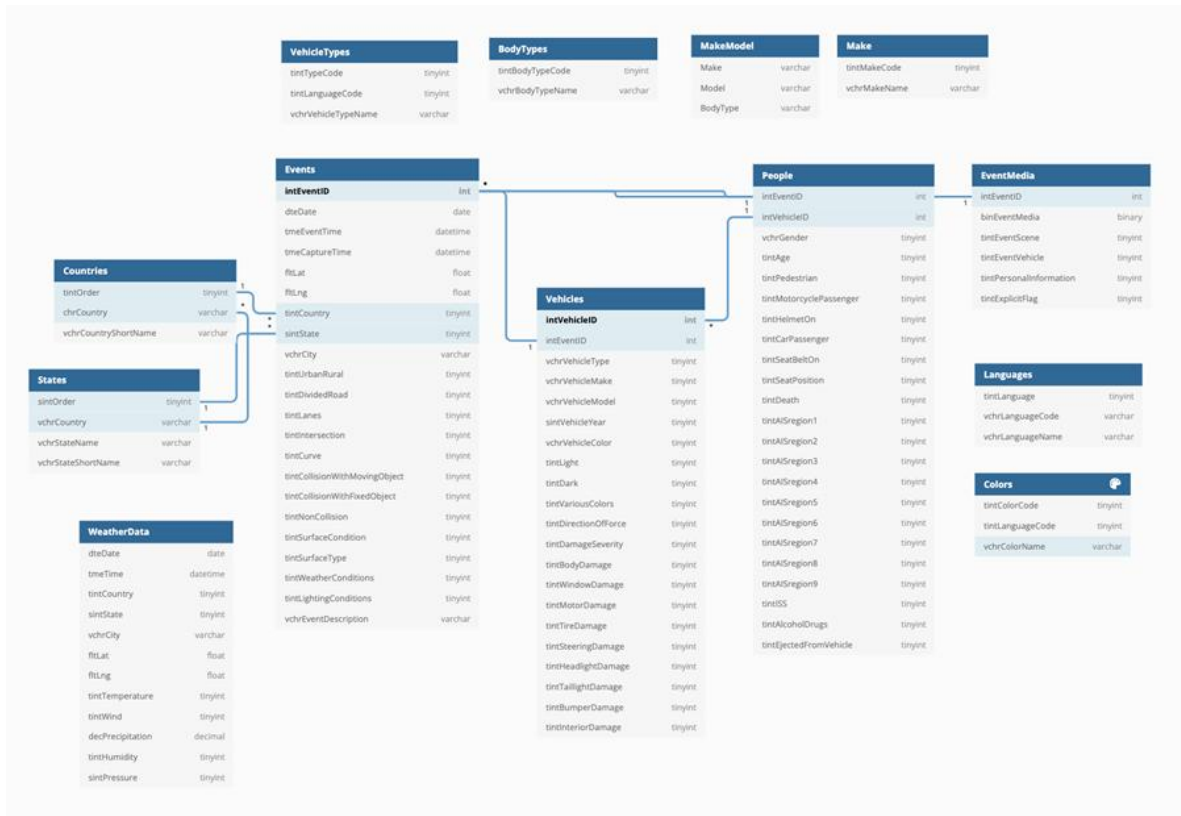


Figura 79. Diagrama entidad relación

Los endpoints fueron desarrollados en Python, en el framework de flask, esto debido a que era importante el ahorro de recursos y la simpleza del proyecto, por esto optamos por flask ya que es uno de los mejores frameworks de Python gracias a que es muy liviano en su versión original, no consume muchos recursos y es amigable para utilizar con postgreSQL.

```

45.229.43.249 - - [02/Dec/2021 21:48:05] "PUT /api/form2Cars HTTP/1.1" 500 -
45.229.43.249 - - [02/Dec/2021 21:48:48] "GET /api/login?email=prueba1@gmail.com&password=password1 HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:48:49] "GET /api/descriptionByDate?date1=2021-12-02&date2=2021-12-02&lang=es HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:48:51] "GET /api/auxinfo?language=es&table=vehicle types HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:48:51] "GET /api/auxinfo?language=es&table=marca HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:48:51] "GET /api/auxinfo?language=es&table=country HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:48:51] "GET /api/auxinfo?language=es&table=colors HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:48:52] "GET /api/states?country=Guatemala HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:49:14] "GET /api/descriptionByDate?date1=2021-12-02&date2=2021-12-02&lang=es HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:12] "GET / HTTP/1.1" 404 -
45.229.43.249 - - [02/Dec/2021 21:51:28] "GET /api/login?email=prueba1@gmail.com&password=password1 HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:29] "GET /api/descriptionByDate?date1=2021-12-02&date2=2021-12-02&lang=es HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:37] "GET /api/auxinfo?language=es&table=colors HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:37] "GET /api/auxinfo?language=es&table=country HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:37] "GET /api/auxinfo?language=es&table=marca HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:37] "GET /api/auxinfo?language=es&table=vehicle types HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:51:38] "GET /api/states?country=Guatemala HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:05] "OPTIONS /api/form1 HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:06] "POST /api/form1 HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:12] "OPTIONS /api/form2Cars HTTP/1.1" 200 -
Le hare update al evento 60
Agregando la cantidad de carros 1
45.229.43.249 - - [02/Dec/2021 21:52:12] "PUT /api/form2Cars HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:12] "OPTIONS /api/form2People HTTP/1.1" 200 -
Le hare update al evento 60
Agregando la cantidad de personas 1
AAAAAAAAAAAAAAAA es
45.229.43.249 - - [02/Dec/2021 21:52:13] "PUT /api/form2People HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:13] "GET /api/description?eventid=60&lang=es HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:29] "GET /api/descriptionByDate?date1=2021-12-02&date2=2021-12-02&lang=es HTTP/1.1" 200 -
45.229.43.249 - - [02/Dec/2021 21:52:34] "GET /api/descriptionByDate?date1=2021-11-25&date2=2021-12-02&lang=es HTTP/1.1" 200 -

```

Figura 80. Comunicación con el servidor

En la Figura 80 podemos ver las respuestas del servidor a los request que provienen de la aplicación, estos son los encargados de poder recibir los datos y poder guardarlos en la base de datos anteriormente mencionados, los endpoints principales fueron diseñados para poder interactuar directamente con la base de datos, los cuales eran la creación del evento, la obtención de la información del vehículo, la información de la persona, el inicio de sesión y la información que era parte de los requerimientos de la IRTAD.

Para poder realizar la predicción de accidentes se ideó un plan para poder llevar a cabo de una forma estandarizada y ordenada, esta fue de primero poder buscar información acerca de los accidentes para procesarla, después empezar a limpiar la información para quedarnos con solo lo necesario y eliminar toda la información basura, después de eso poder dividir la data en train, dev y test para poder así realizar la aplicación del smote para poder así aumentar la data y ver si existe una mejora al utilizar data sintética, por último aplicar los modelos de predicción para poder así obtener los resultados deseados.

Después de aplicar los distintos algoritmos de *machine learning* para la predicción de accidentes los cuales se realizaron con un *dataset* de accidentes los cuales cumplen con los estándares y con la información adecuada se obtuvieron los siguientes resultados.

El *dataset* que se utilizó contenía 285331 filas y 70 columnas, a las cuales se le aplicó un preprocesamiento para poder eliminar todas las filas las cuales tenían datos nulos, los cuales solo aplicarían ruido a los algoritmos de predicción. Después de este preprocesamiento el *dataset* final quedó en 118754 filas y 23 columnas. El cual después se dividió para poder realizar los entrenamientos, las distribuciones quedaron de la siguiente manera.

Train	Dev	Test
47501	47502	23751

Tabla 8. Distribución de dataset para entrenamiento

Todos los algoritmos se entrenaron para poder encontrar si los accidentes se dan más en áreas urbanas o rurales, siendo esta clase 1 y clase 2 respectivamente.

