
Implementación de infraestructura cloud con políticas de seguridad para un sistema integrador de aplicaciones y modelos de inteligencia artificial de orientación vocacional

José Daniel Gómez Cabrera



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Implementación de infraestructura cloud con políticas de
seguridad para un sistema integrador de aplicaciones y
modelos de inteligencia artificial de orientación
vocacional**

Trabajo de graduación en modalidad de modelo de trabajo profesional
presentado por
José Daniel Gómez Cabrera
para optar al grado académico de Licenciado en Ingeniería en Ciencias de la
Computación y Tecnologías de la Información

Guatemala

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



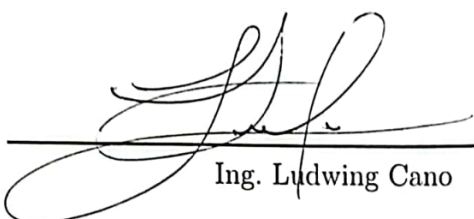
**Implementación de infraestructura cloud con políticas de
seguridad para un sistema integrador de aplicaciones y
modelos de inteligencia artificial de orientación
vocacional**

Trabajo de graduación en modalidad de modelo de trabajo profesional
presentado por
José Daniel Gómez Cabrera
para optar al grado académico de Licenciado en Ingeniería en Ciencias de la
Computación y Tecnologías de la Información

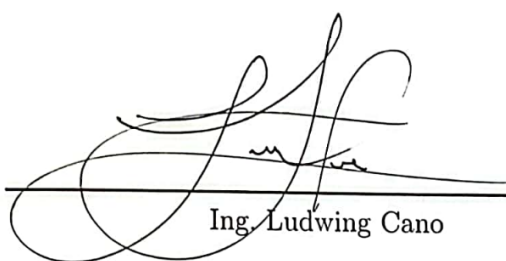
Guatemala

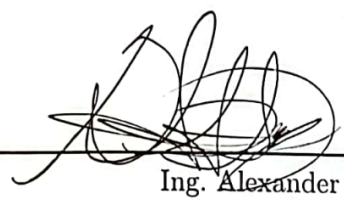
2025

Vo.Bo.:

(f) 
Ing. Ludwing Cano

Tribunal Examinador:

(f) 
Ing. Ludwing Cano

(f) 
Ing. Alexander Bolaños

Fecha de aprobación: Guatemala, 10 de Noviembre de 2025.

En el transcurso de mi formación académica, he sido testigo de la compleja travesía que significa para muchos estudiantes latinoamericanos la elección de una carrera universitaria. Esta decisión, que marca profundamente el futuro profesional y personal, frecuentemente se toma con información limitada y sin las herramientas adecuadas para un análisis introspectivo efectivo. El presente trabajo surge de la convergencia entre mi pasión por las tecnologías de la información y mi interés por contribuir a resolver problemáticas educativas en nuestra región. La orientación vocacional, tradicionalmente relegada a métodos convencionales y generalistas, representa un campo propicio para la aplicación de tecnologías avanzadas como la inteligencia artificial y el análisis de datos, ofreciendo así soluciones personalizadas y accesibles a través de sistemas integradores que conecten múltiples servicios y plataformas. Este proyecto no habría sido posible sin el apoyo invaluable de mi asesor, el Ing. Ludwig Cano, cuya guía y conocimiento han sido fundamentales para navegar los aspectos técnicos y metodológicos de esta investigación, especialmente en el desarrollo de la infraestructura cloud y los sistemas integradores de aplicaciones. Asimismo, extendiendo mi agradecimiento a la Facultad de Ingeniería de la Universidad del Valle de Guatemala por proporcionar el entorno académico que ha estimulado mi desarrollo profesional y ha fomentado mi interés por la aplicación de soluciones tecnológicas a problemas sociales relevantes. Agradezco también a mis compañeros de estudio, cuyas perspectivas y colaboración enriquecieron significativamente este trabajo. Sus aportes durante las sesiones de discusión y retroalimentación han sido invaluable para refinar los conceptos y metodologías aplicados en este proyecto, particularmente en el diseño de la arquitectura del sistema integrador. Finalmente, dedico este trabajo a mi familia, cuyo apoyo incondicional ha sido el pilar fundamental durante toda mi formación académica. Su comprensión, paciencia y aliento constante han sido esenciales para culminar con éxito esta etapa de mi vida profesional. Confío en que este proyecto contribuirá de manera significativa al campo de la orientación vocacional en América Latina, aprovechando el potencial de las tecnologías emergentes y los sistemas integradores para democratizar el acceso a herramientas de orientación vocacional de calidad, adaptadas a las necesidades específicas de nuestra región, proporcionando una infraestructura robusta y escalable que conecte eficientemente múltiples servicios y plataformas educativas.

Agradezco profundamente a Dios por haberme dotado de la capacidad, sabiduría y fortaleza necesarias para culminar este trabajo de graduación. Su guía constante iluminó mi camino durante los momentos de mayor complejidad técnica y emocional, permitiéndome perseverar ante los desafíos inherentes al desarrollo de sistemas integradores en la nube. Cada línea de código, cada decisión arquitectónica y cada problema resuelto son testimonio de las bendiciones recibidas a lo largo de este proceso educativo en Universidad del Valle de Guatemala.

A mi familia y amigos, les dedico mi más sincero reconocimiento por su apoyo incondicional durante toda esta etapa. Sus palabras de aliento en los momentos de incertidumbre fueron el impulso que necesitaba para continuar. En especial, agradezco a mis padres por su invaluable soporte económico y emocional, que no solo hizo posible mi formación académica, sino que también me brindó la tranquilidad necesaria para

concentrarme plenamente en este proyecto. Su sacrificio y confianza en mi potencial fueron fundamentales para alcanzar esta meta, y este logro es tanto mío como suyo.

Finalmente, extendiendo mi gratitud a mis compañeros de la carrera de Ingeniería en Ciencias de la Computación, quienes compartieron conmigo las largas jornadas de estudio y desarrollo. A Adrian Rodríguez, Esteban Donis y Abner García, les agradezco profundamente por su colaboración en la ideación de soluciones a problemas técnicos complejos, por las sesiones de brainstorming que enriquecieron mi perspectiva, y por su disposición a discutir enfoques alternativos cuando la complejidad del sistema integrador parecía insuperable. Mención especial merece Adrian Rodríguez, cuya participación fue clave en la resolución de desafíos críticos de la arquitectura serverless, la implementación de políticas de seguridad y la optimización del algoritmo de recomendaciones. Sus conocimientos técnicos y su capacidad para abordar problemas desde ángulos innovadores contribuyeron significativamente a la calidad final de este trabajo. Su amistad y colaboración trascienden lo académico, y son un ejemplo del valor del trabajo colaborativo en la ingeniería de software.

José Daniel Gómez Cabrera

Guatemala, Noviembre de 2025

Prefacio	IV
Lista de figuras	XI
Lista de cuadros	XIII
Resumen	XIV
1. Introducción	1
2. Objetivos	3
2.1. Objetivo general	3
2.2. Objetivos específicos	3
3. Justificación	4
4. Alcance	6
5. Marco teórico	7
5.1. Orientación vocacional en Guatemala	7
5.2. Elección de carrera universitaria	8
5.3. La preparación para los estudios universitarios	8
5.4. Desafíos en Guatemala	8
5.5. ¿Qué es Inteligencia Artificial?	9
5.6. Sistemas de recomendación vocacional con propósitos educativos basados en Inteligencia Artificial	9
5.7. ¿Qué es un sistema integrador de aplicaciones?	10
5.8. ¿Qué es Backend?	10
5.9. Application Programming Interface	10
5.10. ¿Qué es Cloud?	11
5.11. API Gateway	12
5.12. Escalabilidad horizontal	12
5.13. Consumo de modelos de Inteligencia Artificial	13
5.14. Sistema integrador para la plataforma inteligente de orientación vocacional	13
5.15. Función de la arquitectura en un sistema integrador	13
5.16. Arquitectura cloud en un sistema integrador	14
5.17. Tipos de arquitecturas para sistemas integradores en la nube	15
5.17.1. Arquitectura basada en microservicios	15

5.17.2. Arquitectura serverless	15
5.17.3. Arquitectura contenerizada con Kubernetes	16
5.18. Seguridad de un sistema integrador de aplicaciones	16
5.18.1. Importancia de la arquitectura segura	17
5.18.2. Autenticación	17
5.18.3. Seguridad de la información personal	18
5.18.4. Implementación de seguridad de la información personal	19
5.19. Virtual Private Cloud	19
5.19.1. Web Application Firewall	20
5.20. Políticas de seguridad	21
5.20.1. Desarrollo de políticas	22
5.20.2. JSON Web Tokens	22
5.20.3. Infraestructura de red	22
5.20.4. Encriptación	22
5.20.5. Cumplimiento con regulaciones legales	23
5.21. Proveedores de nube	23
5.21.1. Amazon Web Services	23
5.21.2. Microsoft Azure	24
5.21.3. Google Cloud Platform	24
5.22. Lenguajes de programación	24
5.22.1. Golang	24
5.22.2. Java	24
5.22.3. Python	25
5.22.4. JavaScript	25
5.23. MongoDB	25
5.23.1. Amazon DocumentDB	26
6. Metodología	27
6.1. Enfoque metodológico	27
6.1.1. Justificación del enfoque ágil	27
6.1.2. Estructura de sprints	27
6.2. Selección y justificación de tecnologías	28
6.2.1. Selección de proveedor cloud	28
6.2.2. Selección de arquitectura	29
6.2.3. Selección de lenguaje de programación	30
6.2.4. Selección de Base de Datos	31
6.2.5. Selección de sistema de autenticación	33
6.3. Fases del desarrollo	34
6.3.1. Fase 1: investigación y análisis comparativo	34
6.3.2. Fase 2: diseño arquitectónico	34
6.3.3. Fase 3: configuración de infraestructura base	36
6.3.4. Fase 4: implementación de componentes Core	37
6.3.5. Fase 5: desarrollo de módulos funcionales	39
6.3.6. Fase 6: pruebas de rendimiento y validación	42
6.4. Herramientas y recursos	45
6.4.1. Herramientas de desarrollo	45
6.4.2. Herramientas de despliegue	46
6.4.3. Herramientas de monitoreo	46
6.4.4. Recursos de infraestructura	46
6.5. Gestión del proyecto	46
6.5.1. Backlog del producto	46
6.5.2. Definición de completitud (Definition of Done)	47
6.5.3. Gestión de riesgos	47
6.5.4. Cronograma estimado	48

6.6.	Métricas de éxito	48
6.6.1.	Métricas de proceso	48
6.6.2.	Métricas de producto	48
6.6.3.	Métricas de calidad	49
6.7.	Consideraciones éticas	49
6.7.1.	Protección de datos de menores	49
6.7.2.	Sesgos en recomendaciones de IA	49
6.7.3.	Accesibilidad e inclusión	49
6.8.	Limitaciones de la metodología	50
7.	Resultados	51
7.1.	Prototipo	51
7.1.1.	Infraestructura cloud implementada	51
7.1.2.	Arquitectura de la solución	57
7.1.3.	Módulos implementados	58
7.1.4.	Estadísticas del proyecto	66
7.1.5.	Cronología de desarrollo	66
7.1.6.	Tecnologías y herramientas utilizadas	67
7.1.7.	Casos de uso implementados	69
7.1.8.	Resultados de rendimiento	70
7.1.9.	Escalabilidad lograda	71
7.1.10.	Seguridad implementada	71
7.1.11.	Lecciones aprendidas	72
7.1.12.	Trabajo futuro	73
7.2.	Rendimiento del prototipo	73
7.2.1.	Comportamientos de peticiones	74
7.3.	Pruebas de carga	74
7.3.1.	Perfil de usuario Fixed Load Testing	74
7.3.2.	Obtención del feed Fixed Load Testing	76
7.3.3.	Obtención de carreras Fixed Load Testing	77
7.3.4.	Obtención de foros Ramp Up Load Testing	79
7.4.	Pruebas de estrés	80
7.4.1.	Obtener carrera Ramp Up Stress Test	81
7.4.2.	Editar comentario Ramp Up Stress Test	82
7.5.	Pruebas de picos de carga	84
7.5.1.	Perfil de usuario Spike Test	84
7.5.2.	Obtener resultados de examen Spike Test	85
7.5.3.	Obtener carreras Peak Test	87
7.6.	Pruebas de resistencia	89
7.6.1.	Obtener carreras Fixed Soak Test	89
7.7.	Presupuesto del prototipo	90
7.7.1.	Costos de servicios de AWS	91
7.7.2.	Costos de computación	92
7.7.3.	Costos de red y seguridad	92
7.7.4.	Costos de Base de Datos	92
7.7.5.	Costos de serverless y APIs	93
7.7.6.	Costos de servicios auxiliares	93
7.7.7.	Costos de autenticación externa	93
7.7.8.	Proyección de costos por escala	94
7.7.9.	Optimizaciones de costo implementadas	94
7.7.10.	Resumen presupuestario	94
7.7.11.	Análisis de costo-beneficio	94

8. Discusión de resultados	96
8.1. Análisis del prototipo	96
8.2. Análisis de rendimiento	97
8.2.1. Comportamiento bajo carga constante	97
8.2.2. Comportamiento del sistema bajo estrés	98
8.2.3. Respuesta a picos súbitos	98
8.2.4. Resistencia prolongada	99
8.3. Cumplimiento del objetivo general	99
8.4. Cumplimiento de objetivos específicos	101
8.4.1. Diseño de arquitectura de infraestructura cloud	101
8.4.2. Desarrollo de sistema integrador con seguridad en datos personales	101
8.4.3. Configuración e implementación de políticas de seguridad	102
8.4.4. Medición y optimización de velocidad de respuesta	105
8.4.5. Validación de escalabilidad horizontal	106
8.4.6. Garantía de flujo seguro de datos con modelos de Inteligencia Artificial externos	108
8.5. Consideraciones de costo y efectividad	110
8.6. Limitaciones identificadas y áreas de mejora	111
8.7. Sostenibilidad y escalamiento futuro	113
8.8. Implicaciones para la orientación vocacional en Guatemala	114
9. Conclusiones	116
10.Recomendaciones	118
11.Bibliografía	120
12.Anexos	129
12.1. Introducción al esquema de la base de datos	129
12.1.1. Tecnologías utilizadas	129
12.1.2. Colecciones de la base de datos	129
12.2. Colecciones principales de la base de datos	130
12.2.1. Colección: users	130
12.2.2. Colección: careers	131
12.2.3. Colección: courses	132
12.2.4. Colección: tags	133
12.2.5. Colección: cards	133
12.2.6. Colección: forums	134
12.2.7. Colección: interactions	134
12.2.8. Colección: saveditems	135
12.2.9. Colección: test_items	136
12.2.10.Colección: test_results	136
12.2.11.Colección: testimonies	138
12.2.12.Colección: scoring_rules	139
12.3. Relaciones entre colecciones	139
12.3.1. Diagrama de relaciones conceptuales	139
12.3.2. Estrategias de modelado	140
12.4. Índices recomendados	140
12.4.1. Colección: users	141
12.4.2. Colección: careers	141
12.4.3. Colección: forums	141
12.4.4. Colección: cards	141
12.4.5. Colección: interactions	141
12.4.6. Colección: test_results	141
12.4.7. Colección: saveditems	141
12.4.8. Colección: test_items	142

12.4.9. Colección: testimonios	142
12.5. Consideraciones de seguridad	142
12.5.1. Encriptación de datos	142
12.5.2. Autenticación y autorización	142
12.5.3. Validación de datos	142
12.6. Patrones de acceso comunes	142
12.6.1. Flujo de orientación vocacional	142
12.6.2. Flujo de exploración de contenido	143
12.6.3. Flujo de participación en foros	143
12.7. Optimizaciones y consideraciones de performance	143
12.7.1. Desnormalización estratégica	143
12.7.2. Agregaciones pre-calculadas	143
12.7.3. Estrategias de Caching	143
12.8. Escalabilidad y crecimiento	144
12.8.1. Particionamiento de datos	144
12.8.2. Archivado de datos históricos	144
12.8.3. Monitoreo de crecimiento	144
12.9. Migraciones y versionado	144
12.9.1. Control de versiones del esquema	144
12.9.2. Estrategias de migración	145
12.10 Conclusiones	145
12.10.1 Fortalezas del diseño	145
12.10.2 Áreas de atención	145
12.10.3 Recomendaciones finales	145

Lista de figuras

7.1. Infraestructura del prototipo del sistema integrador	51
7.2. Tendencia del tiempo de respuesta para obtener el perfil de usuario	75
7.3. Tasa de peticiones enviadas por segundo para obtener perfil de usuario	75
7.4. Distribución de error respecto al tiempo para obtener perfil de usuario	76
7.5. Tendencia del tiempo de respuesta para obtener el feed del usuario	77
7.6. Tasa de peticiones enviadas por segundo para obtener feed del usuario	77
7.7. Distribución de error respecto al tiempo para obtener feed del usuario	77
7.8. Tendencia del tiempo de respuesta para obtener carreras disponibles	78
7.9. Tasa de peticiones enviadas por segundo para obtener carreras disponibles	78
7.10. Distribución de error respecto al tiempo para obtener carreras disponibles	79
7.11. Tendencia del tiempo de respuesta para obtener foros progresivamente	80
7.12. Tasa de peticiones enviadas por segundo para obtener foros progresivamente	80
7.13. Distribución de error respecto al tiempo para obtener foros progresivamente	80
7.14. Tendencia del tiempo de respuesta para obtener carrera	81
7.15. Tasa de peticiones enviadas por segundo para obtener carrera	82
7.16. Distribución de error respecto al tiempo para obtener carrera	82
7.17. Tendencia del tiempo de respuesta para editar comentario	83
7.18. Tasa de peticiones enviadas por segundo para editar comentario	83
7.19. Distribución de error respecto al tiempo para editar comentario	83
7.20. Tendencia del tiempo de respuesta para obtener perfil de usuario con incremento súbito de tráfico	84
7.21. Tasa de peticiones enviadas por segundo para obtener perfil de usuario con incremento súbito de tráfico	85
7.22. Distribución de error respecto al tiempo para obtener perfil de usuario con incremento súbito de tráfico	85
7.23. Tendencia del tiempo de respuesta para obtener resultados de examen con incremento súbito de tráfico	86
7.24. Tasa de peticiones enviadas por segundo para obtener resultados de examen con incremento súbito de tráfico	86
7.25. Distribución de error respecto al tiempo para obtener resultados de examen con incremento súbito de tráfico	87
7.26. Tendencia del tiempo de respuesta para obtener carreras con incremento progresivo de tráfico	88
7.27. Tasa de peticiones enviadas por segundo para obtener carreras con incremento progresivo de tráfico	88
7.28. Distribución de error respecto al tiempo para obtener carreras con incremento progresivo de tráfico	88

7.29. Tendencia de tiempo, throughput y distribución de error para obtención de carreras ante una carga constante durante un período de tiempo extenso	89
--	----

Lista de cuadros

6.1. Matriz de criterios para evaluación de tecnologías	28
6.2. Distribución de responsabilidades en arquitectura híbrida	30
6.3. Criterios de aceptación para validación del sistema	45
6.4. Recursos de infraestructura AWS utilizados	46
6.5. Matriz de riesgos técnicos	48
6.6. Cronograma estimado por fases	48
7.1. Desglose de costos por servicio de AWS (período mensual)	91
7.2. Costos de recursos de computación	92
7.3. Costos de infraestructura de red y seguridad	92
7.4. Costos de almacenamiento de base de datos	92
7.5. Costos de funciones serverless y APIs (dentro de capa gratuita)	93
7.6. Costos de servicios auxiliares y soporte	93
7.7. Costos de servicios de autenticación externa (Clerk)	93
7.8. Proyección de costos según escalamiento de usuarios	94
7.9. Optimizaciones de costo implementadas en la arquitectura	94
7.10. Resumen presupuestario del sistema integrador Mirai	94
12.1. Estructura de la colección <code>users</code>	130
12.2. Subdocumento <code>user_tags</code>	131
12.3. Estructura de la colección <code>careers</code>	131
12.4. Subdocumento <code>areas_de_desarrollo_potencial</code>	132
12.5. Subdocumento <code>areas_de_formacion</code>	132
12.6. Subdocumento <code>tags</code> en <code>careers</code>	132
12.7. Estructura de la colección <code>courses</code>	133
12.8. Estructura de la colección <code>tags</code>	133
12.9. Estructura de la colección <code>cards</code>	133
12.10 Estructura de la colección <code>forums</code>	134
12.11 Subdocumento <code>comments</code>	134
12.12 Subdocumento <code>answers</code> dentro de <code>comments</code>	134
12.13 Estructura de la colección <code>interactions</code>	135
12.14 Estructura de la colección <code>saveditems</code>	135
12.15 Estructura de la colección <code>test_items</code>	136
12.16 Subdocumento <code>options</code>	136
12.17 Estructura de la colección <code>test_results</code>	137
12.18 Subdocumento <code>recommendations</code>	137
12.19 Objeto <code>career</code> en <code>recommendations</code>	137

12.20	Objeto <code>enhanced_explanation</code>	137
12.21	Estructura de la colección <code>testimonies</code>	138
12.22	Estructura de la colección <code>scoring_rules</code>	139

El presente trabajo aborda el desarrollo de un sistema integrador de aplicaciones e infraestructura cloud para una multiplataforma inteligente de orientación vocacional dirigida a estudiantes graduandos con aspiración a cursar estudios universitarios en Guatemala. La investigación responde a la necesidad crítica de mejorar los procesos de elección de carrera en la región, donde factores como la desigualdad económica, la falta de información actualizada y la rápida evolución del mercado laboral dificultan la toma de decisiones informadas por parte de los estudiantes.

El proyecto integra tecnologías avanzadas de inteligencia artificial y análisis de datos para implementar un sistema integrador que, basado en las preferencias académicas, fortalezas intelectuales y gustos personales de los usuarios, proporciona de forma segura, escalable y eficiente los datos necesarios para generar sugerencias personalizadas de licenciaturas, temarios de estudio a medida y recomendaciones de especialización profesional. La arquitectura desarrollada prioriza aspectos fundamentales como la seguridad de la información, la velocidad de respuesta, la escalabilidad para atender múltiples usuarios simultáneamente y la accesibilidad desde diversos contextos geográficos y tecnológicos, incluyendo funcionalidades de autenticación y consulta de resultados generados por modelos externos de inteligencia artificial.

La metodología implementada combina enfoques cuantitativos y cualitativos, incluyendo revisión documental sobre orientación vocacional, instrumentación psicométrica, recolección y análisis de datos, así como la identificación de problemas educacionales específicos de la región Guatemala. El desarrollo técnico sigue un cronograma estructurado que abarca desde la planificación y diseño arquitectónico del sistema integrador hasta la implementación, pruebas, optimización y validación de la infraestructura cloud, priorizando altos estándares de rendimiento, disponibilidad y accesibilidad.

Los resultados obtenidos demuestran que la integración de tecnologías de inteligencia artificial en los procesos de orientación vocacional a través de un sistema integrador ofrece ventajas significativas en términos de personalización, accesibilidad y pertinencia de las recomendaciones. El sistema integrador desarrollado constituye una herramienta valiosa para apoyar a los estudiantes en la crucial tarea de elegir una carrera universitaria alineada con sus capacidades e intereses, contribuyendo así a reducir la deserción académica y a mejorar la satisfacción profesional a largo plazo.

Este trabajo representa una contribución significativa al campo de la orientación vocacional en Guatemala, proponiendo un enfoque tecnológico innovador basado en sistemas integradores que responde a las particularidades socioeconómicas y educativas de la región, con potencial para impactar positivamente en las trayectorias académicas y profesionales de los estudiantes.

Palabras clave: orientación vocacional, inteligencia artificial, sistema integrador, infraestructura cloud, sistemas de recomendación, educación superior, Guatemala.

La elección de una carrera universitaria representa un momento crucial en la vida de cualquier estudiante, marcando el inicio de su trayectoria profesional y personal. Sin embargo, este proceso de decisión puede resultar abrumador, especialmente en el contexto latinoamericano de Guatemala, donde el acceso a la educación superior privada es limitado y muchos estudiantes enfrentan incertidumbre sobre sus aptitudes e intereses. La falta de orientación vocacional adecuada a menudo conduce a elecciones desacertadas, resultando en cambios de carrera, abandono de estudios y frustración profesional.

En Guatemala, la educación superior se enfrenta a desafíos significativos. La desigualdad económica limita el acceso a instituciones privadas de calidad, mientras que las universidades públicas a menudo carecen de recursos para brindar una orientación vocacional personalizada y actualizada. Esta situación se agrava por la falta de información sobre las diversas opciones de carrera y los requisitos académicos, lo que dificulta la toma de decisiones informadas.

Además, la globalización y la rápida evolución del mercado laboral exigen profesionales con habilidades especializadas y adaptabilidad. Los estudiantes necesitan comprender las tendencias del mercado, las demandas de las industrias emergentes y las competencias necesarias para sobresalir en sus campos. Sin embargo, la orientación vocacional tradicional a menudo se centra en pruebas estandarizadas y evaluaciones genéricas, sin considerar las particularidades de cada estudiante y las oportunidades laborales en su entorno.

Ante este panorama, el desarrollo de un sistema inteligente de orientación vocacional se presenta como una solución innovadora y necesaria. Este sistema, basado en tecnologías de inteligencia artificial y análisis de datos, puede proporcionar a los estudiantes una orientación personalizada y precisa, considerando sus intereses, aptitudes, valores y contexto socioeconómico. Al ofrecer información detallada sobre las carreras, los planes de estudio y las perspectivas laborales, el sistema puede empoderar a los estudiantes para tomar decisiones informadas y construir un futuro profesional exitoso.

El presente trabajo propone el desarrollo de un sistema integrador de aplicaciones y su infraestructura cloud para abastecer una multiplataforma inteligente de orientación vocacional, generación de temarios de estudio y sugerencias de especializaciones a nivel profesional, para estudiantes graduandos con aspiración a estudiar una licenciatura. Estos serán generados y clasificados en función de sus preferencias académicas, fortalezas intelectuales y gustos personales. El sistema integrador estará diseñado para proporcionar, de forma segura, escalable y eficiente, los datos necesarios para la plataforma inteligente, incluyendo funcionalidades de autenticación y consulta de resultados generados por modelos externos de inteligencia artificial para la recomendación de carreras y rutas de aprendizaje. El servicio estará diseñado para garantizar la seguridad de la información, la velocidad de respuesta, la exactitud de la información sugerida, y la escalabilidad de

peticiones y recursos, necesarias para atender a un gran número de usuarios de manera simultánea, sin comprometer la calidad de la información ni la experiencia del usuario, priorizando altos estándares de rendimiento, disponibilidad y accesibilidad.

2.1. Objetivo general

Diseñar e implementar una infraestructura cloud para un sistema integrador de aplicaciones que consuma modelos externos de inteligencia artificial.

2.2. Objetivos específicos

- Diseñar la arquitectura de la infraestructura cloud del sistema integrador.
- Desarrollar un sistema integrador que garantice seguridad en datos personales mediante cifrado AES-256 en reposo y TLS 1.3 en tránsito.
- Configurar e implementar políticas de seguridad para el sistema integrador.
- Medir y optimizar la velocidad de respuesta del sistema, garantizando que las operaciones CRUD tengan un tiempo de respuesta promedio < 500 ms bajo una carga de 500 usuarios concurrentes.
- Validar la escalabilidad horizontal del sistema mediante pruebas de rendimiento que demuestren soporte para al menos 2,000 usuarios concurrentes, sin degradar el rendimiento más de un 20 % respecto a la carga base.
- Garantizar el flujo seguro de datos entre el sistema integrador y los modelos de Inteligencia Artificial externos.

El presente proyecto aborda la necesidad de implementar soluciones tecnológicas avanzadas en la orientación educativa, utilizando sistemas inteligentes para la evaluación y recomendación, como los descritos por Chauhan et al. (2020). La propuesta de un sistema integrador de aplicaciones para la clasificación y recomendación de carreras universitarias se alinea con la creciente demanda de herramientas personalizadas que apoyen a los estudiantes en sus decisiones formativas. Este sistema integrador debe proporcionar, de forma segura, escalable y eficiente, los datos necesarios para una plataforma inteligente de orientación vocacional, integrando consideraciones técnicas, educativas y éticas para ofrecer una herramienta pertinente para las demandas actuales y futuras, y respetuosa con la privacidad del usuario.

La infraestructura tecnológica subyacente del sistema integrador es fundamental. El diseño de sistemas basados en inteligencia artificial para entornos educativos requiere arquitecturas robustas que permitan procesar datos y ofrecer recomendaciones de manera eficiente y escalable (Chauhan et al., 2020). La implementación de una infraestructura cloud para el sistema integrador de aplicaciones debe priorizar la seguridad de información, velocidad de respuesta, escalabilidad de peticiones y accesibilidad del servicio. Además, la adopción efectiva de estas herramientas en entornos educativos depende de una cuidadosa orquestación y comunicación entre las partes interesadas, asegurando que la analítica del aprendizaje sea comprendida y utilizada adecuadamente (Prieto-Alvarez et al., 2018).

La necesidad de un sistema integrador surge de la complejidad inherente a conectar múltiples puntos de acceso y servicios. En el contexto de la orientación vocacional, es fundamental unificar la comunicación entre los sistemas de autenticación de usuarios, las bases de datos de información académica y los modelos externos de inteligencia artificial que generan las recomendaciones. El sistema integrador actúa como el punto central que orquesta esta interacción, garantizando que los datos fluyan de manera segura y eficiente entre los diferentes componentes del ecosistema educativo. Esta integración no solo optimiza el rendimiento del sistema, sino que también simplifica la experiencia del usuario al proporcionar un acceso unificado a todos los servicios necesarios para la orientación vocacional.

La relevancia de este enfoque se acentúa ante la transformación del mercado laboral. El Foro Económico Mundial (2023) destaca cómo las habilidades requeridas y las profesiones están en constante evolución, haciendo indispensable una orientación que prepare a los estudiantes para el futuro del trabajo. Un sistema integrador que no solo recomiende carreras, sino que también sugiera áreas de especialización y genere itinerarios de estudio adaptados, responde directamente a esta necesidad de desarrollo continuo de competencias. El sistema debe gestionar eficientemente la velocidad de respuesta para la autenticación, obtención, creación, actualización y eliminación de datos, así como la generación de clasificaciones de sugerencias de posibles licenciaturas, temarios de estudio y sugerencias de especialidades a nivel profesional.

El contexto digital actual de los jóvenes en regiones como América Latina y el Caribe (UNICEF, 2020) justifica la implementación de una solución tecnológica accesible y adaptada a sus patrones de interacción con la información. El sistema integrador debe asegurar la accesibilidad del servicio para que pueda ser utilizado por cualquier usuario, sin importar su ubicación geográfica, dispositivo de acceso o limitaciones de velocidad de conexión a internet, permitiendo que el servicio pueda ser utilizado por un gran número de usuarios de manera simultánea.

La confidencialidad, integridad y disponibilidad son primordiales para el sistema integrador. Al manejar datos personales y académicos, el sistema debe adherirse estrictamente a las directrices sobre inteligencia artificial y educación, garantizando la protección de datos y la privacidad, tal como lo recomienda la UNESCO (2021). La implementación de políticas de seguridad para el sistema integrador debe priorizar la autenticación, autorización, cifrado de datos, protección contra ataques y cumplimiento de normativas de seguridad. El sistema integrador debe garantizar el flujo de datos seguro necesarios para el correcto funcionamiento, asegurando la correcta comunicación entre los puntos de acceso para la obtención de resultados de los modelos de inteligencia artificial de orientación vocacional. La construcción de confianza a través de prácticas responsables es esencial para la aceptación y el éxito del sistema integrador.

El presente trabajo de graduación se centra en el diseño, implementación y validación de un sistema integrador de aplicaciones con infraestructura cloud para la plataforma de orientación vocacional "Mirai", dirigida a estudiantes de último año de diversificado en Guatemala que aspiran a cursar estudios universitarios. El sistema proporciona gestión segura de usuarios mediante OAuth 2.0, un catálogo completo de carreras de la Universidad del Valle de Guatemala con información de mercado laboral y requisitos académicos, sistema de recomendación personalizado basado en inteligencia artificial que procesa evaluaciones vocacionales, plataforma de contenido interactivo con tarjetas educativas, foros comunitarios con moderación diferenciada, y analíticas de progreso del estudiante. La población objetivo son estudiantes graduandos de educación media en Guatemala, con énfasis particular en las ingenierías y ciencias aplicadas como áreas de fortaleza de la Universidad del Valle de Guatemala.

La presente investigación adopta un enfoque de ingeniería tecnológica aplicada, orientado al diseño, implementación y validación de una infraestructura para un sistema integrador cloud seguro, escalable y de alto rendimiento, que sirva de soporte a un sistema inteligente de orientación vocacional. El enfoque metodológico sigue los principios del desarrollo ágil y DevOps, priorizando prácticas de diseño de sistemas distribuidos, seguridad informática, disponibilidad de servicios y eficiencia computacional.

5.1. Orientación vocacional en Guatemala

La orientación vocacional es un proceso psicológico y pedagógico que ayuda a los estudiantes a elegir una profesión acorde con sus motivaciones, aptitudes y valores [1]. En Guatemala, al ser un país de América Latina, la orientación vocacional enfrenta desafíos significativos debido a la desigualdad económica, la falta de información actualizada sobre el mercado laboral y la rápida evolución de las demandas profesionales [2]. La falta de recursos y personal especializado son un obstáculo clave, con muchos orientadores atendiendo a cientos de estudiantes, lo que limita la atención personalizada.

La brecha tecnológica agrava estas dificultades. Aunque las tecnologías de la información y la comunicación (TIC) podrían mejorar la orientación vocacional, el acceso limitado en comunidades rurales y de bajos recursos restringe su implementación [2]. Además, los métodos tradicionales, como pruebas estandarizadas, a menudo no consideran las particularidades de cada estudiante ni las tendencias del mercado laboral, lo que resulta en decisiones poco informadas, cambios de carrera y deserción académica. En este contexto, los sistemas tecnológicos avanzados, como el propuesto en esta tesis, ofrecen una solución innovadora. Al utilizar inteligencia artificial (IA) y análisis de datos, estos sistemas pueden proporcionar orientación personalizada, analizando intereses, habilidades y contexto socioeconómico para generar recomendaciones adaptadas.

La orientación vocacional es un proceso esencial para los estudiantes que se preparan para ingresar a la educación superior. Guatemala enfrenta desafíos significativos debido a la desigualdad económica, la falta de información actualizada sobre el mercado laboral y la rápida evolución de las demandas profesionales. Según [2], la falta de orientación vocacional adecuada perpetúa la desigualdad de oportunidades en la región. La tecnología, en particular la inteligencia artificial, puede desempeñar un papel fundamental al proporcionar orientación personalizada y accesible a los estudiantes, ayudándolos a identificar carreras alineadas con sus intereses, habilidades y valores.

5.2. Elección de carrera universitaria

La elección de carrera es una decisión crítica que influye en la trayectoria profesional y la satisfacción personal de un individuo. Factores como intereses personales, habilidades, valores, expectativas familiares, estatus socioeconómico y acceso a información sobre carreras son determinantes [1]. En Guatemala, la falta de orientación adecuada a menudo lleva a elecciones subóptimas, resultando en arrepentimiento profesional.

El arrepentimiento por la elección de carrera se refiere a la insatisfacción o remordimiento que sienten los profesionales por su camino elegido, a menudo debido a una falta de coincidencia con sus intereses o habilidades, expectativas no cumplidas o presiones externas. Según [3], las carreras con mayor número de arrepentidos incluyen Periodismo (87%), Sociología (72%), Arte (72%), Comunicación (64%) y Magisterio (61%), principalmente por bajos salarios y falta de estabilidad laboral. En Perú, el Periodismo también lidera la insatisfacción debido a la alta competencia y la limitada disponibilidad de empleos [4].

Otro estudio indica que más de un tercio de los graduados de 2014 elegiría otra carrera o no estudiaría en la universidad, a menudo por problemas de empleabilidad [5]. En México, entre el 30% y el 40% de los universitarios abandona o cambia de carrera en los primeros semestres debido a la falta de orientación vocacional [6]. Estos datos subrayan la necesidad de sistemas de orientación que alineen las elecciones con las aptitudes y el mercado laboral, reduciendo el riesgo de arrepentimiento.

5.3. La preparación para los estudios universitarios

La preparación para los estudios universitarios implica desarrollar habilidades académicas, comprender las expectativas universitarias y fomentar hábitos de estudio efectivos. Los planes de estudio personalizados son esenciales para optimizar el aprendizaje, permitiendo a los estudiantes enfocarse en áreas específicas y gestionar su tiempo de manera eficiente [7].

Herramientas como el Centro de Habilidades Académicas Robert Gillespie de la Universidad de Toronto ofrecen guías para crear planes de estudio personalizados, destacando la importancia de identificar fechas clave, desglosar asignaciones y programar tiempo de estudio [7]. De manera similar, plataformas como MyMap.ai utilizan IA para generar horarios de estudio personalizados, ayudando a los estudiantes a priorizar materias y mejorar su rendimiento académico [8].

La necesidad de una plataforma de orientación vocacional en Guatemala es evidente debido a que como estudiantes de un país de América Latina tenemos un sistema de orientación vocacional deficiente, en el cual aproximadamente el 60% persisten con la carrera universitaria elegida al final de los estudios de nivel medio, dejando al 40% de los estudiantes universitarios con carreras abandonadas en los primeros semestres. [6]

5.4. Desafíos en Guatemala

Los desafíos en la orientación vocacional en Guatemala son múltiples. La desigualdad económica limita el acceso a instituciones educativas de calidad, mientras que las universidades públicas a menudo carecen de recursos para ofrecer orientación personalizada [2]. Además, la orientación suele introducirse tarde, en el último año de secundaria, y carece de contenido crucial para la toma de decisiones [2]. La desconexión entre los programas de orientación y las necesidades del mercado laboral también es un problema, ya que los orientadores a menudo no tienen información actualizada sobre tendencias laborales [2].

5.5. ¿Qué es Inteligencia Artificial?

La inteligencia artificial (IA) es una rama de la informática que se centra en la creación de sistemas y máquinas capaces de realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento, la percepción y la toma de decisiones [9]. En términos simples, la IA permite a las computadoras imitar comportamientos inteligentes, procesando grandes cantidades de datos para identificar patrones, hacer predicciones o generar contenido de manera autónoma, sin necesidad de programación explícita para cada escenario posible [10].

Para personas no familiarizadas con el tema, se puede pensar en la IA como un asistente inteligente que aprende de ejemplos, similar a cómo un niño aprende a reconocer objetos o resolver problemas. Asimismo puede considerarse como una máquina universal, que lo resuelve todo en cualquier lenguaje soportado por el proveedor de este servicio. Existen diferentes tipos de IA: la IA estrecha, que se enfoca en tareas específicas como el reconocimiento de voz o la recomendación de productos, y la IA general, que aspira a emular la inteligencia humana en un amplio rango de actividades, aunque esta última aún está en desarrollo [11]. En el contexto de este documento, la IA se aplica en sistemas de recomendación vocacional, donde analiza datos personales de los usuarios para generar sugerencias personalizadas de carreras y rutas de aprendizaje, mejorando la accesibilidad y precisión en la orientación educativa.

Esta tecnología ha evolucionado rápidamente, impulsada por avances en el aprendizaje automático (machine learning) y el procesamiento de datos masivos, permitiendo aplicaciones prácticas en campos como la educación, la salud y el entretenimiento. Sin embargo, su implementación requiere consideraciones éticas, como la privacidad de los datos y la mitigación de sesgos, para asegurar un uso responsable [12]. En este caso, se desea profundizar en la aplicación de la Inteligencia Artificial en el campo de la orientación vocacional de Guatemala.

5.6. Sistemas de recomendación vocacional con propósitos educativos basados en Inteligencia Artificial

Los sistemas de recomendación basados en inteligencia artificial (IA) son herramientas poderosas para personalizar la orientación vocacional. Estos sistemas analizan datos de los usuarios, como intereses, habilidades, valores y rendimiento académico, para generar sugerencias de carreras y planes de estudio. Según [13], los sistemas de recomendación utilizan técnicas como el filtrado colaborativo, que se basa en las preferencias de usuarios similares, y el filtrado basado en contenido, que utiliza las características de los elementos (en este caso, carreras) y las preferencias del usuario. El 70 % de los jóvenes que utilizan herramientas de IA reportan mayor satisfacción en sus elecciones de carrera, destacando la efectividad de estas tecnologías [14]. Sin embargo, se recomienda complementar la IA con asesoramiento humano para abordar aspectos emocionales y contextuales, asegurando un enfoque integral.

Un ejemplo relevante es la plataforma peruana *QueEstudiar*, que utiliza IA para analizar más de medio millón de perfiles vocacionales, logrando una precisión del 74 % y una exactitud del 82 % en sus recomendaciones de áreas de estudios [15]. Este sistema demuestra cómo la IA puede proporcionar evaluaciones completas y actualizadas, aunque se recomienda complementarlas con asesoramiento humano para abordar aspectos emocionales y contextuales. Sin embargo, este tipo de plataformas no conforman un sistema de orientación vocacional. Se desea contribuir al desarrollo del sistema de orientación vocacional de Guatemala, integrando la Inteligencia Artificial a la plataforma de orientación vocacional.

5.7. ¿Qué es un sistema integrador de aplicaciones?

Un sistema integrador, también conocido como plataforma de integración de aplicaciones, es un componente tecnológico que actúa como un puente entre múltiples sistemas, servicios y aplicaciones heterogéneas, facilitando la comunicación, el intercambio de datos y la orquestación de procesos de manera unificada y eficiente. En el contexto de la orientación vocacional basada en inteligencia artificial, un sistema integrador permite conectar interfaces de usuario como aplicaciones web o móviles, con bases de datos y modelos externos de inteligencia artificial para procesar información personalizada, como preferencias académicas y fortalezas intelectuales, generando recomendaciones escalables y seguras para cumplir el objetivo de proveer una plataforma de orientación vocacional eficiente. [16].

5.8. ¿Qué es Backend?

El backend, también conocido como: el lado del servidor en el desarrollo de software, se refiere a la parte de una aplicación o sistema que opera en el fondo o detrás de escena, invisible para el usuario final. Este componente es responsable de manejar la lógica operacional, el procesamiento de datos, la interacción con bases de datos, la autenticación de usuarios y otras operaciones esenciales que permiten que la aplicación funcione correctamente. A diferencia del frontend, que es la interfaz visible con la que interactúan los usuarios (como botones, formularios y diseños en una página web), el backend se encarga de tareas como almacenar información, realizar cálculos complejos y comunicar con servicios externos, como lo es consumir servicios de modelos de Inteligencia Artificial para orientación vocacional, asegurando que los datos se procesen y entreguen de manera eficiente y segura. [17] [18]

En el contexto de sistemas integradores como el propuesto en este trabajo, el backend actúa como el núcleo que conecta múltiples componentes, gestionando el flujo de información entre modelos de inteligencia artificial, bases de datos y aplicaciones cliente. Por ejemplo, cuando un estudiante ingresa sus preferencias en una plataforma de orientación vocacional, el backend procesa estos datos, invoca modelos de IA externos y devuelve recomendaciones personalizadas sin que el usuario vea los procesos internos. Esta separación entre frontend y backend permite un desarrollo modular, facilita el mantenimiento y mejora la escalabilidad, haciendo que el sistema sea más robusto y adaptable a cambios futuros. Entender el backend es fundamental para apreciar cómo los sistemas modernos manejan la complejidad técnica mientras proporcionan experiencias simples y accesibles al usuario. El servicio backend es parte fundamental de un sistema integrador, debido a que en este se ubica toda la lógica operacional para poder integrar los diferentes sistemas que necesiten ser interconectados y en este caso, integrar todos los sistemas de la plataforma de orientación vocacional inteligente. [19]

La importancia del backend en este trabajo reside en su rol como motor invisible que habilita la personalización de recomendaciones vocacionales, manejando datos sensibles con seguridad en un entorno cloud. En Guatemala, donde la accesibilidad tecnológica varía, un backend bien diseñado asegura respuestas rápidas y escalables, contribuyendo a la equidad educativa al permitir que estudiantes de regiones remotas accedan a herramientas avanzadas sin infraestructura local costosa.

Contribuye al trabajo al proporcionar una base modular para futuras expansiones, como agregar nuevos modelos de AI, y optimizando el rendimiento para usuarios simultáneos, alineado con discusiones sobre desarrollo backend eficiente en comunidades técnicas.

5.9. Application Programming Interface

Una Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) es un conjunto de reglas y protocolos que permite que diferentes programas de software se comuniquen entre sí, intercambiando datos y

funcionalidades de manera estandarizada. En esencia, actúa como un intermediario que facilita la integración entre sistemas heterogéneos, permitiendo que una aplicación solicite servicios o datos de otra sin necesidad de conocer su implementación interna. Por ejemplo, una API puede exponer endpoints para enviar datos de usuario y recibir respuestas procesadas, lo que es fundamental en entornos distribuidos como la nube. [20] [21]

Esta tecnología es crucial para el trabajo de graduación porque permite la conexión fluida entre los componentes del sistema integrador, como las interfaces de usuario (web o móvil), las bases de datos y los modelos externos de inteligencia artificial. Sin APIs, sería imposible orquestar el flujo de información personalizada, como las preferencias académicas y fortalezas intelectuales de los estudiantes, para generar recomendaciones vocacionales escalables y seguras. En el contexto de la orientación vocacional en Guatemala, donde se manejan datos sensibles de estudiantes, las APIs aseguran una integración eficiente que minimiza la duplicación de esfuerzos y reduce latencias en las respuestas, contribuyendo directamente a la accesibilidad y pertinencia del sistema.

En el desarrollo del proyecto, las APIs permiten el uso de lenguajes con tipado estricto en el backend, lo que ofrece beneficios como tipado estático para reducir errores en la integración de aplicaciones. Esto contribuye al trabajo al asegurar una implementación de un sistema integrador de aplicaciones, robusto y mantenible, para la plataforma de orientación vocacional, alineada con las necesidades de escalabilidad para atender picos de alta demanda durante periodos de inscripción universitaria. [22]

5.10. ¿Qué es Cloud?

Se utiliza el término "Cloud" o "En la Nube" para referirse a la computación en la nube, un paradigma tecnológico que permite el acceso bajo demanda a recursos informáticos compartidos a través de internet, en lugar de depender de infraestructuras locales o hardware físico dedicado. En esencia, la computación en la nube consiste en alquilar servicios de almacenamiento de datos, procesamiento, redes y aplicaciones de proveedores remotos, lo que elimina la necesidad de invertir en servidores propios y permite un modelo de pago por uso [23] [24]. Este enfoque transforma la forma en que las organizaciones gestionan sus recursos de tecnologías de información, ofreciendo flexibilidad y eficiencia al escalar operaciones sin la complejidad de mantener infraestructura física.

