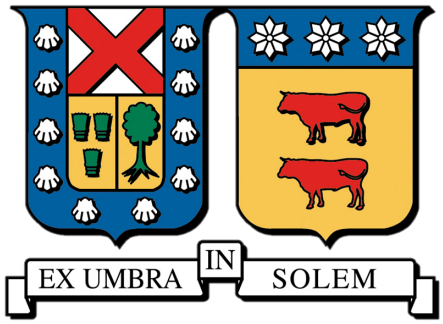
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"DESARROLLO DE MÓDULO DE PROMPT ENGINEERING Y ARQUITECTURA CLOUD BASADA EN COMPONENTES SERVERLESS PARA PLATAFORMA EDUCATIVA CON IA"

MARCELO ESTEBAN DÍAZ MOYA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA: **MAURICIO ARAYA**

PROFESOR CORREFERENTE: **PATRICIO OLIVARES**

OCTUBRE 2025

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: DESARROLLO DE MÓDULO DE PROMPT ENGINEERING Y ARQUITECTURA CLOUD BASADA EN COMPONENTES SERVERLESS PARA PLATAFORMA EDUCATIVA CON IA

Nombre del candidato(a): Marcelo Esteban Díaz Moya

Carrera / Grado: Ingeniería Civil Telemática

Campus: Casa Central Valparaiso **;** **Departamento**: Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Mauricio Araya López, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 03/10/2025 ; **Firma:** _____

Estudiante o Candidato(a):

Fecha: 03/10/2025 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que fueron parte de este proceso, tanto en lo educativo como en lo personal. Este camino, aunque desafiante, fue posible gracias al apoyo, la dedicación y las enseñanzas que recibí a lo largo del recorrido. Cada reto superado me brindó valiosos aprendizajes, no solo a nivel técnico, sino también en la vida misma. Sin la ayuda y el impulso de quienes me acompañaron, este logro no habría sido posible.

A mi familia, por acompañarme y estar presente en cada etapa de mi vida. En especial, agradezco profundamente a mis padres, Marcelo y Sandra, por todo lo que me han enseñado y por el gran apoyo incondicional durante este proceso. A mi hermana Nicole, por estar siempre presente y motivarme a ser una mejor persona.

A mi pareja Rocío, gracias por tu apoyo constante, tu compañía y por acompañarme también en los momentos más difíciles de este camino, haciéndolo más llevadero a lo largo de estos años universitarios.

A mis amigos, por el apoyo, los momentos compartidos y las horas de estudio en conjunto que hicieron este camino más llevadero.

Finalmente, agradezco a mi profesor guía y a cada uno de los docentes que, con sus conocimientos y consejos, aportaron de manera significativa a mi formación durante esta etapa universitaria.

Resumen

Este documento presenta el desarrollo de PedagogIA, una plataforma educativa orientada a optimizar la carga laboral docente mediante herramientas de inteligencia artificial. El trabajo se enfoca en dos componentes técnicos fundamentales: el módulo de ingeniería de prompts y la arquitectura backend basada en componentes serverless.

La investigación aborda la problemática que surge de la convergencia entre la carga laboral docente y la creciente demanda de personalización educativa inclusiva. El estado del arte examina técnicas de prompt engineering y arquitecturas serverless, estableciendo el fundamento teórico para las decisiones de implementación.

Se desarrolló un módulo de ingeniería de prompts aplicando metodología iterativa human-in-the-loop, implementando técnicas few-shot con plantillas estructuradas adaptadas al sistema educativo chileno. Se crearon nueve servicios educativos especializados, cada uno con plantillas específicas para la generación de material personalizado.

La arquitectura utiliza el patrón API Gateway + Lambda con estrategia dual de bases de datos (PostgreSQL en RDS y DynamoDB), proporcionando escalabilidad automática y eficiencia operativa. Las pruebas de concurrencia validaron el rendimiento con tiempos de respuesta de 101ms para autenticación y 3.5 segundos promedio para generación de contenido.

Los resultados confirman la viabilidad técnica de aplicar prompt engineering sistemático en contextos educativos específicos, logrando alta efectividad en la generación de documentos válidos. La arquitectura agnóstica del LLM permite intercambiabilidad entre proveedores sin modificaciones estructurales.

El proyecto demuestra que la convergencia entre ingeniería de software e inteligencia artificial puede generar soluciones contextualizadas para problemáticas educativas reales, estableciendo un modelo replicable para aplicación de tecnologías emergentes.

Keywords: Prompt Engineering, Arquitectura Serverless, Inteligencia Artificial Educativa, AWS Lambda, Plataforma Educativa, Sistema Educativo Chileno

Glosario

Planificación educativa: La planificación educativa se refiere al proceso mediante el cual los docentes organizan y estructuran sus actividades de enseñanza, incluyendo la elaboración de planes de estudio, la secuenciación de contenidos y la programación de actividades didácticas [1].

Gestión Administrativa: La gestión administrativa en el contexto educativo abarca la organización y coordinación de las tareas administrativas relacionadas con la docencia, como el manejo de horarios, la administración de evaluaciones y la comunicación con los padres y otros miembros de la comunidad educativa [2].

Cliente: Es una entidad que adquiere productos o servicios ofrecidos por una organización.

Usuario: Un usuario se refiere a un individuo que interactúa con un producto o servicio.

Inteligencia Artificial: Campo de la informática que se dedica al desarrollo de sistemas y algoritmos capaces de realizar tareas que tradicionalmente requieren inteligencia humana, como el razonamiento, el aprendizaje, la percepción y la toma de decisiones [3].

LLM (Large Language Model): Tipo específico de modelo de inteligencia artificial entrenado con grandes volúmenes de datos textuales, especializado en la comprensión y generación de lenguaje natural de manera coherente y contextualmente relevante.

API: Es el acrónimo de Interfaz de Programación de Aplicaciones (Application Programming Interface en inglés). Se refiere a un conjunto de reglas y herramientas que permiten a diferentes aplicaciones y sistemas comunicarse entre sí. Las API permiten que distintos programas informáticos se integren y compartan datos de manera eficiente y segura.

Frontend: Corresponde al apartado visual del desarrollo de una aplicación, es la capa con la que interactúa el usuario.

Backend: Corresponde al apartado del manejo de los datos en el desarrollo de una

aplicación, aquí reside toda la lógica del negocio.

Prompt: Instrucción o entrada que se le proporciona a un sistema, generalmente a una herramienta de Inteligencia Artificial.

Prompt Engineering: Disciplina que se enfoca en el diseño, optimización y refinamiento de instrucciones (prompts) para obtener respuestas más precisas, relevantes y útiles de los modelos de inteligencia artificial, especialmente de los modelos de lenguaje.

Cloud Computing: Modelo de computación que permite el acceso bajo demanda a un conjunto compartido de recursos informáticos configurables (redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser aprovisionados y liberados rápidamente con un esfuerzo mínimo de administración.

Serverless: Modelo de computación en la nube donde el proveedor administra automáticamente la infraestructura subyacente, permitiendo a los desarrolladores enfocarse únicamente en el código de la aplicación sin preocuparse por la gestión de servidores.

AWS (Amazon Web Services): Plataforma de servicios de computación en la nube que ofrece una amplia gama de servicios de infraestructura como almacenamiento, bases de datos, análisis, redes, entre otros.

Few-Shot Learning: Técnica de aprendizaje automático donde se proporcionan pocos ejemplos (típicamente 2-5) dentro del prompt para que el modelo aprenda el patrón deseado y genere respuestas consistentes con los ejemplos proporcionados.

Lambda (AWS Lambda): Servicio de computación serverless de AWS que ejecuta código en respuesta a eventos sin necesidad de aprovisionar o administrar servidores, cobrando únicamente por el tiempo de cómputo utilizado.

DynamoDB: Base de datos NoSQL completamente administrada de AWS que proporciona rendimiento rápido y predecible con escalabilidad automática, especialmente adecuada para aplicaciones que requieren baja latencia.

API Gateway: Servicio completamente administrado de AWS que facilita a los desarrolladores crear, publicar, mantener, monitorear y proteger APIs a cualquier escala, actuando como puerta de entrada para aplicaciones backend.

VPC (Virtual Private Cloud): Red virtual lógicamente aislada en AWS que permite lanzar recursos en un entorno de red virtual definido por el usuario, proporcionando control sobre el entorno de red.

CloudWatch: Servicio de monitoreo y observabilidad de AWS que proporciona datos e información procesable para monitorear aplicaciones, responder a cambios de rendimiento y optimizar la utilización de recursos.

Taxonomía de Bloom: Marco pedagógico que clasifica objetivos educativos en seis niveles cognitivos jerárquicos: Conocimiento, Comprensión, Aplicación, Análisis, Síntesis y Evaluación, utilizado para estructurar actividades de aprendizaje según la complejidad cognitiva requerida.

Índice de figuras

Índice de cuadros

Índice general

Capítulo 1

Introducción

Los profesores han enfrentado históricamente una carga laboral significativa que, si bien ha sido reconocida en diversos contextos educativos, aún presenta desafíos importantes en términos de optimización y gestión eficiente. A pesar de los esfuerzos realizados para abordar esta problemática, persisten dificultades relacionadas con el tiempo que dedican a planificar, generar material, actividades, evaluaciones, revisiones, etc.

Esta complejidad se ha intensificado considerablemente con el creciente énfasis en la educación inclusiva, un paradigma que ha adquirido mayor relevancia en las últimas décadas. Actualmente, las tareas docentes deben adaptarse no solo a una cantidad importante de alumnos, sino también a sus diversas capacidades, estilos de aprendizaje y necesidades específicas, lo que multiplica considerablemente el esfuerzo requerido para brindar una atención educativa personalizada y equitativa [4].

Por otro lado, existe una brecha digital importante en los profesores, especialmente en aquellos de edad más avanzada. Esta brecha se refiere a la falta de habilidades y conocimientos necesarios para hacer uso de las herramientas tecnológicas modernas. La pandemia puso en evidencia esta falta de adaptación, ya que muchos profesores se encontraron con dificultades significativas para trasladar sus clases y actividades a las plataformas digitales existentes. La resistencia al cambio y la falta de formación adecuada han contribuido a que esta brecha se mantenga e incluso se amplíe en algunos

casos.

La problemática principal radica en la combinación entre la carga laboral de los profesores y la brecha digital. Aunque existen herramientas que podrían facilitar su trabajo, estas no se están utilizando.

Si bien los métodos tradicionales de enseñanza han demostrado su eficacia a lo largo del tiempo, la convergencia entre las demandas crecientes de personalización educativa y las oportunidades que ofrecen las tecnologías emergentes plantea un escenario propicio para explorar alternativas que puedan optimizar los procesos docentes y contribuir al bienestar profesional de los educadores.

En este contexto, el presente trabajo se enfoca en dos aspectos técnicos centrales dentro del proyecto grupal: por un lado, el diseño e implementación de un módulo de ingeniería de prompts que permita aprovechar de forma eficiente los LLM, y por otro, el desarrollo de una arquitectura cloud basada en componentes serverless que sustenta la plataforma PedagogIA.

1.1. Objetivos

1.1.1. Objetivo General

Facilitar y apoyar el trabajo docente en la planificación de cursos y generación de material diversificado para profesores de enseñanza básica y media, a través de una plataforma que integra herramientas de inteligencia artificial.

1.1.2. Objetivos Específicos

El presente trabajo de título se enfoca en dos componentes técnicos específicos dentro del proyecto grupal: el módulo de ingeniería de prompts y la arquitectura cloud basada en componentes serverless que sustenta la plataforma.

	Nombre	Descripción
OE1	**Identificación de las Necesidades Docentes y Definición de los Servicios a ofrecer**	Identificar y documentar las necesidades y desafíos principales que enfrentan los profesores. Definir los servicios específicos que se van a ofrecer en la plataforma, basados en las necesidades identificadas de los profesores.
OE2	Modelo de Negocios	Desarrollar un modelo de negocios sostenible para la plataforma.
OE3	**Modulo de manejo de prompts**	Modulo encargado de realizar un manejo efectivo de los prompts que se vayan a enviar hacia la herramienta de inteligencia artificial, teniendo en cuenta términos monetarios y de rendimiento.
OE4	Investigación de Herramientas IA	Investigar y definir las herramientas de inteligencia artificial que se utilizarán en la plataforma, evaluando su idoneidad y eficacia.
OE5	**Diseño y Arquitectura del sistema**	Definir la arquitectura del sistema, especificando las principales funcionalidades y su interacción. Además, definir las tecnologías a utilizar.
OE6	Diseño e implementación interfaz de Usuario	Diseño de las vistas de la interfaz de usuario del sistema y su respectiva implementación en el frontend de la aplicación Web.
OE7	Implementación Backend	Realizar la implementación del backend y su respectiva interacción con la Base de datos, además, implementación de APIs externas y modelos personalizados.
OE8	Prototipado y Sistema de Retroalimentación	Implementación de los módulos de backend y frontend, para luego crear diversos niveles de prototipo con usuarios, procediendo a iteraciones basadas en el feedback continuo de los usuarios para recopilar comentarios y sugerencias de mejora, garantizando así la evolución constante de la plataforma.
OE9	Establecimiento de Alianzas Estratégicas	Establecer alianzas estratégicas con instituciones educativas, editoriales u otras organizaciones relevantes para ampliar el alcance y la adopción de la plataforma.
OE10	Definición del Manual de Usuario y Capacitaciones	Definir el manual de usuario y planificar las capacitaciones para garantizar una adecuada adopción y uso efectivo de la plataforma por parte de los profesores.
OE11	**Diseño e Implementación del Módulo de Ingeniería de Prompts**	Desarrollar un módulo especializado que implemente técnicas avanzadas de prompt engineering para optimizar la interacción con modelos de lenguaje en el contexto educativo chileno, incorporando plantillas estructuradas, técnicas few-shot y construcción modular de prompts.
OE12	**Arquitectura Cloud Basada en Componentes Serverless**	Diseñar e implementar una arquitectura backend basada en componentes serverless en AWS (API Gateway + Lambda) complementada con servicios administrados, aprovechando los beneficios de escalabilidad automática, modelo pay-per-use y abstracción de infraestructura para el procesamiento dinámico.
OE13	**Aplicación Práctica del Estado del Arte**	Trasladar las técnicas teóricas identificadas en el estado del arte de prompt engineering y arquitecturas basadas en componentes serverless a una implementación práctica que genere valor en el contexto educativo específico.
OE14	**Arquitectura Agnóstica del LLM**	Desarrollar una arquitectura modular que permita independencia del proveedor de LLM, facilitando la migración entre diferentes modelos y optimización continua de costos y rendimiento.

