

**Duoc UC sede Plaza Oeste**

# **Brain Boost: Plataforma Educativa Adaptativa con IA**

Informe para optar al título de Ingeniero en Informática

**Asignatura:** Capstone PTY4614\_006V

**Docente Guía:** Cristian Espinoza Silva

**Alumnos:**

- Macarena Bertero
- Ignacio León
- Claudio Valencia

**Santiago de  
Chile 2025**

## Índice de Contenido

<b>Índice de Contenido</b>	<b>2</b>
1. Agradecimientos	5
2. Resumen Ejecutivo	5
3. Abstract	7
4. Introducción	8
5. Descripción de la Problemática	9
5.1 Rrigidez del Contenido y Desventaja Pedagógica	9
5.2 Brecha en la Adaptabilidad y Diagnóstico en Tiempo Real	10
5.3 Justificación de la Innovación y Vacío de Mercado	10
6. Solución de Problema	10
6.1 Abordaje Técnico y Arquitectura de Microservicios	10
6.2 Estrategia de Persistencia y Modelado de Datos	11
6.3 Recursos Humanos y Metodología	11
6.4. Recursos Técnicos y Financieros	11
7. Objetivos del Proyecto	12
7.1 Objetivo General	12
7.2 Objetivos Específicos	13
8. Competencias del Perfil de Egreso	13
9. Acta de Constitución del Proyecto	14
10. Asignación de Roles	18
11. Metodología	20
12. Carta Gantt	21
12.1 Gestión y Priorización de Tareas	21
12.2 Carta Gantt del Proyecto	22
13. Marco Teórico	23
13.1 Fundamentos Tecnológicos del Proyecto	23
13.2 Estado del Arte y Benchmarking	25
14. Alcances del Proyecto	27
14.1 Funcionalidades Incluidas	27
14.2 Funcionalidades Excluidas	28
15. Requerimientos del Sistema	29
15.1. Requisitos Funcionales	29
15.2. Requisitos No Funcionales	30
15.3 Matriz de Trazabilidad	31
16. Diseño de Interfaz y Experiencia de Usuario	33
16.1. Flujo Principal de la Aplicación Web (Gestión y Contenido)	33
16.2 Inicio de Sesión y Dashboard	33
16.3 Flujo de Carga de Usuarios y Contenido	35

16.4. Flujo de la Aplicación Móvil (Profesor y Alumno)	37
16.4 Vista Móvil del Profesor (Generación de Contenido)	37
2.2 Vista Móvil del Alumno	42
17. Implementación del Proyecto	44
17.1 Descripción General del Sistema	44
17.2 Diagrama de Arquitectura	46
17.2.1. Persistencia y Almacenamiento de Datos	46
17.2.2. Microservicios Principales	47
17.3 Modelo Entidad-Relación	49
17.3.1. Diccionario de Datos	49
17.3.1.1 Tablas de Ubicación e Institución	49
17.3.1.2 Tablas de Roles y Contenido Académico	50
17.3.1.3 Tablas de Usuarios y Relación	52
17.3.1.4 Tablas de Auditoría y Cargas	53
17.3.1.5 Tablas de Gamificación (Genéricas)	53
17.3.1.6 Tablas de Gamificación	54
17.3.4 Modelado Técnico	56
17.4.1 Diagrama de Secuencia	56
17.4.2 Diagrama de Actividad: Flujo Asíncrono	58
17.4.3 Vistas 4+1	60
17.4.3.1 Vista Lógica: Diagrama de Clases	60
17.4.3.2 Vista de Desarrollo : Diagrama de Componentes	62
17.4.3.3 Vista de Procesos:	65
Proceso Principal: Flujo de Juego Adaptativo (Síncrono)	65
17.4.3.4 Vista Física: Diagrama de Despliegue	66
17.4.3.5 Vista de Escenarios: Diagrama de Casos de Uso	68
18. Decisiones Técnicas y Tecnologías Aplicadas	70
18.1 Tecnologías, Frameworks y Librerías:	70
18.2 Patrones de Diseño Implementados:	72
18.3 Principales Decisiones Técnicas y su Impacto en la Solución	73
19. Desarrollo y Soluciones Implementadas	75
19.1 Funcionalidades Desarrolladas y Cumplimiento de Requerimientos	75
19.1.1 Auth Service	76
19.1.2 Game Engine	76
19.1.3 Content Service	77
19.1.4 IA Service	77
19.1.5 Scoring Service	77
19.2 Retos Enfrentados y Soluciones Aplicadas	77
19.2.1 Evidencia de Funcionamiento	78
Carga de Contenido Teórico y Persistencia en MongoDB	82
Recolección de Respuesta de Usuario (Endpoint /games/{gameId}/concepts/submit)	83
Pruebas y Validación del Sistema	84

1. Pruebas Unitarias	84
Despliegue y Entorno de Ejecución	87
C. Testing Automatizado	89
D. Despliegue en Producción (Perfil prod)	89
Documentación Complementaria	90
Especificación de API	90
Manual Técnico	94
Registro de Versiones	94
Aspectos Éticos, Legales y de Seguridad	96
Marco Legal y Cumplimiento Normativo	96
Ley 19.628 sobre Protección de Datos Personales	96
Mecanismos de Autenticación, Cifrado y Resguardo	96
Autenticación y Autorización	96
Cifrado y Resguardo de Información	97
Uso Responsable de Datos e Inteligencia Artificial	97
Ética en el Uso de Datos	97
Uso Responsable de Inteligencia Artificial	97
Conclusiones Técnicas y Aprendizajes	98
1. Implementación Exitosa de Persistencia Híbrida	98
2. Cumplimiento Riguroso del RNF-01 (Rendimiento)	98
3. Integración Segura y Resiliente con IA	99
Factibilidad Económica	101
1. Flujo de Caja	101
Ingresos por Venta	102
Costos Operacionales	102
Inversiones y Capital de Trabajo (KT)	102
2. Valor Actual Neto	103
Cálculo del VAN	103
3. Tasa Interna de Retorno	103
Cálculo e Interpretación del TIR	104
Bibliografía y Referencias	104
Anexos	104

## 1. Agradecimientos

## 2. Resumen Ejecutivo

El sector de la educación superior enfrenta desafíos significativos en la retención de la información y la personalización del estudio. Los métodos tradicionales de evaluación y repaso se basan en contenido estático, lo que resulta en una alta curva de esfuerzo para el docente en la creación constante de material actualizado y una baja motivación por parte del alumno para el repaso activo. Esta ineficiencia se agrava por la desconexión entre el material teórico extenso y las herramientas de evaluación práctica.

Brain Boost se presenta como la solución, siendo una plataforma de gamificación educativa adaptativa que ataca la problemática de la retención mediante la transformación inteligente del contenido. El sistema convierte material teórico no estructurado (PDF/Texto) en juegos dinámicos y atractivos, utilizando Inteligencia Artificial para personalizar la experiencia de aprendizaje. Su factor diferenciador radica en la resiliencia y la lógica híbrida: la IA Google Gemini 2.5 Flash genera el contenido inicial y, crucialmente, el sistema implementa un mecanismo de refuerzo adaptativo condicional. Este mecanismo permite que la IA intervenga solo en tiempo real para generar preguntas de refuerzo si se detecta un bajo rendimiento del alumno en un tema, optimizando el rendimiento y asegurando la continuidad del servicio.

Para soportar esta solución integral, el proyecto se basa en una sólida Arquitectura de Microservicios y una persistencia dual. Los requerimientos principales del sistema incluyen:

1. Seguridad y Acceso: Gestión de acceso basada en JWT y Hashing a través del Auth Service, y funcionalidad de Carga Masiva de Alumnos para la administración de grupos.

2. Generación de Contenido Inteligente: Uso del modelo Google Gemini 2.5 Flash para la generación de preguntas, resúmenes y la transformación de archivos subidos por el profesor.
3. Resiliencia y Adaptabilidad: Implementación de un algoritmo de refuerzo condicional que permite la intervención de la IA sólo cuando es pedagógicamente necesario, y un mecanismo de fallback al banco de datos estático.
4. Persistencia Híbrida: Utilización de PostgreSQL para la integridad transaccional de cuentas y métricas, y MongoDB para la ingestión eficiente de archivos masivos.

El desarrollo se organiza bajo la metodología Scrum en tres fases principales, cubriendo todo el ciclo de vida del proyecto:

1. Planificación y Diseño

Objetivo: Definición de la arquitectura, modelo de datos y requisitos funcionales completos.

2. Implementación de Arquitectura

Objetivo: Desarrollo de la infraestructura base, módulos de seguridad y persistencia, e integración con la IA.

3. Desarrollo de Lógica de Negocio

Objetivo: Codificación final de la lógica compleja de Gamificación, implementación de la adaptabilidad de la IA y despliegue del Frontend.

### 3. Abstract

The higher education sector faces significant challenges in information retention and personalized learning. Traditional methods of assessment and review rely on static content, resulting in a high workload for teachers in the continuous creation of updated material and low student motivation for active review. This inefficiency is further aggravated by the gap between extensive theoretical material and practical assessment tools.

Brain Boost emerges as the solution — an adaptive educational gamification platform that addresses retention issues through the intelligent transformation of content. The system converts unstructured theoretical material (PDF/Text) into dynamic and engaging games, using Artificial Intelligence to personalize the learning experience. Its key differentiator lies in resilience and hybrid logic: the Google Gemini 2.5 Flash AI generates the initial content, while a conditional adaptive reinforcement mechanism allows AI to intervene only in real time when low student performance is detected in a topic. This optimizes performance and ensures service continuity.

To support this comprehensive solution, the project is built upon a robust Microservices Architecture and dual persistence model. The main system requirements include:

- Security and Access: Access management based on JWT and Hashing through the Auth Service, with a Bulk Student Upload functionality for group administration.
- Intelligent Content Generation: Use of the Google Gemini 2.5 Flash model for generating questions, summaries, and transforming files uploaded by instructors.
- Resilience and Adaptability: Implementation of a conditional reinforcement algorithm that allows AI intervention only when pedagogically necessary, with a fallback mechanism to the static data repository.
- Hybrid Persistence: Utilization of PostgreSQL for transactional integrity of accounts and metrics, and MongoDB for efficient ingestion of large files.

The development process follows the Scrum methodology, divided into three main phases that cover the entire project lifecycle:

### 1. Planning and Design

Objective: Define the architecture, data model, and complete functional requirements.

### 2. Architecture Implementation

Objective: Develop the base infrastructure, security and persistence modules, and integrate the AI component.

### 3. Business Logic Development

Objective: Final coding of the gamification logic, implementation of AI adaptability, and Frontend deployment.

## 4. Introducción

El panorama de la educación superior contemporánea exige la adopción de herramientas que mejoren los procesos de retención cognitiva y la participación activa del estudiante. En el contexto chileno y latinoamericano, esta necesidad se vuelve crítica; estudios recientes, como los reportados por la OCDE, señalan que la efectividad de los métodos de enseñanza requiere una evolución constante para enfrentar desafíos como la alta tasa de abandono en los primeros años de carrera o las brechas persistentes en el dominio de competencias básicas. Esta situación se agrava por el uso de métodos de repaso y evaluación basados en contenido estático, lo que impacta negativamente la motivación del alumnado y genera una carga operativa considerable para el cuerpo docente.

El proyecto Brain Boost surge como una respuesta estratégica a esta problemática, posicionándose en el ecosistema de las soluciones EdTech. Su ejecución está enmarcada en la necesidad de transformar el paradigma de estudio pasivo en uno activo y gamificado, integrando la potencia de la Inteligencia Artificial (IA) para crear una plataforma de refuerzo académico adaptativa. El sistema está diseñado para operar en un entorno de alta disponibilidad a través de una Arquitectura de Microservicios, garantizando la escalabilidad necesaria para atender a grandes volúmenes de usuarios y una gestión eficiente de la generación de contenido dinámico.

El presente documento constituye el Informe Final del Proyecto Brain Boost, cubriendo el ciclo completo de desarrollo. A través de sus secciones, se detallará la Arquitectura de Microservicios implementada, se presentará el diseño final del Motor de Gamificación y la Lógica Adaptativa de la IA, y se validará el funcionamiento de la plataforma en su entorno de producción. El objetivo es documentar el cumplimiento exitoso de todos los requerimientos funcionales y técnicos, demostrando la solidez y la viabilidad de la solución.