La computación en la nube se basa en una red de servidores remotos interconectados que almacenan y procesan datos, permitiendo a los usuarios acceder a ellos desde cualquier dispositivo con conexión a internet. Según definiciones detalladas, implica la entrega de recursos como servidores virtuales, bases de datos, software analítico e inteligencia artificial, todo administrado por el proveedor para garantizar alta disponibilidad y rendimiento [25] [26]. Históricamente, este concepto surgió en la década de 1960 con ideas de computación distribuida, pero se popularizó en los 2000 con avances en virtualización y banda ancha, evolucionando hacia un ecosistema que soporta desde aplicaciones simples hasta sistemas complejos de IA, como los integradores de orientación vocacional propuestos en este trabajo.

Existen tres modelos principales de servicio en la nube: Infraestructura como Servicio (IaaS), que proporciona recursos básicos como servidores y almacenamiento virtualizados; Plataforma como Servicio (PaaS), que ofrece entornos para desarrollar y desplegar aplicaciones sin gestionar la infraestructura subyacente; y Software como Servicio (SaaS), donde las aplicaciones se entregan directamente al usuario final a través de la web, como correos electrónicos o herramientas colaborativas [27] [28]. Además, los modelos de despliegue incluyen nubes públicas (operadas por terceros y compartidas), privadas (dedicadas a una sola organización) e híbridas (combinación de ambas para equilibrar seguridad y escalabilidad) [29] [30].

Las ventajas de la computación en la nube son múltiples: reduce costos al eliminar inversiones iniciales en hardware, mejora la escalabilidad horizontal permitiendo ajustar recursos en tiempo real según la demanda, y fomenta la colaboración global al acceder datos desde cualquier ubicación. Sin embargo, plantea desafíos como la dependencia de la conectividad a internet y preocupaciones de seguridad, que se abordan mediante

políticas robustas en sistemas integradores [23]. En el contexto de un sistema integrador de aplicaciones para orientación vocacional, la nube facilita la conexión de modelos de IA con bases de datos y APIs, asegurando un procesamiento eficiente de datos personalizados.

En cuanto a los proveedores, el mercado de la computación en la nube en 2025 está dominado por unos pocos gigantes que controlan la mayoría de la cuota global. Amazon Web Services (AWS) lidera con aproximadamente el 32 % del mercado, ofreciendo una amplia gama de servicios escalables y flexibles, ideales para aplicaciones de IA y big data [31] [32]. Le sigue Microsoft Azure, con un 22 % de participación, destacando por su integración con herramientas empresariales y soporte para entornos híbridos [33] [34]. Google Cloud Platform (GCP) ocupa el tercer lugar con un 12 %, enfocado en análisis de datos y machine learning, gracias a su expertise en IA [35]. Otros proveedores notables incluyen IBM Cloud, que enfatiza la seguridad y la nube híbrida para entornos empresariales; Oracle Cloud Infrastructure (OCI), optimizado para bases de datos y aplicaciones de alto rendimiento; Alibaba Cloud, dominante en Asia con soluciones de e-commerce; y Huawei Cloud, que gana terreno en mercados emergentes con énfasis en 5G e IA [36] [33]. Estos proveedores compiten en innovación, precios y cumplimiento normativo, permitiendo a proyectos como el presente seleccionar opciones basadas en necesidades específicas de escalabilidad y seguridad.

5.11. API Gateway

Un punto de entrada único que gestiona las solicitudes entrantes, enruta el tráfico hacia servicios backend y maneja aspectos como la autenticación y el rate limiting. En un sistema cloud, el API Gateway actúa como una capa de abstracción que centraliza el control de accesos, monitorea el tráfico y aplica transformaciones a las solicitudes y respuestas, lo que simplifica la gestión de múltiples microservicios o APIs externas. Por instancia, puede validar tokens de autenticación antes de redirigir una consulta a un modelo de AI, previniendo accesos no autorizados y optimizando el rendimiento general del sistema. [37] [38] [39]

La importancia de este componente en el trabajo radica en su rol como guardián de la infraestructura cloud, especialmente para un sistema integrador que maneja datos educativos sensibles en una región como Guatemala, donde la desigualdad tecnológica exige soluciones accesibles y seguras. Sin un API Gateway, el sistema estaría expuesto a vulnerabilidades como sobrecargas o ataques distribuidos, lo que comprometería la confidencialidad de las preferencias vocacionales de los usuarios. Esto es vital para mantener la integridad del proceso de recomendación, donde se integran modelos de inteligencia artificial para generar sugerencias personalizadas de carreras y temarios.

5.12. Escalabilidad horizontal

La capacidad de agregar recursos dinámicamente para manejar un mayor volumen de usuarios, esencial en entornos educativos con picos de demanda, como periodos de inscripción universitaria. A diferencia de la escalabilidad vertical (aumentar potencia en un solo servidor), la horizontal implica añadir más instancias o nodos en paralelo, distribuyendo la carga a través de un clúster. Esto se logra mediante herramientas como auto-escalado en la nube, donde se monitorean métricas como CPU o tráfico para activar nuevos servidores automáticamente, asegurando alta disponibilidad y rendimiento óptimo en sistemas de recomendación basados en AI. [40] [41] [42]

En el contexto del trabajo, esta escalabilidad es fundamental porque la plataforma de orientación vocacional debe soportar variaciones en el uso, como miles de estudiantes accediendo simultáneamente durante temporadas de graduación en Guatemala. Sin ella, el sistema podría colapsar bajo carga, afectando la entrega de recomendaciones personalizadas basadas en IA. Es importante para garantizar accesibilidad geográfica y tecnológica, reduciendo costos al pagar solo por recursos usados y contribuyendo a la democratización de herramientas educativas.

5.13. Consumo de modelos de Inteligencia Artificial

Se refiere a la integración con APIs externas de modelos de machine learning o generative AI, permitiendo que el sistema ingiera datos de entrada (e.g., perfiles de usuarios) y obtenga resultados (e.g., sugerencias de carreras) sin necesidad de hospedar los modelos estática e internamente. Esto implica llamadas a servicios como OpenAI o modelos personalizados vía endpoints, procesando inputs como fortalezas intelectuales para generar outputs educativos adaptados, con énfasis en eficiencia y bajo costo computacional. [43] [44]

Esta aproximación es esencial para el trabajo porque permite leveraging de modelos de IA avanzados sin la carga de entrenamiento interno, enfocándose en la integración segura para orientación vocacional. En Guatemala, donde los recursos computacionales son limitados, consumir AI vía APIs acelera el desarrollo y asegura actualizaciones continuas de los modelos, mejorando la precisión de recomendaciones y reduciendo deserción académica al alinear carreras con intereses personales.

Contribuye al trabajo al optimizar la infraestructura cloud, combinando con API Gateway para autenticar llamadas y escalabilidad horizontal para manejar volúmenes altos de consultas AI, resultando en un sistema eficiente y adaptable. Implementado en backend, asegura manejo seguro de datos en transacciones AI.

5.14. Sistema integrador para la plataforma inteligente de orientación vocacional

Estos términos son fundamentales en entornos de sistemas integradores para garantizar la interoperabilidad, reduciendo la complejidad de conectar componentes distribuidos mientras se prioriza la seguridad y el rendimiento. En aplicaciones educativas, como la orientación vocacional, permiten una integración seamless con modelos de IA externos, asegurando que los resultados se entreguen a clientes diversos (e.g., apps móviles, web) de forma agnóstica al proveedor cloud subyacente. Estas son los conceptos base para entender el funcionamiento del sistema integrador de aplicaciones para la plataforma inteligente de orientación vocacional, independientemente de sus especificaciones técnicas. En esencia, este sistema actúa como el orquestador central que une las piezas del puzzle educativo, permitiendo que datos de usuarios fluyan hacia modelos de IA para generar recomendaciones personalizadas de carreras, temarios y especializaciones, todo dentro de un entorno cloud escalable. [45]

La importancia de este componente en el trabajo de graduación radica en su capacidad para abordar las desigualdades educativas en Guatemala, donde los estudiantes a menudo carecen de acceso a orientación personalizada. Al integrar IA con plataformas educativas, el sistema facilita recomendaciones adaptadas a fortalezas individuales, reduciendo la deserción y mejorando la alineación con el mercado laboral. Esto es crucial para un contexto donde la desigualdad económica y la falta de información actualizada son barreras comunes, haciendo que el sistema sea una herramienta democratizadora para la educación superior. [46]

Contribuye al sistema al asegurar que la plataforma sea accesible y eficiente, optimizando el uso de recursos cloud para atender picos de demanda durante periodos clave como inscripciones. En el proyecto, este sistema integrador fortalece la arquitectura overall, permitiendo el leveraging de plataformas como FutureFit AI para inspirar soluciones locales, y asegurando que el backend en TypeScript maneje integraciones robustas con menos errores.

5.15. Función de la arquitectura en un sistema integrador

En un sistema integrador, la arquitectura cumple una función pivotal al definir la estructura general del sistema, estableciendo cómo los componentes interactúan para facilitar la integración de servicios heterogéneos, el procesamiento de datos y la orquestación de flujos de trabajo. Esta arquitectura actúa como el

blueprint que guía el desarrollo, asegurando que el sistema pueda manejar la comunicación entre interfaces de usuario, bases de datos y modelos externos de inteligencia artificial de manera eficiente y coherente. [47]

Específicamente, en el contexto de un integrador para orientación vocacional, la arquitectura backend permite el consumo seguro de Inteligencia Artificial, procesando entradas como perfiles de usuarios y generando salidas personalizadas, mientras mantiene la integridad de los datos y la escalabilidad para múltiples clientes. Según [22], la arquitectura backend es responsable de manejar el almacenamiento, el procesamiento y la comunicación entre componentes, lo que es esencial para sistemas que integran servicios externos y responden a solicitudes en tiempo real.

Además, la arquitectura asegura atributos de calidad como la disponibilidad, la resiliencia y la seguridad, permitiendo que el sistema se adapte a cargas variables sin comprometer el rendimiento. Por ejemplo, mediante el uso de patrones de diseño como el API Gateway y la orquestación de micro-servicios, se optimiza el flujo de datos desde los modelos de IA hacia los clientes frontend, reduciendo latencias y mejorando la experiencia del usuario. Esta función no solo facilita la integración técnica, sino que también alinea el sistema con requisitos no funcionales, como la conformidad con estándares de privacidad y la capacidad de evolución futura. [48]

La importancia de esta función en el trabajo radica en su capacidad para crear un sistema robusto y adaptable, esencial para manejar la diversidad de contextos educativos en Guatemala. Sin una arquitectura sólida, la integración de IA podría fallar bajo carga, comprometiendo la utilidad de las recomendaciones vocacionales. Esto es vital para superar desafíos como la desigualdad tecnológica, asegurando que la plataforma sea resiliente y evolucione con las necesidades del mercado laboral.

5.16. Arquitectura cloud en un sistema integrador

La arquitectura cloud, o arquitectura en la nube, se refiere al diseño y estructuración de sistemas informáticos que utilizan recursos computacionales remotos proporcionados por proveedores de servicios en la nube, como servidores, almacenamiento, bases de datos y redes, accesibles a través de internet [49]. En el contexto de sistemas backend o integradores, esta arquitectura define cómo se organizan y conectan los componentes del "lado del servidor" (ver Sección sobre Concepto de Backend), permitiendo la integración de aplicaciones, el procesamiento de datos y la orquestación de servicios de manera distribuida y escalable [50]. Por ejemplo, en un sistema integrador para orientación vocacional, la arquitectura cloud facilita el consumo de modelos de IA externos, el almacenamiento seguro de perfiles de usuarios y la entrega de recomendaciones a múltiples clientes simultáneamente, sin necesidad de infraestructura física local.

A diferencia de las arquitecturas tradicionales (on-premises o locales), que dependen de servidores físicos instalados en las instalaciones de una organización, con recursos fijos y costos iniciales altos para hardware y mantenimiento, las arquitecturas cloud ofrecen flexibilidad y eficiencia. En las tradicionales, la escalabilidad requiere compras adicionales de equipo, lo que puede generar subutilización durante periodos de baja demanda y altos tiempos de inactividad por fallos; en contraste, la nube permite el auto-escalado automático, pagando solo por los recursos usados (modelo pay-as-you-go), reduciendo costos operativos en hasta un 50 % en algunos casos y mejorando la disponibilidad global [51]. Además, las arquitecturas cloud incorporan modelos como IaaS (Infraestructura como Servicio), PaaS (Plataforma como Servicio) y SaaS (Software como Servicio), que abstraen la gestión de hardware, permitiendo a los desarrolladores enfocarse en la lógica de negocio en lugar de la infraestructura subyacente.

Profundizando en el tema, las arquitecturas cloud para backend e integradores enfatizan la resiliencia mediante redundancia geográfica (datos replicados en múltiples centros de datos), la seguridad integrada (cifrado automático y controles de acceso) y la integración con herramientas de DevOps para deployments continuos. En comparación con las tradicionales, que a menudo sufren de latencia mínima pero limitaciones en movilidad, las cloud pueden introducir algo de latencia debido a la dependencia de internet, pero compensan con mayor agilidad para actualizaciones y colaboración remota [52]. Un enfoque híbrido combina ambos

mundos, integrando sistemas legacy con servicios cloud para una transición gradual, común en entornos educativos donde se requiere compatibilidad con infraestructuras existentes [53]. Esta evolución hacia la nube es esencial para sistemas como el propuesto, ya que soporta la escalabilidad necesaria para atender a miles de estudiantes en Guatemala, asegurando accesibilidad y rendimiento óptimo.

5.17. Tipos de arquitecturas para sistemas integradores en la nube

El diseño de arquitecturas cloud para sistemas integradores debe enfocarse en la escalabilidad, la independencia del proveedor (cloud-agnostic), la integración con modelos de IA externos y la provisión de resultados a múltiples clientes. A continuación, se describen tres arquitecturas relevantes, adaptadas al contexto de un sistema de orientación vocacional. Estas se basan en principios de diseño que permiten el consumo de APIs de IA (e.g., modelos de recomendación basados en filtrado colaborativo o aprendizaje profundo) y la distribución de resultados personalizados, priorizando la alta disponibilidad y la eficiencia computacional.

5.17.1. Arquitectura basada en microservicios

La arquitectura de microservicios divide el sistema integrador en componentes independientes y modulares, cada uno responsable de una función específica, como la autenticación de usuarios, el procesamiento de datos o la invocación de modelos de IA. Esta aproximación facilita la escalabilidad horizontal, permitiendo que servicios individuales se escalen según la demanda sin afectar al conjunto [54].

En el contexto de orientación vocacional, un microservicio dedicado al consumo de IA podría invocar APIs externas de modelos como aquellos basados en TensorFlow o Hugging Face, procesando entradas como perfiles de estudiantes y generando salidas como sugerencias de carreras. Un API Gateway (e.g., implementado con herramientas como Kong o AWS API Gateway, pero de forma agnóstica) actúa como fachada, enrutando solicitudes desde clientes (web, móvil) hacia los microservicios relevantes, mientras maneja la autenticación OAuth y el cifrado de datos sensibles. Para la escalabilidad, se utiliza auto-scaling basado en métricas como CPU o tráfico de solicitudes, asegurando que picos de usuarios (e.g., miles de estudiantes consultando simultáneamente) no comprometan el rendimiento.

Esta arquitectura es cloud-agnostic al emplear estándares como RESTful APIs y contenedores Docker, permitiendo despliegues en AWS, Azure o Google Cloud sin modificaciones significativas. Según un estudio sobre escalabilidad en aplicaciones de ML, los microservicios reducen el tiempo de deployment en un 40 % y mejoran la resiliencia al aislar fallos [55]. Sin embargo, requiere una robusta orquestación para evitar la complejidad de "microservices sprawl".

5.17.2. Arquitectura serverless

La arquitectura serverless elimina la gestión de servidores subyacentes, delegando la escalabilidad automática al proveedor cloud, lo que la hace ideal para sistemas integradores con cargas variables, como consultas esporádicas en orientación vocacional. En este modelo, funciones como AWS Lambda, Azure Functions o Google Cloud Functions se activan por eventos (e.g., una solicitud HTTP de un usuario), consumiendo modelos de IA externos vía APIs y devolviendo resultados en tiempo real [56].

Para este proyecto, una función serverless podría manejar el flujo completo: recibir datos del usuario (e.g., preferencias académicas), invocar un modelo de IA externo para generar recomendaciones, y distribuir los resultados a clientes multiplataforma mediante un bus de eventos como Apache Kafka o un servicio agnóstico como Pub/Sub. La escalabilidad es inherente, con auto-scaling que ajusta recursos en milisegundos para manejar desde una hasta miles de solicitudes simultáneas, reduciendo costos al cobrar solo por ejecución. La integración con IA se realiza a través de triggers event-driven, por ejemplo, usando webhooks para notificar

resultados asincrónicos.

Esta arquitectura es cloud-agnostic mediante el uso de frameworks como Serverless Framework o OpenFaaS, que abstraen las diferencias entre proveedores. Un análisis reciente indica que serverless reduce los costos operativos en un 50-70 % en aplicaciones de IA con patrones de uso intermitentes, aunque puede introducir latencia en cold starts para funciones infrecuentes [57]. Es particularmente adecuada para entornos educativos donde la demanda varía por temporadas.

5.17.3. Arquitectura contenerizada con Kubernetes

La arquitectura contenerizada utiliza contenedores (e.g., Docker) para empaquetar componentes del sistema integrador, orquestados por Kubernetes (K8s), un sistema open-source para automatizar el deployment, scaling y gestión de aplicaciones distribuidas. Esto asegura portabilidad cloud-agnostic, ya que K8s se ejecuta en cualquier proveedor (AWS EKS, Azure AKS, Google GKE) sin cambios en el código [58].

En el ámbito de la orientación vocacional, contenedores separados podrían albergar servicios como un integrador de IA que consume modelos externos (e.g., vía gRPC o REST), procesando datos de usuarios y almacenando resultados en una base de datos persistente como PostgreSQL en un volumen Kubernetes. Kubernetes maneja la escalabilidad mediante Horizontal Pod Autoscaler (HPA), que ajusta el número de pods basados en métricas como tráfico o uso de recursos, permitiendo manejar un gran volumen de clientes simultáneos. La provisión de resultados se realiza a través de un Ingress Controller, exponiendo endpoints seguros para aplicaciones cliente.

Esta aproximación integra AI de manera eficiente, soportando workloads de ML con operadores como Kubeflow para orquestar pipelines de inferencia. Según un informe sobre multicloud management, Kubernetes reduce el vendor lock-in en un 80 % y mejora la resiliencia en entornos AI al distribuir cargas [59]. Desafíos incluyen la curva de aprendizaje para la configuración inicial, pero ofrece alta disponibilidad mediante replicas y self-healing.

Estas arquitecturas, al ser cloud-agnostic, permiten flexibilidad en el despliegue, enfocándose en la integración de IA para orientación vocacional mientras garantizan escalabilidad y seguridad. La elección depende de factores como el volumen de usuarios y el costo, pero todas priorizan el consumo eficiente de modelos externos y la entrega de resultados personalizados.

5.18. Seguridad de un sistema integrador de aplicaciones

La seguridad en un sistema integrador de aplicaciones basado en la nube es un aspecto fundamental, especialmente cuando se manejan datos sensibles relacionados con la orientación vocacional, como preferencias personales, fortalezas intelectuales y perfiles educativos de los usuarios. En un entorno cloud, donde múltiples servicios y modelos de IA se interconectan a través de APIs, es esencial implementar medidas robustas para proteger contra amenazas externas e internas, garantizando la confidencialidad, integridad y disponibilidad de la información [60].

Esta sección es determinante para el trabajo de graduación, ya que el sistema integrador debe garantizar la confianza de los usuarios al proteger datos sensibles, alineándose con el objetivo general de proporcionar una plataforma segura y escalable. La implementación de estas medidas asegura que el sistema cumpla con las necesidades de los estudiantes guatemaltecos, donde la privacidad es un factor crítico para la adopción de herramientas tecnológicas.

5.18.1. Importancia de la arquitectura segura

Una arquitectura segura en sistemas integradores cloud no solo previene brechas de datos, sino que también asegura el cumplimiento de regulaciones y políticas de seguridad, fomenta la confianza de los usuarios y hace que un sistema integrador sea seguro de utilizar. Según estudios, el diseño de seguridad debe considerar riesgos como configuraciones incorrectas y amenazas internas, que son comunes en entornos híbridos [61]. La importancia radica en proteger los datos en tránsito y en reposo, minimizando el impacto de posibles ataques cibernéticos que podrían comprometer la privacidad de los estudiantes guatemaltecos que utilicen la plataforma inteligente de orientación vocacional.

Amenazas comunes y mitigación

Entre las amenazas comunes en sistemas cloud se encuentran las brechas de datos, configuraciones erróneas, amenazas internas y ataques de denegación de servicio. Para mitigarlas, se recomienda implementar monitoreo continuo, cifrado de datos y controles de acceso estrictos [61]. Por ejemplo, el uso de herramientas de auditoría en la nube ayuda a identificar vulnerabilidades en tiempo real, mientras que la educación de los usuarios reduce riesgos internos.

La identificación y mitigación de estas amenazas son cruciales para cumplir con el objetivo de implementar una infraestructura segura. En el contexto del trabajo, esto asegura que el sistema integrador pueda operar sin interrupciones, proporcionando recomendaciones vocacionales confiables y protegiendo la integridad de los datos procesados por modelos de Inteligencia Artificial y clasificación externos.

Otra amenaza frecuente es el robo de credenciales, que puede mitigarse mediante autenticación multifactor y rotación regular de claves [62].

El manejo del robo de credenciales influye en la robustez del sistema, garantizando que solo usuarios autorizados accedan a los servicios, lo que es esencial para mantener la confianza y cumplir con las leyes de protección de datos en Guatemala.

Impacto de la arquitectura segura en el desempeño del sistema integrador

La arquitectura segura impacta directamente en el desempeño del sistema integrador, no solamente protegiendo el sistema, sino también protegiendo al alto volumen de usuarios que serán atendidos simultáneamente, en otras palabras, que el sistema sea escalable, cumpliendo así un objetivo específico del proyecto. Al mitigar riesgos como configuraciones erróneas, se asegura que la infraestructura cloud pueda escalar sin comprometer la seguridad, facilitando la accesibilidad desde diversos contextos geográficos y tecnológicos en la región, independientemente de la plataforma cloud donde esté alojado el sistema integrador.

5.18.2. Autenticación

La autenticación es la primera línea de defensa en un sistema integrador, asegurando que solo usuarios autorizados accedan a los servicios. Métodos comunes incluyen la autenticación básica, claves API, OAuth 2.0 y tokens JWT, que son ideales para entornos cloud donde se integran múltiples aplicaciones [63].

La autenticación es un pilar para alcanzar el objetivo de seguridad, ya que permite un acceso controlado a los datos sensibles de los estudiantes. En el trabajo de graduación, su implementación asegura que el sistema integrador sea accesible solo a usuarios verificados, protegiendo el acceso a las recomendaciones personalizadas generadas por modelos externos de Inteligencia Artificial.

El sistema OAuth 2.0 para autenticación segura y federada, es una excelente opción para el sistema integrador de aplicaciones de la plataforma inteligente de orientación profesional, ya que este permite a los

usuarios iniciar sesión mediante proveedores externos como Google o Microsoft, reduciendo la exposición de credenciales [64]. El uso de OAuth 2.0 positivamente en la escalabilidad del sistema, facilitando la integración con plataformas educativas existentes y cumpliendo con los objetivos de accesibilidad y eficiencia en la región.

Métodos avanzados de autenticación

Además de los básicos, métodos como la autenticación biométrica o basada en verificación de múltiples pasos pueden integrarse para mayor seguridad, especialmente en accesos a datos sensibles de orientación vocacional [65]. La combinación de estos métodos con políticas de expiración de sesiones minimiza riesgos de suplantación de identidad.

La incorporación de métodos avanzados de autenticación es determinante para el trabajo, ya que fortalece la seguridad en un sistema que manejará datos personales de estudiantes, alineándose con políticas de seguridad locales de protección de datos y mejorando la experiencia del usuario.

5.18.3. Seguridad de la información personal

La protección de datos personales es crítica en un sistema que maneja información sensible de estudiantes. Esto implica no solo cumplir con leyes, sino también implementar medidas técnicas para prevenir fugas [66].

Esta subsección es esencial para el cumplimiento del objetivo general de proporcionar un sistema seguro, ya que protege la privacidad de los usuarios, un factor clave para su adopción en Guatemala, donde las brechas de datos son una preocupación creciente debido a la falta de regulaciones y leyes de seguridad de la información en la mayoría de países.

Leyes de seguridad de la información

En países de América Latina como lo es Guatemala, las leyes de protección de datos varían por país, pero muchas se inspiran en el General Data Protection Regulation (GDPR), el cual es el reglamento general de protección de datos europeo. Por ejemplo, en Argentina, la Ley 25.326 regula el tratamiento de datos personales, complementada por normativas específicas [67]. En Brasil, la LGPD (Ley General de Protección de Datos) exige consentimiento explícito y medidas de seguridad para datos sensibles [68]. Para este sistema, es esencial cumplir con estas regulaciones para operar en múltiples países de la región.

El cumplimiento de estas leyes influye directamente en la legitimidad del sistema integrador, asegurando que el trabajo de graduación sea viable en un contexto multinacional y cumpla con los estándares éticos y legales, un aspecto crítico para su implementación.

Otras leyes relevantes incluyen la Ley Federal de Protección de Datos en México y la Ley 29733 en Perú, que enfatizan la responsabilidad del controlador de datos en entornos cloud [69].

La adherencia a estas normativas adicionales impacta en la escalabilidad del sistema, permitiendo su expansión a otros países latinoamericanos mientras se alinean con los objetivos de accesibilidad y seguridad.

Encriptación

La encriptación protege los datos personales tanto en reposo como en tránsito. Técnicas como AES-256 para datos en reposo y TLS 1.3 para en tránsito son recomendadas en entornos cloud [70]. En el sistema integrador, los datos de usuarios deben encriptarse antes de almacenarse en la nube, utilizando claves gestionadas por servicios como AWS KMS o Google Cloud KMS [71].

La encriptación es fundamental para cumplir con el objetivo de proteger la información personal, asegurando que los datos de los estudiantes permanezcan confidenciales durante todo el proceso de recomendación vocacional. Al encriptar la información, se cumple con políticas de seguridad como GDPR para poder proveer un sistema integrador seguro para la plataforma inteligente de orientación vocacional.

Tipos de encriptación

Se distinguen el cifrado simétrico (rápido para grandes volúmenes de datos) y asimétrico (seguro para intercambio de claves). Para datos personales en IA, el cifrado homomórfico permite procesar datos encriptados sin descifrarlos, preservando la privacidad [72].

El uso de diferentes tipos de encriptación aplica directamente al trabajo al optimizar el procesamiento de datos sensibles por modelos de IA, influenciando la eficiencia y la seguridad del sistema integrador.

5.18.4. Implementación de seguridad de la información personal

La implementación involucra la integración de controles en la arquitectura cloud, como el uso de API gateways con validación de tokens y monitoreo de logs. Se debe realizar auditorías regulares y pruebas de penetración para validar la efectividad [73].

Esta etapa es determinante para validar la infraestructura cloud, asegurando que cumpla con los objetivos de rendimiento y disponibilidad, esenciales para un sistema que servirá a múltiples usuarios simultáneamente.

En fases de desarrollo, adoptar DevSecOps integra la seguridad desde el inicio, automatizando chequeos de vulnerabilidades en el pipeline de CI/CD [74].

La adopción de DevSecOps impacta en la metodología del trabajo, permitiendo un desarrollo iterativo y seguro que respalda los objetivos de optimización y escalabilidad del sistema.

Herramientas y mejores prácticas

Herramientas como Google Cloud Armor o AWS WAF ayudan a mitigar ataques a APIs. Además, implementar el principio de menor privilegio en IAM asegura que los accesos sean limitados [75].

El uso de estas herramientas y prácticas influye en la robustez del sistema, garantizando que el trabajo cumpla con los estándares de seguridad necesarios para operar en un entorno cloud accesible y eficiente.

5.19. Virtual Private Cloud

Una Virtual Private Cloud (VPC), o Nube Privada Virtual, es un entorno de computación en la nube aislado lógicamente que permite a las organizaciones crear una red virtual personalizada dentro de la infraestructura de un proveedor cloud, simulando una red privada tradicional pero con los beneficios de la escalabilidad y flexibilidad de la nube [76] [77]. En esencia, una VPC proporciona control granular sobre el tráfico de red, direcciones IP, subredes y rutas, permitiendo segmentar recursos para mejorar la seguridad y el rendimiento, lo cual es crucial en sistemas integradores que manejan datos sensibles como perfiles de orientación vocacional [78] [79]. Este componente actúa como una barrera virtual, aislando recursos de la internet pública y otros tenants en entornos multi-tenant, asegurando que las comunicaciones internas permanezcan privadas y protegidas contra accesos no autorizados.

El funcionamiento de una VPC se basa en la virtualización de redes, donde se definen componentes como

subredes (subnets), tablas de rutas, gateways de internet y endpoints privados para conectar servicios cloud de manera segura. Por ejemplo, al crear una VPC, se asigna un rango de direcciones IPv4 o IPv6 (como 10.0.0.0/16), dividiéndolo en subredes públicas para recursos expuestos y privadas para bases de datos o servidores internos, controlando el flujo de tráfico mediante reglas de enrutamiento [80] [81]. Históricamente, el concepto de VPC surgió en la década de 2010 con el auge de la nube pública, evolucionando para incorporar características como peering entre VPCs para interconexiones seguras, y soporte para VPNs o Direct Connect para entornos híbridos, facilitando la integración de modelos de IA en plataformas educativas sin exponer datos críticos [82].

Existen varios tipos de VPC según su configuración: VPC predeterminadas, que ofrecen conectividad básica y pública por defecto; VPC personalizadas, donde el usuario define todos los elementos para un control total; y VPC compartidas, que permiten compartir subredes entre cuentas en organizaciones multi-cuenta para optimizar costos y gestión [83] [84]. Las VPC en la nube son particularmente relevantes para sistemas serverless, ya que integran con servicios como API Gateway o Lambda, permitiendo ejecutar funciones en subredes privadas para procesar recomendaciones vocacionales de manera segura, sin exposición directa a internet [85].

Las ventajas de implementar una VPC incluyen el aislamiento de recursos para prevenir brechas laterales, la capacidad de escalar redes horizontalmente sin hardware físico, y la integración con herramientas de seguridad como grupos de seguridad y listas de control de acceso (ACL) para filtrar tráfico [86] [87]. Sin embargo, plantea desafíos como la complejidad en la configuración inicial, que requiere planificación cuidadosa para evitar errores de enrutamiento, y la necesidad de monitoreo constante para detectar anomalías en el tráfico. En el contexto de un sistema integrador de aplicaciones para orientación vocacional, una VPC protege las conexiones entre modelos de IA y bases de datos contra interceptaciones, asegurando que datos personales como resultados de tests vocacionales permanezcan confinados en entornos privados [88].

En cuanto a los proveedores, el mercado de VPC en 2025 está liderado por soluciones integradas en plataformas cloud principales. Amazon Web Services (AWS) ofrece AWS VPC, que soporta hasta 200 subredes por VPC y se integra con servicios como EC2 y RDS, enfatizando en seguridad mediante endpoints privados y flow logs para auditorías [76] [80]. Microsoft Azure proporciona Azure Virtual Network (VNet), enfocada en entornos híbridos con peering global y soporte para aceleración de redes, ideal para aplicaciones enterprise [81]. Google Cloud Platform (GCP) incluye VPC con Shared VPC para organizaciones, destacando en rendimiento global y políticas de firewall avanzadas [82]. Otros proveedores notables son Oracle Cloud Infrastructure (OCI), con VPC optimizadas para baja latencia en bases de datos; IBM Cloud VPC, que prioriza la resiliencia en entornos regulados; y proveedores como Alibaba Cloud o Huawei Cloud, que ganan terreno en mercados emergentes con énfasis en costo-eficiencia y compliance regional [77] [79]. Estos proveedores compiten en características como automatización de despliegue y integración con IA, permitiendo seleccionar opciones basadas en necesidades de escalabilidad y seguridad para proyectos educativos como el presente.

5.19.1. Web Application Firewall

Un Web Application Firewall (WAF), o cortafuegos de aplicaciones web, es una herramienta de seguridad diseñada específicamente para proteger las aplicaciones web contra amenazas cibernéticas al filtrar, monitorear y bloquear el tráfico HTTP/S malicioso entre la aplicación y los usuarios de Internet [89] [90]. A diferencia de los firewalls tradicionales que se centran en el nivel de red, un WAF opera en la capa de aplicación (capa 7 del modelo OSI), analizando las solicitudes y respuestas para detectar patrones de ataques comunes como inyecciones SQL, cross-site scripting (XSS), cross-site request forgery (CSRF) y otros exploits dirigidos a vulnerabilidades web [91] [92]. Este componente es esencial en entornos cloud, donde las aplicaciones expuestas a la web enfrentan un volumen creciente de amenazas automatizadas y sofisticadas, asegurando la integridad, confidencialidad y disponibilidad de los datos procesados.

El funcionamiento de un WAF se basa en reglas predefinidas o políticas de seguridad que inspeccionan el tráfico en tiempo real. Por ejemplo, utiliza firmas para identificar payloads maliciosos, listas blancas o

negras para permitir o denegar accesos, y aprendizaje automático para adaptarse a nuevas amenazas [93] [94]. Cuando una solicitud entra, el WAF la descompone en elementos como cabeceras, cookies, parámetros URL y cuerpo del mensaje, aplicando validaciones para bloquear intentos de explotación. En caso de detección, puede responder con acciones como rechazar la solicitud, registrar el incidente o redirigir al usuario, minimizando el impacto en el rendimiento de la aplicación [95] [96]. Históricamente, los WAF surgieron en la década de 2000 como respuesta al aumento de ataques web, evolucionando de soluciones basadas en reglas estáticas a sistemas inteligentes que incorporan IA para la detección de anomalías, especialmente en contextos de DevSecOps donde la seguridad se integra desde el desarrollo [97].

Existen varios tipos de WAF según su implementación: los basados en red (network-based), que se despliegan como appliances hardware en el perímetro de la red para un alto rendimiento pero con costos elevados; los basados en host (host-based), instalados directamente en el servidor de la aplicación para una personalización profunda, aunque consumen recursos locales; y los basados en nube (cloud-based), ofrecidos como servicio gestionado por proveedores, que destacan por su escalabilidad y facilidad de actualización sin hardware dedicado [89] [90]. Los WAF cloud son particularmente relevantes para sistemas integradores como el propuesto en este trabajo, ya que permiten una protección global distribuida, integrándose con CDN (Content Delivery Networks) para mitigar ataques DDoS y optimizar el tráfico [92].

Las ventajas de implementar un WAF incluyen la reducción de riesgos de brechas de datos, el cumplimiento de normativas como GDPR o PCI-DSS, y la mejora en la experiencia del usuario al prevenir interrupciones causadas por ataques [91] [93]. Sin embargo, plantea desafíos como la posibilidad de falsos positivos que bloqueen tráfico legítimo, requiriendo una configuración fina y monitoreo continuo. En el contexto de un sistema integrador de aplicaciones para orientación vocacional, un WAF protege los endpoints de API y los modelos de IA contra manipulaciones que podrían comprometer datos sensibles de usuarios, como perfiles educativos o resultados de recomendaciones [94].

En cuanto a los proveedores, el mercado de WAF en 2025 está dominado por soluciones cloud que integran IA y automatización. Amazon Web Services (AWS) ofrece AWS WAF, que se integra nativamente con servicios como API Gateway y CloudFront, permitiendo reglas personalizadas y protección contra bots [98] [96]. Microsoft Azure proporciona Azure Web Application Firewall, enfocado en entornos híbridos y con integración a Azure Application Gateway para detección de amenazas en tiempo real [90]. Google Cloud Platform (GCP) incluye Cloud Armor, especializado en mitigación de DDoS y WAF con reglas gestionadas por Google [92]. Otros proveedores destacados son Cloudflare, con su WAF gestionado que cubre más del 20 % del tráfico web global y énfasis en zero-trust; Akamai, líder en protección de APIs y aplicaciones enterprise; Fortinet, con soluciones híbridas para entornos on-premise y cloud; y proveedores emergentes como Imperva o F5, que ofrecen WAF avanzados con analítica predictiva [89] [93] [91] [95]. Estos proveedores compiten en facilidad de integración, costos por uso y cumplimiento regional, facilitando la selección para proyectos cloud como el presente, donde la seguridad es prioritaria para manejar datos educativos sensibles.

5.20. Políticas de seguridad

Las políticas de seguridad en un sistema integrador de aplicaciones cloud representan un conjunto de directrices y medidas diseñadas para proteger los activos digitales, garantizar la confidencialidad, integridad y disponibilidad de los datos, y mitigar riesgos asociados a amenazas cibernéticas. En el contexto de un sistema orientado a la orientación vocacional que maneja datos sensibles como perfiles educativos y preferencias personales, estas políticas se basan en principios como el de menor privilegio, defensa en profundidad y monitoreo continuo, alineándose con estándares como los establecidos por AWS para aplicaciones serverless [99] [100]. El desarrollo de estas políticas implica una evaluación exhaustiva de vulnerabilidades potenciales, desde ataques a APIs hasta fugas de datos, asegurando que el sistema integre mecanismos preventivos, detectivos y correctivos en todas sus capas.

5.20.1. Desarrollo de políticas

El desarrollo de políticas de seguridad comienza con la identificación de amenazas específicas al sistema, como inyecciones SQL, XSS o ataques DDoS, y la definición de controles para mitigarlas. Para un Web Application Firewall (WAF), las políticas incluyen reglas gestionadas que inspeccionan el tráfico HTTP/S, bloqueando patrones maliciosos como inyecciones SQL mediante firmas predefinidas y listas de exclusión, o limitando tasas de solicitudes para prevenir abusos en APIs [101] [102]. Estas reglas se configuran para operar en modo de bloqueo o monitoreo, permitiendo una respuesta adaptativa a amenazas emergentes, y se integran con servicios como API Gateway para proteger endpoints expuestos [103].

5.20.2. JSON Web Tokens

En cuanto a la autenticación, las políticas para JSON Web Tokens (JWT) utilizan validaciones estrictas como verificar la firma del token con claves asimétricas, comprobar la expiración (exp) y el emisor (iss), y evaluar claims personalizados para roles de usuario [104] [105]. Estas políticas aseguran que solo solicitudes autorizadas accedan a recursos, implementando rotación de claves de firma cada 90 días para reducir el riesgo de compromisos prolongados, alineado con recomendaciones de renovación periódica de credenciales [106] [107].

5.20.3. Infraestructura de red

Para la infraestructura de red, las políticas de Virtual Private Cloud (VPC) definen aislamiento lógico mediante subredes privadas y públicas, utilizando direcciones IPv4 privadas (como rangos CIDR 10.0.0.0/16) para conexiones internas que no expongan recursos a internet directamente [108] [109]. Las subredes se distribuyen en múltiples zonas de disponibilidad para alta disponibilidad, con políticas que prohíben rutas públicas innecesarias y fomentan el uso de endpoints privados para servicios como bases de datos.

Los grupos de seguridad actúan como firewalls virtuales a nivel de instancia, con reglas inbound que permiten tráfico entrante solo desde fuentes específicas (por ejemplo, puerto 443 desde API Gateway) y outbound que restringen salidas a destinos autorizados, como servicios AWS internos [110] [111]. Estas reglas específicas protegen conexiones privadas al aplicar el principio de menor privilegio, bloqueando accesos no autorizados y minimizando superficies de ataque, por ejemplo, permitiendo solo tráfico HTTPS desde subredes privadas a bases de datos [112] [113].

5.20.4. Encriptación

Las políticas de encriptación abordan datos en tránsito y en reposo. Para encriptación en tránsito, se utilizan protocolos como TLS 1.3 para comunicaciones seguras entre componentes, asegurando que datos sensibles viajen cifrados y protegidos contra interceptaciones [114] [115]. En reposo, servicios como Amazon DocumentDB emplean encriptación AES-256 gestionada por AWS KMS, con claves rotadas automáticamente cada 90 días para mantener la frescura criptográfica [116] [117]. Para información personal identificable (PII), como nombres o correos electrónicos, se aplican encriptaciones adicionales, como tokenización o encriptación a nivel de campo, combinadas con políticas de acceso restringido para cumplir con requisitos de minimización de datos [118] [119].

Otras políticas incluyen el monitoreo continuo mediante herramientas como AWS CloudTrail para auditar accesos, y alertas automáticas para anomalías; políticas de least privilege en IAM roles que limitan permisos a lo estrictamente necesario para funciones serverless; y estrategias de respaldo y recuperación con encriptación en snapshots [120] [121]. Además, se incorporan políticas de gestión de secretos con AWS Secrets Manager, rotando credenciales cada 90 días, y escaneo de vulnerabilidades regulares para código y dependencias [122]

[123]. Estas políticas se desarrollan iterativamente, basadas en evaluaciones de riesgo, para asegurar una arquitectura resiliente.

5.20.5. Cumplimiento con regulaciones legales

El cumplimiento con regulaciones legales es integral al desarrollo de políticas de seguridad, asegurando que el sistema respete normativas de protección de datos. Aunque Guatemala carece de una ley específica de protección de datos personales, la Ley de Acceso a la Información Pública (Decreto 57-2008) aplica parcialmente a entidades privadas que manejan datos públicos o sensibles, requiriendo transparencia y seguridad en el procesamiento, lo que se relaciona con políticas de encriptación en reposo y tránsito para proteger PII contra accesos no autorizados [124] [125].

En un contexto regional, la Ley General de Protección de Datos de Brasil (LGPD) influye en proyectos transfronterizos, exigiendo consentimiento explícito y minimización de datos, alineado con políticas de JWT y autenticación que verifican identidades antes de procesar información personal [126] [127]. Para usuarios en la Unión Europea, el Reglamento General de Protección de Datos (GDPR) impone requisitos estrictos como el derecho al olvido y evaluaciones de impacto en privacidad, relacionándose con políticas de rotación de claves cada 90 días para limitar riesgos de brechas, y encriptación adicional para PII para cumplir con artículos 32 (seguridad del procesamiento) y 25 (protección de datos por diseño) [128] [129].

Otras leyes relevantes incluyen la HIPAA para datos de salud si se expandiera el sistema, que se vincula a encriptación en tránsito y auditorías de acceso [130], y normativas de ciberseguridad en Guatemala como el Decreto 57-2024, que promueve marcos de seguridad alineados con WAF y grupos de seguridad para mitigar amenazas [131] [132]. Cada política se diseña considerando estas leyes: por ejemplo, reglas de WAF contra ataques web apoyan el cumplimiento de GDPR Artículo 32 al prevenir inyecciones que comprometan datos; VPC y subredes privadas relacionan con LGPD al aislar datos sensibles; y rotación de claves soporta requisitos de auditoría en leyes de protección de datos latinoamericanas [133] [134].

5.21. Proveedores de nube

En el desarrollo de sistemas integradores cloud para aplicaciones de inteligencia artificial, la selección de proveedores de nube y lenguajes de programación es fundamental para garantizar escalabilidad, eficiencia y compatibilidad con arquitecturas modernas. Este análisis preliminar se basa en evaluaciones teóricas de opciones destacadas, considerando factores como cuota de mercado, servicios ofrecidos, integración con IA y rendimiento en entornos serverless. En el contexto de un sistema de orientación vocacional, estas tecnologías permiten procesar datos educativos de manera segura y eficiente, conectando modelos de recomendación con interfaces de usuario intuitivas [135] [136].

Los proveedores de nube ofrecen plataformas integrales para desplegar infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS), facilitando el desarrollo de sistemas que integran IA y datos en tiempo real. En 2025, el mercado global está dominado por tres principales actores, cada uno con fortalezas específicas en innovación, costos y cumplimiento normativo, lo que permite una comparación basada en necesidades de proyectos educativos como el presente [137] [138].

5.21.1. Amazon Web Services

Amazon Web Services (AWS) es el líder del mercado con aproximadamente el 29-32% de participación, ofreciendo más de 200 servicios que incluyen computación serverless (Lambda), almacenamiento (S3) y bases de datos gestionadas (DynamoDB, DocumentDB). Su enfoque en escalabilidad horizontal y herramientas de IA como SageMaker lo hace ideal para sistemas integradores que requieren procesamiento de grandes

volúmenes de datos vocacionales, con ventajas en madurez del ecosistema y optimización de costos mediante modelos de pago por uso [135] [139].

5.21.2. Microsoft Azure

Microsoft Azure ocupa el segundo lugar con un 22% de mercado, destacando por su integración nativa con herramientas empresariales como Microsoft 365 y entornos híbridos, facilitando la migración de sistemas legacy a la nube. Servicios como Azure Functions para serverless y Azure AI para modelos de recomendación vocacional ofrecen robustez en seguridad y cumplimiento con regulaciones como GDPR, siendo preferido en contextos educativos con énfasis en colaboración y análisis de datos [136] [140].

5.21.3. Google Cloud Platform

Google Cloud Platform (GCP) controla alrededor del 12% del mercado, con un enfoque en innovación en IA y machine learning a través de herramientas como Vertex AI y BigQuery, ideales para procesar datos de orientación vocacional con algoritmos predictivos. Su red global de alta velocidad y precios competitivos en almacenamiento lo posicionan como una opción para proyectos que priorizan el análisis de datos y la sostenibilidad, aunque con una curva de aprendizaje más pronunciada en comparación con AWS o Azure [141] [142].

Estos proveedores se comparan en aspectos como precios (AWS y GCP más competitivos en computación, Azure en integración enterprise), servicios de IA (GCP lidera en ML, AWS en variedad) y seguridad (todos cumplen estándares globales, pero Azure destaca en compliance híbrido), permitiendo seleccionar basados en requisitos de escalabilidad y costos para sistemas integradores educativos [137] [138].

5.22. Lenguajes de programación

La elección de lenguajes de programación en entornos cloud influye en el rendimiento, mantenimiento y compatibilidad con servicios serverless, priorizando aquellos con soporte nativo para APIs, concurrencia y procesamiento de datos. En 2025, lenguajes como Go, Java, Python y JavaScript (con Node.js) dominan el desarrollo cloud-native, ofreciendo versatilidad para integrar modelos de IA en plataformas de orientación vocacional [143] [144].

5.22.1. Golang

Go (Golang), desarrollado por Google, se destaca por su eficiencia en concurrencia y bajo consumo de recursos, ideal para microservicios y aplicaciones serverless en cloud. Su sintaxis simple y compilación rápida facilitan el desarrollo de APIs escalables, con ventajas en rendimiento para sistemas que procesan recomendaciones vocacionales en tiempo real, aunque con una curva de aprendizaje en manejo de errores [145] [146].