El primer algoritmo que se utilizó fue Catboost el cual se realizó sin optimización de hiperparámetros dio como resultado 0.62 en aprendizaje y 0.62 en test.

```
0:   learn: 0.6837743   test: 0.6838303 best: 0.6838303 (0)   total: 16.2ms   remaining: 146ms
9:   learn: 0.6227487   test: 0.6235132 best: 0.6235132 (9)   total: 181ms   remaining: 0us

bestTest = 0.6235131886
bestIteration = 9
```

Figura 81. Resultados de Catboost sin hyperparametros

Después de esto se utilizó la librería de Hpbansdter para poder realizar el entrenamiento de hiperparámetros para poder mejorar de manera automática los resultados obtenidos. Los resultados obtenidos con el entreno de hiperparámetros fue el siguiente

```
best result: 0.8272732092678013
best parameters: [0.02541651640962197, 10, 1.5608039498225423, 0.2222324144264605, 55]
```

Figura 82. Resultados con hyperparametros

Y los mejores parámetros que se obtuvieron son los siguientes

```
{'loss_function': 'Logloss',
 'iterations': 1200,
 'early_stopping_rounds': 30,
 'learning_rate': 0.02541651640962197,
 'depth': 10,
 'l2_leaf_reg': 1.5608039498225423,
 'random_strength': 0.2222324144264605,
 'border_count': 55}
```

Figura 83. Hyperparametros de catboost

Después de encontrar los mejores parámetros para el modelo se entrenó por última vez con esos parámetros para poder obtener los resultados que se presentan a continuación:

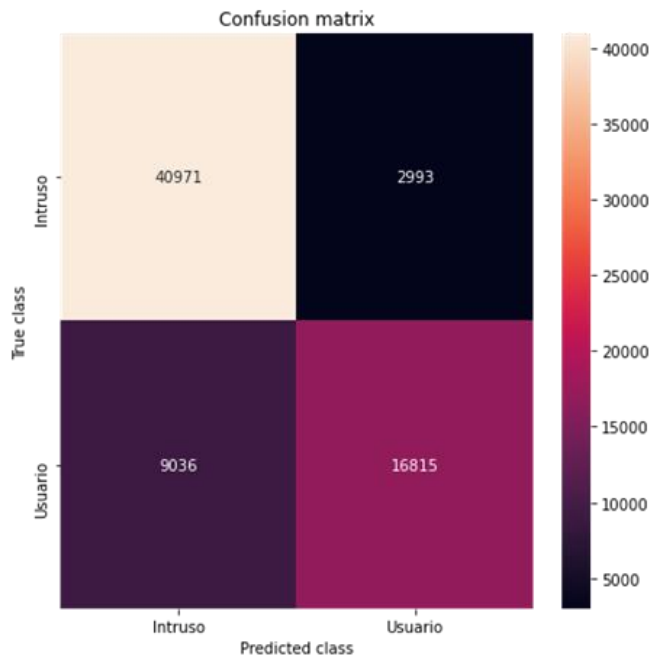


Figura 84. Matriz de confusión para train

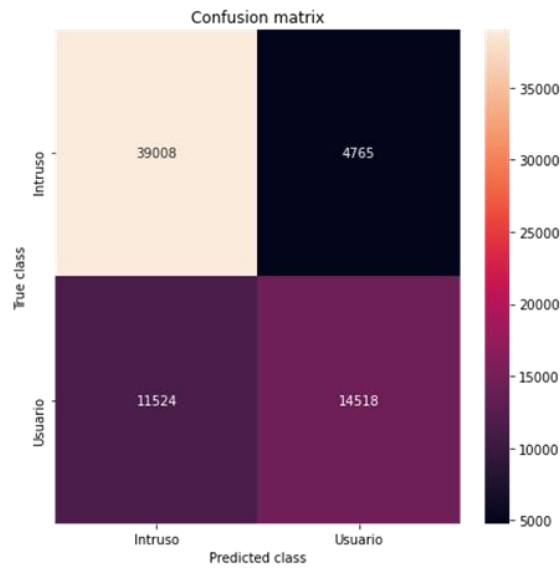


Figura 85. Matriz de confusión para dev

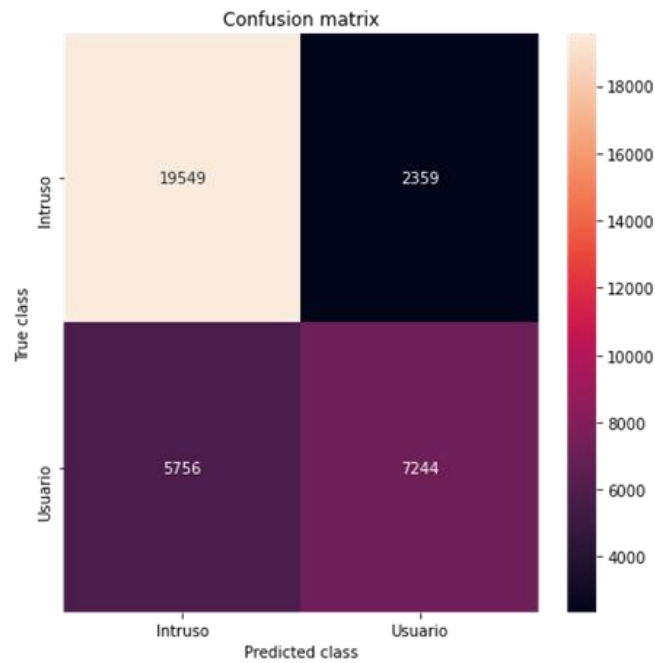


Figura 86. Matriz de confusión para test

Después de ver las matrices de resultados se puede analizar que el modelo es bastante aceptable tanto para la clase 1, pero es un poco bajo para la clase 2.

	precision	recall	f1-score	support
1	0.77	0.89	0.83	21908
2	0.75	0.56	0.64	13000

Figura 87. Resultados finales Catboost

El siguiente algoritmo que se utilizó fue Random Forest el cual se realizó sin optimización de hiperparámetros dio como resultado 0.66 en aprendizaje y 0.65 en test.

```
{'ccp_alpha': 0.004308676217740922, 'max_depth': 7, 'max_features': 21, 'min_samples_leaf': 39, 'min_samples_split': 26}
{'loss': -0.6611211082062947, 'info': {'test f1': 0.6595384867164912, 'train f1': 0.6602890291064523, 'val f1': 0.6611211082062947}}
```

Figura 88. Resultados Random Forest sin hiperparametros

Después de esto se utilizó la librería de Hpbansdter para poder realizar el entrenamiento de hiperparámetros para poder mejorar de manera automática los resultados obtenidos. Los resultados obtenidos con el entreno de hiperparámetros fue el siguiente

```
[config_id: (12, 0, 5) budget: 6.666667 loss: -0.8049340063159599
time_stamps: 752.1242249011993 (submitted), 752.1292135715485 (started), 757.084963798523 (finished)
info: {'test f1': 0.8038442476350551, 'train f1': 0.9633547412164715, 'val f1': 0.8049340063159599},
config_id: (12, 0, 5) budget: 20.000000 loss: -0.8163475619677821
time_stamps: 777.8825128078461 (submitted), 777.8865022659302 (started), 794.0562884807587 (finished)
info: {'test f1': 0.81480353335364, 'train f1': 0.9884723158230765, 'val f1': 0.8163475619677821},
config_id: (12, 0, 5) budget: 60.000000 loss: -0.8195464186653614
time_stamps: 807.3637006282806 (submitted), 807.3696851730347 (started), 856.070554971695 (finished)
info: {'test f1': 0.8202732789103767, 'train f1': 0.993265586221119, 'val f1': 0.8195464186653614}]
```

Figura 89. Resultados del entreno de hiperparámetros Random Forest

Y la mejor configuración de parámetros encontrada fue.

```
Best found configuration: {'ccp_alpha': 1.1829177176274374e-08, 'max_depth': 32, 'max_features': 22, 'min_samples_leaf': 1, 'min_samples_split': 2}
A total of 84 unique configurations where sampled.
A total of 113 runs where executed.
Total budget corresponds to 43.0 full function evaluations.
```

Figura 90. Hiperparámetros Random Forest

Después de encontrar los mejores parámetros se continuó a realizar un último entreno con los parámetros encontrados y los resultados fueron los siguientes