## 5. Descripción de la Problemática

El proyecto Brain Boost se origina como una respuesta directa a una necesidad crítica en la educación superior: la falta de mecanismos tecnológicos que automaticen la personalización del refuerzo académico y superen la rigidez inherente del material didáctico. Esta problemática se manifiesta en tres áreas clave que definen la justificación de este proyecto:

### 5.1 Rigidez del Contenido y Desventaja Pedagógica

La institución proporciona el contenido didáctico y evaluativo centralizado (pruebas y exámenes). Sin embargo, la naturaleza de este material crea una brecha en la personalización:

- Dependencia de Formatos Rígidos: El contenido didáctico y evaluativo se encuentra encapsulado en formatos que no admiten la generación dinámica de variaciones.
- Imposibilidad de Variación: Esta rigidez impide que el docente pueda ofrecer ejercicios de repaso focalizados más allá del material estándar, lo que compromete la capacidad de la institución para ofrecer experiencias de aprendizaje diferenciadas.

## 5.2 Brecha en la Adaptabilidad y Diagnóstico en Tiempo Real

La deficiencia más crítica de los sistemas actuales es su incapacidad para adaptarse al ritmo individual del alumno:

- Falta de Diagnóstico en Tiempo Real: Los métodos de evaluación existentes no tienen la capacidad de diagnosticar instantáneamente y con precisión el bajo rendimiento de un alumno en un concepto específico.
- Ausencia de Mecanismo de Refuerzo Automatizado: Consecuentemente, no existe una herramienta automatizada que active una estrategia de refuerzo focalizado y condicional. Los alumnos con brechas de conocimiento reciben el mismo material que el resto, lo cual disminuye la motivación y la eficacia de la intervención pedagógica.

## 5.3 Justificación de la Innovación y Vacío de Mercado

La necesidad de Brain Boost se justifica por un vacío tecnológico y pedagógico que el proyecto resuelve:

- Vacío de Mercado: Actualmente, no existe una plataforma integral que combine la transformación inteligente de documentos (para crear material de apoyo dinámico), la gamificación y un mecanismo de refuerzo condicional adaptado al rendimiento del alumno en un solo entorno.

El problema central que justifica este desarrollo es la ausencia de un sistema pionero capaz de automatizar el refuerzo pedagógico individual y proveer contenido dinámico, llenando así esta necesidad fundamental en el mercado EdTech.

# 6. Solución de Problema

El desarrollo de Brain Boost se aborda mediante una solución integral basada en una arquitectura moderna de microservicios y una estrategia de recursos planificada para asegurar la escalabilidad, la resiliencia, y el cumplimiento de los requerimientos funcionales.

## 6.1 Abordaje Técnico y Arquitectura de Microservicios

La solución técnica se centra en una Arquitectura *Cloud-Native*, desacoplando las funcionalidades en seis servicios principales que comunican a través de un BFF (Backend for Frontend), el cual maneja la orquestación y actúa como punto de acceso seguro.

## 6.2 Estrategia de Persistencia y Modelado de Datos

Se implementa una estrategia de Persistencia Dual para optimizar el rendimiento por tipo de dato:

- PostgreSQL (Relacional): Utilizado por los servicios críticos (Auth, User, Scoring, Game) para datos transaccionales y de alta integridad, como cuentas de usuario, métricas de rendimiento y el banco de preguntas final.
- MongoDB (NoSQL): Utilizado por el Content Service para la ingestión y almacenamiento eficiente de archivos de gran volumen (PDF/Texto) antes de su procesamiento por la IA.

El modelo de datos relacional se optimiza con tablas de unión y UUIDs como claves primarias para garantizar la unicidad a nivel distribuido y facilitar el crecimiento del sistema sin dependencia de secuencias centralizadas.

## 6.3 Recursos Humanos y Metodología

El proyecto es ejecutado por un equipo de 3 desarrolladores bajo la metodología Scrum. La clave del éxito en la implementación ha sido la especialización de roles por microservicio, asegurando la alta cohesión del código.

- Metodología: Uso de Sprints definidos para la entrega incremental y continua, permitiendo la adaptación rápida a los *feedback* y la mitigación temprana de riesgos técnicos.
- Control de Versiones: Git es utilizado como repositorio central, con ramas específicas para cada funcionalidad y revisiones de código obligatorias (*Code Reviews*) para mantener la calidad y estándares.

## 6.4. Recursos Técnicos y Financieros

El despliegue del proyecto está planificado para ser eficiente en costos desde su concepción, aprovechando la infraestructura *Cloud* para la alta disponibilidad y escalabilidad elástica.

- Pila Tecnológica: El *backend* se desarrolla utilizando Kotlin con Spring Boot. Esta elección garantiza un alto rendimiento y una gestión eficiente de la arquitectura de microservicios, optimizando los costos de la infraestructura en la nube. El frontend se desarrolla en Android Jetpack Compose Material 3. Esto permite crear una interfaz nativa y fluida con ciclos de desarrollo más rápidos, esencial para una experiencia de usuario de calidad.
- Despliegue y Contención de Costos: Se utiliza Docker para la contenerización y una herramienta de orquestación para gestionar el *deploy* y el balanceo de carga.
- Costo Operacional: El proyecto se implementa utilizando las capas gratuitas (Free Tier) de Google Cloud Platform y la API de Google Gemini. La resiliencia del mecanismo de *fallback* a la base de datos (PostgreSQL) actúa como un mecanismo de control de consumo, asegurando que no se excedan los límites de llamadas gratuitas, lo que garantiza la operatividad y la viabilidad económica a largo plazo.

## 7. Objetivos del Proyecto

### 7.1 Objetivo General

Desarrollar una aplicación de aprendizaje adaptativo, basada en gamificación e inteligencia artificial, para los estudiantes, con el fin de mejorar su retención de conocimiento en materias teóricas y fomentar el repaso activo.

## 7.2 Objetivos Específicos

- Gestión de Usuarios: Diseñar e Implementar un sistema robusto de gestión de usuarios con tres roles diferenciados (Estudiante, Profesor, Administrador), asegurando un protocolo de autenticación seguro basado en JWT y la gestión eficiente de permisos a nivel de microservicio.
- Integración de Contenido Inteligente: Construir un microservicio (IA Service) que se integre funcionalmente con la API de Google Gemini para generar preguntas dinámicas y personalizadas, complementando el contenido estático y analizando el rendimiento del estudiante para identificar áreas de debilidad.
- Implementación de Gamificación: Desarrollar el Game Service que incorpore mecánicas de juego esenciales como puntajes, rankings y desafíos (*leaderboards*), incentivando la participación activa del estudiante y proporcionando un entorno de aprendizaje lúdico y motivador.
- Construcción de una Arquitectura Escalable: Establecer una arquitectura de microservicios resiliente sobre Google Cloud Platform (GCP) con una estrategia de persistencia dual (PostgreSQL para transacciones y MongoDB para ingestas), garantizando la escalabilidad elástica y el mantenimiento a largo plazo de la solución.

## 8. Competencias del Perfil de Egreso

El desarrollo del proyecto Brain Boost valida la adquisición e implementación de diversas competencias técnicas del perfil de egreso, demostrando la capacidad del equipo para abordar soluciones complejas e innovadoras. A continuación, se detallan las competencias directamente asociadas a la ejecución del proyecto:

1. Construir el modelo arquitectónico de una solución sistémica que soporte los procesos de negocio de acuerdo los requerimientos de la organización y estándares industria.
- Justificación: Esta competencia se aplica directamente al diseño e implementación de la Arquitectura de Microservicios de Brain Boost. El proyecto resuelve la

necesidad de adaptabilidad y escalabilidad de la plataforma mediante la división de la lógica de negocio en servicios desacoplados, cumpliendo con los estándares de la industria para arquitecturas *Cloud-Native* y *Distributed Systems*.

2. Implementar soluciones sistémicas integrales para automatizar u optimizar procesos de negocio de acuerdo con las necesidades de la organización.
- Justificación: La automatización es el núcleo del proyecto. Esta competencia se demuestra con la funcionalidad de transformación de contenido (automatizar la conversión de PDFs en preguntas mediante el *IA Service*), la implementación del Scoring Service (optimizar el cálculo dinámico de rendimiento) y la integración de la lógica de refuerzo adaptativo (automatizar la intervención pedagógica).
3. Construir modelos de datos para soportar los requerimientos de la organización de acuerdo a un diseño definido y escalable en el tiempo.
- Justificación: Esta competencia se valida mediante la implementación de la estrategia de Persistencia Dual. El equipo construyó y gestionó dos modelos de datos distintos (PostgreSQL para datos transaccionales de alta integridad y MongoDB para la ingestión de archivos masivos), asegurando que el diseño sea escalable y cumpla con los requerimientos de cohesión y eficiencia.

## 9. Acta de Constitución del Proyecto

Proyecto	Plataforma de Refuerzo Adaptativo impulsada por IA (Brain Boost)
Fecha de Creación	16 de Agosto de 2025
Líder / Gerente de Proyecto	Cristian Espinoza
Patrocinador / Cliente	Cristian Espinoza

Nombre del Proyecto	Brain Boost: Plataforma de Refuerzo Académico Adaptativo
Breve Descripción	Desarrollo e implementación de una plataforma educativa de alto rendimiento, modular (microservicios) e impulsada por Inteligencia Artificial (Gemini 2.5 Flash). El objetivo principal es ofrecer a los estudiantes de Educación Superior sesiones de juego personalizadas y adaptativas, diagnosticando y reforzando sus debilidades conceptuales en tiempo real.
Objetivo General	Entregar una solución EdTech de alta concurrencia y escalabilidad que mejore las tasas de retención y el rendimiento académico de los estudiantes de Educación Superior a través del aprendizaje adaptativo y la gamificación.
Objetivos Específicos	<ol style="list-style-type: none"> <li>Implementar una Arquitectura de Microservicios desacoplada(Escalabilidad).</li> <li>Desarrollar el Motor de Juego que integra la lógica adaptativa con el Historial de Errores.</li> <li>Implementar un flujo de procesamiento asíncrono para el cálculo de rankings y análisis de IA, cumpliendo con el RNF-01 (Bajo Tiempo de Respuesta).</li> <li>Crear el Módulo de Carga Masiva para que los profesores puedan subir contenido curricular.</li> </ol>
Factores de Éxito	<ol style="list-style-type: none"> <li>Cumplimiento del RNF-01 (Rendimiento): El tiempo de respuesta en la ruta crítica síncrona debe ser inferior a 2 segundos.</li> <li>Funcionalidad Adaptativa: El 85% de las preguntas generadas por el sistema deben basarse correctamente en la debilidad conceptual del estudiante.</li> </ol>

	<p>3. Arquitectura Funcional: Despliegue exitoso de la arquitectura de microservicios en GCP con comunicación funcional entre componentes.</p>
Hitos Principales	<ol style="list-style-type: none"> <li>1. Finalización de la Arquitectura Lógica (Vista Lógica/Clases): 26 de Agosto 2025</li> <li>2. Integración Total con Gemini API y Fallback: 20 de Septiembre 2025</li> <li>3. Despliegue de Final: 18 de Noviembre 2025</li> <li>4. Entrega Final del Proyecto (Todas las Vistas y Código Base): 22 de Noviembre 2025</li> </ol>

### Fases del Proyecto y Entregables

Fase	Foco	Entregables Principales
I. Inicio y Planificación	Definición de Alcance y Arquitectura.	Acta de Constitución, Requisitos Funcionales y No Funcionales, Modelo 4+1 (Vistas Lógica y de Desarrollo).
II. Diseño Detallado	Definición de Procesos y Flujos.	Modelo 4+1 (Vistas de Procesos y Física), Diagramas de Secuencia/Actividad, Contratos de API.
III. Implementación Base	Desarrollo de la Capa de Seguridad y Orquestación.	Auth Service (Login/Registro/JWT), Game Engine (Flujo Base de la partida), Conexión inicial a DB.

IV. Implementación de Adaptabilidad	Desarrollo de la lógica de negocio y analítica.	AI Service y Scoring Service (Flujo Asíncrono), Content Service (Carga Masiva), Vista de Escenarios (Validación).
V. Integración y Pruebas	Aseguramiento de la Calidad y Rendimiento.	Planes de Pruebas (Unitarias, de Integración, de Rendimiento), Reporte de Pruebas, Código Validado y Aprobado.
VI. Despliegue y Cierre	Puesta en Producción y Cierre Formal.	Infraestructura de Despliegue Configurada (Clúster/Contenedores), Documentación Operacional, Presentación Final y Acta de Cierre del Proyecto.

#### Interesados, Riesgos y Presupuesto

Ítem	Detalle
Interesados Clave	Estudiantes: Usuarios finales. Profesores: Gestores de contenido y consumidores de reportes. Departamento de TI: Responsables del despliegue y soporte de la infraestructura (GCP). Gerente de Proyecto: Responsable de la ejecución.