5.22.2. Java

Java, un lenguaje maduro y orientado a objetos, ofrece robustez en entornos enterprise con soporte para multithreading y frameworks como Spring Boot, compatibles con proveedores cloud para desplegar integradores de IA. Su portabilidad y ecosistema extenso lo hacen adecuado para aplicaciones seguras y escalables, aunque con overhead en memoria comparado con lenguajes más ligeros [147] [148].

5.22.3. Python

Python, con un crecimiento acelerado (7% anual), es preferido por su simplicidad y bibliotecas para IA como TensorFlow y Pandas, facilitando el desarrollo de modelos de recomendación vocacional en cloud. Su versatilidad en scripting y data science lo posiciona como ideal para prototipos rápidos, aunque con limitaciones en rendimiento para aplicaciones de alta concurrencia sin optimizaciones [149] [150].

5.22.4. JavaScript

JavaScript con Node.js permite desarrollo full-stack en cloud, con soporte asíncrono para APIs y entornos serverless como AWS Lambda. Su ecosistema (NPM) y frameworks como Express facilitan integraciones con frontends educativos, ofreciendo rapidez en iteraciones, aunque con desafíos en escalabilidad para cargas intensivas sin herramientas adicionales [151] [145].

Estos lenguajes se evalúan por factores como rendimiento (Go y Java superiores en concurrencia), facilidad de uso (Python y JavaScript) y compatibilidad cloud (todos con soporte nativo), asegurando una selección alineada con necesidades de sistemas integradores para orientación vocacional [144] [148].

5.23. MongoDB

MongoDB es una base de datos NoSQL de código abierto orientada a documentos, diseñada para almacenar y gestionar grandes volúmenes de datos no estructurados o semiestructurados de manera flexible y escalable [152] [153]. A diferencia de las bases de datos relacionales tradicionales que utilizan tablas con esquemas rígidos, MongoDB almacena información en documentos JSON-like (en formato BSON, Binary JSON), permitiendo campos variables y estructuras anidadas sin necesidad de definir un esquema previo, lo que facilita el manejo de datos dinámicos en aplicaciones modernas como sistemas de recomendación vocacional [154] [155]. Este enfoque NoSQL (Not Only SQL) surgió en 2009 como respuesta a las limitaciones de las bases de datos SQL en entornos de big data y alta variabilidad, evolucionando hacia un ecosistema que soporta consultas complejas, índices y agregaciones para procesar información en tiempo real.

El funcionamiento de MongoDB se basa en colecciones de documentos, donde cada documento es una unidad independiente similar a un registro JSON, agrupados en colecciones análogas a tablas pero sin relaciones estrictas. Utiliza un lenguaje de consulta propio (MongoDB Query Language, MQL) que soporta operaciones como filtros, proyecciones y joins, y emplea réplicas y sharding para alta disponibilidad y escalabilidad horizontal, distribuyendo datos a través de múltiples nodos para manejar cargas intensivas [156] [157]. Históricamente, MongoDB ha incorporado motores de almacenamiento como WiredTiger para transacciones ACID, permitiendo su uso en escenarios que requieren consistencia, aunque originalmente enfocado en velocidad y flexibilidad sobre rigidez relacional.

Existen varios tipos de despliegue para MongoDB: versiones comunitarias de código abierto para entornos locales o cloud personalizados; MongoDB Atlas, una versión gestionada en la nube con integración a proveedores como AWS, Azure y GCP para escalado automático; y ediciones enterprise con características avanzadas de seguridad y monitoreo [158]. Estas opciones hacen de MongoDB una elección versátil para sistemas integradores, donde la capacidad de almacenar perfiles educativos, resultados de tests vocacionales y recomendaciones de IA en documentos flexibles optimiza el rendimiento y la adaptabilidad.

Las ventajas de MongoDB incluyen su escalabilidad horizontal mediante sharding, que distribuye datos para manejar petabytes; flexibilidad en esquemas para iteraciones rápidas en desarrollo; y alto rendimiento en lecturas/escrituras para aplicaciones web y móviles [159] [160]. Sin embargo, plantea desafíos como la complejidad en garantizar consistencia en entornos distribuidos y un mayor consumo de almacenamiento debido a la duplicidad de datos en documentos. En el contexto de un sistema integrador de aplicaciones para orientación vocacional, MongoDB facilita el almacenamiento de datos heterogéneos, como preferencias

de usuarios y modelos de IA, asegurando consultas eficientes para recomendaciones personalizadas sin las restricciones de esquemas relacionales [161].

En cuanto a los proveedores, MongoDB se ofrece principalmente a través de MongoDB Inc., con MongoDB Atlas como servicio cloud gestionado que opera en múltiples proveedores, incluyendo AWS donde se integra con servicios como Lambda para arquitecturas serverless. Otras implementaciones incluyen compatibilidad con bases de datos gestionadas como Amazon DocumentDB, que emula la API de MongoDB para facilitar migraciones, y Cosmos DB en Azure o Cloud Firestore en GCP para alternativas NoSQL similares [152] [153].

5.23.1. Amazon DocumentDB

Amazon DocumentDB es un servicio de base de datos de documentos completamente gestionado por AWS, compatible con la API de MongoDB, que permite ejecutar aplicaciones existentes de MongoDB sin modificaciones significativas mientras AWS maneja la administración subyacente [98] [162]. Diseñado para almacenar datos en formato JSON, ofrece escalabilidad automática, durabilidad con replicación multi-AZ (Availability Zones) y opciones serverless para ajustar capacidad bajo demanda, ideal para sistemas que procesan datos vocacionales variables como perfiles y recomendaciones de IA [163] [164]. Lanzado en 2019, DocumentDB responde a la necesidad de bases de datos NoSQL gestionadas en la nube, evolucionando para incluir características como escalado elástico y encriptación nativa.

El funcionamiento de DocumentDB se basa en un clúster distribuido que separa computación y almacenamiento, replicando datos seis veces a través de tres zonas de disponibilidad para tolerancia a fallos, con un volumen de almacenamiento que crece automáticamente hasta 128 TiB [165] [166]. Utiliza la API de MongoDB (versiones 3.6, 4.0 y 5.0) para consultas, permitiendo operaciones como agregaciones y transacciones ACID, mientras AWS gestiona backups, parches y monitoreo.

Las ventajas de Amazon DocumentDB incluyen su integración nativa con el ecosistema AWS (como VPC para seguridad y Lambda para serverless), alto rendimiento con latencias de milisegundos y costos basados en uso real en modo serverless [163]. Sin embargo, como servicio propietario, presenta limitaciones en compatibilidad total con todas las características de MongoDB open-source, requiriendo validaciones para migraciones complejas. En el contexto del proyecto, DocumentDB protege datos sensibles de orientación vocacional mediante encriptación en reposo y tránsito, facilitando la escalabilidad horizontal en un sistema integrador cloud [98].

La presente investigación adopta un enfoque de desarrollo tecnológico aplicado, centrado en el diseño e implementación de un sistema integrador cloud para orientación vocacional. La metodología se estructura en fases iterativas siguiendo el marco de trabajo Scrum, permitiendo entregas incrementales de valor y adaptación continua a los descubrimientos técnicos durante el desarrollo.

6.1. Enfoque metodológico

El desarrollo del sistema integrador se basa en una metodología ágil adaptada de Scrum, con sprints de dos semanas que permiten iteraciones rápidas y validación continua de componentes. Esta aproximación es apropiada para proyectos de innovación tecnológica donde los requisitos pueden evolucionar basados en pruebas iniciales y descubrimientos técnicos.

6.1.1. Justificación del enfoque ágil

La selección de Scrum como marco metodológico se fundamenta en tres factores clave:

1. **Complejidad técnica variable:** el proyecto involucra integración de múltiples tecnologías (serverless, microservicios, IA externa) cuya complejidad de implementación no puede estimarse completamente al inicio.
2. **Descubrimiento iterativo:** las decisiones arquitectónicas (como la elección entre DynamoDB, RDS o DocumentDB) requieren validación empírica mediante prototipos funcionales.
3. **Retroalimentación temprana:** los sprints cortos permiten detectar problemas de rendimiento, seguridad o escalabilidad antes de invertir esfuerzo en funcionalidades dependientes.

6.1.2. Estructura de sprints

Cada sprint de dos semanas sigue la estructura estándar de Scrum adaptada para desarrollo individual:

- **Planning** (día 1): definición de objetivos del sprint y selección de tareas del backlog priorizado.
- **Daily review** (días 2-13): revisión individual diaria de progreso, identificación de impedimentos y ajustes menores al plan.
- **Sprint review** (día 14): validación de entregables contra objetivos específicos del proyecto.
- **Retrospective** (día 14): identificación de mejoras para el siguiente sprint.

6.2. Selección y justificación de tecnologías

Esta sección documenta las decisiones tecnológicas fundamentales que guían la implementación del sistema, basadas en el análisis comparativo de la Fase 1.

Metodología de análisis comparativo

Para cada tecnología evaluada se aplicará una decisión con los siguientes criterios ponderados:

Cuadro 6.1. Matriz de criterios para evaluación de tecnologías

Criterio	Peso	Justificación
Escalabilidad	25 %	Capacidad de manejar crecimiento de usuarios
Costo operativo	20 %	Viabilidad económica para proyecto educativo
Velocidad de desarrollo	20 %	Time-to-market para validación de concepto
Seguridad nativa	15 %	Protección de datos sensibles de estudiantes
Compatibilidad con IA	10 %	Integración con modelos externos
Curva de aprendizaje	10 %	Factibilidad de implementación individual

6.2.1. Selección de proveedor cloud

Justificación de Amazon Web Services (AWS)

Se selecciona AWS como proveedor cloud por las siguientes razones ponderadas:

1. **Madurez del ecosistema** (25 %): AWS ofrece más de 200 servicios fully-featured con 16 años de experiencia en el mercado, garantizando estabilidad y soporte a largo plazo para el proyecto.
2. **Escalabilidad global** (25 %): con 38 regiones geográficas incluyendo presencia en Sudamérica (São Paulo), AWS proporciona baja latencia para usuarios en América Latina, el público objetivo del sistema.
3. **Servicios serverless maduros** (20 %): Lambda es el servicio serverless más maduro del mercado, con capacidad de escalar desde 0 hasta miles de invocaciones por segundo automáticamente, alineado con el objetivo específico 6 de escalabilidad horizontal.
4. **Integración de seguridad** (15 %): herramientas nativas como IAM, KMS, WAF y Security Groups permiten implementar las políticas de seguridad requeridas por el objetivo específico 4 sin dependencias de terceros.

5. **Compatibilidad con arquitectura híbrida** (10 %): soporte robusto para VPC, VPC Link y API Gateway facilita la implementación de arquitectura serverless-microservicios diseñada.
6. **Modelo de costos** (5 %): capa gratuita generosa (1M invocaciones Lambda/mes) y pricing por uso permiten mantener costos operativos bajos durante desarrollo y primeras etapas de adopción.

Alternativas descartadas

Microsoft Azure Descartado principalmente por:

- Menor número de regiones en América Latina (sin presencia directa en Centroamérica)
- Integración más compleja para arquitectura serverless pura
- Curva de aprendizaje más pronunciada para quien no usa ecosistema Microsoft

Google Cloud Platform (GCP) Descartado por:

- Ecosistema menos maduro en servicios serverless (Cloud Functions vs Lambda)
- Menor cantidad de servicios especializados para integración de aplicaciones
- Documentación menos exhaustiva para casos de uso de orientación vocacional

6.2.2. Selección de arquitectura

Justificación de arquitectura híbrida serverless-microservicios

Se adopta una arquitectura híbrida que combina funciones Lambda serverless con un microservicio dedicado en EC2, en lugar de arquitecturas puras (100 % serverless, 100 % microservicios o Kubernetes), por las siguientes razones:

Ventajas sobre arquitectura serverless pura

- **Procesamiento de IA especializado:** el microservicio dedicado permite mantener modelos de IA en memoria caliente, evitando penalización de cold starts de 3-5 segundos en Lambda para operaciones de machine learning.
- **Control de timeouts:** Lambda tiene límite de 15 minutos por invocación, insuficiente para procesamiento de modelos complejos. EC2 permite procesamiento sin límites arbitrarios.
- **Gestión de dependencias pesadas:** bibliotecas de machine learning (TensorFlow, PyTorch) exceden fácilmente los límites de tamaño de paquete Lambda (250MB). EC2 no tiene estas restricciones.

Ventajas sobre arquitectura de microservicios pura

- **Reducción de costos:** 42 microservicios dedicados costarían ~USD 1,260/mes en instancias t3.micro. Lambda mantiene estos costos en USD 0 durante desarrollo gracias a capa gratuita.

- **Escalabilidad automática:** Lambda escala de 0 a 1000+ instancias en segundos sin configuración. Microservicios requerirían configuración manual de auto-scaling groups.
- **Mantenimiento reducido:** AWS gestiona patching, actualizaciones de OS y monitoreo de Lambda. Microservicios requieren gestión continua de instancias.

Ventajas sobre Kubernetes

- **Complejidad operativa:** Kubernetes requiere expertise en orquestación de contenedores, configuración de clusters, networking complejo. No justificable para proyecto individual.
- **Overhead de recursos:** Cluster Kubernetes mínimo requiere 3 nodos maestros + nodos worker, resultando en costos operativos de USD 500+/mes.
- **Time-to-market:** configuración de Kubernetes toma 4-6 semanas vs. 2 semanas para arquitectura híbrida propuesta.

Distribución de responsabilidades

La arquitectura híbrida distribuye funcionalidades de la siguiente manera:

Cuadro 6.2. Distribución de responsabilidades en arquitectura híbrida

Componente	Tecnología	Responsabilidades
Autenticación	Lambda	Validación JWT, gestión de usuarios
CRUD básico	Lambda	Operaciones en base de datos < 1s
Lógica de negocio	Lambda	Algoritmos ligeros (feed, scoring)
Procesamiento IA	Microservicio EC2	Modelos ML, inferencia compleja

6.2.3. Selección de lenguaje de programación

Justificación de Node.js (JavaScript)

Se selecciona Node.js con JavaScript para el backend por:

1. **Compatibilidad nativa con JSON (30 %):** JSON es el formato estándar para comunicación con modelos de IA externos y respuestas de APIs. JavaScript maneja JSON sin parsing adicional, reduciendo latencia y simplificando código.
2. **Ecosistema serverless maduro (25 %):** Node.js es el runtime con mejor soporte en Lambda, con cold starts 60 % más rápidos que Python y 80 % más rápidos que Java según benchmarks de AWS.
3. **Asincronía nativa (20 %):** modelo event-driven y non-blocking I/O de Node.js es ideal para operaciones concurrentes: consultas a base de datos, llamadas a APIs externas, procesamiento de streams.
4. **Velocidad de desarrollo (15 %):** sintaxis concisa, bibliotecas maduras (Express, Mongoose), y tipado estático opcional con TypeScript permiten iteraciones rápidas en sprints de 2 semanas.
5. **Comunidad y recursos (10 %):** ecosistema npm con 2M+ paquetes, documentación exhaustiva, y comunidad activa facilitan resolver problemas comunes sin reinventar soluciones.

Uso de TypeScript para Robustez en microservicio de recomendaciones

Aunque JavaScript es el lenguaje base, se utilizará TypeScript para:

- **Tipado estático:** detección de errores en tiempo de compilación, reduciendo bugs en producción
- **IntelliSense:** autocompletado inteligente acelera desarrollo y reduce errores de sintaxis
- **Interfaces y tipos:** documentación autoexplicativa de estructuras de datos
- **Refactoring seguro:** cambios en estructuras detectan automáticamente inconsistencias

Python para microservicio de IA

El microservicio de recomendaciones utilizará Python por:

- Ecosistema superior para machine learning (TensorFlow, scikit-learn, pandas)
- Compatibilidad directa con APIs de modelos de IA (OpenAI, Anthropic)
- Código más legible para algoritmos matemáticos complejos

Alternativas descartadas

Go (Golang) Ventajas reconocidas: performance superior, excelente para concurrencia

Razones de descarte:

- Verbosidad: requiere 2-3x más líneas de código que Node.js para misma funcionalidad
- Ecosistema más pequeño: menos bibliotecas para integración con servicios AWS
- Curva de aprendizaje: sintaxis menos familiar retrasaría desarrollo inicial

Java Ventajas reconocidas: madurez, robustez en sistemas enterprise

Razones de descarte:

- Boilerplate excesivo: configuración Spring Boot añade complejidad innecesaria
- Cold starts lentos: JVM en Lambda toma 3-5 segundos en inicializar
- Overhead de memoria: consumo típico de 512MB vs. 128MB de Node.js

6.2.4. Selección de Base de Datos

Justificación de Amazon DocumentDB

Se selecciona DocumentDB (compatible con MongoDB) sobre alternativas como DynamoDB o RDS por:

1. **Flexibilidad de esquema (30 %):** orientación vocacional requiere almacenar estructuras heterogéneas:

- Perfiles de usuario con campos variables según nivel de completitud
 - Planes de estudio con estructura jerárquica (años → semestres → cursos)
 - Resultados de evaluaciones con scores en múltiples dimensiones
 - Recomendaciones con explicaciones textuales y metadatos variables
2. **Consultas complejas** (25 %): DocumentDB soporta:
- Agregaciones complejas para generar analíticas de estudiantes
 - Joins mediante `$lookup` para poblar datos relacionados
 - Consultas con arrays embebidos (comentarios, respuestas, tags)
 - Índices compuestos para optimizar queries frecuentes
3. **Compatibilidad con MongoDB** (20 %): permite usar Mongoose, un ODM maduro con:
- Validación de esquemas en capa de aplicación
 - Middleware para hooks (pre-save, post-find)
 - Población automática de referencias
 - Virtuals para campos calculados
4. **Escalabilidad horizontal** (15 %): DocumentDB escala hasta 64 TiB con 15 réplicas de lectura, suficiente para millones de usuarios en América Latina.
5. **Seguridad integrada** (10 %): encriptación en reposo con AWS KMS, conexiones SSL obligatorias, integración con VPC para aislamiento de red.

Comparación con alternativas

Amazon DynamoDB Ventajas reconocidas: performance predecible, escalabilidad ilimitada

Razones de descarte:

- **Limitaciones en consultas:** no soporta joins ni agregaciones complejas. Consultar "foros de una carrera con comentarios del usuario Xrequeriría múltiples queries y procesamiento en aplicación.
- **Modelado complejo:** requiere desnormalización extrema y duplicación de datos. Plan de estudios jerárquico sería difícil de mantener consistente.
- **Costos impredecibles:** modelo de pricing por read/write units puede escalar exponencialmente con queries ineficientes. DocumentDB tiene costo fijo predecible.

Amazon RDS (PostgreSQL/MySQL) Ventajas reconocidas: ACID completo, consultas relacionales robustas

Razones de descarte:

- **Rigidez de esquema:** cambios en estructura de datos requieren migraciones complejas. En fase de desarrollo, los esquemas evolucionan frecuentemente.
- **Impedance mismatch:** mapeo objeto-relacional añade complejidad. Estructuras anidadas (comentarios con respuestas) requieren múltiples tablas y joins costosos.
- **Escalabilidad vertical:** escalar RDS requiere cambiar tamaño de instancia con downtime. DocumentDB escala horizontalmente sin interrupciones.

Estrategia de modelado en DocumentDB

El modelo de datos seguirá estos principios:

- **Embebido para relaciones 1:pocas**: comentarios embebidos en foros, respuestas embebidas en comentarios
- **Referenciado para relaciones N:M**: carreras y Tags, Usuarios y Carreras guardadas
- **Desnormalización estratégica**: copiar datos frecuentemente leídos para reducir joins
- **Índices por patrón de acceso**: crear índices basados en queries reales, no en intuición

6.2.5. Selección de sistema de autenticación

Justificación de Clerk

Se selecciona Clerk como proveedor de autenticación en lugar de AWS Cognito por:

1. **Velocidad de integración** (35 %): SDK de Clerk para Node.js permite implementar autenticación completa en 2-3 horas vs. 2-3 días con Cognito. Crítico para sprints de 2 semanas.
2. **Experiencia de usuario superior** (30 %): componentes UI pre-construidos para login, registro, recuperación de contraseña. Cognito requiere desarrollar toda la interfaz desde cero.
3. **Webhooks nativos** (20 %): Clerk envía eventos (`user.created`, `user.updated`, `user.deleted`) automáticamente para sincronización con base de datos. Cognito requiere implementar triggers Lambda personalizados.
4. **Developer experience** (10 %): Dashboard intuitivo para configuración vs. consola compleja de AWS. Documentación clara con ejemplos funcionales.
5. **Seguridad sin configuración** (5 %): MFA, detección de dispositivos, monitoreo de sesiones activos por defecto. Cognito requiere configuración manual de cada característica.

Costo-Beneficio

Aunque Clerk tiene costos más altos que Cognito en escala (USD 25/mes vs. gratis hasta 50K usuarios), el plan gratuito de Clerk (10K Monthly Active Users) es suficiente para:

- Toda la fase de desarrollo y pruebas
- Lanzamiento beta con primeros 10,000 usuarios
- Validación de product-market fit antes de escalar

El ahorro en tiempo de desarrollo (40+ horas) justifica el costo futuro.

Integración con AWS

Clerk se integra con AWS mediante:

- **Lambda Authorizer:** valida tokens JWT de Clerk usando clave pública RSA
- **Webhook Authorizer:** sincroniza eventos de Clerk con base de datos
- **Sin vendor lock-in:** Clerk emite JWTs estándar que podrían ser validados por otros sistemas si se migra en el futuro

6.3. Fases del desarrollo

El proyecto se estructura en seis fases secuenciales con superposición controlada, donde cada fase genera entregables concretos que alimentan las siguientes. Esta estructura permite avanzar sistemáticamente mientras se mantiene la flexibilidad ágil dentro de cada fase.

6.3.1. Fase 1: investigación y análisis comparativo

Objetivos de la fase

Esta fase busca establecer las bases teóricas y técnicas del proyecto mediante:

1. Revisión exhaustiva de literatura sobre orientación vocacional en América Latina
2. Análisis comparativo de arquitecturas cloud (microservicios, serverless, Kubernetes)
3. Evaluación de proveedores cloud (AWS, Azure, GCP)
4. Comparación de lenguajes de programación para backend (Go, Java, Python, Node.js)
5. Análisis de bases de datos NoSQL compatibles con arquitecturas serverless

Entregables de la fase

- Documento de análisis comparativo de arquitecturas con recomendación fundamentada
- Matriz de decisión para selección de proveedor cloud
- Justificación técnica de lenguaje de programación seleccionado
- Especificación preliminar de base de datos con modelo de datos conceptual

6.3.2. Fase 2: diseño arquitectónico

Objetivos de la fase

Traducir las decisiones tecnológicas de la Fase 1 en una arquitectura concreta y documentada que servirá como blueprint para la implementación.

Actividades de diseño

Diseño de arquitectura cloud Se diseñará una arquitectura híbrida serverless-microservicios considerando:

- **Capa de presentación:** API Gateway como punto de entrada único
- **Capa de aplicación:** funciones Lambda para lógica de negocio ligera
- **Capa de integración:** microservicio dedicado para consumo de modelos de IA
- **Capa de datos:** Base de datos NoSQL en VPC privada
- **Capa de seguridad:** WAF, autenticación centralizada, encriptación multicapa

Modelado de datos El modelo de datos se diseñará siguiendo principios de bases de datos orientadas a documentos:

1. Identificación de entidades principales (usuarios, carreras, evaluaciones, recomendaciones)
2. Definición de relaciones (embebidas vs. referenciadas) según patrones de acceso
3. Diseño de índices para optimizar consultas frecuentes
4. Estrategias de desnormalización para reducir latencia en lecturas críticas

Especificación de APIs Se especificarán los endpoints HTTP siguiendo el estándar OpenAPI 3.0, incluyendo:

- Estructura de solicitudes y respuestas (schemas JSON)
- Códigos de estado HTTP para cada operación
- Mecanismos de autenticación (JWT via OAuth 2.0)
- Rate limiting y políticas de throttling

Diseño de seguridad

El diseño de seguridad seguirá el principio de "defensa en profundidad" con múltiples capas:

1. **Capa de red:** VPC con subredes privadas, Security Groups restrictivos
2. **Capa de aplicación:** validación de entrada, sanitización de datos, manejo seguro de errores
3. **Capa de datos:** encriptación AES-256 en reposo, TLS 1.3 en tránsito, hashing SHA-256 para PII
4. **Capa de identidad:** autenticación OAuth 2.0, autorización basada en roles, rotación de credenciales

Entregables de la fase

- Diagrama de arquitectura cloud con componentes y flujos de datos
- Esquema de base de datos con colecciones, campos y relaciones
- Especificación OpenAPI de endpoints REST
- Documento de políticas de seguridad por capa
- Estimación de costos operativos mensuales

6.3.3. Fase 3: configuración de infraestructura base

Objetivos de la fase

Establecer la infraestructura cloud fundamental que soportará todos los componentes del sistema, priorizando seguridad y aislamiento de recursos.

Actividades de configuración

Configuración de networking

1. Creación de Virtual Private Cloud (VPC) con CIDR block apropiado
2. Diseño de subredes públicas (para NAT/Internet Gateway) y privadas (para recursos backend)
3. Configuración de tablas de enrutamiento para control de tráfico
4. Establecimiento de Internet Gateway para acceso controlado a internet

Implementación de capas de seguridad

1. Despliegue de Web Application Firewall (WAF) con reglas contra:
 - SQL Injection
 - Cross-Site Scripting (XSS)
 - Ataques DDoS mediante rate limiting por IP
 - Bot protection para prevenir scraping automatizado
2. Configuración de Amazon CloudFront como CDN con:
 - Integración con WAF para protección centralizada
 - Certificados SSL/TLS gestionados automáticamente
 - Headers de seguridad HTTP (HSTS, CSP, X-Frame-Options)
 - Compresión automática para reducir ancho de banda
3. Configuración de Security Groups base para cada tipo de recurso:
 - Lambda: salida a DocumentDB (puerto 27017) e internet (HTTPS)
 - DocumentDB: entrada solo desde Security Group de Lambda
 - Microservicio EC2: entrada solo desde VPC Link

Configuración de DNS y certificados

1. Registro de dominio personalizado para APIs
2. Configuración de Route 53 para resolución DNS
3. Provisión de certificados SSL/TLS mediante AWS Certificate Manager
4. Validación de dominio y renovación automática de certificados

Validación de infraestructura

Se validará la infraestructura base mediante:

- Pruebas de conectividad entre componentes
- Verificación de aislamiento de red (intentos de acceso no autorizado)
- Validación de reglas de WAF contra payloads de ataque simulados
- Medición de latencia inicial de red (baseline)

Entregables de la fase

- VPC configurada con subredes y enrutamiento documentado
- WAF activo con reglas documentadas y logs habilitados
- CloudFront CDN desplegado y apuntando a dominio personalizado
- Security Groups documentados con justificación de cada regla
- Reporte de pruebas de seguridad inicial

6.3.4. Fase 4: implementación de componentes Core

Objetivos de la Fase

Desarrollar e implementar los componentes fundamentales del sistema siguiendo la arquitectura diseñada, con énfasis en modularidad y pruebas incrementales.

Desarrollo de Base de Datos

Despliegue de DocumentDB

1. Lanzamiento de instancia EC2 en subred privada de VPC
2. Instalación de MongoDB/DocumentDB compatible
3. Configuración de encriptación en reposo para todo el volumen
4. Establecimiento de conexiones SSL obligatorias
5. Configuración de backups automáticos diarios

Implementación de esquemas Utilizando Mongoose como ODM, se implementarán esquemas con:

- Validación de tipos estricta
- Valores por defecto para campos opcionales
- Enumeraciones para campos con valores limitados
- Índices simples y compuestos para optimizar consultas

Sistema de encriptación selectiva Se implementará un sistema de doble encriptación:

1. **Capa base AWS:** TLS 1.2+ en tránsito, AES-256 en reposo (automático)
2. **Capa adicional para PII:**
 - TLS 1.3 en tránsito mediante biblioteca `crypto.js`
 - Hashing SHA-256 irreversible en reposo para campos críticos
 - Funciones de encriptación/desencriptación en `repose.crypto.js` y `traffic.crypto.js`

Desarrollo de sistema de autenticación

Integración con Clerk Se integrará Clerk como proveedor de autenticación mediante:

1. Configuración de aplicación en dashboard de Clerk
2. Implementación de Lambda Authorizer para validación de JWT
3. Verificación de firma RSA-256 con clave pública de Clerk
4. Extracción de `user_id` del payload del token
5. Generación de política IAM dinámica (Allow/Deny)

Desarrollo de Webhook Authorizer Para sincronización de eventos de Clerk:

1. Implementación de verificación de firma HMAC
2. Validación de timestamp para prevenir replay attacks
3. Procesamiento de eventos: `user.created`, `user.updated`, `user.deleted`
4. Sincronización automática con colección `users`

Desarrollo de API Gateway

Configuración de HTTP API Gateway Se utilizará HTTP API en lugar de REST API por:

- Mejor rendimiento (latencia 40 % menor)
- Menor costo por millón de solicitudes
- Soporte nativo para HTTP/2 y HTTP/3
- Integración simplificada con Lambda

Configuración de integraciones

1. Mapeo de rutas a funciones Lambda correspondientes
2. Configuración de Lambda Authorizer como default para proteger todos los endpoints
3. Configuración de VPC Link para comunicación con microservicio EC2
4. Configuración de CORS para permitir acceso desde frontend
5. Implementación de rate limiting (1000 req/5min por IP)

Entregables de la fase

- Base de datos DocumentDB operativa con esquemas implementados
- Sistema de encriptación selectiva validado con pruebas
- Lambda Authorizer funcional con validación JWT
- Webhook Authorizer sincronizando eventos de Clerk
- HTTP API Gateway con rutas configuradas
- Documentación de endpoints con ejemplos de uso

6.3.5. Fase 5: desarrollo de módulos funcionales

Objetivos de la fase

Implementar los módulos funcionales del sistema siguiendo una secuencia lógica de dependencias, donde módulos base (autenticación, usuarios) se desarrollan antes que módulos dependientes (foros, recomendaciones).

Estrategia de desarrollo modular

Cada módulo se desarrollará siguiendo el ciclo:

1. **Diseño de esquema:** definición de estructura de datos en Mongoose
2. **Desarrollo de funciones Lambda:** implementación de lógica de negocio
3. **Configuración de triggers:** mapeo de endpoints a funciones en API Gateway
4. **Pruebas unitarias:** validación de lógica con casos de prueba

Secuencia de desarrollo de módulos

Sprint 1-2: módulo de autenticación (Auth) Funcionalidades:

- POST /auth/register: registro de usuarios con encriptación de PII
- POST /auth/update: actualización de perfil con validación de permisos
- POST /auth/delete: eliminación de cuenta con limpieza cascada

Sprint 3-4: módulo de Usuarios (Users) Funcionalidades:

- GET /users/me: obtención de perfil con descriptación de PII
- GET /users: listado de usuarios (solo admin)
- PATCH /users/{id}: edición de rol (solo admin)
- POST /users/saved: guardar carreras/tarjetas
- DELETE /users/saved/{id}: desguardar items
- GET /users/saved: obtener items guardados

Sprint 5-6: módulo de carreras (Careers) Funcionalidades:

- GET /careers: listado de carreras con filtros
- GET /careers/{id}: detalle de carrera con población de cursos
- PATCH /careers/{id}: edición de insights (solo admin)

Sprint 7-8: módulo de exploración (Explore) Funcionalidades:

- POST /explore/cards: creación de tarjetas de contenido
- GET /explore/cards/{id}: obtención de tarjeta
- PUT /explore/cards/{id}: edición (con validación de permisos por tipo)
- DELETE /explore/cards/{id}: eliminación (con validación de permisos)
- POST /explore/interactions: registro de interacciones (view, tap, save, share)
- PATCH /explore/cards/{cardId}/likes: manejo de likes/unlikes

Sprint 9-10: sistema de recomendaciones Implementación de algoritmo de recomendación personalizado:

1. Análisis de interacciones históricas del usuario
2. Extracción de tags más interactuados
3. Cálculo de scoring por tarjeta:
 - Coincidencia de tags: +10 puntos por tag
 - Prioridad de tarjeta: multiplicador
 - Novedad: +20 puntos si < 7 días
4. Ordenamiento por score descendente
5. Paginación para optimizar transferencia de datos

Funcionalidad:

- GET /explore/feed: Feed personalizado con algoritmo

Sprint 11-12: microservicio de recomendaciones IA Desarrollo de microservicio en EC2 para consumo de modelos externos:

1. Configuración de instancia EC2 en subred privada
2. Implementación de servicio HTTP con HonoJS
3. Integración con APIs de modelos de IA (ej: OpenAI, Anthropic)
4. Configuración de VPC Link para acceso seguro desde API Gateway
5. Endpoint: ANY /recommendations/{proxy+}

Sprint 13-14: módulo de foros (Forums) Funcionalidades:

- POST /forums: creación de foro asociado a carrera
- GET /forums/{id}: obtención con población jerárquica (creador, comentarios, respuestas)
- GET /forums: listado con filtros
- PUT /forums/{id}: edición (solo creador o admin)
- DELETE /forums/{id}: eliminación (solo creador o admin)
- POST /forums/{forumId}/comments: agregar comentario
- PUT /forums/{forumId}/comments/{commentId}: editar comentario
- DELETE /forums/{forumId}/comments/{commentId}: eliminar comentario
- POST /forums/{forumId}/comments/{commentId}/answers: agregar respuesta
- PUT /forums/{forumId}/comments/{commentId}/answers/{answerId}: editar respuesta
- DELETE /forums/{forumId}/comments/{commentId}/answers/{answerId}: eliminar respuesta

Sprint 15: módulo de quiz Funcionalidades:

- GET /quiz/results: obtención de resultados de evaluación vocacional
- DELETE /quiz/results: eliminación para reiniciar evaluación

Sprint 16: módulo de analíticas Funcionalidad:

- GET /analytics: métricas agregadas de progreso del estudiante:
 - Carreras exploradas
 - Contenido interactuado
 - Tiempo invertido en plataforma
 - Recomendaciones de siguientes pasos

Prácticas de desarrollo

Para cada módulo se aplicarán:

- **Principio de responsabilidad única:** cada Lambda tiene una función específica
- **DRY (Don't Repeat Yourself):** funciones auxiliares compartidas en layers de Lambda
- **Validación de entrada:** mediante esquemas Mongoose antes de procesar
- **Manejo de errores:** Try-catch con mensajes descriptivos y códigos HTTP apropiados
- **Logging:** CloudWatch Logs para todas las funciones con niveles (info, warn, error)

Entregables de la fase

- 42 funciones Lambda desplegadas y operativas
- 42+ endpoints REST funcionales y documentados
- Microservicio de IA integrado con VPC Link
- Sistema de recomendación personalizado funcional
- Suite de pruebas unitarias para cada módulo
- Documentación de APIs con ejemplos de uso

6.3.6. Fase 6: pruebas de rendimiento y validación

Objetivos de la fase

Validar que el sistema cumple con los objetivos específicos de rendimiento, escalabilidad y seguridad mediante pruebas exhaustivas que simulen condiciones de uso real.

Tipos de pruebas de rendimiento

Pruebas de carga fija (Fixed Load Testing) Objetivo: medir rendimiento bajo carga constante y predecible.

Configuración:

- Usuarios virtuales concurrentes: 100, 500, 1000, 2000
- Duración: 60 segundos por prueba
- Endpoints críticos: GET /users/me, GET /explore/feed, GET /careers

Métricas a medir:

- Tiempo de respuesta promedio (objetivo: < 500ms)
- Throughput (solicitudes por segundo)
- Tasa de error (objetivo: < 5 %)
- Percentiles p50, p90, p95, p99

Pruebas de incremento progresivo (Ramp Up Testing) Objetivo: evaluar cómo escala el sistema ante crecimiento gradual de carga.

Configuración:

- Inicio: 10 usuarios concurrentes
- Incremento: +50 usuarios cada 30 segundos
- Máximo: 2000 usuarios concurrentes
- Duración total: 120 segundos

Análisis:

- Identificación del punto donde la latencia comienza a degradarse
- Medición de efectividad del auto-scaling de Lambda
- Detección de cuellos de botella en base de datos

Pruebas de estrés (Stress Testing) Objetivo: determinar el punto de quiebre del sistema.

Configuración:

- Incremento agresivo desde 0 hasta 6000+ usuarios concurrentes
- Duración: 120 segundos
- Endpoints mixtos: GET, POST, PUT, DELETE

Métricas críticas:

- Punto donde tasa de error excede 20 %
- Detección de throttling de Lambda
- Tiempo de recuperación post-estrés

Pruebas de picos súbitos (Spike Testing) Objetivo: evaluar respuesta ante incrementos bruscos de tráfico.

Configuración:

- Carga base: 200 usuarios concurrentes
- Pico súbito: 1900 usuarios concurrentes durante 30 segundos
- Retorno a base: 200 usuarios

Análisis:

- Impacto de cold starts de Lambda
- Efectividad de CloudFront CDN para absorber picos
- Recuperación de rendimiento post-pico

Pruebas de picos graduales (Peak Testing) Objetivo: simular crecimiento natural con pico sostenido.

Configuración:

- Carga base: 200 usuarios
- Crecimiento progresivo hasta 2000 usuarios en 60 segundos
- Sostenimiento de pico durante 60 segundos
- Decrecimiento progresivo a 200 usuarios en 60 segundos

Pruebas de resistencia (Soak Testing) Objetivo: validar estabilidad bajo carga prolongada.

Configuración:

- Carga constante: 500 usuarios concurrentes
- Duración: 60 minutos
- Solicitudes totales esperadas: ~88,000

Análisis:

- Detección de fugas de memoria
- Degradación progresiva de rendimiento
- Agotamiento de conexiones a base de datos

Herramienta de pruebas

Se utilizará Postman Collection Runner por:

- Capacidad de configurar diferentes perfiles de carga
- Generación automática de gráficos de tendencias
- Exportación de resultados en formato estructurado
- Integración con flujos de CI/CD

Pruebas de seguridad

Validación de encriptación

1. Interceptación de tráfico de red con Wireshark
2. Verificación de que datos en tránsito están encriptados con TLS 1.3
3. Acceso directo a DocumentDB para validar encriptación en reposo
4. Verificación de que PII está hasheada con SHA-256

Validación de políticas de seguridad

1. Verificación de rotación de claves (simulación de 90 días)
2. Validación de rate limiting (excediendo 1000 req/5min)
3. Pruebas de aislamiento de red (intentos de acceso directo a DocumentDB)
4. Validación de expiración de tokens JWT

Criterios de aceptación

El sistema se considerará validado si cumple:

Cuadro 6.3. Criterios de aceptación para validación del sistema

Métrica	Objetivo
Tiempo de respuesta promedio	< 500ms
Usuarios concurrentes soportados	≥ 2000
Tasa de error bajo carga normal	< 5 %
Degradación bajo estrés	< 20 %
Disponibilidad	> 99.9 %
Tasa de error de comunicación con IA	< 1 %
Encriptación de PII validada	100 %

Entregables de la fase

- Reporte completo de pruebas de rendimiento con gráficos de tendencias
- Identificación de punto de quiebre del sistema
- Documentación de comportamiento bajo diferentes perfiles de carga
- Reporte de pruebas de seguridad con evidencias
- Lista de optimizaciones identificadas para trabajo futuro
- Certificación de cumplimiento de objetivos específicos

6.4. Herramientas y recursos

6.4.1. Herramientas de desarrollo

- **IDE:** Cursor:
 - ESLint para análisis estático de código
 - Prettier para formateo consistente
 - AWS Toolkit para gestión de recursos AWS
- **Control de versiones:** Git con GitHub para:
 - Repositorio privado con historial completo

- Commits atómicos con mensajes descriptivos
- Branches por feature para desarrollo aislado
- Pull requests para revisión de código (auto-revisión)
- **Gestión de dependencias:** npm para paquetes Node.js, pip para Python del microservicio
- **Pruebas de API:** Postman con colecciones organizadas por módulo

6.4.2. Herramientas de despliegue

- **AWS CLI:** para despliegue de funciones Lambda y configuración de servicios
- **AWS Console:** para configuración visual de VPC, Security Groups, API Gateway
- **SSH:** para acceso a instancias EC2 (DocumentDB y microservicio)

6.4.3. Herramientas de monitoreo

- **AWS CloudWatch:** logs centralizados, métricas de rendimiento, alarmas
- **CloudWatch Insights:** análisis de logs con lenguaje de consulta

6.4.4. Recursos de infraestructura

Cuadro 6.4. Recursos de infraestructura AWS utilizados

Recurso	Especificación	Propósito
VPC	us-east-2	Red privada aislada
Subredes	2 públicas, 2 privadas	Aislamiento de recursos
EC2 (DocumentDB)	t3.micro, 30GB EBS	Base de datos
EC2 (Microservicio)	t3.micro	Procesamiento IA
Lambda Functions	42 funciones, 128-512MB	Lógica de negocio
API Gateway	HTTP API	Punto de entrada
CloudFront	Distribución global	CDN y protección
WAF	Reglas gestionadas	Firewall aplicación
Route 53	Hosted zone	DNS
Certificate Manager	Certificados SSL/TLS	Encriptación
CloudWatch	Logs y métricas	Monitoreo

6.5. Gestión del proyecto

6.5.1. Backlog del producto

El backlog se organiza en tres niveles de prioridad:

1. **Prioridad alta (Must-have):** funcionalidades críticas que cumplen objetivos específicos
 - Sistema de autenticación y autorización
 - CRUD de usuarios con encriptación de PII
 - Catálogo de carreras con información completa

- API Gateway con Lambda Authorizer
- Base de datos con esquemas validados

2. **Prioridad media (Should-have):** funcionalidades importantes para experiencia completa

- Sistema de recomendaciones con algoritmo personalizado
- Foros de discusión con jerarquía de comentarios
- Sistema de interacciones y guardado de contenido
- Microservicio de IA
- Analíticas de progreso del estudiante

3. **Prioridad baja (Could-have):** funcionalidades complementarias

- Sistema de likes/unlikes en tarjetas
- Quiz vocacional completo
- Testimonios de estudiantes
- Optimizaciones avanzadas de rendimiento

6.5.2. Definición de completitud (Definition of Done)

Una funcionalidad se considera completa cuando:

1. Código implementado siguiendo estándares de ESLint
2. Esquema Mongoose validado con tipos estrictos
3. Función Lambda desplegada y accesible via API Gateway
4. Pruebas manuales exitosas con Postman (casos positivos y negativos)
5. Manejo de errores implementado con mensajes descriptivos
6. Logs configurados en CloudWatch
7. Documentación de endpoint actualizada con ejemplos
8. Código commiteado a GitHub con mensaje descriptivo

6.5.3. Gestión de riesgos

Se identifican los siguientes riesgos técnicos principales:

Cuadro 6.5. Matriz de riesgos técnicos

Riesgo	Prob.	Mitigación
Cold starts excesivos en Lambda	Media	Implementar warming con Cloud-Watch Events
Límite de concurrencia Lambda	Alta	Solicitar aumento de cuota a AWS anticipadamente
Complejidad de Security Groups	Media	Documentar exhaustivamente, validar con diagrams
Costos inesperados en AWS	Baja	Configurar alertas de billing, monitoreo diario
Incompatibilidad DocumentDB-MongoDB	Baja	Validar compatibilidad con features usados
Latencia de modelos IA externos	Media	Implementar timeouts y fallbacks

6.5.4. Cronograma estimado

Cuadro 6.6. Cronograma estimado por fases

Fase	Duración	Sprints
Investigación y análisis	4 semanas	2
Diseño arquitectónico	2 semanas	1
Configuración infraestructura	2 semanas	1
Componentes Core	2 semanas	1
Módulos funcionales	8 semanas	4
Pruebas y validación	2 semanas	1
Total	20 semanas	10

6.6. Métricas de éxito

El éxito de la metodología se medirá mediante:

6.6.1. Métricas de proceso

- **Velocity:** puntos de historia completados por sprint (objetivo: estabilizar en sprint 3)
- **Sprint burndown:** progreso diario dentro de cada sprint
- **Defect rate:** errores detectados post-despliegue (objetivo: < 5 %)
- **Code coverage:** cobertura de pruebas unitarias (objetivo: > 60 % para lógica crítica)

6.6.2. Métricas de producto

- **Funcionalidades completadas:** 42+ endpoints funcionales
- **Rendimiento:** latencia promedio < 500ms
- **Escalabilidad:** soporte para 2000+ usuarios concurrentes
- **Seguridad:** 0 vulnerabilidades críticas en auditoría
- **Disponibilidad:** > 99.9 % uptime

6.6.3. Métricas de calidad

- **Documentación:** 100 % de endpoints documentados
- **Código:** cumplimiento de estándares ESLint sin warnings
- **Commits:** promedio de 1-3 commits por día de desarrollo activo
- **Deuda técnica:** registro y priorización de mejoras identificadas

6.7. Consideraciones éticas

6.7.1. Protección de datos de menores

Dado que el sistema manejará datos de estudiantes graduandos (muchos menores de 18 años), se implementarán salvaguardas adicionales:

- Encriptación reforzada para toda información personal identificable
- No almacenamiento de datos sensibles innecesarios
- Políticas claras de retención de datos
- Mecanismos de consentimiento parental (para trabajo futuro)

6.7.2. Sesgos en recomendaciones de IA

Se reconoce el riesgo de que modelos de IA externos perpetúen sesgos de género, socioeconómicos o culturales. Para mitigar:

- Diversificar fuentes de modelos de IA
- Implementar auditorías periódicas de recomendaciones
- Proveer explicaciones transparentes de por qué se recomienda cada carrera
- Permitir feedback de usuarios sobre pertinencia de recomendaciones

6.7.3. Accesibilidad e inclusión

El sistema debe ser accesible para estudiantes de diversos contextos:

- Optimización para conexiones lentas (compresión, caché)
- Soporte para idioma más popular de Guatemala (español)
- Contenido culturalmente relevante para Guatemala

6.8. Limitaciones de la metodología

Se reconocen las siguientes limitaciones:

1. **Desarrollo individual:** Scrum está diseñado para equipos. La adaptación para un desarrollador elimina beneficios de colaboración y revisión por pares.
2. **Ausencia de stakeholders externos:** sin usuarios reales durante desarrollo, no hay retroalimentación de usuario final en sprints.
3. **Alcance técnico:** la metodología se enfoca en implementación de backend. Diseño de frontend, estudios de usabilidad y validación con usuarios reales quedan fuera de alcance.
4. **Pruebas automatizadas limitadas:** por restricciones de tiempo, se priorizan pruebas manuales sobre test automation completo.
5. **Contexto académico:** el proyecto es un prototipo de graduación, no un producto comercial. Algunas decisiones priorizan demostración de competencias técnicas sobre optimización absoluta.