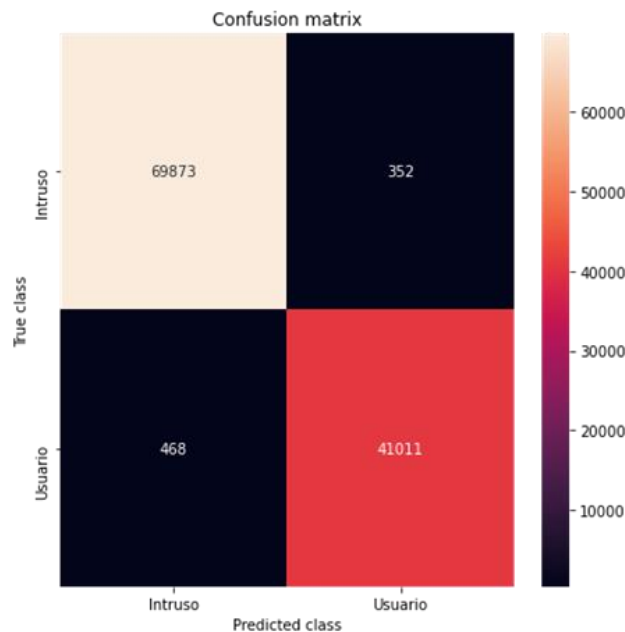


Figura 91. Matriz de confusión para train

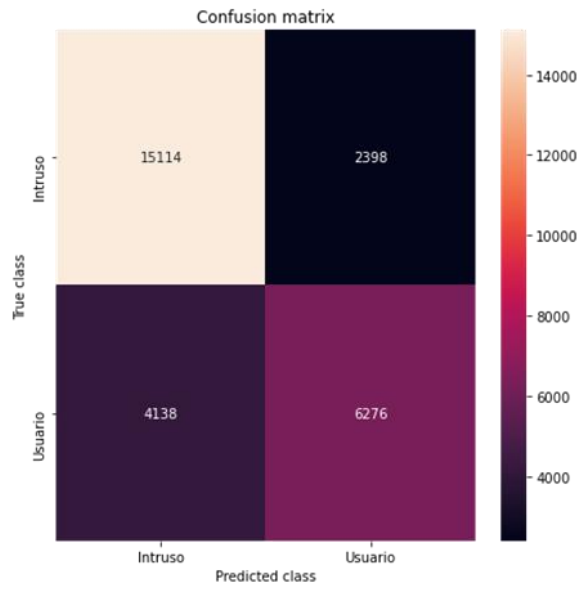


Figura 92. Matriz de confusión para dev

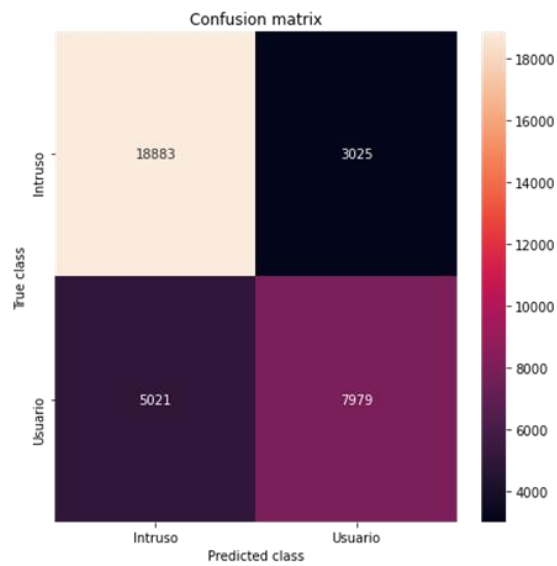


Figura 93. Matriz de confusión para test

Después de ver las matrices de resultados se puede analizar que el modelo es bastante aceptable tanto para la clase 1, pero es un poco bajo para la clase 2.

	precision	recall	f1-score	support
1	0.79	0.86	0.82	21908
2	0.73	0.61	0.66	13000

Figura 94. Resultados finales Random Forest

Después de obtener estos resultados se procedió a aplicar en los modelos data sintética para poder balancear las clases y así poder ver si es posible mejorar los resultados. Esta técnica se le aplicó a la subdivisión de train para que no se vean afectados los resultados de test. Las clases estaban distribuidas en 87737 para la clase uno mientras que para la clase dos se contaba con 51893.

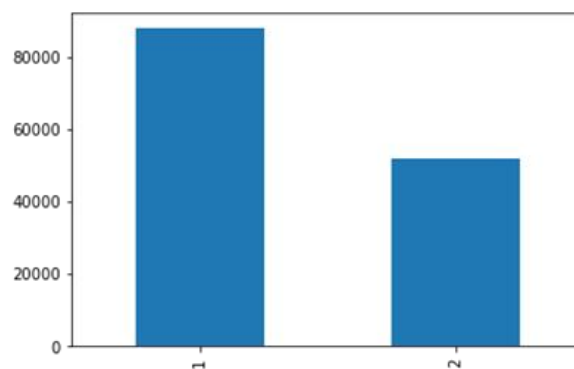


Figura 95. Distribución de las clases.

Como se puede ver las clases estaban desbalanceadas por lo que se pasó a aplicar la técnica de smote para poder aumentar la clase que estaba en desventaja, después de la aplicación las clases quedaron distribuidas de la siguiente manera, 87737 para la clase uno y la clase dos aumentó hasta llegar a 78963.

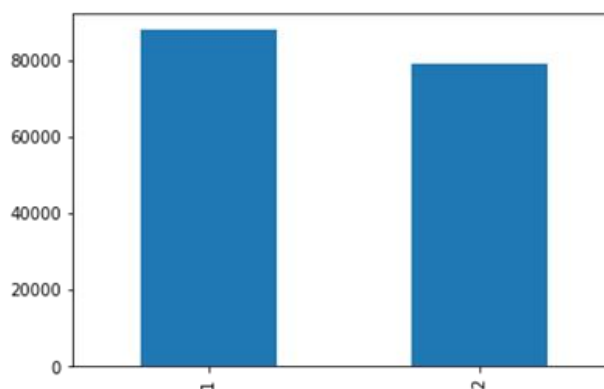


Figura 96. Distribución de clases con smote

Después de eso se volvió a realizar el entrenamiento con los algoritmos para ver los cambios de resultados con las clases ya balanceadas.

Catboost con *smote* el cual se realizó sin optimización de hiperparámetros dio como resultado 0.63 en aprendizaje y 0.63 en test.

```
0:   learn: 0.6849211      test: 0.6848916 best: 0.6848916 (0)   total: 18.5ms  remaining: 167ms
9:   learn: 0.6315715      test: 0.6313902 best: 0.6313902 (9)   total: 164ms   remaining: 0us

bestTest = 0.6313901947
bestIteration = 9
```

Figura 97. Resultados de Catboost con smote sin hiperparámetros

Después de esto se utilizó la librería de *skopt* para poder realizar el entrenamiento de hiperparámetros para poder mejorar de manera automática los resultados obtenidos. Los resultados obtenidos con el entreno de hiperparámetros fue el siguiente.

```
best result: 0.8080827675396056
best parameters: [0.1524219535295399, 4, 1.4390359709610292, 0.1, 512]
```

Figura 98. Resultados de smote con hiperparámetros

Y los mejores parámetros que se obtuvieron son los siguientes

```
{'loss_function': 'Logloss',
 'iterations': 1200,
 'early_stopping_rounds': 30,
 'learning_rate': 0.1524219535295399,
 'depth': 4,
 'l2_leaf_reg': 1.4390359709610292,
 'random_strength': 0.1,
 'border_count': 512}
```

Figura 99. Hiperparámetros de catboost con smote

Después de encontrar los mejores parámetros para el modelo se entrenó por última vez con esos parámetros para poder obtener los resultados que se presentan a continuación:

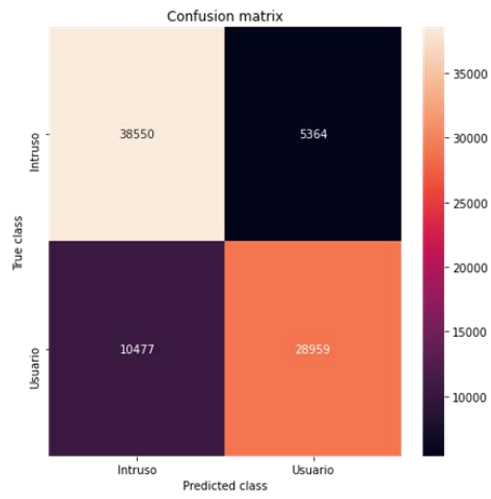


Figura 100. Matriz de confusión para train

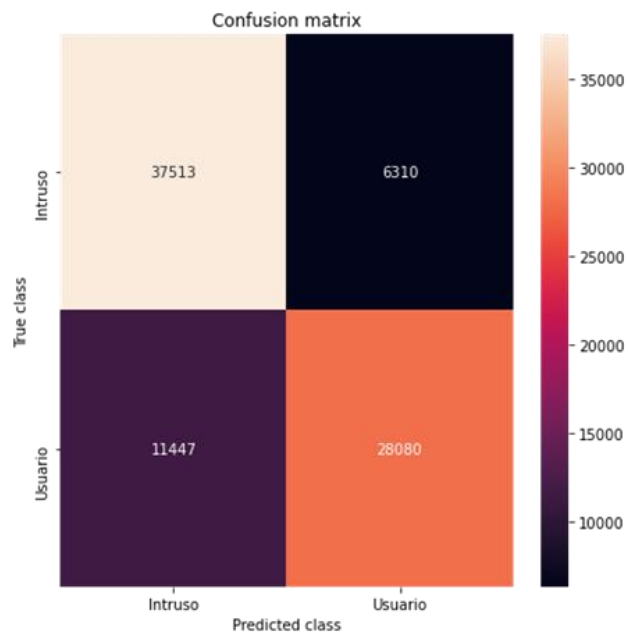


Figura 101. Matriz de confusión para dev

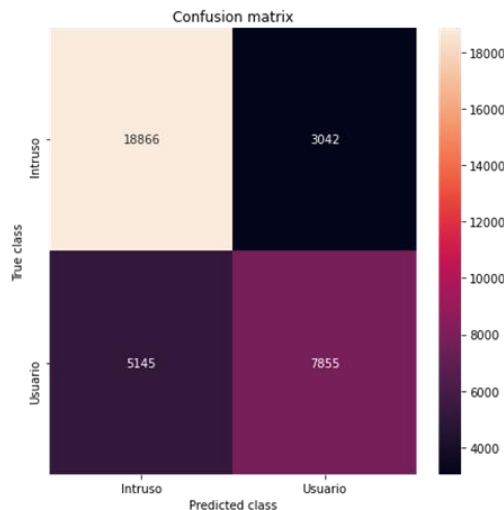


Figura 102. Matriz de confusión para test

Después de ver las matrices de resultados se puede analizar que el modelo es bastante aceptable tanto para la clase 1, pero es un poco bajo para la clase 2. Siendo estos mejores a los obtenidos sin *smote*

	precision	recall	f1-score	support
1	0.79	0.86	0.82	21908
2	0.72	0.60	0.66	13000

Figura 103. Resultados finales Catboost smote

Después se le aplicó *smote* Random Forest el cual se realizó sin optimización de hiperparámetros dio como resultado 0.79 en aprendizaje y 0.80 en test.

```
{'ccp_alpha': 1.1227630090235656e-05, 'max_depth': 38, 'max_features': 11, 'min_samples_leaf': 23, 'min_samples_split': 13}
{'loss': -0.7683992898716825, 'info': {'test f1': 0.7911684377223384, 'train f1': 0.8049040392143667, 'val f1': 0.7683992898716825}}
```

Figura 104. Resultados Random Forest con smote sin hiperparametros

Después de esto se utilizó la librería de Hpbansdter para poder realizar el entrenamiento de hiperparametros para poder mejorar de manera automática los resultados obtenidos. Los resultados obtenidos con el entreno de hiperparametros fue el siguiente

```
[config_id: (15, 0, 0) budget: 6.666667 loss: -0.7812722689611895
time_stamps: 664.1838524341583 (submitted), 664.1888418197632 (started), 667.2955358028412 (finished)
info: {'test f1': 0.7938604087287842, 'train f1': 0.9673591828086492, 'val f1': 0.7812722689611895},
config_id: (15, 0, 0) budget: 20.000000 loss: -0.7927026436530479
time_stamps: 686.5431005954742 (submitted), 686.5490839481354 (started), 696.4286723136902 (finished)
info: {'test f1': 0.8005389644813571, 'train f1': 0.9929820576879401, 'val f1': 0.7927026436530479},
config_id: (15, 0, 0) budget: 60.000000 loss: -0.7959926738310127
time_stamps: 714.9421825408936 (submitted), 714.9471695423126 (started), 744.9310185909271 (finished)
info: {'test f1': 0.8025565651166934, 'train f1': 0.9960968171421419, 'val f1': 0.7959926738310127}]
```

Figura 105. Resultados Random Forest con smote sin hiperparametros

Y la mejor configuración de parámetros encontrada fue.

```
Best found configuration: {'ccp_alpha': 8.064503199399813e-09, 'max_depth': 40, 'max_features': 11, 'min_samples_leaf': 1, 'min_samples_split': 2}
A total of 84 unique configurations where sampled.
A total of 113 runs where executed.
Total budget corresponds to 43.0 full function evaluations.
```

Figura 106. Hiperparametros Random Forest con smote

Después de encontrar los mejores parámetros se continuó a realizar un último entreno con los parámetros encontrados y los resultados fueron los siguientes

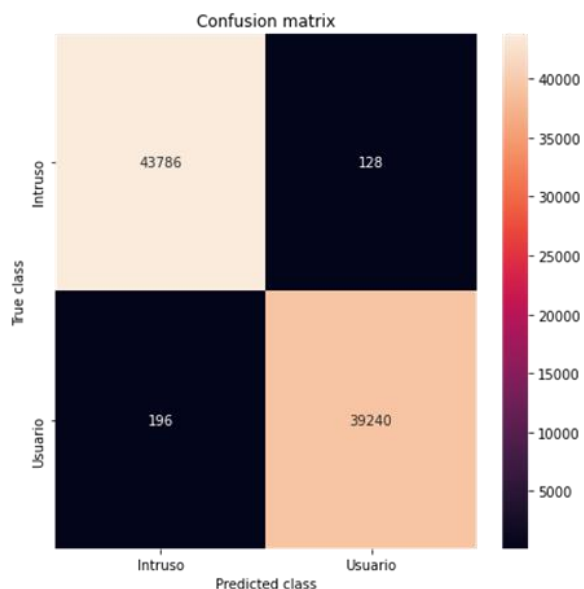


Figura 107. Matriz de confusión para train

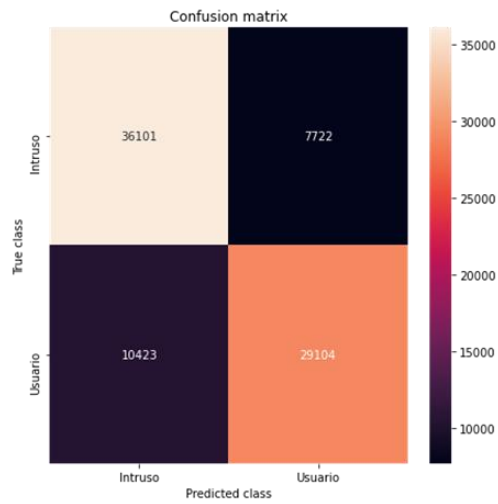


Figura 108. Matriz de confusión para dev

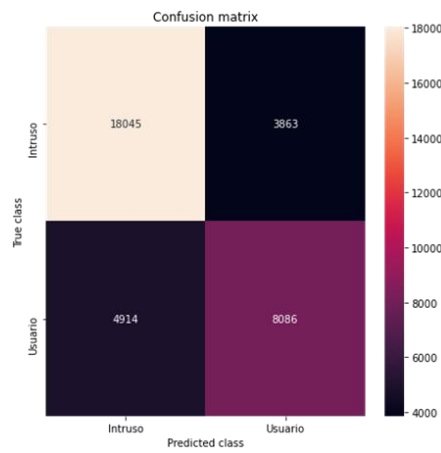


Figura 109. Matriz de confusión para test

Después de ver las matrices de resultados se puede analizar que el modelo es bastante aceptable tanto para la clase 1, pero es un poco bajo para la clase 2. Tomando en cuenta de que esta vez los resultados fueron peores a los dados sin smote por pocos valores.

	precision	recall	f1-score	support
1	0.79	0.82	0.80	21908
2	0.68	0.62	0.65	13000

Figura 110. Resultados finales Random Forest con smote

Se logró desarrollar los algoritmos de detección de posibles percances vehiculares a través del uso del GPS del dispositivo móvil y acelerómetros. Se realizaron pruebas tanto en emuladores como en dispositivos físicos. En los emuladores se establecieron distintas rutas, en las cuales las velocidades se fueron variando para poder simular lo más realista posible un percance vehicular.