Cuadro 1.1: Tabla de objetivos específicos

Este trabajo de memoria se centra en el cumplimiento de los objetivos específicos OE11–OE14, vinculados directamente con dos de los componentes técnicos del proyecto: el módulo de ingeniería de prompts y la arquitectura cloud basada en componentes serverless que constituye la infraestructura de la plataforma.

Capítulo 2

Estado del Arte

En este capítulo se presenta una revisión de la literatura relacionada con tres ámbitos directamente vinculados al proyecto: en primer lugar, las plataformas de apoyo docente con inteligencia artificial, con el fin de identificar funcionalidades, limitaciones y buenas prácticas que sirvan de referencia para el diseño del sistema; en segundo lugar, el módulo de ingeniería de prompts; y en tercer lugar, la arquitectura cloud basada en componentes serverless que soporta la plataforma.

Los papers y referencias fueron seleccionados mediante búsqueda sistemática en bases de datos académicas, priorizando publicaciones recientes (2021-2024) sobre plataformas educativas con IA, técnicas avanzadas de prompt engineering y arquitecturas serverless. El enfoque se centró en metodologías aplicables al proyecto, evaluando eficiencia técnica y viabilidad en un entorno educativo.

El análisis se divide en tres secciones: la primera examina plataformas de apoyo docente con IA, describiendo sus principales servicios y aspectos diferenciadores; la segunda revisa avances en ingeniería de prompts, incluyendo optimización, automatización y gestión de recursos para el módulo de la plataforma; y la tercera aborda arquitecturas cloud y componentes serverless, estudiando patrones de diseño, servicios de AWS y estrategias de implementación que sustentan la infraestructura.

Esta revisión identifica técnicas relevantes, oportunidades de mejora y brechas exis-

tentes, proporcionando la base teórica para las decisiones de diseño e implementación del proyecto.

2.1. Plataformas de apoyo docente con IA

La herramienta más popular corresponde a MagicSchool [5], plataforma de habla inglesa, que cuenta con más de 50 herramientas para ofrecer. Entre las herramientas que más destacan se encuentran la generación de rúbricas, evaluaciones, preguntas relacionadas con textos y videos de YouTube, entre otras. En esta plataforma, los profesores tienen un perfil en el cual indican los cursos que imparten y las asignaturas que enseñan. Otros puntos importantes a destacar son la generación de programas individualizados para los alumnos que lo requieran.

La siguiente plataforma es la única que encontramos en español, la cual corresponde a MegaProfe [6]. Esta plataforma también cuenta con una gran cantidad de herramientas, como la generación de planificaciones, evaluaciones, rúbricas, etc. Dichas herramientas se ven limitadas en la versión gratuita, afectando el número de tokens mensuales (caracteres totales obtenidos). Esta plataforma se destaca en la implementación de chatbots específicos por asignaturas. Por otro lado, otra prestación relevante es la opción de copiar el documento generado para poder editarlo directamente en caso de necesitar un cambio. Es importante destacar que esta web, según las reseñas, no es del todo intuitiva y requiere cierto conocimiento para poder hacer un mejor uso de ella.

Siguiendo con el análisis, la siguiente plataforma es TeachMateAI [7], donde también se encuentran una cantidad importante de herramientas similares a las mencionadas anteriormente: generación de planificaciones, evaluaciones, rúbricas, etc. Sin embargo, la característica destacada es un apartado donde los profesores pueden proponer ideas para nuevas herramientas, lo cual consideramos muy importante, ya que se obtiene una retroalimentación constante de parte de los profesores. Por otro lado, un aspecto negativo de esta plataforma es su diseño, ya que no es nada amigable a la vista y además, presenta un proceso poco intuitivo al ingresar el prompt y obtener la salida.

En la plataforma de Eduaide [8] podemos percatarnos de la presencia de los servicios mencionados en las plataformas anteriores, pero con la salvedad de encontrarnos con una interfaz que peca de sobrecargar la pantalla de información, esto por la elección de asignaturas, nivel y servicio en una misma vista. Esta propuesta resalta en la entrega de un servicio estandarizado de acceso igualitario para todos, pero con una propuesta diferenciadora nula y que provoca una interfaz menos amigable e intuitiva para el usuario.

En resumen, hemos observado que la mayoría de las plataformas se centran en ofrecer servicios similares, como generación de evaluaciones, actividades y rúbricas, entre otros. Sin embargo, cada una añade prestaciones específicas que las diferencian entre sí. Por otro lado, este análisis nos permitió comprender de mejor manera el impacto que tiene una interfaz amigable para el usuario.

2.2. Ingeniería de Prompts

En el estudio presentado en [9], se comparan dos técnicas para manejar los prompts. La primera técnica, denominada “0-shot”, consiste en enviar la solicitud al modelo sin proporcionar ejemplos previos. La segunda técnica, “few-shot”, implica proporcionar ejemplos al modelo para que pueda aprender de ellos antes de generar una respuesta. El estudio concluye que la técnica “0-shot” puede igualar o incluso superar el rendimiento de “few-shot”, lo que sugiere que los modelos ya poseen el conocimiento necesario para completar la tarea. Además, se destaca que este enfoque es útil porque permite al modelo operar de manera efectiva sin necesidad de ejemplos adicionales, lo que ahorra tiempo y esfuerzo en la preparación de datos y reduce el número de consultas al modelo.

En [10] se presenta una solución al manejo de prompts utilizando LLMs mediante el método Auto-CoT (Automatic Chain of Thought). A diferencia de Manual-CoT, Auto-CoT supera en rendimiento al automatizar el proceso de razonamiento. Este método consta de dos etapas: primero, agrupa preguntas similares en clústeres usando técnicas como Sentence-BERT y k-means; luego, genera automáticamente demostraciones de

resolución para cada clúster, mejorando así la eficiencia y precisión en diversas tareas.

Continuando con [11], se presentan diversos métodos de Ingeniería de Prompts para mejorar el razonamiento, algunos relacionados a lo mencionado en el paper anterior (CoT). Dentro de los métodos, destacan técnicas como el Árbol de Pensamientos (ToT), que extiende la Chain of Thought con una estructura de árbol; el Gráfico de Pensamientos (GoT), que facilita la interacción dinámica entre ideas; el Ingeniero de Prompts Automático (APE), que automatiza la generación y selección de prompts; y el Razonamiento Automático y Uso de Herramientas (ART), que integra herramientas externas para resolver problemas complejos. Además, introduce la Cadena de Pensamiento Contrastiva (CCoT) para enseñar a los modelos mediante ejemplos válidos e inválidos y la Cadena de Conocimiento (CoK), que descompone tareas complejas en pasos estructurados.

Por otro lado, en [12] se presenta una visión más general de la ingeniería de prompts, comenzando por destacar por qué es esencial una comunicación efectiva con los modelos de LLMs. Además, se mencionan diversas técnicas para enfrentar este problema. Entre ellas, una de las más destacadas, que no está presente en los demás papers, es el uso de plantillas para proporcionar formatos estructurados para las salidas, asegurando consistencia y calidad.

En [13] se presenta un análisis completo de los diversos desafíos actuales con los LLMs, destacando especialmente el tema de la “Optimización a Nivel de Datos” en el manejo de prompts. El paper explora tres técnicas clave para mejorar esta optimización. La primera, Prompt Pruning, consiste en eliminar elementos, oraciones o documentos no relevantes para cada entrada del modelo, reduciendo así la carga computacional sin afectar el rendimiento, como se ve en el método DYNAICL. La segunda técnica, Prompt Summary, busca condensar el prompt original en una versión más corta mientras conserva la información semántica esencial, utilizando métodos como RECOMP para transformar el prompt completo en un conjunto resumido de elementos clave. Finalmente, Soft Prompt-based Compression se enfoca en diseñar un prompt suave, más corto y ajustable durante el entrenamiento, con técnicas como PromptCompression que

ajustan estos prompts suaves según los datos de entrenamiento.

Por último, en [14] se presenta un estudio de 39 técnicas de Ingeniería de Prompts (algunas ya mencionadas en los papers anteriores), que abarcan desde métodos simples, como entregar el prompt tal cual llega, hasta técnicas más avanzadas utilizando modelos automáticos. Además, tras presentar estas técnicas, se mencionan diversos datasets para evaluar los diferentes métodos y categorizarlos según distintas tareas de procesamiento de lenguaje natural.

En este estado del arte se han revisado y comparado diversas técnicas de Ingeniería de Prompts aplicadas en LLMs. Los estudios analizados destacan desde enfoques básicos, como el "0-shotz "few-shot", hasta métodos avanzados como Auto-CoT, que automatiza el razonamiento, y técnicas de optimización de prompts que mejoran la eficiencia computacional. Además, se identificaron metodologías innovadoras como la Cadena de Pensamiento Contrastiva (CCoT) y el Árbol de Pensamientos (ToT), que refuerzan la capacidad de los modelos para resolver tareas complejas. En conjunto, estas técnicas no solo mejoran la calidad de las respuestas, sino que también optimizan el uso de recursos, lo que subraya la importancia creciente de la Ingeniería de Prompts y la necesidad de incorporarlo en nuestro proyecto.

2.3. Cloud Computing y Arquitecturas Serverless

El Cloud Computing permite el acceso a recursos informáticos como almacenamiento, cómputo, redes, IA, entre otros, bajo demanda con un modelo de pago por uso, eliminando la necesidad de infraestructura local, lo que aporta flexibilidad, escalabilidad y reducción en los costos, en comparación con infraestructuras tradicionales [15]. Además, las plataformas que usan LLM necesitan recursos bastante elevados en términos de potencia y almacenamiento, lo que la nube proporciona de forma dinámica y eficiente.

Dentro del paradigma de Cloud Computing, el modelo emergente serverless representa una mejora importante, la cual abstrae completamente la infraestructura, llevando

a los desarrolladores a centrarse en el código, delegando al proveedor Cloud la administración de servidores, escalado, disponibilidad y actualizaciones [16]. Las ventajas técnicas principales incluyen la escalabilidad automática, donde el proveedor Cloud ajusta dinámicamente la capacidad, dependiendo del servicio, según eventos o cargas, sin intervención manual [17], permitiendo que las aplicaciones se adapten automáticamente a las cambiantes demandas [17]. El modelo de pago por uso se factura en función del tiempo efectivo de cómputo, evitando costos por ”tiempo muerto”[18], con modelos de precios pay-per-use donde solo se paga por los recursos que se consumen [18]. Adicionalmente, la menor complejidad operativa resulta de delegar la automatización de infraestructura, reduciendo la carga de mantenimiento y acelerando el desarrollo [16], eliminando la gestión del lado del servidor [16]. Los patrones de diseño más comunes, como API Gateway + Lambda para microservicios o funciones activadas por eventos (subida de archivos, notificaciones, etc.), favorecen un diseño modular y escalable con despliegue rápido de funciones [18].

La fundamentación técnica específica en el proveedor AWS demuestra ventajas concretas frente a modelos tradicionales. En [19] implementan 9 prototipos de la misma aplicación de microservicios utilizando diferentes tecnologías, comparando desde un entorno de despliegue tradicional (Kubernetes) hasta una migración completa a arquitectura serverless combinando AWS ECS Fargate, AWS Lambda y DynamoDB, hallando que la versión serverless mejora la escalabilidad, agiliza el desarrollo y reduce los costos operativos gracias al modelo de pago según consumo real [19]. En términos de optimización y rendimiento en Lambda, [20] presenta las investigaciones más recientes sobre estrategias para mejorar el funcionamiento de estas funciones, incluyendo la gestión eficiente de recursos, la selección del entorno de ejecución más adecuado y el monitoreo del sistema, técnicas que permiten obtener mayor eficiencia, rentabilidad y escalabilidad [20]. Por su parte, [21] realiza una evaluación práctica de rendimiento y costos mediante pruebas detalladas que comparan la infraestructura serverless de AWS con alternativas tradicionales para el procesamiento de grandes volúmenes de datos. Este estudio identifica las limitaciones específicas de rendimiento en redes y almacena-

miento serverless, pero también establece los puntos donde la infraestructura serverless resulta más eficiente económicamente para el procesamiento de datos [21].

En conclusión, el uso de Cloud Computing y su evolución hacia las arquitecturas serverless en AWS representan una evolución que maximiza los beneficios de la nube mediante la abstracción completa de la infraestructura, ofreciendo escalabilidad automática, eficiencia económica y simplicidad operativa. Estas características resultan especialmente ventajosas para el desarrollo y despliegue de plataformas web modernas, ya que permiten reducir significativamente los tiempos de desarrollo, minimizar la complejidad de la gestión de infraestructura y optimizar los costos operativos mediante un modelo de pago basado únicamente en el uso real de los recursos.

Capítulo 3

Desarrollo de la plataforma

Este capítulo presenta el desarrollo de la plataforma PedagogIA, cubriendo tanto los aspectos grupales como las contribuciones técnicas individuales. Se estructura en cuatro secciones: requerimientos del sistema, arquitectura general, módulo de ingeniería de prompts y arquitectura cloud serverless.

Los requerimientos establecen las bases funcionales y técnicas del sistema. La arquitectura describe la evolución del diseño hacia componentes serverless y las tecnologías implementadas. Las dos secciones finales detallan los componentes técnicos desarrollados individualmente, demostrando la aplicación de técnicas del estado del arte adaptadas al contexto educativo chileno.