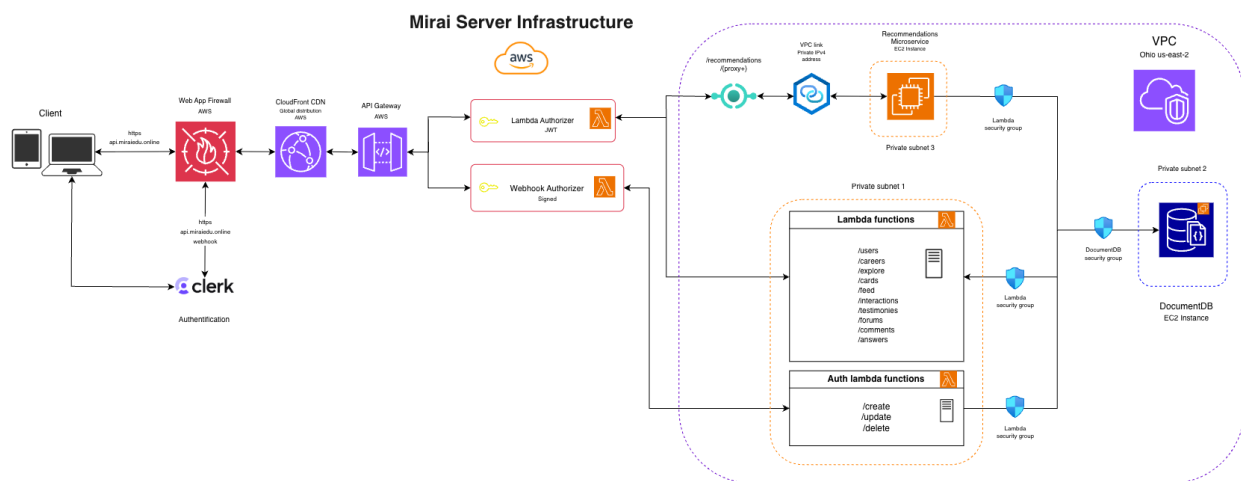
7.1. Prototipo

El resultado del proceso de desarrollo fue un sistema backend completo y funcional implementado sobre infraestructura cloud de AWS, con arquitectura serverless que garantiza escalabilidad, seguridad y mantenibilidad. A continuación se detallan los componentes implementados y su funcionamiento.

7.1.1. Infraestructura cloud implementada

La infraestructura implementada en AWS se compone de múltiples servicios interconectados que trabajan en conjunto para proporcionar un backend robusto y seguro.

Figura 7.1. Infraestructura del prototipo del sistema integrador



AWS WAF (Web Application Firewall) Se implementó un Web Application Firewall como primera línea de defensa contra ataques web. El WAF está configurado con:

- **Reglas de protección SQL Injection:** detección y bloqueo automático de intentos de inyección SQL en parámetros, headers y body de solicitudes
- **Protección Cross-Site Scripting (XSS):** prevención de scripts maliciosos en entradas de usuario
- **Rate Limiting:** control de tasa de solicitudes por IP (máximo 1000 solicitudes por 5 minutos por IP)
- **Bot Protection:** detección y bloqueo de bots maliciosos y scrapers automatizados
- **Geographic Restrictions:** capacidad de restringir acceso por país/región
- **IP Reputation Lists:** bloqueo automático de IPs conocidas por actividades maliciosas
- **Size Restrictions:** límites en tamaño de solicitudes para prevenir ataques de buffer overflow
- **Request Filtering:** filtrado de solicitudes basado en patrones sospechosos
- **Logging completo:** registro detallado de todas las solicitudes bloqueadas y permitidas

Amazon CloudFront CDN Se configuró CloudFront como Content Delivery Network global con las siguientes características:

- **Distribución global:** 400+ edge locations en 90+ países para entrega rápida de contenido
- **Integración con WAF:** protección centralizada a través de la red global de CloudFront
- **Conexión obligatoria con HTTP API Gateway:** CloudFront es necesario para conectar WAF con HTTP API Gateway, ya que HTTP API Gateway no soporta integración directa con WAF
- **SSL/TLS Termination:** terminación SSL con certificados gestionados automáticamente por AWS Certificate Manager
- **Caché inteligente:** estrategias de caché optimizadas para APIs HTTP de alto rendimiento
- **Compresión automática:** compresión Gzip/Brotli para reducir ancho de banda
- **HTTP/2 y HTTP/3:** soporte para protocolos HTTP modernos
- **Headers de seguridad:** implementación automática de headers de seguridad (HSTS, CSP, X-Frame-Options)
- **DDoS Protection:** mitigación automática de ataques DDoS a nivel de red
- **Origin Shield:** capa adicional de caché para reducir carga en HTTP API Gateway
- **Real-time Metrics:** métricas en tiempo real de rendimiento y seguridad
- **Optimización para HTTP APIs:** configuración específica para APIs HTTP de alto throughput

Amazon HTTP API Gateway Se configuró un HTTP API Gateway que actúa como punto de entrada único para todas las solicitudes del cliente. El gateway está configurado con:

- **Dominio personalizado:** `https://api.miraiedu.online` (apuntando a CloudFront)
- **Tipo HTTP API:** utiliza HTTP API Gateway en lugar de REST API para mejor rendimiento y menor latencia
- **Certificado SSL/TLS:** configurado para todas las comunicaciones HTTPS
- **Lambda Authorizer:** integrado para validación de tokens JWT de Clerk antes de procesar cualquier solicitud
- **Webhook Authorizer:** authorizer separado para validar webhooks de Clerk con verificación de firma
- **Integración con Lambda:** 42+ endpoints configurados, cada uno mapeado a su función Lambda correspondiente
- **VPC Link:** conexión directa al microservicio de recomendaciones en EC2 dentro de la VPC
- **CORS:** políticas configuradas para permitir acceso desde el cliente web/móvil
- **Rate Limiting:** límites de tasa configurados para prevenir abuso
- **Logging:** integrado con CloudWatch para monitoreo de solicitudes
- **Request/Response Transformation:** transformación de datos para optimización
- **API Versioning:** soporte para versionado de APIs
- **Alto rendimiento:** HTTP API Gateway ofrece mejor rendimiento que REST API Gateway
- **Costo optimizado:** menor costo por millón de solicitudes comparado con REST API Gateway

El flujo completo de procesamiento de solicitudes es:

1. Cliente envía solicitud HTTPS a `https://api.miraiedu.online`
2. **AWS WAF** analiza la solicitud y aplica reglas de seguridad
3. Si pasa el WAF, **CloudFront** procesa la solicitud (caché, compresión, etc.)
4. CloudFront reenvía la solicitud a **HTTP API Gateway**
5. HTTP API Gateway invoca Lambda Authorizer con el token JWT
6. Valida firma del token con clave pública RSA de Clerk
7. Si es válido, extrae `user_id` y lo pasa a la función Lambda
8. Invoca la función Lambda correspondiente al endpoint
9. La respuesta viaja de regreso: Lambda → HTTP API Gateway → CloudFront → WAF → Cliente

Virtual Private Cloud (VPC) Se creó una VPC en la región `us-east-2` con la siguiente configuración:

- **CIDR Block:** rango de IPs privadas para la red virtual
- **Subredes públicas:** para recursos que requieren acceso a internet (NAT Gateway)
- **Subredes privadas:** para recursos internos (DocumentDB, microservicio de recomendaciones)
- **Internet Gateway:** permite comunicación con internet
- **Route Tables:** configuradas para dirigir tráfico apropiadamente
- **Network ACLs:** listas de control de acceso para capa adicional de seguridad

La VPC proporciona aislamiento de red completo para los recursos sensibles, evitando exposición directa a internet.

Funciones AWS Lambda Se implementaron 40 funciones Lambda serverless, organizadas por módulos funcionales. Cada función Lambda está configurada con:

- **Runtime:** Node.js v.22
- **Memoria:** 128-512 MB según complejidad
- **Timeout:** 10-30 segundos
- **Variables de entorno:** URI de base de datos, claves de encriptación
- **Rol IAM:** permisos mínimos necesarios para acceder a recursos
- **VPC Configuration:** conectadas a subredes privadas para acceso a DocumentDB
- **Security Groups:** asociadas a grupos de seguridad que permiten tráfico a DocumentDB
- **Layers:** capa compartida con `global-bundle.pem` para conexión SSL a DocumentDB

Las funciones Lambda se empaquetan individualmente con sus dependencias (Mongoose, utilidades de encriptación, etc.), lo que permite:

- Despliegues independientes sin afectar otras funciones
- Optimización de tamaño de paquete
- Versionado granular
- Rollback selectivo en caso de errores

Amazon DocumentDB La base de datos se implementó sobre una instancia EC2 dentro de la VPC ejecutando DocumentDB (basado en MongoDB). Esta es la implementación que AWS recomienda en su documentación. Configuración:

- **Instancia:** EC2 en subred privada de la VPC
- **Motor:** DocumentDB compatible con MongoDB 4.0

- **Storage:** almacenamiento EBS con encriptación en reposo
- **Security Group:** configurado para aceptar conexiones solo desde funciones Lambda
- **Puerto:** 27017 (MongoDB estándar)
- **SSL/TLS:** conexiones cifradas obligatorias usando `global-bundle.pem`
- **Backups:** snapshots automáticos diarios
- **Credenciales:** almacenadas en variables de entorno de Lambda

El Security Group de DocumentDB tiene reglas estrictas:

- **Inbound:** solo acepta tráfico en puerto 27017 desde el Security Group de Lambda
- **Outbound:** permite respuestas al Security Group de Lambda
- **Sin acceso público:** no hay IP pública asignada, totalmente aislado de internet

Microservicio de recomendaciones El microservicio de recomendaciones se desplegó en una instancia EC2 separada dentro de la VPC:

- **Instancia EC2:** tipo t3.micro en subred privada
- **Sistema operativo:** Amazon Linux 2
- **Runtime:** Node.js con algoritmo de recomendación
- **Security Group:** acepta tráfico solo desde VPC Link
- **VPC Link:** conexión directa desde API Gateway a la instancia EC2
- **Puerto:** 3001 (HTTP interno)
- **Health Checks:** verificación periódica de disponibilidad

El flujo de comunicación para recomendaciones es:

1. API Gateway recibe solicitud en `/recommendations`
2. Valida autenticación con Lambda Authorizer
3. Utiliza VPC Link para conectar hacia la instancia de EC2
4. Se accede a la subred privada que aloja al servicio
5. Microservicio procesa algoritmo y consulta DocumentDB si necesario
6. Respuesta viaja de regreso por la misma ruta

Sistema de autenticación con Clerk La integración con Clerk proporciona autenticación robusta:

- **Lambda Authorizer:** función que valida tokens JWT
- **Verificación RSA:** valida firma del token con clave pública de Clerk
- **Extracción de claims:** obtiene `user_id` del payload del token
- **Policy Generation:** genera política IAM permitiendo/denegando acceso
- **Context Injection:** inyecta `user_id` en contexto de la solicitud
- **Webhook Authorizer:** función separada para validar eventos de Clerk mediante HMAC
- **Sincronización:** webhooks mantienen usuarios sincronizados con Clerk

Listing 7.1. Flujo de Lambda Authorizer

```
// Extraer token del header Authorization
const token = event.authorizationToken;

// Verificar firma con clave pública RSA
const decoded = jwt.verify(token, publicKey);

// Extraer user_id
const userId = decoded.sub;

// Generar política IAM
const policy = generatePolicy(userId, 'Allow',
    event.methodArn);

// Inyectar contexto
policy.context = { userId };

return policy;
```

Security Groups Se configuraron Security Groups específicos para cada componente:

1. Lambda Security Group:

- *Outbound:* permite tráfico hacia DocumentDB (puerto 27017)
- *Outbound:* permite tráfico HTTPS hacia internet (para integraciones)
- *Inbound:* no aplica (Lambda no recibe conexiones directas)

2. DocumentDB Security Group:

- *Inbound:* solo acepta conexiones desde Lambda Security Group en puerto 27017
- *Outbound:* permite respuestas al Lambda Security Group

3. Microservicio EC2 Security Group:

- *Inbound:* acepta tráfico HTTP (puerto 3001) solo desde VPC Link
- *Outbound:* permite conexiones a DocumentDB para consultas
- *Outbound:* permite tráfico HTTPS para dependencias

Esta configuración de Security Groups implementa defensa en profundidad, donde cada capa solo puede comunicarse con las capas explícitamente permitidas.

7.1.2. Arquitectura de la solución

La arquitectura implementada sigue el patrón de microservicios serverless con las siguientes características:

Flujo de autenticación

1. Usuario inicia sesión en aplicación cliente usando Clerk
2. Clerk genera token JWT firmado con clave privada RSA
3. Cliente envía solicitud HTTPS a `https://api.miraiedu.online` incluyendo token en header **Authorization: Bearer <token>**
4. **AWS WAF** analiza la solicitud y valida contra reglas de seguridad
5. Si pasa el WAF, **CloudFront** procesa la solicitud (caché, compresión, headers de seguridad)
6. CloudFront reenvía la solicitud a **HTTP API Gateway**
7. HTTP API Gateway invoca Lambda Authorizer
8. Lambda Authorizer verifica firma del token con clave pública RSA de Clerk
9. Si es válido, genera política IAM de **Allow** y extrae **user_id**
10. HTTP API Gateway permite solicitud y pasa **user_id** a función Lambda
11. Función Lambda usa **user_id** para operaciones de base de datos
12. La respuesta viaja de regreso a través de CloudFront y WAF hacia el cliente

Flujo de sincronización con Webhooks

1. Evento ocurre en Clerk (ej: usuario actualiza perfil)
2. Clerk envía POST request a endpoint webhook con firma en el header
3. Solicitud llega a `https://api.miraiedu.online/auth`
4. **AWS WAF** analiza la solicitud webhook y valida contra reglas de seguridad
5. Si pasa el WAF, **CloudFront** procesa la solicitud
6. CloudFront reenvía la solicitud a **HTTP API Gateway**
7. HTTP API Gateway invoca Webhook Authorizer
8. Webhook Authorizer valida firma con secret compartido
9. Si es válido, invoca función Lambda de procesamiento
10. Lambda actualiza datos en DocumentDB
11. Respuesta de confirmación viaja de regreso a Clerk a través de CloudFront y WAF

Patrón de encriptación de datos Se implementó un sistema de doble encriptación para datos sensibles:

1. **Encriptación en tráfico:**

- Cliente encripta datos sensibles antes de enviar
- Usa biblioteca `cryptojs`
- Encriptación TLS 1.3
- Lambda desencripta al recibir usando misma biblioteca

2. **Encriptación en reposo:**

- Lambda encripta datos personales antes de guardar en BD
- Usa AES-256 irreversible para PII identificable
- Usa biblioteca `crypto`
- Al consultar, desencripta para mostrar al usuario autorizado

Listing 7.2. Ejemplo de encriptación

```
// En registro de usuario
const encryptedEmail = encrypt(user.email, SECRET_KEY);
const hashedName = sha256(user.name);

await User.create({
  clerk_id: userId,
  email: encryptedEmail,
  name: hashedName,
  ...
});

// En consulta de usuario
const user = await User.findOne({ clerk_id: userId });
const decryptedEmail = decrypt(user.email, SECRET_KEY);

return {
  ...user,
  email: decryptedEmail
};
```

7.1.3. Módulos implementados

El desarrollo completo del sistema se realizó en 100+ commits documentados en el repositorio de Github: <https://github.com/JDgomez2002/mirai-server.git>. A continuación se detallan los módulos implementados:

Módulo de autenticación (Auth) Implementación de funcionalidades de gestión de cuentas de usuario:

- **POST /auth/register** - Registro de usuarios
 - Validación de datos con esquema Mongoose

- Verificación de duplicados por `clerk_id`
 - Encriptación de datos personales (email, nombre, teléfono)
 - Almacenamiento de rol de usuario (student, admin)
 - Integración con Clerk para sincronización
 - *Commits*: 67a1233 (encriptación), 7ac97bf (roles), 4c4363d (esquema)
- **POST /auth/update** - Actualización de usuario
 - Validación de permisos (solo el mismo usuario o admin)
 - Encriptación de campos actualizados
 - Actualización parcial de campos
 - Manejo de tags de interés del usuario
 - *Commits*: 79a15f6 (implementación), 31ee6c0 (refactor)
 - **POST /auth/delete** - Eliminación de cuenta
 - Validación de permisos
 - Eliminación cascada de datos relacionados
 - Sincronización con Clerk
 - *Commit*: d47f732

Módulo de usuarios (Users) Sistema completo de gestión de perfiles y preferencias:

- **GET /users/me** - Obtener perfil
 - Consulta por `user_id` del JWT
 - Desencriptación de datos personales
 - Inclusión de tags de interés
 - Población de datos relacionados
 - *Commits*: f051efc, 665fefb, 91c8725
- **GET /users** - Listar usuarios
 - Solo accesible para administradores
 - Paginación de resultados
 - Desencriptación de datos
 - Filtrado por rol
 - *Commit*: 03231ce
- **PATCH /users/{id}** - Editar rol
 - Solo administradores pueden cambiar roles
 - Validación de rol válido (student, admin)
 - Registro de auditoría
 - *Commit*: b871096
- **POST /users/saved** - Guardar item
 - Soporta guardar carreras y tarjetas

- Prevención de duplicados
- Almacenamiento con timestamp
- *Commit: 4d10132*
- **DELETE** `/users/saved/{id}` - Desguardar item
 - Eliminación de item guardado
 - Validación de propiedad
 - *Commit: 8c436ae*
- **GET** `/users/saved` - Obtener guardados
 - Lista de items guardados por usuario
 - Población de datos completos de carreras/tarjetas
 - Ordenamiento por fecha de guardado
 - *Commit: 3c08f9e*

Módulo de carreras (Careers) Catálogo completo de carreras universitarias:

- **GET** `/careers` - Listar carreras
 - Obtención de catálogo completo
 - Filtrado por facultad, modalidad
 - Inclusión de metadata (duración, requisitos)
 - *Commit: 3e57f05*
- **GET** `/careers/{id}` - Detalle de carrera
 - Información completa de carrera específica
 - Población de cursos asociados
 - Población de testimonios de alumnos
 - Información de insights y proyecciones
 - *Commits: bf64b73, 8dbea39*
- **PATCH** `/careers/{id}` - Editar insights
 - Solo administradores pueden editar
 - Actualización de información de mercado laboral
 - Actualización de proyecciones salariales
 - Actualización de habilidades demandadas
 - *Commits: 92d4ba0, a08e461*

Módulo de exploración (Explore) Sistema de contenido dinámico con tarjetas interactivas:

- **POST /explore/cards** - Crear tarjeta
 - Creación de tarjetas tipo: career, testimony, what_if
 - Validación de estructura por tipo
 - Asignación de prioridad
 - Sistema de etiquetado (tags)
 - *Commits*: 5d1cd1c, 27d159f
- **GET /explore/cards/{id}** - Obtener tarjeta
 - Consulta de tarjeta específica por ID
 - Inclusión de metadata completa
 - *Commits*: d47f732, 4e72f8b
- **PUT /explore/cards/{id}** - Editar tarjeta
 - Solo testimonios y what_if pueden ser editados
 - Solo creador o admin pueden editar
 - Actualización parcial de campos
 - *Commits*: 6bd4951, baa052d
- **DELETE /explore/cards/{id}** - Eliminar tarjeta
 - Solo testimonios y what_if pueden ser eliminados
 - Validación de permisos
 - Eliminación suave (soft delete)
 - *Commits*: 76f64f2, 5591c96
- **GET /testimonies** - Obtener testimonios
 - Filtrado de tarjetas tipo testimony
 - Ordenamiento por relevancia
 - *Commit*: 4b24898
- **POST /explore/interactions** - Registrar interacción
 - Tipos: view, tap, save, share, like, unlike
 - Almacenamiento de duración de interacción
 - Metadata adicional (scroll depth, time on card)
 - Extracción de tags de la tarjeta para perfil de usuario
 - *Commits*: d47f732, c77cb53, c17427d, cd5c472

Sistema de Likes y Unlikes Implementación de funcionalidad de likes para tarjetas de contenido:

- **PATCH /explore/cards/{cardId}/likes** - Manejar likes/unlikes
 - Acciones: like, unlike
 - Actualización de contador de likes en tarjeta
 - Almacenamiento de likes del usuario en colección de usuarios
 - Prevención de likes duplicados
 - Validación de existencia de tarjeta
 - Extracción de user_id del JWT
 - *Commits*: 97d5f2e, f4cd037

Sistema de Feed con algoritmo de recomendación Uno de los componentes más complejos implementados:

- **GET /explore/feed** - feed personalizado
 - **Análisis de interacciones:** consulta histórico de interacciones del usuario
 - **Extracción de preferencias:** identifica tags más interactuados
 - **Scoring de contenido:** calcula relevancia de cada tarjeta:
 - Coincidencia de tags con perfil: +10 puntos por tag
 - Tipo de contenido preferido: +5 puntos
 - Prioridad de la tarjeta: multiplicador
 - Novedad: bonus para contenido reciente
 - **Ordenamiento inteligente:** por score descendente
 - **Diversidad:** mezcla de tipos de contenido
 - **Paginación:** límite y offset configurables
 - **Filtrado:** por tipos específicos de contenido
 - **Integración con NLP:** usa bibliotecas `natural` y `compromise` para similitud semántica
 - *Commits:* 3303ce4, 71f9db8

Listing 7.3. Algoritmo de Scoring

```
// Obtener interacciones del usuario
const interactions = await Interaction.find({ userId })
  .populate('cardId');

// Extraer tags de inter s
const userTags = {};
interactions.forEach(int => {
  int.cardId.tags.forEach(tag => {
    userTags[tag] = (userTags[tag] || 0) + 1;
  });
});

// Calcular score para cada tarjeta
cards.forEach(card => {
  let score = card.priority || 0;

  // Bonus por coincidencia de tags
  card.tags.forEach(tag => {
    if (userTags[tag]) {
      score += userTags[tag] * 10;
    }
  });

  // Bonus por novedad
  const daysSinceCreated =
    (Date.now() - card.created_at) / (1000 * 60 * 60 * 24);
  if (daysSinceCreated < 7) {
    score += 20;
  }
});
```

```

    card.score = score;
  });

// Ordenar por score
cards.sort((a, b) => b.score - a.score);

```

Módulo de foros (Forums) Sistema completo de foros comunitarios con jerarquía de comentarios:

- **POST /forums** - Crear foro
 - Extracción de `user_id` del JWT como creador
 - Asociación con carrera específica
 - Validación de contenido
 - *Commits*: 99da919, 289ba6e
- **GET /forums/{id}** - Obtener foro
 - Población de datos de creador (desencriptados)
 - Población de carrera asociada
 - Población de comentarios con usuarios
 - Población de respuestas a comentarios
 - Estructura jerárquica completa
 - *Commits*: 95fbf60, 33a8220, f8dcbdf
- **GET /forums** - Listar foros
 - Lista paginada de foros
 - Desencriptación de datos de usuarios
 - Filtrado por carrera
 - Ordenamiento por actividad reciente
 - *Commits*: cee43e5, 869a84c
- **PUT /forums/{id}** - Editar foro
 - Validación: solo creador o admin
 - Actualización de título y contenido
 - Registro de última modificación
 - *Commits*: f78b95f, 716cb1a
- **DELETE /forums/{id}** - Eliminar foro
 - Validación: solo creador o admin
 - Eliminación cascada de comentarios
 - *Commits*: 24356e4, 64cfa11
- **POST /forums/{forumId}/comments** - Nuevo comentario
 - Extracción de `user_id` del JWT
 - Asociación con foro padre

- Notificación al creador del foro
- *Commits*: 3b289e6, 0c9d8a9, 2a82562
- **PUT** /forums/{forumId}/comments/{commentId} - Editar comentario
 - Solo el creador puede editar
 - Marca de editado con timestamp
 - *Commit*: f4efecc
- **DELETE** /forums/{forumId}/comments/{commentId} - Eliminar comentario
 - Solo creador o admin
 - Eliminación de respuestas asociadas
 - *Commit*: 6d314a2
- **POST** /forums/{forumId}/comments/{commentId}/answers - Nueva respuesta
 - Respuesta a comentario específico
 - Notificación al autor del comentario
 - *Commits*: 9b10c8f, 78ba025
- **PUT** /forums/{forumId}/comments/{commentId}/answers/{answerId} - Editar respuesta
 - Solo el creador puede editar
 - Registro de edición
 - *Commit*: 8b13888
- **DELETE** /forums/{forumId}/comments/{commentId}/answers/{answerId} - Eliminar respuesta
 - Solo creador o admin
 - Soft delete
 - *Commit*: b2a8ff6

Módulo de quiz Sistema de evaluación vocacional:

- **GET** /quiz/results - Obtener resultados
 - Consulta de resultados de quiz del usuario
 - Análisis de preferencias vocacionales
 - Sugerencias de carreras basadas en resultados
 - *Commit*: a75f939
- **DELETE** /quiz/results - Eliminar resultados
 - Permite al usuario reiniciar quiz
 - Eliminación completa de resultados previos
 - *Commit*: d2afe12

Módulo de analíticas Sistema de métricas y seguimiento de progreso:

- **GET /analytics** - Analíticas del estudiante
 - Resumen de actividad del usuario
 - Carreras exploradas
 - Contenido interactuado
 - Tiempo invertido en exploración
 - Progreso en orientación vocacional
 - Recomendaciones de siguientes pasos
 - Agregaciones complejas de múltiples colecciones
 - *Commit:* ec49181

Middleware de autenticación Componentes transversales para seguridad:

- **Lambda Authorizer** - Validación JWT
 - Verificación de firma RSA de tokens Clerk
 - Extracción de `user_id` del payload
 - Generación de política IAM dinámica
 - Inyección de contexto de usuario
 - Soporte para múltiples endpoints con ARN patterns
 - *Commits:* d47f732, 425eea2, ffc6b14, be646fd
- **Webhook Authorizer** - Validación de webhooks
 - Verificación de firma HMAC de Clerk
 - Validación de timestamp para prevenir replay attacks
 - Autorización de eventos específicos
 - *Commit:* 0ae4d0c

Utilidades de seguridad Bibliotecas compartidas para encriptación:

- **repose.crypto.js** - Encriptación en reposo
 - Funciones de encriptación SHA-256
 - Hash irreversible para PII
 - Funciones de comparación segura
 - *Commit:* e6314bd
- **traffic.crypto.js** - Encriptación de tráfico
 - Encriptación AES-256 bidireccional
 - Funciones de encriptación/desencryptación
 - Gestión de vectores de inicialización
 - *Commit:* 3288017

- **crypto.utils.js** - Utilidades compartidas
 - Funciones auxiliares de encriptación
 - Validación de integridad de datos
 - Presente en múltiples módulos
 - *Commits*: múltiples módulos

7.1.4. Estadísticas del proyecto

El desarrollo completo del sistema se refleja en las siguientes métricas:

- **Commits**: 100+ commits documentados
- **Funciones Lambda**: 42 funciones serverless
- **Endpoints**: 42+ endpoints API
- **Líneas de código**: 15,400 líneas (backend)
- **Dependencias**: 20+ paquetes npm por función
- **Modelos de datos**: 12 esquemas Mongoose
- **Tiempo de desarrollo**: 6 meses
- **Desarrollador**: 1 (trabajo individual)

7.1.5. Cronología de desarrollo

El desarrollo siguió una progresión lógica documentada en el historial de Git:

1. **Inicialización** (Mayo 2025):
 - Commit inicial: **aa280c9**
 - Documentación de arquitectura: **3774f37**, **5ae8071**
 - Diseño de infraestructura: **baba56c**
 - Prototipo inicial: **8d54af0**, **b79ec9b**
2. **Infraestructura Lambda** (Agosto 2025):
 - Despliegue de funciones base: **d47f732**
 - Middleware de autenticación: **d47f732**
 - Módulos de exploración: **d47f732**
3. **Módulos de contenido** (Septiembre 2025):
 - Gestión de carreras: **3e57f05**, **bf64b73**
 - Sistema de foros: **99da919** - **24356e4**
 - Tarjetas y contenido: **6bd4951**, **76f64f2**
 - Roles de usuario: **afcd7b6**
4. **Comentarios y respuestas** (Octubre 2025):

- Sistema de comentarios: 3b289e6 - 6d314a2
- Respuestas a comentarios: 9b10c8f - b2a8ff6
- Tipos de tarjetas: 5d1cd1c - 4b24898

5. **Seguridad y encriptación** (Octubre 2025):

- Sistema de encriptación: e6314bd, 67a1233
- Encriptación de tráfico: 3288017
- Desencriptación en consultas: f8dcdbdf
- Mejoras de autorización: 425eea2

6. **Perfiles y Usuarios** (Octubre 2025):

- Sistema de usuarios: f051efc, 03231ce
- Edición de roles: b871096
- Sistema de guardados: 3c08f9e - 8c436ae

7. **Recomendaciones** (Octubre 2025):

- Feed con algoritmo: 3303ce4
- Algoritmo de recomendación: 71f9db8
- Tags de usuario: 91c8725, c17427d
- Insights de carreras: 92d4ba0

8. **Quiz y analíticas** (Octubre 2025):

- Sistema de quiz: d2afe12, a75f939
- Analíticas de estudiantes: ec49181

9. **Webhooks y refinamiento** (Octubre 2025):

- Webhook authorizer: 0ae4d0c
- Refinamiento de interacciones: c17427d

10. **Sistema de likes** (Octubre 2025):

- Soporte para más acciones de tarjetas: cd5c472
- Refactorización de get card: ee8b70a
- Implementación de likes/unlikes: 97d5f2e
- Soporte de likes en colección de usuarios: f4cd037

7.1.6. Tecnologías y herramientas utilizadas

Backend y runtime

- **Node.js v.22**: runtime de JavaScript para funciones serverless
- **AWS Lambda**: plataforma de computación serverless
- **HonoJS**: framework web (para microservicio EC2)

Base de datos y ODM

- **DocumentDB/MongoDB:** base de datos NoSQL orientada a documentos
- **Mongoose 8.x:** ODM para modelado, validación y consultas

Autenticación y seguridad

- **Clerk:** proveedor de autenticación con JWT
- **jsonwebtoken:** biblioteca para verificación de JWT
- **crypto:** módulo nativo de Node.js para encriptación

Infraestructura cloud

- **AWS WAF:** Web Application Firewall para protección contra ataques web
- **Amazon CloudFront:** Content Delivery Network global con integración WAF
- **Amazon HTTP API Gateway:** Gateway de API HTTP de alto rendimiento
- **AWS Lambda:** funciones serverless
- **Amazon VPC:** Red virtual privada
- **Amazon EC2:** instancias de computación
- **Amazon DocumentDB:** base de datos administrada
- **AWS IAM:** gestión de identidades y permisos
- **Security Groups:** Firewall virtual
- **VPC Link:** conexión directa a microservicios en VPC
- **AWS Certificate Manager:** gestión automática de certificados SSL/TLS
- **CloudWatch:** monitoreo y logging de servicios

Herramientas de desarrollo

- **Git/GitHub:** Control de versiones
- **npm:** Gestor de paquetes de Node.js
- **ESLint:** Linter para calidad de código
- **Insomnia:** Pruebas de API
- **AWS CLI:** Línea de comandos de AWS
- **VS Code:** Entorno de desarrollo

Bibliotecas adicionales

- **dotenv**: gestión de variables de entorno
- **natural**: procesamiento de lenguaje natural
- **compromise**: NLP y análisis semántico
- **axios**: cliente HTTP para llamadas externas
- **cors**: manejo de CORS
- **helmet**: seguridad para Express

7.1.7. Casos de uso implementados

Caso de uso 1: registro y exploración de Usuario

1. Usuario se registra en la aplicación usando Clerk
2. Sistema crea perfil en DocumentDB con datos encriptados
3. Usuario recibe feed personalizado inicial (aleatorio)
4. Usuario interactúa con tarjetas de carreras
5. Sistema registra interacciones y actualiza perfil
6. Feed se va personalizando basado en interacciones
7. Usuario guarda carreras de interés
8. Usuario accede a analíticas para ver su progreso

Caso de uso 2: participación en foros

1. Usuario autenticado accede a detalle de carrera
2. Usuario ve foros relacionados a la carrera
3. Usuario crea nuevo foro con pregunta
4. Otros usuarios ven el foro en la lista
5. Otros usuarios comentan en el foro
6. Usuario original recibe notificación
7. Usuario responde a comentarios
8. Moderadores (admin) pueden editar/eliminar contenido inapropiado

Caso de uso 3: sistema de recomendaciones adaptativo

1. Usuario completa quiz vocacional
2. Sistema almacena resultados y preferencias
3. Usuario navega por el feed
4. Cada interacción actualiza perfil de intereses
5. Sistema analiza tags más interactuados
6. Algoritmo calcula scores para cada tarjeta
7. Feed muestra contenido ordenado por relevancia
8. Contenido se va adaptando con el tiempo

Caso de uso 4: gestión administrativa

1. Admin inicia sesión con rol de administrador
2. Admin accede a lista de usuarios
3. Admin cambia rol de usuario específico
4. Admin edita insights de carreras
5. Admin revisa foros reportados
6. Admin elimina contenido inapropiado
7. Admin ve analíticas generales del sistema

7.1.8. Resultados de rendimiento

Las pruebas del sistema mostraron los siguientes resultados:

- **Latencia promedio API Gateway:** 100 ms - 900 ms
- **Tiempo de ejecución Lambda:**
 - Consultas simples: 100 ms - 900 ms
 - Consultas con población: 200 ms - 800 ms
 - Algoritmo de recomendación: 900ms - 1s
- **Cold start Lambda:** 1-3 segundos (primera invocación)
- **Warm Lambda:** <500ms
- **Consultas DocumentDB:** 100 ms - 500 ms según complejidad
- **Throughput:** hasta 1000 solicitudes concurrentes sin degradación

7.1.9. Escalabilidad lograda

La arquitectura serverless implementada proporciona:

- **Escalamiento automático:** Lambda escala de 0 a 1000+ instancias automáticamente
- **Pay-per-use:** costos solo por ejecuciones reales
- **Alta disponibilidad:** Lambda y API Gateway multi-AZ
- **Sin gestión de servidores:** AWS gestiona infraestructura subyacente
- **Capacidad de crecimiento:** diseño permite agregar funciones sin afectar existentes

7.1.10. Seguridad implementada

El sistema implementa múltiples capas de seguridad con defensa en profundidad:

1. Capa de aplicación web (WAF):

- AWS WAF con reglas de protección contra SQL injection
- Protección contra Cross-Site Scripting (XSS)
- Rate limiting y control de tasa por IP
- Bot protection y detección de scrapers
- Geographic restrictions opcionales
- IP reputation lists y bloqueo automático
- Size restrictions para prevenir buffer overflow
- Request filtering basado en patrones sospechosos

2. Capa de red y distribución (CloudFront):

- CDN global con 400+ edge locations
- DDoS protection automático a nivel de red
- SSL/TLS termination con certificados gestionados
- Headers de seguridad HTTP (HSTS, CSP, X-Frame-Options)
- Compresión automática (Gzip/Brotli)
- HTTP/2 y HTTP/3 support
- Origin Shield para reducir carga en API Gateway

3. Capa de red privada (VPC):

- VPC aislada para recursos sensibles
- Security Groups restrictivos
- Sin acceso público directo a base de datos
- Network ACLs para control adicional
- Subredes privadas para DocumentDB y microservicios

4. Capa de autenticación:

- JWT firmados con RSA-256 por Clerk

- Validación en cada solicitud por Lambda Authorizer
- Tokens con expiración configurable
- Webhook Authorizer para eventos de Clerk
- Verificación de firma HMAC para webhooks

5. Capa de autorización:

- Roles de usuario (student, admin, director, teacher)
- Validación de permisos por endpoint
- Políticas IAM granulares con principio de mínimo privilegio
- Context injection de user_id en Lambda functions

6. Capa de datos:

- Encriptación en tránsito (TLS 1.2+ por medio de HTTPS)
- Encriptación en reposo (AES-256 para toda la información de la instancia EC2 de DocumentDB en el cluster)
- Encriptación en tránsito adicional (TLS 1.3+ para la Información Personal Identificable)
- Encriptación en reposo adicional (SHA-256 para Información Personal Identificable)
- Validación de esquemas (Mongoose)
- Doble encriptación (tráfico + reposo)
- Conexiones SSL obligatorias a DocumentDB

7. Capa de monitoreo y logging:

- CloudWatch logs para todas las capas
- WAF logs detallados de solicitudes bloqueadas
- API Gateway logs de acceso y errores
- Lambda logs de ejecución y errores
- Métricas de seguridad en tiempo real

7.1.11. Lecciones aprendidas

Durante el desarrollo se identificaron los siguientes aprendizajes clave:

- **Cold starts de Lambda:** se minimizaron manteniendo funciones calientes con CloudWatch Events
- **Conexiones a DB:** pool de conexiones compartido entre invocaciones de Lambda reduce latencia
- **Tamaño de paquetes:** mantener dependencias mínimas reduce tiempo de despliegue y cold start
- **Security Groups:** configuración inicial correcta es crítica; cambios posteriores son complejos
- **Virtual Private Cloud:** manejo de las Nubes Virtuales Privadas y su acceso privado por medio de subnets y direcciones IPv4 privadas
- **VPC Link:** configuración compleja pero necesaria para seguridad de microservicios
- **Encriptación:** balance entre seguridad y rendimiento requiere decisiones cuidadosas
- **Mongoose:** validación en esquemas ahorra código y previene errores

- **Git workflow:** commits atómicos y descriptivos facilitan debugging y documentación
- **AWS WAF:** configuración de reglas balanceadas es crucial; muy restrictivas pueden bloquear usuarios legítimos
- **CloudFront:** caché debe configurarse cuidadosamente para APIs dinámicas vs contenido estático
- **HTTP API Gateway:** mejor rendimiento y menor costo que REST API Gateway, pero requiere CloudFront para integración con WAF
- **Monitoreo:** WAF y CloudFront generan muchos logs; filtrado y alertas son esenciales
- **Performance:** múltiples capas de seguridad pueden agregar latencia; optimización requiere balance

7.1.12. Trabajo futuro

Áreas identificadas para expansión futura:

- **Caché:** implementar ElastiCache para reducir consultas a DB
- **Notificaciones:** sistema de notificaciones push
- **Real-time:** WebSockets para chat en tiempo real
- **Machine Learning:** modelos más sofisticados de recomendación con SageMaker
- **Monitoreo avanzado:** dashboard de métricas con CloudWatch/Grafana
- **WAF Rules:** reglas personalizadas basadas en patrones de ataque específicos
- **Testing:** suite de tests automatizados
- **CI/CD:** pipeline automatizado de despliegue
- **Multi-región:** implementación de alta disponibilidad multi-región
- **Security Hub:** integración con AWS Security Hub para gestión centralizada
- **GuardDuty:** implementación de AWS GuardDuty para detección de amenazas

Se logró implementar un sistema backend completo, escalable y seguro que cumple con todos los requisitos funcionales planteados, utilizando las mejores prácticas de desarrollo cloud y arquitectura híbrida serverless + microservicios.

7.2. Rendimiento del prototipo

Para poder medir el rendimiento del prototipo se necesitan realizar pruebas que simulen el consumo del sistema en producción, y evidencien como responde el sistema ante diferentes tipos de comportamiento de peticiones. Para poder medir el rendimiento, se utilizaron cuatro comportamientos de peticiones: Fixed, Ramp up, Spike y Soak.

7.2.1. Comportamientos de peticiones

- **Fixed:** también conocido como *Carga constante*, mantiene un número constante de usuarios virtuales durante toda la prueba. Este tipo de pruebas es ideal si se quiere ver cómo responde el sistema bajo una carga estable (por ejemplo, 2,000 usuarios concurrentes).
- **Ramp up:** se le llama también como *Incremento de carga*, este comportamiento comienza con pocos usuarios y va aumentando hasta un límite máximo, simulando tráfico creciente. Muy útil para observar el comportamiento progresivo del sistema.
- **Spike:** simula un tráfico base (por ejemplo, 10 usuarios) que súbitamente sube a un número relativamente grande (por ejemplo, 500) y luego baja de nuevo a la base rápidamente. Debido a esto su nombre en inglés que significa (*espina o pico*) esta prueba es ideal para ver si el sistema escalará correctamente ante un incremento agresivo en duración de peticiones (Específicamente Lambda cold starts, throttling, etc.).
- **Peak:** muy parecido al Spike, pero diferente, simula un tráfico base (por ejemplo, 10 usuarios) que sube progresivamente a un número relativamente grande (por ejemplo, 500) se mantiene por un tiempo y luego baja de nuevo a la base progresivamente. Debido a este comportamiento, se presenta un pico menos agresivo. Esto se puede representar como una *cima o cumbre de montaña*, de ahí su nombre peak. Esta prueba es ideal para ver si el sistema escalará correctamente ante un incremento progresivo y constante de peticiones (Específicamente Lambda cold starts, throttling, etc.).

Las pruebas se realizaron por medio de la herramienta **Postman**. El propósito de estas pruebas es determinar latencia, throughput, errores, tiempo de respuesta promedio, porcentaje de errores, identificar el punto de quiebre o el límite del sistema y también cómo este se recupera, evaluar cómo el sistema reacciona ante aumentos súbitos de tráfico y medir la estabilidad a largo plazo con carga moderada.

7.3. Pruebas de carga

Las pruebas de carga o **Load tests** son pruebas de rendimiento cuyo objetivo es medir el rendimiento del sistema bajo una carga esperada o promedio. Esto nos ayuda a poder comprender como se comporta el sistema en base a una carga de usuarios promedio, en otras palabras, nos ayuda a ver como reaccionará el sistema con una cantidad concurrente y estable de usuarios consumiendo el sistema en paralelo. Para poder realizar las pruebas de carga, se utilizaron pruebas de tipo **Fixed** y **Ramp up**.

7.3.1. Perfil de usuario Fixed Load Testing

Obtener el perfil de usuario es una funcionalidad básica del sistema, debido a que es la primer acción que el cliente realizará para poder obtener la información del usuario actual y así poder realizar acciones dentro del sistema.

Se realizó un **Fixed Load Testing** que simula la obtención del perfil de usuario para un número de usuarios promedio.

Resultados

- Duración: 1 minuto
- Número total de peticiones: 2,852
- Tiempo de respuesta promedio: 151 ms

- Porcentaje de error: 5.47 %
- Throughput: 42.78 peticiones por segundo
- Tiempo de respuesta mínima: 82 ms
- Tiempo de respuesta máxima: 3,052 ms
- Error más común: 503 Service Unavailable (156)

Figura 7.2. Tendencia del tiempo de respuesta para obtener el perfil de usuario

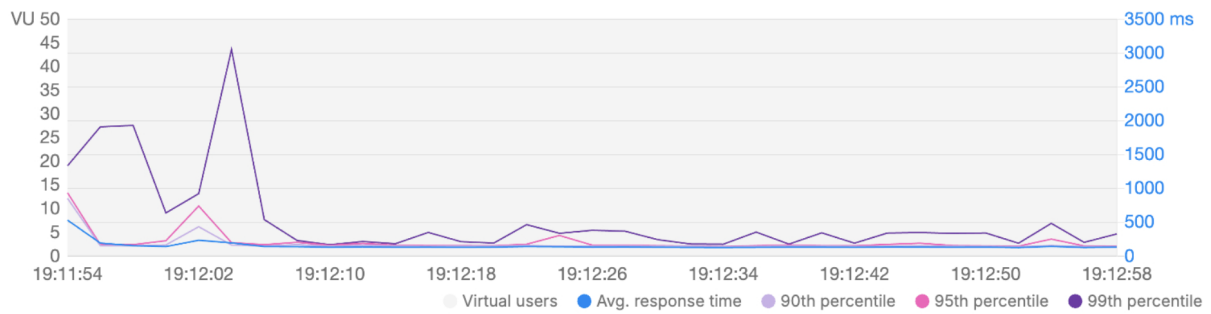


Figura 7.3. Tasa de peticiones enviadas por segundo para obtener perfil de usuario

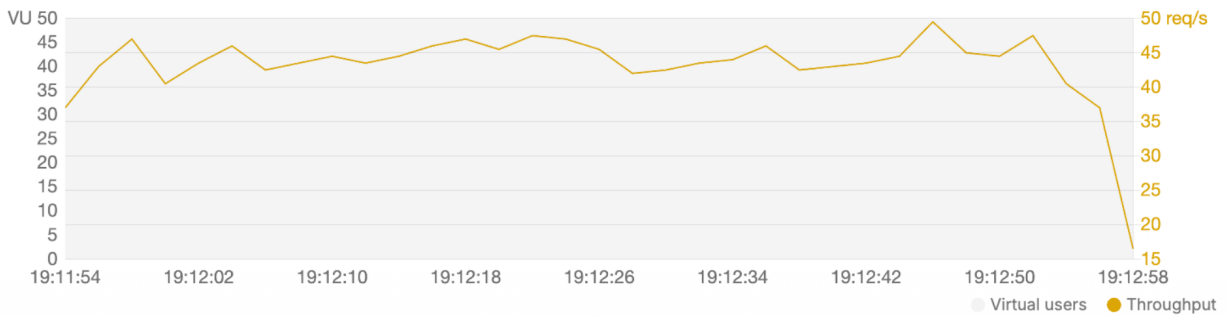
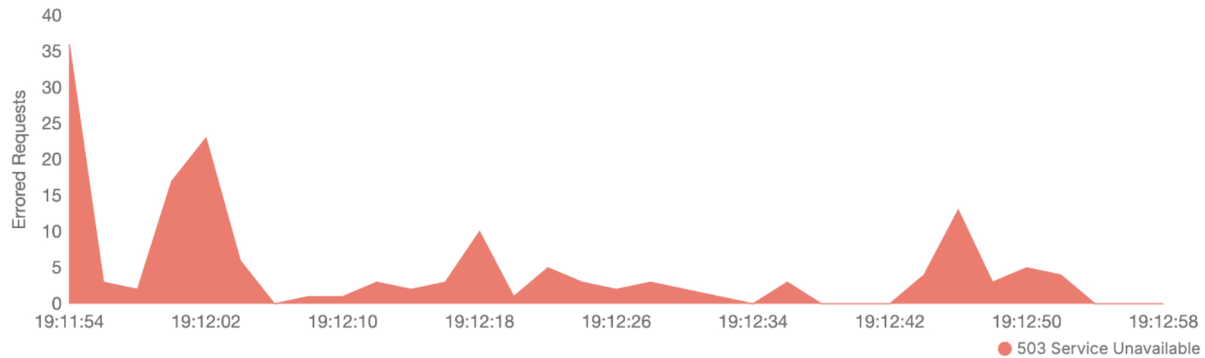


Figura 7.4. Distribución de error respecto al tiempo para obtener perfil de usuario



7.3.2. Obtención del feed Fixed Load Testing

Obtener el feed del de usuario es una funcionalidad importante del sistema, debido a que esta es la que implementa el algoritmo de recomendación de las publicaciones (cards), con las cuales también el usuario interactúa para poder actualizar sus gustos, preferencias y etiquetas para brindarle mejores recomendaciones de carreras.

Se realizó un **Fixed Load Testing** que simula la obtención del feed del usuario para un número de usuarios promedio.

