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

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

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, Noviembre del 2025

Vo.Bo.:

(f) _____

Ing. Ludwing Cano

Tribunal Examinador:

(f) _____

Ing. Ludwing Cano

(f) _____

Desconocido

(f) _____

Desconocido

Fecha de aprobación: Guatemala, n de diciembre de 2025.

Prefacio

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. Ludwing 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, extiendo 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 invaluables 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.

José Daniel Gómez Cabrera

Guatemala, Noviembre del 2025

Agradecimientos

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus.

xxxxxxx

Índice

Lista de Figuras

Lista de Cuadros

Resumen

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 América Latina. 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 latinoamericana. 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 América Latina, 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, América Latina.

CAPÍTULO 1

Introducción

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, 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 América Latina, 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 ge-

nerados 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.

CAPÍTULO 2

Objetivos

2.1. Objetivo General

Diseñar e implementar una infraestructura cloud segura, escalable y accesible para un sistema integrador de aplicaciones que consuma modelos externos de inteligencia artificial y proporcione, con una latencia menor a 300 ms en promedio y disponibilidad $> 99.9\,\%$, los datos necesarios para una multiplataforma de orientación vocacional dirigida a estudiantes graduandos.

2.2. Objetivos Específicos

- Diseñar la arquitectura de la infraestructura cloud del sistema integrador en un plazo no mayor a 4 semanas, asegurando independencia del proveedor (cloud-agnostic).
- Implementar la infraestructura definida utilizando un proveedor cloud, alcanzando un tiempo de despliegue automatizado < 10 minutos por ambiente (desarrollo, testing y producción).
- Desarrollar un backend integrador que garantice seguridad en datos personales mediante cifrado AES-256 en reposo y TLS 1.3 en tránsito, validado por pruebas de penetración.
- Configurar e implementar políticas de seguridad (autenticación OAuth 2.0, autorización basada en roles, rotación de claves cada 90 días, mitigación contra ataques DoS) para el sistema integrador.
- 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 < 500 ms bajo una carga de 500 usuarios concurrentes.
- 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.
- Asegurar la accesibilidad de la plataforma garantizando compatibilidad con $> 95\,\%$ de navegadores modernos y validando disponibilidad de acceso en condiciones de conexión de > 5 Mbps en pruebas de campo.

- Garantizar el flujo seguro de datos entre el sistema integrador y los modelos de IA externos mediante pruebas de integración que aseguren una tasa de error de comunicación $< 1\,\%$.

CAPÍTULO 3

Justificación

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.

CAPÍTULO 4

Marco Teórico

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.

4.1. Orientación vocacional en América Latina

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 América Latina, este proceso 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 es 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. En América Latina, 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.

Los métodos tradicionales de orientación vocacional, como pruebas estandarizadas, a menudo no consideran las particularidades de cada estudiante ni las tendencias del mercado laboral. Esto resulta en decisiones poco informadas, lo que puede llevar a cambios de carrera o deserción académica. En este contexto, el desarrollo de sistemas tecnológicos avanzados se presenta como una solución innovadora para abordar estas problemáticas.

4.2. 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 América Latina, 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.

4.3. 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 Latinoamérica es evidente debido a que como estudiantes 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]

4.4. Desafíos en América Latina

Los desafíos en la orientación vocacional en América Latina 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].

4.5. ¿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 y Latinoamérica.

4.6. 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.

4.7. ¿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].

4.7.1. 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. [17] [18]

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 América Latina, 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 términos de objetivos, esta sección cumple con el objetivo específico de diseñar una arquitectura que integre múltiples servicios y plataformas educativas (según se detalla en la sección 2.2), al proporcionar las bases para la comunicación estandarizada entre módulos. Además, contribuye al objetivo general de implementar una infraestructura cloud segura, ya que las APIs bien diseñadas facilitan la aplicación de políticas de autenticación y encriptación, mejorando la protección de datos personales. 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. [19]

4.7.2. 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. [20] [21] [22]

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 América Latina, 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.

Respecto a los objetivos, cumple con el específico de implementar políticas de seguridad en la integración de aplicaciones (sección 2.2), al centralizar mecanismos como autenticación y limitación de tasas, lo que reduce el riesgo de brechas. Contribuye al trabajo de graduación al proporcionar una arquitectura unificada que facilita la escalabilidad y el mantenimiento, permitiendo que el sistema atienda a múltiples usuarios simultáneamente sin degradar el rendimiento. En la implementación, un API Gateway en entornos cloud como AWS o Azure, combinado con un backend con tipado estricto, mejora la robustez al manejar errores tipados y rutas seguras, alineándose con discusiones sobre desarrollo backend eficiente.

4.7.3. 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. [23] [24] [25]

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 América Latina. 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.

Cumple con el objetivo específico de diseñar una arquitectura escalable (sección 2.2), al permitir el manejo eficiente de datos de múltiples usuarios sin comprometer la velocidad de respuesta. Contribuye al trabajo al fortalecer la infraestructura cloud, integrando con contenedores como Kubernetes para una distribución automática de cargas, lo que optimiza el consumo de modelos de AI y reduce tiempos de procesamiento. En combinación con backend en TypeScript, facilita implementaciones modulares que escalan horizontalmente sin refactorizaciones mayores.

4.7.4. 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. [26] [27]

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 América Latina, 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.

Cumple con el objetivo específico de integrar modelos de IA en el sistema integrador (sección 2.2), al facilitar la personalización de sugerencias educativas. 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 con TypeScript, asegura manejo seguro de datos en transacciones AI.

4.7.5. 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 apliaciones 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. [28]

La importancia de este componente en el trabajo de graduación radica en su capacidad para abordar las desigualdades educativas en América Latina, 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. [29]

En términos de objetivos, cumple con el específico de implementar una infraestructura que conecte múltiples servicios educativos (sección 2.2), al proporcionar una integración fluida que soporta consultas simultáneas y autenticación segura. Contribuye al objetivo general 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.

4.7.6. ¿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. [30] [31]

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. [32]

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 América Latina, 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.

Cumple con el objetivo específico de desarrollar la lógica de integración (sección 2.2), al gestionar autenticación y procesamiento de datos de IA. 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.

4.7.7. 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. [33]

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 microservicios, 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. [34]

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 América Latina. 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.

4.8. 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 [35]. 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 [36]. 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 [37]. 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 [38]. 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 [39]. 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 América Latina, asegurando accesibilidad y rendimiento óptimo.

4.9. 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.

4.9.1. 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 [40].

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 [41]. Sin embargo, requiere una robusta orquestación para evitar la complejidad de "microservices sprawl".

4.9.2. 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 [42].

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 1 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 [43]. Es particularmente adecuada para entornos educativos donde la demanda varía por temporadas.

4.9.3. 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 [44].

En el ámbito de la orientación vocacional, contenedores separados podrían albergar servicios como un ïntegrador 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 [45]. 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.

4.10. 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 [46].

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 latinoamericanos, donde la privacidad es un factor crítico para la adopción de herramientas tecnológicas.

4.10.1. 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 [47]. 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 en América Latina 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 [47]. 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 [48].

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 América Latina.

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, cumpliento 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.

4.10.2. Autentificación

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 [49].

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 [50]. 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 [51]. 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.

4.10.3. 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 [52].

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 América Latina, 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 América Latina, 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 [53]. En Brasil, la LGPD (Ley General de Protección de Datos) exige consentimiento explícito y medidas de seguridad para datos sensibles [54]. 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 [55].

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 [56]. 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 [57].

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 [58].

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.

4.10.4. 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 [59].

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 [60].

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 [61].

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.

4.11. Políticas de seguridad

Las políticas de seguridad definen las reglas y procedimientos para mantener la integridad del sistema. Estas incluyen políticas de acceso, respaldo de datos y respuesta a incidentes, adaptadas al entorno cloud [62].

Las políticas de seguridad son esenciales para estructurar el sistema integrador, asegurando que cumpla con los objetivos de confidencialidad y disponibilidad, lo que es clave para su implementación exitosa en América Latina.

Una política efectiva debe cubrir la clasificación de datos, obligando a encriptar información sensible y realizar revisiones periódicas. En el contexto latinoamericano, incorporar cumplimiento con leyes locales fortalece la política [63].

La clasificación de datos y el cumplimiento legal aplican al sistema integrador al garantizar que el sistema sea éticamente aceptable y funcional en un entorno regulado, apoyando los objetivos de accesibilidad y seguridad.

4.11.1. Desarrollo de políticas

El desarrollo implica identificar riesgos específicos del sistema integrador, como exposición de modelos IA, y definir controles como rotación de claves y entrenamiento en seguridad para el equipo [64].

El desarrollo de políticas impacta en la planificación del proyecto, permitiendo mitigar riesgos específicos y alineándose con el objetivo de crear un sistema seguro y sostenible a largo plazo.

Políticas específicas para la arquitectura cloud

Incluyen políticas de configuración segura, como deshabilitar accesos públicos innecesarios y usar redes privadas virtuales (VPC) para aislar recursos [65].

Estas políticas específicas influyen en la arquitectura del sistema, asegurando que la infraestructura cloud sea resistente a ataques y cumpla con los requisitos de escalabilidad y rendimiento establecidos en los objetivos.

CAPÍTULO 5

Metodología

El desarrollo del presente trabajo de graduación se basa en una aproximación estructurada y sistemática, enfocada exclusivamente en la implementación del sistema integrador para la plataforma inteligente de orientación vocacional "Mirai". Dado que el proyecto involucra la integración de tecnologías emergentes como la inteligencia artificial y la infraestructura cloud, se adopta una metodología ágil, específicamente adaptando principios de Scrum, para permitir iteraciones rápidas, flexibilidad ante cambios y una entrega continua de valor. Esta elección se justifica por la naturaleza dinámica del proyecto, donde los requisitos pueden evolucionar basados en pruebas iniciales y retroalimentación técnica, sin comprometer la calidad ni la seguridad. Esta aproximación general contribuye al cumplimiento del objetivo general, al diseñar e implementar la infraestructura cloud de manera iterativa y enfocada en estándares altos de rendimiento, disponibilidad y accesibilidad.