Riesgos Principales	<ol style="list-style-type: none"> <li>1. Riesgo Técnico (IA): Exceso de consumo de la cuota <i>Free-Tier</i> de la API de Gemini (Mitigación: Estrategia de Fallback a Contenido Estático).</li> <li>2. Riesgo de Alcance: Dificultad en la integración del <i>Game Engine</i> con los servicios de IA y <i>Scoring</i>.</li> <li>3. Riesgo de Rendimiento: Latencia inaceptable en el flujo síncrono debido a la sobrecarga de la BD Relacional.</li> </ol>
Presupuesto (Estimado)	Desarrollo: [4.032 horas / \$48.600.000 (Suma de los 6 meses de costos fijos salariales)] Infraestructura (GCP Dev/Test): \$\$1.800.000 Servicios de Terceros (API Gemini): \$3.600.000 (Costo anual).
Gerente de Proyecto	Cristian Espinoza
Requerimientos de Aprobación	El proyecto se considera exitoso y aprobado cuando el Patrocinador confirma la validez de los 5 pilares del Modelo 4+1 y se demuestra la operatividad del flujo adaptativo y de rendimiento.

## 10. Asignación de Roles

Para la ejecución exitosa del proyecto Brain Boost bajo la metodología Scrum, el equipo se ha organizado en los siguientes roles, asegurando una clara segregación de funciones y responsabilidades:

Product Owner (PO) - Macarena Bertero

El Product Owner es el único responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo.

Funciones y tareas:

- Definición de la Visión: Establecer la visión del producto, priorizar la funcionalidad del sistema y validar la arquitectura en función de los requisitos de negocio.
- Gestión del Backlog: Crear, mantener y priorizar el Product Backlog (lista de requisitos y tareas), asegurando que el backlog sea transparente, visible y entendido por el Equipo de Desarrollo.
- Aceptación y Cierre: Inspeccionar el incremento del producto en cada Sprint Review para aceptar o rechazar los resultados y tomar decisiones finales sobre el producto.
- Validación Técnica: Liderar la validación de la información de diseño para asegurar que el sistema cumple con los estándares exigidos.

#### Scrum Master (SM) - Claudio Valencia

El Scrum Master es el responsable de promover y apoyar Scrum, actuando como un líder de servicio para el Equipo y la Organización.

#### Funciones y tareas:

- Facilitación: Eliminar impedimentos y obstáculos que ralenticen al equipo de desarrollo.
- Coaching: Asegurar que el equipo comprenda y aplique las prácticas de Scrum.
- Gestión de Ceremonias: Facilitar las reuniones de Sprint Planning, Daily Scrum, Sprint Review y Retrospective.

#### Equipo de Desarrollo - Ignacio Léon

El Equipo de Desarrollo es el conjunto de profesionales que crea un incremento de producto potencialmente entregable en cada sprint.

#### Funciones y tareas:

- Implementación Técnica: Diseño, codificación y pruebas unitarias de los Microservicios CORE en Kotlin/Spring Boot.
- Integración: Asegurar la integración continua de los diferentes microservicios y la conexión con las bases de datos (PostgreSQL y MongoDB).
- Control de Calidad: Participar activamente en las Code Reviews y aplicar las pruebas de calidad (unitarias y de integración) para asegurar que el código cumple con los estándares definidos.

- Gestión de la Arquitectura: Contribuir al diseño de la solución arquitectónica e implementar el deploy de la solución en la plataforma Cloud.

## 11. Metodología

Para la gestión y desarrollo de este proyecto, se ha seleccionado la metodología ágil Scrum. Esta elección se fundamenta en su capacidad para gestionar proyectos complejos de manera eficiente, flexible y adaptable a los requerimientos técnicos y funcionales de una solución basada en microservicios.

Las principales razones para la elección de Scrum son:

- Adaptabilidad: Permite reaccionar de manera ágil a los cambios en los requisitos del proyecto o a los hallazgos inesperados durante el desarrollo, asegurando que el producto final siempre cumpla con los objetivos.
- Entrega de Valor Temprana y Continua: El trabajo se divide en iteraciones cortas, o Sprints, lo que permite entregar funcionalidades operativas de forma incremental y obtener retroalimentación constante.
- Gestión del Riesgo: Al abordar el desarrollo en iteraciones cortas, los riesgos se identifican y gestionan de forma proactiva, evitando que se conviertan en problemas mayores.
- Transparencia: Con la planificación y revisión de cada sprint, todos los involucrados en el proyecto tienen una visibilidad completa del estado del proyecto y de los avances logrados.

El desarrollo se organiza mediante el ciclo iterativo de Scrum, el cual se compone de las siguientes fases o ceremonias:

Fase/Ceremonia de Scrum	Función Principal en el Proyecto
Sprint Planning	Definir el alcance de la iteración. Se selecciona un conjunto de historias de usuario del Product Backlog que se abordarán en el próximo sprint.
Sprint	Desarrollo y Pruebas. Es el periodo de tiempo fijo donde

	el equipo desarrolla, integra y prueba las funcionalidades comprometidas.
Daily Scrum	Sincronización y Detección de Bloqueos. Reunión diaria breve para revisar el avance del día anterior, planificar el día en curso y reportar cualquier impedimento que amenace la entrega.
Sprint Review	Demostración y Retroalimentación. Al final del sprint, se presenta el incremento del producto funcional a los stakeholders para recibir feedback formal.
Sprint Retrospective	Mejora Continua del Proceso. El equipo analiza lo que funcionó y lo que se puede mejorar en el proceso, aplicando estas lecciones al siguiente ciclo de planificación.

## 12. Carta Gantt

La gestión del proyecto se realizó bajo la metodología Scrum con Sprints de dos semanas, lo que permitió un desarrollo iterativo e incremental. La siguiente planificación está alineada con la Carta Gantt del proyecto, que provee una vista detallada de las tareas, su duración y los recursos asignados.

### 12.1 Gestión y Priorización de Tareas

Para la gestión de las tareas y el seguimiento del progreso, el equipo utilizó la herramienta Jira.

- Historias de Usuario y Priorización: Todas las funcionalidades se descompusieron en Historias de Usuario, las cuales fueron registradas en Jira, priorizándose semanalmente según su impacto en el valor del producto y su dependencia técnica.
- Asignación al Sprint: Las tareas se asignaron al Sprint según su prioridad técnica.

- Reuniones Diarias (Daily Scrum): El equipo realizó reuniones cortas y frecuentes para sincronizar el avance, identificar impedimentos y ajustar el foco del trabajo del día.

## 12.2 Carta Gantt del Proyecto

La siguiente tabla resume las principales etapas y hitos del proyecto, con una duración total de 7 Sprints (14 semanas), desde el 16 de agosto hasta el 22 de noviembre.

Etapa / Sprint	Duración	Hito Principal	Recursos Asociados	Dependencias (Predecesoras)
Sprint 1: Pre-desarrollo	2 Semanas (16 al 30 de agosto)	Definición de Arquitectura (Modelo 4+1)	2 Ingenieros en Informática, Product Owner (PO)	N/A
Sprint 2: Base Tecnológica	2 Semanas (31 de agosto al 14 de septiembre)	Auth Service Operativo / Conexión a PostgreSQL	2 Ingenieros en Informática, Product Owner (PO)	Sprint 1
Sprint 3: Integración con IA	2 Semanas (15 al 29 de septiembre)	IA Service Operativo y Mecanismo de Fallback	Diseñador UX/UI, 2 Ingenieros en Informática	Sprint 2 (Auth Service)
Sprint 4: Lógica de Gamificación	2 Semanas (30 de septiembre al 13 de octubre)	Contenerización y Despliegue de Staging	2 Ingenieros en Informática, Diseñador UX/UI, Product Owner (PO)	Sprint 3
Sprint 5: Conexión Frontend	2 Semanas (14 al 27 de octubre)	Flujo Crítico Login → Carga Contenido	2 Ingenieros en Informática, Diseñador	Sprint 2, Sprint 3

		Funcional	UX/UI, Product Owner (PO)	
Sprint 6: Lógica de Juego y Scoring	2 Semanas (28 de octubre al 10 de noviembre)	Flujo de Juego Completo / Scoring Asíncrono Funcional	2 Ingenieros en Informática, Diseñador UX/UI, Product Owner (PO)	Sprint 5
Sprint 7: Despliegue, Cierre y Entrega Final	2 Semanas (11 al 22 de noviembre)	UAT / Corrección de Bugs /Despliegue/ Informe Final	Todo el equipo	Sprint 6

Para la revisión detallada de la planificación, el cronograma completo del proyecto se encuentra disponible en el siguiente enlace:

## 13. Marco Teórico

### 13.1 Fundamentos Tecnológicos del Proyecto

El proyecto Brain Boost se sustenta en paradigmas de desarrollo y arquitecturas modernas diseñadas para la escalabilidad, rendimiento y mantenibilidad, abordando la naturaleza dinámica y de alta concurrencia de una aplicación educativa.

#### 1. Arquitectura de Microservicios:

La arquitectura se basa en el Patrón de Microservicios, donde la aplicación se estructura como una colección de servicios pequeños, autónomos e implementados de forma independiente. Este enfoque:

- **Aísla Fallos:** Un fallo en un servicio no necesariamente derriba todo el sistema.
- **Permite la Escalabilidad Horizontal:** Cada servicio puede escalarse de forma independiente según su carga de trabajo (ej., el Game Engine puede requerir más réplicas que el User Service).

- Alta Cohesión y Bajo Acoplamiento: Cada servicio encapsula su propia lógica de negocio y persistencia, facilitando la evolución tecnológica y el desarrollo paralelo.

## 2. Mecanismos de Conurrencia y Asincronía

Para garantizar el Rendimiento, se utiliza el patrón de Comunicación Asíncrona basado en Colas de Mensajes (Pub/Sub). Este patrón permite que el flujo de juego síncrono delegue inmediatamente las tareas pesadas (cálculo de *ranking* y análisis de IA) a servicios secundarios. Esto previene el bloqueo de recursos y mejora la experiencia del usuario.

## 3. Estrategia de Persistencia Híbrida

El sistema emplea una Estrategia de Persistencia Híbrida, utilizando múltiples tipos de bases de datos, cada una elegida por su idoneidad para una tarea específica:

- Bases de Datos Relacionales (PostgreSQL): Utilizada para datos que requieren alta integridad transaccional , como perfiles de usuario, credenciales y las relaciones estables de contenido y *ranking*.
- Bases de Datos NoSQL (MongoDB): Utilizada por el Content Service para el almacenamiento de archivos o la gestión de datos con esquema flexible, como *logs* de carga masiva o métricas de juego de gran volumen, donde la velocidad y flexibilidad son primordiales sobre la integridad transaccional estricta.

## 4. Inteligencia Artificial como Servicio

El proyecto integra modelos de IA Generativa (Google Gemini 2.5 Flash) como un servicio dedicado (AI Service). Este enfoque de permite:

- Innovación Continua: El proyecto se beneficia de los avances del modelo de IA sin requerir grandes inversiones en hardware o entrenamiento de modelos propios.
- Generación Dinámica de Contenido: La IA se utiliza para la generación just-in-time de preguntas o retroalimentación inteligente, crucial para la Adaptabilidad.

## 13.2 Estado del Arte y Benchmarking

La propuesta de Brain Boost se posiciona en el contexto de las plataformas de educación adaptativa (EdTech), diferenciándose por su foco en la micro-personalización del contenido impulsada por IA para el segmento de Educación Superior.

Para establecer el Estado del Arte, se comparan soluciones internacionales y locales que utilizan estrategias de gamificación y adaptabilidad, destacando nuestra diferenciación por nivel de complejidad y arquitectura:

Solución / Proyecto	Origen	Enfoque Principal	Diferenciador Clave de Brain Boost
Duolingo / Babbel	Internacional	Aprendizaje de idiomas gamificado.	Brain Boost: Se enfoca en la adaptación por debilidad conceptual y la generación de contenido temático a demanda mediante IA (Gemini), más allá de un banco de preguntas estático.
Quizizz / Kahoot	Internacional	Cuestionarios y evaluaciones en tiempo real.	Brain Boost: Énfasis en el refuerzo asíncrono y el cálculo de <i>ranking</i> desacoplado. Estos sistemas suelen ser síncronos y no ofrecen un análisis de debilidades tan

			profundo.
Umáximo	Chile	Plataforma educativa de Matemática, Comprensión Lectora y Educación Socioemocional, basada en el juego.	Brain Boost: Nuestra solución está diseñada para la integración modular con sistemas académicos universitarios y maneja la complejidad de los programas de Educación Superior.
Adaptativamente	Chile	Plataforma que entrega refuerzos personalizados alineados con el currículum del MINEDUC.	Brain Boost: La diferenciación principal de Brain Boost es la Estrategia de Persistencia Híbrida y el diseño de la Vista de Procesos Asíncronos para manejar el cálculo de métricas en sistemas distribuidos.