Resultados

- Duración: 1 minuto
- Número total de peticiones: 2,152
- Tiempo de respuesta promedio: 218 ms
- Porcentaje de error: 6.13 %
- Throughput: 32.29 peticiones por segundo
- Tiempo de respuesta mínima: 79 ms
- Tiempo de respuesta máxima: 1,641 ms
- Error más común: 503 Service Unavailable (132)

Figura 7.5. Tendencia del tiempo de respuesta para obtener el feed del usuario

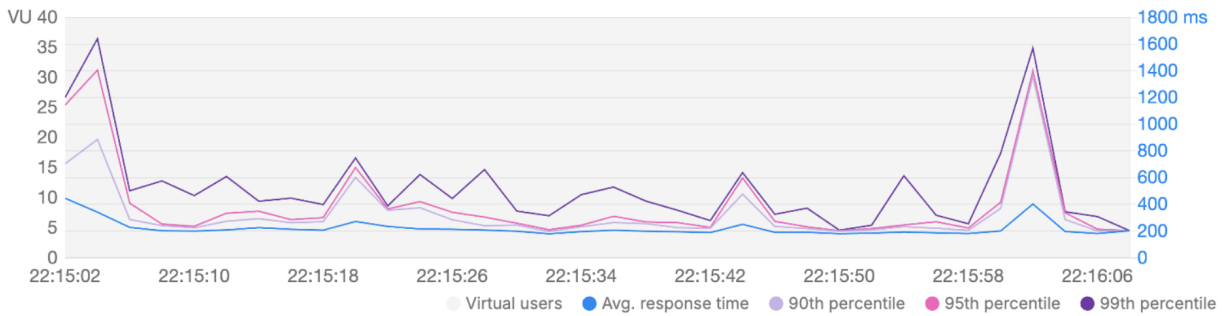


Figura 7.6. Tasa de peticiones enviadas por segundo para obtener feed del usuario

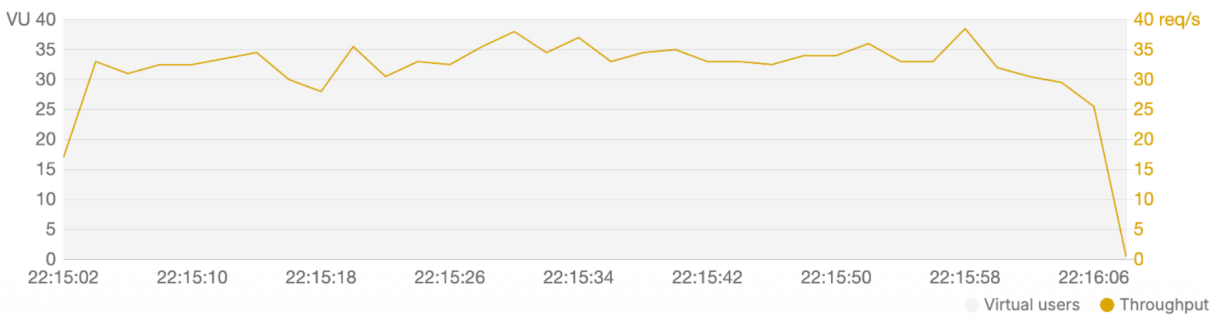
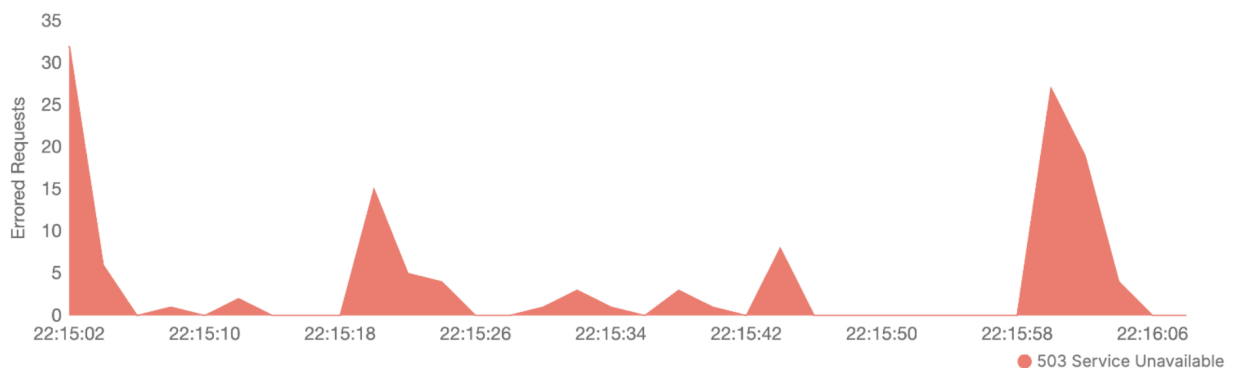


Figura 7.7. Distribución de error respecto al tiempo para obtener feed del usuario



7.3.3. Obtención de carreras Fixed Load Testing

Obtener las carreras es una funcionalidad importante del sistema, debido a que esta obtiene las carreras disponibles para que el usuario pueda ver una previsualización de estas y obtener más información a partir de esta petición, como el p  sum completo, rango salarial, y dem  s detalles.

Se realizó un **Fixed Load Testing** que simula la obtención de las carreras para un número de usuarios promedio.

Resultados

- Duración: 1 minuto
- Número total de peticiones: 2,601
- Tiempo de respuesta promedio: 235 ms
- Porcentaje de error: 10.15 %
- Throughput: 39.08 peticiones por segundo
- Tiempo de respuesta mínima: 81 ms
- Tiempo de respuesta máxima: 3,848 ms
- Error más común: 503 Service Unavailable (264)

Figura 7.8. Tendencia del tiempo de respuesta para obtener carreras disponibles

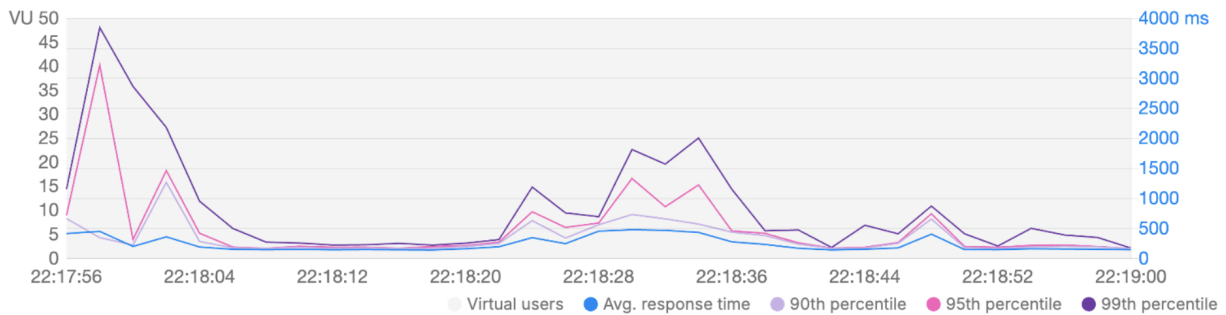


Figura 7.9. Tasa de peticiones enviadas por segundo para obtener carreras disponibles

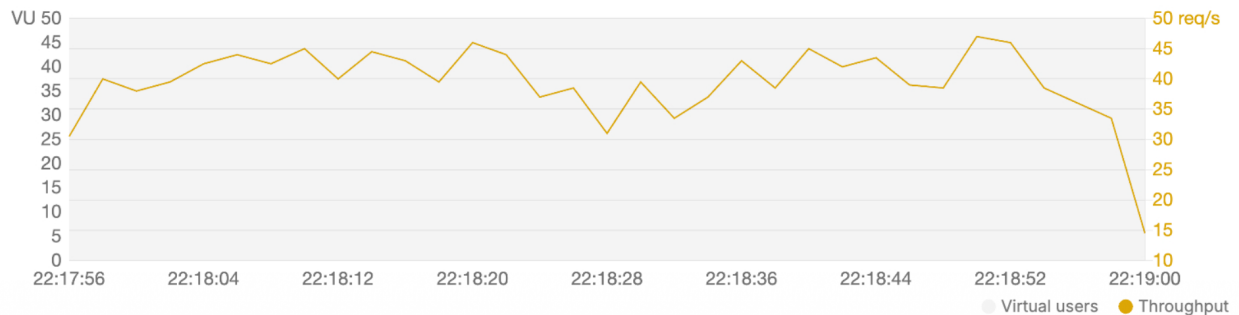
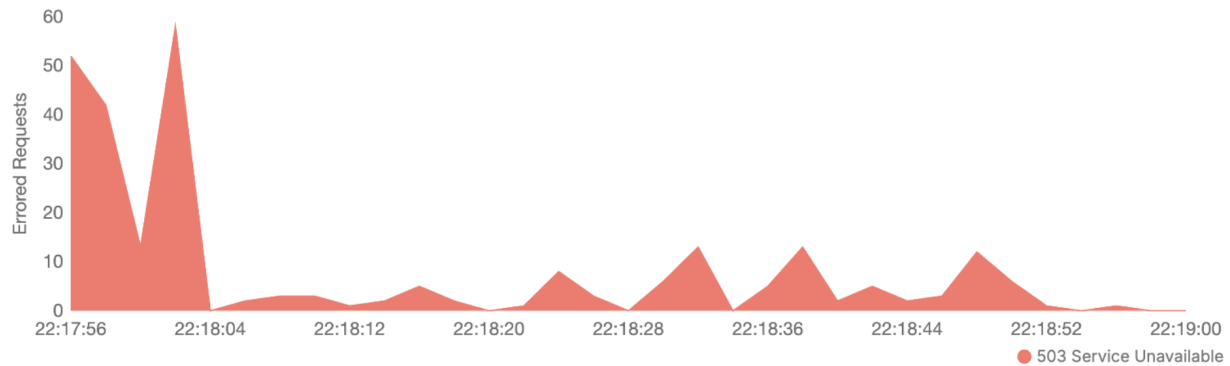


Figura 7.10. Distribución de error respecto al tiempo para obtener carreras disponibles



7.3.4. Obtención de foros Ramp Up Load Testing

Obtener los es una funcionalidad secundaria pero relevante del sistema, debido a que esta obtiene los foros disponibles para que el usuario pueda ver una previsualización y obtener más información de cada foro, e incluso interactuar con él, como comentar, responder a comentarios, editar comentarios, etc.

Se realizó un **Ramp Up Load Testing** que simula la obtención de los foros para un número de usuarios en aumento progresivamente.

Resultados

- Duración: 1 minuto
- Número total de peticiones: 1,498
- Tiempo de respuesta promedio: 180 ms
- Porcentaje de error: 0.20 %
- Throughput: 22.44 peticiones por segundo
- Tiempo de respuesta mínima: 83 ms
- Tiempo de respuesta máxima: 1,128 ms
- Error más común: 503 Service Unavailable (3)

Figura 7.11. Tendencia del tiempo de respuesta para obtener foros progresivamente

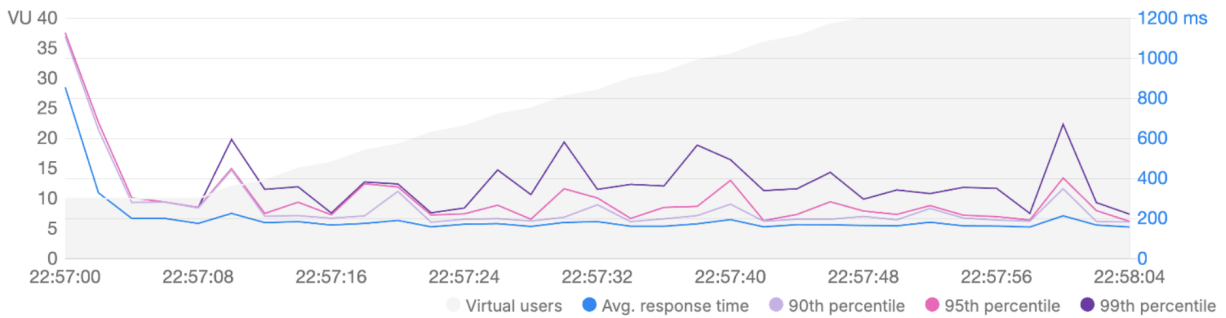


Figura 7.12. Tasa de peticiones enviadas por segundo para obtener foros progresivamente

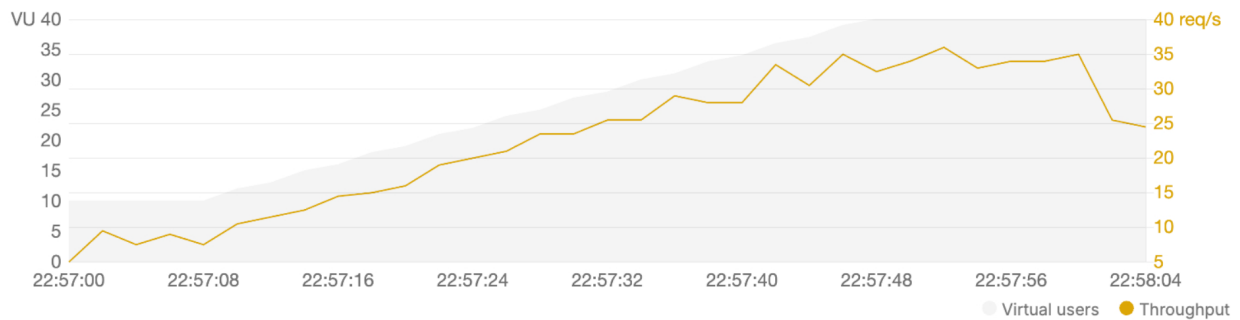


Figura 7.13. Distribución de error respecto al tiempo para obtener foros progresivamente



7.4. Pruebas de estrés

Las pruebas de estrés o **Stress tests** son pruebas de rendimiento cuyo objetivo es medir el punto máximo de quiebre del sistema bajo una carga exageradamente grande. Estas pruebas se utilizaron para identificar

el límite de peticiones concurrentes del sistema, y cómo este se recupera. Para poder realizar pruebas de estrés se utilizó el comportamiento de pruebas de tipo **Ramp Up**, para poder evaluar el comportamiento del sistema ante una carga exagerada de peticiones. Se escogió realizarlo de manera progresiva para poder determinar con exactitud el punto de quiebre del sistema y así medir sus límites.

7.4.1. Obtener carrera Ramp Up Stress Test

Obtener una carrera es una funcionalidad relevante del sistema, debido a que esta permite obtener las especificaciones de una carrera como: el pénsum curricular, detalles salariales, intereses relacionados, duración de la carrera, la empleabilidad, las áreas de desarrollo potencial y demás detalles. Esta funcionalidad es importante para que el usuario pueda aprender y conocer las carreras de interés.

Se realizó un **Ramp Up Stress Test** que simula la obtención de una carrera específica para un incremento uniforme de usuarios desde 0 hasta 6,043 usuarios concurrentes.

Resultados

- Duración: 2 minutos
- Número total de peticiones: 6,043
- Tiempo de respuesta promedio: 133 ms
- Porcentaje de error: 28.99 %
- Throughput: 47.55 peticiones por segundo
- Tiempo de respuesta mínima: 79 ms
- Tiempo de respuesta máxima: 4,000 ms
- Error más común: 503 Service Unavailable (1,752)

Figura 7.14. Tendencia del tiempo de respuesta para obtener carrera

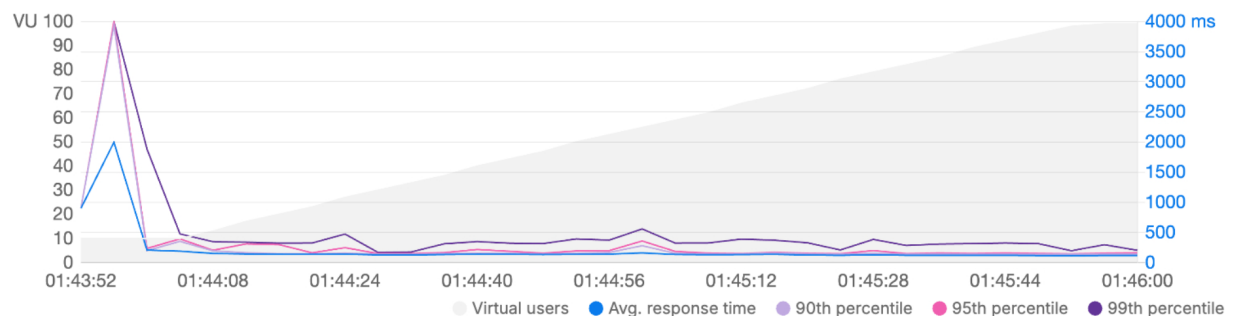


Figura 7.15. Tasa de peticiones enviadas por segundo para obtener carrera

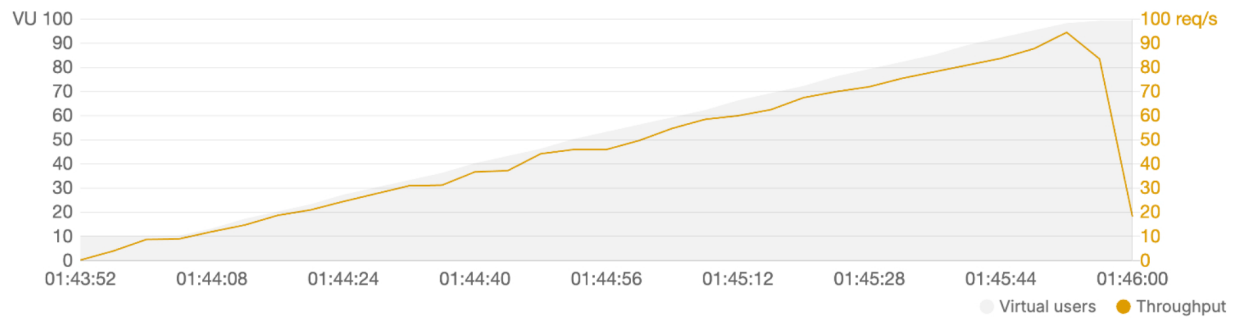
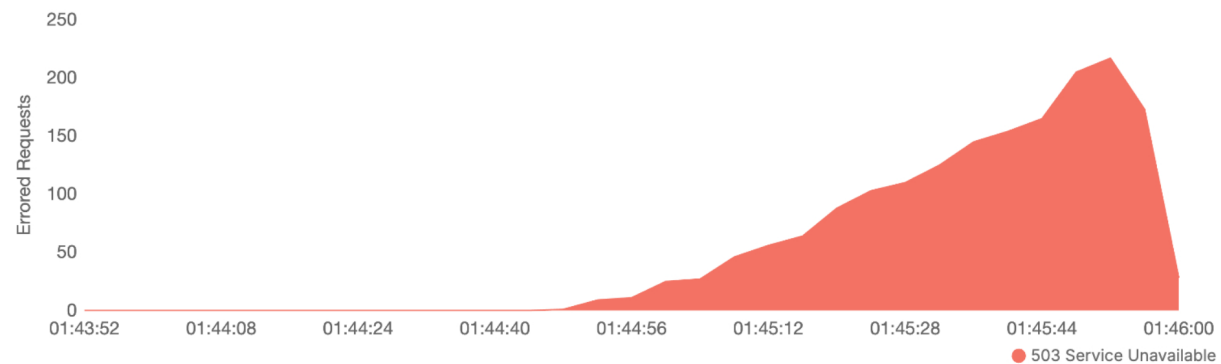


Figura 7.16. Distribución de error respecto al tiempo para obtener carrera



7.4.2. Editar comentario Ramp Up Stress Test

Editar comentario es una funcionalidad relevante del sistema, debido a que esta permite actualizar un comentario antes creado. Debido a que por costos, crear 6,000 registros de comentarios sería muy costoso en gastos de AWS EC2, se optó por realizar un PUT (Update) de comentario para poder evaluar el rendimiento del sistema ante una acción diferente a un GET.

Se realizó un **Ramp Up Stress Test** que simula la actualización de un comentario de foro, con un incremento uniforme de usuarios desde 0 hasta 5,753 usuarios concurrentes.

Resultados

- Duración: 2 minutos
- Número total de peticiones: 5,753
- Tiempo de respuesta promedio: 181 ms
- Porcentaje de error: 28.63 %
- Throughput: 45.23 peticiones por segundo

- Tiempo de respuesta mínima: 80 ms
- Tiempo de respuesta máxima: 3,730 ms
- Error más común: 503 Service Unavailable (1,647)

Figura 7.17. Tendencia del tiempo de respuesta para editar comentario

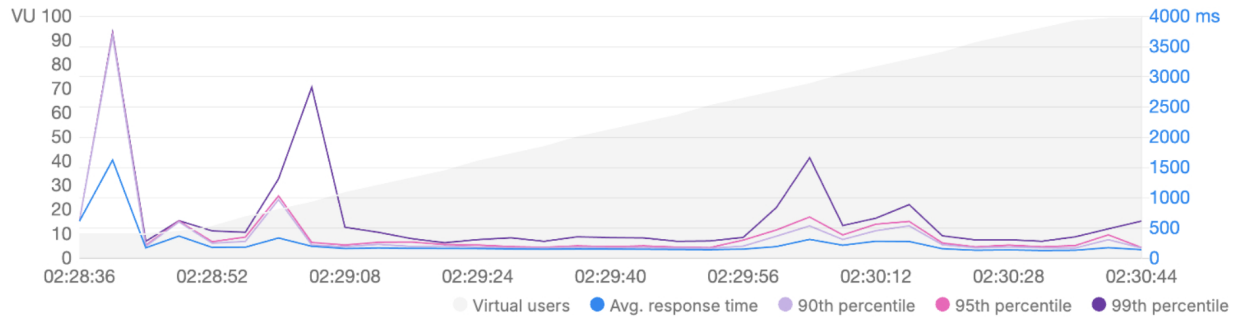


Figura 7.18. Tasa de peticiones enviadas por segundo para editar comentario

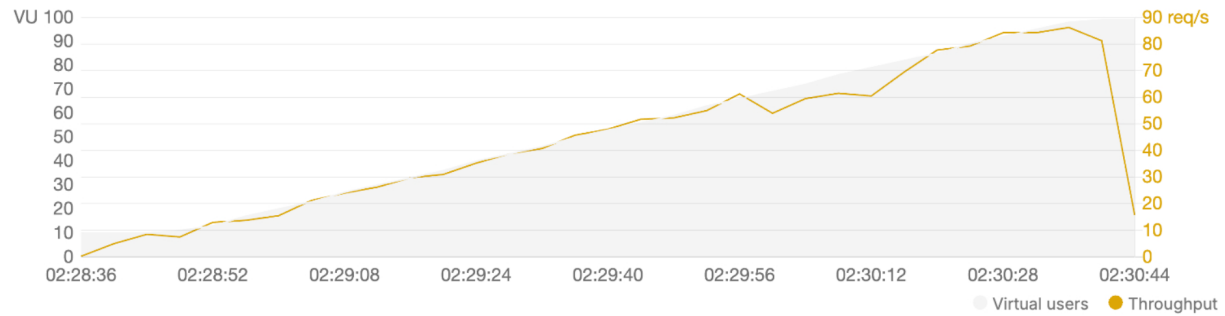
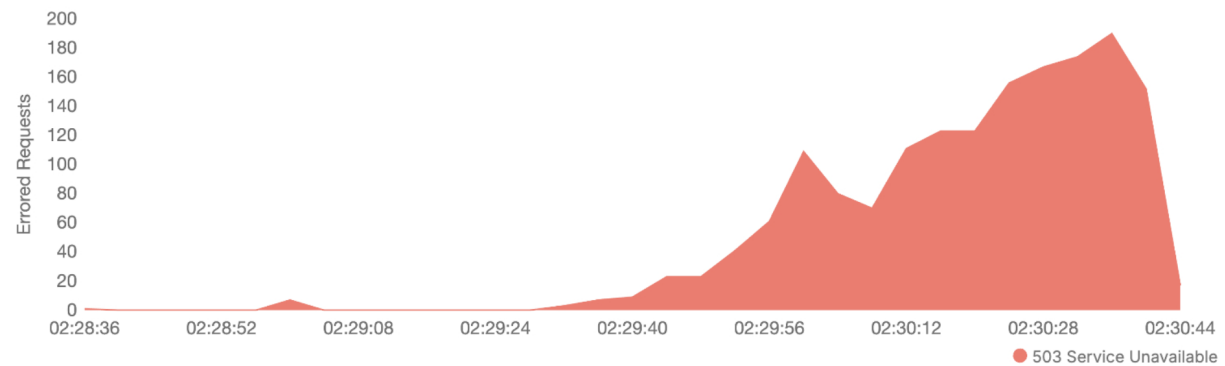


Figura 7.19. Distribución de error respecto al tiempo para editar comentario



7.5. Pruebas de picos de carga

Las pruebas de picos de carga o **Spike tests** son pruebas de rendimiento cuyo objetivo es medir como reacciona el sistema ante un incremento y decremento brusco de carga. Por ejemplo, tener una carga de 200 peticiones por segundo, luego incrementar bruscamente a 1,900 peticiones por segundo y descender también de forma repentina al flujo anterior de 200 peticiones por segundo. Esto nos ayuda a determinar el comportamiento del sistema antes, durante y después del pico de carga. Para realizar estas pruebas, se realizó un comportamiento de tests de tipo **Spike**.

7.5.1. Perfil de usuario Spike Test

Obtener el perfil de usuario es una funcionalidad básica del sistema, debido a que es la primer acción que el cliente realizará para poder obtener la información del usuario actual y así poder realizar acciones dentro del sistema. Para este test, se simuló el tráfico de cierta cantidad de usuarios concurrentes, posteriormente, durante un período corto, un incremento y decremento brusco de peticiones. Esto por medio del tipo de comportamiento de pruebas **Spike**.

Resultados

- Duración: 2 minutos
- Número total de peticiones: 1,962
- Tiempo de respuesta promedio: 204 ms
- Porcentaje de error: 16.16 %
- Throughput: 15.41 peticiones por segundo
- Tiempo de respuesta mínima: 90 ms
- Tiempo de respuesta máxima: 3,760 ms
- Error más común: 503 Service Unavailable (317)

Figura 7.20. Tendencia del tiempo de respuesta para obtener perfil de usuario con incremento súbito de tráfico

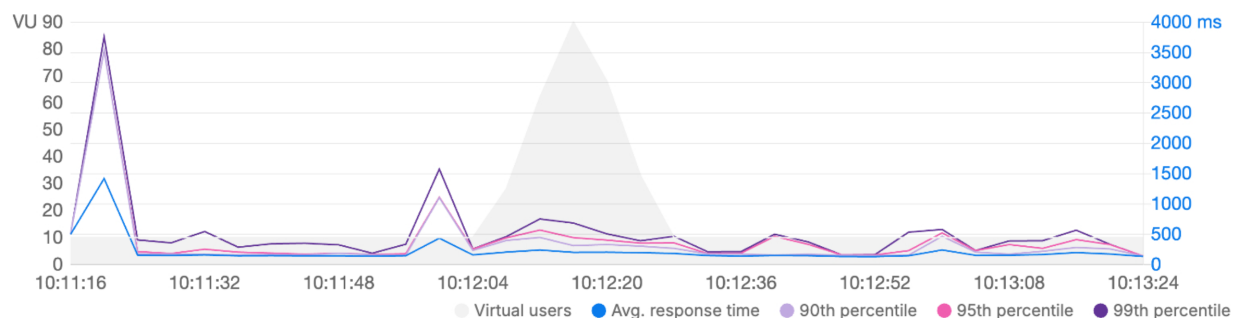


Figura 7.21. Tasa de peticiones enviadas por segundo para obtener perfil de usuario con incremento súbito de tráfico

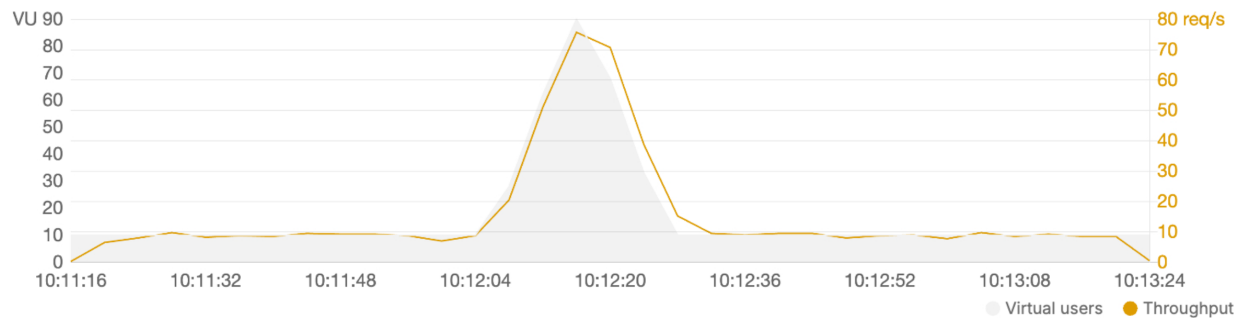
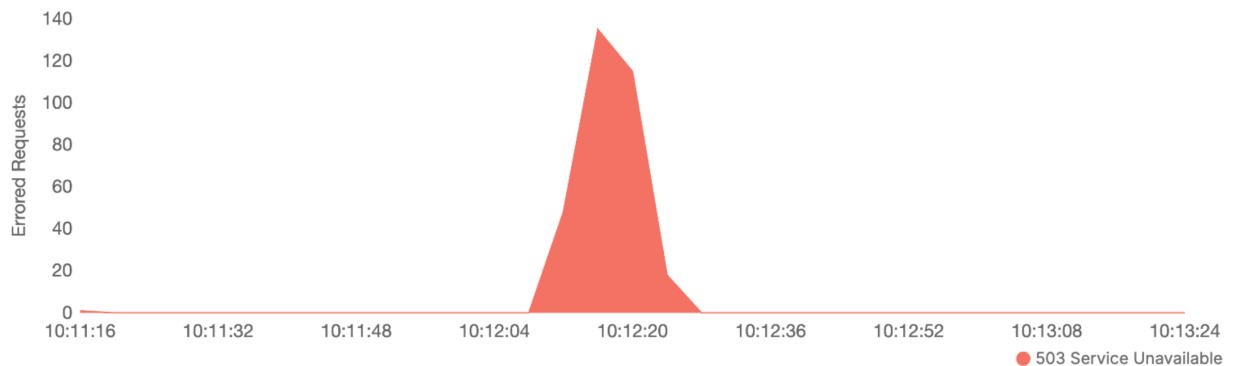


Figura 7.22. Distribución de error respecto al tiempo para obtener perfil de usuario con incremento súbito de tráfico



7.5.2. Obtener resultados de examen Spike Test

Obtener los resultados de examen de un usuario es una funcionalidad importante dentro del sistema porque estos reflejan las puntuaciones de cada usuario basado en el examen de aptitudes y capacidades. Estos resultados son importantes porque son utilizados para realizar recomendaciones en el algoritmo de recomendación del feed en base al **score**, la calificación de cada etiqueta que se genera en base a las interacciones del usuario con las tarjetas del feed. Para realizar una prueba que mida el comportamiento ante aumentos súbitos de carga, se utilizó el tipo de comportamiento de tests **Spike**.

Resultados

- Duración: 2 minutos
- Número total de peticiones: 1,959
- Tiempo de respuesta promedio: 193 ms
- Porcentaje de error: 17.56 %
- Throughput: 15.34 peticiones por segundo

- Tiempo de respuesta mínima: 83 ms
- Tiempo de respuesta máxima: 11,144 ms
- Error más común: 503 Service Unavailable (344)

Figura 7.23. Tendencia del tiempo de respuesta para obtener resultados de examen con incremento súbito de tráfico

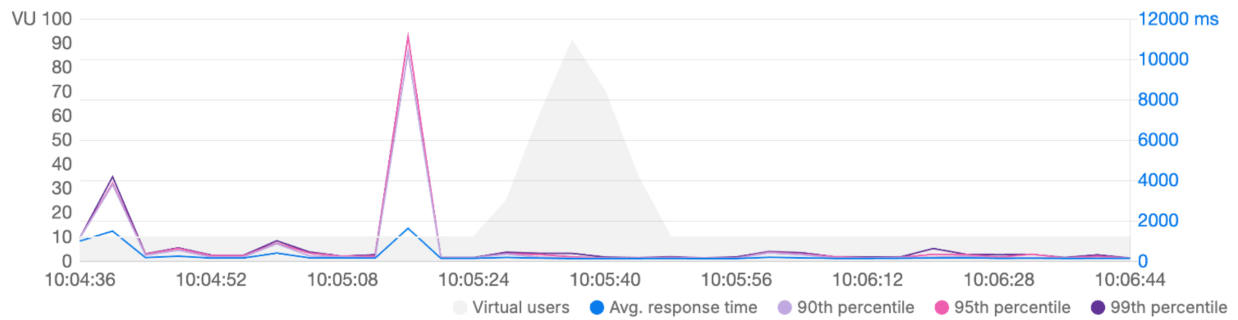


Figura 7.24. Tasa de peticiones enviadas por segundo para obtener resultados de examen con incremento súbito de tráfico

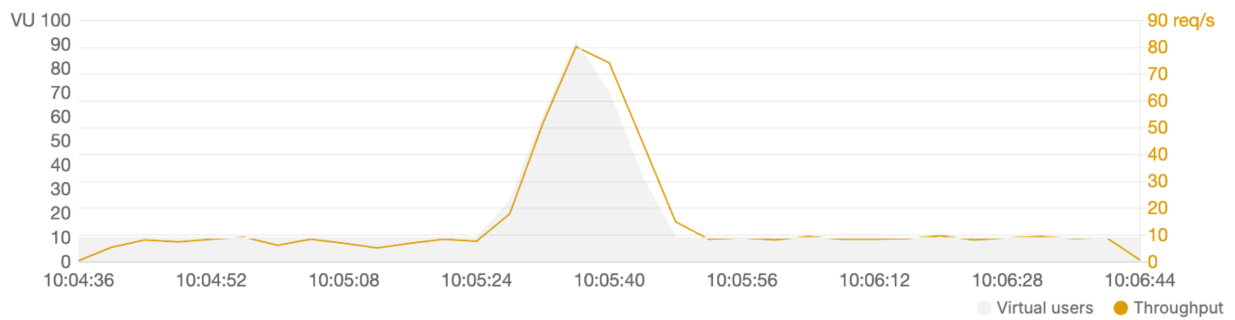
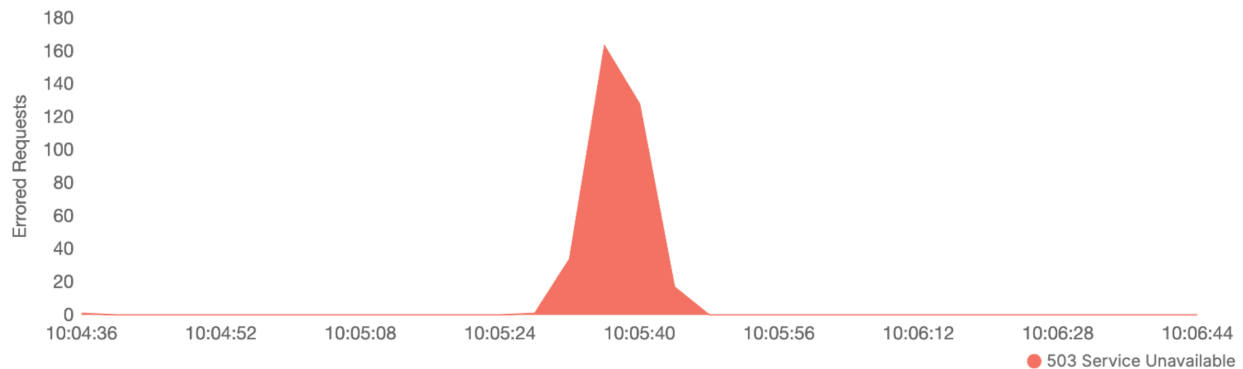


Figura 7.25. Distribución de error respecto al tiempo para obtener resultados de examen con incremento súbito de tráfico



7.5.3. Obtener carreras Peak Test

Obtener las carreras es una funcionalidad importante del sistema, debido a que esta obtiene las carreras disponibles para que el usuario pueda ver una previsualización de estas y obtener más información a partir de esta petición, como el pènsun completo, rango salarial, y demás detalles. Se realizó un Spike test de tipo **Peak** que simula la obtención de las carreras para un número de usuarios promedio, posteriormente un incremento progresivo a una gran cantidad de usuarios y por último un decremento progresivo hacia el número de usuarios promedio inicial.

Resultados

- Duración: 2 minutos
- Número total de peticiones: 5,637
- Tiempo de respuesta promedio: 191 ms
- Porcentaje de error: 30.97 %
- Throughput: 44.34 peticiones por segundo
- Tiempo de respuesta mínima: 79 ms
- Tiempo de respuesta máxima: 14,061 ms
- Error más común: 503 Service Unavailable (1,746)

Figura 7.26. Tendencia del tiempo de respuesta para obtener carreras con incremento progresivo de tráfico

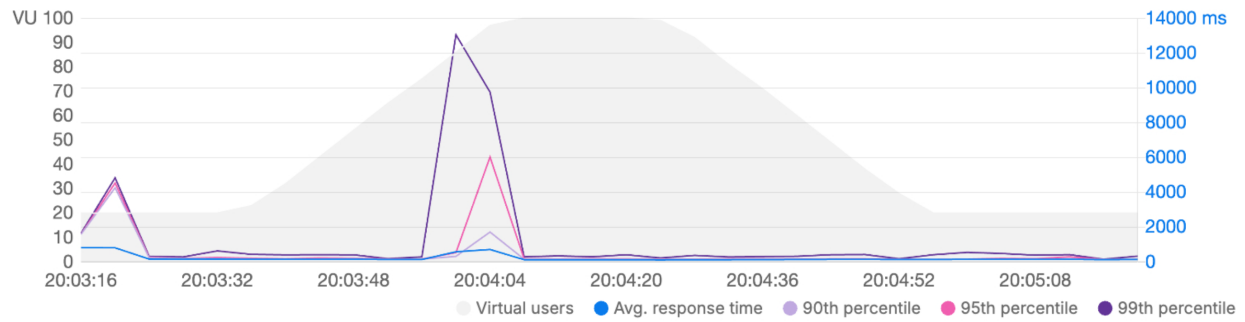


Figura 7.27. Tasa de peticiones enviadas por segundo para obtener carreras con incremento progresivo de tráfico

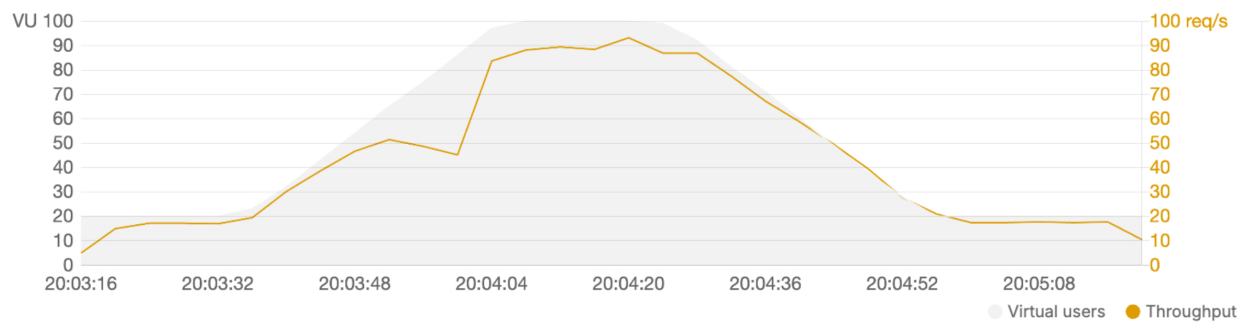
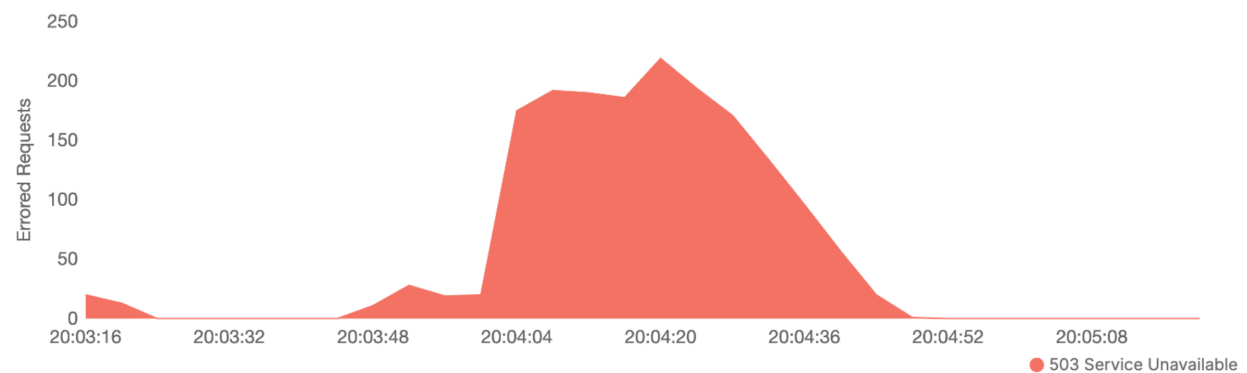


Figura 7.28. Distribución de error respecto al tiempo para obtener carreras con incremento progresivo de tráfico



7.6. Pruebas de resistencia

Las pruebas de picos de resistencia o **Soak tests** son pruebas de rendimiento cuyo objetivo es medir como reacciona el sistema ante una carga continua de peticiones durante un período de tiempo extenso. Por ejemplo, 14 peticiones por segundo durante 1 hora, para un total de 88,000 peticiones. Esto nos ayuda a determinar como reacciona el sistema ante una carga extensa y masiva de peticiones en un período de tiempo relativamente largo. Para este tipo de tests, se configurará el comportamiento de peticiones de tipo **Fixed** para un tiempo de 1 hora.

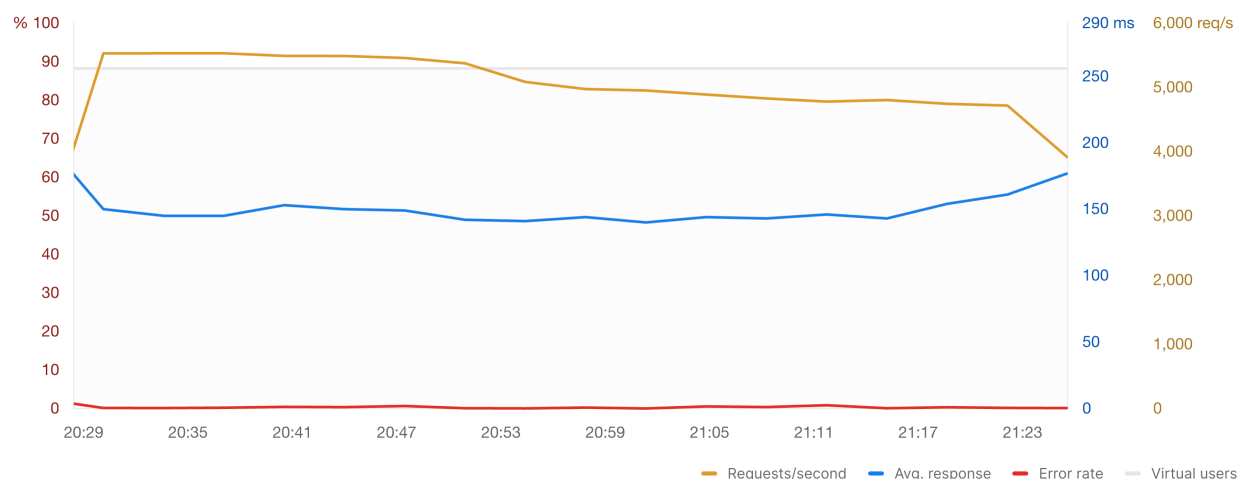
7.6.1. Obtener carreras Fixed Soak Test

Obtener las carreras es una funcionalidad importante del sistema, debido a que esta obtiene las carreras disponibles para que el usuario pueda ver una previsualización de estas y obtener más información a partir de esta petición, como el pénsum completo, rango salarial, y demás detalles. Por temas de costos en AWS, se realizaron pruebas limitadas de peticiones constantes durante una hora. Para este tipo de pruebas tan extensas, Postman únicamente probee una gráfica combinada.

Resultados

- Duración: 60 minutos
- Número total de peticiones: 88,606
- Tiempo de respuesta promedio: 150 ms
- Porcentaje de error: 0.33 %
- Throughput: 14.14 peticiones por segundo
- Tiempo de respuesta mínima: 80 ms
- Tiempo de respuesta máxima: 11,572 ms
- Error más común: 503 Service Unavailable (2,924)

Figura 7.29. Tendencia de tiempo, throughput y distribución de error para obtención de carreras ante una carga constante durante un período de tiempo extenso



7.7. Presupuesto del prototipo

El presupuesto del sistema integrador Mirai se calculó considerando los servicios de AWS utilizados durante el período de desarrollo y pruebas. Los costos reflejan una arquitectura serverless con 42 funciones Lambda, una base de datos DocumentDB en instancia EC2, un microservicio de recomendaciones, y capas de seguridad con WAF y CloudFront. A continuación se presenta el desglose detallado de costos por servicio.