La metodología ágil se implementará a través de sprints de dos semanas, cada uno con objetivos claros, reuniones individuales diarias y breves para monitoreo de progreso, revisiones al final de cada sprint para validar avances y retrospectivas para identificar mejoras. El backlog del producto incluirá tareas como el diseño arquitectónico, la implementación de componentes de integración, la configuración de políticas de seguridad y las pruebas unitarias e integrales. Se utilizarán herramientas como Git para control de versiones, Notion para gestión de tareas y entornos de desarrollo en cloud como AWS para prototipado rápido. Esta aproximación asegura que el sistema integrador se desarrolle de manera iterativa, priorizando funcionalidades críticas como la autenticación segura y la consulta de modelos externos de IA, alineándose con los objetivos específicos relacionados con la seguridad y la eficiencia, independientemente del servicio cloud que se utilice.

Antes de detallar las fases del desarrollo, se analizan las tres arquitecturas investigadas en el marco teórico: basada en microservicios, serverless y contenerizada con Kubernetes. Este análisis considera sus puntos fuertes y débiles en el contexto del sistema integrador, que debe manejar integraciones con modelos de IA externos para orientación vocacional, priorizando escalabilidad, seguridad y eficiencia en costos. Esta sección contribuye al objetivo específico de diseñar la infraestructura del sistema integrador (objetivo específico 1), al evaluar opciones para una arquitectura óptima.

5.1. Análisis de Arquitecturas

5.1.1. Arquitectura Basada en Microservicios

Esta arquitectura divide el sistema en servicios independientes, cada uno responsable de una funcionalidad específica, comunicándose vía APIs. Esta evaluación apoya el objetivo específico de priorizar la escalabilidad (objetivo específico 6), ya que permite el escalado individual de componentes.

Puntos fuertes:

- Alta escalabilidad: Cada microservicio puede escalarse individualmente según la demanda.
- Flexibilidad: Facilita la integración de nuevos componentes, como APIs de modelos de IA externos.
- Mantenibilidad: Errores en un servicio no afectan al sistema completo, y permite actualizaciones independientes.

Puntos débiles:

- Complejidad operativa: Requiere orquestación para manejar múltiples servicios, lo que aumenta la overhead de gestión.
- Latencia: La comunicación entre servicios puede introducir retrasos si no se optimiza.
- Costos iniciales: Mayor esfuerzo en diseño y monitoreo distribuido.

5.1.2. Arquitectura Serverless

En esta aproximación, el código se ejecuta en funciones gestionadas por el proveedor cloud, sin necesidad de servidores dedicados. Este análisis se alinea con el objetivo específico de gestionar la velocidad de respuesta (objetivo específico 5), al enfocarse en ejecuciones eficientes y de bajo costo.

Puntos fuertes:

- Escalabilidad automática: Se ajusta a la demanda sin intervención manual, ideal para cargas variables en consultas de IA.
- Reducción de costos: Se paga solo por uso, minimizando gastos en periodos de baja actividad.
- Desarrollo rápido: Enfoque en lógica de negocio sin preocuparse por infraestructura subyacente.

Puntos débiles:

- Limitaciones en estado: Dificultad para manejar estados persistentes sin servicios adicionales.
- Cold starts: Retrasos iniciales en ejecuciones infrecuentes, que podrían afectar la respuesta en tiempo real.
- Dependencia del proveedor: Riesgo de vendor lock-in y menos control sobre el entorno.

5.1.3. Arquitectura Contenerizada con Kubernetes

Utiliza contenedores (e.g., Docker) orquestados por Kubernetes para desplegar y gestionar aplicaciones. Esta revisión contribuye al objetivo específico de asegurar la accesibilidad (objetivo específico 7), al promover portabilidad y consistencia en diferentes entornos.

Puntos fuertes:

- Portabilidad: Los contenedores son consistentes en diferentes entornos, facilitando despliegues híbridos.
- Orquestación avanzada: Kubernetes maneja autoescalado, balanceo de carga y recuperación automática.
- Seguridad integrada: Soporte para políticas de red y secrets management.

Puntos débiles:

- Curva de aprendizaje: Complejidad en configuración y mantenimiento de clústers.
- Overhead de recursos: Requiere más recursos computacionales que serverless para operaciones básicas.
- Costos operativos: Gestión continua de Kubernetes puede ser costosa en términos de tiempo y expertise.

Tras evaluar estas arquitecturas, se determina que ninguna por sí sola satisface óptimamente los requisitos del proyecto, que demanda eficiencia en integración de inteligencia artificial externa, bajo costo inicial y escalabilidad. Por ello, se opta por una arquitectura híbrida, combinando serverless como base principal y microservicios. Esta arquitectura hibrida parte de funciones serverless para manejar integraciones ligeras y de bajo costo como la autentificación y operaciones sencillas como leer, actualizar, crear y eliminar registros, como lo son los datos de usuarios, mientras que microservicios gestionan componentes persistentes como consumir agentes y modelos externos de inteligencia artificial (e.g., APIs de recomendación vocacional). Kubernetes se considera como opción futura para escalado avanzado, pero no se implementa en esta fase para evitar complejidad innecesaria del prototipo. Esta elección equilibra los puntos fuertes de escalabilidad y costo bajo, mitigando debilidades como latencia mediante optimizaciones en APIs y caching, y apoya múltiples objetivos específicos, como el desarrollo de un sistema integrador seguro (objetivo específico 3) y la garantía del flujo de datos seguro (objetivo específico 8).

5.2. Análisis de Lenguajes de Programación

Antes de proceder con las fases del desarrollo, se realiza un análisis comparativo de lenguajes de programación adecuados para implementar el backend del sistema integrador. Este análisis evalúa opciones populares para desarrollo de APIs y servicios cloud, considerando factores como rendimiento, facilidad de integración con APIs externas de IA, manejo de datos en formato JSON y compatibilidad con entornos serverless o de microservicios. Esta sección contribuye al objetivo específico de diseñar la infraestructura del sistema integrador (objetivo específico 1), al seleccionar un lenguaje que optimice la eficiencia y la seguridad en el manejo de datos vocacionales sensibles.

5.2.1. Go (Golang)

Go es un lenguaje compilado diseñado por Google, enfocado en simplicidad, eficiencia y concurrencia. Esta evaluación apoya el objetivo específico de priorizar la velocidad de respuesta (objetivo específico 5), ya que su rendimiento nativo es ideal para sistemas de alto tráfico.

Puntos fuertes:

- Alto rendimiento: Compilación a binarios nativos resulta en ejecuciones rápidas y bajo consumo de memoria, perfecto para escalabilidad en cloud.
- Concurrencia integrada: Las goroutines facilitan el manejo de múltiples solicitudes simultáneas, como consultas a modelos de IA.
- Seguridad: Tipado estático reduce errores en tiempo de ejecución, y su ecosistema incluye herramientas para autenticación segura.

Puntos débiles:

- Curva de aprendizaje: Menos intuitivo para desarrolladores sin experiencia en lenguajes compilados, lo que podría ralentizar el prototipado.
- Ecosistema limitado: Menos bibliotecas para integración rápida con JSON o APIs web comparado con lenguajes interpretados.
- Verboso: Requiere más código para tareas comunes, aumentando el tiempo de desarrollo inicial.

5.2.2. Java

Java es un lenguaje orientado a objetos, multiplataforma y maduro, ampliamente usado en entornos enterprise. Este análisis se alinea con el objetivo específico de garantizar la seguridad de los datos (objetivo específico 8), gracias a su robustez en aplicaciones seguras.

Puntos fuertes:

- Madurez y ecosistema: Amplias bibliotecas (e.g., Spring Boot) para APIs REST, integración con cloud y manejo de seguridad.
- Portabilidad: "Write once, run anywhere"facilita despliegues en diferentes proveedores cloud.
- Rendimiento: Optimizado para aplicaciones de gran escala, con buen soporte para multihilo.

Puntos débiles:

- Verboso y complejo: Requiere más boilerplate code, lo que incrementa el tiempo de desarrollo y mantenimiento.
- Consumo de recursos: Mayor uso de memoria comparado con lenguajes más ligeros, elevando costos en entornos cloud.
- Curva de aprendizaje: Configuraciones complejas para proyectos simples, no ideal para iteraciones rápidas.

5.2.3. Python

Python es un lenguaje interpretado, versátil y de alto nivel, popular en IA y desarrollo web. Esta revisión contribuye al objetivo específico de integrar modelos de IA externos (objetivo específico 3), dada su compatibilidad con bibliotecas de machine learning.

Puntos fuertes:

- Facilidad de uso: Sintaxis clara y concisa acelera el desarrollo, ideal para prototipado rápido de integradores.
- Ecosistema rico: Bibliotecas como Flask, FastAPI y requests facilitan la creación de APIs y manejo de JSON.
- Integración con IA: Nativo soporte para herramientas como TensorFlow o APIs de modelos externos.

Puntos débiles:

- Rendimiento: Más lento en ejecuciones intensivas debido a ser interpretado, lo que podría afectar la escalabilidad.
- GIL (Global Interpreter Lock): Limita la concurrencia real en multihilo, problemático para cargas altas.
- Tipado dinámico: Mayor riesgo de errores en runtime, especialmente en sistemas con datos sensibles.

5.2.4. JavaScript (con Node.js)

JavaScript, ejecutado en el runtime Node.js para backend, es un lenguaje interpretado y event-driven. Esta evaluación apoya el objetivo específico de asegurar la accesibilidad y compatibilidad (objetivo específico 7), por su nativa afinidad con JSON y web standards.

Puntos fuertes:

- Compatibilidad con JSON: JSON es nativo en JavaScript, facilitando parsing y manipulación sin overhead, ideal para integraciones con APIs de IA que usan este formato.
- Asincronía: Modelo non-blocking I/O maneja eficientemente solicitudes concurrentes, mejorando la velocidad en consultas vocacionales.
- Ecosistema web: Bibliotecas como Express.js y Axios permiten desarrollo rápido de APIs RESTful y serverless (e.g., con AWS Lambda).

Puntos débiles:

- Tipado dinámico: Puede llevar a errores sutiles si no se maneja con herramientas como TypeScript.
- Rendimiento en CPU-intensive: Menos eficiente para tareas computacionales pesadas comparado con lenguajes compilados.
- Madurez en enterprise: Aunque creciente, requiere más cuidado en seguridad para aplicaciones críticas.