3.1. Requerimientos del Sistema

3.1.1. Requerimientos funcionales

A continuación, se presentan los requerimientos funcionales, los cuales definen las capacidades específicas que debe poseer el sistema para satisfacer las necesidades de los usuarios finales. Estos requerimientos fueron identificados a partir del análisis de las problemáticas docentes planteadas en la introducción y las funcionalidades necesarias para abordar la carga laboral y la brecha digital en el ámbito educativo. Su propósito

es garantizar que la plataforma proporcione todas las herramientas esenciales para la planificación, creación de material y gestión educativa.

1. Autenticación de usuarios

   El sistema permite a los establecimientos educacionales crear cuentas para sus profesores, el inicio de sesión de las mismas y el restablecimiento de contraseñas. Pudiendo discernir el establecimiento al que corresponda cada usuario.

2. Interfaz de usuario

   El sistema proporciona una plataforma Web donde el usuario podrá acceder a los servicios ofrecidos mediante una interfaz visual.

3. Servicio de planificación

   El sistema proporciona un servicio de planificación de asignatura y/o unidades en base a un prompts por parte del usuario.

4. Servicio de creación de material

   El sistema proporciona un servicio de creación de material arbitrario, ya sea generación de evaluaciones como de actividades, esto en base a prompts por parte del usuario.

5. Gestión de cursos

   El sistema proporciona un servicio de administración de cursos, donde se gestionarán las actividades realizadas, actividades futuras y el avance del curso/materia.

6. Soporte y capacitación

   El sistema proporciona una serie de videos a modo de capacitación para los usuarios y además una sección correspondiente a preguntas frecuentes y soporte.

3.1.2. Requerimientos no funcionales

Los requerimientos no funcionales establecen las características de calidad y restricciones operativas que debe cumplir el sistema. Estos requerimientos fueron definidos

considerando las expectativas de rendimiento, seguridad y disponibilidad necesarias para un entorno educativo profesional, así como los desafíos técnicos específicos del manejo de inteligencia artificial y arquitecturas en la nube. Su objetivo es asegurar que la plataforma opere de manera confiable, eficiente y segura.

- Rendimiento

  El sistema realizará el proceso de login en un tiempo inferior a 3.3[s], además, tendrá un tiempo de consulta a los servicios generativos definido por el tamaño de los datos generados.

- Seguridad

  El sistema se acoge a los estándares de seguridad informática presentes en el país.

- Disponibilidad

  El sistema brindará un servicio continuado los 365 días del año con una tasa de disponibilidad del 99 %.

- Escalabilidad

  El sistema abordará el apartado de escalamiento en base a la demanda actual.

- Manejo eficiente de prompts

  Debido a que para la generación de material se hace uso constante de prompts, es importante contar con un manejo efectivo de los mismos, tanto en términos monetarios como de rendimiento.

3.1.3. Requerimientos de interfaces

Los requerimientos de interfaces describen las interacciones específicas entre el usuario y el sistema, definiendo los eventos que desencadenan las funcionalidades de la plataforma y las respuestas esperadas. Estos requerimientos fueron establecidos considerando la experiencia de usuario y los flujos de trabajo típicos de los profesores

al utilizar herramientas educativas digitales. Su propósito es asegurar una interacción intuitiva y eficiente que facilite la adopción de la plataforma por parte de los docentes.

Evento	**Descripción**	**Iniciador**	**Respuesta**
Inicio de sesión	Al ingresar a la web PedagogIA.cl el usuario (profesor) debe ingresar sus credenciales para autenticarse.	Usuario ingresa sus credenciales.	Permiso para acceder a la web, ingreso a la vista principal.
Selección de curso	El usuario debe seleccionar el curso al cual desea acceder para una posterior generación de material.	Usuario hace clic en alguno de los cursos en pantalla.	Acceso al curso y vista de herramientas ofrecidas.
Selección de servicio	El usuario debe seleccionar el servicio que desea utilizar, ya sea generación de evaluaciones, material, planificación, entre otros.	Usuario hace clic en alguno de los servicios en pantalla.	Acceso al servicio y vista del servicio específico.
Generación de planificaciones	El usuario debe indicar los datos necesarios para generar su planificación.	Ingreso de los datos solicitados al usuario.	Generación de la planificación y acceso a la vista del material generado.
Generación de evaluaciones	El usuario debe indicar los datos necesarios para generar su evaluación.	Ingreso de los datos solicitados al usuario.	Generación de la evaluación y acceso a la vista del material generado.
Generación de actividades y tareas	El usuario debe indicar los datos necesarios para generar su actividad/tarea.	Ingreso de los datos solicitados al usuario.	Generación de la actividad/tarea y acceso a la vista del material generado.
Descarga material generado	El usuario al obtener el material, tiene la opción de descargarlo.	Clic en editar o descargar.	Descarga del PDF generado.
Edición de material generado	El usuario al obtener el material, puede editarlo.	Edición respectiva al documento.	Cambio del documento y opción de descarga.

Cuadro 3.1: Eventos externos

Respuesta	**Descripción**	**Parámetros**
Permiso para acceder a la web, ingreso a la vista principal.	El usuario tiene permitido acceder a las herramientas ofrecidas en PedagogIA.cl.	Usuario, contraseña, ID usuario, Token de autenticación.
Acceso al curso y vista de herramientas ofrecidas.	Tras seleccionar el curso, el usuario procede a seleccionar el tipo de material que desea generar.	ID usuario, ID curso, ID servicio seleccionado.
Acceso al servicio y vista del servicio específico.	Tras elegir el servicio, el usuario pasa a la vista para generar el material.	Tipo de servicio, ID servicio, parámetros del servicio.
Generación de la planificación y acceso a la vista del material generado.	Se genera la planificación solicitada por el usuario, el flujo continúa con el acceso a la vista de material generado.	ID curso, detalles planificación.
Generación de la evaluación y acceso a la vista del material generado.	Se genera la evaluación solicitada por el usuario, el flujo continúa con el acceso a la vista de material generado.	ID curso, detalles evaluación.
Generación de la actividad/tarea y acceso a la vista del material generado.	Se genera la actividad/tarea solicitada por el usuario, el flujo continúa con el acceso a la vista de material generado.	ID curso, detalles actividad/-tarea.
Descarga del PDF generado.	El usuario obtiene un PDF con el material generado.	ID documento, formato de descarga.
Cambio del documento y opción de descarga.	El usuario realiza el cambio en el documento y tiene la opción de descargarlo.	ID documento, detalles del cambio, formato de descarga.

Cuadro 3.2: Respuestas del sistema

3.1.4. Requerimientos de ambiente

Para los requisitos de ambiente, se necesita un servidor que soporte el despliegue completo de una aplicación web, incluyendo frontend, backend y bases de datos. Este servidor puede ser alojado localmente o en la nube.

3.2. Arquitectura del sistema

A continuación, se detalla la arquitectura del sistema, presentando el diseño técnico que sustenta la plataforma. Se incluye el diagrama de contexto, el esquema general del sistema, la descripción de módulos, las tecnologías seleccionadas y el entorno de desarrollo. Esta arquitectura refleja las decisiones de diseño orientadas a crear una solución escalable, eficiente y centrada en el usuario, integrando tecnologías modernas de inteligencia artificial y computación en la nube.

3.2.1. Diagrama de contexto

El diagrama de contexto de la plataforma mostrará una vista general del sistema, donde se indicarán las entidades externas y su interacción con el sistema. El diagrama se muestra a continuación:

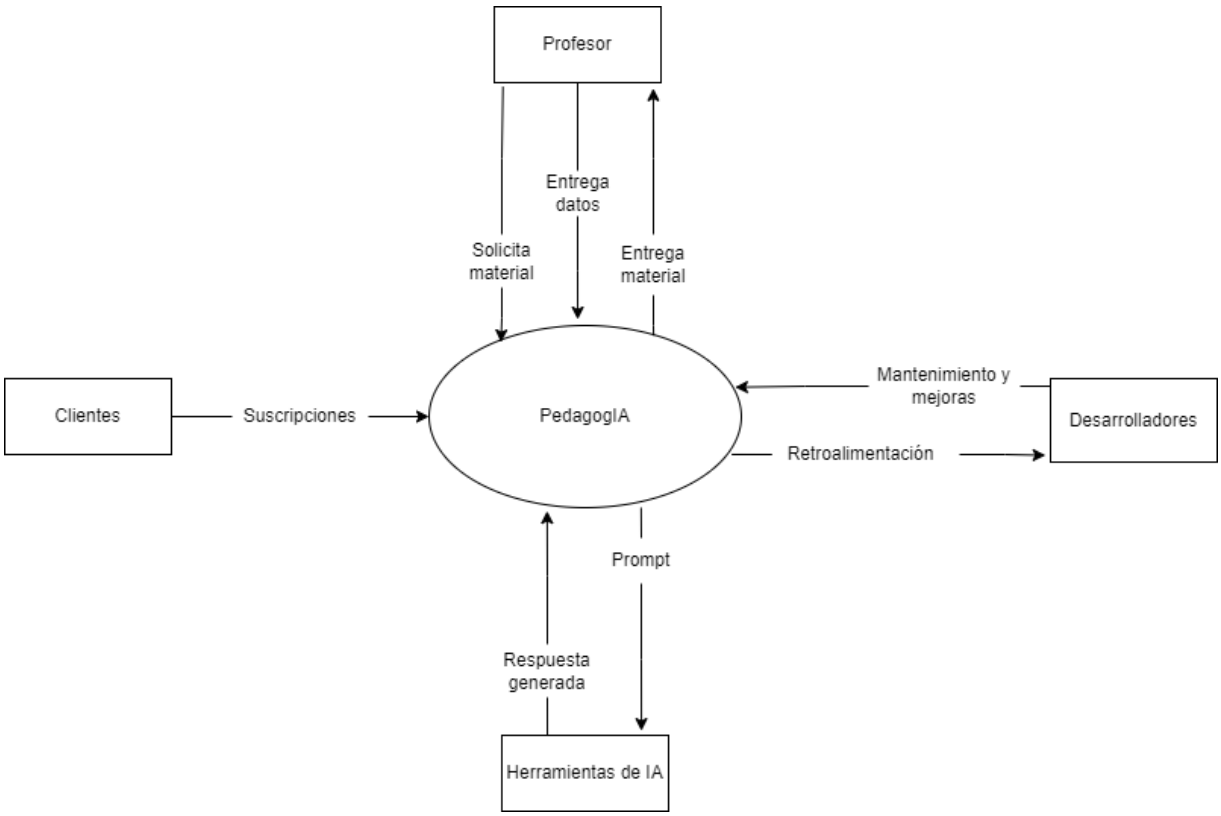


Figura 3.1: Diagrama de contexto

En el centro podemos ver al sistema, que en este caso es la plataforma PedagogIA. Este sistema se encarga de tomar ciertos datos que son proporcionados por los usuarios, estos datos sumados con otros internos de la plataforma crean una consulta, la cual es utilizada por herramientas de inteligencia artificial. Como entidades externas del sistema tenemos a:

- Profesor

  Esta entidad es el usuario final el cual hará uso del sistema (plataforma), este se encarga de solicitar alguna generación de material y también es quien entrega los datos para dicha solicitud. En consecuencia, recibe el material generado.

- Clientes

  Esta entidad abarca a instituciones las cuales se encargan de gestionar las suscripciones de sus profesores, así como las suscripciones de profesores independientes.

- Herramientas de IA

  Las herramientas de IA son el pilar del sistema, ya que, estas serán las encargadas de generar el material tras las consultas realizadas por los profesores.

- Desarrolladores

  Los desarrolladores son una de las entidades importantes, debido a que gracias a estos la plataforma se puede seguir desarrollando, implementando nuevos servicios, funcionalidades nuevas, corrección de errores, etc. El sistema proporcionará una retroalimentación hacia los desarrolladores, donde en ésta irán las opiniones y comentarios de los usuarios.

3.2.2. Esquema general del sistema

Este diagrama corresponde a la arquitectura inicial del sistema originalmente propuesta, siguiendo un enfoque convencional con separación clara entre frontend, bac-

kend y bases de datos. Esta versión representa una aproximación clásica al desarrollo de aplicaciones web con componentes bien definidos y módulos independientes.

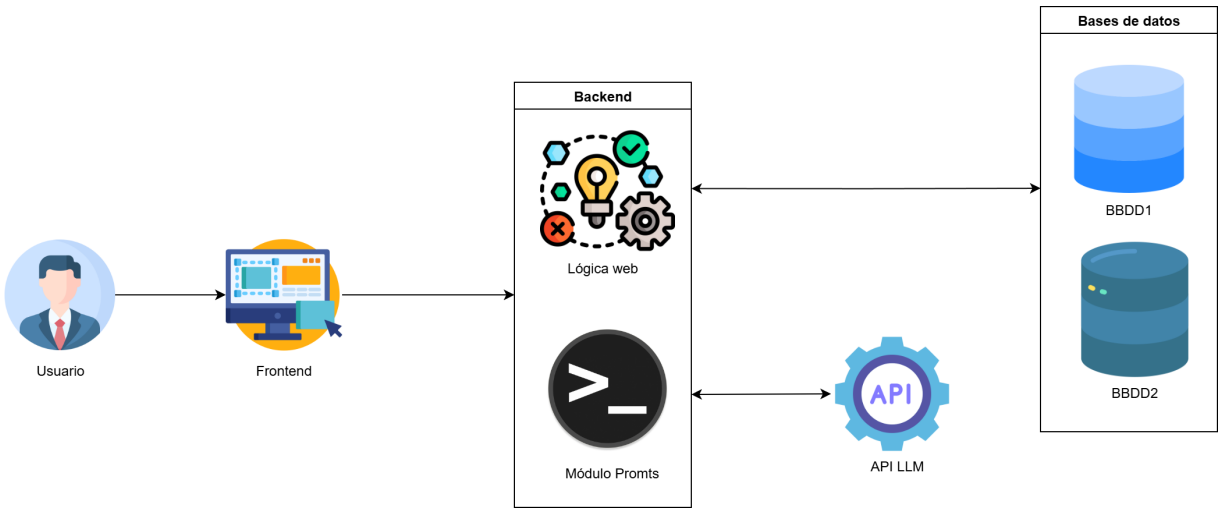


Figura 3.2: Arquitectura general

3.2.3. Actualización del esquema general del sistema

Basándose en los fundamentos teóricos revisados en el estado del arte sobre componentes serverless y sus ventajas en términos de escalabilidad, eficiencia económica y simplicidad operativa, se decidió migrar el backend del sistema hacia una arquitectura basada en componentes serverless en AWS. Esta actualización aprovecha los beneficios identificados en la literatura, implementando un diseño basado en funciones Lambda, API Gateway y servicios administrados de AWS que eliminan la necesidad de gestión de infraestructura tradicional para el procesamiento dinámico.