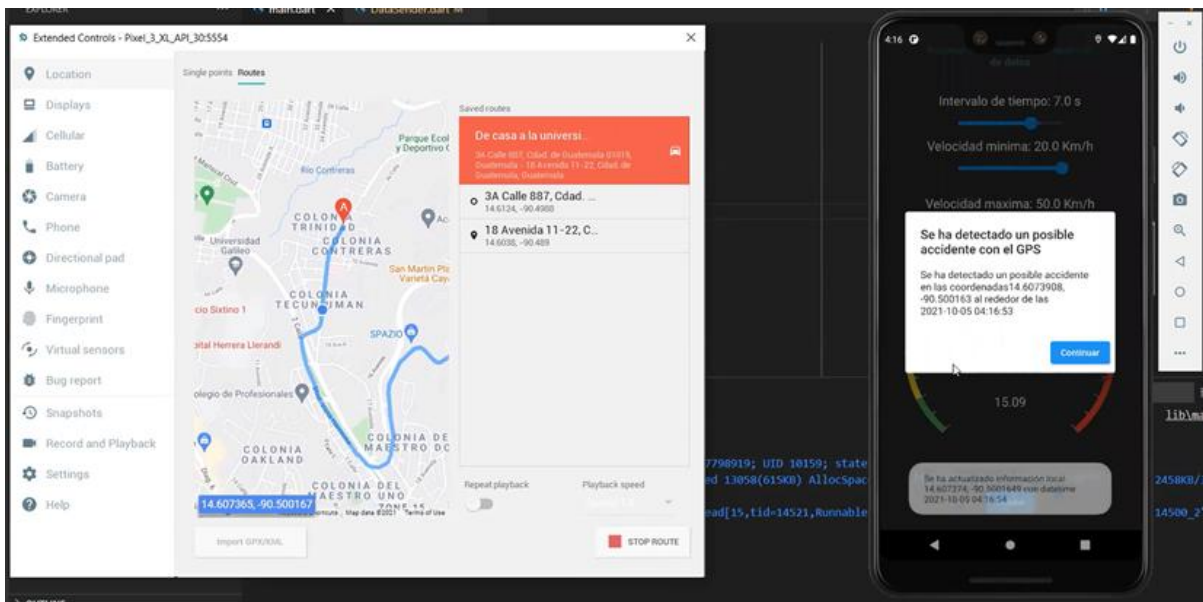


Figura 111. simulación de posible percance vehicular a través de información del GPS.

Para evaluar el funcionamiento de la detección a través de los acelerómetros se utilizó el teléfono Galaxy A02. El dispositivo fue lanzado en caída libre desde distintas distancias con diferentes aceleraciones necesarias en los ejes para detectar el posible percance vehicular.

Aceleración	Número de lanzamientos	Lanzamientos en los que se detectó un posible percance
10 m/s ²	10	10
20 m/s ²	10	10
30 m/s ²	10	0
40 m/s ²	10	0
50 m/s ²	10	0

Tabla 9. Tabla de lanzamientos de caída libre desde un metro de altura

Aceleración	Número de lanzamientos	Lanzamientos en los que se detectó un posible percance
10 m/s ²	10	10
20 m/s ²	10	10
30 m/s ²	10	0
40 m/s ²	10	0
50 m/s ²	10	0

Tabla 10. Tabla de lanzamientos de caída libre desde un metro y medio de altura.

Aceleración	Número de lanzamientos	Lanzamientos en los que se detectó un posible percance
10 m/s ²	10	10
20 m/s ²	10	10
30 m/s ²	10	0
40 m/s ²	10	0
50 m/s ²	10	0

Tabla 11. Tabla de lanzamientos de caída libre desde dos metros de altura

Se lograron realizar pruebas de carga al *endpoint* que se encarga de recibir y almacenar los datos de posibles percances vehiculares. Esto con el fin de simular la carga que puede llegar a recibir en caso de que se lance la aplicación a todo el público. Ya que estas pruebas se realizaron en un sitio web de manera gratuita que solo permitía hacer tests de un minuto de largo, se decidió hacer pruebas con 300 usuarios por minuto y 10000 usuarios por minuto (que era lo máximo que permitía).



Figura 112. Resultados de prueba de carga con 300 usuarios por minuto



Figura 113. Resultados de prueba de carga con 10,000 usuarios por minuto

Los resultados muestran que con 300 usuarios por minutos no existen errores durante la prueba. Lo cual demuestra que el endpoint puede soportar una carga realista de accidentes por segundo, además de que el tiempo promedio de respuesta por parte del servidor está por debajo de medio segundo. Por otra parte, cuando el endpoint recibe 10,000 solicitudes de usuarios por minuto muestra un porcentaje de error del 2.7% y con promedio de respuesta muy cercano al segundo. Este sigue siendo un buen resultado ya que está recibiendo aproximadamente 166 solicitudes por segundo.

VII: Conclusiones

1. La aplicación brinda facilidad en el llenado de información por múltiples usuarios en el formulario de accidentes vehiculares, pues permite que varios agentes institucionales puedan editar un accidente, previamente creado, al mismo tiempo sin alterar el proceso de otro.
2. La consulta de información es eficiente pues permite identificar los datos requeridos de forma simple y entendible, a través de buscadores y filtros que facilitan el proceso de encontrar un accidente para analizarlo o editarlo, según sea el caso.
3. El llenado de información es mucho más rápido y accesible pues la digitalización del proceso de captura de datos permite que los usuarios tengan una plataforma para registrar accidentes vehiculares porque actualmente la única alternativa que existe en Guatemala es hacerlo de forma escrita por parte del agente presente o una llamada a un colaborador de servicio al cliente.
4. La información ingresada a la aplicación está estandarizada y carece de inconsistencias gracias a que los datos están centralizados en una sola aplicación y base de datos, y que todas las opciones de captura de datos están diseñadas para que el usuario no deba escribir texto, sino que solo presione su elección correspondiente.
5. La funcionalidad de la pantalla de historial de accidentes permite localizar un evento fácilmente, a través de filtros, para luego utilizar la información del accidente registrado para realizar trámites o resolver casos que requieren tiempo.
6. Como se mencionó anteriormente, ya hay un estándar internacional para almacenar la información de un accidente, IRTAD, y para la aplicación se utilizaron los campos que nos proporcionó el stakeholder, los cuales se basan en el estándar IRTAD, y la aplicación tiene como intención ser utilizada por instituciones relacionadas a accidentes vehiculares en Guatemala, por lo que se fue a PROVIAL a mostrar la aplicación para mostrar su potencial, y se recibieron buenos comentarios sobre esta, entonces se puede decir con la evidencia que se cumple el objetivo de utilizar el estándar IRTAD en Guatemala para almacenar la información acerca de accidentes de tránsito.
7. La evidencia mostrada en los resultados, respecto a las imágenes de una aplicación con un formulario, dividido en subsecciones que poseen campos que se adecuan respecto al estándar IRTAD, para el registro de accidentes vehiculares, permite concluir que se cumplió el objetivo de desarrollar una aplicación que permita por medio de formularios, almacenar la información de un accidente vial.
8. Trabajar con una PWA, da muchas ventajas relacionadas a velocidad, al necesitar únicamente la creación de un código fuente para que funcionara en distintos dispositivos y que sea

instalable en computadoras o smartphones, lo que favorece el uso de Julius debido que esta aplicación será utilizada en distintos tipos de dispositivos.

9. La funcionalidad de subir imágenes a la aplicación debe ser considerada en una futura implementación, debido a la cantidad de información que se puede obtener de ellas.
10. Al implementar la funcionalidad de imágenes, en el futuro es importante considerar detalles clave, como el tamaño de las imágenes para que no haya problemas de lentitud al momento de subirlas.
11. La funcionalidad de cambio de lenguaje ya realizada será de gran utilidad si la aplicación se utiliza en otros países.
12. En la identificación de puntos vulnerables y fortalezas, hubo una reacción positiva de los formularios, lo cual indica que, si gustó el diseño de los formularios, exceptuando el formulario de Descripción que tuvo una reacción no muy positiva por parte de un usuario por lo cuál sería importante revisar con más usuarios si opinan de la misma manera del formulario de Descripción o fue una opinión subjetiva.
13. Y finalmente al observar los distintos resultados, como el aspecto de la velocidad, que se utiliza el estándar IRTAD, la opinión acerca de la aplicación con la retroalimentación recibida fue positiva, se puede concluir que el objetivo general de mejorar la recopilación de información sobre accidentes vehiculares en Guatemala con ayuda de un estándar internacional se cumplió.
14. En países como Guatemala es importante poder tener recursos tecnológicos para poder obtener información importante para su estudio, en este caso los accidentes viales
15. Se logró la correcta clasificación de accidentes dependiendo de sus características gracias a los algoritmos de clasificación siendo Random Forest el más efectivo
16. Es posible predecir accidentes de cualquier parte del mundo y adaptarlos al contexto de Guatemala ya que no dependen de datos geográficos concretos.
17. La optimización de hyperparametros es un proceso fundamental para poder obtener los mejores resultados de las clasificaciones
18. Los tratamientos para aumentar las clases desbalanceadas pueden ayudar a evitar el sobre entrenamiento de los algoritmos.
19. En caso de que se cuente con poca data también se puede aplicar *smote* para poder aumentar la cantidad de información en general y no solo las clases que se encuentran desbalanceadas
20. A través del GPS del dispositivo se logró determinar y enviar la información sobre la geolocalización, hora y fecha del dispositivo hacia la base de datos tan pronto se detecte un posible percance vehicular a través de los sensores del teléfono.
21. Haciendo uso de los dos sensores (acelerómetros y GPS) en conjunto se puede mejorar la detección de posibles percances vehiculares con el fin de reducir el número de falsos positivos.