Al profundizar en los diferentes lenguajes de programación, se determinó que JavaScript con Node.js es la opción óptima para el sistema integrador, ya que ninguna alternativa equilibra tan bien la facilidad de desarrollo, alta compatibilidad con proveedores cloud, el rendimiento en entornos web y la compatibilidad nativa con JSON. Esta elección se justifica por la necesidad de manejar datos en formato JSON de manera eficiente, proveniente de modelos de IA externos y respuestas personalizadas de orientación vocacional, reduciendo latencia en parsing y serialización. Además, Node.js soporta perfectamente arquitecturas híbridas serverless-microservicios, permitiendo despliegues rápidos en proveedores cloud sin vendor lock-in. Las ventajas incluyen desarrollo iterativo rápido alineado con metodología ágil, integración seamless con frontend (si se expande), y un ecosistema vasto para seguridad (e.g., JWT para autenticación). Para mitigar debilidades, se incorporará TypeScript para tipado estático, asegurando robustez sin sacrificar velocidad. Esta selección apoya múltiples objetivos específicos, como el desarrollo de un sistema integrador seguro (objetivo específico 3) y la optimización de flujos de datos (objetivo específico 8), priorizando compatibilidad y eficiencia en un contexto de orientación educativa accesible. No obstante, se considera que la arquitectura de microservicios permite realizar implementaciones de diferentes lenguajes a la vez, por lo que se consideró Python para realizar un microservicio de consumo de modelos de inteligencia artificial y machine learning para facilitar su consumo e implementación.

5.3. Análisis de Proveedores de Nube

Antes de proceder con las fases del desarrollo, se realiza un análisis comparativo de proveedores de nube líderes para hospedar la infraestructura del sistema integrador. Este análisis evalúa opciones populares considerando factores como servicios disponibles, integración con IA, escalabilidad, seguridad, costos y presencia global, especialmente en América Latina. Esta sección contribuye al objetivo específico de diseñar la infraestructura del sistema integrador (objetivo específico 1), al seleccionar un proveedor que optimice la eficiencia, la accesibilidad y la seguridad en el manejo de datos vocacionales sensibles.

5.3.1. Amazon Web Services (AWS)

AWS, lanzado en 2006, es el proveedor de nube más maduro y con la mayor cuota de mercado global (alrededor del 31 % en 2025) [66]. Ofrece una amplia gama de servicios, incluyendo cómputo serverless (Lambda), almacenamiento (S3), bases de datos (RDS, DynamoDB) y herramientas de IA (SageMaker). Esta evaluación apoya el objetivo específico de priorizar la escalabilidad (objetivo específico 6), gracias a su robusta infraestructura global [67]. **Puntos fuertes:**

- Amplia gama de servicios: Más de 200 servicios fully-featured, facilitando integraciones complejas como APIs de IA externas y sistemas de recomendación vocacional [68].
- Escalabilidad y rendimiento: Autoescalado global con 38 regiones geográficas, ideal para usuarios en América Latina con baja latencia [67].
- Seguridad avanzada: Herramientas como IAM, KMS y GuardDuty para políticas de seguridad granular, alineadas con el manejo de datos personales sensibles.
- Integración con IA: Servicios como Bedrock y SageMaker permiten consumo eficiente de modelos externos, apoyando el objetivo de integración de IA (objetivo específico 3).

Puntos débiles:

- Complejidad: La vasta oferta puede ser abrumadora para principiantes, requiriendo más tiempo de aprendizaje.

- Costos: Puede ser más caro para cargas impredecibles si no se optimiza, aunque ofrece modelos de pago por uso.
- Vendor lock-in: Dependencia de servicios propietarios podría complicar migraciones futuras.

5.3.2. Microsoft Azure

Azure, introducido en 2010, se integra seamlessly con el ecosistema Microsoft, ofreciendo servicios como Azure Functions para serverless, Blob Storage y Azure AI. Con una cuota de mercado del 24 % en 2025, es fuerte en entornos enterprise [66]. Este análisis se alinea con el objetivo específico de garantizar la seguridad de los datos (objetivo específico 8), por sus certificaciones y herramientas de cumplimiento. **Puntos fuertes:**

- Integración con Microsoft: Ideal para organizaciones usando Office 365 o Windows, facilitando autenticación híbrida con Active Directory.
- Servicios de IA: Azure OpenAI y Cognitive Services para recomendaciones personalizadas, útil para orientación vocacional.
- Híbrido y multi-nube: Soporte fuerte para entornos on-premise, beneficiando accesibilidad en regiones con infraestructura limitada.
- Seguridad: Azure Sentinel y compliance con GDPR/HIPAA, protegiendo datos personales.

Puntos débiles:

- Menos regiones globales: Más de 70 regiones, pero potencialmente aumentando latencia en América Latina comparado con competidores si no se elige óptimamente [69].
- Costos variables: Precios pueden ser más altos para IA en comparación con GCP.
- Curva de aprendizaje: Interfaz menos intuitiva para no-usuarios de Microsoft.

5.3.3. Google Cloud Platform (GCP)

GCP, lanzado en 2011, destaca en datos y IA, con servicios como Cloud Functions, BigQuery y Vertex AI. Posee el 11 % del mercado en 2025 y es conocido por innovación en big data [66]. Esta revisión contribuye al objetivo específico de integrar modelos de IA externos (objetivo específico 3), dada su fortaleza en machine learning. **Puntos fuertes:**

- Fortalezas en IA y datos: Vertex AI y AutoML para recomendaciones vocacionales eficientes y análisis de datos.
- Costos competitivos: A menudo más barato para cargas de datos intensivas, con descuentos sostenidos.
- Sostenibilidad: Enfoque en energía renovable, atractivo para proyectos educativos.
- Innovación: Servicios como Anthos para multi-nube, mejorando portabilidad.

Puntos débiles:

- Menos maduro: Menos servicios generales comparado con AWS, limitando opciones para integradores complejos.
- Cobertura global: 40 regiones, pero menos presencia en América Latina que AWS [70].
- Soporte enterprise: Menos enfocado en grandes empresas, con interfaz que puede ser confusa para escalabilidad masiva.

En base a las diferentes especificaciones de cada uno de los proveedores cloud, se determinó que Amazon Web Services (AWS) es la opción óptima para el sistema integrador, ya que equilibra mejor la amplitud de servicios, la escalabilidad global y la integración con IA en comparación con Azure y GCP. Aunque Azure ofrece integración superior con Microsoft y GCP destaca en costos para IA, AWS proporciona la mayor madurez y ecosistema completo, con herramientas como API Gateway para integraciones seguras y Lambda para serverless híbrido, reduciendo latencia en consultas vocacionales. Su presencia global, incluyendo regiones en Sudamérica, asegura accesibilidad (objetivo específico 7) para usuarios latinoamericanos, mientras que sus características de seguridad (IAM, encryption) garantizan el flujo de datos seguro (objetivo específico 8). Independientemente, AWS es útil por su modelo de pago por uso que minimiza costos iniciales, su soporte para Node.js en entornos híbridos y su liderazgo en mercado, permitiendo un desarrollo iterativo alineado con la metodología ágil y apoyando el objetivo general de una infraestructura robusta para orientación vocacional personalizada.

5.4. Análisis de Proveedores de Bases de Datos en AWS

Antes de proceder con las fases del desarrollo, se realiza un análisis comparativo de proveedores de bases de datos en AWS para almacenar y gestionar los datos del sistema integrador. Este análisis evalúa opciones populares considerando factores como tipo de base de datos, escalabilidad, rendimiento, seguridad, costos y compatibilidad con entornos cloud, especialmente para datos estructurados como información de cursos y datos flexibles como perfiles de estudiantes en América Latina. Esta sección contribuye al objetivo específico de diseñar la infraestructura del sistema integrador (objetivo específico 1), al seleccionar soluciones que optimicen la eficiencia, la accesibilidad y la seguridad en el manejo de datos vocacionales sensibles.

5.4.1. Amazon DynamoDB

Amazon DynamoDB es una base de datos NoSQL completamente gestionada, orientada a clave-valor y documentos, lanzada en 2012. Ofrece escalabilidad automática y rendimiento de milisegundos, ideal para aplicaciones con cargas variables [71]. Esta evaluación apoya el objetivo específico de priorizar la escalabilidad (objetivo específico 6), gracias a su capacidad para manejar grandes volúmenes de datos sin intervención manual [72].

Puntos fuertes:

- Escalabilidad horizontal ilimitada: Puede manejar petabytes de datos y millones de solicitudes por segundo, ajustándose automáticamente a la demanda [72].
- Alto rendimiento y baja latencia: Respuestas en milisegundos, perfecto para consultas en tiempo real como recomendaciones vocacionales personalizadas.
- Flexibilidad en el modelo de datos: Soporta esquemas dinámicos, facilitando el almacenamiento de datos no estructurados como preferencias de usuarios.

- Integración con AWS: Fácil conexión con servicios como Lambda y API Gateway, apoyando la integración de modelos de IA externos (objetivo específico 3).
- Seguridad integrada: Encriptación en reposo y tránsito, con IAM para control de acceso granular.

Puntos débiles:

- Limitado en consultas complejas: No soporta joins o transacciones ACID completas, lo que complica queries relacionales [71].
- Costos impredecibles: Modelo de pago por uso puede aumentar con picos de tráfico si no se optimiza.
- Curva de aprendizaje: Requiere diseño de tablas basado en patrones de acceso, diferente de SQL tradicional [73].

5.4.2. Amazon RDS

Amazon RDS (Relational Database Service) es un servicio gestionado para bases de datos relacionales, compatible con motores como MySQL, PostgreSQL y SQL Server, lanzado en 2009. Proporciona automatización en backups, patching y escalado [74]. Este análisis se alinea con el objetivo específico de garantizar la seguridad de los datos (objetivo específico 8), por sus características de alta disponibilidad y cumplimiento normativo.

Puntos fuertes:

- Soporte para consultas complejas: Joins, transacciones ACID y esquemas estructurados, ideal para datos relacionales como catálogos de cursos y relaciones entre entidades.
- Alta disponibilidad: Multi-AZ para failover automático y read replicas para distribución de carga [74].
- Gestión automatizada: Backups, actualizaciones y monitoreo integrados, reduciendo overhead operativo.
- Seguridad robusta: Encriptación, VPC isolation y compliance con estándares como GDPR y HIPAA.
- Integración con AWS: Fácil uso con otros servicios cloud, mejorando la accesibilidad (objetivo específico 7).

Puntos débiles:

- Escalabilidad vertical limitada: Requiere instancias más grandes para crecer, no tan flexible como NoSQL para cargas masivas [71].
- Costos más altos para alta disponibilidad: Multi-AZ incrementa precios, aunque ofrece pago por uso.
- Dependencia de esquema fijo: Menos adaptable a cambios en datos no estructurados [73].