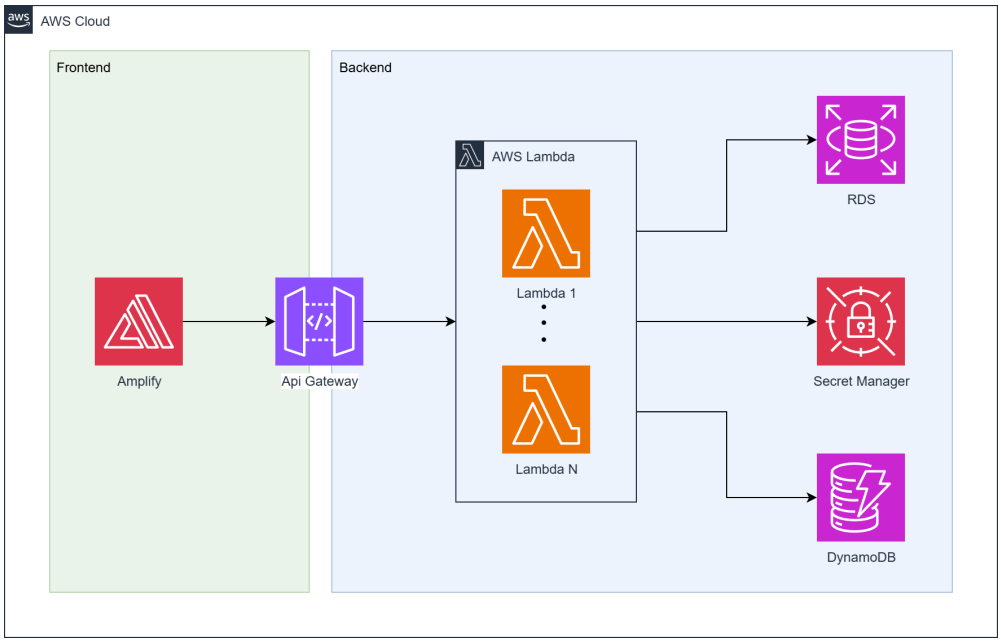


Figura 3.3: Arquitectura serverless en AWS

Esta migración arquitectónica se justifica considerando tanto las fases de desarrollo como de producción del sistema. Durante el desarrollo, la abstracción de infraestructura para el backend facilita la implementación de prácticas de despliegue continuo y permite la centralización de recursos en un entorno unificado, eliminando dependencias de servidores específicos. En producción, los componentes serverless garantizan que el sistema se adapte automáticamente a las variaciones de demanda típicas de plataformas educativas, donde el uso puede fluctuar significativamente según períodos académicos y horarios escolares.

La selección específica de AWS como proveedor se fundamenta en la combinación de experiencia técnica del equipo de desarrollo y la madurez de su ecosistema de servicios serverless. La plataforma AWS proporciona integración nativa entre sus componentes (Lambda, API Gateway, DynamoDB), lo que simplifica la implementación del módulo de manejo de prompts y optimiza la gestión de recursos computacionales necesarios para el procesamiento de solicitudes de inteligencia artificial.

Un aspecto particularmente ventajoso para el desarrollo del proyecto fue la utilización de la capa gratuita de AWS, que proporciona acceso a una amplia gama de servi-

cios y herramientas sin costo alguno. Esta capa gratuita incluye límites generosos para Lambda, API Gateway, DynamoDB, CloudWatch y otros servicios esenciales, lo cual resultó ideal para los propósitos del proyecto, permitiendo implementar y probar toda la arquitectura serverless sin incurrir en gastos operativos durante la fase de desarrollo y prototipado.

3.2.4. Descripción de módulos

La plataforma cuenta con 2 grandes módulos, conectados por un API Gateway para el funcionamiento del sistema, donde el backend cuenta con tres componentes internos. A continuación se van a describir cada uno de ellos:

Frontend

La plataforma va a contar con una interfaz gráfica agradable, intuitiva y fácil de usar. El diseño se va a centrar en la experiencia del usuario, garantizando que los profesores puedan navegar y utilizar todas las funciones disponibles sin mayores complicaciones.

Backend

Este componente es fundamental para la plataforma, ya que, será el encargado de toda la lógica del negocio, la gestión y manejo de los datos, las interacciones con la inteligencia artificial, base de datos, entre otras funciones. Estará compuesto por cuatro sub-componentes principales, que son:

- **Lógica Backend**: corresponde al motor principal de la web, donde se desarrolla la lógica general.
- **Secret**: parte del backend donde se almacenarán credenciales sensibles.
- **Base de datos SQL**: se almacenarán datos asociados a los usuarios, como por ejemplo, nombre, edad, credenciales, etc.

- **Base de datos NoSQL**: se almacenarán las consultas y respuestas de las herramientas de inteligencia artificial para su posterior manejo.

3.2.5. Tecnologías

Habiendo definido la migración hacia una arquitectura serverless en AWS y detallado los módulos que componen la plataforma, es necesario especificar las tecnologías concretas seleccionadas para la implementación de cada componente. La elección de estas tecnologías se deriva directamente de las decisiones arquitectónicas tomadas, particularmente la adopción de servicios administrados de AWS para el backend, y considera factores como la experiencia del equipo de desarrollo y la disponibilidad de recursos durante la fase de prototipado.

A continuación, se detallan las tecnologías seleccionadas para la implementación de cada componente del sistema, especificando las herramientas y servicios de AWS que sustentarán la arquitectura backend basada en componentes serverless.

Las tecnologías a utilizar para implementar en cada módulo son las siguientes:

- **Frontend**: para el desarrollo del frontend, se considera utilizar React, una biblioteca de JavaScript para construir interfaces de usuario, junto con Bootstrap, un framework CSS para crear sitios web y aplicaciones web responsivas y móviles. La elección de estas dos tecnologías se debe a que parte del equipo tiene conocimiento previo en ellas, además de que ofrecen múltiples componentes predefinidos, buena documentación y una colaboración activa entre usuarios.

  En cuanto a AWS, la tecnología a utilizar para el hospedaje y disponibilidad del frontend es AWS Amplify.

- **Backend**: en cuanto al backend, se desarrollará principalmente utilizando Python. Los motivos de su elección, al igual que para el frontend, son el conocimiento previo del equipo, su eficiencia, escalabilidad y la extensa documentación disponible, tanto oficial como de la comunidad para poder resolver problemas que vayan surgiendo.

En AWS, la tecnología utilizada para el desarrollo y la disponibilidad del backend, tanto para la lógica web como para el módulo de prompts, será mediante funciones Lambda de AWS.

- **API Gateway**: esta tencología esta disponible en AWS, y es utilizada para disponibilizar los desarrollos del back para ser consumidos por el frontend.

- **Bases de Datos**: Como se muestra en la arquitectura, se tienen dos bases de datos. La primera de ellas tiene la finalidad de almacenar los datos directos de la web, como datos de usuarios, credenciales, entre otros. Para dicha base de datos, la elección es una base de datos relacional, en AWS se utiliza RDS, donde se utilizará PostgreSQL.

  Por otro lado, la segunda base de datos tiene como objetivo almacenar todos los datos asociados a la interacción con la API, tanto solicitudes como respuestas. Esto con el fin de usar esta información para el módulo de prompts y posterior análisis estadísticos. Para esta base de datos, se tiene considerada en AWS la utilización de DynamoDB.

- **Inteligencia artificial generativa**: Son un tipo de inteligencia artificial las cuales tienen como función crear nuevos datos o contenido como texto, imágenes, vídeos, etc. Debido a su función estas serán las encargadas de realizar las tareas de generar el material personalizado que se vaya solicitando. La IA generativas a integrar en el sistema es, Gemini (Google). La elección pasa por la disponibilidad y el funcionamiento de su capa gratuita, la cual es simple y rápida de implementar.

- **Patrón de Arquitectura**: Para el desarrollo se va a seguir el patrón de arquitectura MVC (Modelo-Vista-Controlador) con el fin de tener una separación clara entre la lógica de negocio (Modelo), la interfaz de usuario (Vista) y el flujo de la aplicación (Controlador).

3.2.6. Entorno de Soporte y Desarrollo

Otra ventaja que ofrece el despliegue en la nube es la generación de un entorno de soporte y desarrollo robusto y colaborativo. Esto se logra gracias a las diferentes etapas de implementación de las APIs en API Gateway, la duplicación de funciones Lambda, y la disponibilidad de despliegues tanto para el entorno productivo como para el de soporte y desarrollo. Además, se pueden habilitar tablas en las bases de datos específicamente para propósitos de desarrollo.

Esta configuración resultó fundamental durante el proceso de desarrollo del proyecto, ya que permitió que cada integrante del equipo pudiera avanzar de manera independiente manteniendo acceso constante a los desarrollos realizados por los demás miembros. La arquitectura serverless facilitó la integración continua del trabajo, donde cada función Lambda podía ser desplegada y probada de forma aislada sin afectar el trabajo de otros componentes del sistema.

Adicionalmente, la implementación incluyó el uso de herramientas de monitoreo nativas de AWS como CloudWatch, que proporcionaron visibilidad en tiempo real sobre el funcionamiento de todos los procesos. Esta capacidad de monitoreo continuo permitió identificar y resolver problemas de rendimiento, errores en las funciones Lambda, y optimizar el consumo de recursos durante las fases de desarrollo y pruebas, asegurando así un proceso de desarrollo más eficiente y controlado.

3.3. Módulo de Ingeniería de Prompts

3.3.1. Introducción

El desarrollo del módulo de ingeniería de prompts representa una implementación práctica de las técnicas avanzadas revisadas en el estado del arte, diseñado a través de múltiples iteraciones para optimizar la interacción con modelos de lenguaje en el contexto educativo chileno. Este módulo incorpora metodologías de prompt engineering sistemáticas, manteniendo una arquitectura agnóstica del proveedor de LLM que

garantiza flexibilidad y escalabilidad.

3.3.2. Fundamentos Teóricos Implementados

La arquitectura del módulo se basa en la aplicación directa de técnicas identificadas en el estado del arte:

Técnicas Few-Shot con Plantillas Estructuradas: Siguiendo los principios establecidos por [9], el sistema implementa una aproximación few-shot mediante plantillas predefinidas que proporcionan ejemplos específicos de formato y estructura para cada tipo de documento educativo. Esta metodología se combina con el uso de plantillas estructuradas mencionado por [12], asegurando consistencia y calidad en las respuestas generadas.

Construcción Modular de Prompts: El sistema implementa una arquitectura modular donde diferentes componentes del prompt se activan según el tipo de servicio solicitado, optimizando la eficiencia al incluir únicamente la información relevante para cada contexto específico.

Refinamiento Iterativo de Prompts: Durante el desarrollo se aplicó un proceso de optimización iterativo human-in-the-loop, donde cada versión del prompt fue probada con el LLM, analizada en términos de calidad de respuesta, y refinada basándose en los resultados obtenidos, siguiendo las mejores prácticas de ingeniería de prompts identificadas en la literatura.

3.3.3. Arquitectura del Sistema

Diseño Modular y Agnóstico del LLM

Una característica fundamental del módulo es su arquitectura independiente del proveedor de LLM. Esta decisión de diseño permite:

- **Flexibilidad de proveedor**: Capacidad de migrar entre diferentes LLMs (Gemini, GPT, Claude) modificando únicamente el módulo de conexión.

- **Optimización de costos**: Posibilidad de seleccionar el proveedor más eficiente según precio y rendimiento.

Proceso de Generación de Prompts

El flujo de procesamiento implementa un enfoque sistemático basado en las prácticas más relevantes identificadas:

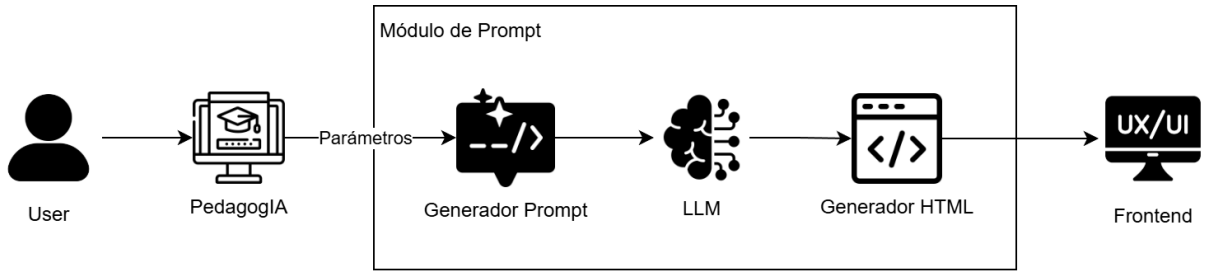


Figura 3.4: Proceso módulo de ingeniería de prompts

1. Recepción y Procesamiento de Parámetros: el sistema distingue entre parámetros implícitos (contexto educativo chileno, estructura pedagógica) y explícitos (curso, asignatura, tipo de solicitud), aplicando técnicas de contextualización específica para cada dominio educativo.