## 14. Alcances del Proyecto

### 14.1 Funcionalidades Incluidas

- Gestión de Perfiles por Rol: La aplicación incluirá tres tipos de roles de usuario: Estudiante, Profesor, y Administrador. Cada rol tendrá un conjunto de permisos y funcionalidades específicas.
- Módulo de Estudiante: Este perfil tendrá acceso exclusivo al módulo de juego. Los estudiantes podrán seleccionar editar aspectos básicos de su perfil, como su foto o ícono.ar materias precargadas según su carrera, jugar, y visualizar su progreso. Podrá
- Módulo de Profesor: Los profesores tendrán acceso a un módulo de visualización en el cual podrán ver los rankings y el progreso de los estudiantes de sus cursos. Además, tendrán la capacidad de cargar contenido teórico (documentos, extractos, etc.) por materia. Este contenido es la fuente que posteriormente la Inteligencia Artificial (IA) analizará para generar preguntas dinámicas. Adicionalmente, el profesor podrá crear, editar y eliminar preguntas de forma manual en el banco de contenido teórico de las materias que tenga asignadas. Finalmente, tendrán la capacidad de cargar masivamente alumnos a las asignaturas que tengan a su cargo, facilitando la inscripción y la administración de grandes grupos de alumnos.
- Módulo de Administrador: Este rol tendrá los privilegios más altos, con acceso a todas las funcionalidades del sistema. Podrá gestionar usuarios, contenido, y configuraciones generales de la aplicación.
- Módulo de Gamificación: Implementación de mecánicas de juego como niveles, desafíos, puntajes, y un sistema de logros para mantener a los estudiantes motivados con el aprendizaje. El sistema registrará los errores de los usuarios, lo que permitirá generar cuestionarios de refuerzo adaptativos, enfocados en las áreas de debilidad detectadas.
- Banco de Contenido Teórico: La aplicación contendrá un banco de preguntas y respuestas relacionadas con las materias teóricas de la carrera, organizadas de manera jerárquica: una carrera tiene muchas asignaturas, y cada asignatura contiene muchas preguntas. Esto permite un manejo y una categorización del contenido clara.
- Integración Inteligente con Google Gemini: Se desarrollará un microservicio dedicado a la comunicación con la API de Gemini. Esta integración no solo generará preguntas dinámicas y personalizadas, sino que también analizará el

historial de respuestas del usuario para identificar áreas de debilidad. La inteligencia artificial fomentará el contenido en el que el usuario comete más errores, reforzando así el aprendizaje de manera dirigida y eficiente. El sistema cuenta con un módulo de inteligencia que, al recibir la solicitud de un nuevo cuestionario, procesa el historial de rendimiento del usuario. Este análisis permite identificar los temas o conceptos en los que el estudiante ha tenido más dificultades. Con esta información, el sistema genera preguntas de refuerzo, asegurando que el contenido se adapte de forma inteligente y dirigida a las necesidades de aprendizaje del usuario. Este enfoque asegura un proceso de aprendizaje adaptativo, donde la inteligencia artificial utiliza el rendimiento del estudiante para dirigir y reforzar el conocimiento de manera personalizada.

- Sistemas de Estadísticas y Analíticas: Se registrarán las interacciones y el progreso del usuario en una base de datos no relacional para futuras analíticas, permitiendo comprender mejor los patrones de aprendizaje.
- Versión web: Se incluirá el desarrollo de una aplicación web cuyo alcance estará estrictamente limitado a las funcionalidades de gestión y administración de contenido, siendo la plataforma móvil la única destinada a la experiencia de juego. El administrador utilizará la versión web para gestionar usuarios, roles y configuraciones generales del sistema. Mientras que el perfil de profesor podrá cargar contenido teórico masivo, gestionar el banco de preguntas y cargar alumnos de manera masiva a las asignaturas asignadas.
- Interfaz de Usuario (UI) y Experiencia de Usuario (UX) intuitiva: El diseño se centrará en una interfaz limpia y una navegación sencilla, asegurando una experiencia de juego fluida y accesible.

## 14.2 Funcionalidades Excluidas

- Desarrollo para iOS: El proyecto se enfocará exclusivamente en la plataforma Android utilizando lenguajes nativos. No se contempla una versión nativa para el sistema operativo iOS en esta etapa.
- Integración con el Sistema Académico del Duoc UC: El juego funcionará de manera independiente del sistema de registro y notas oficial de la institución.

- Modo Multijugador en Tiempo Real: Las funcionalidades multijugador se limitarán a tablas de clasificación (rankings) y desafíos asincrónicos, para mantener el foco en el aprendizaje individual.

## 15. Requerimientos del Sistema

### 15.1. Requisitos Funcionales

El sistema debe permitir que los usuarios se autentiquen de manera segura, asignando roles con permisos específicos:

- **Requisitos del Estudiante:**

- El estudiante debe poder acceder al juego y seleccionar su carrera y materia.
- El sistema debe analizar el historial de respuestas del estudiante para identificar los temas o conceptos en los que comete más errores.
- El juego debe ajustar dinámicamente el contenido, priorizando la presentación de preguntas y desafíos sobre los temas que el estudiante necesita reforzar. Esto asegura un aprendizaje personalizado y enfocado en sus debilidades.
- El sistema debe mostrar al estudiante su puntaje, progreso individual y las áreas específicas de mejora.
- El estudiante debe poder editar su foto o ícono de perfil.

- **Requisitos del Profesor:**

- El profesor debe tener acceso únicamente a los datos de los estudiantes y el contenido de las asignaturas que le han sido asignadas por el Administrador.

- El sistema debe permitir al profesor ver los rankings y acceder a estadísticas de rendimiento detalladas del grupo y de los estudiantes individuales en sus cursos asignados.
- El profesor debe tener la funcionalidad para cargar contenido teórico (ej. archivos PDF o texto) por materia. Este contenido debe ser almacenado para que el Módulo de IA lo analice posteriormente.
- El profesor debe tener acceso a un módulo de administración para crear, editar o eliminar preguntas de forma manual en el banco de contenido teórico de las materias bajo su cargo.
- El profesor debe tener la capacidad de cargar masivamente alumnos a las asignaturas que tenga asignadas, utilizando un archivo, lo que facilitará el proceso de inscripción

- **Requisitos del Administrador:**

- El administrador debe tener privilegios para gestionar todos los aspectos de la aplicación, incluyendo usuarios, roles, materias y el contenido general.

- **Requisitos Generales:**

- El sistema debe permitir al usuario iniciar una sesión de juego o desafío en cualquier momento.
  - El sistema debe presentar preguntas teóricas de forma aleatoria o temática.
  - El sistema debe validar la respuesta del usuario y proporcionar retroalimentación inmediata, utilizando la API de Google Gemini para generar respuestas dinámicas y personalizadas.
  - El sistema debe registrar y actualizar el puntaje y el progreso del usuario después de cada sesión de juego.
  - El sistema debe mostrar un ranking general y por curso de los puntajes más altos entre los usuarios.

## 15.2. Requisitos No Funcionales

- Rendimiento: El tiempo de respuesta de la aplicación para las interacciones del usuario no debe exceder los 2 segundos, y el tiempo de carga de las preguntas de Gemini no debe superar los 5 segundos.

- Seguridad: Las contraseñas de los usuarios deben ser encriptadas. La comunicación con los microservicios debe realizarse a través de canales seguros (HTTPS).
- Usabilidad: La interfaz de usuario debe ser intuitiva y fácil de usar para un público estudiantil.
- Escalabilidad: La arquitectura de microservicios debe permitir un crecimiento futuro en el número de usuarios y la adición de nuevos módulos sin comprometer el rendimiento general.%
- Disponibilidad: El sistema debe tener una disponibilidad del 99% para garantizar que los usuarios puedan estudiar en cualquier momento.

### 15.3 Matriz de Trazabilidad

La siguiente matriz asegura la correcta vinculación entre los requerimientos del sistema, los módulos desarrollados para su implementación y las pruebas realizadas para validar su correcto funcionamiento.

ID Requerimiento	Descripción (RF / RNF)	Módulo/Microservicio	Tipo de Prueba	Estado
RF-E.01	El estudiante debe poder autenticarse de forma segura (Login).	Auth Service	Prueba Unitaria / Integración	Cumplido
RF-E.02	El juego debe ajustar dinámicamente el contenido, priorizando el refuerzo.	IA Service / Game Engine	Prueba de Integración / Funcional	Cumplido

RF-P.03	El profesor debe poder ver los rankings y estadísticas detalladas.	Scoring Service / User Service	Prueba Funcional	Cumplido
RF-P.04	El profesor debe cargar contenido teórico (PDFs o texto).	Content Service	Prueba Unitaria / Funcional	Cumplido
RF-G.01	El sistema debe validar la respuesta y generar retroalimentación inmediata.	IA Service / Game Engine	Prueba Funcional	Cumplido
RNF-01	Tiempo de respuesta de la aplicación no debe exceder 2 segundos.	BFF (Backend for Frontend)	Prueba de Rendimiento	Cumplido
RNF-02	Contraseñas de los usuarios deben ser encriptadas (Hashing).	Auth Service	Prueba de Seguridad	Cumplido
RNF-04	Arquitectura de microservicios debe permitir crecimiento futuro (Escalabilidad).	Arquitectura General (GCP)	Prueba de Carga	Cumplido

## 16. Diseño de Interfaz y Experiencia de Usuario

El diseño de la interfaz de Brain Boost se centra en la simplicidad, coherencia visual y eficiencia de flujo, dado que atiende a dos tipos de usuarios (Administrativos/Profesores), y un usuario final (Alumno). Se aplica el principio de Diseño Responsive en la Web para la gestión y Diseño Nativo (Material 3) en la aplicación móvil para el juego.