5.4.3. Amazon DocumentDB

Amazon DocumentDB es una base de datos de documentos completamente gestionada y compatible con MongoDB, lanzada en 2019, que emula la API de MongoDB para facilitar migraciones. Funciona sobre instancias EC2 subyacentes en la infraestructura de AWS, ofreciendo un modelo serverless que escala automáticamente la capacidad en incrementos finos según la demanda, sin necesidad de aprovisionar servidores manualmente [75]. Esta evaluación contribuye al objetivo específico de integrar modelos de IA externos (objetivo específico 3), gracias a su soporte para datos JSON flexibles y consultas complejas en estructuras anidadas [76].

Puntos fuertes:

- Compatibilidad con MongoDB: Permite migrar aplicaciones existentes sin cambios significativos en el código, utilizando drivers y APIs estándar de MongoDB, lo que ofrece ventajas sobre DynamoDB para equipos familiarizados con MongoDB [77].
- Escalabilidad y rendimiento: Escala horizontalmente hasta 64 TiB con latencia de milisegundos y hasta 15 réplicas de lectura para alta disponibilidad, ideal para cargas variables en orientación vocacional [78].
- Seguridad integrada: Encriptación en reposo y en tránsito mediante AWS KMS, con soporte para VPC y control de acceso granular vía IAM, cumpliendo estándares como SOC 2 y GDPR, superior en algunos aspectos a implementaciones auto-gestionadas de MongoDB al eliminar overhead de gestión de seguridad [75].
- Gestión automatizada: Backups automáticos, monitoreo y parches manejados por AWS, reduciendo costos operativos en comparación con clústeres MongoDB auto-hospedados en EC2, donde se requiere mantenimiento manual [76].
- Beneficios frente a otros basados en MongoDB: Como servicio gestionado, ofrece menor TCO (hasta 43 % de ahorro en instancias optimizadas para memoria) y escalado automático sin vendor lock-in completo, a diferencia de MongoDB Atlas, que puede tener limitaciones en integración exclusiva con AWS; también supera a DynamoDB en consultas complejas anidadas y a RDS en flexibilidad para datos no estructurados [78].

Puntos débiles:

- Compatibilidad limitada: No soporta todas las características avanzadas de MongoDB (e.g., sharding completo), lo que puede requerir reescrituras para aplicaciones complejas, a diferencia de MongoDB nativo [77].
- Costos para I/O intensivo: Aunque ofrece configuraciones I/O-optimizadas con hasta 40
- Dependencia de AWS: Vendor lock-in y curva de aprendizaje para optimizar instancias EC2 subyacentes, menos flexible que RDS para datos relacionales estrictos [78].

Tras evaluar estos proveedores de bases de datos, incluyendo Amazon DocumentDB como alternativa para workloads compatibles con MongoDB, se determina que Amazon DocumentDB es la opción óptima para el sistema integrador. Aunque RDS ofrece robustez en consultas estructuradas y DynamoDB destaca en escalabilidad para datos variables, se opta por DocumentDB por su superior seguridad con características integradas como encriptación en reposo y en tránsito mediante AWS KMS [79], escalabilidad horizontal automática hasta 64 TiB con baja latencia [78], compatibilidad con la API de MongoDB que facilita migraciones sin cambios en el código [75], facilidad de desarrollo gracias a su gestión automatizada y uso de drivers estándar [80], compatibilidad con otros

sistemas para integraciones o transiciones futuras [81], facilidad en la realización de queries complejas y anidadas, capacidad para realizar llamadas foráneas mediante populate en bibliotecas como Mongoose para resolver referencias similares a joins [78], y la gran cantidad de librerías compatibles (e.g., MongoDB Node.js driver, Mongoose) que permiten un manejo eficiente y extendido [75]. Esta elección se justifica por la necesidad de manejar datos flexibles como perfiles de estudiantes y cursos con integridad y eficiencia, facilitando el escalado para miles de estudiantes en América Latina y reduciendo latencia en accesos personalizados. Su integración seamless en AWS asegura seguridad (objetivo específico 8) y eficiencia en flujos de datos. Independientemente, DocumentDB minimiza costos iniciales mediante pago por uso, soporta desarrollo iterativo con Node.js y alinea con el objetivo general de una infraestructura escalable para orientación vocacional, adaptándose a cargas variables y datos sensibles [71].

5.5. Análisis de Sistemas de Autenticación para AWS

Antes de proceder con las fases del desarrollo, se realiza un análisis comparativo de sistemas de autenticación para integrar en la infraestructura AWS del sistema integrador. Este análisis evalúa opciones considerando factores como facilidad de integración, seguridad, escalabilidad, soporte para autenticación multifactor, manejo de sesiones y compatibilidad con aplicaciones web, especialmente para usuarios en América Latina con datos sensibles vocacionales. Esta sección contribuye al objetivo específico de garantizar la seguridad de los datos (objetivo específico 8), al seleccionar una solución que optimice la autenticación segura y el control de acceso en el manejo de información personal.

5.5.1. AWS Cognito

AWS Cognito es un servicio de autenticación y autorización nativo de AWS, lanzado en 2014, que proporciona user pools para registro y login, e identity pools para acceso a recursos AWS [82]. Ofrece integración profunda con otros servicios AWS como Lambda y API Gateway. Esta evaluación apoya el objetivo específico de priorizar la escalabilidad (objetivo específico 6), gracias a su capacidad para manejar millones de usuarios sin servidores dedicados [83]. **Puntos fuertes:**

- Integración nativa con AWS: Facilita el acceso seguro a recursos como S3 o DynamoDB, alineado con la arquitectura híbrida del proyecto.
- Escalabilidad ilimitada: Maneja picos de tráfico automáticamente, ideal para un sistema accesible en múltiples regiones latinoamericanas [82].
- Seguridad avanzada: Soporte para MFA, OAuth, SAML y encriptación, protegiendo datos personales sensibles.
- Costos basados en uso: Pago por usuarios activos mensuales, minimizando gastos iniciales para un proyecto educativo.

Puntos débiles:

- Complejidad de configuración: Requiere conocimiento profundo de AWS, aumentando la curva de aprendizaje y tiempo de desarrollo [83].
- Limitaciones en UX: Interfaz de autenticación predeterminada es básica, requiriendo personalización adicional para una experiencia amigable.
- Vendor lock-in: Dependencia total de AWS, complicando migraciones futuras si se expande el proyecto.

5.5.2. Clerk

Clerk es una plataforma de autenticación y gestión de usuarios moderna, lanzada en 2020, que ofrece componentes listos para usar en aplicaciones web, con soporte para social logins, passwordless y sesiones seguras [84]. Se integra fácilmente con backends Node.js y AWS mediante SDKs. Este análisis se alinea con el objetivo específico de asegurar la accesibilidad (objetivo específico 7), por su enfoque en experiencias de usuario intuitivas y multiplataforma [85]. **Puntos fuertes:**

- Facilidad de integración: SDKs para JavaScript/Node.js permiten implementación rápida en entornos serverless como Lambda, acelerando el desarrollo iterativo alineado con la metodología ágil [85].
- Seguridad integral: Monitoreo de dispositivos, revocación de sesiones y cumplimiento con estándares como SOC 2, garantizando protección de datos vocacionales sensibles [84].
- Experiencia de usuario superior: Componentes UI prebuilt para login, registro y MFA, ideal para estudiantes en contextos educativos con baja tolerancia a fricciones.
- Escalabilidad y flexibilidad: Maneja crecimiento de usuarios sin overhead, con soporte para integraciones con IA externas mediante JWT seguros.

Puntos débiles:

- Costos para escalado: Plan gratuito limitado; tiers pagos por usuarios activos, potencialmente más caros que Cognito para volúmenes muy altos [86].
- Dependencia externa: No nativo de AWS, requiriendo configuración adicional para integración con servicios como IAM.
- Menos maduro en enterprise: Aunque creciente, puede faltar algunas características avanzadas de federación en comparación con Cognito.

Ambos son sistemas de autentificación robustos y seguros, pero Clerk es la mejor opción para el sistema integrador en AWS, ya que equilibra mejor la facilidad de desarrollo, la seguridad y la experiencia de usuario en comparación con Cognito. Aunque Cognito ofrece integración nativa y escalabilidad robusta, Clerk simplifica la implementación en un backend Node.js, reduciendo tiempo de desarrollo y permitiendo enfocarse en funcionalidades core como la integración de modelos de IA (objetivo específico 3). Su enfoque en UI intuitiva asegura accesibilidad para estudiantes latinoamericanos (objetivo específico 7), mientras que las características de seguridad como MFA y monitoreo de sesiones garantizan el flujo de datos seguro (objetivo específico 8). Independientemente, Clerk es útil por su modelo developer-friendly que minimiza boilerplate code, soporta autenticación social para mayor inclusión en regiones con diversidad tecnológica, y alinea con el objetivo general de una infraestructura segura y escalable para orientación vocacional personalizada, facilitando un despliegue rápido y mantenible en AWS [83].

5.6. Prototipo

Se realizó un prototipo del sistema integrador de aplicaciones con el propósito de cumplir los objetivos propuestos. Para esto, se realizó una planificación de desarrollo, con el fin de poder cumplir con buenas prácticas de desarrollo y así realizar un trabajo práctico, ordenado, secuencial y planificado.

5.6.1. Nombre de la plataforma de orientación vocacional

Para determinar el nombre del sistema, se realizó una reunión con el fin de establecer un nombre comercial para la plataforma de orientación vocacional. Se llegó al acuerdo de nombrar a la plataforma: **Mirai** (palabra en japonés) que significa *futuro*. La plataforma de orientación vocacional tiene como principal fin contribuir al futuro de la orientación vocacional de Latinoamérica.

5.7. Planificación del desarrollo

El desarrollo del sistema integrador Mirai se realizó siguiendo una metodología basada en arquitectura serverless en la nube, priorizando la escalabilidad, seguridad y mantenibilidad del sistema. La planificación se estructuró en fases progresivas que permitieron la construcción incremental de funcionalidades.