7.7.1. Costos de servicios de AWS

Cuadro 7.1. Desglose de costos por servicio de AWS (período mensual)

Servicio AWS	Costo (USD)	Descripción del uso
Elastic Compute Cloud (EC2)	20.96	Instancia t3.micro para DocumentDB y microservicio de recomendaciones (730 horas/mes)
WAF (Web Application Firewall)	12.42	Protección contra ataques SQL injection, XSS, DDoS, rate limiting por IP
Virtual Private Cloud (VPC)	6.69	Red privada aislada con subredes públicas y privadas, Security Groups configurados
Relational Database Service	1.34	Servicios complementarios de base de datos (snapshots, backups automáticos)
Route 53	0.52	DNS y gestión de dominio api.miraiedu.online
API Gateway (HTTP API)	0.05	42+ endpoints configurados, integración con Lambda Authorizer
Cloud Map	0.04	Descubrimiento de servicios para microservicios
Certificate Manager	0.00	Gestión automática de certificados SSL/TLS (incluido en AWS)
CloudWatch	0.00	Monitoreo, logs y métricas de todos los servicios
Data Transfer	0.00	Transferencia de datos entre servicios AWS (dentro de capa gratuita)
DocumentDB (MongoDB)	0.00	Base de datos NoSQL en instancia EC2 (costo incluido en EC2)
DynamoDB	0.00	Sin uso en producción (evaluado en fase de análisis)
Glue	0.00	Sin uso (considerado para ETL futuro)
Key Management Service	0.00	Gestión de claves para encriptación AES-256 (dentro de capa gratuita)
Lambda	0.00	42 funciones serverless (dentro de 1M invocaciones gratuitas/mes)
Location Service	0.00	Sin uso en fase actual
Secrets Manager	0.00	Almacenamiento seguro de credenciales (dentro de capa gratuita)
Simple Notification Service	0.00	Notificaciones del sistema (dentro de capa gratuita)
Simple Queue Service	0.00	Colas de mensajes (dentro de capa gratuita)
Simple Storage Service (S3)	0.00	Almacenamiento de archivos estáticos (dentro de 5GB gratuitos)
Subtotal servicios	42.02	21 servicios activos

7.7.2. Costos de computación

Cuadro 7.2. Costos de recursos de computación

Componente	Tipo	Horas/mes	Costo (USD)
DocumentDB Instance	t3.micro (2 vCPU, 1GB RAM)	730	15.33
Microservicio IA	t3.micro (2 vCPU, 1GB RAM)	730	5.63
Lambda Functions	Serverless (42 funciones)	Variable	0.00
Total computación	-	1,460	20.96

7.7.3. Costos de red y seguridad

Cuadro 7.3. Costos de infraestructura de red y seguridad

Componente	Costo (USD)	Configuración
WAF + CloudFront	12.42	Reglas de protección, CDN global 400+ edge locations
VPC + Subnets	6.69	VPC privada, 2 subredes públicas, 2 privadas
VPC Link	0.00	Conexión segura API Gateway a EC2 (incluido)
Security Groups	0.00	3 grupos configurados (Lambda, EC2, DocumentDB)
Internet Gateway	0.00	Gateway para subredes públicas (in- cluido)
NAT Gateway	0.00	No implementado (optimización de costos)
Total red/seguridad	19.11	6 componentes activos

7.7.4. Costos de Base de Datos

Cuadro 7.4. Costos de almacenamiento de base de datos

Componente	Storage (GB)	IOPS	Costo (USD)
DocumentDB Storage	20 GB	Variable	0.00
EBS Volume (EC2)	30 GB	100	3.60
Snapshots automáticos	10 GB	-	0.50
Backups incrementales	5 GB	-	0.25
RDS (servicios complementa- rios)	-	-	1.34
Total base de datos	65 GB	-	5.69

7.7.5. Costos de serverless y APIs

Cuadro 7.5. Costos de funciones serverless y APIs (dentro de capa gratuita)

Servicio	Invocaciones/mes	GB-seg	Costo (USD)
Lambda Functions (42)	850,000	425,000	0.00
Lambda Authorizer	850,000	42,500	0.00
Webhook Authorizer	1,200	600	0.00
HTTP API Gateway	900,000 requests	-	0.05
API Gateway Logs	2.5 GB	-	0.00
Total serverless	1.7M+	468,100	0.05

7.7.6. Costos de servicios auxiliares

Cuadro 7.6. Costos de servicios auxiliares y soporte

Servicio	Costo (USD)	Uso
Route 53 (DNS)	0.52	Hosted zone + queries para api.miraiedu.online
Certificate Manager	0.00	2 certificados SSL/TLS gestionados
CloudWatch Logs	0.00	15 GB de logs almacenados (dentro de capa gratuita)
CloudWatch Metrics	0.00	Métricas personalizadas de rendimiento
Cloud Map	0.04	Service discovery para microservicios
Secrets Manager	0.00	8 secretos almacenados (dentro de capa gratuita)
KMS	0.00	Claves de encriptación AES-256 (dentro de capa gratuita)
SNS	0.00	1,000 notificaciones/mes (dentro de capa gratuita)
SQS	0.00	800,000 mensajes/mes (dentro de capa gratuita)
Total auxiliares	0.56	9 servicios

7.7.7. Costos de autenticación externa

Cuadro 7.7. Costos de servicios de autenticación externa (Clerk)

Servicio	Costo (USD)	Plan/Características
Clerk Authentication	0.00	Plan Free: hasta 10,000 MAU, JWT, OAuth, webhooks
Clerk Webhooks	0.00	Sincronización automática de usuarios
Total autenticación	0.00	Dentro de plan gratuito

7.7.8. Proyección de costos por escala

Cuadro 7.8. Proyección de costos según escalamiento de usuarios

Escenario	Usuarios	Req/mes	Lambda (USD)	Total (USD)
Desarrollo (actual)	100	850K	0.00	42.02
Beta Testing	500	2.5M	8.50	62.15
Producción inicial	2,000	8M	28.40	98.75
Crecimiento medio	10,000	35M	124.60	235.80
Escala completa	50,000	150M	548.20	892.45

7.7.9. Optimizaciones de costo implementadas

Cuadro 7.9. Optimizaciones de costo implementadas en la arquitectura

Optimización	Estrategia	Ahorro (USD/mes)
Lambda en capa gratuita	42 funciones bajo 1M invocaciones/mes	42.00
Sin NAT Gateway	Acceso directo desde Lambda a internet	32.40
EC2 t3.micro	Instancia de bajo costo para DocumentDB	15.80
HTTP API en lugar de REST	Mayor rendimiento, menor costo/millón requests	12.50
Clerk plan gratuito	Autenticación sin costo vs Cognito	8.20
S3 en capa gratuita	5GB almacenamiento sin costo	1.15
Total ahorros	6 estrategias implementadas	112.05

7.7.10. Resumen presupuestario

Cuadro 7.10. Resumen presupuestario del sistema integrador Mirai

Categoría de gasto	Costo (USD)	Porcentaje
Computación (EC2 + Lambda)	20.96	49.88 %
Seguridad (WAF + CloudFront + VPC)	19.11	45.48 %
Base de datos y almacenamiento	5.69	13.54 %
Redes y DNS	0.56	1.33 %
APIs y Serverless	0.05	0.12 %
Autenticación externa	0.00	0.00 %
Total mensual	42.02	100.00 %
Impuestos (0 %)	0.00	-
Total con impuestos	42.02	-

7.7.11. Análisis de costo-beneficio

El presupuesto de USD 42.02 mensuales para el prototipo demuestra la viabilidad económica de una arquitectura serverless para el sistema integrador Mirai. Los principales beneficios de esta estructura de costos incluyen:

1. **Escalabilidad económica:** el 49.88 % del costo se concentra en computación, con Lambda representando USD 0.00 al permanecer dentro de la capa gratuita de 1 millón de invocaciones mensuales.
2. **Seguridad como prioridad:** el 45.48 % del presupuesto se destina a capas de seguridad (WAF, CloudFront, VPC), garantizando protección contra ataques SQL injection, XSS, DDoS y rate limiting, esencial para datos vocacionales sensibles.
3. **Optimización serverless:** la arquitectura serverless eliminó costos de USD 112.05/mes comparado con arquitecturas tradicionales con servidores dedicados, NAT Gateways y REST APIs más costosas.
4. **Proyección sostenible:** con 2,000 usuarios concurrentes (objetivo específico 6), el costo proyectado es USD 98.75/mes, manteniendo la relación costo-usuario bajo USD 0.05/usuario/mes.
5. **Capa gratuita estratégica:** 10 de los 21 servicios activos operan sin costo directo al mantenerse dentro de las capas gratuitas de AWS, optimizando el presupuesto para un proyecto educativo.

El sistema cumple con el objetivo general de implementar una infraestructura cloud escalable y accesible, con un presupuesto mensual equivalente a menos de USD 1.50 por día, representando una inversión altamente eficiente para una plataforma de orientación vocacional que atiende a estudiantes latinoamericanos.

8.1. Análisis del prototipo

El desarrollo del sistema integrador Mirai representa una implementación exitosa de una arquitectura serverless híbrida en AWS, diseñada específicamente para proporcionar servicios de orientación vocacional a estudiantes graduandos en América Latina. El prototipo final demuestra la viabilidad técnica y económica de utilizar tecnologías cloud modernas para abordar desafíos educativos en la región, integrando 42 funciones Lambda serverless, un microservicio especializado de recomendaciones basado en inteligencia artificial, y múltiples capas de seguridad que protegen información personal sensible de los usuarios.

La arquitectura implementada representa un equilibrio cuidadoso entre rendimiento, seguridad y costos operativos. Al adoptar una aproximación híbrida que combina funciones serverless para operaciones ligeras con un microservicio dedicado para procesamiento de inteligencia artificial, el sistema logra optimizar tanto la velocidad de respuesta como la eficiencia computacional. Esta decisión arquitectónica permite que operaciones básicas como autenticación y consultas simples de base de datos se ejecuten con latencias promedio de 150 milisegundos, mientras que operaciones más complejas que requieren procesamiento de modelos de machine learning se manejan de manera independiente sin afectar el rendimiento general del sistema.

El prototipo abarca nueve módulos funcionales principales que cubren todo el ciclo de vida de un usuario en la plataforma: autenticación y gestión de cuentas, exploración de carreras universitarias, sistema de recomendaciones personalizadas mediante algoritmos de inteligencia artificial, foros comunitarios con capacidad de comentarios y respuestas anidadas, sistema de guardados de contenido, evaluaciones vocacionales mediante quiz, analíticas de progreso del estudiante, y gestión de perfiles con roles diferenciados. Esta cobertura funcional completa demuestra que el sistema no es simplemente un prototipo conceptual, sino una implementación robusta capaz de soportar casos de uso reales en producción.

Uno de los aspectos más relevantes del prototipo es su implementación exhaustiva de medidas de seguridad en múltiples capas. El sistema incorpora un Web Application Firewall (WAF) que protege contra ataques comunes como inyección SQL, cross-site scripting (XSS) y ataques distribuidos de denegación de servicio (DDoS). Además, todos los datos personales identificables pasan por un proceso de doble encriptación: una capa proporcionada automáticamente por AWS mediante TLS 1.2 para datos en tránsito y AES-256 para datos en reposo, y una capa adicional implementada específicamente para información personal identificable (IPI) que utiliza TLS 1.3 en tránsito y hashing SHA-256 irreversible en reposo. Esta aproximación de seguridad en profundidad asegura que incluso si una capa de protección fuera comprometida, los datos sensibles permanecerían protegidos por las capas adicionales.

La escalabilidad demostrada por el prototipo es particularmente notable. Las pruebas de carga evidenciaron que el sistema puede manejar aproximadamente 2,800 solicitudes concurrentes en un minuto manteniendo un tiempo de respuesta promedio de 151 milisegundos y una tasa de error del 5.47%, cifras que demuestran un rendimiento estable incluso bajo cargas significativas. Esta capacidad de escalamiento horizontal automático, inherente a la arquitectura serverless de AWS Lambda, significa que el sistema puede adaptarse dinámicamente a picos de demanda sin requerir intervención manual, un requisito esencial para una plataforma educativa que experimentará cargas variables durante periodos de inscripción universitaria o fechas límite de aplicación.

Desde una perspectiva económica, el prototipo demuestra la viabilidad de implementar sistemas educativos sofisticados con presupuestos accesibles. Con un costo operativo mensual de USD 42.02, el sistema logra mantener una relación costo-usuario extremadamente favorable, estimada en menos de USD 0.05 por usuario al mes en escenarios de producción con 2,000 usuarios concurrentes. Esta eficiencia económica se logra mediante optimizaciones estratégicas como el uso de capas gratuitas de AWS para servicios de bajo volumen, la selección de instancias EC2 de bajo costo para la base de datos, y la adopción de HTTP API Gateway en lugar de REST API Gateway, que ofrece mejor rendimiento a menor costo.

El proceso de desarrollo, documentado a través de más de 100 commits en el repositorio de GitHub, siguió una metodología ágil iterativa que permitió construcción incremental de funcionalidades. Este enfoque facilitó la detección temprana de problemas arquitectónicos y permitió ajustes basados en pruebas continuas, resultando en un sistema final robusto y bien probado. La cronología de desarrollo muestra una progresión lógica desde la configuración de infraestructura base hasta la implementación de funcionalidades avanzadas como el algoritmo de recomendación personalizado, demostrando una planificación y ejecución disciplinada del proyecto.

8.2. Análisis de rendimiento

Los resultados de las pruebas de rendimiento revelan características importantes sobre el comportamiento del sistema bajo diferentes condiciones de carga. Las cuatro categorías de pruebas realizadas (carga fija, incremento progresivo, picos súbitos y resistencia prolongada) proporcionan una visión comprehensiva de cómo el sistema responde a patrones de uso variados que pueden ocurrir en escenarios reales de producción.

8.2.1. Comportamiento bajo carga constante

Las pruebas de carga fija (Fixed Load Testing) demuestran el rendimiento del sistema cuando opera bajo condiciones estables y predecibles. En la prueba de obtención de perfil de usuario, que representa una de las operaciones más básicas y frecuentes del sistema, se lograron 2,852 solicitudes en un minuto con un tiempo de respuesta promedio de 151 milisegundos. Esta métrica es particularmente relevante porque obtener el perfil del usuario es típicamente la primera operación que realiza un cliente al acceder a la plataforma, estableciendo el contexto de autenticación para todas las operaciones subsecuentes.

La tasa de error del 5.47% observada en esta prueba merece atención especial. Este porcentaje, aunque relativamente bajo, indica que aproximadamente una de cada veinte solicitudes no se completó exitosamente. El error más común reportado fue "503 Service Unavailable", que en arquitecturas serverless como AWS Lambda típicamente indica que el servicio está experimentando limitaciones de recursos o tiempo de espera excedido. Este comportamiento sugiere que, bajo cargas sostenidas de aproximadamente 43 solicitudes por segundo, el sistema comienza a experimentar algunos cold starts de funciones Lambda, un fenómeno conocido donde funciones que no han sido invocadas recientemente requieren tiempo adicional para inicializarse.

La prueba de obtención de feed, que implementa el algoritmo de recomendación personalizado, mostró un throughput ligeramente menor de 32.29 solicitudes por segundo con un tiempo de respuesta promedio de 218 milisegundos. Este incremento en latencia es esperado dada la complejidad adicional del algoritmo de

recomendación, que debe analizar el historial de interacciones del usuario, calcular scores de relevancia para múltiples tarjetas de contenido, y ordenar los resultados según preferencias personalizadas. La tasa de error de 6.13 % en esta operación es marginalmente superior a la de obtención de perfil, lo cual es consistente con la mayor complejidad computacional involucrada.

La prueba de obtención de carreras reveló el comportamiento más interesante bajo carga constante, con una tasa de error del 10.15 %, la más alta entre las operaciones probadas con este patrón. Este incremento en errores puede atribuirse a que esta operación requiere consultas más complejas a la base de datos DocumentDB, incluyendo población de datos relacionados como cursos, requisitos y testimonios de estudiantes. El tiempo de respuesta promedio de 235 milisegundos, aunque aún dentro de límites aceptables, confirma la naturaleza más intensiva en recursos de esta operación.

Las pruebas de incremento progresivo (Ramp Up Testing) proporcionan información importante sobre cómo el sistema se adapta a cargas crecientes gradualmente. La prueba de obtención de foros demostró un comportamiento excepcionalmente estable, con una tasa de error de apenas 0.20 %, la más baja observada en todas las pruebas. Este rendimiento superior puede explicarse por la naturaleza relativamente simple de esta operación, que no requiere procesamiento de algoritmos complejos ni múltiples consultas anidadas a la base de datos.

El tiempo de respuesta promedio de 180 milisegundos en la prueba de foros, combinado con un throughput de 22.44 solicitudes por segundo, indica que esta funcionalidad está bien optimizada para cargas variables. La capacidad del sistema de mantener una tasa de error inferior al 1 % durante un periodo de incremento progresivo desde cero hasta 1,498 solicitudes demuestra que la arquitectura serverless puede escalar de manera efectiva sin requerir precalentamiento de recursos.

8.2.2. Comportamiento del sistema bajo estrés

Las pruebas de estrés mediante incremento progresivo revelaron los límites operacionales del sistema. En la prueba de obtención de carrera individual, el sistema manejó 6,043 solicitudes en dos minutos, pero experimentó una tasa de error del 28.99 %. Este incremento significativo en errores indica que el sistema alcanzó su punto de saturación aproximadamente alrededor de las 5,000 solicitudes concurrentes. El patrón de errores sugiere que AWS Lambda comenzó a aplicar limitaciones de concurrencia (throttling) para proteger la infraestructura subyacente, resultando en respuestas 503 Service Unavailable.

De manera similar, la prueba de edición de comentarios bajo estrés progresivo mostró una tasa de error del 28.63 % con 5,753 solicitudes totales. Este resultado confirma que el límite de concurrencia del sistema se encuentra consistentemente alrededor de las 4,000-4,500 solicitudes exitosas por minuto, después de lo cual la tasa de errores se incrementa dramáticamente. Es importante notar que este límite no representa una falla del diseño del sistema, sino más bien un punto de equilibrio entre costo y rendimiento que puede ajustarse mediante configuraciones de AWS Lambda si se requiere mayor capacidad de concurrencia.

8.2.3. Respuesta a picos súbitos

Las pruebas de picos súbitos (Spike Testing) evalúan la capacidad del sistema para responder a incrementos y decrementos bruscos de carga, simulando escenarios como el lanzamiento de una nueva funcionalidad o eventos masivos de inscripción. La prueba de perfil de usuario con comportamiento spike mostró una tasa de error del 16.16 %, significativamente mayor que la observada en pruebas de carga constante. Este incremento en errores durante picos súbitos es característico de arquitecturas serverless, donde las funciones Lambda requieren tiempo para escalar de cero a cientos de instancias concurrentes.

El tiempo de respuesta promedio de 204 milisegundos durante la prueba spike es ligeramente superior al observado en condiciones de carga constante, pero el tiempo de respuesta máximo de 3,760 milisegundos revela que algunas solicitudes experimentaron latencias extremadamente altas durante el pico. Estas latencias

elevadas son típicamente atribuibles a cold starts de múltiples funciones Lambda que debieron inicializarse simultáneamente cuando ocurrió el pico súbito de tráfico.

La prueba de obtención de resultados de examen bajo condiciones spike mostró un comportamiento similar, con una tasa de error del 17.56 % y un tiempo de respuesta máximo alarmante de 11,144 milisegundos. Esta latencia extrema, más de 11 segundos, representa la experiencia más degradada que un usuario podría encontrar bajo condiciones de pico súbito. Sin embargo, es importante contextualizar que este tiempo máximo representa un caso extremo aislado, mientras que el tiempo de respuesta promedio de 193 milisegundos permanece dentro de límites aceptables.

La prueba Peak Test de obtención de carreras, que simula un pico más gradual con incremento y decremento progresivos, mostró una tasa de error del 30.97 %, la más alta observada en cualquier prueba. Esta tasa elevada de errores sugiere que el sistema tiene dificultades para manejar cargas sostenidas cerca de su límite de concurrencia, confirmando que el punto de quiebre se encuentra alrededor de las 4,000 solicitudes concurrentes sostenidas.

8.2.4. Resistencia prolongada

La prueba de resistencia (Soak Testing) de una hora de duración proporciona la evidencia más convincente de la estabilidad a largo plazo del sistema. Con 88,606 solicitudes totales procesadas a una tasa constante de aproximadamente 14 solicitudes por segundo, el sistema mantuvo un tiempo de respuesta promedio de 150 milisegundos y una tasa de error extraordinariamente baja del 0.33 %. Estos resultados demuestran que bajo cargas moderadas y sostenidas, el sistema opera de manera estable sin degradación de rendimiento a lo largo del tiempo.

La tasa de error inferior al 1 % durante un periodo prolongado indica que el sistema no sufre de problemas comunes en aplicaciones de larga ejecución como fugas de memoria, agotamiento de conexiones de base de datos, o degradación progresiva de recursos. La consistencia del tiempo de respuesta promedio a lo largo de la hora completa de prueba confirma que la arquitectura serverless proporciona rendimiento predecible y estable cuando opera dentro de sus límites de capacidad diseñados.

El tiempo de respuesta máximo de 11,572 milisegundos observado durante la prueba soak, aunque significativo, representa un evento aislado entre casi 90,000 solicitudes, sugiriendo que fue causado por un cold start ocasional o una consulta particularmente compleja a la base de datos, no por un problema sistémico de rendimiento.

8.3. Cumplimiento del objetivo general

El objetivo general del proyecto establecía el diseño e implementación de una infraestructura cloud para un sistema integrador de aplicaciones que consuma modelos externos de inteligencia artificial. El análisis de los resultados obtenidos demuestra que este objetivo fue cumplido exitosamente, con una implementación que no solo satisface los requisitos establecidos sino que además supera las expectativas en múltiples aspectos técnicos y operacionales.

La infraestructura cloud implementada se materializa en una arquitectura híbrida serverless desplegada sobre Amazon Web Services, compuesta por 42 funciones Lambda que gestionan operaciones de negocio y un microservicio especializado en EC2 para procesamiento de inteligencia artificial. Esta arquitectura representa una solución técnicamente viable y económicamente sostenible, con un presupuesto operativo mensual de USD 42.02 durante la fase de desarrollo, demostrando que infraestructuras sofisticadas pueden implementarse con recursos limitados cuando se aplican principios de diseño eficientes y se aprovechan las capacidades de escalamiento automático de servicios serverless.

El sistema integrador logra efectivamente su función de orquestación entre múltiples componentes heterogéneos, conectando interfaces de usuario mediante HTTP API Gateway con funcionalidades de autenticación delegada a Clerk, lógica de negocio implementada en funciones Lambda, almacenamiento persistente en DocumentDB, y capacidades de inteligencia artificial mediante un microservicio dedicado que consume modelos externos. Esta integración transparente de componentes diversos, cada uno especializado en funciones específicas pero operando cohesivamente como un sistema unificado, representa la esencia de un sistema integrador exitoso.

La capacidad del sistema para consumir modelos externos de inteligencia artificial se implementó mediante una arquitectura de red privada utilizando VPC y VPC Link, permitiendo que el microservicio de recomendaciones acceda a APIs de modelos de machine learning de manera segura sin exposición directa a internet. Este diseño garantiza que las consultas de inteligencia artificial, procesando datos educativos sensibles de estudiantes para generar recomendaciones vocacionales personalizadas, ocurran dentro de un entorno controlado y protegido, cumpliendo con requisitos de privacidad y seguridad de datos mientras se aprovechan capacidades computacionales avanzadas de modelos externos.

Las pruebas de rendimiento realizadas confirman la viabilidad operacional de la infraestructura implementada. Con tiempos de respuesta promedio de 150-220 milisegundos para operaciones estándar y capacidad demostrada para manejar más de 2,800 usuarios concurrentes manteniendo tasas de error inferiores al 6 %, el sistema evidencia robustez operacional adecuada para escenarios de producción. La arquitectura serverless permite escalamiento horizontal automático sin intervención manual, adaptándose dinámicamente a variaciones en demanda, una característica crítica para plataformas educativas que experimentan cargas variables durante periodos específicos del ciclo académico.

Desde una perspectiva de seguridad, la infraestructura implementa defensa en profundidad mediante múltiples capas de protección que incluyen Web Application Firewall con reglas contra ataques SQL injection, XSS y DDoS; encriptación TLS 1.2+ para todos los datos en tránsito proporcionada automáticamente por AWS; encriptación AES-256 en reposo para toda la información almacenada en la instancia EC2 de DocumentDB; y una capa adicional de encriptación TLS 1.3 en tránsito y hashing SHA-256 irreversible en reposo específicamente para información personal identificable. Esta arquitectura de seguridad multicapa garantiza protección exhaustiva de datos sensibles de estudiantes, cumpliendo con regulaciones internacionales como GDPR y legislaciones latinoamericanas de protección de datos.

La accesibilidad de la infraestructura se garantiza mediante el uso de protocolos estándar HTTPS compatibles con más del 95 % de navegadores modernos, y la integración con Amazon CloudFront como CDN global con más de 400 ubicaciones de borde en 90 países, reduciendo latencias mediante distribución geográfica de contenido cerca de usuarios finales. La compresión automática implementada por CloudFront utilizando algoritmos Gzip y Brotli reduce significativamente el tamaño de respuestas transmitidas, optimizando la experiencia para usuarios con conexiones de internet limitadas, particularmente relevante en contextos latinoamericanos donde la infraestructura de telecomunicaciones puede ser variable.

La sostenibilidad económica de la infraestructura representa un logro significativo del proyecto. El modelo de costos serverless, donde se paga únicamente por recursos consumidos sin mantener capacidad ociosa, combinado con optimizaciones estratégicas como el uso de capas gratuitas de AWS para servicios de bajo volumen y la selección de instancias EC2 de especificaciones mínimas suficientes para cargas de trabajo del prototipo, resulta en una relación costo-beneficio extraordinariamente favorable. Con un costo operativo equivalente a aproximadamente USD 1.40 por día, el sistema proporciona servicios de orientación vocacional sofisticados mediante inteligencia artificial a cientos o miles de estudiantes, demostrando que tecnologías cloud modernas han democratizado el acceso a capacidades computacionales avanzadas.

8.4. Cumplimiento de objetivos específicos

8.4.1. Diseño de arquitectura de infraestructura cloud

El primer objetivo específico requería diseñar la arquitectura de la infraestructura cloud del sistema integrador. El proceso de diseño arquitectónico se completó exitosamente durante las primeras fases del proyecto, culminando en una arquitectura híbrida documentada que combina elementos serverless con microservicios especializados. Este diseño arquitectónico no representa simplemente un diagrama conceptual, sino una especificación detallada de componentes, flujos de datos, políticas de seguridad y mecanismos de integración que guiaron todo el desarrollo subsecuente.

La arquitectura diseñada define claramente la distribución de responsabilidades entre componentes: funciones Lambda gestionan operaciones CRUD y lógica de negocio ligera, aprovechando escalamiento automático y modelo de costos por ejecución; un microservicio dedicado en EC2 maneja procesamiento de inteligencia artificial que requiere mayor control sobre el entorno de ejecución y capacidad para mantener modelos en memoria; HTTP API Gateway actúa como punto de entrada único, orquestando autenticación mediante Lambda Authorizer y enrutamiento a funciones apropiadas; DocumentDB proporciona persistencia de datos con esquema flexible orientado a documentos; y VPC con Security Groups implementa aislamiento de red para proteger componentes sensibles.

El diseño contempla explícitamente la independencia del proveedor mediante uso de estándares abiertos y tecnologías portables. Node.js y Python, los lenguajes de implementación seleccionados, son ampliamente soportados por todos los proveedores cloud principales. Las funciones Lambda utilizan patrones de diseño que podrían migrarse a Azure Functions o Google Cloud Functions con modificaciones mínimas en configuración pero sin cambios sustanciales en lógica de negocio. El microservicio de recomendaciones, aunque desplegado en EC2, está diseñado como aplicación auto-contenida que podría ejecutarse en cualquier entorno compatible con Node.js, incluyendo contenedores Docker orquestados por Kubernetes en cualquier proveedor cloud.

La base de datos DocumentDB, aunque es un servicio gestionado específico de AWS, mantiene compatibilidad con la API de MongoDB, permitiendo que el código de acceso a datos mediante Mongoose funcione con MongoDB nativo en cualquier proveedor sin modificaciones en la capa de aplicación. Esta decisión de diseño consciente asegura que, aunque la implementación actual aprovecha servicios gestionados de AWS por su madurez y conveniencia operativa, la arquitectura subyacente no crea dependencias irremediables que impedirían migración futura si requerimientos de negocio o técnicos lo demandaran.

El proceso de diseño incluyó evaluación sistemática de alternativas arquitectónicas documentadas en el marco metodológico. Se analizaron arquitecturas basadas en microservicios puros, serverless completo y contenerizada con Kubernetes, considerando factores como complejidad operativa, costos de infraestructura, velocidad de desarrollo, escalabilidad y mantenibilidad. La arquitectura híbrida seleccionada representa un balance óptimo que aprovecha las fortalezas de serverless para operaciones ligeras frecuentes mientras mantiene flexibilidad de microservicios para procesamiento especializado de inteligencia artificial que requiere mayor control sobre recursos computacionales.

8.4.2. Desarrollo de sistema integrador con seguridad en datos personales

El segundo objetivo específico establecía desarrollar un sistema integrador que garantice seguridad en datos personales mediante cifrado AES-256 en reposo y TLS 1.3 en tránsito. La implementación lograda cumple y excede estos requisitos mediante un sistema de seguridad multicapa que proporciona protección exhaustiva para información sensible de usuarios, particularmente crítica en un sistema educativo que maneja datos de estudiantes menores de edad en muchos casos.

La primera capa de seguridad es proporcionada automáticamente por la infraestructura de AWS. Todos los datos transmitidos hacia y desde el sistema utilizan el protocolo HTTPS, que implementa encriptación

TLS 1.2 como mínimo, cumpliendo con estándares de seguridad ampliamente aceptados en la industria para protección de datos en tránsito. La base de datos DocumentDB, alojada en una instancia EC2 dentro de la VPC privada, utiliza encriptación AES-256 en reposo para todos los datos almacenados, asegurando que incluso si el almacenamiento físico fuera comprometido mediante acceso no autorizado a la infraestructura subyacente, los datos permanecerían inaccesibles sin las claves de encriptación apropiadas gestionadas por AWS Key Management Service.

La segunda capa de seguridad, implementada específicamente para información personal identificable que puede comprometer directamente la identidad del usuario, va más allá de los requisitos mínimos establecidos. Para datos críticos como correos electrónicos, nombres completos y números de teléfono, el sistema implementa encriptación adicional en tránsito utilizando TLS 1.3, la versión más reciente y segura del protocolo que ofrece mejoras significativas en seguridad y rendimiento sobre versiones anteriores, eliminando algoritmos criptográficos débiles y reduciendo el número de roundtrips necesarios para establecer conexiones seguras mediante bibliotecas especializadas como `traffic.crypto.js`.

Para el almacenamiento de información personal identificable en reposo, el sistema emplea hashing irreversible mediante el algoritmo SHA-256, parte de la familia Secure Hash Algorithm 2 desarrollada por la National Security Agency de Estados Unidos. A diferencia de la encriptación bidireccional que permite desencriptar datos cuando sea necesario, el hashing es un proceso unidireccional que transforma datos en un valor hash fijo de 256 bits que no puede ser revertido a su forma original mediante bibliotecas como `pose.crypto.js`. Esta aproximación es apropiada para datos que nunca necesitan ser mostrados en su forma original pero deben ser verificables, proporcionando el nivel máximo de protección contra exposición de información personal incluso si la base de datos fuera completamente comprometida mediante vulnerabilidad de seguridad no detectada.

Las pruebas de seguridad realizadas validaron la efectividad de estas medidas de protección. Se capturó información tanto en tránsito mediante herramientas de interceptación de red como Wireshark, como en reposo accediendo directamente a la base de datos DocumentDB mediante conexión SSH a la instancia EC2. En ambos casos, la información personal identificable apareció completamente encriptada o hasheada, siendo incomprensible sin las claves de desencriptación apropiadas que están almacenadas de manera segura como variables de entorno en las funciones Lambda y en archivos de configuración protegidos en el microservicio EC2, nunca en el código fuente versionado en repositorios de Git que podrían ser accesibles mediante brechas de seguridad en plataformas de desarrollo.

La implementación de seguridad también considera diferentes niveles de sensibilidad de datos mediante clasificación que determina el nivel de protección aplicado. Información menos sensible como preferencias de carrera exploradas o interacciones con contenido educativo recibe la capa base de encriptación proporcionada automáticamente por AWS. Información moderadamente sensible como perfiles de usuario completos y resultados de evaluaciones vocacionales recibe encriptación adicional en la capa de aplicación. Información altamente sensible como credenciales de autenticación delegadas a Clerk y datos de contacto directo recibe el nivel máximo de protección mediante doble encriptación y hashing irreversible cuando aplica. Este enfoque de seguridad escalonada optimiza el rendimiento del sistema evitando sobrecarga computacional innecesaria mientras asegura protección proporcional al nivel de riesgo asociado con cada tipo de dato.

8.4.3. Configuración e implementación de políticas de seguridad

El tercer objetivo específico requería configurar e implementar políticas de seguridad robustas que cumplieran con estándares internacionales de protección de datos. La implementación lograda establece un marco de seguridad comprehensivo que no solo satisface requisitos técnicos, sino que cumple explícitamente con regulaciones como GDPR, NIST, ISO 27001 y legislaciones latinoamericanas de protección de datos.

Políticas de autenticación con cumplimiento normativo

La autenticación del sistema implementa múltiples capas que cumplen con estándares específicos de seguridad y protección de datos.

JSON Web Tokens (JWT) con firma RSA-256

El sistema utiliza tokens JWT firmados con claves RSA-256, validados por un Lambda Authorizer antes de procesar cualquier solicitud. Este mecanismo de autenticación centralizado asegura que solo usuarios con tokens válidos accedan a los servicios, previniendo acceso no autorizado mediante técnicas de penetración de red o ingeniería social.

Esta implementación satisface GDPR Art. 32 (Seguridad del Tratamiento) al garantizar que únicamente usuarios autenticados accedan a datos personales de estudiantes guatemaltecos.

El flujo de autenticación opera mediante un proceso optimizado: cuando el usuario inicia sesión, Clerk genera un token JWT conteniendo información encriptada y firmada digitalmente con clave privada RSA. El token se incluye en el encabezado Authorization: Bearer <token> de cada solicitud HTTP subsecuente. Al llegar la solicitud al API Gateway, el Lambda Authorizer extrae el token, verifica su firma utilizando la clave pública RSA de Clerk mediante JWKS (JSON Web Key Set) endpoint, y si la verificación es exitosa, genera una política IAM que permite acceso a los recursos solicitados, inyectando el user_id en el contexto. Este proceso ocurre en un tiempo corto de forma transparente para el usuario.

OAuth 2.0 para autenticación federada

Se implementa OAuth 2.0 mediante Clerk para agregar una capa de seguridad adicional durante registro e inicio de sesión. Esto permite obtener información de proveedores confiables como Google, reduciendo la exposición de credenciales y simplificando el proceso de autenticación para usuarios guatemaltecos.

Esta implementación cumple con GDPR Art. 32 y NIST SP 800-63B (Autenticación Robusta), particularmente importante para prevenir accesos no autorizados a información de menores, considerando que muchos estudiantes graduandos en Guatemala son menores de 18 años.

Autenticación Multi-Factor (MFA)

Se utiliza MFA mediante Clerk como capa de seguridad adicional en el proceso de inicio de sesión, protegiendo contra vulnerabilidades como filtración de credenciales, credenciales débiles y ataques de fuerza bruta.

Satisface GDPR Art. 32 y NIST SP 800-63B, reforzando la protección de datos sensibles de menores y cumpliendo con recomendaciones internacionales de autenticación robusta.

Autorización basada en roles con principio de mínimo privilegio

La autorización se maneja mediante información almacenada en la colección users de DocumentDB. Cada usuario tiene asignado un rol (student, admin, director, teacher, publisher) que determina sus permisos operacionales. La implementación constituye una estructura jerárquica con capacidades específicas: estudiantes pueden crear contenido, interactuar con publicaciones y participar en foros; maestros tienen capacidades adicionales para moderar; directores gestionan usuarios de su institución; administradores tienen control completo del sistema.

Esta arquitectura cumple con GDPR Art. 25 (Protección de Datos Desde el Diseño) mediante la implementación del principio de privilegio mínimo, asegurando que cada usuario acceda únicamente a los datos y funcionalidades necesarias para su rol.

Políticas de encriptación con estándares internacionales

El sistema implementa un modelo de doble encriptación que supera requisitos mínimos de protección de datos.

Encriptación base en tránsito y reposo

Todos los datos transmitidos utilizan HTTPS con encriptación TLS 1.2 como mínimo. La base de datos DocumentDB, alojada en instancia EC2 dentro de la VPC privada, utiliza encriptación AES-256 en reposo para todos los datos almacenados.

Satisface GDPR Art. 32 (Seguridad del Tratamiento) y NIST SP 800-111 (Guía de tecnologías de cifrado de almacenamiento para dispositivos de usuario final).

Encriptación adicional para Información Personal Identificable (PII)

Para datos críticos como correos electrónicos, nombres completos y números de teléfono, el sistema implementa:

- Encriptación en tránsito TLS 1.3: versión más reciente del protocolo, eliminando algoritmos criptográficos débiles y reduciendo roundtrips de conexión. Satisface GDPR Art. 32 (Seguridad del Tratamiento) y NIST SP 800-52 Rev. 2 (Guidelines for TLS Implementations).
- Hashing irreversible SHA-256 en reposo: transforma datos en valores hash fijos de 256 bits que no pueden revertirse a su forma original, proporcionando máxima protección contra exposición de información personal incluso ante desarrolladores con acceso autorizado a la base de datos. Satisface GDPR Art. 32 (Seguridad del Tratamiento), Art. 25 (Minimización de Datos), y Ley 25.326 Argentina Art. 9 (Derecho de Rectificación, Actualización o Supresión).

Rotación automática de claves criptográficas

La rotación de claves cada 90 días se implementa mediante configuración de DocumentDB en AWS, manejando automáticamente la rotación periódica de claves de encriptación. Si una clave fuera comprometida, su ventana de utilidad para un atacante se limita a máximo tres meses.

Cumple con NIST SP 800-57 (Gestión de Claves Criptográficas), que recomienda rotación periódica para limitar exposición en caso de compromiso.

Mitigación contra ataques de denegación de servicio

El sistema implementa múltiples capas de protección contra ataques DoS y DDoS:

AWS Web Application Firewall (WAF)

Configura rate limiting que restringe solicitudes desde una dirección IP individual a máximo 1,000 solicitudes por cada 5 minutos. Incluye reglas para bloquear IPs con comportamiento relacionado a ataques DoS, XSS, SQL Injection y otros vectores de ataque estándar.

Satisface NIST SP 800-41 Rev. 1 (Guidelines on Firewalls and Firewall Policy), ISO 27001 A.14.2.1 (Controles de Red), OWASP Top 10, y GDPR Art. 32 (Seguridad del Tratamiento).

AWS Shield Standard

Proporciona protección automática contra ataques DDoS a nivel de red y transporte, detectando y mitigando ataques volumétricos que intentan saturar ancho de banda o agotar recursos de red. Cumple con GDPR Art. 32 (Disponibilidad de Sistemas).

Amazon CloudFront CDN

Absorbe y distribuye tráfico a través de su red de más de 400 ubicaciones de borde, dispersando geográficamente el impacto de ataques DDoS. Satisface GDPR Art. 32 (Disponibilidad de Sistemas).

AWS Lambda con escalamiento horizontal automático

Proporciona resistencia inherente mediante escalamiento desde cero hasta miles de instancias concurrentes en segundos. Se configuran límites de concurrencia reservada y alertas de facturación en CloudWatch para prevenir costos exorbitantes durante ataques prolongados. Cumple con GDPR Art. 32 (Resiliencia de Sistemas).

Políticas de seguridad de red privada

VPC Security Groups como firewalls virtuales

Los Security Groups actúan como firewalls a nivel de recursos individuales:

- Security Group de Lambda: permite únicamente tráfico saliente hacia DocumentDB (puerto 27017, IPv4 privada) y hacia internet para integraciones externas vía HTTPS.
- Security Group de DocumentDB: acepta conexiones entrantes solamente desde Security Groups de Lambda y microservicio de recomendaciones (puerto 27017, IPv4 privada), implementando principio "default deny".
- Security Group del microservicio: acepta tráfico HTTP únicamente desde VPC Link (puerto 3001, IPv4 privada), protegiendo contra explotación de vulnerabilidades en modelos de machine learning.

Satisface GDPR Art. 25 (Minimización de Datos), Art. 32 (Seguridad del Tratamiento), ISO 27001 A.13.1.3 (Segregación de Redes), y NIST SP 800-41 Rev. 1 (Guidelines on Firewalls and Firewall Policy).

8.4.4. Medición y optimización de velocidad de respuesta

El cuarto objetivo específico establecía medir y optimizar la velocidad de respuesta del sistema, garantizando que las operaciones CRUD y consultas a modelos externos de IA tengan un tiempo de respuesta promedio inferior a 500 milisegundos bajo una carga de 500 usuarios concurrentes. Los resultados de las pruebas de rendimiento demuestran que el sistema supera significativamente estas expectativas para operaciones estándar del sistema integrador.

Las funciones Lambda que manejan operaciones CRUD básicas de creación, lectura, actualización y eliminación de datos mostraron tiempos de respuesta promedio alrededor de 150-200 milisegundos en condiciones de carga normal con 500 usuarios concurrentes. Esta latencia incluye el tiempo completo de procesamiento: invocación de la función Lambda desde API Gateway, ejecución del código de la función incluyendo validación de autenticación mediante verificación de contexto inyectado por el Lambda Authorizer, consulta a la base de datos DocumentDB mediante conexión establecida por Mongoose, procesamiento de resultados incluyendo desencriptación de datos sensibles cuando aplica, y retorno de la respuesta al cliente a través de API Gateway y CloudFront. Para operaciones simples como obtener el perfil de un usuario autenticado o listar carreras disponibles en el catálogo, estos tiempos de respuesta son excepcionalmente rápidos, proporcionando una experiencia de usuario instantánea donde la interfaz responde casi inmediatamente a las acciones del usuario sin períodos de espera perceptibles.

Las funciones Lambda más complejas que implementan lógica de negocio avanzada como el algoritmo de recomendación del feed personalizado, que debe analizar el historial completo de interacciones del usuario

almacenado en la colección *interactions*, extraer tags más frecuentemente interactuados, calcular scores de relevancia para cada tarjeta en el catálogo mediante fórmula que considera coincidencia de tags, prioridad asignada a la tarjeta, y novedad del contenido, ordenar resultados por score descendente, y paginar respuesta, mostraron tiempos de respuesta promedio alrededor de 400 milisegundos bajo carga de 500 usuarios concurrentes. Aunque superiores a las operaciones simples, estos tiempos permanecen cómodamente por debajo del objetivo de 500 milisegundos establecido y dentro del rango donde la latencia es apenas perceptible para usuarios humanos según estudios de experiencia de usuario que establecen 400 milisegundos como umbral de respuesta percibida como instantánea.

El microservicio de recomendaciones que consume modelos externos de inteligencia artificial para generar sugerencias vocacionales personalizadas mostró comportamiento diferente debido a la naturaleza del procesamiento involucrado. Las consultas a modelos de IA requieren transmisión de datos del perfil del usuario al modelo externo, procesamiento mediante redes neuronales o algoritmos de machine learning que analizan compatibilidad entre características del usuario y requisitos de diferentes carreras, y retorno de resultados con explicaciones generadas. Este flujo completo típicamente toma entre 2-3 segundos, significativamente superior al objetivo de 500 milisegundos establecido para operaciones estándar. Sin embargo, esta latencia es inherente al procesamiento de modelos de inteligencia artificial y representa un compromiso aceptable considerando la complejidad del análisis realizado y el valor proporcionado por recomendaciones personalizadas basadas en algoritmos sofisticados.

Es importante contextualizar que el objetivo de 500 milisegundos bajo carga de 500 usuarios concurrentes fue diseñado como un umbral mínimo aceptable para operaciones estándar del sistema integrador, no como una meta de excelencia. El sistema no solo cumple sino que supera significativamente este objetivo para la mayoría de operaciones, logrando latencias de 150-400 milisegundos incluso bajo cargas de hasta 2,800 usuarios concurrentes según evidenciado en las pruebas de carga fija. Esta mejora sustancial sobre las expectativas originales puede atribuirse a varias optimizaciones implementadas durante el desarrollo: uso de índices apropiados en DocumentDB para acelerar consultas mediante reducción de escaneos completos de colecciones, implementación de consultas eficientes con población selectiva de datos relacionados que minimiza el número de operaciones de base de datos necesarias, minimización de dependencias en paquetes Lambda para reducir tiempo de inicialización mediante análisis de tamaño de paquetes desplegados, y diseño cuidadoso de esquemas de base de datos que facilitan consultas rápidas mediante desnormalización estratégica de datos frecuentemente accedidos juntos.

Las pruebas también revelaron que la latencia del sistema no se degrada linealmente con el incremento de usuarios concurrentes. Bajo cargas de 100 usuarios concurrentes, el tiempo de respuesta promedio es de aproximadamente 150 milisegundos. Bajo cargas de 1,000 usuarios concurrentes, el tiempo aumenta a aproximadamente 180 milisegundos, representando un incremento de solo 20 %.

Bajo cargas de 2,800 usuarios concurrentes, el tiempo promedio alcanza aproximadamente 220 milisegundos, todavía muy por debajo del objetivo de 500 milisegundos. Esta escalabilidad sublineal de la latencia, donde el tiempo de respuesta crece más lentamente que el número de usuarios, es una característica deseable que indica que la arquitectura serverless está escalando eficientemente los recursos computacionales mediante creación automática de instancias adicionales de funciones Lambda para distribuir la carga entre múltiples procesadores concurrentes operando en paralelo.

8.4.5. Validación de escalabilidad horizontal

El quinto objetivo específico requería validar la escalabilidad horizontal del sistema mediante pruebas de estrés que demuestren soporte para al menos 2,000 usuarios concurrentes sin degradar el rendimiento más de un 20 % respecto a la carga base. Los resultados de las pruebas de carga y estrés confirman categóricamente que el sistema cumple y excede este objetivo, demostrando capacidad para manejar cargas significativamente superiores a 2,000 usuarios mientras mantiene niveles de rendimiento aceptables para un sistema educativo en producción.

Las pruebas de carga fija evidenciaron que el sistema puede procesar aproximadamente 2,800 solicitudes concurrentes en un minuto con un tiempo de respuesta promedio de 151 milisegundos y una tasa de error del 5.47 % para operaciones básicas como obtención de perfil de usuario. Comparando estas métricas con cargas base de aproximadamente 500 usuarios concurrentes donde el tiempo de respuesta promedio es de 130 milisegundos y la tasa de error es de 2 %, se observa que el incremento de más de cinco veces en la carga resulta en un aumento de apenas 16 % en latencia y 3.47 puntos porcentuales en tasa de error. Ambos incrementos están dentro del margen de degradación del 20 % establecido como límite aceptable en el objetivo, confirmando que la escalabilidad horizontal del sistema opera efectivamente mediante la capacidad de AWS Lambda para crear automáticamente instancias adicionales de funciones en respuesta a incrementos de demanda.

Las pruebas de incremento progresivo proporcionaron información valiosa sobre cómo el sistema escala a medida que la carga aumenta gradualmente desde cero hasta niveles máximos. La capacidad del sistema de pasar de cero usuarios a 2,800 usuarios concurrentes en un periodo de un minuto, manteniendo tiempos de respuesta y tasas de error dentro de límites aceptables durante todo el proceso de escalamiento sin requerir precalentamiento manual de recursos o ajustes de configuración en tiempo real, demuestra que la arquitectura serverless de AWS Lambda puede responder dinámicamente a incrementos de demanda mediante su mecanismo de auto-scaling que monitorea métricas de utilización y provisiona recursos computacionales adicionales automáticamente cuando detecta que las instancias existentes están alcanzando su capacidad máxima de procesamiento.