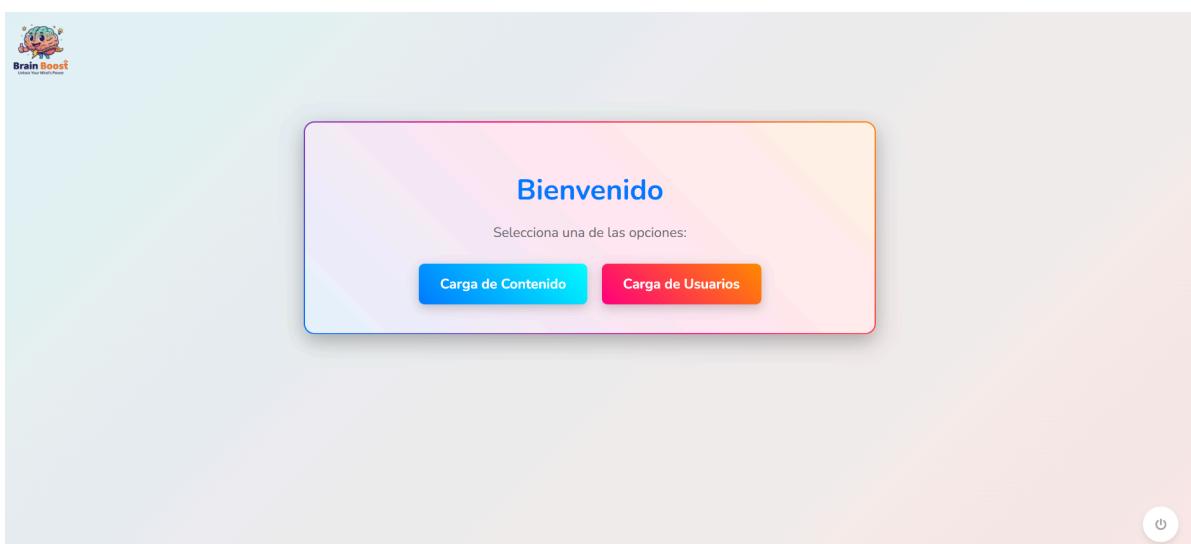
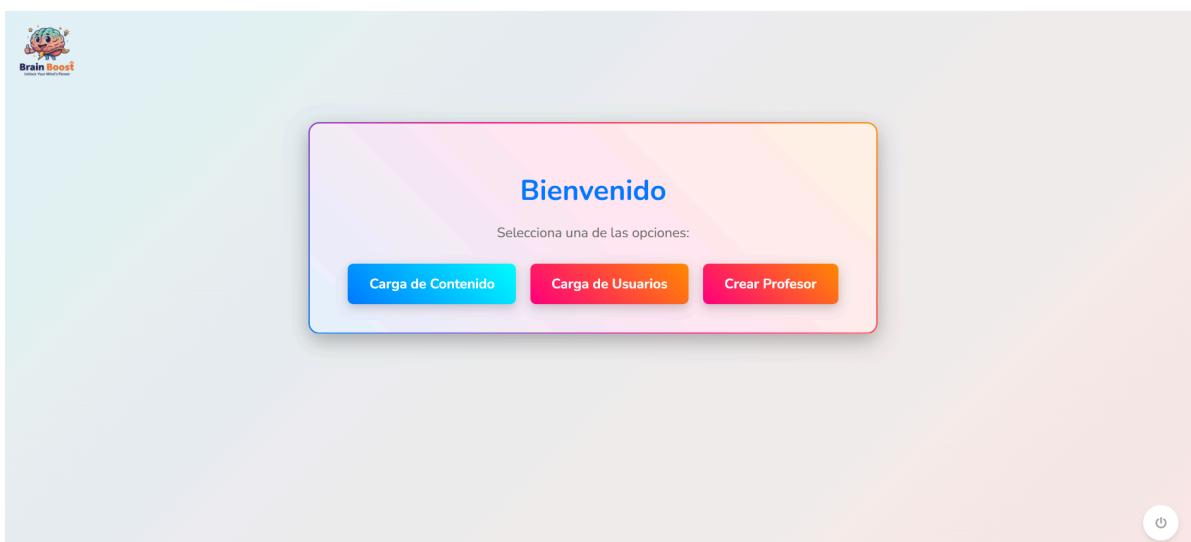
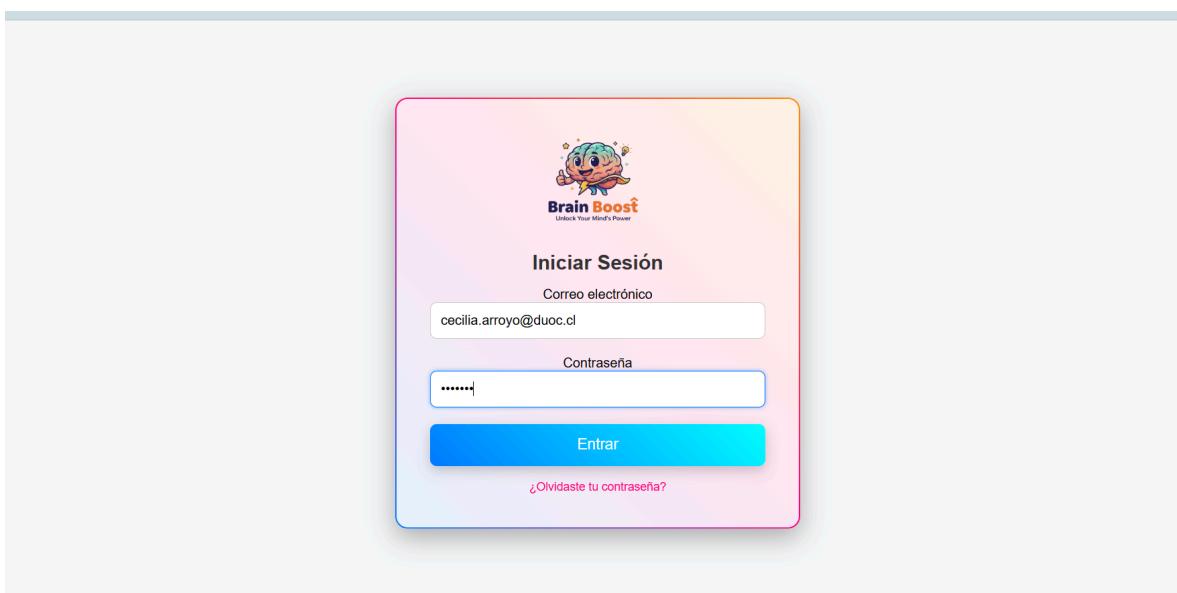
Las funciones clave de Carga de Contenido y Creación de Usuarios están disponibles tanto en la Aplicación Web como en la App Móvil. Sin embargo, anticipamos que los administradores y profesores darán prioridad a la Aplicación Web para estas tareas intensivas, debido a la comodidad de la carga masiva y la mejor visibilidad en la interfaz de escritorio.

### 16.1. Flujo Principal de la Aplicación Web (Gestión y Contenido)

El flujo de la aplicación Web está diseñado para los perfiles de Administrador y Profesor, priorizando la gestión de usuarios y la carga de contenido de manera intuitiva.

### 16.2 Inicio de Sesión y Dashboard

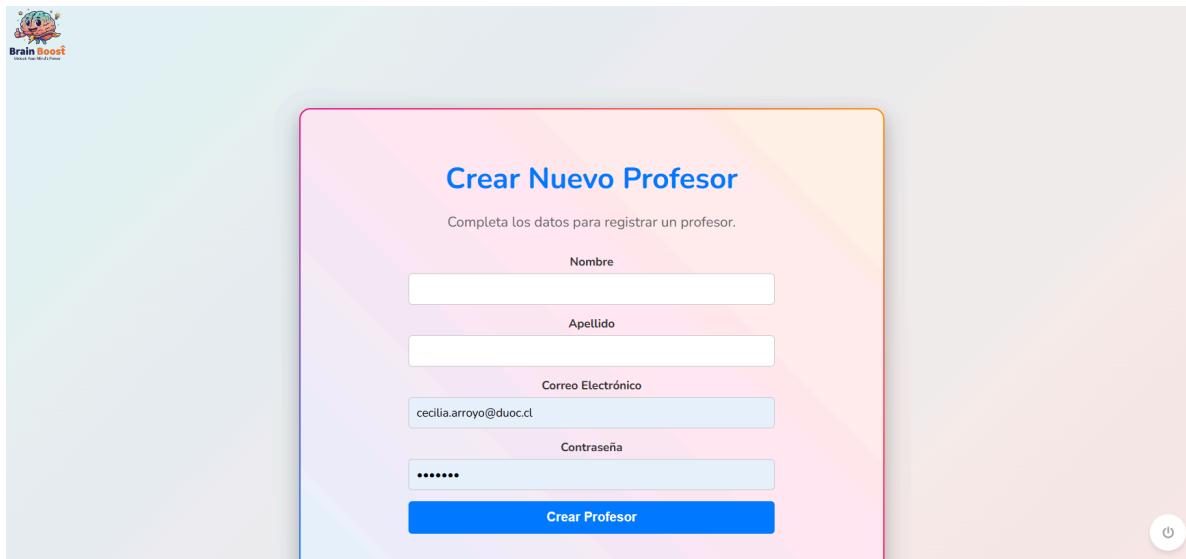
Inicio de Sesión (Login): Es la misma interfaz para todos los perfiles (Administrador y Profesor).



Perfil	Pantalla Principal (Dashboard)	Funcionalidades Clave
Administrador	Acceso a gestión de cuentas.	Carga de Usuarios (Masiva) o Creación de Profesores (Individual).
Profesor	Acceso a sus archivos, contenido, y listado de alumnos.	Carga de Contenido y Gestión de Secciones de Alumnos.

### 16.3 Flujo de Carga de Usuarios y Contenido

Creación de Profesor: Interfaz simple donde el Administrador ingresa los datos básicos para habilitar la cuenta del nuevo docente.



Carga de Contenido (Profesor): El profesor puede acceder a la interfaz para subir sus documentos (PDF o texto) para su procesamiento.

← Volver

## Carga de Contenido

Hola Mundo

Arrastra y suelta documentos aquí

Enviar



Archivos Procesados: Pantalla donde el profesor visualiza un listado de sus archivos y el estado de procesamiento (listo o pendiente), permitiéndole gestionarlos y asignarlos.

Brain Boost

Carga de Contenido

## Contenido

### Estado Archivos Procesados

Leyenda de estados: ● Completado ● Procesando ● Errores

SECCIÓN	ASIGNATURA	TIPO DE CARGA	FECHA ACTUALIZACIÓN	ESTADO
A	Asignatura	Web	20-09-2025	●
B	Asignatura	App	21-09-2025	●
C	Asignatura	Web	22-09-2025	●

Enviar



Listado de Alumnos: El profesor puede visualizar la lista de alumnos inscritos en sus secciones, permitiendo la gestión rápida de la sección.

[Carga de Alumnos](#)

### Alumnos Inscritos

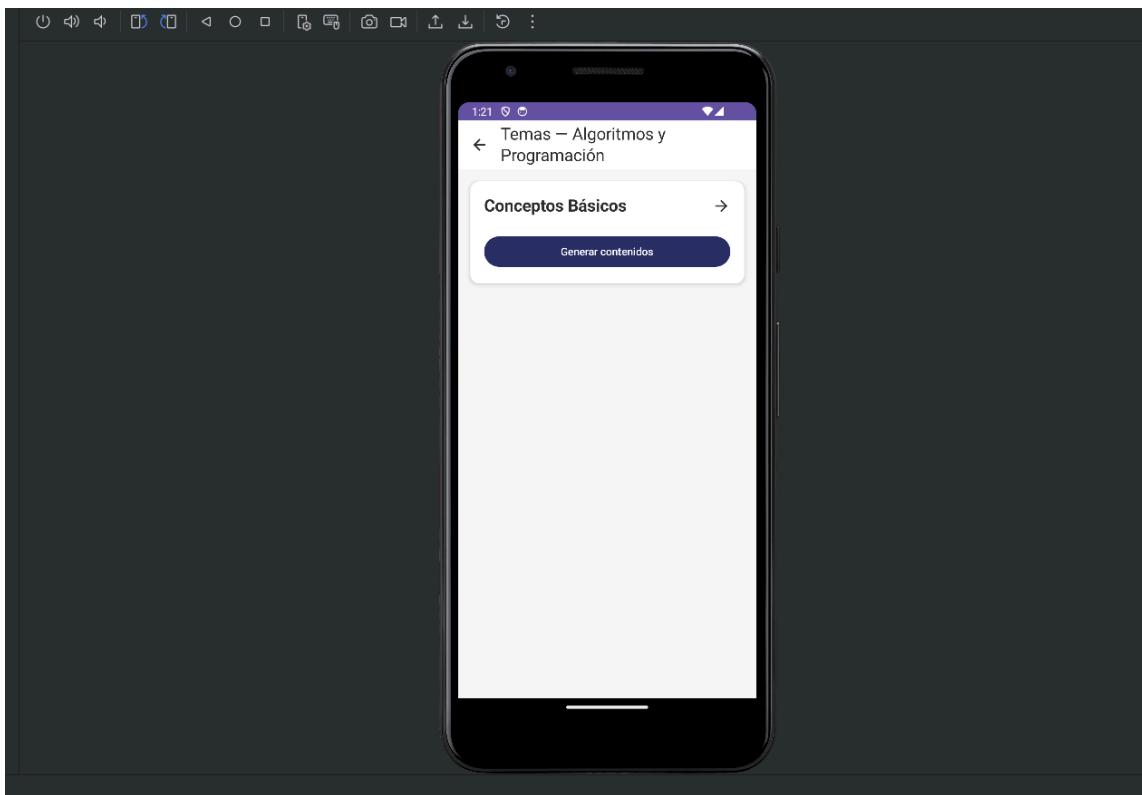
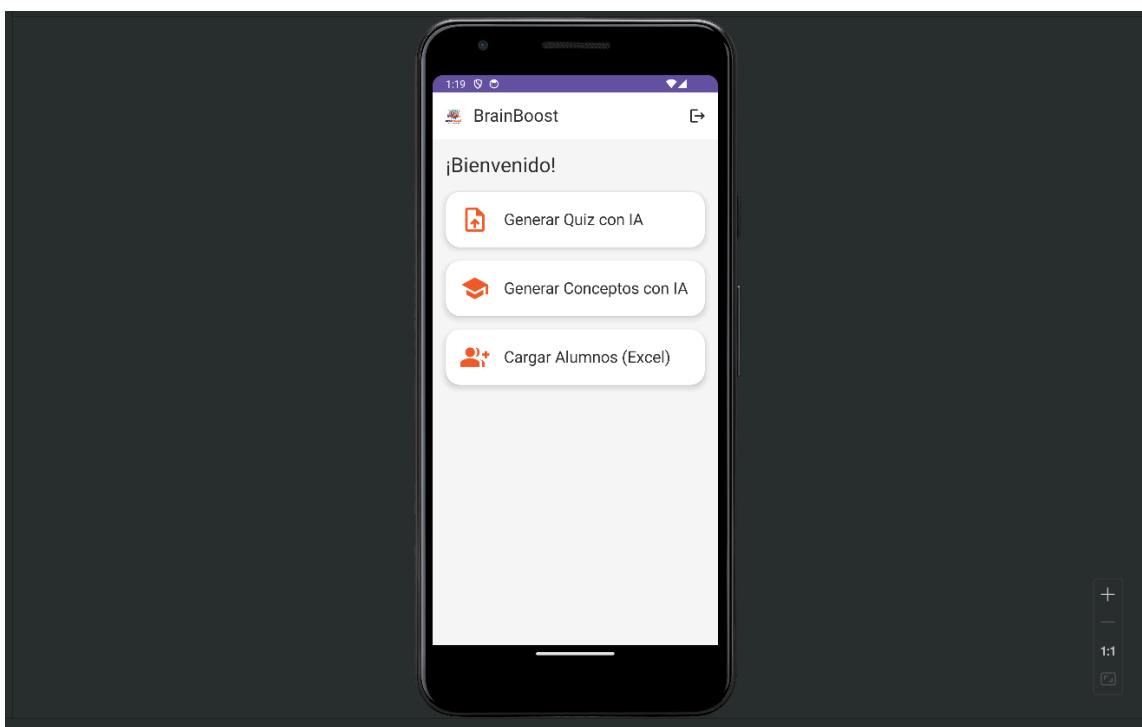
NOMBRE	RUT	CORREO	SECCIÓN
Nombre Apellido	11.222.333-4	usuario@ejemplo.com	A
Otro Nombre	98.765.432-1	otro.usuario@ejemplo.com	B