5.7.1. Fases del Desarrollo

El proyecto se divide en fases iterativas alineadas con la metodología ágil:

1. **Investigación:** Indagación profunda en los conceptos y métodos necesarios para poder implementar el diseño y arquitectura del sistema integrador para la plataforma inteligente de orientación vocacional, así como para el desarrollo del prototipo del mismo.
2. **Planificación y Diseño:** Definición de requisitos del backend, modelado de datos y diseño de APIs para integración con modelos de IA. Incluye configuración inicial de entornos cloud y políticas de seguridad (e.g., IAM, encriptación). Esta fase cumple directamente con el objetivo específico de diseñar la infraestructura (objetivo específico 1).
3. **Implementación:** Desarrollo de microservicios y funciones serverless en lenguajes como Python o Node.js, utilizando frameworks como Express o FastAPI. Se integra autenticación (e.g., OAuth) y consultas seguras a APIs externas. Esta etapa aborda el objetivo específico de implementar la infraestructura priorizando seguridad, velocidad, escalabilidad y accesibilidad (objetivo específico 2), así como la implementación de políticas de seguridad (objetivo específico 4).
4. **Pruebas:** Unitarias, de integración y de seguridad. Simulando cargas para validar escalabilidad. Esta fase contribuye a priorizar la escalabilidad (objetivo específico 6) y gestionar la velocidad de respuesta (objetivo específico 5) mediante validaciones empíricas.
5. **Optimización y Validación:** Monitoreo de rendimiento con herramientas cloud (e.g., CloudWatch) y ajustes basados en métricas. Validación final contra objetivos específicos. Esta fase asegura la accesibilidad (objetivo específico 7) y el flujo de datos seguro (objetivo específico 8), cerrando el ciclo hacia el objetivo general.

Esta metodología asegura un desarrollo eficiente, enfocado en el backend, con énfasis en seguridad y escalabilidad para el sistema integrador, cumpliendo integralmente con los objetivos propuestos.

5.7.2. Análisis y Selección de Tecnologías

Análisis de Plataformas Cloud

Basándose en un proyecto que requiere alta escalabilidad y disponibilidad, se analizaron diferentes opciones de plataformas cloud computing, evaluando tanto soluciones en la nube pública como on-premise. Los criterios de selección incluyeron:

- **Capacidad de escalabilidad automática**: Necesaria para manejar cargas variables de usuarios
- **Servicios de computación serverless**: Para reducir costos operativos y complejidad de mantenimiento
- **Servicios de base de datos administrados**: Con soporte para bases de datos NoSQL orientadas a documentos
- **Sistemas de autenticación y autorización**: Con integración de terceros
- **API Gateway**: Para gestión centralizada de endpoints
- **Redes privadas virtuales (VPC)**: Para seguridad y aislamiento de recursos
- **Costos**: Modelo de pago por uso con capa gratuita para desarrollo

Tras el análisis, se seleccionó **Amazon Web Services (AWS)** como plataforma cloud principal debido a:

- Amplia documentación y comunidad de soporte
- Servicios maduros y confiables de computación serverless (Lambda)
- Integración nativa entre servicios (API Gateway, Lambda, VPC, EC2)
- Modelo de facturación favorable para proyectos escalables
- Disponibilidad de servicios de seguridad avanzados (Security Groups, IAM)

Selección de Servicios AWS

Los servicios de AWS seleccionados fueron:

1. **AWS Lambda**: Servicio de computación serverless para ejecutar código sin provisionar servidores. Se utilizó para implementar todas las funciones de negocio del backend.
2. **Amazon API Gateway HTTP API**: Servicio administrado para crear APIs HTTP de alto rendimiento. Proporciona el punto de entrada para todas las solicitudes del cliente con mejor rendimiento que REST API.
3. **Amazon VPC (Virtual Private Cloud)**: Red virtual aislada para alojar recursos con requisitos de seguridad adicionales.
4. **Amazon EC2**: Instancia de computación para alojar el microservicio de recomendaciones y la base de datos DocumentDB.

5. **Amazon DocumentDB**: Base de datos compatible con MongoDB, alojada dentro de la VPC para mayor seguridad.
6. **AWS WAF (Web Application Firewall)**: Firewall de aplicación web que protege contra ataques comunes como SQL injection, XSS, DDoS, y otros vectores de ataque.
7. **Amazon CloudFront**: Content Delivery Network (CDN) global que proporciona entrega de contenido rápida y segura, integrado con WAF. CloudFront es necesario para conectar WAF con HTTP API Gateway, ya que HTTP API Gateway no soporta integración directa con WAF.
8. **AWS IAM (Identity and Access Management)**: Para gestión de permisos y políticas de acceso entre servicios.
9. **Security Groups**: Firewalls virtuales para controlar el tráfico de entrada y salida de los recursos.
10. **VPC Link**: Para permitir que API Gateway se conecte de forma segura con recursos dentro de la VPC.

Selección de Base de Datos

Para el almacenamiento de datos se seleccionó **MongoDB** (a través de Amazon DocumentDB) por las siguientes razones:

- **Esquema flexible**: Permite iteración rápida durante el desarrollo sin migraciones complejas
- **Modelo de documentos**: Ideal para almacenar estructuras complejas como perfiles de usuario, contenido de tarjetas e interacciones
- **Escalabilidad horizontal**: Mediante réplicas y sharding
- **Consultas eficientes**: Soporte para índices y agregaciones complejas
- **Integración con ODM**: Uso de Mongoose para validación y modelado de datos

Sistema de Autenticación

Se seleccionó **Clerk** como proveedor de autenticación por:

- Integración sencilla mediante JWT (JSON Web Tokens)
- Soporte para múltiples métodos de autenticación
- Gestión de sesiones y tokens de forma segura
- Webhooks para sincronización de eventos de usuario
- Reducción del tiempo de desarrollo al no implementar autenticación desde cero

5.7.3. Arquitectura del Sistema

El sistema Mirai se diseñó con una arquitectura serverless de tres capas:

1. **Capa de presentación**: Cliente (aplicación móvil/web) que se comunica vía HTTPS
2. **Capa de aplicación**: API Gateway con funciones Lambda serverless
3. **Capa de datos**: DocumentDB alojado en VPC privada

Flujo de Comunicación

1. El cliente envía solicitudes HTTPS al dominio `api.miraiedu.online`
2. **AWS WAF** analiza la solicitud y aplica reglas de seguridad (protección contra DDoS, SQL injection, XSS, etc.)
3. Si la solicitud pasa las validaciones del WAF, se redirige a **Amazon CloudFront**
4. CloudFront actúa como CDN y proxy, optimizando la entrega de contenido
5. CloudFront reenvía la solicitud a **HTTP API Gateway** en la región us-east-2
6. API Gateway valida la autenticación mediante el **Lambda Authorizer** que verifica tokens JWT de Clerk
7. Si la autenticación es exitosa, API Gateway invoca la función Lambda correspondiente
8. Las funciones Lambda se conectan a DocumentDB dentro de la VPC
9. Para el microservicio de recomendaciones, API Gateway utiliza **VPC Link** para comunicarse con la instancia EC2
10. La respuesta se devuelve al cliente a través de la misma cadena (HTTP API Gateway → CloudFront → WAF → Cliente)

5.7.4. Planificación por Fases

El desarrollo se estructuró en fases incrementales, cada una construyendo sobre la anterior:

Fase 1: Configuración de Infraestructura Base

Objetivo: Establecer la infraestructura cloud fundamental y servicios de red con capas de seguridad avanzadas.

Actividades:

- Creación de cuenta AWS y configuración de región (us-east-2)
- Diseño y creación de VPC con subredes públicas y privadas
- Configuración de Internet Gateway y tablas de enrutamiento
- Configuración de Security Groups base para Lambda y EC2
- Registro de dominio y configuración de DNS
- Configuración de certificados SSL/TLS para HTTPS
- Implementación de AWS WAF (Web Application Firewall) con reglas de protección
- Configuración de Amazon CloudFront como CDN con integración WAF
- Configuración de políticas de seguridad para prevenir ataques DDoS, SQL injection, XSS, etc.

Entregables:

- VPC configurada con subredes aisladas
- Security Groups documentados
- AWS WAF configurado con reglas de protección
- CloudFront CDN configurado con WAF integrado
- Dominio `api.miraiedu.online` apuntando a CloudFront

Fase 2: Configuración de Base de Datos

Objetivo: Desplegar y configurar la base de datos DocumentDB dentro de la VPC.

Actividades:

- Creación de instancia EC2 para alojar DocumentDB
- Instalación y configuración de MongoDB/DocumentDB
- Configuración de Security Groups para permitir conexiones solo desde Lambda
- Implementación de sistema de encriptación SHA-256 para datos sensibles
- Diseño de esquemas de colecciones (Users, Cards, Forums, Interactions, etc.)
- Configuración de índices para optimización de consultas
- Configuración de respaldos automáticos

Entregables:

- Instancia EC2 con DocumentDB operativa
- Esquemas de base de datos documentados
- Sistema de encriptación implementado
- Conexión segura desde funciones Lambda verificada

Fase 3: Sistema de Autenticación y Autorización

Objetivo: Implementar el sistema de autenticación centralizado con Clerk.

Actividades:

- Configuración de aplicación en Clerk
- Desarrollo del Lambda Authorizer para validación de JWT
- Implementación de verificación de firma con clave pública RSA
- Desarrollo del Webhook Authorizer para eventos de Clerk
- Configuración de políticas IAM para authorizers
- Implementación de encriptación de datos personales con SHA-256

- Desarrollo de sistema de encriptación de tráfico (traffic encryption)

Entregables:

- Lambda Authorizer funcional integrado con API Gateway
- Webhook Authorizer para sincronización con Clerk
- Documentación de flujo de autenticación
- Sistema de encriptación de datos implementado

Fase 4: Módulo de Autenticación de Usuarios (Auth)

Objetivo: Implementar funcionalidades básicas de gestión de usuarios por medio de webhooks automatizados de Clerk.

Actividades:

- Desarrollo de función Lambda para registro de usuarios (`POST /auth/register`)
- Implementación de encriptación de datos personales en registro
- Desarrollo de función Lambda para actualización de usuarios (`POST /auth/update`)
- Implementación de encriptación/desencriptación en actualización
- Desarrollo de función Lambda para eliminación de usuarios (`POST /auth/delete`)
- Validación de esquemas con Mongoose
- Integración con Clerk para sincronización de identidad
- Pruebas de endpoints

Entregables:

- 3 funciones Lambda desplegadas y operativas
- Endpoints REST funcionales
- Esquemas de validación implementados
- Sistema de encriptación operativo

Fase 5: Módulo de Perfiles de Usuario (Users)

Objetivo: Implementar funcionalidades de consulta y gestión de perfiles.