22. Se llevó a cabo la implementación de las regulaciones impuestas por “International Traffic Safety Data and Analysis Group” de manera exitosa en el nuevo diseño de base de datos.
23. El nuevo diseño de base de datos permite relacionar tanto los incidentes como con involucrados y vehículos afectados.

VIII: Recomendaciones

1. Externalizar todas las listas de valores utilizadas en la aplicación hacia tablas en la base de datos o queries hacia esta misma, pues actualmente, algunas listas de valores las maneja la aplicación por lo que si se requiere un cambio es más complejo el proceso de hacerlo.
2. Programar en el módulo de backend la funcionalidad de silent-refresh y realizar la configuración correspondiente en la aplicación para tener la información del usuario autenticado aún si la aplicación se refresca.
3. Agregar la funcionalidad de almacenar, editar y visualizar fotografías pues, según PROVIAL, en un accidente de tránsito se toman alrededor de 16 fotos en las mismas posiciones de la dirección de fuerza de un accidente.
4. Implementar un algoritmo que identifique la creación de eventos repetidos, para que la aplicación los unifique automáticamente y así evitar información redundante.
5. Añadir al equipo, miembros con conocimiento de diseño gráfico para crear imágenes de carros y motos y utilizarlas para dar retroalimentación al usuario, como el número de asiento, lugar donde ocurrió el choque, entre otros.
6. Desarrollar una pantalla adicional o otro software que permita realizar simulaciones de huellas de frenado. en base a factores como velocidad, metros recorridos, clima, etc.
7. Realizar entrevistas con más instituciones que tienen relación con accidentes vehiculares, como PNC, PMT, MP, bomberos, Inacif, etc, para que puedan ser partícipes del proyecto.
8. Para ampliar el proyecto a aún más países se recomienda investigar los datos de las calles de cada ubicación, como por ejemplo qué tipos de calles pueden existir, qué diferentes tipos de vehículos circulan etc.
9. Implementar la funcionalidad de DPI y placa del vehículo debido a la importancia de estos datos al momento de registrar y buscar un accidente o una persona.
10. Se recomienda poder aplicar regresiones lineales con la información para así poder aplicar la búsqueda de accidentes aparte de las localizaciones
11. Se recomienda utilizar GPU para los algoritmos ya que ellos cuentan con soporte para las mismas y así poder reducir tiempo de entreno.
12. Se recomienda utilizar información de más países para poder comparar si las características de la locación de los accidentes son iguales en todos los países
13. Se recomienda utilizar la herramienta de anaconda para poder instalar las librerías de optimización de hiperparametros ya que algunas son difíciles de instalar.
14. Se recomienda la utilización de ONNX para poder exportar el algoritmo para poder hacerlo por tabla y poder aplicarlo en casi cualquier lenguaje de programación popular

15. Se recomienda implementar que la aplicación encargada de detectar posibles percances vehiculares pueda trabajar en segundo plano o incluso cuando no esté activa. Para esto se pueden agregar paquetes de Flutter los cuales permiten la ejecución de la aplicación en segundo plano para facilitar el proceso de desarrollo. Para lograr esto, se puede utilizar el paquete flutter_background, el cual permite trabajar a las aplicaciones incluso cuando están en segundo plano.
16. Para reducir la cantidad de falsos positivos se recomienda agregar el uso en conjunto de más sensores. Como podría ser a través de micrófonos, los cuales miden el nivel de decibelios constantemente. Para lograr esto, se puede incorporar el paquete noise_meter, el cual permite medir los decibelios del entorno a través del micrófono. Una vez implementado el paquete, al momento de detectar un posible percance vehicular con los sensores principales se podría evaluar también el nivel de sonido del entorno como apoyo para saber si sucedió un accidente. Así al momento de detectar un posible percance con los demás sensores, si no se alcanza o se supera un nivel determinado (recomendablemente 150 decibelios) se puede categorizar como un falso positivo.
17. Se recomienda agregarle persistencia de datos a la aplicación encargada de detectar posibles percances vehiculares, ya que si el dispositivo no cuenta con acceso a internet no guarda la información para luego ser enviada. En caso de que trate de hacer la solicitud al servidor y fallé, se pierden los datos. Y solo quedarían los datos locales, los cuales serían cambiados luego de tres segundos en caso de que no se destruya el dispositivo. En caso de que la información que se desee guardar localmente no sea crítica o de gran tamaño se puede seguir utilizando el plugin shared_preference. El cuál ya está instalado en el prototipo y guarda la última posición y tiempo del dispositivo. En caso de que se desee enviar información pendiente al servidor que se haya guardado con este plugin se recomienda crear una nueva clave la cual indique si existe información pendiente a enviar y otra clave la cual tenga la información que se desea enviar.
18. Este proyecto abre las puertas para poder recolectar información de los accidentes. No solo la geolocalización sino las últimas velocidades a las que viajó el usuario. Para lograr esto, se puede tomar las últimas velocidades registradas por dispositivo y enviarlas a la base de datos en el momento en el cual se detecte un posible percance. Lo cual permitiría alimentar otros proyectos de Machine Learning para mejorar la detección de posibles accidentes a través de la implementación de algoritmos como el Dynamic Time Warping el cual permite medir la similitud entre dos secuencias temporales.

IX: Bibliografía

- Abu-Salma, R., Al-Ali, H., Al-Merri, M., Aloul, F., Zualkernan, I. ibump: *Smartphone application to detect car accidents*. Agosto, 2014.
- Aguilar, D. (29 de septiembre de 2020). *SAT ha logrado recaudar 54% del impuesto de vehículos*:
- Arsys. (2016, junio 12) *.Por qué elegir PostgreSQL y llevarlo a Cloud*. Recuperado de: <https://www.arsys.es/blog/soluciones/postgresql-servidores/>
- AWS. (2021). *¿Qué es NoSQL?* Recuperado de AWS: <https://aws.amazon.com/es/nosql/>
- Baeza-Yates, R. (2005). Recuperado de Edgar F. Codd. <https://users.dcc.uchile.cl/~rbaeza/inf/codd.html>
- Blancarte, O. (2018). *Métodos HTTP (REST)*. Recuperado de: <https://www.oscarblancarteblog.com/2018/12/03/metodos-http-rest/>
- Borges, S. (2019). *Servidor PostgreSQL*. Recuperado de: https://blog.infranetworking.com/servidor-postgresql/#Que_es_PostgreSQL
- Bull, A. (2013). *Congestión de tránsito el problema y como enfrentarlo (1.a ed.)*. Recuperado de las Naciones Unidas. https://repositorio.cepal.org/bitstream/handle/11362/27813/6/S0301049_es.pdf
- Bull, A. (2013). *Congestión de tránsito el problema y cómo enfrentarlo (1.a ed.)*. Recuperado de la publicación de las Naciones Unidas. https://repositorio.cepal.org/bitstream/handle/11362/27813/6/S0301049_es.pdf
- Cambridge. (s.f.). *Framework*. Obtenido de Cambridge Dictionary: Recuperado de: <https://dictionary.cambridge.org/es/diccionario/ingles/framework>
- Castañon, M. (2019, septiembre 23), *Accidentes de tránsito han cobrado la vida de más de 11 mil personas*. Recuperado de la hora gt: <https://lahora.gt/accidentes-de-transito-han-cobradola-vida-de-mas-de-11-mil-personas-mariela-castanon-nacionales-lahora/>