## 16.4. Flujo de la Aplicación Móvil (Profesor y Alumno)

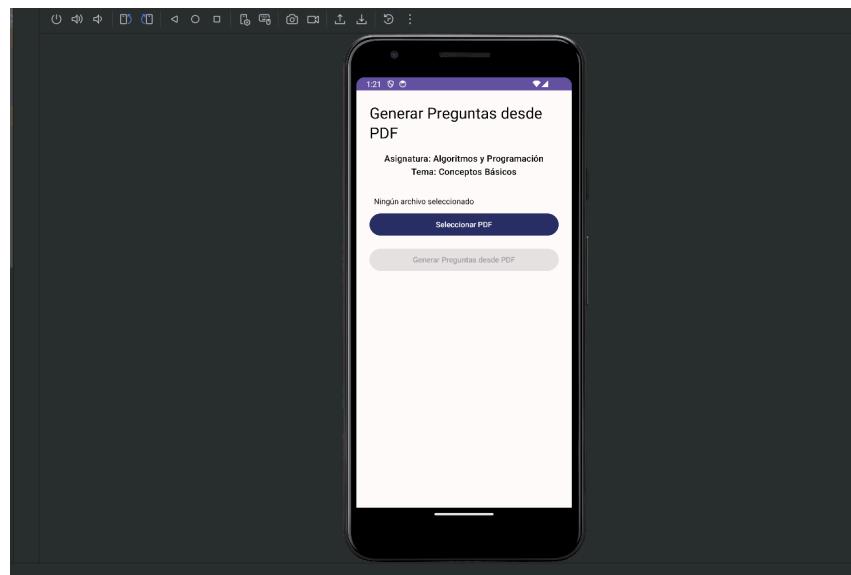
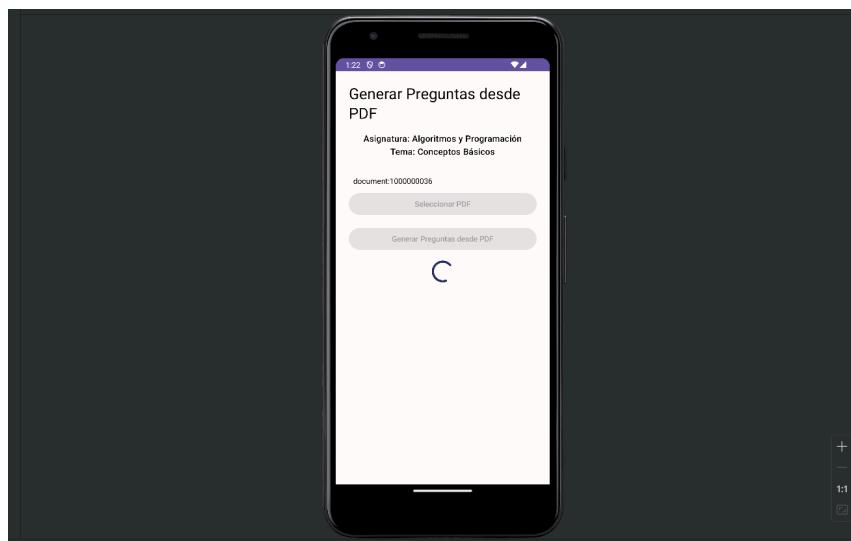
El diseño de la aplicación móvil se enfoca en la accesibilidad y la gamificación de la experiencia. Se utiliza Android Jetpack Compose Material 3 para asegurar una interfaz moderna.

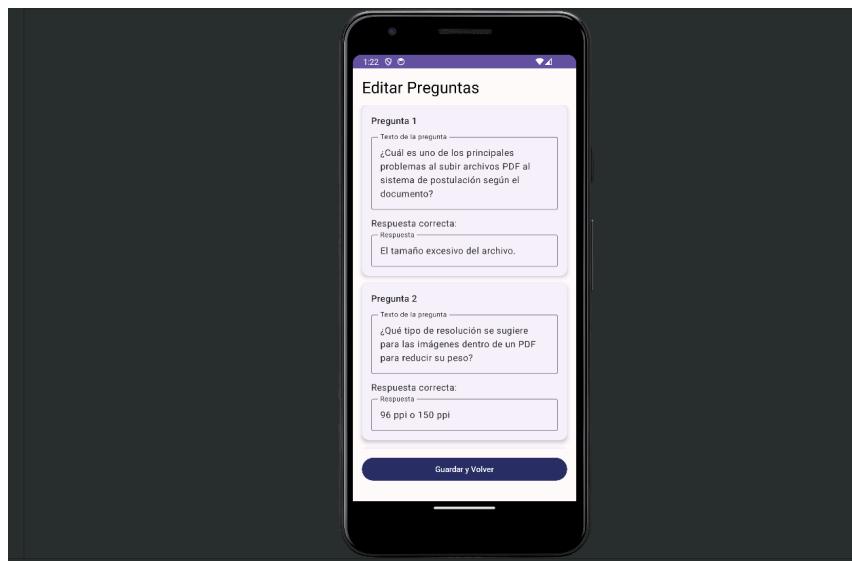
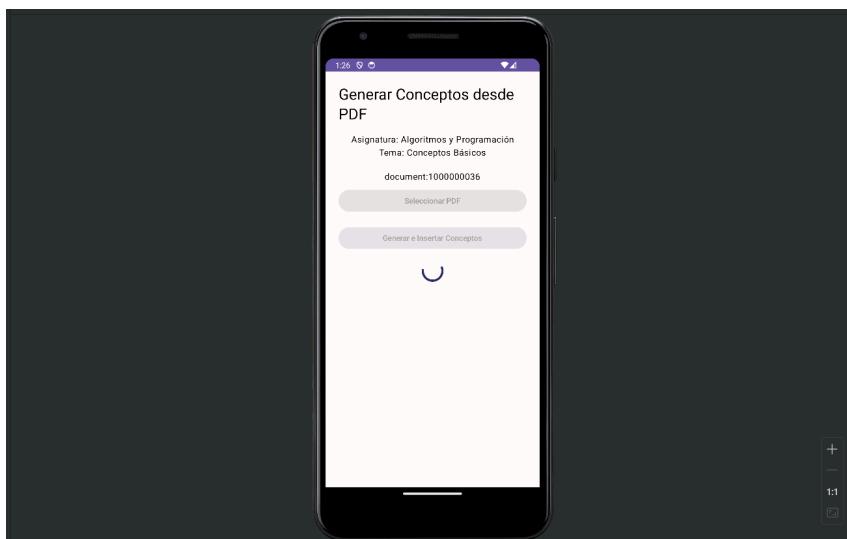
### 16.4 Vista Móvil del Profesor (Generación de Contenido)

Pantalla Principal: Acceso rápido a Asignaturas y Temas, y a las opciones de generación.

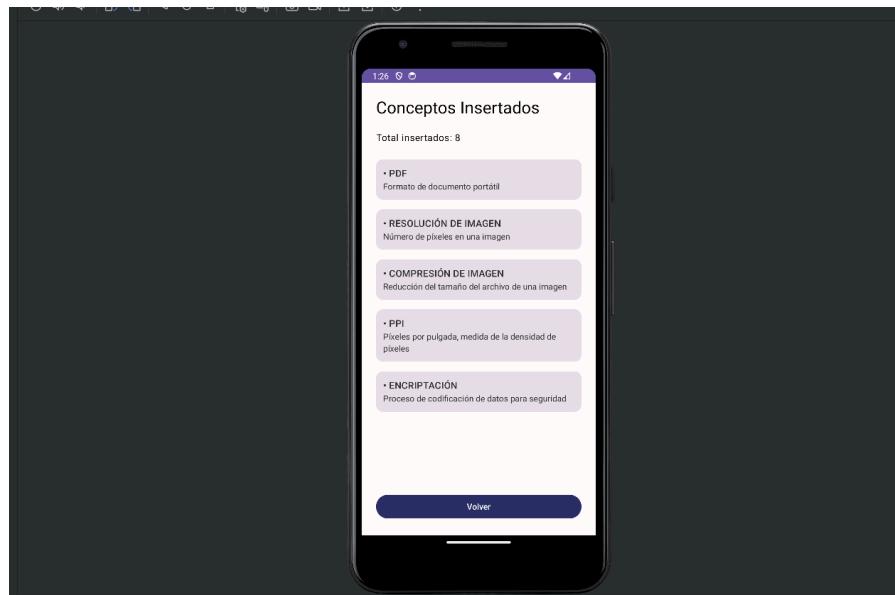
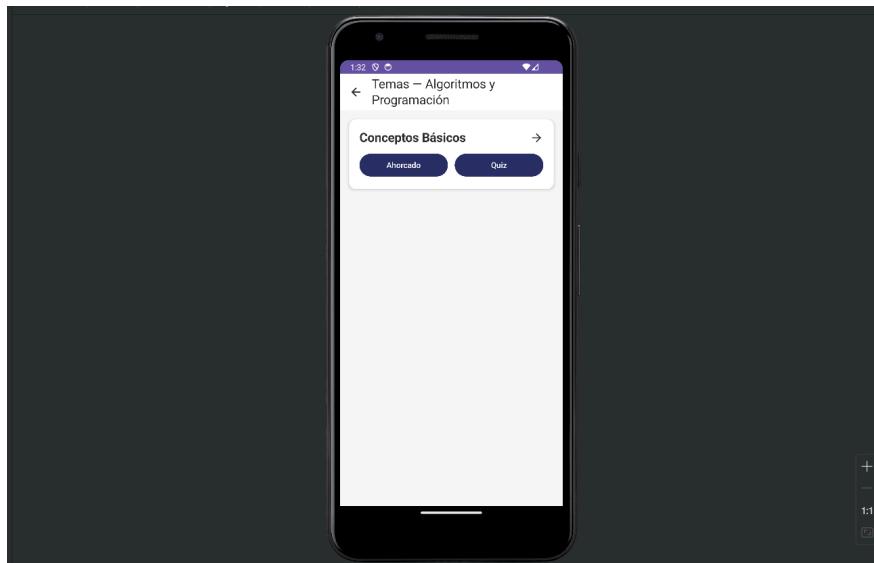
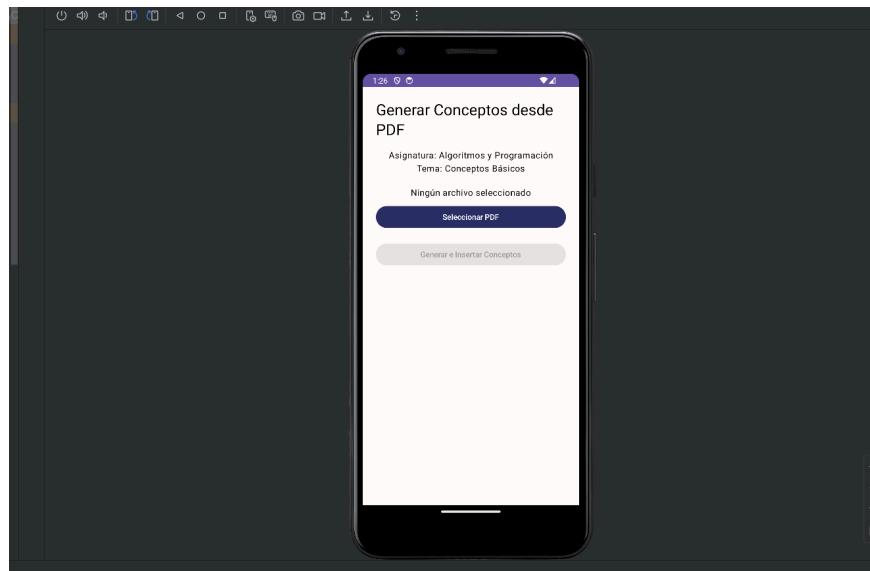


Generación de Quiz con IA: El profesor sube un PDF. El sistema genera 5 preguntas mediante IA. El profesor revisa el resultado antes de guardarlo.