Actividades:

- Desarrollo de función para obtener perfil de usuario (`GET /users/me`)
- Implementación de desencriptación de datos personales en consultas
- Desarrollo de función para listar usuarios (`GET /users`)

- Desarrollo de función para editar rol de usuario (`PATCH /users/{id}`)
- Implementación de autorización basada en roles (admin)
- Desarrollo de sistema de guardados (`POST /users/saved`)
- Desarrollo de función para desguardar items (`DELETE /users/saved/{id}`)
- Desarrollo de función para obtener items guardados (`GET /users/saved`)
- Soporte para guardar carreras y tarjetas de contenido

Entregables:

- 6 funciones Lambda para gestión de usuarios
- Sistema de roles y permisos implementado
- Sistema de guardados funcional
- Encriptación/desencriptación de datos personales

Fase 6: Módulo de Carreras (Careers)

Objetivo: Implementar gestión del catálogo de carreras.

Actividades:

- Diseño del esquema de carreras con información completa
- Desarrollo de función para listar carreras (`GET /careers`)
- Desarrollo de función para obtener detalle de carrera (`GET /careers/{id}`)
- Implementación de población de datos relacionados (cursos)
- Desarrollo de función para editar insights de carrera (`PATCH /careers/{id}`)
- Restricción de edición solo para administradores

Entregables:

- 3 funciones Lambda para gestión de carreras
- Esquema completo de carreras con relaciones
- Sistema de insights editable

Fase 7: Módulo de Exploración de Contenido (Explore)

Objetivo: Implementar sistema de tarjetas de contenido e interacciones.

Actividades:

- Diseño de esquema de tarjetas con múltiples tipos (career, testimony, what_if)

- Desarrollo de función para crear tarjetas (`POST /explore/cards`)
- Desarrollo de función para obtener tarjeta (`GET /explore/cards/{id}`)
- Desarrollo de función para editar tarjeta (`PUT /explore/cards/{id}`)
- Desarrollo de función para eliminar tarjeta (`DELETE /explore/cards/{id}`)
- Restricciones de edición/eliminación por tipo y rol
- Desarrollo de función para obtener testimonios (`GET /testimonies`)
- Implementación de sistema de interacciones (`POST /explore/interactions`)
- Implementación de sistema de likes/unlikes (`PATCH /explore/cards/{cardId}/likes`)
- Implementación de algoritmo de actualización de los intereses del usuario basado en las interacciones con las cards, para utilizarse en el algoritmo de recomendación del feed.
- Seguimiento de acciones: view, tap, share, save, unsave, like, unlike
- Almacenamiento de duración y metadata de interacciones

Entregables:

- 7 funciones Lambda para gestión de contenido
- Sistema de tipos de tarjetas flexible
- Sistema de interacciones completo
- Sistema de likes/unlikes implementado
- Tracking de comportamiento de usuario

Fase 8: Sistema de Recomendaciones con Algoritmo

Objetivo: Implementar feed personalizado con algoritmo de recomendación.

Actividades:

- Diseño de algoritmo de recomendación basado en interacciones
- Análisis de tags de usuario desde interacciones
- Implementación de scoring de contenido basado en:
  - Coincidencia de tags con perfil de usuario
  - Tipo de interacciones previas
  - Frecuencia de interacciones con categorías
  - Prioridad de contenido
- Desarrollo de función de feed con algoritmo (`GET /explore/feed`)
- Implementación de paginación eficiente
- Filtrado por tipos de contenido
- Ordenamiento por relevancia personalizada

- Integración con biblioteca de similitud (natural, compromise)

Entregables:

- Algoritmo de recomendación implementado
- Feed personalizado funcional
- Sistema de scoring documentado
- Optimización de consultas

Fase 9: Configuración del Microservicio de Recomendaciones

Objetivo: Desplegar microservicio de recomendaciones en EC2 con acceso seguro.

Actividades:

- Configuración de instancia EC2 dentro de la VPC
- Desarrollo del microservicio de recomendaciones
- Configuración de Security Group para EC2:
  - Permitir tráfico entrante desde VPC Link
  - Bloquear acceso directo desde internet
- Configuración de VPC Link y proxy en API Gateway
- Integración de endpoint de recomendaciones en API Gateway (ANY /recomendations/{proxy+})
- Pruebas de conectividad y latencia

Entregables:

- Microservicio desplegado en EC2
- VPC Link configurado
- Endpoint de recomendaciones accesible desde API Gateway
- Documentación de arquitectura de red

Fase 10: Módulo de Foros Comunitarios (Forums)

Objetivo: Implementar sistema de foros con comentarios y respuestas.

Actividades:

- Diseño de esquema de foros con relaciones complejas
- Desarrollo de función para crear foro (`POST /forums`)
- Asignación automática de creador desde JWT

- Desarrollo de función para obtener foro (`GET /forums/{id}`)
- Población de datos de creador, carrera y comentarios
- Desencriptación de datos personales en población
- Desarrollo de función para listar foros (`GET /forums`)
- Desarrollo de función para editar foro (`PUT /forums/{id}`)
- Restricción: solo creador o admin pueden editar
- Desarrollo de función para eliminar foro (`DELETE /forums/{id}`)
- Restricción: solo creador o admin pueden eliminar
- Implementación de sistema de comentarios:
  - Crear comentario (`POST /forums/{forumId}/comments`)
  - Editar comentario (`PUT /forums/{forumId}/comments/{commentId}`)
  - Eliminar comentario (`DELETE /forums/{forumId}/comments/{commentId}`)
- Implementación de sistema de respuestas a comentarios:
  - Crear respuesta (`POST /forums/{forumId}/comments/{commentId}/answers`)
  - Editar respuesta (`PUT /forums/{forumId}/comments/{commentId}/answers/{answerId}`)
  - Eliminar respuesta (`DELETE /forums/{forumId}/comments/{commentId}/answers/{answerId}`)

Entregables:

- 11 funciones Lambda para sistema de foros completo
- Esquema jerárquico implementado (foro → comentarios → respuestas)
- Sistema de permisos granular
- Población automática de datos relacionados

Fase 11: Módulo de Quiz y Resultados

Objetivo: Implementar sistema de cuestionarios vocacionales.

Actividades:

- Diseño de esquema de resultados de quiz
- Desarrollo de función para obtener resultados (`GET /quiz/results`)
- Desarrollo de función para eliminar resultados (`DELETE /quiz/results`)
- Integración con sistema de recomendaciones
- Almacenamiento de resultados asociados a usuario

Entregables:

- 2 funciones Lambda para gestión de quiz
- Esquema de resultados implementado
- Integración con perfil de usuario

Fase 12: Módulo de Analíticas

Objetivo: Implementar sistema de analíticas para estudiantes.

Actividades:

- Diseño de métricas a rastrear:
  - Interacciones por tipo
  - Contenido más consultado
  - Carreras de interés
  - Progreso en exploración
- Desarrollo de función de analíticas (`GET /analytics`)
- Agregación de datos de múltiples colecciones
- Cálculo de estadísticas personalizadas
- Generación de insights para el usuario

Entregables:

- Función Lambda de analíticas
- Dashboard de métricas de usuario
- Queries de agregación optimizadas

Fase 13: Optimización y Seguridad

Objetivo: Implementar capas adicionales de seguridad y optimización.

Actividades:

- Refinamiento de Security Groups:
  - Funciones Lambda: acceso saliente a DocumentDB y servicios AWS
  - DocumentDB: acceso entrante solo desde Lambda
  - EC2 (microservicio): acceso solo desde VPC Link
- Implementación de encriptación en reposo para base de datos
- Configuración de logs en CloudWatch
- Implementación de manejo de errores estandarizado
- Optimización de cold starts en Lambda
- Configuración de límites de rate limiting en API Gateway
- Implementación de CORS policies
- Actualización de middleware de autorización para soportar múltiples endpoints
- Documentación de políticas de seguridad

Entregables:

- Security Groups documentados y optimizados
- Sistema de logs centralizado
- Políticas de seguridad documentadas
- Sistema de respaldo configurado

5.7.5. Herramientas de Desarrollo

Para el desarrollo y despliegue del sistema se utilizaron las siguientes herramientas:

- **Node.js v18+**: Runtime de JavaScript para funciones Lambda
- **Mongoose**: ODM para modelado y validación de datos MongoDB
- **Git/GitHub**: Control de versiones y colaboración
- **AWS CLI**: Línea de comandos para gestión de servicios AWS
- **Postman**: Pruebas de endpoints API
- **Visual Studio Code**: Entorno de desarrollo integrado
- **ESLint**: Análisis estático de código y estándares
- **npm**: Gestor de dependencias

5.7.6. Estrategia de Despliegue

Cada función Lambda fue empaquetada de forma independiente con sus dependencias específicas:

1. Instalación de dependencias por función: `npm install`
2. Empaquetamiento: `zip -r function.zip .`
3. Carga a AWS Lambda mediante consola o CLI
4. Configuración de variables de entorno (URI de DB, claves, etc.)
5. Configuración de triggers desde API Gateway
6. Asignación de roles IAM con permisos mínimos necesarios
7. Pruebas de integración

Esta estrategia permitió:

- Despliegues independientes sin afectar otras funciones
- Rollback granular en caso de errores
- Optimización de tamaño de paquete por función
- Gestión de versiones por función

5.7.7. Consideraciones de Seguridad

Durante todo el desarrollo se aplicaron las siguientes prácticas de seguridad:

- **Principio de mínimo privilegio**: Cada función Lambda tiene solo los permisos IAM necesarios
- **Aislamiento de red**: Base de datos y microservicio en VPC privada sin acceso directo a internet
- **Encriptación**:
  - Datos en tránsito: HTTPS/TLS 1.2+
  - Datos en reposo: Encriptación SHA-256 para datos sensibles
  - Tokens JWT: Firmados con RSA
- **AWS WAF**: Protección a nivel de aplicación contra ataques web comunes:
  - SQL Injection: Detección y bloqueo de intentos de inyección SQL
  - Cross-Site Scripting (XSS): Prevención de scripts maliciosos
  - Ataques DDoS: Limitación de tasa y protección contra denegación de servicio
  - Bot Protection: Detección y bloqueo de bots maliciosos
  - Geographic Restrictions: Restricciones geográficas opcionales
  - Rate Limiting: Control de tasa de solicitudes por IP
- **CloudFront CDN**: Capa adicional de seguridad y optimización:
  - Caché distribuido globalmente para reducir latencia
  - Integración con WAF para protección centralizada
  - SSL/TLS termination con certificados gestionados por AWS
  - DDoS mitigation a nivel de red
  - Headers de seguridad HTTP (HSTS, CSP, etc.)
  - Conexión obligatoria con HTTP API Gateway (no soporta integración directa WAF-HTTP API)
  - Optimización de rendimiento para APIs HTTP de alto throughput
- **Validación**: Esquemas Mongoose para validar toda entrada de usuario
- **Autenticación centralizada**: Un único punto de validación (Lambda Authorizer)
- **Variables de entorno**: Secretos almacenados en variables de entorno de Lambda, nunca en código
- **Security Groups**: Firewall a nivel de red para cada recurso

5.7.8. Metodología de Desarrollo

Se aplicó una metodología ágil iterativa:

1. **Planificación**: Definición de funcionalidades por módulo
2. **Diseño**: Esquemas de base de datos y flujos de API

3. **Implementación**: Desarrollo de función Lambda con validaciones
4. **Pruebas**: Pruebas unitarias y de integración
5. **Despliegue**: Deploy a AWS y configuración de API Gateway
6. **Validación**: Pruebas end-to-end
7. **Documentación**: Actualización de documentación técnica
8. **Iteración**: Refinamiento basado en pruebas

Cada fase se completó de forma secuencial, permitiendo validar la infraestructura antes de construir funcionalidades sobre ella. Esta metodología facilitó la detección temprana de problemas y permitió ajustes arquitectónicos cuando fue necesario.