- Castañón, M. (2019, septiembre 23), *Accidentes de tránsito han cobrado la vida de más de 11 mil personas*. Recuperado de: <https://lahora.gt/accidentes-de-transito-han-cobrado-la-vida-de-mas-de-11-mil-personas-mariela-castanon-nacionales-lahora/>
- CodingPotions. (2018). *Angular - Cómo crear servicios y llamadas a una API*. Recuperado de: <https://codingpotions.com/angular-servicios-llamadas-http>
- CONCEPTOS Y DEFINICIONES. (s. f.). *Instituto Nacional de Estadística y Censos*. Recuperado 21 de septiembre de 2021, de <https://www.inec.gob.pa/archivos/P4361CONCEPTOS.pdf>
- Cuervo, V. (2019). *¿Qué es Postman?* Recuperado de: <https://www.arquitectoit.com/postman/que-es-postman/>
- de Souza, I. (2021, 12 febrero). *Entiende las diferencias entre Front-End y Back-end en el ambiente de los sitios web*. Rock Content - ES. Recuperado de: <https://rockcontent.com/es/blog/front-end-y-back-end/>
- Decreto 1813 de 1994. (s.f.). Obtenido de Función Pública República de Colombia: Recuperado de: <https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=5246>
- Domingo, J. (2017). *¿Qué es FLASK?*. Recuperado de: <https://openwebinars.net/blog/que-es-flask/>
- Doshi, K. (2022, enero 7). *Audio Deep Learning Made Simple: Sound Classification, Step-by-Step*. Recuperado de Medium: <https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5>
- Dougherty, B., Albright, A., Thompson, C., White, J., Schmidt, D. *Using smartphones to detect car accidents and provide situational awareness to emergency responders*. Institute for Software Integrated Systems. 2016.
- Empleos TI. (s. f.). *¿Qué hace un Desarrollador Backend?*. Recuperado 23 de septiembre de 2021, de <https://empleosti.com.mx/enciclopedia-de-perfiles-ti/que-hace-un-desarrollador-backend>
- Eseme, S. (12 de agosto de 2021). *10 cosas que debes saber sobre Vue.js Frontend Framework*. Recuperado de Kinsta: <https://kinsta.com/es/blog/vue-js/>
- Etxarri, J. (2018, 5 septiembre). *¿Sabes qué es un «Service Worker»?*. Recuperado de Ekinbe:

<https://ekinbe.com/blog/2018/09/03/sabes-que-es-un-service-worker/>

Euroinnova. (s.f.). *Conoce ¿Qué Tipos De Accidentes De Tráfico Existen?* Recuperado de: Euroinnova Business School: <https://gt.euroinnova.edu.es/que-tipos-de-accidentes-de-trafico-existen>

Fenollosa, A. (2018). *¿Qué aporta Python para el desarrollo web?*. Recuperado de: <https://programadorwebvalencia.com/que-aporta-python-para-el-desarrollo-web/>

Fernández Y. (2019). *API: qué es y para qué sirve*. Recuperado de: <https://www.xataka.com/basics/api-que-sirve>

Gamarro, U. (2021, 16 enero). *Pese a la pandemia, el parque vehicular creció un 8% en 2020 en Guatemala*. Recuperado de Prensa Libre. <https://www.prensalibre.com/economia/pese-a-la-pandemia-parque-vehicular-crecio-8-en-2020/#:~:text=En%20total%20se%20sumaron%20315,8.2%25%2C%20seg%C3%BAn%20las%20estad%C3%ADsticas.>

Gamarro, U. (2021, 16 enero). *Pese a la pandemia, el parque vehicular creció un 8% en 2020 en Guatemala*. Recuperado de Prensa Libre: <https://www.prensalibre.com/economia/pese-a-la-pandemia-parque-vehicularcrecio-8-en2020/#:~:text=En%20total%20s%20sumaron%20315,8.2%25%2C%20seg%C3%BAn%20las%20estad%C3%ADsticas.>

García, F. (27 de Mayo de 2020). *Vue.js: Qué es y por qué usarlo como framework de referencia*. Recuperado de arsys: <https://www.arsys.es/blog/vuejs/>

García, O. (2019, enero 31). *Sabía que en siete de cada 10 vehículos que circulan en la ciudad solo viaja una persona*. Recuperado de Prensa Libre: <https://www.prensalibre.com/ciudades/guatemala-ciudades/sabia-que-en-siete-de-cada-10-vehiculos-que-circulan-en-la-ciudad-solo-viaja-una-persona/#:~:text=En%20la%20ciudad%20de%20Guatemala%20circulan%201.5%20millones%20de%20vehículos,10%20conductores%20com>

García, O. (31 de enero de 2019). *Sabía que en siete de cada 10 vehículos que circulan en la ciudad solo viaja una persona*. Recuperado de Prensa Libre:

<https://www.prensalibre.com/ciudades/guatemala-ciudades/sabia-que-en-siete-de-cada-10-vehiculos-que-circulan-en-la-ciudad-solo-viaja-una-persona/#:~:text=En%20la%20ciudad%20de%20Guatemala%20circulan%201.5%20millones%20de%20vehículos,10%20conductores%20com>

Gaunt, M. (s.f.). *Service Workers: An Introduction*. Recuperado de Web Fundamentals: <https://developers.google.com/web/fundamentals/primers/service-workers?hl=es>

Gazarov, P. (19 de diciembre de 2019). *What is an API? In English, please*. Recuperado de freeCodeCamp: <https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/>

Gélvez, S. (2009). *Evaluación de las escalas ISS y NISS en trauma penetrante grave*. Recuperado de: <http://www.scielo.org.co/pdf/rcci/v24n4/v24n4a4.pdf>

Google. (2010). *Introduction to Angular concepts*. Recuperado de Angular: <https://angular.io/guide/architecture>

Google. (2010). *What is Angular?* Recuperado de Angular: <https://angular.io/guide/what-is-angular>

Google. (2021). *Gestionar AppSheet en tu organización*. Recuperado de Ayuda del Administrador de Google Workspace: <https://support.google.com/a/answer/10100275?hl=es>

Google. (s.f.). *AppSheet*. Recuperado de Google Cloud: <https://cloud.google.com/appsheet?hl=es>

Hernández, K. Backend y Frontend, (s.f) *¿Qué es y cómo funcionan en la programación?* <https://www.servnet.mx/blog/backend-y-frontend-partes-fundamentales-de-la-programación-de-una-aplicación-web>

Herrera, C. (2020). *Ventajas de Angular para crear aplicaciones web*. Recuperado de: <https://blogueroopro.com/blog/ventajas-de-angular-para-crear-aplicaciones-web>

Instituto Nacional de Estadística Guatemala. (s.f) *¿Qué es el INE?*. Recuperado del INE: <https://www.ine.gob.gt/ine/institucion/>

Instituto Nacional de Estadística Panamá. (s.f) *Conceptos y definiciones Accidentes*. Recuperado de INEC: <https://www.inec.gob.pa/archivos/P4361CONCEPTOS.pdf>

International Transport Forum / OECD. (2021). *International Traffic Safety Data and Analysis Group (IRTAD)*. Recuperado de International Transport Forum: <https://www.itf-oecd.org/IRTAD>

International Transport Forum. (2018). *Road Safety Annual Report 2018*. Recuperado de Road Safety Annual Report 2018: https://www.gub.uy/unidad-nacional-seguridad-vial/sites/unidad-nacional-seguridad-vial/files/documentos/publicaciones/Reporte%20Anual%202018_IRTAD.pdf

International Transport Forum. (2018). *Road Safety Annual Report 2018. Road Safety Annual Report 2018*. Recuperado de ITF: https://www.gub.uy/unidad-nacional-seguridad-vial/sites/unidad-nacional-seguridad-vial/files/documentos/publicaciones/Reporte%20Anual%202018_IRTAD.pdf

IRTAD. (2021, 12 febrero). *International Traffic Safety Data and Analysis Group (IRTAD)*. ITF. Recuperado 1 de noviembre de 2021, de <https://www.itf-oecd.org/IRTAD>

ITF. (2021, 12 febrero). *International Traffic Safety Data and Analysis Group (IRTAD)*. Recuperado de: <https://www.itf-oecd.org/IRTAD>

ITF. (2021, 12 febrero). *International Traffic Safety Data and Analysis Group (IRTAD)*.
Recuperado de:

Jackson, B. (2021, enero 26). *Las 7 Ventajas Principales de Escoger Google Cloud Hosting*.
Recuperado de:

Jackson, B. (2021, enero 26). *Las 7 Ventajas Principales de Escoger Google Cloud Hosting*.
Recuperado de: <https://kinsta.com/es/blog/google-cloud-hosting/>

Lázaro, D. (2016). *Códigos de estado HTTP*. Recuperado de: <https://diego.com.es/codigos-de-estado-http>