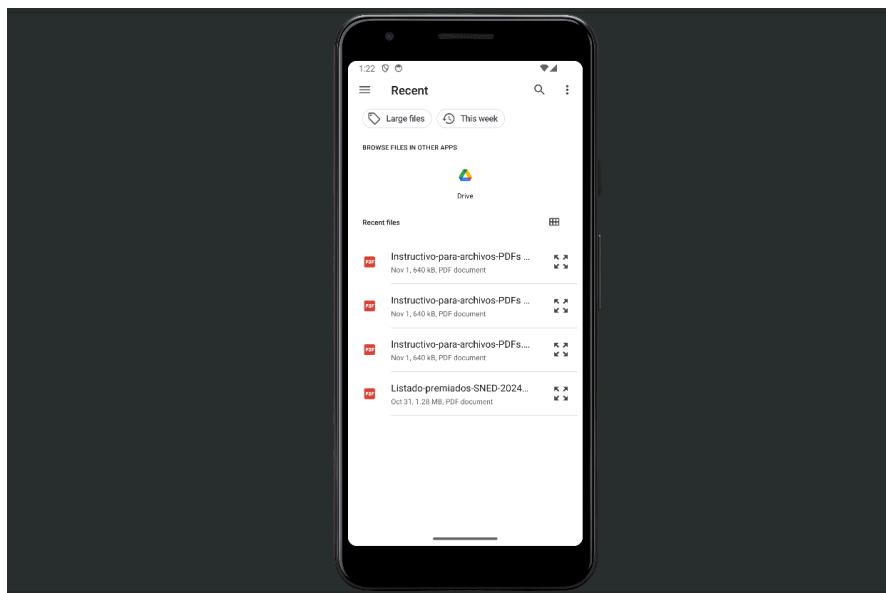




Generación de Conceptos con IA: El profesor sube un PDF y el sistema extrae Conceptos Clave para el juego Hangman

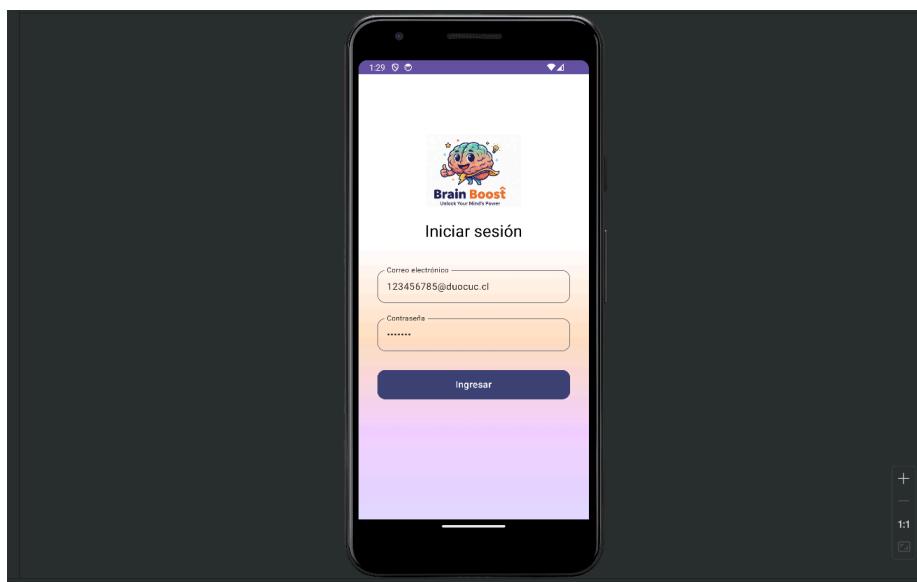


Carga de Alumnos: Permite al profesor subir una Plantilla Excel de alumnos.

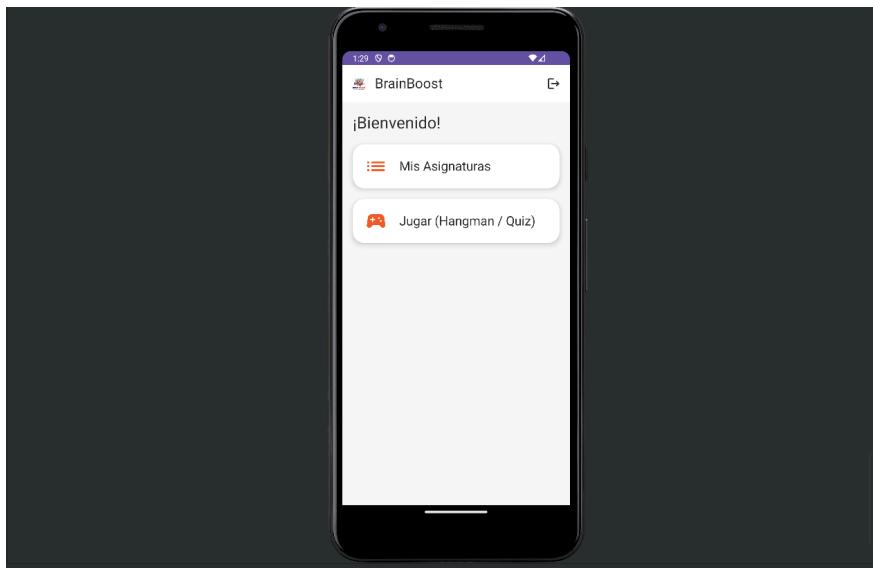


## 2.2 Vista Móvil del Alumno

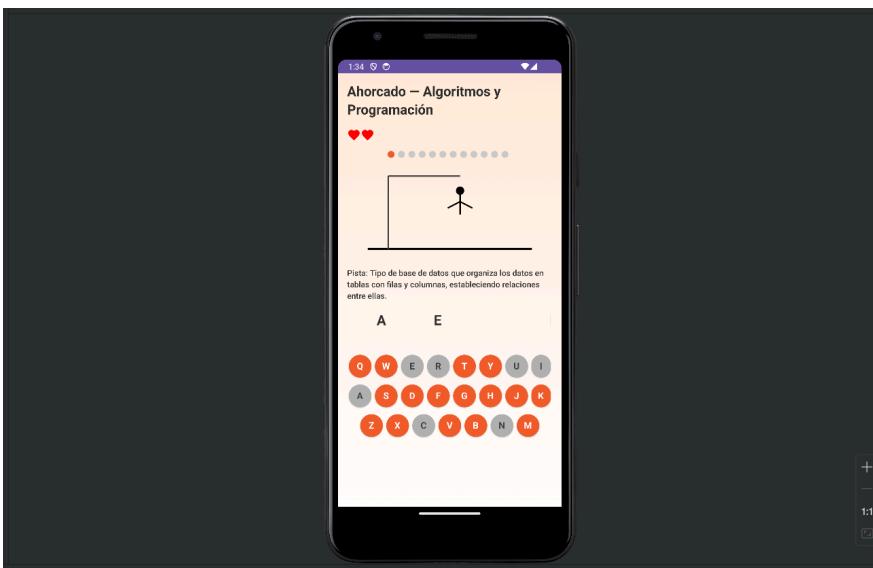
Login Alumno: Pantalla de inicio de sesión simple.



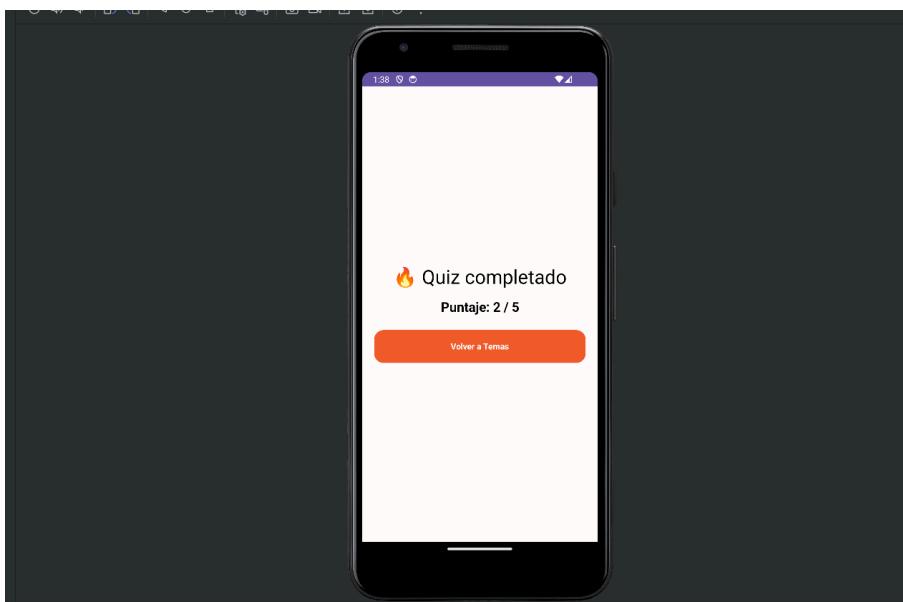
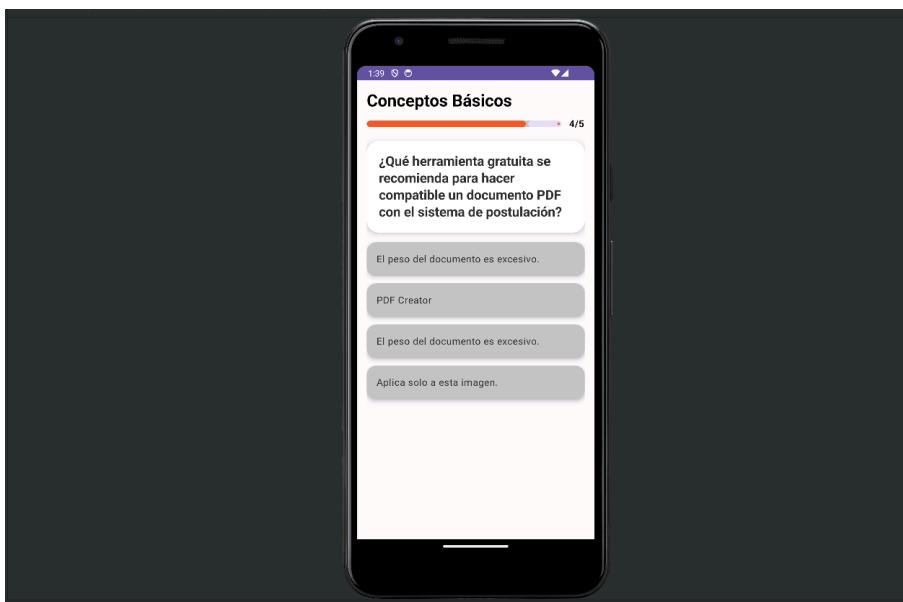
Asignaturas: Pantalla principal que lista las asignaturas y los Juegos disponibles para cada una.



Juego Hangman: El alumno selecciona un tema y juega Hangman utilizando los conceptos generados por el profesor.



Juego Quiz: El alumno responde 5 preguntas aleatorias y adaptativas generadas por el sistema.



## 17. Implementación del Proyecto

### 17.1 Descripción General del Sistema

El sistema Brain Boost es una plataforma de refuerzo académico Cloud-Native, diseñada como una solución EdTech de vanguardia para la educación superior.

El propósito principal de Brain Boost es transformar el repaso pasivo de contenidos teóricos en una experiencia de aprendizaje activo, gamificado y adaptativo. Mediante la

implementación de un motor de Inteligencia Artificial, busca mejorar significativamente la retención de conocimiento de los estudiantes al identificar sus áreas de debilidad y proporcionar refuerzo personalizado de manera instantánea.