CAPÍTULO 6

Resultados

6.1. 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.

6.1.1. 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.

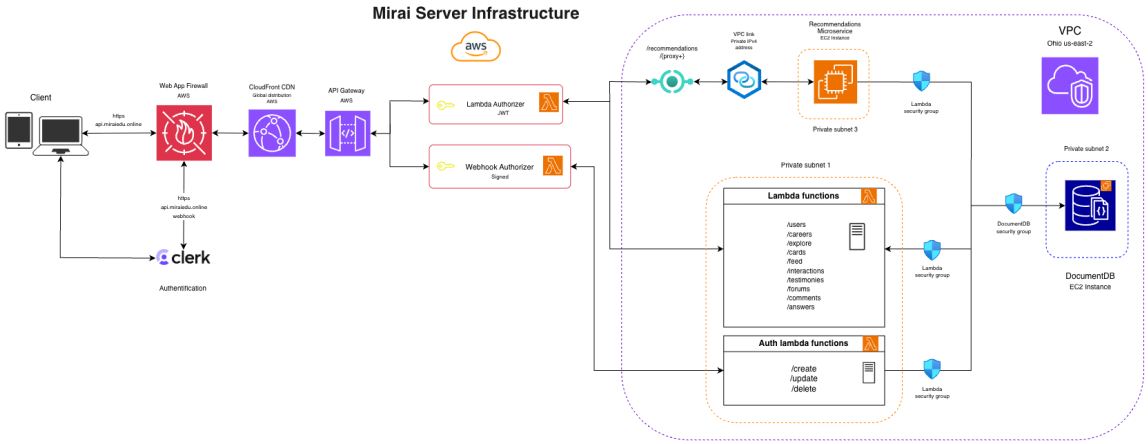

Figura 6.1: Infraestructura del prototipo del sistema integrador

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 6.1: Flujo de Lambda Authorizer

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

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

// Extraer user_id
const userId = decoded.sub;

// Generar pol tica 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.