2. Generación Contextual del Prompt: utilizando las plantillas estructuradas y la construcción modular, el sistema construye prompts específicos que incorporan:

- Contexto educativo chileno adaptado por edad
- Mapeo curso-edad específico del sistema educativo nacional
- Estructura few-shot con ejemplos de formato JSON
- Instrucciones de nivel cognitivo y habilidades específicas
- Componentes modulares activados según el tipo de servicio

3. Optimización del Formato de Salida: el prompt generado incluye especificaciones técnicas para garantizar la procesabilidad posterior:

```
Contexto: esta es una solicitud realizada por un profesor, en el
    contexto de la educación escolar en Chile. A continuación, te
    detallo la solicitud para que generes lo que se pide. Los
    detalles de la solicitud son los siguientes:
Este es un requerimiento de evaluación.
- Curso: 7moB
- Edad de los alumnos que resolverán la evaluación (preocúpate de que
     sea apta para esta edad): 12
- Asignatura: Ciencias
- Detalle de la solicitud de evaluación: quiero una evaluación sobre
    la fotosíntesis.
- Tipos de preguntas: Alternativas: 10, Desarrollo: 5
Necesito la respuesta con los campos en formato JSON, ya que voy a
    transformar esta respuesta a PDF. Además, para el PDF necesito
    que se use notación HTML estándar. Es importante que los saltos
    de línea estén bien marcados, para que en la generación del PDF
    no se superpongan líneas.
Por otro lado, lo que se busca trabajar en esta petición es el nivel
    cognitivo ''Conocimiento'' y la habilidad ''Reconocer''.
Es importante que cada evaluación cumpla con el número y tipo de
    preguntas indicadas en el prompt, ya que es crítico que se cumpla
     dicho número: se necesitan 10 preguntas de alternativas y 5
    preguntas de desarrollo. Además, recuerda que necesito que el
    texto use notación HTML estándar.
Quiero que la respuesta contenga los siguientes campos, y que solo se
     incluya lo siguiente (sin acotaciones extras):
- Título ('titulo' en el JSON):
- Objetivos de la evaluación ('objetivos_evaluacion' en el JSON,
    texto con los objetivos de la evaluación):
- Instrucciones para el alumno ('instrucciones' en el JSON):
- Preguntas ('preguntas' en el JSON): El campo preguntas debe
    contener un arreglo de JSON, donde cada JSON es una pregunta.
    Dentro de cada JSON de pregunta se debe tener 'Tipo' (para saber
    a qué tipo de pregunta corresponde), 'Enunciado' (para ver el
    enunciado como tal), y 'Opciones' y 'Respuesta' en caso de
    corresponder a preguntas de ''Alternativa''. Los tipos de
    pregunta deben venir como: ''Alternativas'', ''Desarrollo'' o ''
    Verdadero o Falso''.
En caso de que necesites utilizar comillas ('') dentro de alguna
    respuesta, es necesario que vengan así: \'', ya que de otra forma
     se rompe la sintaxis de JSON.
```

3.3.4. Proceso Iterativo de Desarrollo

El desarrollo del módulo de ingeniería de prompts siguió un enfoque iterativo de mejora continua, donde cada refinamiento se basaba en el análisis de las respuestas

obtenidas del LLM y la identificación de oportunidades de optimización. Este proceso permitió evolucionar desde prompts básicos hasta una arquitectura sofisticada que incorpora múltiples técnicas del estado del arte.

Inicialmente, el sistema comenzó con prompts simples que contenían únicamente la solicitud básica del usuario. Sin embargo, las respuestas obtenidas carecían de la estructura y especificidad necesarias para el contexto educativo. Esto llevó a la incorporación progresiva del contexto educativo chileno, incluyendo la adaptación específica por edad de los alumnos y el mapeo preciso entre cursos y edades según el sistema educativo nacional.

Un aspecto fundamental del refinamiento fue la implementación de plantillas estructuradas diferenciadas por tipo de documento. Esta mejora surgió de la necesidad de garantizar consistencia en las respuestas y facilitar el procesamiento posterior. Las plantillas evolucionaron para incluir campos específicos según el tipo de material solicitado (evaluaciones, planificaciones, proyectos, etc.), incorporando ejemplos concretos de formato JSON que actúan como técnicas few-shot.

El manejo de caracteres especiales y la optimización del formato de salida representó otro hito importante en el desarrollo. La necesidad de generar documentos PDF procesables llevó a la implementación de especificaciones técnicas precisas, incluyendo el uso de notación HTML estándar y el manejo correcto de caracteres como comillas para evitar errores de sintaxis JSON.

La construcción modular del sistema emergió como una optimización natural, donde diferentes componentes del prompt se activan según el tipo de servicio solicitado. Esta aproximación no solo mejora la eficiencia al incluir únicamente información relevante, sino que también facilita el mantenimiento y la extensión del sistema.

A lo largo de todo este proceso, cada modificación fue validada mediante pruebas con el LLM, análisis de la calidad y estructura de las respuestas generadas, y ajustes basados en los resultados obtenidos. Esta metodología de refinamiento “human in the loop“ permitió alcanzar un nivel de precisión y confiabilidad adecuado para el entorno educativo objetivo.

3.3.5. Gestión de Datos y Análisis

El módulo incluye un sistema de persistencia en DynamoDB que almacena tanto las solicitudes como las respuestas, implementando un enfoque dual que sirve tanto para análisis del sistema como para funcionalidad directa del usuario. Este diseño permite:

- Historial de documentos para reutilización por parte de los profesores
- Monitoreo de patrones de uso por tipo de documento y contexto educativo
- Análisis de efectividad de las diferentes plantillas implementadas
- Identificación de oportunidades de mejora en la estructura de prompts
- Recopilación de datos para futuras optimizaciones del sistema

El módulo de ingeniería de prompts desarrollado integra técnicas del estado del arte adaptadas al contexto educativo chileno mediante una arquitectura agnóstica del LLM. La combinación de plantillas estructuradas, construcción modular y refinamiento iterativo, junto con el sistema de persistencia en DynamoDB, proporciona una base sólida para la generación de material educativo personalizado manteniendo la flexibilidad necesaria para futuras optimizaciones tecnológicas.

3.4. Arquitectura Cloud Basada en Componentes Serverless

3.4.1. Fundamentos de Diseño

El diseño de la arquitectura backend se fundamenta en los principios serverless revisados en el estado del arte, implementando una solución que facilita la escalabilidad automática, optimiza los costos mediante el modelo pay-per-use, y reduce la complejidad operativa a través de la abstracción de infraestructura. La arquitectura adopta patrones probados como API Gateway + Lambda para crear un sistema modular y altamente

escalable, complementado con servicios administrados donde se requiere persistencia relacional.

3.4.2. Arquitectura Completa del Sistema

La arquitectura implementada separa claramente las responsabilidades mediante funciones Lambda especializadas, cada una diseñada para una tarea específica dentro del ecosistema de la plataforma educativa:

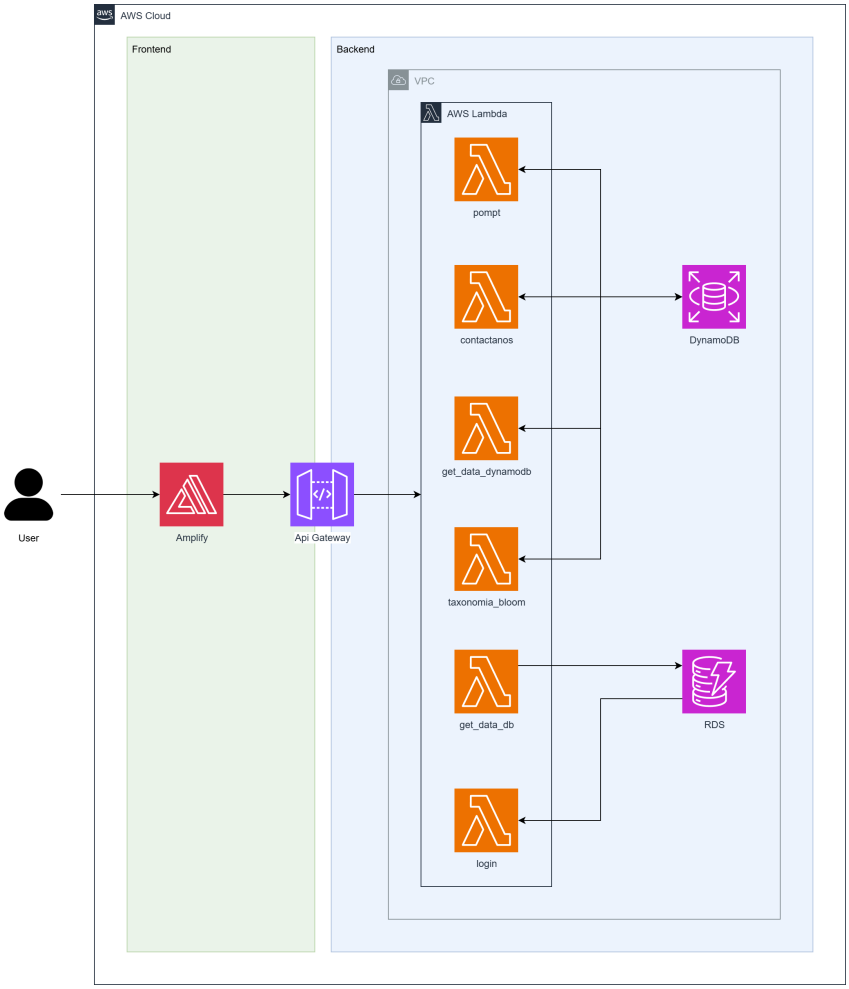


Figura 3.5: Arquitectura Cloud completa

3.4.3. Componentes y Funciones

Funciones Lambda Especializadas

El backend se estructura mediante seis funciones Lambda dentro de una VPC, cada una con responsabilidades específicas que siguen el principio de separación de responsabilidades:

- **Función prompt**: Implementa el módulo de ingeniería de prompts desarrollado, procesando las solicitudes educativas y generando respuestas estructuradas mediante técnicas de prompt engineering.
- **Función contactanos**: Gestiona la información institucional y de contacto de la plataforma.
- **Función get_data_dynamodb**: Maneja las consultas a DynamoDB para recuperar datos funcionales del sistema.
- **Función get_data_db**: Administra las consultas a la base de datos relacional para información de usuarios y credenciales.
- **Función taxonomia_bloom**: Proporciona acceso a la información de la Taxonomía de Bloom utilizada en el sistema educativo.
- **Función login**: Gestiona la autenticación y autorización de usuarios en la plataforma.

Estrategia de Bases de Datos

La arquitectura implementa una estrategia dual de bases de datos, optimizando el almacenamiento según la naturaleza de los datos:

Amazon RDS con PostgreSQL: Utilizada como base de datos relacional para almacenar información estructurada proveniente directamente de la plataforma, incluyendo datos de usuarios y credenciales.

Amazon DynamoDB: Implementada como base de datos NoSQL para gestionar información relacionada con las interacciones de la API, almacenando solicitudes, respuestas, mensajes de ayuda para potenciar los prompts, y datos vinculados a la Taxonomía de Bloom. Esta elección se fundamenta en la necesidad de escalabilidad automática y el manejo eficiente de datos semi-estructurados.

Configuración de Red y Seguridad

La arquitectura implementa una configuración de red híbrida que optimiza tanto la seguridad como el rendimiento. Las funciones Lambda y la base de datos RDS se configuran dentro de una Virtual Private Cloud (VPC), estableciendo un perímetro de seguridad que permite el acceso controlado entre estos componentes. Esta configuración es fundamental para que las funciones Lambda puedan conectarse de manera segura a la base de datos relacional PostgreSQL.

Por otro lado, DynamoDB opera fuera de la VPC como servicio completamente administrado de AWS, lo que permite un acceso más directo y eficiente desde las funciones Lambda sin la necesidad de configuración de red adicional. Esta configuración híbrida aprovecha las ventajas de ambos enfoques: la seguridad y control granular de la VPC para datos sensibles (usuarios y credenciales en RDS), y la simplicidad y rendimiento optimizado del acceso directo para datos funcionales del sistema (interacciones y prompts en DynamoDB).

Frontend y Conectividad

AWS Amplify proporciona el hosting del frontend, aprovechando su capacidad de escalabilidad automática y distribución global de contenido. La conectividad entre el frontend y el backend se establece mediante llamadas HTTP a los endpoints expuestos por API Gateway, que actúa como punto de entrada único y gestiona el enrutamiento hacia las funciones Lambda correspondientes.

3.4.4. Beneficios de los Componentes Serverless

La implementación de componentes serverless proporciona ventajas concretas alineadas con los beneficios identificados en el estado del arte:

- **Escalabilidad automática**: Las funciones Lambda y Amplify se ajustan automáticamente a la demanda sin intervención manual.
- **Optimización de costos**: El modelo pay-per-use elimina costos por recursos no utilizados, especialmente relevante para una plataforma educativa con patrones de uso variables.
- **Reducción de complejidad operativa**: La gestión de infraestructura queda delegada a AWS, permitiendo al equipo enfocarse en la lógica de negocio.
- **Alta disponibilidad**: Los servicios administrados de AWS proporcionan redundancia y recuperación automática ante fallos.
- **Flexibilidad de desarrollo**: La separación en funciones especializadas facilita el desarrollo independiente y el despliegue continuo.

Esta arquitectura demuestra la aplicación práctica de los principios serverless en un contexto educativo real, proporcionando una base técnica sólida que responde a los requerimientos de escalabilidad, seguridad y eficiencia operativa necesarios para una plataforma de inteligencia artificial educativa.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos del desarrollo e implementación de la plataforma educativa PedagogIA. Los resultados se organizan en cinco secciones principales que abarcan desde la funcionalidad general de la plataforma hasta análisis específicos de rendimiento y eficiencia de los componentes desarrollados.

La primera sección describe la plataforma educativa completa desarrollada, incluyendo sus servicios, capacidades y adaptación al contexto educativo chileno. Las siguientes secciones analizan el rendimiento del sistema bajo carga, la eficiencia del módulo de ingeniería de prompts implementado, el análisis de escalabilidad de los componentes serverless, y finalmente la cobertura funcional lograda. Estos resultados demuestran tanto la viabilidad técnica de la solución como su aplicabilidad práctica en el contexto educativo objetivo.