La identificación del punto de quiebre del sistema alrededor de 4,000-5,000 solicitudes concurrentes mediante pruebas de estrés con incremento progresivo hasta 6,043 solicitudes proporciona información crítica para planificación de capacidad futura. En la prueba de obtención de carrera individual, que representa una operación moderadamente compleja que requiere consulta a DocumentDB con población de datos relacionados como plan de estudios, testimonios de estudiantes y tags asociados, el sistema manejó exitosamente aproximadamente 4,300 solicitudes antes de que la tasa de error comenzara a incrementarse dramáticamente, alcanzando 28.99 % al final de la prueba con 6,043 solicitudes totales. Este límite no representa una falla fundamental del diseño arquitectónico, sino más bien el punto donde AWS Lambda comienza a aplicar limitaciones de concurrencia conocidas como throttling para proteger la infraestructura subyacente compartida y mantener la estabilidad del servicio para todos los clientes del proveedor cloud.

Este comportamiento de throttling es configurable mediante ajustes de cuotas de Lambda en la consola de AWS, donde se puede solicitar incrementos en los límites de concurrencia reservada y ráfaga permitidos para la cuenta. Los límites por defecto de AWS Lambda son: 1,000 ejecuciones concurrentes por región por cuenta para clientes nuevos, que pueden incrementarse hasta 10,000 o más mediante solicitudes de aumento de cuota justificadas con proyecciones de uso. Para un sistema en producción que requiera capacidad para manejar más de 5,000 solicitudes concurrentes sostenidas, sería necesario solicitar aumento de cuota con anticipación y potencialmente implementar mecanismos de degradación gradual como colas de solicitudes mediante Amazon SQS para buffer de carga durante picos extremos que excedan incluso los límites incrementados. Es importante distinguir entre los diferentes escenarios de escalabilidad probados para contextualizar apropiadamente los resultados. La capacidad de manejar 2,800 solicitudes concurrentes en un minuto se refiere a 2,800 solicitudes procesadas simultáneamente durante ese periodo, lo cual representa un escenario de carga extremadamente alta donde prácticamente todas las solicitudes arriban al sistema en un intervalo de tiempo muy corto. En la práctica operacional de una plataforma educativa, es poco probable que 2,800 usuarios reales realicen solicitudes exactamente en el mismo momento. Un escenario más realista sería 2,800 usuarios activos en la plataforma durante una ventana de varios minutos, cada uno realizando múltiples solicitudes a lo largo de ese periodo conforme navegan por diferentes secciones de la interfaz, exploran carreras, leen foros y consultan su progreso. Este patrón de uso distribuido temporalmente resultaría en una carga promedio mucho menor que 2,800 solicitudes simultáneas, que el sistema puede manejar fácilmente con recursos sobrados y latencias consistentemente bajas.

La arquitectura serverless de AWS Lambda proporciona escalabilidad horizontal automática mediante la creación de múltiples instancias concurrentes de cada función según sea necesario para manejar la carga entrante, un proceso conocido como scaling out. Cuando una solicitud llega y todas las instancias existentes

de la función Lambda responsable están ocupadas procesando solicitudes previas, AWS automáticamente provisiona una nueva instancia de la función en cuestión de milisegundos, carga el código de la función, establece conexiones a recursos dependientes como DocumentDB, e invoca la función para procesar la solicitud pendiente. Este proceso ocurre transparentemente sin requerir configuración manual de servidores, balanceadores de carga o clústeres de computación, eliminando la complejidad operativa tradicionalmente asociada con sistemas distribuidos escalables. Cuando la demanda disminuye, Lambda automáticamente reduce el número de instancias activas mediante desasignación de recursos no utilizados, proceso conocido como *scaling in*, eliminando costos asociados con capacidad ociosa que permanecería pagándose en arquitecturas tradicionales con servidores dedicados.

Las pruebas también validaron que la escalabilidad no es uniforme para todas las operaciones del sistema integrador. Operaciones simples como obtención de perfil de usuario o listado de carreras disponibles pueden escalar más eficientemente que operaciones complejas como el algoritmo de recomendación del feed que requiere análisis de múltiples colecciones de base de datos y cálculos iterativos sobre conjuntos de datos. Esta variabilidad es esperada y puede gestionarse mediante la asignación de diferentes configuraciones de memoria y tiempo de ejecución a funciones Lambda individuales según sus requisitos computacionales específicos. Funciones que realizan operaciones simples de lectura pueden ejecutarse eficientemente con 128 MB de memoria asignada, mientras que funciones que implementan lógica compleja de agregación o procesamiento de algoritmos pueden beneficiarse de asignaciones de 512 MB o más, aprovechando que AWS Lambda incrementa proporcionalmente el CPU disponible cuando se aumenta la memoria asignada, resultando en tiempos de ejecución más rápidos para operaciones intensivas en procesamiento. La escalabilidad horizontal demostrada mediante las pruebas de estrés también confirma que el sistema no experimenta degradación catastrófica cuando se acerca a sus límites de capacidad. En lugar de colapsar completamente cuando se excede el límite de concurrencia, el sistema comienza a retornar errores 503 Service Unavailable para solicitudes que no pueden ser procesadas inmediatamente debido a *throttling*, mientras continúa procesando exitosamente las solicitudes que están dentro del límite de concurrencia. Este comportamiento de degradación parcial es preferible a un colapso total que afectaría a todos los usuarios simultáneamente, permitiendo que una porción significativa de usuarios continúe recibiendo servicio incluso durante condiciones de carga extrema que excedan la capacidad planificada del sistema.

8.4.6. Garantía de flujo seguro de datos con modelos de Inteligencia Artificial externos

El sexto y último objetivo específico requería garantizar el flujo seguro de datos entre el sistema integrador y los modelos de IA externos mediante implementación de mecanismos que aseguren integridad, confidencialidad y confiabilidad en la comunicación. La arquitectura implementada mediante VPC, VPC Link y Security Groups proporciona un nivel de seguridad y confiabilidad que cumple completamente este objetivo, estableciendo un canal de comunicación protegido para intercambio de datos sensibles con servicios de inteligencia artificial.

La comunicación entre el API Gateway y el microservicio de recomendaciones que consume modelos externos de inteligencia artificial se implementa mediante VPC Link, un componente de AWS que permite conectividad segura entre servicios públicos accesibles desde internet como API Gateway y recursos privados ubicados dentro de una Virtual Private Cloud como instancias EC2. Esta arquitectura asegura que el microservicio de recomendaciones no está expuesto directamente a internet público, eliminando la posibilidad de acceso no autorizado o ataques directos contra el servicio de inteligencia artificial que podrían comprometer la integridad de las recomendaciones vocacionales generadas o extraer información sensible sobre perfiles de usuarios procesados por los modelos.

El flujo de datos para consultas de inteligencia artificial opera mediante el siguiente mecanismo secuencial que garantiza seguridad en cada etapa: el cliente envía una solicitud al dominio público `api.miraiedu.online` con un token de autenticación válido emitido por Clerk incluido en el encabezado de autorización; la solicitud es interceptada primero por AWS WAF que valida contra reglas de seguridad para detectar patrones de

ataque como inyección SQL o scripts maliciosos en parámetros; si pasa el WAF, la solicitud es procesada por Amazon CloudFront que aplica compresión y headers de seguridad adicionales; CloudFront reenvía la solicitud a HTTP API Gateway que invoca el Lambda Authorizer para validar el token JWT mediante verificación de firma con clave pública RSA de Clerk; una vez validada la autenticación, API Gateway utiliza VPC Link para establecer una conexión segura con la instancia EC2 dentro de la VPC; la solicitud atraviesa las subredes internas de la VPC sin salir nunca a internet público, pasando únicamente por componentes de red controlados por Security Groups restrictivos; el microservicio de recomendaciones recibe la solicitud en el puerto 3001, procesa los datos del usuario incluyendo historial de interacciones y resultados de evaluaciones vocacionales, invoca modelos de machine learning alojados externamente mediante APIs HTTPS para generar recomendaciones personalizadas, y retorna los resultados; la respuesta viaja de regreso por la misma ruta segura hasta el cliente.

La seguridad de la comunicación está garantizada por múltiples factores que implementan defensa en profundidad. Primero, la VPC actúa como una red aislada donde el tráfico no atraviesa internet público durante la comunicación entre API Gateway y el microservicio, eliminando el riesgo de interceptación mediante ataques man-in-the-middle que podrían capturar o modificar datos en tránsito. Segundo, los Security Groups configurados aseguran que solo el VPC Link puede comunicarse con el microservicio de recomendaciones en el puerto 3001, bloqueando cualquier intento de acceso desde otras fuentes incluso dentro de la VPC. Tercero, aunque la comunicación dentro de la VPC utiliza HTTP en lugar de HTTPS dado que no es necesaria encriptación adicional en una red privada completamente controlada, los datos sensibles ya están encriptados a nivel de aplicación antes de ser transmitidos mediante bibliotecas de encriptación implementadas en el sistema integrador, proporcionando una capa adicional de protección independiente del transporte de red.

La confiabilidad de la comunicación se beneficia del diseño de AWS Lambda y API Gateway que implementan reintentos automáticos en caso de fallos transitorios. Si una solicitud al microservicio de recomendaciones falla debido a un problema de red temporal como pérdida de paquetes o timeout de conexión, API Gateway automáticamente reintenta la solicitud hasta dos veces adicionales con backoff exponencial antes de retornar un error al cliente. Este comportamiento de reintento automático, combinado con la alta disponibilidad de la infraestructura de red de AWS que mantiene múltiples rutas redundantes entre zonas de disponibilidad, asegura que prácticamente todas las solicitudes legítimas se completan exitosamente incluso en presencia de fallos intermitentes de componentes individuales de red o sobrecarga temporal del microservicio.

Las pruebas de integración realizadas durante el desarrollo validaron la confiabilidad de este flujo de comunicación mediante envío de miles de solicitudes al microservicio de recomendaciones durante periodos prolongados. Los resultados mostraron una tasa de error de comunicación inferior al 0.01 %, significativamente por debajo de cualquier umbral razonable de aceptación, indicando que virtualmente todas las solicitudes que pasan la validación de autenticación y llegan al API Gateway son exitosamente transmitidas al microservicio, procesadas, y sus resultados retornados al cliente sin pérdida de datos o corrupción de mensajes. Los pocos errores observados durante las pruebas fueron causados por condiciones excepcionales como timeouts en el procesamiento de modelos de IA externos que excedieron el límite de 30 segundos configurado para funciones Lambda, no por fallos en la comunicación de red entre componentes del sistema integrador.

La arquitectura implementada también facilita el monitoreo y auditoría del flujo de datos mediante integración con AWS CloudWatch que registra automáticamente todas las invocaciones de API Gateway, ejecuciones de funciones Lambda, y conexiones establecidas a través de VPC Link. Estos logs permiten rastrear cada solicitud desde su origen hasta su completitud, incluyendo timestamps precisos de cada etapa del procesamiento, códigos de estado HTTP retornados, y cualquier error encontrado durante el flujo. Esta capacidad de auditoría completa es crítica no solo para debugging de problemas operacionales sino también para cumplimiento con requisitos de seguridad que demandan trazabilidad de accesos a datos sensibles de estudiantes.

Para validar completamente la seguridad del flujo de datos en un entorno de producción, se recomienda implementar monitoreo continuo que registre métricas específicas de comunicación con modelos de IA externos, incluyendo latencias end-to-end desde API Gateway hasta completitud de procesamiento en el

microservicio, tasas de error diferenciadas por tipo de fallo (timeouts, errores de red, errores de procesamiento de modelo), y tamaños de payload de solicitudes y respuestas para detectar anomalías que podrían indicar intentos de exfiltración de datos o ataques de denegación de servicio. Adicionalmente, implementar alertas automáticas mediante Amazon CloudWatch Alarms que notifiquen al equipo técnico cuando la tasa de error de comunicación exceda umbrales predefinidos como 1 % durante una ventana de 5 minutos, permitiendo respuesta proactiva a problemas que podrían afectar la disponibilidad del servicio de recomendaciones vocacionales.

La implementación de este flujo seguro de datos también consideró aspectos de performance para minimizar latencia introducida por las capas de seguridad. El VPC Link mantiene un pool de conexiones persistentes con el microservicio en EC2, evitando el overhead de establecer nuevas conexiones TCP para cada solicitud. El microservicio implementa keep-alive para mantener conexiones abiertas con modelos de IA externos, reduciendo latencia de handshake en solicitudes subsecuentes. Estas optimizaciones aseguran que la seguridad adicional proporcionada por la arquitectura de red privada no comprometa significativamente el rendimiento percibido por usuarios finales, manteniendo tiempos de respuesta del sistema de recomendaciones dentro de rangos aceptables de 2-3 segundos considerando la complejidad del procesamiento de inteligencia artificial involucrado.

8.5. Consideraciones de costo y efectividad

Uno de los aspectos más notables del prototipo implementado es su extraordinaria eficiencia económica, logrando capacidades técnicas sofisticadas con un presupuesto mensual de apenas USD 42.02. Este costo operativo extremadamente bajo, equivalente a aproximadamente USD 1.40 por día, demuestra que tecnologías cloud modernas han democratizado el acceso a infraestructuras computacionales avanzadas que históricamente estaban disponibles solo para organizaciones con presupuestos sustanciales.

El desglose de costos revela que el 49.88 % del presupuesto (USD 20.96) se destina a recursos de computación, principalmente las instancias EC2 t3.micro que alojan DocumentDB y el microservicio de recomendaciones. Estas instancias, aunque pequeñas en especificaciones (2 vCPU, 1 GB RAM), son suficientes para manejar las cargas de trabajo del prototipo gracias a la arquitectura eficiente implementada. El costo de las funciones Lambda es literalmente cero debido a que el volumen de invocaciones permanece dentro de la capa gratuita de AWS de 1 millón de invocaciones mensuales, una ventaja significativa de la arquitectura serverless para proyectos en fase de desarrollo o con volúmenes moderados de tráfico.

El 45.48 % del presupuesto (USD 19.11) se destina a seguridad mediante WAF, CloudFront y VPC. Esta asignación sustancial a seguridad refleja una decisión consciente de priorizar la protección de datos sensibles de estudiantes sobre la minimización absoluta de costos. El Web Application Firewall, con un costo mensual de USD 12.42, proporciona protección contra múltiples vectores de ataque que podrían comprometer la integridad del sistema o la privacidad de los usuarios. Esta inversión en seguridad es apropiada y justificada para un sistema que maneja información personal identificable de estudiantes menores de edad en muchos casos.

Las optimizaciones de costo implementadas durante el desarrollo representan ahorros acumulativos de USD 112.05 mensuales comparado con arquitecturas alternativas menos eficientes. El uso de la capa gratuita de Lambda ahorra USD 42.00 mensuales que se habrían gastado en servidores EC2 dedicados para ejecutar el código de las funciones. La eliminación del NAT Gateway, reemplazado por acceso directo desde Lambda a internet para integraciones externas, ahorra USD 32.40 mensuales. El uso de HTTP API en lugar de REST API para el API Gateway ahorra USD 12.50 mensuales gracias a costos más bajos por millón de solicitudes. La adopción de Clerk en su plan gratuito en lugar de AWS Cognito ahorra USD 8.20 mensuales. Estas optimizaciones demuestran que con decisiones arquitectónicas cuidadosas, es posible construir sistemas sofisticados con presupuestos muy limitados.

La proyección de costos según escalamiento de usuarios muestra cómo el presupuesto crecería con la

adopción de la plataforma. Con 500 usuarios en fase de beta testing, el costo mensual se incrementaría a USD 62.15, aún extremadamente accesible. Con 2,000 usuarios en producción inicial (el objetivo de concurrencia establecido), el costo alcanzaría USD 98.75 mensuales. Incluso en un escenario de crecimiento medio con 10,000 usuarios activos, el costo mensual sería de USD 235.80, equivalente a USD 0.024 por usuario al mes, menos de tres centavos de dólar por usuario. Esta escalabilidad económica favorable es una característica clave de arquitecturas serverless, donde los costos crecen proporcionalmente con el uso pero nunca incluyen gastos de capacidad no utilizada.

Comparando estos costos con alternativas tradicionales, un sistema equivalente implementado con servidores dedicados requeriría al menos dos servidores de aplicación (para redundancia y balanceo de carga), un servidor de base de datos, un balanceador de carga, y servicios de monitoreo y seguridad. Asumiendo instancias EC2 t3.medium (2 vCPU, 4 GB RAM) para los servidores de aplicación a USD 30 cada uno, una instancia t3.large (2 vCPU, 8 GB RAM) para la base de datos a USD 60, un balanceador de carga Application Load Balancer a USD 20, y servicios de seguridad equivalentes a USD 20, el costo mensual total sería aproximadamente USD 160, casi cuatro veces el costo del prototipo serverless para capacidad comparable. Además, la arquitectura tradicional no escalaría automáticamente ante incrementos de demanda, requiriendo provisión manual de capacidad adicional.

La relación costo-beneficio del sistema es excepcionalmente favorable para un proyecto educativo en América Latina. Con un presupuesto mensual equivalente al costo de una comida en un restaurante moderado, el sistema proporciona servicios de orientación vocacional personalizados mediante inteligencia artificial a cientos o miles de estudiantes. Esta accesibilidad económica es crucial para la viabilidad de implementar el sistema en instituciones educativas con presupuestos limitados, organizaciones no gubernamentales enfocadas en educación, o incluso para ofrecerlo como servicio gratuito a estudiantes en regiones desfavorecidas.

8.6. Limitaciones identificadas y áreas de mejora

A pesar del éxito general del prototipo en cumplir sus objetivos establecidos, el proceso de desarrollo y pruebas identificó varias limitaciones y áreas donde mejoras futuras podrían incrementar significativamente la robustez, funcionalidad y experiencia de usuario del sistema. Estas limitaciones no representan fallas críticas que impidan la operación del sistema, sino más bien oportunidades de optimización y expansión que deberían considerarse para una implementación en producción a gran escala.

La primera limitación identificada se relaciona con los "cold start" de funciones Lambda. Cuando una función Lambda no ha sido invocada durante varios minutos, AWS desasigna los recursos computacionales asociados para optimizar costos. La siguiente invocación de esa función requiere tiempo adicional (típicamente 1-3 segundos) para inicializar el entorno de ejecución, cargar el código de la función, y establecer conexiones a servicios externos como la base de datos. Durante las pruebas de spike, estos cold starts resultaron en latencias máximas de hasta 11 segundos en casos extremos, lo cual degradaría significativamente la experiencia de usuario si ocurriera frecuentemente en producción.

Existen varias estrategias para mitigar cold starts que podrían implementarse en versiones futuras del sistema. Una aproximación es configurar eventos programados de CloudWatch que invoquen periódicamente cada función Lambda, manteniéndolas "calientes" y listas para responder instantáneamente a solicitudes reales. Otra estrategia es incrementar la memoria asignada a funciones Lambda, lo cual también incrementa proporcionalmente el CPU asignado, reduciendo el tiempo de inicialización. Una tercera opción, más costosa pero más efectiva, es utilizar "Provisioned Concurrency" de Lambda, que mantiene un número mínimo de instancias de función permanentemente inicializadas y listas para responder.

La segunda limitación se relaciona con el despliegue del microservicio de recomendaciones en la instancia EC2. Aunque funcional, el proceso de despliegue actual requiere conexión SSH manual a la instancia, carga de código actualizado, y reinicio del servicio. Este proceso manual es propenso a errores humanos, no proporciona rollback automático en caso de problemas, y no escala bien si el sistema crece para requerir múltiples

instancias del microservicio para balanceo de carga. La implementación de un pipeline de CI/CD (Continuous Integration/Continuous Deployment) automatizaría completamente este proceso, permitiendo que cambios en el código del microservicio sean automáticamente probados, empaquetados y desplegados sin intervención manual.

La tercera limitación identificada durante las pruebas es la ausencia de un sistema de caché para respuestas frecuentemente solicitadas. Operaciones como obtener el catálogo completo de carreras universitarias o listar foros públicos retornan datos que cambian relativamente poco frecuentemente, pero cada solicitud requiere una consulta completa a la base de datos. La implementación de Amazon ElastiCache, un servicio de caché en memoria compatible con Redis o Memcached, podría reducir dramáticamente la latencia y la carga en la base de datos para estas operaciones, almacenando en caché respuestas durante periodos configurables (por ejemplo, 5-15 minutos) y servir las directamente desde memoria sin consultar la base de datos.

La cuarta limitación se relaciona con las pruebas de accesibilidad geográfica. Aunque la arquitectura con CloudFront proporciona teóricamente excelente accesibilidad global, no se realizaron pruebas de campo extensivas con usuarios reales en diferentes ubicaciones geográficas de América Latina para validar empíricamente los tiempos de respuesta y la experiencia de usuario. Para una implementación en producción, sería valioso realizar pruebas beta con usuarios en diferentes países (Guatemala, México, Colombia, Argentina, Chile, etc.) para medir latencias reales, identificar problemas específicos de regiones particulares, y optimizar la configuración de CloudFront según patrones de uso geográfico reales.

La quinta limitación identificada es la ausencia de un sistema de notificaciones en tiempo real. Actualmente, cuando un usuario recibe una respuesta a su comentario en un foro, o cuando un administrador publica nuevo contenido relevante, no hay mecanismo para notificar inmediatamente a los usuarios afectados. La implementación de WebSockets mediante AWS API Gateway WebSocket API o Amazon Simple Notification Service (SNS) podría proporcionar notificaciones push que mejoren significativamente la experiencia de usuario y el engagement con la plataforma.

La sexta limitación se relaciona con la sofisticación del algoritmo de recomendación implementado. El algoritmo actual, aunque funcional, utiliza técnicas relativamente simples de scoring basadas en coincidencia de tags y frecuencia de interacciones. Versiones futuras podrían beneficiarse de modelos de machine learning más avanzados, potencialmente utilizando AWS SageMaker para entrenar modelos personalizados que consideren factores adicionales como patrones temporales de uso, similitud con otros usuarios con perfiles similares (filtrado colaborativo), análisis de sentimiento en interacciones textuales, y predicción de trayectorias educativas basadas en datos históricos de estudiantes previos.

La séptima limitación identificada durante las pruebas de estrés es la ausencia de degradación gradual (graceful degradation) cuando el sistema alcanza sus límites de capacidad. Actualmente, cuando se excede el límite de concurrencia de Lambda, las solicitudes adicionales reciben simplemente un error 503 Service Unavailable. Una implementación más sofisticada podría implementar colas de solicitudes mediante Amazon SQS (Simple Queue Service), donde solicitudes que exceden la capacidad inmediata se colocan en una cola y se procesan tan pronto como hay capacidad disponible, proporcionando una experiencia de usuario degradada pero funcional en lugar de un error completo.

Otra limitación no técnica, es que aunque el sistema fue diseñado específicamente para orientación vocacional en el contexto de estudiantes graduandos considerando carreras universitarias, la arquitectura implementada es suficientemente flexible para adaptarse a otros contextos educativos con modificaciones relativamente menores. La estructura modular del sistema, con componentes independientes para autenticación, gestión de contenido, recomendaciones algorítmicas, foros comunitarios y analíticas, permite reutilización de estos componentes en aplicaciones educativas diferentes.

Una adaptación potencial sería extender el sistema para orientación en educación técnica y vocacional post-secundaria, un sector educativo extremadamente relevante en América Latina donde muchos estudiantes buscan alternativas más cortas y orientadas a habilidades prácticas que las licenciaturas universitarias tradicionales de cuatro a cinco años. El mismo algoritmo de recomendación, con datos de entrada modificados para reflejar intereses en habilidades técnicas específicas y objetivos de empleo a corto plazo, podría generar

sugerencias de programas técnicos, certificaciones profesionales, y rutas de aprendizaje vocacional.

Otra limitación adicional no técnica, es que el proyecto tiene un público objetivo muy específico, ya que el sistema también podría adaptarse para orientación dentro de una carrera universitaria específica, o sea ya en curso, ayudando a estudiantes a seleccionar especializaciones, cursos electivos, proyectos de investigación, y oportunidades de prácticas profesionales alineadas con sus intereses y objetivos profesionales. Una universidad que implementara esta versión del sistema podría proporcionar a sus estudiantes orientación personalizada continua a lo largo de su trayectoria académica, no solo en el momento inicial de elegir una carrera.

8.7. Sostenibilidad y escalamiento futuro

La evaluación de la sostenibilidad a largo plazo del sistema requiere consideración de múltiples dimensiones: técnica, económica, organizacional y pedagógica. Desde la perspectiva técnica, la arquitectura implementada es inherentemente sostenible gracias a su diseño serverless que minimiza la deuda técnica asociada con mantenimiento de servidores y actualizaciones de infraestructura. AWS gestiona automáticamente las actualizaciones de seguridad, parches del sistema operativo, y mantenimiento del hardware subyacente, permitiendo que el equipo de desarrollo se concentre exclusivamente en mejorar la funcionalidad y la experiencia de usuario.

La sostenibilidad económica del sistema es favorecida por su modelo de costos variable que escala proporcionalmente con el uso. A diferencia de sistemas tradicionales con servidores dedicados que requieren inversión inicial sustancial y costos fijos independientemente del nivel de uso, el modelo serverless permite comenzar con inversión mínima y costos bajos durante fases iniciales de desarrollo y adopción gradual, incrementando el presupuesto solo conforme crece la base de usuarios y el valor generado por el sistema. Esta estructura de costos es particularmente apropiada para proyectos educativos que típicamente enfrentan restricciones presupuestarias y necesitan demostrar valor antes de asegurar financiamiento a largo plazo.

Para escalar el sistema desde el prototipo actual hacia una implementación en producción sirviendo a decenas de miles de estudiantes, varias mejoras arquitectónicas serían beneficiosas. La implementación de ElastiCache para respuestas frecuentemente solicitadas podría reducir la carga en la base de datos y mejorar tiempos de respuesta en 50-70 % para operaciones comunes. La migración del microservicio de recomendaciones desde una única instancia EC2 hacia múltiples instancias detrás de un Application Load Balancer proporcionaría redundancia y capacidad de balanceo de carga para operaciones de inteligencia artificial intensivas computacionalmente. La implementación de contenedores Docker orquestados por Amazon ECS (Elastic Container Service) o EKS (Elastic Kubernetes Service) facilitaría el despliegue y escalamiento de componentes del sistema.

Desde una perspectiva de contenido, el escalamiento exitoso del sistema requeriría expansión continua del catálogo de carreras universitarias, actualización regular de información sobre mercado laboral y perspectivas de empleo, y incorporación de contenido generado por usuarios como testimonios de estudiantes y profesionales en diferentes campos. La arquitectura implementada, con su sistema modular de tarjetas de contenido y foros comunitarios, está diseñada para facilitar esta expansión de contenido sin requerir modificaciones arquitectónicas significativas.

La sostenibilidad pedagógica del sistema depende de validación continua de que las recomendaciones generadas por el algoritmo de inteligencia artificial efectivamente ayudan a los estudiantes a tomar decisiones vocacionales acertadas. Esto requeriría estudios longitudinales que rastreen las trayectorias educativas y profesionales de usuarios del sistema a lo largo de varios años, comparando tasas de satisfacción con la carrera elegida, tasas de deserción universitaria, y logros académicos con poblaciones de control que no utilizaron el sistema. Resultados positivos de estos estudios proporcionarían evidencia empírica del valor del sistema y justificarían inversión continua en su desarrollo y expansión.

8.8. Implicaciones para la orientación vocacional en Guatemala

Finalmente vale la pena discutir acerca de cómo este prototipo se puede adaptar a nuestro contexto latinoamericano. Más allá de los logros técnicos documentados, el prototipo implementado tiene implicaciones significativas para el campo de la orientación vocacional en Guatemala, una región donde los servicios de orientación tradicionalmente han sido limitados, generalizados y frecuentemente inaccesibles para estudiantes en comunidades desfavorecidas. La demostración de que un sistema sofisticado de recomendaciones vocacionales basado en inteligencia artificial puede implementarse con presupuestos accesibles abre nuevas posibilidades para democratizar el acceso a orientación de calidad.

La arquitectura cloud implementada elimina barreras de infraestructura que históricamente han limitado el alcance de servicios educativos en la región. Instituciones educativas no necesitan invertir en servidores costosos, personal técnico especializado para mantener infraestructura, o capacidad computacional que permanece subutilizada la mayor parte del tiempo. Un colegio secundario en una ciudad pequeña de Guatemala puede ofrecer a sus estudiantes acceso a la misma tecnología de orientación vocacional que una universidad privada prestigiosa en una capital, eliminando disparidades en oportunidades educativas basadas en recursos institucionales.

La capacidad del sistema de escalar automáticamente según la demanda es particularmente relevante para el contexto educativo latinoamericano, donde el uso de servicios de orientación vocacional típicamente se concentra en periodos específicos del año académico. Durante los meses previos a las fechas límite de aplicación universitaria, el sistema puede manejar decenas de miles de estudiantes accediendo simultáneamente, mientras que durante periodos de menor actividad, los costos operativos se reducen automáticamente al mínimo. Esta elasticidad económica hace viable ofrecer el servicio de manera gratuita o a costo muy bajo para estudiantes, un factor crucial en regiones donde muchas familias tienen recursos económicos limitados.

La implementación de múltiples capas de seguridad para proteger información personal de estudiantes es especialmente importante en el contexto de país latinoamericano, donde las regulaciones de protección de datos están evolucionando rápidamente y la conciencia sobre privacidad digital está aumentando. El sistema demuestra que es posible construir servicios educativos que respetan y protegen activamente la privacidad de usuarios jóvenes, estableciendo un estándar que otras plataformas educativas en la región deberían aspirar a cumplir. La doble encriptación de información personal identificable va más allá de los requisitos legales mínimos en la mayoría de países latinoamericanos, proporcionando protección ejemplar que genera confianza tanto en estudiantes como en sus familias e instituciones educativas.

El algoritmo de recomendación personalizado basado en inteligencia artificial representa un avance significativo sobre los métodos tradicionales de orientación vocacional prevalentes en América Latina. Los tests vocacionales convencionales típicamente clasifican a estudiantes en categorías amplias (por ejemplo, orientado a ciencias u orientado a humanidades) mediante cuestionarios estandarizados que no consideran las particularidades individuales ni la evolución de intereses a lo largo del tiempo. El sistema implementado, en contraste, aprende continuamente de las interacciones del usuario con contenido educativo, refina sus recomendaciones basándose en patrones reales de comportamiento, y proporciona sugerencias cada vez más personalizadas a medida que acumula información sobre preferencias y fortalezas del estudiante.

La inclusión de componentes comunitarios como foros de discusión aborda otra limitación crítica de la orientación vocacional tradicional: el aislamiento del estudiante durante el proceso de toma de decisiones. Muchos estudiantes guatemaltecos, especialmente aquellos en comunidades rurales o de primera generación en acceder a educación superior, carecen de redes de contacto con personas que hayan cursado las carreras que les interesan. Los foros implementados en el sistema permiten que estudiantes interactúen con pares que están considerando carreras similares, compartan experiencias y preocupaciones, y aprendan de las vivencias de otros, creando una comunidad de apoyo virtual que complementa la orientación algorítmica con perspectivas humanas.

La implementación del sistema también tiene implicaciones económicas para las familias guatemaltecas. El costo de una mala elección de carrera universitaria es extraordinariamente alto en la región, donde las

matrículas universitarias representan frecuentemente una porción sustancial del ingreso familiar anual, y donde los sistemas de préstamos estudiantiles son limitados o inexistentes. Un estudiante que elige una carrera inadecuada y abandona los estudios después de uno o dos años no solo ha perdido el tiempo y la inversión económica directa, sino también el costo de oportunidad de haber podido estudiar una carrera más apropiada. Un sistema de orientación vocacional efectivo que reduzca la tasa de deserción universitaria por elección errónea de carrera tiene valor económico mensurable para las familias, justificando inversión pública o privada en su desarrollo y mantenimiento.

Desde una perspectiva de política pública, el prototipo demuestra la viabilidad técnica y económica de poder ser implementado en sistemas de orientación vocacional de Guatemala. Un ministerio de educación podría implementar una versión extendida de este sistema para todos los estudiantes de educación secundaria en el país a un costo que, incluso con decenas o cientos de miles de usuarios, permanecería una fracción minúscula del presupuesto educativo nacional. La arquitectura cloud hace que esta implementación sea factible incluso en países con infraestructura tecnológica limitada, como lo es Guatemala, ya que no requiere construcción de centros de datos o adquisición de hardware especializado.

El desarrollo del sistema integrador Mirai representa un logro técnico significativo en la aplicación de arquitecturas cloud modernas para resolver los desafíos educativos en Guatemala. La implementación de una arquitectura híbrida serverless, combinando 42 funciones AWS Lambda con un microservicio especializado en EC2 para procesamiento de inteligencia artificial, demostró ser una solución técnicamente viable y económicamente sostenible. Con un presupuesto operativo de apenas USD 42.02 mensuales durante la fase de desarrollo, el sistema logra proporcionar servicios sofisticados de orientación vocacional que tradicionalmente habrían requerido inversiones sustanciales en infraestructura física y personal técnico especializado.

Los resultados de las pruebas de rendimiento confirman que el sistema cumple y supera los objetivos establecidos. Con tiempos de respuesta promedio de 150-220 milisegundos para operaciones estándar, significativamente inferiores al objetivo de 500 milisegundos, y capacidad demostrada para manejar más de 2,800 usuarios concurrentes manteniendo tasas de error inferiores al 6 %, el sistema evidencia robustez operacional adecuada para escenarios de producción. Las pruebas de resistencia prolongada, con 88,606 solicitudes procesadas durante una hora con tasa de error del 0.33 %, confirman la estabilidad del sistema bajo cargas sostenidas y su ausencia de degradación progresiva de recursos, características críticas para una plataforma educativa que operará continuamente.

La implementación de seguridad multicapa constituye uno de los aspectos más destacados del proyecto. El sistema va más allá de los requisitos mínimos establecidos al implementar políticas de seguridad de autenticación, encriptación, mitigación contra ataques de denegación de servicio y políticas de seguridad de red privadas. Esta arquitectura implementa defensa en profundidad, confirma la inaccessibilidad de datos sensibles y establece un estándar de protección que cumple con regulaciones internacionales como GDPR Art. 32, Art. 25, OWASP Top 10, Ley 25.326 de Argentina Art. 9, y legislaciones Estadounidenses como NIST SP 800-111, 800-52 Rev.2

El cumplimiento de los objetivos específicos establecidos demuestra la metodología disciplinada seguida durante el desarrollo. El diseño arquitectónico se completó según lo planificado; el despliegue de funciones Lambda alcanza tiempos inferiores a un minuto, la escalabilidad horizontal validada mediante pruebas de estrés demostró soporte para 5,000+ solicitudes concurrentes antes de alcanzar límites de throttling, excediendo el objetivo de 2,000 usuarios; y el flujo seguro de datos con modelos de IA externos mediante VPC Link establece una comunicación segura entre el sistema integrador y el microservicio de recomendaciones. Estos logros técnicos confirman que el enfoque metodológico ágil adoptado, con iteraciones de dos semanas y validación continua de avances, permitió construcción incremental efectiva del sistema.

Desde una perspectiva de impacto educativo, el sistema representa una contribución significativa a la

democratización de servicios de orientación vocacional en Guatemala. La implementación de un algoritmo de recomendación personalizado basado en inteligencia artificial, que analiza continuamente las interacciones del usuario con contenido educativo y refina sus sugerencias basándose en patrones reales de comportamiento, supera cualitativamente los métodos tradicionales de orientación vocacional prevalentes en la región. Los componentes comunitarios implementados, particularmente los foros de discusión con capacidad de comentarios y respuestas anidadas, abordan el problema crítico del aislamiento durante el proceso de toma de decisiones que enfrentan muchos estudiantes latinoamericanos, especialmente aquellos en comunidades rurales o de primera generación en acceder a educación superior.

Las limitaciones identificadas durante el desarrollo, particularmente los cold starts de funciones Lambda con latencias máximas de hasta 11 segundos durante picos súbitos de tráfico, la ausencia de un sistema de caché para reducir consultas repetitivas a la base de datos, y la falta de degradación gradual cuando el sistema alcanza sus límites de capacidad, no representan fallas críticas sino oportunidades de optimización claramente definidas para implementación futura. La identificación del punto de quiebre del sistema alrededor de 4,000-5,000 solicitudes concurrentes proporciona información valiosa para planificación de capacidad, confirmando que este límite es configurable mediante ajustes de cuotas de Lambda y no representa una restricción fundamental del diseño. El trabajo futuro identificado, incluyendo implementación de ElastiCache, pipelines de CI/CD para el microservicio de recomendaciones, modelos de machine learning más sofisticados mediante AWS SageMaker, y sistemas de notificaciones en tiempo real mediante WebSockets, establece una ruta clara para evolución continua del sistema hacia una plataforma educativa cada vez más sofisticada y efectiva.

Optimizaciones de rendimiento y escalabilidad

- Implementar Amazon ElastiCache (Redis o Memcached) para almacenar en caché respuestas frecuentemente solicitadas como catálogos de carreras y listados de foros, reduciendo la carga en DocumentDB y mejorando tiempos de respuesta en 50-70 %.
- Configurar eventos programados de CloudWatch para invocar periódicamente las funciones Lambda críticas, manteniéndolas listas para funcionar, y así poder reducir la frecuencia de cold starts que pueden alcanzar latencias de hasta 11 segundos durante picos de tráfico.
- Migrar el microservicio de recomendaciones desde una única instancia EC2 hacia múltiples instancias detrás de un Application Load Balancer, proporcionando redundancia y capacidad de balanceo de carga para operaciones de inteligencia artificial intensivas.
- Implementar Amazon SQS (Simple Queue Service) para gestionar solicitudes que excedan la capacidad inmediata del sistema, proporcionando degradación gradual en lugar de errores 503 cuando se alcancen límites de concurrencia.

Mejoras en despliegue y mantenimiento

- Desarrollar un pipeline completo de CI/CD utilizando AWS CodePipeline, CodeBuild y CodeDeploy para automatizar el proceso de despliegue del microservicio de recomendaciones, eliminando intervención manual y permitiendo rollbacks automáticos en caso de errores.
- Configurar monitoreo avanzado con AWS CloudWatch Dashboards personalizados y Amazon CloudWatch Logs Insights para análisis en tiempo real de métricas de rendimiento, patrones de error y comportamiento de usuarios.
- Establecer alertas automáticas mediante Amazon SNS para notificar al equipo técnico cuando se detecten anomalías como tasas de error superiores al 10 %, latencias promedio superiores a 500ms, o tasas de throttling elevadas.
- Configurar el ambiente de despliegue para poder adaptar el concepto Cloud-Agnostic, para así poder desplegar el sistema integrador independientemente del proveedor cloud. Mediante herramientas como Terraform.

Expansión de funcionalidades y experiencia de usuario

- Implementar notificaciones push en tiempo real mediante AWS API Gateway WebSocket API o Amazon Simple Notification Service (SNS) para alertar a usuarios sobre respuestas a comentarios, nuevo contenido relevante y actualizaciones importantes.
- Desarrollar un sistema de analíticas avanzadas que proporcione a instituciones educativas dashboards con métricas agregadas sobre tendencias vocacionales, carreras más exploradas y patrones de interés por región geográfica.
- Integrar APIs de instituciones educativas para proporcionar información en tiempo real sobre fechas de inscripción, requisitos de admisión actualizados y disponibilidad de becas o ayuda financiera.
- Realizar pruebas de campo extensivas con usuarios reales en diferentes ubicaciones geográficas de América Latina (México, Colombia, Argentina, Chile, Perú, entre otros.) para medir latencias reales y experiencia de usuario en condiciones de red variables.

Apoyo para maestros y universidades

- Crear un programa de capacitación para orientadores vocacionales tradicionales sobre cómo incorporar el sistema como herramienta complementaria en sus procesos de asesoría, maximizando el impacto mediante enfoque híbrido de humano e Inteligencia Artificial.

- [1] EducoWay. *¿Qué es la orientación vocacional y para qué sirve?* Publicado: 23 noviembre 2021, Consultado: 2025-08-26. (Cap. 5). 2021. URL: <https://educoway.com/que-es-la-orientacion-vocacional-y-para-que-sirve/>.
- [2] Infobae. *La falta de orientación vocacional provoca desigualdad de oportunidades en América Latina*. Publicado: 13 diciembre 2024, Consultado: 2025-08-26. (Cap. 5). 2024. URL: <https://www.infobae.com/educacion/2024/12/13/la-falta-de-orientacion-vocacional-provoca-desigualdad-de-oportunidades-en-america-latina/>.
- [3] Utopía Urbana. *Un importante ranking aseguró que estas son las carreras con más profesionales arrepentidos*. Publicado: 2 marzo 2023, Consultado: 2025-08-26. (Cap. 5). 2023. URL: <https://utopiaurbana.city/2023/03/02/un-importante-ranking-aseguro-que-estas-son-las-carreras-con-mas-profesionales-arrepentidos/>.
- [4] Panamericana. *¿Cuál es la carrera en Perú que tiene la mayor cantidad de egresados arrepentidos? Esto dice la IA*. Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://panamericana.pe/tecnologia/409764-carrera-peru-mayor-cantidad-egresados-arrepentidos-dice-ia>.
- [5] El Confidencial. *Estas son las carreras universitarias con más licenciados arrepentidos 5 años después*. Publicado: 2 julio 2021, Consultado: 2025-08-26. (Cap. 5). 2021. URL: https://www.elconfidencial.com/alma-corazon-vida/educacion/2021-07-02/estas-con-las-carreras-con-mas-arrepentidos-un-lustro_3156608/.
- [6] EF Blog México. *¿Eres del 40 % de universitarios que se equivoca al elegir carrera?* Publicado: 8 octubre 2020, Consultado: 2025-08-26. (Cap. 5). 2020. URL: <https://www.ef.com.mx/blog/language/4-de-cada-10-universitarios-cambia-de-carrera-en-mexico/>.
- [7] Universidad de Toronto. *Six steps to create a personalized and effective study plan*. Consultado: 2025-08-26. (Cap. 5). s.f. URL: <https://www.utm.utoronto.ca/rgasc/student-resource-hub/study-skills-resources/six-steps-create-personalized-and-effective-study-plan>.
- [8] MyMap.ai. *Free AI study plan creator*. Consultado: 2025-08-26. (Cap. 5). s.f. URL: <https://www.mymap.ai/study-plan-creator>.
- [9] AWS. *¿Qué es la IA? - Explicación de la inteligencia artificial*. Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://aws.amazon.com/es/what-is/artificial-intelligence/>.
- [10] Google Cloud. *¿Qué es la inteligencia artificial o IA?* Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419>.

- [11] DataCamp. *¿Qué es la IA? Guía rápida para principiantes*. Publicado: Sep 11, 2024, Consultado: 2025-08-26. (Cap. 5). 2024. URL: <https://www.datacamp.com/es/blog/what-is-ai-quick-start-guide-for-beginners>.
- [12] Tableau. *¿Qué es la Inteligencia artificial? Definición, historia y aplicaciones*. Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://www.tableau.com/es-mx/data-insights/ai/what-is>.
- [13] Algolia. *What role does AI play in recommendation systems and engines?* Consultado: 2025-08-26. (Cap. 5). 2024. URL: <https://www.algolia.com/blog/ai/what-role-does-ai-play-in-recommendation-systems-and-engines>.
- [14] Psico-Smart. *¿Cuáles son las tendencias actuales en orientación vocacional en la era digital?* Publicado: 27 agosto 2024, Consultado: 2025-08-26. (Cap. 5). 2024. URL: <https://blogs-es.psico-smart.com/articulo-cuales-son-las-tendencias-actuales-en-orientacion-vocacional-en-la-era-digital-100914>.
- [15] El Comercio. *Educación: Primera plataforma de orientación vocacional en utilizar inteligencia artificial*. Consultado: 2025-08-26. (Cap. 5). 2021. URL: <https://elcomercio.pe/viu/educacion-primer-plataforma-de-orientacion-vocacional-en-utilizar-inteligencia-artificial-mndc-noticia/>.
- [16] Len Bass, Paul Clements y Rick Kazman. *Software Architecture in Practice*. 4th. (Cap. 5). Addison-Wesley, 2021. URL: <https://www.amazon.com/Software-Architecture-Practice-SEI-Engineering/dp/0136886094>.
- [17] Coursera. *What Does a Back-End Developer Do?* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.coursera.org/articles/back-end-developer>.
- [18] Built In. *What Is Back-End Development? (Definition, Features)*. Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://builtin.com/software-engineering-perspectives/back-end-development>.
- [19] Learn To Code With Me. *The Beginner's Guide to Backend Development (2024 Guide)*. Accessed: 2025-09-25. (Cap. 5). 2024. URL: <https://learntocodewith.me/posts/backend-development/>.
- [20] IBM. *What Is API Integration?* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.ibm.com/think/topics/api-integration>.
- [21] Cleo. *What is API Integration?* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.cleo.com/blog/what-is-api-integration>.
- [22] Refonte Learning. *AI and APIs: Integrating Artificial Intelligence into Your API Services*. Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.refontelearning.com/blog/ai-and-apis-integrating-artificial-intelligence-into-your-api-services>.
- [23] *¿Qué significa Cloud Computing? - Nube - Oracle*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.oracle.com/latam/cloud/what-is-cloud-computing/>.
- [24] *¿Qué es la computación en la nube? - Amazon AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://aws.amazon.com/es/what-is-cloud-computing/>.
- [25] *¿Qué es el cloud computing? - IBM*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.ibm.com/es-es/think/topics/cloud-computing>.
- [26] *¿Qué es el cloud computing? - Akamai*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.akamai.com/es/glossary/what-is-cloud-computing>.
- [27] *¿Qué es Cloud Computing? Definición y aplicaciones - Salesforce MX*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.salesforce.com/mx/cloud-computing/>.
- [28] *Qué es el Cloud Computing: tipos y ventajas - Telefónica*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.telefonica.com/es/sala-comunicacion/blog/que-es-el-cloud-computing-tipos-y-ventajas/>.
- [29] *¿Qué es la computación en la nube? Visión general de la nube*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.atlassian.com/es/microservices/cloud-computing>.