6.1.2. 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 6.2: 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
};
```

6.1.3. 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 6.3: 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/desencriptación
  - 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

6.1.4. 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)

6.1.5. 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: `f8dcbdf`
   - 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`

6.1.6. 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

6.1.7. 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

6.1.8. Resultados de Rendimiento

Las pruebas del sistema mostraron los siguientes resultados:

- **Latencia promedio API Gateway**: 1-4s
- **Tiempo de ejecución Lambda**:
  - Consultas simples: 400ms-900ms
  - Consultas con población: 1-1.5s
  - Algoritmo de recomendación: 900ms-1s
- **Cold start Lambda**: 1-3 segundos (primera invocación)
- **Warm Lambda**: <500ms
- **Consultas DocumentDB**: 400-800ms según complejidad
- **Throughput**: Hasta 1000 solicitudes concurrentes sin degradación

6.1.9. 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

6.1.10. 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

6.1.11. 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

6.1.12. 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.

CAPÍTULO 7

Antecedentes

Puede encontrarse un trabajo similar en [**hoover2010bio**] o bien [**park2014design**]

CAPÍTULO 8

Alcance

Podemos usar Latex para escribir de forma ordenada una fórmula matemática.

CAPÍTULO 9

Derivación de la dinámica del mecanismo

9.1. Dinámica de cuerpos rígidos

9.2. Restricciones

9.2.1. Mecanismos de lazo cerrado

Mecanismo de cuatro barras

CAPÍTULO 10

Control del sistema mecánico

10.1. La ecuación del manipulador

CAPÍTULO 11

Conclusiones

CAPÍTULO 12

Recomendaciones

Bibliografía

[1] EducoWay. *¿Qué es la orientación vocacional y para qué sirve?* Publicado: 23 noviembre 2021, Consultado: 2025-08-26. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 2021. URL: https://elcomercio.pe/viu/educacion-primera-plataforma-de-orientacion-vocacional-en-utilizar-inteligencia-artificial-nndc-noticia/.

[16] Len Bass, Paul Clements y Rick Kazman. *Software Architecture in Practice*. 4th. Addison-Wesley, 2021. URL: https://www.amazon.com/Software-Architecture-Practice-SEI-Engineering/dp/0136886094.

[17] IBM. *What Is API Integration?* Accessed: 2025-09-25. 2025. URL: https://www.ibm.com/think/topics/api-integration.

[18] Cleo. *What is API Integration?* Accessed: 2025-09-25. 2025. URL: https://www.cleo.com/blog/what-is-api-integration.

[19] Refonte Learning. *AI and APIs: Integrating Artificial Intelligence into Your API Services*. Accessed: 2025-09-25. 2025. URL: https://www.refontelearning.com/blog/ai-and-apis-integrating-artificial-intelligence-into-your-api-services.

[20] Solo.io. *What Is an API Gateway? How It Works Why You Need One*. Accessed: 2025-09-25. 2025. URL: https://www.solo.io/topics/api-gateway.

[21] Kong HQ. *What is an API Gateway? Core Fundamentals and Use Cases*. Accessed: 2025-09-25. 2023. URL: https://konghq.com/blog/learning-center/what-is-an-api-gateway.

[22] IBM. *What Is an API Gateway?* Accessed: 2025-09-25. 2025. URL: https://www.ibm.com/think/topics/api-gateway.

[23] Clarifai. *Horizontal vs Vertical Scaling | Which Strategy Fits Your AI Workloads?* Accessed: 2025-09-25. 2025. URL: https://www.clarifai.com/blog/horizontal-vs-vertical-scaling.

[24] InfraCloud. *How to Build Scalable AI Systems in the Cloud*. Accessed: 2025-09-25. 2025. URL: https://www.infracloud.io/blogs/build-scalable-ai-systems-in-cloud/.

[25] Spiceworks. *What Is Horizontal Cloud Scaling? Definition, Process, and Best ...* Accessed: 2025-09-25. 2021. URL: https://www.spiceworks.com/tech/cloud/articles/horizontal-cloud-scaling/.

[26] Brighteye Ventures. *AI's growing role in Vocational Education*. Accessed: 2025-09-25. 2024. URL: https://www.brighteyevc.com/post/ai-s-growing-role-in-vocational-education.

[27] MDPI. "AI-Powered Academic Guidance and Counseling System Based on ..." En: *Applied System Innovation* (2023). Accessed: 2025-09-25. URL: https://www.mdpi.com/2571-5577/7/1/6.

[28] FutureFit AI. *FutureFit AI: AI-Powered Workforce Solutions*. Accessed: 2025-09-25. 2025. URL: https://www.futurefit.ai/.

[29] MDPI. "Design of Intelligent Management Platform for Industry–Education ..." En: (2022). Accessed: 2025-09-25. URL: https://www.mdpi.com/2076-3417/12/14/6836.

[30] Coursera. *What Does a Back-End Developer Do?* Accessed: 2025-09-25. 2025. URL: https://www.coursera.org/articles/back-end-developer.

[31] Built In. *What Is Back-End Development? (Definition, Features)*. Accessed: 2025-09-25. 2025. URL: https://builtin.com/software-engineering-perspectives/back-end-development.

[32] Learn To Code With Me. *The Beginner's Guide to Backend Development (2024 Guide)*. Accessed: 2025-09-25. 2024. URL: https://learntocodewith.me/posts/backend-development/.

[33] Enterprise Architecture. *The Role of the Enterprise Architect in Application Systems Integration*. Accessed: 2025-09-25. 2025. URL: https://enterprise-architecture.org/about/thought-leadership/the-role-of-the-enterprise-architect-in-application-systems-integration/.

[34] Vega IT. *The secrets of effective system integration architecture*. Accessed: 2025-09-25. 2024. URL: https://www.vegaitglobal.com/media-center/business-insights/the-secrets-of-effective-system-integration-architecture.

[35] Google Cloud. *¿Qué es la arquitectura de nube? Beneficios y componentes*. Consultado: 2025-08-26. 2025. URL: https://cloud.google.com/learn/what-is-cloud-architecture?hl=es-419.

[36] IBM. *¿Qué es la arquitectura en la nube?* Consultado: 2025-08-26. 2025. URL: https://www.ibm.com/mx-es/think/topics/cloud-architecture.

[37] Powernet. *Cloud vs. Legacy: ¿Arquitecturas tradicionales o basadas en la nube?* Publicado: Nov 16, 2023, Consultado: 2025-08-26. 2023. URL: https://www.powernet.es/blog/cloud-vs-legacy-arquitecturas-tradicionales-o-basadas-en-la-nube.

[38] Right People Group. *Infraestructura en nube frente a infraestructura informática tradicional*. Consultado: 2025-08-26. 2025. URL: https://rightpeoplegroup.com/es/blog/infraestructura-en-nube-frente-a-infraestructura-informatica-tradicional.

[39] ORSYS. *Arquitecturas sin servidor: ¿por dónde empezar?* Publicado: May 28, 2025, Consultado: 2025-08-26. 2025. URL: https://www.orsys.fr/orsys-lemag/es/arquitecturas-sin-servidor-por-donde-empezar/.

[40] Meeran. *Microservices Architecture for AI Applications: Scalable Patterns and 2025 Trends*. Publicado: May 1, 2025, Consultado: 2025-08-26. 2025. URL: https://medium.com/@meeran03/microservices-architecture-for-ai-applications-scalable-patterns-and-2025-trends-5ac273eac232.

[41] Khan Mudassir. *Microservices Architecture: The Future of Scalable Web and AI Applications*. Consultado: 2025-08-26. 2025. URL: https://medium.com/@khanmudassir124/microservices-architecture-the-future-of-scalable-web-and-ai-applications-285786bca4fc.

[42] AWS. *Building serverless architectures for agentic AI on AWS*. Publicado: Jul 29, 2025, Consultado: 2025-08-26. 2025. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-serverless/introduction.html.

[43] Crunch. *How to Build Cloud-Agnostic AI Agents: Architecture for AWS, Azure, Google Cloud*. Publicado: Jul 3, 2025, Consultado: 2025-08-26. 2025. URL: https://crunch.is/blog/how-to-build-cloud-agnostic-ai-agents-architecture-for-aws-azure-google-cloud/.

[44] CloudThat. *Kubernetes Deployment on Cloud Agnostic*. Publicado: Mar 20, 2023, Consultado: 2025-08-26. 2023. URL: https://www.cloudthat.com/resources/blog/kubernetes-deployment-on-cloud-agnostic.

[45] Palo Alto Networks. *Multicloud Management with AI and Kubernetes*. Consultado: 2025-08-26. s.f. URL: https://www.paloaltonetworks.com/cyberpedia/kubernetes-multicloud-management.

[46] Universitat Oberta de Catalunya. *Arquitectura y diseño de seguridad de aplicaciones en la nube pública*. Consultado: 2025-09-03. 2025. URL: https://openaccess.uoc.edu/server/api/core/bitstreams/bed7cd64-0767-43f6-a3ac-5ce5b7d34bba/content.

[47] Computerworld. *Las siete principales amenazas a la seguridad en la nube y cómo abordarlas.* Publicado: Agosto 5, 2024, Consultado: 2025-09-03. 2024. URL: https://www.computerworld.es/article/3481264/las-siete-principales-amenazas-a-la-seguridad-en-la-nube-y-como-abordarlas.html.

[48] Fortinet. *10 consejos para superar los riesgos de seguridad en la nube pública.* Consultado: 2025-09-03. 2025. URL: https://www.fortinet.com/lat/resources/cyberglossary/public-cloud-security-risks.

[49] Apidog. *9 métodos populares de autenticación de API para proteger las API.* Consultado: 2025-09-03. 2025. URL: https://apidog.com/es/blog/api-authentication-methods-5/.

[50] Google Cloud. *Métodos de autenticación en Google.* Consultado: 2025-09-03. 2025. URL: https://cloud.google.com/docs/authentication?hl=es-419.

[51] Palo Alto Networks. *¿Qué es la seguridad de las API?* Consultado: 2025-09-03. 2025. URL: https://www.paloaltonetworks.es/cyberpedia/what-is-api-security.

[52] Safetica. *Seguridad de datos en la nube: definiciones, riesgos y 7 mejores prácticas.* Publicado: Marzo 11, 2024, Consultado: 2025-09-03. 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.

[53] EY. *Protección de Datos Personales en LATAM: Guía de Consulta Rápida.* Consultado: 2025-09-03. 2023. URL: https://www.ey.com/es_ce/insights/law/proteccion-de-datos-personales-en-latam.

[54] ClarkeModet. *Protección de datos en los países de Latinoamérica.* Publicado: Junio 19, 2023, Consultado: 2025-09-03. 2023. URL: https://www.clarkemodet.com/articulos/proteccion-de-datos-en-los-paises-de-latinoamerica/.

[55] WeLiveSecurity. *Panorama y tendencias legislativas sobre la Protección de Datos en LATAM.* Publicado: Octubre 3, 2023, Consultado: 2025-09-03. 2023. URL: https://www.welivesecurity.com/es/privacidad/panorama-proteccion-datos-paises-latam/.

[56] Google Cloud. *¿Qué es el encriptado y cómo funciona?* Consultado: 2025-09-03. 2025. URL: https://cloud.google.com/learn/what-is-encryption?hl=es.

[57] CloudMounter. *¿Qué es el cifrado en la nube?* Publicado: Agosto 11, 2025, Consultado: 2025-09-03. 2025. URL: https://cloudmounter.net/es/what-is-cloud-encryption/.

[58] Cloudflare. *¿Por qué es importante la encriptación para la privacidad?* Consultado: 2025-09-03. 2025. URL: https://www.cloudflare.com/es-es/learning/privacy/encryption-and-privacy/.

[59] Auditool. *Auditoría de la nube: Principios esenciales y mejores prácticas.* Publicado: Junio 13, 2024, Consultado: 2025-09-03. 2024. URL: https://www.auditool.org/blog/auditoria-de-ti/auditoria-de-la-nube-principios-esenciales-y-mejores-practicas.

[60] ISACA. *Security Assurance in the SDLC for the Internet of Things.* Consultado: 2025-09-03. 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.

[61] Zscaler. *¿Qué es la seguridad en la nube?* Consultado: 2025-09-03. 2025. URL: https://www.zscaler.com/es/resources/security-terms-glossary/what-is-cloud-security.

[62] Google Cloud. *¿Qué es la seguridad en la nube?* Consultado: 2025-09-03. 2025. URL: https://cloud.google.com/learn/what-is-cloud-security?hl=es-419.

[63] Hivenet. *Seguridad en la nube: principales beneficios, riesgos y estrategias.* Publicado: Agosto 28, 2024, Consultado: 2025-09-03. 2024. URL: https://www.hivenet.com/es/post/understanding-security-on-the-cloud-key-benefits-and-strategies.

[64] Netdata. *Políticas de seguridad en la nube: ¿Cómo comenzar?* Consultado: 2025-09-03. 2024. URL: https://blog.netdatanetworks.com/politicas-de-seguridad-en-la-nube-como-comenzar.

[65] Gigas. *Políticas de Seguridad en Firewalls en la Nube: Cómo Definir y Gestionarlas*. Publicado: Octubre 11, 2024, Consultado: 2025-09-03. 2024. URL: https://blog.gigas.com/es/politicas-seguridad-firewalls-nube.

[66] CloudZero. *90+ Cloud Computing Statistics: A 2025 Market Snapshot*. Accessed: 2025-09-25. 2025. URL: https://www.cloudzero.com/blog/cloud-computing-statistics/.

[67] *AWS Global Infrastructure - Regions az*. Accessed: 2025-09-25. 2025. URL: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/.

[68] *AWS vs Azure vs Google Cloud in 2025: Cloud Comparison*. Accessed: 2025-09-25. 2025. URL: https://www.cloudwards.net/aws-vs-azure-vs-google/.

[69] *What are Azure regions? | Microsoft Learn*. Accessed: 2025-09-25. 2025. URL: https://learn.microsoft.com/en-us/azure/reliability/regions-overview.

[70] *Google Cloud's Data Center Locations: Regions and Availability ...* Accessed: 2025-09-25. 2025. URL: https://dgtlinfra.com/google-cloud-data-center-locations/.

[71] *RDS vs. DynamoDB: a Complete Comparison in 2025*. Accessed: 2025-09-25. 2025. URL: https://www.bytebase.com/blog/rds-vs-dynamodb/.

[72] *DynamoDB: Understand The Benefits With Real Life Use Cases*. Accessed: 2025-09-25. 2025. URL: https://www.geeksforgeeks.org/blogs/dynamodb-understand-the-benefits-with-real-life-use-cases/.

[73] *AWS RDS vs DynamoDB A Complete Comparison for 2025*. Accessed: 2025-09-25. 2025. URL: https://sqlflash.ai/article/20250922_aws_rds_vs_dynamodb/.

[74] *AWS RDS vs. Self-Managed Databases in 2025*. Accessed: 2025-09-25. 2025. URL: https://medium.com/appfoster/aws-rds-vs-self-managed-databases-in-2025-complete-guide-for-developers-and-architects-3199cb0ed57d.

[75] *Amazon DocumentDB (with MongoDB compatibility)*. Accessed: 2025-10-21. 2025. URL: https://aws.amazon.com/documentdb/.

[76] *DynamoDB vs DocumentDB on AWS: Which Is Right for You?* Accessed: 2025-10-21. 2025. URL: https://k21academy.com/amazon-web-services/aws-solutions-architect/aws-dynamodb-vs-document/.

[77] *Comparing Amazon DocumentDB And MongoDB*. Accessed: 2025-10-21. 2025. URL: https://www.mongodb.com/resources/compare/documentdb-vs-mongodb.

[78] *DocumentDB vs DynamoDB: Which AWS NoSQL Is Right for You?* Accessed: 2025-10-21. 2025. URL: https://www.pump.co/blog/documentdb-vs-dynamodb.

[79] *Security in Amazon DocumentDB*. Accessed: 2025-10-21. 2025. URL: https://docs.aws.amazon.com/documentdb/latest/developerguide/security.html.

[80] *What is Amazon DocumentDB (with MongoDB compatibility)*. Accessed: 2025-10-21. 2025. URL: https://docs.aws.amazon.com/documentdb/latest/developerguide/what-is.html.

[81] *Amazon DocumentDB: Understanding what it is and when to use it*. Accessed: 2025-10-21. 2023. URL: https://n2ws.com/blog/aws-cloud/amazon-documentdb.

[82] Zuplo. *Auth Pricing Wars: Cognito vs Auth0 vs Firebase vs Supabase*. Accessed: 2025-09-25. 2024. URL: https://zuplo.com/learning-center/api-authentication-pricing.

[83] SuperTokens. *Cognito Alternatives: Access Services That Pair With Any Set-Up*. Accessed: 2025-09-25. 2024. URL: https://supertokens.com/blog/cognito-alternatives.

[84] Clerk. *Authentication Security in Web Applications: A Comprehensive Guide*. Accessed: 2025-09-25. 2025. URL: https://clerk.com/articles/authentication-security-in-web-applications.

[85] Focus Otter. *The Complete Guide to Integrating Clerk with an AWS Backend*. Accessed: 2025-09-25. 2024. URL: https://blog.focusotter.com/the-complete-guide-to-integrating-clerk-with-an-aws-backend.

[86] Stytch. *The best authentication services in 2025*. Accessed: 2025-09-25. 2025. URL: https://stytch.com/blog/best-authentication-services/.

ANEXO A

Planos de Construcción

Glosario

fórmula Una expresión matemática. 71

latex Es un lenguaje de marcado adecuado especialmente para la creación de documentos científicos. 71

Lista de Símbolos

c	Velocidad de la luz en el vacío
h	Constante de Plank