4.1. Servicios y Funcionalidades de PedagogIA

Se desarrolló exitosamente PedagogIA, una plataforma web completa orientada a la optimización del trabajo docente mediante herramientas de inteligencia artificial. La plataforma se encuentra completamente operativa y desplegada en la arquitectura cloud basada en componentes serverless desarrollada en AWS, proporcionando un conjunto integral de servicios para la generación de material educativo personalizado.

4.1.1. Sistema de Autenticación y Registro

La plataforma implementa un sistema de registro y autenticación específicamente diseñado para el contexto educativo chileno. Los usuarios pueden crear cuentas asociadas a instituciones educativas específicas, seleccionando su asignatura de enseñanza y curso correspondiente. Este sistema permite el control de acceso diferenciado y la personalización de contenido según el contexto institucional y académico del profesor.

4.1.2. Servicios de Generación de Material

PedagogIA ofrece nueve servicios especializados de generación de contenido educativo, sustentados por las plantillas de prompts desarrolladas que constituyen elementos clave para la generación efectiva de material personalizado:

- **Evaluación Escrita**: Creación de evaluaciones escritas en múltiples formatos (alternativas, verdadero/falso y desarrollo) con configuración personalizable de cantidad de preguntas por tipo.
- **Planificación de Clase**: Planificación de clase expositiva para el profesor.
- **Presentación**: Indicaciones para una presentación a desarrollar por los alumnos.
- **Proyecto**: Indicaciones para un proyecto de investigación a desarrollar por los alumnos.
- **Guía**: Creación de guías en múltiples formatos (alternativas, verdadero/falso y desarrollo).
- **Actividad Interactiva**: Indicaciones para una actividad interactiva a desarrollar con los alumnos.
- **Correo**: Generación de correos dado un contexto dado.
- **Texto Adaptado**: Adaptación de textos a formatos más simples.

- **Resume Texto**: Resumidor de texto indicado por profesor.

La Figura 4.1 muestra la interfaz principal donde los usuarios acceden a estos nueve servicios, organizados en categorías funcionales que facilitan la navegación y selección del tipo de material a generar.

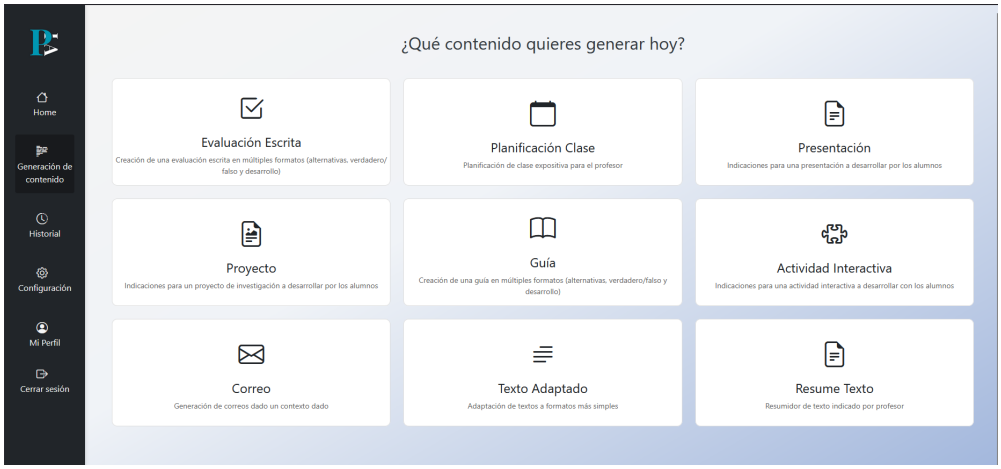


Figura 4.1: Interfaz principal de servicios disponibles en PedagogIA

4.1.3. Interfaz de Usuario y Experiencia

La plataforma cuenta con una interfaz web responsive que permite a los usuarios navegar intuitivamente entre los diferentes servicios. Cada servicio presenta formularios específicos que capturan los parámetros necesarios para la generación personalizada, incluyendo selección de curso, asignatura, nivel cognitivo según la Taxonomía de Bloom, y habilidades específicas a desarrollar.

4.1.4. Sistema de Previsualización y Descarga

Una vez generado el material, la plataforma proporciona un sistema de previsualización que permite a los docentes revisar el contenido antes de su utilización. Los documentos pueden ser descargados en dos formatos: PDF para impresión directa y DOC para edición posterior. Esta funcionalidad se simplifica considerablemente mediante el generador HTML implementado como parte del módulo de ingeniería de prompts, que

transforma las respuestas estructuradas del LLM en documentos procesables y descargables.

La Figura 4.2 ilustra el resultado del proceso de generación, mostrando la previsualización del documento generado y las opciones de descarga disponibles, demostrando la efectividad del generador HTML en la transformación de respuestas estructuradas en documentos utilizables.

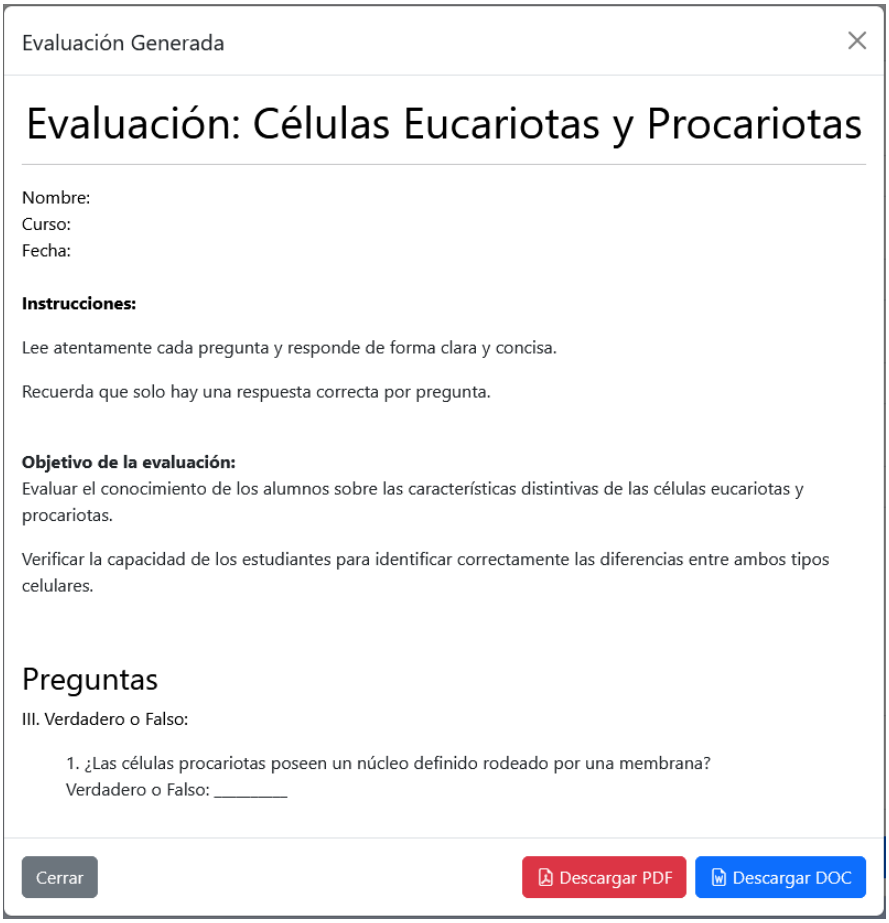


Figura 4.2: Sistema de previsualización y descarga de documentos generados

4.1.5. Gestión de Historial

La plataforma incluye un sistema de historial que permite a los usuarios acceder a documentos generados previamente, facilitando la reutilización y descarga de material educativo creado en sesiones anteriores. Esta funcionalidad se sustenta en el almacena-

miento automático de todas las solicitudes y respuestas en DynamoDB, optimizando el flujo de trabajo docente al centralizar el acceso a todos los recursos generados.

4.1.6. Arquitectura Basada en Componentes Serverless Implementada

La funcionalidad completa de la plataforma se sustenta en la arquitectura backend basada en componentes serverless desarrollada en AWS, que logra proporcionar escalabilidad automática, alta disponibilidad y eficiencia de costos. La implementación exitosa de seis funciones Lambda especializadas, junto con la estrategia dual de bases de datos (PostgreSQL en RDS y DynamoDB), permite el funcionamiento estable y eficiente de todos los servicios educativos ofrecidos.

4.1.7. Adaptación al Contexto Chileno

Todos los servicios están específicamente adaptados al sistema educativo chileno, incorporando el mapeo de cursos (desde 1° básico a 4° medio) con sus edades correspondientes. Esta especialización contextual, implementada mediante el módulo de ingeniería de prompts, asegura que el material generado sea pertinente y aplicable al entorno educativo local.

El resultado es una plataforma educativa completa y funcional que integra exitosamente tecnologías de inteligencia artificial con las necesidades específicas del contexto educativo chileno, proporcionando a los docentes herramientas eficaces para la optimización de su carga laboral.

4.2. Resultados de Rendimiento del Sistema

Para validar la capacidad de la arquitectura basada en componentes serverless desarrollada y el funcionamiento del módulo de ingeniería de prompts bajo condiciones

de carga, se realizaron dos pruebas de concurrencia que evaluaron tanto el rendimiento técnico como la experiencia de usuario de la plataforma.

4.2.1. Diseño de las Pruebas de Concurrencia

Se usaron dos enfoques de prueba para una evaluación integral del sistema:

Prueba de Concurrencia Técnica: Evaluación con 11 usuarios simultáneos simulando el flujo completo de registro y generación de material educativo, enfocada en medir la capacidad de respuesta del sistema bajo carga.

Validación con Usuarios Reales: Prueba con 9 docentes reales divididos en dos grupos diferenciados por nivel de instrucciones, orientada a evaluar tanto el rendimiento como la experiencia de usuario en condiciones realistas de uso.

4.2.2. Métricas de Rendimiento de las Funciones Lambda

Las métricas extraídas de AWS CloudWatch durante las pruebas de concurrencia demuestran el comportamiento del sistema bajo carga real. La metodología de extracción y procesamiento de estas métricas se realizó mediante la arquitectura que se detalla en la Figura 4.3 de la sección de análisis del módulo de ingeniería de prompts.

Rendimiento de la Función Login

Métrica	**Valor**
Total de ejecuciones	39
Ejecuciones exitosas	38
Ejecuciones fallidas	1
Tasa de éxito	97.4 %
Duración mínima (ms)	73.34
Duración máxima (ms)	446.79
Duración promedio (ms)	101.22
Memoria usada promedio (MB)	48.79

Cuadro 4.1: Métricas de rendimiento de la función Login en pruebas de concurrencia

La función de login demuestra un rendimiento excepcional con una tasa de éxito del 97.4 % y un tiempo de respuesta promedio de 101ms, muy por debajo del requerimiento establecido de 3.3 segundos. El percentil 95 de 110ms confirma la consistencia del sistema, mientras que el uso promedio de memoria de 49MB indica una optimización eficiente de recursos.

La única ejecución fallida se debió a un error en la lógica de consultas a la base de datos que existía durante el momento de las pruebas, el cual fue posteriormente identificado y corregido.

Rendimiento del Módulo de Ingeniería de Prompts

Métrica	**Valor**
Total de ejecuciones	22
Ejecuciones exitosas	17
Ejecuciones fallidas	5
Tasa de éxito técnica	77.3 %
Duración mínima (ms)	1.87
Duración máxima (ms)	6,473.26
Duración promedio (ms)	3,488.90
Memoria usada promedio (MB)	135.27

Cuadro 4.2: Métricas de rendimiento del módulo de ingeniería de prompts durante pruebas de concurrencia

El módulo de ingeniería de prompts presenta un tiempo de procesamiento promedio de 3.5 segundos, lo que resulta competitivo para la generación de contenido educativo mediante inteligencia artificial. Es importante destacar que las 5 ejecuciones fallidas corresponden a errores de validación de entrada (parámetros faltantes o inválidos en los formularios), no a fallos del módulo de procesamiento de prompts. Cuando el módulo recibe parámetros válidos, la tasa de éxito aumenta considerablemente, confirmando la robustez del diseño implementado.

4.2.3. Evaluación con Usuarios Reales

La prueba con 9 docentes reales proporcionó insights tanto sobre el rendimiento del sistema como sobre la experiencia de usuario. La tabla presenta comentarios reales específicos de algunos participantes que fueron seleccionados por abordar directamente los aspectos que se buscaba evaluar, junto con el tiempo promedio registrado para cada grupo:

Aspecto	**Grupo 1 (Con instrucciones)**	**Grupo 2 (Sin instrucciones)**
Percepción de velocidad	"Rápida y amigable para navegar"	"Rápidez increíble"
Usabilidad	"Muy útil para evaluaciones"	"Muy práctica y útil"
Tiempo promedio de prueba	3 minutos	8 minutos

Cuadro 4.3: Resultados de la evaluación con usuarios reales

Los resultados muestran que la plataforma es percibida como rápida y eficiente por usuarios reales. La diferencia en tiempo de completación entre grupos (3 vs 8 minutos) sugiere la importancia de la interfaz intuitiva y la documentación, mientras que ambos grupos valoraron positivamente la velocidad de respuesta del sistema.

4.2.4. Análisis de Escalabilidad

Las pruebas demuestran que la arquitectura serverless implementada cumple exitosamente con los principios de escalabilidad automática identificados en el estado del arte:

- **Manejo de carga concurrente**: El sistema procesó exitosamente múltiples usuarios simultáneos sin degradación significativa del rendimiento.
- **Escalabilidad automática**: Las funciones Lambda se escalaron automáticamente para manejar la demanda sin intervención manual.
- **Eficiencia de recursos**: El uso promedio de memoria se mantuvo en niveles óptimos (49MB para login, 135MB para procesamiento de prompts).

- **Robustez del sistema**: La diferenciación entre errores de validación y errores del módulo confirma la estabilidad del core del sistema.

Los resultados confirman que la implementación de componentes serverless proporciona la escalabilidad y rendimiento necesarios para soportar el uso educativo de la plataforma, validando las decisiones arquitectónicas basadas en el estado del arte revisado.