Lazywill. (2019, noviembre 6). *Choque de vehículos: ¿Cómo se absorbe la energía mecánica?*.
Recuperado de: <https://www.enmotive.com/es/choque-vehiculos-energia-mecanica/>

Marqués, H. (2020). *¿Qué es TypeScript y para qué sirve?*. Recuperado de: <https://marquesfernandes.com/es/tecnologia-es/what-and-typescript-and-for-that-serves/>

- Martinez, J. (Septiembre, 2020). *Random Forest (Bosque Aleatorio) Combinando árboles*. Recuperado de: <https://www.iartificial.net/random-forest-bosque-aleatorio/>
- Molina, O. A., & Sotelo, S. (2016). *Construcción de aplicativos de programación por restricciones en Microsoft Solver Foundation y Windows Azure*. *Scientia et technica*, 21(4), 336-341.
- Moringo, L. (26 de noviembre de 2019). *Progressive Web Apps: Making app-like experiences in the browser*. Recuperado de Medium: <https://medium.com/samsung-internet-dev/progressive-web-apps-making-app-like-experiences-in-the-browser-a15bd388963b>
- MuleSoft. (2021). *What is an API? (Application Programming Interface)*. Recuperado de MuleSoft: <https://www.mulesoft.com/resources/api/what-is-an-api>
- Mutz, V. (2020, Octubre 24). *PMT de la capital reporta 35 accidentes de tránsito diarios desde el fin del toque de queda*. Recuperado de República.gt: <https://republica.gt/2020/10/24/pmt-de-la-capital-reporta-35-accidentes-de-transito-diarios-desde-el-fin-del-toque-de-queda>
- Mutz, V. (24 de octubre de 2020).. *PMT de la capital reporta 35 accidentes de tránsito diarios desde el fin del toque de queda*: Recuperado de República.gt: <https://republica.gt/2020/10/24/pmt-de-la-capital-reporta-35-accidentes-de-transito-diarios-desde-el-fin-del-toque-de-queda/>
- Oleksandr, S. (13 de septiembre de 2021). *What Is an API: Concept and Architecture Types Explained on Real-Life Examples*. Obtenido de Cleveroad: <https://www.cleveroad.com/blog/what-is-an-api>
- OMS. (2021, junio 21). *Road traffic injuries. Road Traffic Injuries*. Recuperado de: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>
- Oracle (2021). *¿Qué es una base de datos? Recuperado de Oracle*: <https://www.oracle.com/mx/database/what-is-database/>
- Oracle. (2021.). *¿Qué es una base de datos relacional?* Recuperado 17 de octubre de 2021, de <https://www.oracle.com/ar/database/what-is-a-relational-database/>
- Peritos de Accidentes. (31 de julio de 2018). *Factores que intervienen en un accidente de tráfico*. Recuperado de Peritos de Accidentes: <https://www.peritosdeaccidentes.com/factores-que-intervienen-en-un-accidente-de-trafico/>

- PowerData, (2019). *¿Qué es un gestor de datos y para qué sirve?*. Recuperado de: <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirve>
- Quiroz, J. (2003). *El modelo relacional de bases de datos*. DoAnalytics. Recuperado 23 de noviembre de 2021, de https://www.doanalytics.net/Documents/Modelo_Relacional.pdf
- Raffino, M. (2020 junio 24). *Base de datos*. Recuperado de: <https://concepto.de/base-de-datos/>
- Recuperado de la hora.gt: <https://lahora.gt/sat-ha-logrado-recaudar-54-del-impuesto-de-vehiculos/#:~:text=Para%20este%202020%2C%20el%20total,311%20mil%20825%20vehículos%20comerciales.>
- Red Hat. (2020). *What is a REST API?*. Recuperado de: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- RedHat. (s. f.). *¿Qué es una API?*. Recuperado 24 de octubre de 2021, de <https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>
- Richard, S., & LePage, P. (24 de febrero de 2020). *What are Progressive Web Apps?* Recuperado de web.dev: <https://web.dev/what-are-pwas/>
- Richard, S., & LePage, P. (24 de febrero de 2020). *What makes a good Progressive Web App?* Recuperado de web.dev: <https://web.dev/pwa-checklist/>
- Rodríguez, E. (2020, 22 octubre). *¿Qué es un framework? Descubre todas sus ventajas*. Recuperado de Seo studio.: <https://www.seoestudios.es/blog/que-es-un-framework/>
- Rosa, J. (2018). *¿Qué es REST? Conoce su potencia*. Recuperado de: <https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>
- Salim Alwan, Z., Ali Alshaibani, H. M. *Car accident detection and notification system using smartphone*. IJCSMC, Vol. 4:620–635, Abril, 2015.
- Shi, J., Shen, J., Zhu, M., Wheeler, K., Lu, B., Kenney, B., . . . Xiang, H. (2019). *A new weighted injury severity scoring*. Recuperado de Injury Epidemiology: <https://injejournal.biomedcentral.com/articles/10.1186/s40621-019-0217-8>

- SILVA CARES, M. I. (2017). *Diseño y construcción de un sistema para detectar, localizar y caracterizar accidentes automovilísticos*. [tesis pregrado, Universidad de Chile]. Recuperado de: <http://repositorio.uchile.cl/handle/2250/149475>
- Singh, V. (15 de mayo de 2021). *What is a Framework? [Definición] Tipos of Frameworks*. Recuperado de hackr.io: <https://hackr.io/blog/what-is-frameworks>
- Singhal, P. (31 de agosto de 2021). *Frontend vs Backend*. Recuperado de GeeksforGeeks: <https://www.geeksforgeeks.org/frontend-vs-backend/>
- Sneha R.S., Gawande A. D. (2013). "Crash Notification System for Portable Devices", *International Journal of Advanced Computer Technology (IJACT)*. PDF recuperado de <https://www.ijact.org/ijactold/volume2issue3/IJ0230022.pdf>
- State of JS. (2020). *Front-end Frameworks*. Recuperado de StateofJS: <https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/>
- Stefaniak, P. (26 de julio de 2019). *¿Qué es Backend y Frontend?* Recuperado de Descubre Comunicación: <https://descubrecomunicacion.com/que-es-backend-y-frontend/>
- Stewart, L. (22 de julio de 2021). *Front End Development vs Back End Development*. Recuperado de Course Report: <https://www.coursereport.com/blog/front-end-development-vs-back-end-development-where-to-start>
- tíThink (2018). *Framework o librerías: ventajas y desventajas*. Recuperado de: <https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas>
- Turner , H. (2001). *WreckWatch: Automatic Traffic Accident Detection and Notification with Smartphones*. Recuperado de: https://www.researchgate.net/publication/220133799_WreckWatch_Automatic_Traffic_Accident_Detection_and_Notification_with_Smartphones
- Valero, J. (2017, febrero). *Ruido rosa para minimizar las consecuencias de un accidente*. *Hipertextual*. Recuperado de: <https://hipertextual.com/2017/02/ruido-rosa-accidente-coche>

- Vidal, M. (2019, 6 noviembre). *PWA: Qué son y cómo funcionan las Progressive Web Apps*. Thinking for Innovation. Recuperado de: <https://www.iebschool.com/blog/progressive-web-apps-analitica-usabilidad/>
- Vue.js. (21 de octubre de 2020). *Introduction*. Recuperado de Vue.js: <https://vuejs.org/v2/guide/>
- Wales, M. (8 de diciembre de 2020). *3 Web Dev Carreers Decoded: Front-End vs Back-End vs Full Stack*. Recuperado de Udacity: <https://www.udacity.com/blog/2020/12/front-end-vs-back-end-vs-full-stack-web-developers.html>
- Wong, S. (2016, noviembre 24). *Understanding data augmentation for classification: when to warp?*. Recuperado de: <https://arxiv.org/pdf/1609.08764.pdf>
- Wong, S. (2016, noviembre 24). *Understanding data augmentation for classification: when to warp?*. Recuperado de: <https://arxiv.org/pdf/1609.08764.pdf> IIHS-HLDI. (2021, junio 28). *New crash tests show modest speed increases can have deadly consequences*. IIHS-HLDI *Crash Testing and Highway Safety*. Recuperado de: <https://www.iihs.org/news/detail/new-crash-tests-show-modest-speed-increases-can-have-deadly-consequences>

X: Anexos



Figura 114. Visita en las instalaciones de call center de provial.



Figura 115. Mapa de cobertura de patrullas de provial.



Figura 116. Equipo de provial y equipo de proyecto Julius.