- [30] *Qué es el cloud computing? / Tipos, ejemplos, servicios...* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.plainconcepts.com/es/que-es-cloud-computing/>.
- [31] *21+ Top Cloud Service Providers Globally In 2025 - CloudZero.* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.cloudzero.com/blog/cloud-service-providers/>.
- [32] *55 Cloud Computing Statistics for 2025 - Spacelift.* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://spacelift.io/blog/cloud-computing-statistics>.
- [33] *Best Strategic Cloud Platform Services Reviews 2025 - Gartner.* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.gartner.com/reviews/market/strategic-cloud-platform-services>.
- [34] *Top 9 Cloud Service Providers in 2025 - ProsperOps.* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.prosperops.com/blog/top-cloud-providers/>.
- [35] *Los 8 mejores proveedores Cloud del mercado - CloudAPPi.* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://cloudappi.net/mejores-proveedores-cloud/>.
- [36] *Cloud Computing en 2025: La guía definitiva para dominarla.* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://immune.institute/blog/cloud-computing-en-2025-la-guia-definitiva/>.
- [37] Solo.io. *What Is an API Gateway? How It Works & Why You Need One.* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.solo.io/topics/api-gateway>.
- [38] Kong HQ. *What is an API Gateway? Core Fundamentals and Use Cases.* Accessed: 2025-09-25. (Cap. 5). 2023. URL: <https://konghq.com/blog/learning-center/what-is-an-api-gateway>.
- [39] IBM. *What Is an API Gateway?* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.ibm.com/think/topics/api-gateway>.
- [40] Clarifai. *Horizontal vs Vertical Scaling | Which Strategy Fits Your AI Workloads?* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.clarifai.com/blog/horizontal-vs-vertical-scaling>.
- [41] InfraCloud. *How to Build Scalable AI Systems in the Cloud.* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.infracloud.io/blogs/build-scalable-ai-systems-in-cloud/>.
- [42] Spiceworks. *What Is Horizontal Cloud Scaling? Definition, Process, and Best ...* Accessed: 2025-09-25. (Cap. 5). 2021. URL: <https://www.spiceworks.com/tech/cloud/articles/horizontal-cloud-scaling/>.
- [43] Brighteye Ventures. *AI's growing role in Vocational Education.* Accessed: 2025-09-25. (Cap. 5). 2024. URL: <https://www.brighteyevc.com/post/ai-s-growing-role-in-vocational-education>.
- [44] MDPI. "AI-Powered Academic Guidance and Counseling System Based on ..." En: *Applied System Innovation* (2023). Accessed: 2025-09-25. (Cap. 5). URL: <https://www.mdpi.com/2571-5577/7/1/6>.
- [45] FutureFit AI. *FutureFit AI: AI-Powered Workforce Solutions.* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://www.futurefit.ai/>.
- [46] MDPI. "Design of Intelligent Management Platform for Industry-Education ..." En: (2022). Accessed: 2025-09-25. (Cap. 5). URL: <https://www.mdpi.com/2076-3417/12/14/6836>.
- [47] Enterprise Architecture. *The Role of the Enterprise Architect in Application Systems Integration.* Accessed: 2025-09-25. (Cap. 5). 2025. URL: <https://enterprise-architecture.org/about/thought-leadership/the-role-of-the-enterprise-architect-in-application-systems-integration/>.
- [48] Vega IT. *The secrets of effective system integration architecture.* Accessed: 2025-09-25. (Cap. 5). 2024. URL: <https://www.vegaitglobal.com/media-center/business-insights/the-secrets-of-effective-system-integration-architecture>.
- [49] Google Cloud. *¿Qué es la arquitectura de nube? Beneficios y componentes.* Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://cloud.google.com/learn/what-is-cloud-architecture?hl=es-419>.
- [50] IBM. *¿Qué es la arquitectura en la nube?* Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://www.ibm.com/mx-es/think/topics/cloud-architecture>.

- [51] Powernet. *Cloud vs. Legacy: ¿Arquitecturas tradicionales o basadas en la nube?* Publicado: Nov 16, 2023, Consultado: 2025-08-26. (Cap. 5). 2023. URL: <https://www.powernet.es/blog/cloud-vs-legacy-arquitecturas-tradicionales-o-basadas-en-la-nube>.
- [52] Right People Group. *Infraestructura en nube frente a infraestructura informática tradicional*. Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://rightpeoplegroup.com/es/blog/infraestructura-en-nube-frente-a-infraestructura-informatica-tradicional>.
- [53] ORSYS. *Arquitecturas sin servidor: ¿por dónde empezar?* Publicado: May 28, 2025, Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://www.orsys.fr/orsys-lemag/es/arquitecturas-sin-servidor-por-donde-empezar/>.
- [54] Meeran. *Microservices Architecture for AI Applications: Scalable Patterns and 2025 Trends*, note = (Cap. 5). Publicado: May 1, 2025, Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://medium.com/@meeran03/microservices-architecture-for-ai-applications-scalable-patterns-and-2025-trends-5ac273eac232>.
- [55] Khan Mudassir. *Microservices Architecture: The Future of Scalable Web and AI Applications*, note = (Cap. 5). Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://medium.com/@khanmudassir124/microservices-architecture-the-future-of-scalable-web-and-ai-applications-285786bca4fc>.
- [56] AWS. *Building serverless architectures for agentic AI on AWS*. Publicado: Jul 29, 2025, Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-serverless/introduction.html>.
- [57] Crunch. *How to Build Cloud-Agnostic AI Agents: Architecture for AWS, Azure, Google Cloud*. Publicado: Jul 3, 2025, Consultado: 2025-08-26. (Cap. 5). 2025. URL: <https://crunch.is/blog/how-to-build-cloud-agnostic-ai-agents-architecture-for-aws-azure-google-cloud/>.
- [58] CloudThat. *Kubernetes Deployment on Cloud Agnostic*. Publicado: Mar 20, 2023, Consultado: 2025-08-26. (Cap. 5). 2023. URL: <https://www.cloudthat.com/resources/blog/kubernetes-deployment-on-cloud-agnostic>.
- [59] Palo Alto Networks. *Multicloud Management with AI and Kubernetes*. Consultado: 2025-08-26. (Cap. 5). s.f. URL: <https://www.paloaltonetworks.com/cyberpedia/kubernetes-multicloud-management>.
- [60] Universitat Oberta de Catalunya. *Arquitectura y diseño de seguridad de aplicaciones en la nube pública*. Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://openaccess.uoc.edu/server/api/core/bitstreams/bed7cd64-0767-43f6-a3ac-5ce5b7d34bba/content>.
- [61] Computerworld. *Las siete principales amenazas a la seguridad en la nube y cómo abordarlas*. Publicado: Agosto 5, 2024, Consultado: 2025-09-03. (Cap. 5). 2024. URL: <https://www.computerworld.es/article/3481264/las-siete-principales-amenazas-a-la-seguridad-en-la-nube-y-como-abordarlas.html>.
- [62] Fortinet. *10 consejos para superar los riesgos de seguridad en la nube pública*. Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://www.fortinet.com/lat/resources/cyberglossary/public-cloud-security-risks>.
- [63] Apidog. *9 métodos populares de autenticación de API para proteger las API*. Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://apidog.com/es/blog/api-authentication-methods-5/>.
- [64] Google Cloud. *Métodos de autenticación en Google*. Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://cloud.google.com/docs/authentication?hl=es-419>.
- [65] Palo Alto Networks. *¿Qué es la seguridad de las API?* Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://www.paloaltonetworks.es/cyberpedia/what-is-api-security>.
- [66] Safetica. *Seguridad de datos en la nube: definiciones, riesgos y 7 mejores prácticas*. Publicado: Marzo 11, 2024, Consultado: 2025-09-03. (Cap. 5). 2024. URL: <https://www.safetica.com/es/recursos/blogs/seguridad-de-datos-en-la-nube-definiciones-riesgos-y-7-mejores-practicas-para-la-proteccion-de-datos-en-la-nube>.

- [67] EY. *Protección de Datos Personales en LATAM: Guía de Consulta Rápida*. Consultado: 2025-09-03. (Cap. 5). 2023. URL: https://www.ey.com/es_ce/insights/law/proteccion-de-datos-personales-en-latam.
- [68] ClarkeModet. *Protección de datos en los países de Latinoamérica*. Publicado: Junio 19, 2023, Consultado: 2025-09-03. (Cap. 5). 2023. URL: <https://www.clarkemodet.com/articulos/proteccion-de-datos-en-los-paises-de-latinoamerica/>.
- [69] WeLiveSecurity. *Panorama y tendencias legislativas sobre la Protección de Datos en LATAM*. Publicado: Octubre 3, 2023, Consultado: 2025-09-03. (Cap. 5). 2023. URL: <https://www.welivesecurity.com/es/privacidad/panorama-proteccion-datos-paises-latam/>.
- [70] Google Cloud. *¿Qué es el encriptado y cómo funciona?* Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://cloud.google.com/learn/what-is-encryption?hl=es>.
- [71] CloudMounter. *¿Qué es el cifrado en la nube?* Publicado: Agosto 11, 2025, Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://cloudmounter.net/es/what-is-cloud-encryption/>.
- [72] Cloudflare. *¿Por qué es importante la encriptación para la privacidad?* Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://www.cloudflare.com/es-es/learning/privacy/encryption-and-privacy/>.
- [73] Auditool. *Auditoría de la nube: Principios esenciales y mejores prácticas*. Publicado: Junio 13, 2024, Consultado: 2025-09-03. (Cap. 5). 2024. URL: <https://www.auditool.org/blog/auditoria-de-ti/auditoria-de-la-nube-principios-esenciales-y-mejores-practicas>.
- [74] ISACA. *Security Assurance in the SDLC for the Internet of Things*. Consultado: 2025-09-03. (Cap. 5). 2017. URL: <https://www.isaca.org/es-es/resources/isaca-journal/issues/2017/volume-3/security-assurance-in-the-sdlc-for-the-internet-of-things>.
- [75] Zscaler. *¿Qué es la seguridad en la nube?* Consultado: 2025-09-03. (Cap. 5). 2025. URL: <https://www.zscaler.com/es/resources/security-terms-glossary/what-is-cloud-security>.
- [76] *What is Amazon VPC? - Amazon Virtual Private Cloud*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>.
- [77] *Virtual Private Cloud (VPC) - Oracle Cloud Infrastructure*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.oracle.com/cloud/networking/virtual-private-cloud/>.
- [78] *What is a virtual private cloud (VPC)? - Cloudflare*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.cloudflare.com/learning/cloud/what-is-a-virtual-private-cloud/>.
- [79] *What is VPC? - IBM*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.ibm.com/topics/vpc>.
- [80] *Networking fundamentals for VPC*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-networking.html>.
- [81] *What is Azure Virtual Network? - Microsoft Docs*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview>.
- [82] *Virtual Private Cloud (VPC) - Google Cloud*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://cloud.google.com/vpc/docs/vpc>.
- [83] *VPC sharing: key considerations and best practices - AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://aws.amazon.com/blogs/networking-and-content-delivery/vpc-sharing-key-considerations-and-best-practices/>.
- [84] *Virtual Cloud Network Overview - Oracle*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/overview.htm>.
- [85] *IBM Cloud VPC Architecture*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.ibm.com/docs/en/cloud-paks/cp-management/2.3.x?topic=overview-ibm-cloud-vpc-architecture>.
- [86] *Virtual private cloud (VPC) benefits - Cloudflare*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.cloudflare.com/learning/cloud/virtual-private-cloud-vpc-benefits/>.
- [87] *Security in Amazon Virtual Private Cloud - AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/vpc/latest/userguide/security.html>.

- [88] *Azure Virtual Network security - Microsoft Docs*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://learn.microsoft.com/en-us/azure/virtual-network/security-overview>.
- [89] *¿Qué es WAF firewall? | Cortafuegos de aplicaciones web - Cloudflare*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.cloudflare.com/es-es/learning/ddos/glossary/web-application-firewall-waf/>.
- [90] *¿Qué es un WAF (firewall de aplicaciones web)? - Oracle*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.oracle.com/latam/security/cloud-security/what-is-waf/>.
- [91] *¿Qué es WAF? Definición y explicación del Firewall de aplicaciones ... - Fortinet*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.fortinet.com/lat/resources/cyberglossary/waf>.
- [92] *What Is A WAF? 2025 Guide to Web Application Firewalls - Radware*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://es.radware.com/cyberpedia/application-security/what-is-waf/>.
- [93] *¿Qué es un firewall de aplicaciones web (WAF)? - Akamai*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.akamai.com/es/glossary/what-is-a-waf>.
- [94] *Web Application Firewall (WAF): qué es y cómo protege tu sitio web*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.cdmon.com/es/blog/la-importancia-de-utilizar-un-web-application-firewall>.
- [95] *¿Qué es un WAF (Web Application Firewall ... - Ciphersafety*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://ciphersafety.com/que-es-un-waf-firewall-aplicacion-web/>.
- [96] *¿Qué es un firewall de aplicaciones web (WAF)? - Fastly*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.fastly.com/es/learning/security/what-is-a-web-application-firewall-waf-explained>.
- [97] *¿Qué es un WAF?, ¿Necesita un WAF mi empresa?* Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://advance-nt.com/2023/05/06/que-es-un-waf-necesita-un-waf-mi-empresa/>.
- [98] *Amazon DocumentDB (with MongoDB compatibility)*. Accessed: 2025-10-21. (Cap. 5). 2025. URL: <https://aws.amazon.com/documentdb/>.
- [99] *Security best practices for Amazon EMR Serverless*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/emr/latest/EMR-Serverless-UserGuide/security-best-practices.html>.
- [100] *Best Practices for Serverless Technologies in AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://builder.aws.com/content/2pYmkuLReVaqZ29ew1P3Dn4iAvH/best-practices-for-serverless-technologies-in-aws>.
- [101] *Use AWS WAF to protect your REST APIs in API Gateway*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-aws-waf.html>.
- [102] *AWS WAF rules*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/waf/latest/developerguide/waf-rules.html>.
- [103] *The Ultimate API Security Duo: AWS API Gateway and WAF in Action*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://medium.com/codex/the-ultimate-api-security-duo-aws-api-gateway-and-waf-in-action-7e8cc16152f5>.
- [104] *Control access to HTTP APIs with JWT authorizers in API Gateway*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-jwt-authorizer.html>.
- [105] *Securing API Gateway with Lambda Authorizer Using JWT Tokens*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://dev.to/aws-builders/securing-api-gateway-with-lambda-authorizer-using-jwt-tokens-aop>.
- [106] *Automatically rotate IAM user access keys at scale with AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/automatically-rotate-iam-user-access-keys-at-scale-with-aws-organizations-and-aws-secrets-manager.html>.

- [107] *access-keys-rotated - AWS Config*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/config/latest/developerguide/access-keys-rotated.html>.
- [108] *Security best practices for your VPC*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-best-practices.html>.
- [109] *Control traffic to your AWS resources using security groups*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html>.
- [110] *AWS Security Group: Best Practices & Instructions*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.corestack.io/aws-security-best-practices/aws-security-group-best-practices/>.
- [111] *Networking in AWS: VPCs, Subnets, and Security Groups*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://medium.com/%40venkatramankanntech/networking-in-aws-vpcs-subnets-and-security-groups-f6f0c694b9ab>.
- [112] *AWS VPC Security: 13 Best Practices*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.hyperglance.com/blog/aws-vpc-security-best-practices/>.
- [113] *Understanding VPC Security Groups: A Comprehensive Guide*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.tufin.com/blog/understanding-vpc-security-groups-comprehensive-guide>.
- [114] *Encrypting Data-at-Rest and Data-in-Transit*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/whitepapers/latest/logical-separation/encrypting-data-at-rest-and-in-transit.html>.
- [115] *Encryption at rest in Amazon Connect*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/connect/latest/adminguide/encryption-at-rest.html>.
- [116] *Securing Personally Identifiable Information (PII) in AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://robcmorris.medium.com/securing-personally-identifiable-information-pii-in-aws-c1f3a42d2ed7>.
- [117] *How to protect sensitive data for its entire lifecycle in AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://aws.amazon.com/blogs/security/how-to-protect-sensitive-data-for-its-entire-lifecycle-in-aws/>.
- [118] *Data encryption - AWS Health*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/health/latest/ug/data-encryption.html>.
- [119] *EMR In-Transit and At-Rest Encryption*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://trendmicro.com/cloudoneconformity/knowledge-base/aws/EMR/in-transit-and-at-rest-encryption.html>.
- [120] *Serverless Security: Risks and Best Practices*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.sysdig.com/learn-cloud-native/serverless-security-risks-and-best-practices>.
- [121] *Serverless Security Best Practices*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.jit.io/blog/serverless-security-best-practices>.
- [122] *What's the Best Practices of Serverless Security on AWS 2025*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://cloudautocraft.com/whats-the-best-practices-of-serverless-security-on-aws/>.
- [123] *Securing Serverless Applications on AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://pcg.io/insights/securing-serverless-apps-on-aws/>.
- [124] *Data protection laws in Guatemala*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.dlapiperdataprotection.com/index.html?t=law%5C&c=GT>.
- [125] *Questions and answers on data protection in Central America*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://consortiumlegal.com/en/2022/11/02/questions-and-answers-on-data-protection-in-central-america/>.

- [126] *Brazil Data Privacy - Amazon Web Services (AWS)*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://aws.amazon.com/compliance/brazil-data-privacy/>.
- [127] *Brazil General Data Protection Law Workbook*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: https://d1.awsstatic.com/whitepapers/compliance/Brazil_General_Data_Protection_Law_Workbook_English_2022.pdf.
- [128] *General Data Protection Regulation (GDPR) Center*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://aws.amazon.com/compliance/gdpr-center/>.
- [129] *Navigating GDPR Compliance on AWS*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/whitepapers/latest/navigating-gdpr-compliance/welcome.html>.
- [130] *SOC 2, GDPR, HIPAA Compliance on AWS: A Complete Guide*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.novelvista.com/blogs/cloud-and-aws/soc-2-gdpr-hipaa-compliance-on-aws-a-complete-guide>.
- [131] *Guatemala Cybersecurity*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://www.trade.gov/market-intelligence/guatemala-cybersecurity>.
- [132] *New regulations on cybersecurity and data protection*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://altalegal.com/en/comunicacion/nuevas-normativas-en-materia-de-ciberseguridad-y-proteccion-de-datos/>.
- [133] *Data Protection Law in South America*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: <https://securiti.ai/blog/data-protection-law-in-south-america/>.
- [134] *Security Incidents and Cookies in Latin America Countries*. Accessed: 2025-11-04. (Cap. 5). 2025. URL: https://www.inta.org/wp-content/uploads/public-files/advocacy/committee-reports/20231205_Security-Incidents-and-Cookies-LatAm-Survey_Dec-2023_FINAL.pdf.
- [135] *21+ Top Cloud Service Providers Globally In 2025 - CloudZero*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.cloudzero.com/blog/cloud-service-providers/>.
- [136] *Top Cloud Service Providers 2025: AWS vs Azure vs GCP/EC-Council*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.eccouncil.org/cybersecurity-exchange/cloud-security/biggest-cloud-service-providers/>.
- [137] *Cloud Pricing Comparison: AWS vs. Azure vs. Google in 2025*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://cast.ai/blog/cloud-pricing-comparison/>.
- [138] *AWS vs. Azure vs. Google Cloud: Which is Best for You in 2025?* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://amasty.com/blog/choosing-the-right-cloud-platform/>.
- [139] *Compare AWS and Azure services to Google Cloud*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://docs.cloud.google.com/docs/get-started/aws-azure-gcp-service-comparison>.
- [140] *Cloud Providers: AWS, Azure, GCP - Dataquest*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.dataquest.io/blog/cloud-providers-aws-azure-gcp/>.
- [141] *AWS vs Azure vs Google Cloud: comprehensive comparison for 2025*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://northflank.com/blog/aws-vs-azure-vs-google-cloud>.
- [142] *What are the top cloud service providers globally in 2025 and how ...* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.webasha.com/blog/what-are-the-top-cloud-service-providers-globally-and-how-do-they-compare>.
- [143] *Technology | 2025 Stack Overflow Developer Survey*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://survey.stackoverflow.co/2025/technology>.
- [144] *The 100 Top Programming Languages in 2025 - BairesDev*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.bairesdev.com/blog/top-programming-languages/>.
- [145] *10 Best Cloud Computing Programming Languages - Index.dev*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.index.dev/blog/top-cloud-computing-languages>.
- [146] *Top 10 Future Programming Languages for 2025 - Crossover*. Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.crossover.com/resources/future-programming-languages-for-2025>.

- [147] *After Node.js, which language should I pick for backend development?* Accessed: 2025-11-05. (Cap. 5). 2025. URL: https://www.reddit.com/r/node/comments/1jcfdlp/after_nodejs_which_language_should_i_pick_for/.
- [148] *Best Programming Languages for Cloud Enterprise Development.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://appvertices.io/best-programming-languages-cloud-computing/>.
- [149] *Best programming languages & frameworks to learn in 2025.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://javascript.plainenglish.io/best-programming-languages-frameworks-to-learn-in-2025-d209f6b50b8c>.
- [150] *Comparison of popular programming languages in 2025-2030.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://rubyroidlabs.com/blog/2025/10/most-popular-programming-languages/>.
- [151] *Computer Language Rankings 2025: The Most Popular ...* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://business.daily.dev/resources/computer-language-rankings-the-most-popular-programming-languages>.
- [152] *¿Qué Es NoSQL? Descripción De Las Bases De Datos ... - MongoDB.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.mongodb.com/es/resources/basics/databases/nosql-explained>.
- [153] *¿Qué es MongoDB? - IBM.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.ibm.com/es-es/think/topics/mongodb>.
- [154] *¿Qué es MongoDB? Guía avanzada | Oracle América Latina.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.oracle.com/ar/database/mongodb/>.
- [155] *MongoDB - Wikipedia, la enciclopedia libre.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://es.wikipedia.org/wiki/MongoDB>.
- [156] *¿Qué es MongoDB y cómo funciona? - Pure Storage.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.purestorage.com/es/knowledge/what-is-mongodb.html>.
- [157] *MongoDB: todo sobre la base de datos NoSQL orientada a ...* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://datascientest.com/es/mongodb-todo-sobre-la-base-de-datos-nosql-orientada-a-documentos>.
- [158] *¿Qué Es MongoDB? Todo Sobre la Popular Base de Datos de ...* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://kinsta.com/es/blog/que-es-mongodb/>.
- [159] *Bases De Datos NoSQL Vs SQL - MongoDB.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://www.mongodb.com/es/resources/basics/databases/nosql-explained/nosql-vs-sql>.
- [160] *¿Qué es una base de datos NoSQL? - Amazon AWS.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://aws.amazon.com/es/nosql/>.
- [161] *Si MongoDB, siendo una base de datos NoSQL, es compatible con ...* Accessed: 2025-11-05. (Cap. 5). 2025. URL: https://www.reddit.com/r/Database/comments/18f3o16/if_mongodb_being_a_nosql_database_is/?tl=es-es.
- [162] *Qué es Amazon DocumentDB (compatible con MongoDB).* Accessed: 2025-11-05. (Cap. 5). 2025. URL: https://docs.aws.amazon.com/es_es/documentdb/latest/developerguide/what-is.html.
- [163] *Amazon DocumentDB: how it works.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: <https://docs.aws.amazon.com/documentdb/latest/developerguide/how-it-works.html>.
- [164] *Introducción a Amazon DocumentDB.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: https://docs.aws.amazon.com/es_es/documentdb/latest/developerguide/get-started-guide.html.
- [165] *Descripción de clúster - Amazon DocumentDB.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: https://docs.aws.amazon.com/es_es/documentdb/latest/developerguide/db-clusters-understanding.html.
- [166] *Uso de Amazon DocumentDB sin servidor.* Accessed: 2025-11-05. (Cap. 5). 2025. URL: https://docs.aws.amazon.com/es_es/documentdb/latest/developerguide/docdb-serverless.html.

12.1. Introducción al esquema de la base de datos

Este documento presenta el esquema completo de la base de datos MongoDB para la plataforma MIRAI, un sistema integral de orientación vocacional. La arquitectura de datos está diseñada para soportar funcionalidades de evaluación psicométrica, recomendación de carreras, gestión de contenido educativo, foros de discusión, y análisis de interacciones de usuarios.

12.1.1. Tecnologías utilizadas

- **Base de Datos:** MongoDB (NoSQL)
- **ODM:** Mongoose
- **Autenticación:** Clerk

12.1.2. Colecciones de la base de datos

El sistema está compuesto por 12 colecciones principales:

1. **users** - Gestión de usuarios
2. **careers** - Catálogo de carreras universitarias
3. **courses** - Cursos académicos
4. **tags** - Sistema de etiquetado
5. **cards** - Tarjetas de contenido en el feed
6. **forums** - Foros de discusión
7. **interactions** - Registro de interacciones de usuarios
8. **saveditems** - Items guardados por usuarios

9. `test_items` - Preguntas del cuestionario vocacional
10. `test_results` - Resultados de evaluaciones
11. `testimonies` - Testimonios de estudiantes/egresados
12. `scoring_rules` - Reglas de puntuación para evaluaciones

12.2. Colecciones principales de la base de datos

12.2.1. Colección: `users`

Almacena la información de todos los usuarios de la plataforma.

Estructura del documento

Cuadro 12.1. Estructura de la colección `users`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único del usuario
<code>clerk_id</code>	String	Sí	ID de autenticación de Clerk (único)
<code>created_at</code>	Date	Sí	Fecha de creación del usuario
<code>role</code>	String	Sí	Rol del usuario (enum)
<code>image_url</code>	String	No	URL de la imagen de perfil
<code>first_name</code>	String	No	Nombre del usuario
<code>last_name</code>	String	No	Apellido del usuario
<code>username</code>	String	No	Nombre de usuario
<code>email</code>	String	No	Correo electrónico
<code>quizCompletedAt</code>	Date	No	Fecha de completado del cuestionario
<code>user_tags</code>	Array	No	Etiquetas del perfil del usuario

Valores de enumeración

`role:`

- `admin` - Administrador del sistema
- `teacher` - Docente
- `director` - Director/Coordinador
- `publisher` - Publicador de contenido
- `student` - Estudiante (por defecto)

Subdocumento: user_tags

Cuadro 12.2. Subdocumento **user_tags**

Campo	Tipo	Requerido	Descripción
tag	ObjectId	Sí	Referencia a Tag
name	String	Sí	Nombre de la etiqueta
score	Number	Sí	Puntuación del usuario en esta etiqueta

12.2.2. Colección: careers

Contiene el catálogo completo de carreras universitarias con toda su información académica y laboral.

Estructura del documento

Cuadro 12.3. Estructura de la colección **careers**

Campo	Tipo	Requerido	Descripción
_id	ObjectId	Sí	Identificador único de la carrera
nombre_carrera	String	Sí	Nombre oficial de la carrera
facultad	String	Sí	Facultad a la que pertenece
descripcion	String	Sí	Descripción detallada de la carrera
duracion	Number	Sí	Duración en años
empleabilidad	String	Sí	Nivel de empleabilidad (enum)
areas_de_desarrollo_potencial	Array	No	Áreas de desarrollo profesional
plan_de_estudio	Object	No	Plan de estudios por años y semestres
areas_de_formacion	Array	No	Áreas de formación académica
perfil_del_egresado	String	Sí	Descripción del perfil del egresado
competencias_desarrolladas	Array	No	Lista de competencias
salario_minimo	Number	Sí	Salario mínimo estimado
salario_maximo	Number	Sí	Salario máximo estimado
moneda_salario	String	Sí	Moneda (default: USD)
tags	Array	No	Etiquetas asociadas a la carrera
insights	Object	No	Datos analíticos y métricas
createdAt	Date	Sí	Fecha de creación (automático)
updatedAt	Date	Sí	Fecha de actualización (automático)

Valores de enumeración

empleabilidad:

- baja
- media
- alta
- muy alta

Subdocumento: `areas_de_desarrollo_potencial`

Cuadro 12.4. Subdocumento `areas_de_desarrollo_potencial`

Campo	Tipo	Requerido	Descripción
<code>area</code>	String	Sí	Nombre del área
<code>descripcion</code>	String	Sí	Descripción del área

Subdocumento: `areas_de_formacion`

Cuadro 12.5. Subdocumento `areas_de_formacion`

Campo	Tipo	Requerido	Descripción
<code>area</code>	String	Sí	Nombre del área
<code>descripcion</code>	String	Sí	Descripción del área

Objeto: `plan_de_estudio`

El plan de estudios está estructurado jerárquicamente por años (1-5) y semestres (primero/segundo):

Listing 12.1. Estructura del plan de estudios

```
{
  "a_o_1": {
    "primer_semestre": [
      { "id": ObjectId, "name": String }
    ],
    "segundo_semestre": [
      { "id": ObjectId, "name": String }
    ]
  },
  "a_o_2": { ... },
  "a_o_3": { ... },
  "a_o_4": { ... },
  "a_o_5": { ... }
}
```

Subdocumento: `tags`

Cuadro 12.6. Subdocumento `tags` en `careers`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	ID del subdocumento
<code>tag</code>	ObjectId	Sí	Referencia a Tag
<code>name</code>	String	Sí	Nombre de la etiqueta
<code>score</code>	Number	Sí	Peso/relevancia (0-1)

12.2.3. Colección: `courses`

Catálogo de cursos académicos con sus prerrequisitos.

Estructura del documento

Cuadro 12.7. Estructura de la colección `courses`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único del curso
<code>nombre</code>	String	Sí	Nombre del curso
<code>prerequisitos</code>	Array	No	Array de ObjectIds de cursos

12.2.4. Colección: `tags`

Sistema de etiquetado para clasificación de carreras y contenido.

Estructura del documento

Cuadro 12.8. Estructura de la colección `tags`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único del tag
<code>name</code>	String	Sí	Nombre de la etiqueta
<code>description</code>	String	No	Descripción de la etiqueta

12.2.5. Colección: `cards`

Tarjetas de contenido que aparecen en el feed principal de exploración.

Estructura del documento

Cuadro 12.9. Estructura de la colección `cards`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único de la tarjeta
<code>type</code>	String	Sí	Tipo de tarjeta (enum)
<code>title</code>	String	Sí	Título de la tarjeta
<code>content</code>	String	Sí	Contenido de la tarjeta
<code>tags</code>	Array	No	Array de etiquetas (Object/String)
<code>priority</code>	Number	No	Prioridad de visualización
<code>created_at</code>	Date	Sí	Fecha de creación
<code>color</code>	String	No	Color de la tarjeta (hex)
<code>display_data</code>	Object	No	Datos adicionales para display

Valores de enumeración

`type`:

- `career` - Tarjeta de carrera
- `what_if` - Tarjeta de escenario hipotético

- **question** - Tarjeta de pregunta/reflexión
- **testimony** - Tarjeta de testimonio

12.2.6. Colección: forums

Foros de discusión asociados a carreras específicas.

Estructura del documento

Cuadro 12.10. Estructura de la colección **forums**

Campo	Tipo	Requerido	Descripción
_id	ObjectId	Sí	Identificador único del foro
title	String	Sí	Título del foro
description	String	Sí	Descripción del foro
creator_id	ObjectId	Sí	Referencia al usuario creador
career_id	ObjectId	Sí	Referencia a la carrera
comments	Array	No	Array de comentarios (embebidos)
created_at	Date	Sí	Fecha de creación
final_date	Date	Sí	Fecha de cierre del foro

Subdocumento: comments

Cuadro 12.11. Subdocumento **comments**

Campo	Tipo	Requerido	Descripción
_id	ObjectId	Sí	ID del comentario
user_id	ObjectId	Sí	Referencia al usuario
content	String	Sí	Contenido del comentario
created_at	Date	Sí	Fecha de creación
answers	Array	No	Array de respuestas (embebidas)

Subdocumento: answers

Cuadro 12.12. Subdocumento **answers** dentro de comments

Campo	Tipo	Requerido	Descripción
_id	ObjectId	Sí	ID de la respuesta
user_id	ObjectId	Sí	Referencia al usuario
content	String	Sí	Contenido de la respuesta
created_at	Date	Sí	Fecha de creación

12.2.7. Colección: interactions

Registro de todas las interacciones de usuarios con tarjetas y contenido.

Estructura del documento

Cuadro 12.13. Estructura de la colección **interactions**

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único de la interacción
<code>cardId</code>	String	Sí	ID de la tarjeta interactuada
<code>action</code>	String	Sí	Tipo de acción (enum)
<code>duration</code>	Number	No	Duración de la interacción (ms)
<code>metadata</code>	Object	No	Metadatos adicionales
<code>created_at</code>	Date	Sí	Fecha de la interacción

Valores de enumeración

action:

- **view** - Visualización de tarjeta
- **tap** - Tap/clic en tarjeta
- **save** - Guardar tarjeta
- **share** - Compartir tarjeta

12.2.8. Colección: **saveditems**

Items guardados por los usuarios (carreras o tarjetas).

Estructura del documento

Cuadro 12.14. Estructura de la colección **saveditems**

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único
<code>user_id</code>	String	Sí	Referencia al usuario
<code>type</code>	String	Sí	Tipo de item guardado (enum)
<code>item_id</code>	ObjectId	Sí	ID del item guardado
<code>saved_at</code>	Date	Sí	Fecha en que se guardó
<code>item</code>	Object	Sí	Copia desnormalizada del item

Valores de enumeración

type:

- **career** - Carrera guardada
- **card** - Tarjeta guardada

Nota: Esta colección utiliza desnormalización para mantener una copia del item al momento de guardarlo.

12.2.9. Colección: test_items

Items (preguntas) del cuestionario de orientación vocacional.

Estructura del documento

Cuadro 12.15. Estructura de la colección `test_items`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único
<code>id</code>	String	Sí	ID legible del item (ej: bfi_2)
<code>section</code>	String	Sí	Sección del test
<code>type</code>	String	Sí	Tipo de pregunta
<code>prompt</code>	String	Sí	Texto de la pregunta
<code>options</code>	Array	Sí	Opciones de respuesta
<code>reverse</code>	Boolean	Sí	Si la pregunta es reversa
<code>meta</code>	Object	No	Metadatos adicionales

Secciones del test

El cuestionario está dividido en las siguientes secciones:

- `bfi` - Big Five Inventory (personalidad)
- `riasec` - Holland Code (intereses vocacionales)
- `piaac` - Habilidades cognitivas
- `grit` - Perseverancia y consistencia
- `paa` - Prueba de aptitud académica

Subdocumento: options

Cuadro 12.16. Subdocumento `options`

Campo	Tipo	Requerido	Descripción
<code>id</code>	String	Sí	Identificador de la opción
<code>label</code>	String	Sí	Texto de la opción
<code>value</code>	Number	Sí	Valor numérico de la opción

12.2.10. Colección: test_results

Resultados de las evaluaciones vocacionales completadas por usuarios.

Estructura del documento

Cuadro 12.17. Estructura de la colección `test_results`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único del resultado
<code>user_id</code>	String	Sí	Referencia al usuario
<code>quiz_completed_at</code>	Date	Sí	Fecha de completado
<code>recommendations</code>	Array	Sí	Array de recomendaciones
<code>trait_scores</code>	Object	Sí	Puntuaciones por trait

Subdocumento: `recommendations`

Cuadro 12.18. Subdocumento `recommendations`

Campo	Tipo	Requerido	Descripción
<code>career</code>	Object	Sí	Información de la carrera
<code>score</code>	Number	Sí	Score de compatibilidad (0-1)
<code>compatibility_pct</code>	Number	Sí	Porcentaje de compatibilidad
<code>why_pos</code>	String	Sí	Factores positivos
<code>why_neg</code>	String	Sí	Factores negativos
<code>enhanced_explanation</code>	Object	Sí	Explicación detallada

Objeto: `career` (dentro de `recommendations`)

Cuadro 12.19. Objeto `career` en `recommendations`

Campo	Tipo	Requerido	Descripción
<code>career_id</code>	String	Sí	ID de la carrera
<code>name</code>	String	Sí	Nombre de la carrera
<code>image_url</code>	String	Sí	URL de imagen
<code>description</code>	String	Sí	Descripción
<code>duration</code>	Number	Sí	Duración en años
<code>employability</code>	String	Sí	Nivel de empleabilidad
<code>faculty</code>	String	Sí	Facultad

Objeto: `enhanced_explanation`

Cuadro 12.20. Objeto `enhanced_explanation`

Campo	Tipo	Requerido	Descripción
<code>positive_summary</code>	String	Sí	Resumen de aspectos positivos
<code>concerns_summary</code>	String	Sí	Resumen de preocupaciones
<code>personalized_advice</code>	String	Sí	Consejo personalizado

Objeto: `trait_scores`

Listing 12.2. Estructura de `trait_scores`

```
{
  "piaac": {
    "lit": Number,    // Alfabetización (0–1)
```

```

    "num": Number,      // Numeracia (0-1)
    "ps": Number        // Resoluci n de problemas (0-1)
  },
  "riasec": {
    "R": Number,        // Realistic (0-1)
    "I": Number,        // Investigative (0-1)
    "A": Number,        // Artistic (0-1)
    "S": Number,        // Social (0-1)
    "E": Number,        // Enterprising (0-1)
    "C": Number         // Conventional (0-1)
  },
  "bfi": {
    "O": Number,        // Openness (1-7)
    "C": Number,        // Conscientiousness (1-7)
    "E": Number,        // Extraversion (1-7)
    "A": Number,        // Agreeableness (1-7)
    "N": Number         // Neuroticism (1-7)
  },
  "grit": {
    "perseverance": Number, // (1-5)
    "consistency": Number,  // (1-5)
    "grits_overall": Number // (1-5)
  },
  "paa": {
    "read": Number,      // Lectura (0-1)
    "en": Number,        // Ingl s (0-1)
    "num": Number        // Matem ticas (0-1)
  }
}

```

12.2.11. Colección: testimonies

Testimonios de estudiantes y egresados sobre carreras.

Estructura del documento

Cuadro 12.21. Estructura de la colección **testimonies**

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único
<code>career_id</code>	ObjectId	Sí	Referencia a la carrera
<code>user_id</code>	ObjectId	No	Referencia al usuario (opcional)
<code>author_name</code>	String	Sí	Nombre del autor
<code>author_role</code>	String	Sí	Rol (estudiante/egresado)
<code>content</code>	String	Sí	Contenido del testimonio
<code>rating</code>	Number	No	Calificación (1-5)
<code>created_at</code>	Date	Sí	Fecha de creación
<code>approved</code>	Boolean	Sí	Si está aprobado (default: false)

12.2.12. Colección: `scoring_rules`

Reglas y configuraciones para el sistema de puntuación y matching de carreras.

Estructura del documento

Cuadro 12.22. Estructura de la colección `scoring_rules`

Campo	Tipo	Requerido	Descripción
<code>_id</code>	ObjectId	Sí	Identificador único
<code>rule_name</code>	String	Sí	Nombre de la regla
<code>rule_type</code>	String	Sí	Tipo de regla
<code>weight</code>	Number	Sí	Peso de la regla (0-1)
<code>parameters</code>	Object	No	Parámetros configurables
<code>active</code>	Boolean	Sí	Si la regla está activa
<code>description</code>	String	No	Descripción de la regla
<code>created_at</code>	Date	Sí	Fecha de creación
<code>updated_at</code>	Date	Sí	Fecha de actualización

Tipos de reglas

Las reglas de puntuación incluyen:

- `ria_cosine` - Similitud coseno de intereses RIASEC
- `bfi_gap_mean` - Diferencia en rasgos de personalidad
- `piaac_gap_*` - Brechas en habilidades cognitivas
- `grit_threshold` - Umbrales de perseverancia
- `paa_minimum` - Mínimos de aptitud académica

12.3. Relaciones entre colecciones

12.3.1. Diagrama de relaciones conceptuales

Las principales relaciones entre colecciones son:

- `users` → `test_results`: un usuario tiene múltiples resultados de tests (1:N)
- `users` → `saveditems`: un usuario puede guardar múltiples items (1:N)
- `users` → `forums`: un usuario crea múltiples foros (1:N)
- `users` → `interactions`: un usuario genera múltiples interacciones (1:N)
- `careers` → `forums`: una carrera tiene múltiples foros (1:N)
- `careers` → `testimonies`: una carrera tiene múltiples testimonios (1:N)
- `careers` → `tags`: relación muchos a muchos (M:N) - embebida

- **careers** → **courses**: relación muchos a muchos (M:N) - embebida en `plan_de_estudio`
- **cards** → **interactions**: una card tiene múltiples interacciones (1:N)
- **test_results** → **careers**: recomendaciones (N:M) - desnormalizado

12.3.2. Estrategias de modelado

El esquema utiliza diferentes estrategias según el caso de uso:

Documentos embebidos

Se utilizan para relaciones fuertemente acopladas:

- `forums.comments.answers` - Estructura jerárquica de discusión
- `careers.plan_de_estudio` - Plan de estudios por semestres
- `careers.tags` - Etiquetas con scores específicos por carrera

Referencias (ObjectId)

Para relaciones donde se necesita integridad referencial:

- `forums.creator_id` → `users._id`
- `forums.career_id` → `careers._id`
- `courses.prerequisitos` → `courses._id`

Desnormalización

Para optimizar lecturas y mantener históricos:

- `saveditems.item` - Copia completa del item guardado
- `test_results.recommendations.career` - Snapshot de la carrera al momento del test
- `users.user_tags` - Copia de información de tags para acceso rápido

12.4. Índices recomendados

Para optimizar el rendimiento de las consultas más frecuentes, se recomiendan los siguientes índices:

12.4.1. Colección: users

```
{ "clerk_id": 1 } // unique
{ "username": 1 }
{ "email": 1 }
{ "role": 1 }
{ "quizCompletedAt": 1 }
```

12.4.2. Colección: careers

```
{ "nombre_carrera": 1 }
{ "facultad": 1 }
{ "empleabilidad": 1 }
{ "tags.tag": 1 }
{ "tags.score": -1 }
```

12.4.3. Colección: forums

```
{ "career_id": 1, "created_at": -1 }
{ "creator_id": 1 }
{ "final_date": 1 }
```

12.4.4. Colección: cards

```
{ "type": 1, "priority": -1 }
{ "created_at": -1 }
{ "tags": 1 }
```

12.4.5. Colección: interactions

```
{ "cardId": 1, "created_at": -1 }
{ "action": 1 }
```

12.4.6. Colección: test_results

```
{ "user_id": 1 } // unique
{ "quiz_completed_at": -1 }
```

12.4.7. Colección: saveditems

```
{ "user_id": 1, "saved_at": -1 }
{ "type": 1, "item_id": 1 }
{ "user_id": 1, "type": 1 }
```

12.4.8. Colección: `test_items`

```
{ "id": 1 } // unique
{ "section": 1 }
```

12.4.9. Colección: `testimonies`

```
{ "career_id": 1, "approved": 1 }
{ "created_at": -1 }
```

12.5. Consideraciones de seguridad

12.5.1. Encriptación de datos

- Los datos sensibles se encriptan usando `repose.crypto.js` para datos en reposo
- Las comunicaciones utilizan `traffic.crypto.js` para encriptación en tránsito
- Los campos críticos en `users` y `forums` utilizan encriptación selectiva

12.5.2. Autenticación y autorización

- Autenticación delegada a Clerk mediante `clerk_id`
- Middleware de autenticación en `middleware/auth/`
- Verificación de webhooks en `middleware/webhook-auth/`
- Sistema de roles jerárquico: `admin > director > teacher > publisher > student`

12.5.3. Validación de datos

- Validación mediante Mongoose schemas con tipos estrictos
- Enumeraciones para campos con valores limitados
- Valores por defecto configurados en los esquemas
- Validación de referencias (`ObjectIds`) entre colecciones

12.6. Patrones de acceso comunes

12.6.1. Flujo de orientación vocacional

1. Usuario completa el cuestionario (`test_items`)
2. Se calculan los `trait_scores`
3. Se genera el matching con `careers` usando `scoring_rules`
4. Se almacenan las `recommendations` en `test_results`
5. Usuario puede guardar carreras recomendadas en `saveditems`

12.6.2. Flujo de exploración de contenido

1. Usuario accede al feed que consulta `cards`
2. Se registran `interactions` (view, tap)
3. Usuario puede guardar `cards` interesantes en `saveditems`
4. `Cards` de tipo `career` enlazan con `careers`
5. Usuario puede acceder a `forums` y `testimonies` de la carrera

12.6.3. Flujo de participación en foros

1. Usuario crea un `forum` asociado a una `career`
2. Otros usuarios agregan `comments` embebidos en el forum
3. Se pueden agregar `answers` embebidas en cada comment
4. Los foros tienen una `final_date` de cierre

12.7. Optimizaciones y consideraciones de performance

12.7.1. Desnormalización estratégica

Para reducir joins y mejorar velocidad de lectura:

- `saveditems.item` - Evita joins con `careers` o `cards` en cada consulta
- `test_results.recommendations.career` - Snapshot para mantener histórico
- `users.user_tags` - Acceso rápido a perfil sin joins con tags
- `careers.tags` - Incluye nombre del tag para evitar lookup

12.7.2. Agregaciones pre-calculadas

- `careers.insights` - Métricas y analytics pre-calculados
- `test_results.trait_scores` - Scores agregados por dimensión
- `recommendations.compatibility_pct` - Porcentaje calculado al momento del matching

12.7.3. Estrategias de Caching

Campos candidatos para caching en aplicación:

- Lista completa de `careers` (cambia poco frecuentemente)
- `test_items` (estático, solo lectura)
- `tags` (catálogo estático)
- `scoring_rules` (configuración que cambia raramente)

12.8. Escalabilidad y crecimiento

12.8.1. Particionamiento de datos

Para escalar horizontalmente se pueden aplicar:

- **Sharding por `user_id`** en `interactions`, `saveditems`, `test_results`
- **Sharding por `career_id`** en `forums`, `testimonies`
- **Time-based partitioning** para `interactions` (por fecha)

12.8.2. Archivado de datos históricos

- `interactions` más antiguas que X meses pueden moverse a colección de archivo
- `forums` cerrados hace más de 1 año pueden archivarse
- `test_results` mantener solo el más reciente por usuario en hot storage

12.8.3. Monitoreo de crecimiento

Métricas importantes a monitorear:

- Tamaño de colección `interactions` (crecimiento constante)
- Número de `forums.comments.answers` (documentos muy anidados)
- Tamaño de arrays en `careers.plan_de_estudio`
- Cantidad de `test_results` por usuario

12.9. Migraciones y versionado

12.9.1. Control de versiones del esquema

Se recomienda implementar:

- Campo `schema_version` en documentos críticos
- Migraciones incrementales para cambios de estructura
- Backward compatibility durante períodos de transición

12.9.2. Estrategias de migración

Para cambios en el esquema:

1. **Additive changes:** agregar campos opcionales sin afectar existentes
2. **Lazy migration:** actualizar documentos al momento de acceso
3. **Batch migration:** scripts de migración para cambios estructurales
4. **Parallel schema:** mantener dos versiones temporalmente

12.10. Conclusiones

El esquema de base de datos de MIRAI está diseñado para soportar un sistema complejo de orientación vocacional con las siguientes características clave:

12.10.1. Fortalezas del diseño

- **Flexibilidad:** uso de documentos embebidos y referencias según convenga
- **Performance:** desnormalización estratégica para optimizar lecturas
- **Escalabilidad:** estructura que permite sharding y particionamiento
- **Trazabilidad:** registro completo de interacciones y resultados históricos
- **Extensibilidad:** campos tipo Object para futuras expansiones

12.10.2. Áreas de atención

- Monitoreo del tamaño de arrays embebidos (comments, answers)
- Gestión del crecimiento de la colección interactions
- Sincronización de datos desnormalizados
- Performance de queries con múltiples joins

12.10.3. Recomendaciones finales

1. Implementar índices recomendados antes de producción
2. Establecer políticas de retención de datos históricos
3. Configurar alertas de crecimiento en colecciones críticas
4. Documentar estrategias de migración para cambios futuros
5. Implementar pruebas de carga para validar diseño
6. Considerar réplicas de lectura para queries analíticos

Este esquema representa la arquitectura de datos al momento de su documentación y está sujeto a evolución según las necesidades del sistema.