4.3. Eficiencia del Módulo de Ingeniería de Prompts

Para evaluar la eficiencia del módulo de ingeniería de prompts desarrollado, se implementó una arquitectura provisoria de análisis que permitió obtener métricas detalladas sobre el rendimiento y uso de los diferentes servicios durante la fase de desarrollo y pruebas.

4.3.1. Arquitectura de Análisis de Métricas

Se desarrolló una arquitectura provisoria específica para la extracción y análisis de métricas del módulo de prompts, como se muestra en la Figura 4.3.

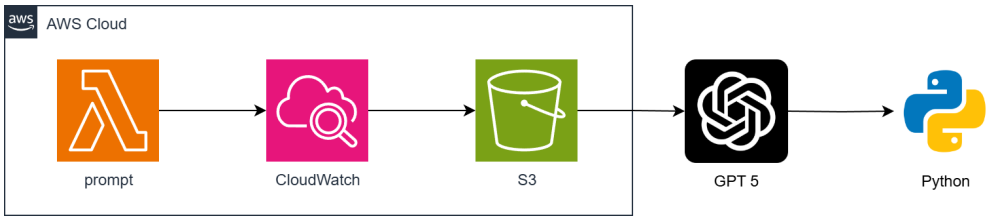


Figura 4.3: Arquitectura provisoria para análisis de métricas del módulo de prompts

Esta arquitectura sigue el flujo: Lambda Prompt → CloudWatch → S3 → GPT-5 → Python, donde GPT-5 genera resúmenes estructurados de los datos de cada ejecución, los cuales son posteriormente procesados con Python para obtener estadísticas y visualizaciones. Si bien esta es una arquitectura provisoria, representa una solución efectiva para el análisis de métricas. La implementación ideal consistiría en almacenar

estas métricas directamente en DynamoDB por cada ejecución, lo cual se abordará en las mejoras futuras del sistema.

4.3.2. Distribución de Servicios Utilizados

El análisis de los servicios utilizados durante la fase de desarrollo y pruebas permite demostrar la capacidad del sistema para recopilar métricas de uso detalladas:

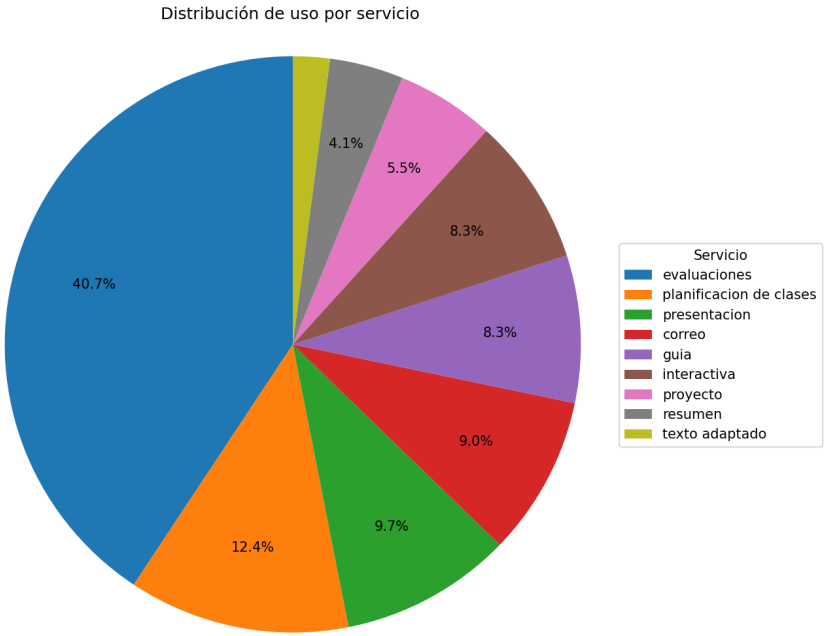


Figura 4.4: Distribución de servicios utilizados del módulo de ingeniería de prompts

Los datos muestran evaluaciones (41 %), planificaciones de clase (12 %) y presentaciones (10 %) como los servicios más utilizados durante las pruebas. Aunque estos datos no son representativos de usuarios finales, demuestran la capacidad del sistema para capturar y analizar patrones de uso detallados, funcionalidad que resultará valiosa cuando la plataforma entre en operación oficial.

4.3.3. Métricas de Rendimiento y Eficiencia

El análisis detallado de las ejecuciones del módulo de prompts proporciona métricas específicas sobre eficiencia y uso de recursos:

Métrica	Duration (ms)	Memory (MB)	Input Tokens	Output Tokens	Total Tokens
Promedio	3,508.1	137.2	680.4	674.4	1,354.4
Mínimo	347.0	135.0	269.0	113.0	382.0
Máximo	43,366.8	140.0	1,289.0	1,428.0	2,260.0

Cuadro 4.4: Métricas detalladas de eficiencia del módulo de ingeniería de prompts

Los resultados revelan un tiempo promedio de procesamiento de 3.5 segundos con un uso eficiente de memoria (137MB promedio). La variabilidad en la duración (347ms a 43.3s) refleja la diferencia en complejidad entre tipos de documentos, mientras que el uso promedio de 1,354 tokens totales demuestra la eficiencia de las plantillas desarrolladas en la optimización del consumo de recursos del LLM.

4.3.4. Procesamiento de Datos y Filtrado

Durante el proceso de análisis, se aplicó un filtrado riguroso de los datos obtenidos de CloudWatch, eliminando ejecuciones conflictivas, erróneas, sin tokens registrados, o con logs interrumpidos. Si bien la descarga de datos completos de CloudWatch es una estrategia efectiva, ocasionalmente se presentan desórdenes en el trazeo que requieren este proceso de limpieza de datos.

El análisis se realizó exclusivamente sobre ejecuciones exitosas para obtener métricas representativas del rendimiento real del módulo cuando opera bajo condiciones normales. Esta metodología permite evaluar la eficiencia intrínseca del diseño implementado mediante técnicas few-shot y plantillas estructuradas.

4.3.5. Eficiencia por Tipo de Servicio

El análisis detallado por tipo de servicio confirma la eficiencia diferencial esperada según la complejidad de cada tipo de documento:

Servicio	Duration (ms)	Memory (MB)	Input Tokens	Output Tokens	Total Tokens
Guía	7,242.9	137.9	613.8	673.8	1,287.6
Planificación de clases	3,554.7	136.8	641.4	670.8	1,312.3
Evaluaciones	3,524.9	136.9	571.4	743.7	1,315.1
Proyecto	3,260.9	138.0	601.8	667.8	1,269.5
Presentación	2,872.6	137.1	808.4	659.5	1,467.9
Interactiva	1,714.7	137.5	788.4	564.1	1,352.5
Resumen	1,699.9	137.5	924.5	626.0	1,550.5
Texto adaptado	1,641.7	137.3	995.0	834.0	1,829.0
Correo	1,091.1	137.6	848.2	528.1	1,407.1

Cuadro 4.5: Métricas de eficiencia por tipo de servicio del módulo de prompts

Los resultados confirman la correlación esperada entre complejidad del documento y tiempo de procesamiento. Los servicios más complejos como guías (7.2s) y evaluaciones (3.5s) requieren mayor tiempo de procesamiento, mientras que servicios más simples como correos (1.1s) y adaptación de texto (1.6s) se procesan más rápidamente.

La consistencia en el uso de memoria (137MB ± 1MB) confirma la estabilidad del sistema independientemente del tipo de servicio.

Esta diferenciación valida la efectividad de la construcción modular implementada, donde cada tipo de servicio utiliza únicamente los componentes de prompt necesarios para su función específica, optimizando recursos mientras mantiene la calidad del output generado.

Los resultados confirman que el módulo de ingeniería de prompts logra un balance óptimo entre versatilidad funcional y eficiencia de recursos, adaptándose dinámicamente a la complejidad requerida por cada tipo de documento educativo.

Es importante destacar que servicios como “Texto adaptado“ y “Resumen“ presentan un mayor conteo de tokens de entrada debido a que procesan textos largos propor-

cionados por el usuario, además del contenido agregado por el módulo de prompts. Esta característica explica por qué estos servicios, aunque conceptualmente simples, pueden mostrar mayor consumo de tokens que servicios más complejos estructuralmente. Por esta razón, la duración de procesamiento se considera un indicador más representativo de la complejidad computacional real que el conteo de tokens para estos tipos de servicio.

La consistencia en el uso de memoria (137MB ± 1MB) confirma la estabilidad del sistema independientemente del tipo de servicio.

4.4. Análisis de Escalabilidad

La arquitectura basada en componentes serverless implementada fue diseñada con escalabilidad como principio fundamental. Los resultados obtenidos durante las pruebas validan la capacidad del sistema para adaptarse dinámicamente a diferentes niveles de demanda sin intervención manual.

4.4.1. Escalabilidad Horizontal Validada

Las pruebas de concurrencia demuestran que la arquitectura maneja exitosamente múltiples usuarios simultáneos manteniendo la consistencia en el rendimiento. El sistema procesó 39 solicitudes de login concurrentes con una tasa de éxito del 97.4 % y 22 solicitudes de generación de material con tiempos de respuesta estables, confirmando que las funciones Lambda escalan automáticamente según la demanda.

La ausencia de cuellos de botella durante los picos de carga valida que la separación en seis funciones especializadas permite un escalamiento independiente por componente, optimizando el uso de recursos según las necesidades específicas de cada servicio.

4.4.2. Adaptabilidad por Complejidad

El sistema demuestra capacidad para escalar no solo horizontalmente sino también en términos de complejidad de procesamiento. La variabilidad en tiempos de ejecución (1.1s para correos vs 7.2s para guías) confirma que la arquitectura se adapta eficientemente a diferentes niveles de demanda computacional sin impactar otros servicios concurrentes.

El uso consistente de memoria (137MB ± 1MB) independientemente del tipo de servicio indica que el sistema mantiene eficiencia de recursos incluso cuando maneja simultáneamente documentos de diferentes complejidades.

4.4.3. Elasticidad de los Componentes Serverless

Los componentes serverless implementados proporcionan elasticidad automática donde los recursos se asignan y liberan dinámicamente según la demanda real. Durante períodos de baja actividad, el sistema no consume recursos, mientras que en picos de demanda se escala automáticamente sin límites predefinidos.

La arquitectura elimina la necesidad de planificación de capacidad tradicional, permitiendo que el sistema responda instantáneamente a variaciones impredecibles de carga, característica especialmente relevante para plataformas educativas donde la demanda puede fluctuar según calendarios académicos y horarios de clase.

4.4.4. Proyección de Crecimiento

Los resultados confirman que la arquitectura implementada puede expandirse sin modificaciones estructurales. La separación de responsabilidades entre funciones Lambda permite agregar nuevos servicios educativos o modificar existentes sin impactar la funcionalidad global del sistema.

La estrategia dual de bases de datos (PostgreSQL para datos estructurados, DynamoDB para datos funcionales) está diseñada para escalar independientemente según los

patrones de crecimiento específicos de cada tipo de información, manteniendo el rendimiento óptimo a medida que aumenta el volumen de usuarios y contenido generado.

La validación de escalabilidad obtenida confirma que la arquitectura basada en componentes serverless desarrollada cumple con los objetivos de adaptabilidad y crecimiento necesarios para una plataforma educativa de producción.

4.5. Alcance Funcional Logrado

El desarrollo del módulo de ingeniería de prompts logró implementar un sistema completo de nueve servicios educativos especializados, cumpliendo con el objetivo de proporcionar una cobertura integral para las principales necesidades de generación de material educativo identificadas. Este alcance funcional demuestra la capacidad del sistema para abordar diferentes contextos pedagógicos mediante la aplicación de técnicas few-shot y construcción modular de prompts.

4.5.1. Servicios Educativos Implementados

Se logró la implementación completa de nueve servicios diferenciados que cubren el espectro de necesidades docentes, desde la generación de material evaluativo hasta el procesamiento de contenido existente. Estos servicios se organizan en dos categorías según su funcionalidad final.

Servicios de Generación de Documentos

Seis servicios implementan el flujo completo desde la generación del prompt hasta la transformación en documentos descargables mediante plantillas HTML especializadas:

Servicio	Plantilla Prompt	Plantilla HTML
Evaluación Escrita	✓	evaluaciones.html
Planificación de Clase	✓	planificaciones.html
Presentación	✓	presentacion.html
Proyecto	✓	proyecto.html
Guía	✓	guia.html
Actividad Interactiva	✓	interactiva.html

Cuadro 4.6: Servicios con generación completa de documentos

Servicios de Procesamiento de Texto

Tres servicios adicionales proporcionan capacidades de procesamiento y transformación de contenido mediante plantillas de prompt especializadas. Estos servicios entregan texto como respuesta directa, sin generar documentos estructurados descargables:

Servicio	Plantilla Prompt	Funcionalidad
Correo	✓	Redacción de comunicaciones formales
Texto Adaptado	✓	Adaptación de contenidos por nivel educativo
Resume Texto	✓	Síntesis automatizada de contenidos

Cuadro 4.7: Servicios de procesamiento de texto

4.5.2. Especialización al Contexto Chileno Lograda

El sistema logró una adecuada adaptación al contexto educativo nacional, incorporando elementos específicos que aseguran la pertinencia y aplicabilidad del material generado:

- **Mapeo curso-edad**: Correspondencia completa desde 1° básico a 4° medio con edades específicas para cada nivel educativo
- **Integración con Taxonomía de Bloom**: Implementación exitosa de los seis niveles cognitivos (Conocimiento, Comprensión, Aplicación, Análisis, Síntesis, Evaluación) con habilidades específicas disponibles
- **Contextualización educativa**: Los prompts incorporan referencias al contexto escolar chileno, asegurando que el LLM genere contenido apropiado al sistema educativo nacional

4.5.3. Completitud del Sistema

El módulo de ingeniería de prompts desarrollado logró implementar las siguientes capacidades técnicas que garantizan un sistema completo y funcional:

- **Cobertura funcional integral**: Nueve tipos diferentes de servicios educativos que abordan las principales necesidades docentes identificadas
- **Consistencia metodológica**: Aplicación sistemática y exitosa de técnicas few-shot y plantillas estructuradas en todos los servicios
- **Optimización de recursos**: Construcción modular que adapta el procesamiento según la complejidad específica de cada tipo de documento
- **Flexibilidad tecnológica**: Diseño agnóstico del LLM implementado que permite evolución tecnológica sin reconstrucción del sistema

El alcance funcional logrado confirma que el sistema desarrollado aborda de manera integral las necesidades de generación de material educativo en el contexto chileno, proporcionando una solución completa que optimiza efectivamente el trabajo docente mediante la aplicación exitosa de técnicas avanzadas de ingeniería de prompts en un entorno educativo específico.

Capítulo 5

Conclusiones

Los objetivos específicos individuales planteados para este trabajo de título se cumplieron exitosamente, demostrando la viabilidad técnica y práctica de la propuesta desarrollada.

El módulo de ingeniería de prompts se implementó completamente mediante nueve plantillas especializadas que incorporan técnicas few-shot, construcción modular y adaptación contextual al sistema educativo chileno. La arquitectura iterativa desarrollada demuestra la aplicación práctica de metodologías avanzadas de prompt engineering, logrando consistencia y calidad en la generación de material educativo personalizado.

La arquitectura backend basada en componentes serverless se materializó mediante la implementación exitosa del patrón API Gateway + Lambda complementado con servicios administrados. Las pruebas de concurrencia validaron la escalabilidad automática y eficiencia operativa del sistema, confirmando que las decisiones arquitectónicas adoptadas responden efectivamente a los requerimientos de una plataforma educativa.

La transición desde técnicas teóricas del estado del arte hacia una implementación funcional se logró exitosamente, generando valor educativo real y demostrando la viabilidad de aplicar investigación académica en contextos prácticos específicos. El sistema desarrollado trasciende la mera aplicación de tecnologías existentes, integrándolas de manera coherente para resolver problemáticas educativas concretas.

El diseño modular agnóstico del proveedor LLM se implementó satisfactoriamente, permitiendo intercambiabilidad entre diferentes modelos de lenguaje sin modificaciones estructurales del sistema. Esta flexibilidad arquitectónica valida la sostenibilidad técnica de la solución y su capacidad de evolución ante cambios tecnológicos futuros.

Los resultados obtenidos confirman que los objetivos técnicos planteados no solo se alcanzaron, sino que se superaron mediante la implementación de funcionalidades adicionales como el sistema de métricas provisorio y la integración completa con el contexto educativo chileno.

5.1. Lecciones Aprendidas y Desafíos

El desarrollo del proyecto proporcionó aprendizajes técnicos significativos sobre la importancia de decisiones arquitectónicas tempranas. La experiencia demostró que establecer una estructura sólida inicial impacta exponencialmente en el desarrollo posterior, validando la inversión de tiempo en diseño conceptual antes de la implementación.

El proceso de refinamiento iterativo human-in-the-loop con el LLM presentó desafíos inesperados en términos de consistencia y predictibilidad de respuestas. La solución mediante plantillas estructuradas y técnicas few-shot requirió múltiples iteraciones hasta alcanzar la estabilidad necesaria para un entorno educativo.

Las limitaciones de acceso a LLMs gratuitos durante el desarrollo obligaron a optimizar el uso de recursos y desarrollar estrategias eficientes de testing, lo que paradójicamente resultó en un diseño más robusto y eficiente en el manejo de tokens.

Los aspectos de integración, incluyendo las conexiones entre frontend y backend mediante API Gateway y la configuración de RDS PostgreSQL, se resolvieron más rápidamente de lo anticipado, validando la madurez del ecosistema AWS para desarrollo serverless.

La decisión de no implementar Secret Manager por limitaciones de tiempo representa la principal limitación arquitectónica, aunque no compromete la funcionalidad core del sistema desarrollado.

5.2. Trabajo Futuro e Implicaciones

El trabajo futuro se orienta hacia tres principales mejora y evolución del sistema.

5.2.1. Sistema de Métricas Avanzado

Se propone implementar un sistema de métricas que evolucione la arquitectura provisoria actual hacia una solución completamente automatizada y autooptimizante. Este sistema aborda las limitaciones identificadas en el análisis de métricas manual y establece un ciclo de retroalimentación inteligente para la mejora continua del rendimiento.

Arquitectura Propuesta

La arquitectura del sistema de métricas se basa en un enfoque híbrido que combina recolección en tiempo real con análisis automatizado, como se muestra en la Figura 5.1.

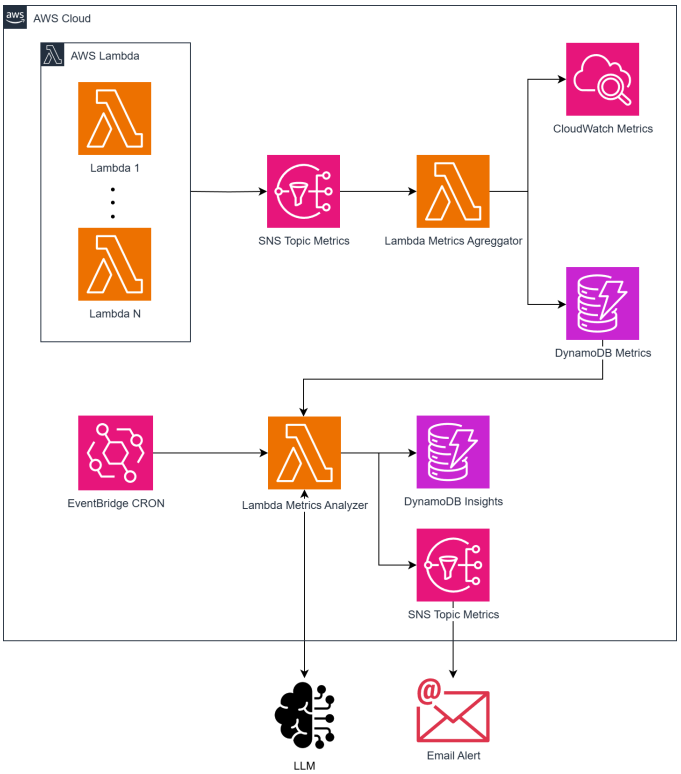


Figura 5.1: Arquitectura propuesta para sistema de métricas avanzado

El sistema propuesto implementa un flujo de cuatro etapas principales:

1. Recolección Directa desde Origen: Cada función Lambda del sistema (prompt, login, get_data_dynamodb, get_data_db, taxonomia_bloom, contactanos) enviará métricas detalladas directamente a un tópico SNS especializado. Estas métricas incluirán duración de ejecución, uso de memoria, tokens de entrada y salida, estado de ejecución (OK/NOK), y contexto específico del servicio.

2. Agregación y Almacenamiento Híbrido: Una función Lambda Aggregator procesará los mensajes SNS y realizará almacenamiento dual: métricas completas y estructuradas en DynamoDB para análisis detallado, y métricas clave en CloudWatch para dashboards y alertas en tiempo real.

3. Análisis Inteligente Automatizado: Un sistema de análisis basado en EventBridge ejecutará periódicamente una función Lambda Analyzer que utilizará APIs de LLM para procesar las métricas almacenadas, detectar patrones de rendimiento, identificar anomalías y generar recomendaciones de optimización.

4. Retroalimentación y Alertas: El sistema generará insights que se almacenan en DynamoDB y activará alertas automáticas vía SNS cuando detecte degradación de rendimiento o anomalías significativas.

Ventajas Técnicas

Esta arquitectura elimina completamente la dependencia de procesamiento manual de logs de CloudWatch, proporcionando métricas precisas desde el origen de cada ejecución. La recolección en tiempo real mediante SNS garantiza disponibilidad inmediata de datos sin necesidad de procesar logs de texto, mientras que el almacenamiento híbrido optimiza tanto el análisis detallado como la visualización operacional.

El componente de análisis inteligente representa una innovación significativa al aplicar LLM para la interpretación automática de métricas de sistemas, generando insights que van más allá de simples thresholds estadísticos para incluir recomendaciones contextuales sobre optimización de prompts y ajustes arquitectónicos.

Ciclo de Retroalimentación Autooptimizante

El sistema establecerá un ciclo de mejora continua donde las recomendaciones generadas por el análisis LLM se utilizan para refinar automáticamente las plantillas de prompts y ajustar parámetros de configuración. Esta capacidad de autooptimización representa una evolución hacia sistemas de inteligencia artificial que no solo funcionan, sino que mejoran continuamente su propio rendimiento basándose en datos operacionales reales.

La implementación de este sistema de métricas avanzado transformaría la plataforma actual en un sistema verdaderamente inteligente y autoadaptativo, estableciendo un nuevo estándar para la aplicación de técnicas de observabilidad en sistemas educativos basados en IA.

5.2.2. Migración a Arquitectura Completamente Serverless

La evolución natural del sistema incluye migrar de RDS PostgreSQL a Amazon Aurora Serverless v2, completando la transición hacia una arquitectura completamente serverless. Esta migración consolidaría el modelo pay-per-use en todos los componentes, optimizando costos operativos y manteniendo la consistencia arquitectónica.

5.2.3. Diversificación de Proveedores LLM

El acceso mejorado a APIs de LLM gratuitos o de bajo costo post-desarrollo abre oportunidades para implementar estrategias multi-proveedor. La arquitectura agnóstica desarrollada facilita la evaluación comparativa de diferentes modelos, permitiendo optimización continua de calidad y costos según las características específicas de cada tipo de documento educativo.

5.2.4. Integración con Bases Curriculares Chilenas

Una evolución natural del módulo de ingeniería de prompts incluye la integración directa con las Bases Curriculares chilenas oficiales del Ministerio de Educación. Esta mejora permitiría que los prompts incorporen objetivos de aprendizaje específicos, habilidades curriculares por asignatura y criterios de evaluación definidos por el MINEDUC, transformando la contextualización actual en una alineación precisa con el marco curricular nacional.

Esta evolución se ve favorecida por los avances continuos en la capacidad contextual de los LLMs, que han incrementado progresivamente su capacidad para procesar y mantener coherencia con contextos cada vez más extensos y complejos.

5.3. Reflexión Final

La convergencia entre inteligencia artificial e ingeniería de software aplicada al contexto educativo demuestra un potencial significativo para abordar problemáticas estructurales como la carga laboral docente. El proyecto evidencia que la aplicación rigurosa de metodologías de ingeniería de software a soluciones basadas en IA puede generar sistemas robustos, escalables y contextualmente relevantes.

El valor diferencial de combinar técnicas avanzadas de prompt engineering con arquitecturas cloud modernas radica en la capacidad de crear soluciones que no solo funcionan técnicamente, sino que se adaptan específicamente a las necesidades y restricciones del contexto educativo chileno. Esta especialización contextual, sustentada por una base técnica sólida, representa un modelo replicable para la aplicación de tecnologías emergentes en sectores con necesidades específicas.

El impacto potencial trasciende la optimización individual del trabajo docente, sugiriendo un paradigma donde la tecnología puede ser un habilitador efectivo para la mejora sistémica de procesos educativos, siempre que se implemente con rigor técnico y sensibilidad contextual.

Bibliografía

[1] P. Morales and R. Pineda. *Planificación educativa: teoría y práctica*. Editorial Educativa, 2009.

[2] L. González and M. Hernández. Creación de contenidos educativos digitales: Una guía práctica. *Revista de Tecnología Educativa*, 29(2):45–60, 2017.

[3] Stuart Russell, Peter Norvig, and Ernest Davis. Artificial intelligence: A modern approach. *Malaysia; Pearson*, 2016.

[4] Pablo Castillo Armijo. Inclusión educativa en la formación docente en chile: tensiones y perspectivas de cambio. *Revista de Estudios y Experiencias en Educación*, 2021.

[5] MagicSchool. Magicschool platform, 2024.

[6] MegaProfe. Megaprofe platform, 2024.

[7] TeachMateAI. Teachmateai platform, 2024.

[8] Eduaide. Eduaide platform, 2024.

[9] L. Reynolds and K. McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. *arXiv preprint arXiv:2102.07350*, 2021.

[10] Z. Zhang, A. Zhang, M. Li, and A. Smola. Automatic chain of thought prompting in large language models. *Papers with Code*, 2022.

[11] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[12] H. Patel and S. Parmar. Prompt engineering for large language model. *ResearchGate*, 2023.

[13] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.

[14] S. Vatsal and H. Dubey. A survey of prompt engineering methods in large language models for different nlp tasks. *Papers with Code*, 2023.

[15] Chinazor Prisca Amajuoyi, Luther Kington Nwobodo, and Mayokun Daniel Adegbola. Transforming business scalability and operational flexibility with advanced cloud computing technologies. *Computer Science & IT Research Journal*, 2024.

[16] Hassan B. Hassan, Saman A. Barakat, and Qusay I. Sarhan. Survey on serverless computing. *Journal of Cloud Computing*, 2021.

[17] Mohammad Tari, Mostafa Ghobaei-Arani, Jafar Pouramini, and Mohsen Ghorbian. Auto-scaling mechanisms in serverless computing: A comprehensive review. *Computer Science Review*, 2024.

[18] Joel Lopes and Ceres Dbritto. Implementing serverless computing architectures for expandable and cost-effective cloud applications. *International Journal of Computer Trends and Technology*, 2024.

[19] Juan Mera Menéndez, Jose Emilio Labra Gayo, Enrique Riesgo Canal, and Aitor Echevarría Fernández. A comparison between traditional and serverless technologies in a microservices setting. *arXiv preprint arXiv:2305.13933*, 2023.

[20] Mohamed Lemine El Bechir, Cheikh Sad Bouh, and Abobakr Shuwail. Comprehensive review of performance optimization strategies for serverless applications on aws lambda. *arXiv preprint arXiv:2407.10397*, 2024.

[21] Thomas Bodner, Theo Radig, David Justen, Daniel Ritter, and Tilmann Rabl. An empirical evaluation of serverless cloud infrastructure for large-scale data processing. *arXiv preprint arXiv:2501.07771*, 2025.