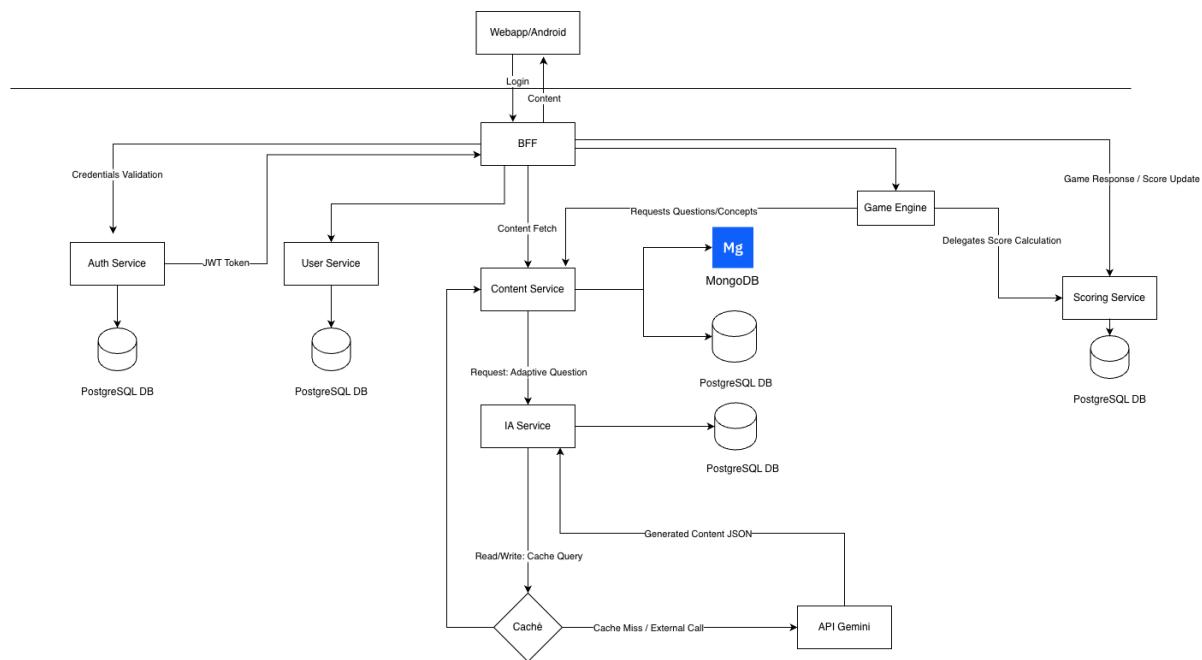
El sistema está diseñado para atender a tres perfiles de usuario dentro del ecosistema educativo del Duoc UC:

1. Estudiantes: Usuarios principales que interactúan con el módulo de juego para reforzar materias teóricas.
2. Profesores: Usuarios encargados de la carga de contenido base (documentos) y de la supervisión del progreso y rendimiento del grupo.
3. Administradores: Usuarios con privilegios de gestión total de la plataforma (usuarios, roles y configuración).

La funcionalidad de Brain Boost se estructura en tres módulos principales, soportados por una arquitectura de microservicios:

- Módulo de Juego (Gamificación): Es la interfaz principal del estudiante. Incorpora mecánicas de juego (puntajes, *rankings*, desafíos) para motivar la participación y hacer el aprendizaje lúdico.
- Módulo de Gestión de Contenido: Permite a los profesores cargar documentos teóricos (PDFs/Texto) y administrar el banco de preguntas. Es el input inicial de la Inteligencia Artificial.
- Módulo de Inteligencia Artificial: El motor adaptativo. Se comunica con la API de Google Gemini para generar preguntas dinámicas de refuerzo, basándose en el análisis en tiempo real del desempeño individual del estudiante.
- Módulo de Estadísticas: Proporciona al profesor y administrador analíticas detalladas sobre el rendimiento grupal e individual, incluyendo métricas de progreso y áreas de conocimiento débiles.

## 17.2 Diagrama de Arquitectura



arquitectura-v1.3.5.png

El proyecto se desarrollará bajo una arquitectura de Microservicios, lo cual garantiza la escalabilidad horizontal y la modularidad de cada componente. El sistema se desplegará en la nube de Google Cloud Platform (GCP), utilizando el patrón Backend for Frontend (BFF) para optimizar el rendimiento de la aplicación móvil y la interfaz web.

El BFF actúa como el único punto de entrada para todos los clientes (aplicación móvil y versión web). Su función principal no es solo enrutar peticiones, sino también orquestar y consolidar la información proveniente de múltiples microservicios en una única respuesta eficiente. Este componente es esencial para la seguridad, ya que actúa como un perímetro que protege las claves privadas del sistema de ser expuestas al cliente final.

### 17.2.1. Persistencia y Almacenamiento de Datos

El sistema utiliza dos tipos de almacenamiento, cada uno adaptado a un propósito específico para optimizar la carga de trabajo y el rendimiento:

- **Base de Datos Transaccional (PostgreSQL):** Esta es la base de datos principal y central del sistema. Se utiliza para almacenar datos críticos que requieren alta integridad referencial, incluyendo:

- Información de usuarios y la asignación de roles.
  - Estructura definitiva del banco de contenido académico (carreras, asignaturas, temas y las preguntas que el profesor gestiona mediante el CRUD).
  - Registros de la lógica de gamificación (juegos, métricas y puntajes).
- 
- Base de Datos No Relacional (MongoDB): Este almacenamiento se destina exclusivamente a la ingesta y el *staging* de datos voluminosos. Se utiliza solamente para recibir y almacenar el contenido teórico (archivos o texto) que el profesor carga de manera masiva. Una vez que este contenido es procesado por el IA Service y transformado en preguntas estructuradas, el contenido original puede ser gestionado, liberando esta base de datos de tareas transaccionales.

#### 17.2.2. Microservicios Principales

La lógica de negocio se divide en los siguientes microservicios independientes:

##### Auth Service

Es la autoridad de identidad del sistema. Se encarga de procesar el registro y el inicio de sesión, validando credenciales y generando el JSON Web Token (JWT), que se usa para la autorización del usuario.

##### User Service

El User Service gestiona el ciclo de vida de las cuentas de usuario, incluyendo la creación de nuevos perfiles, la asignación de roles y la gestión de la carga masiva.

##### Content Service

Maneja la fuente del contenido académico. Es el único servicio con acceso a ambas bases de datos:

- Se conecta a MongoDB para leer el contenido en bruto que ha subido el profesor.
- Se conecta a PostgreSQL para la gestión CRUD del banco de preguntas estructuradas (preguntas y conceptos).

- Se comunica con el IA Service para solicitar contenido adaptativo o la transformación de datos brutos.

### Game Engine

Es el orquestador de la experiencia de juego y el responsable de la lógica transaccional de la partida. Recibe las peticiones de inicio de juego del BFF, interactúa con el Content Service para obtener el pool de preguntas/conceptos necesarios (según el tipo de juego: Quizz o Ahorcado) y gestiona el estado en la tabla juegos (Ej: intentos\_restantes, estado\_partida). Registra cada respuesta o acción del usuario en la tabla metricas y al finalizar la partida, envía el resultado detallado al Scoring Service para el cálculo del puntaje.

### IA Service (Servicio de Inteligencia Artificial)

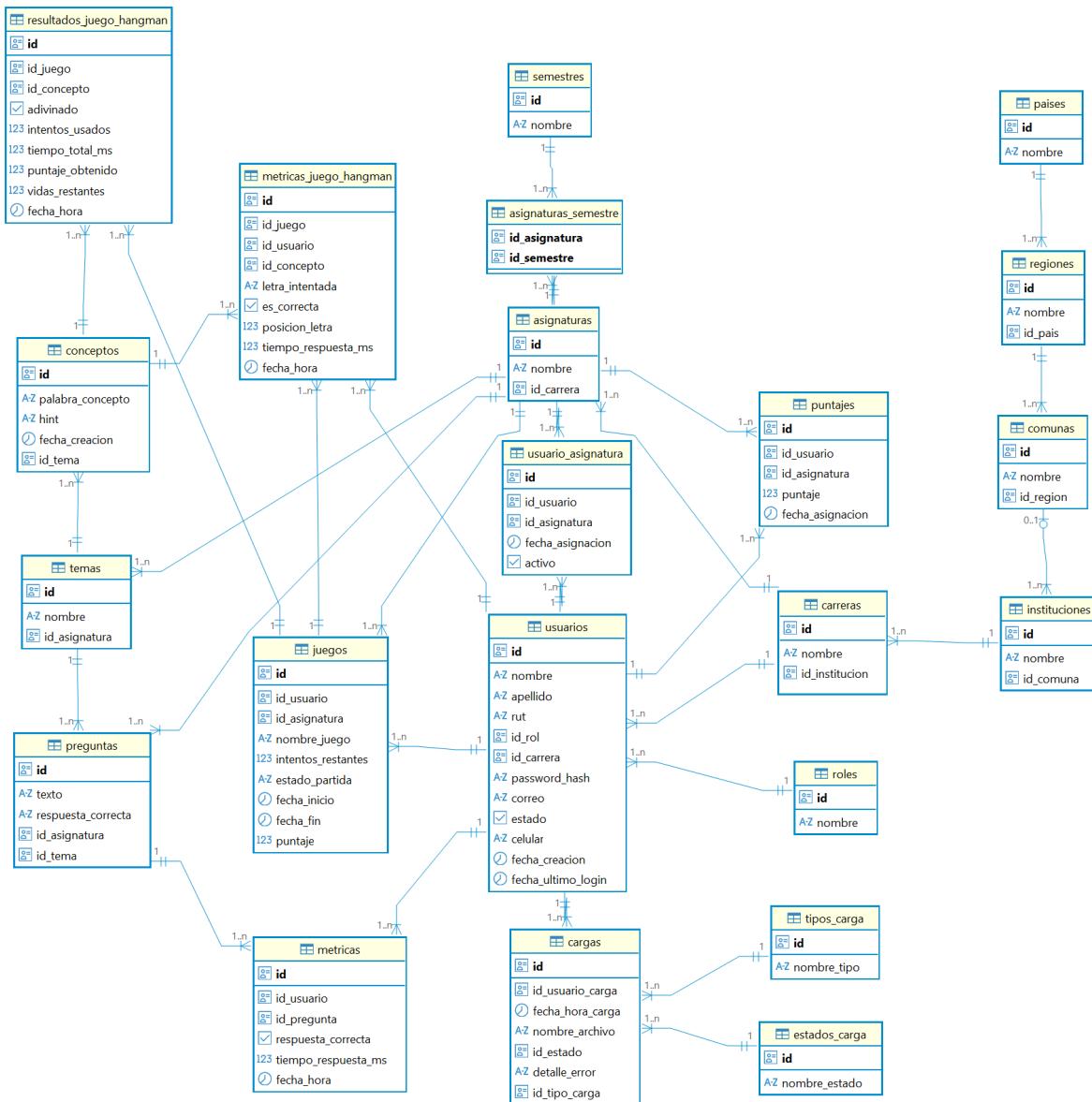
Actúa como un *proxy* seguro y es el motor de la personalización:

- Recibe la solicitud Request: Adaptive Question desde el Content Service.
- Gestiona el recurso mediante Read/Write: Cache Query, consultando primero el Caché para mitigar la latencia y los costos de la API externa.
- Analiza el historial de errores del usuario para solicitar preguntas dinámicas a la API de Google Gemini y reforzar el aprendizaje.

### Scoring Service:

Es el motor de cálculo de la lógica de gamificación. Recibe las interacciones de los jugadores desde el Game Service, calcula los puntajes (puntajes) y el progreso, y es el responsable de calcular el ranking en tiempo real (no persiste una tabla ranking física)

## 17.3 Modelo Entidad-Relación



capstonebbdd-v4.png

### 17.3.1.Diccionario de Datos

#### 17.3.1.1 Tablas de Ubicación e Institución

Estas tablas definen las entidades geográficas y organizacionales necesarias para el contexto educativo.

- **paises:** Almacena los países. Su objetivo es proporcionar una referencia de ubicación global para las instituciones.
  - id: Identificador único del país (Clave Primaria).
  - nombre: Nombre del país (Único).
- **regiones:** Contiene las regiones de cada país. Ayuda a organizar la ubicación de manera jerárquica.
  - id: Identificador único de la región (Clave Primaria).
  - nombre: Nombre de la región (Único).
  - id\_pais: Clave foránea que la relaciona con la tabla paises.
- **comunas:** Almacena las comunas de cada región. Proporciona el nivel más específico de ubicación.
  - id: Identificador único de la comuna (Clave Primaria).
  - nombre: Nombre de la comuna (Único).
  - id\_region: Clave foránea que la relaciona con la tabla regiones.
- **instituciones:** Almacena las instituciones educativas (como Duoc UC).
  - id: Identificador único de la institución (Clave Primaria).
  - nombre: Nombre de la institución (Único).
  - id\_comuna: Clave foránea que la relaciona con la tabla comunas.

#### 17.3.1.2 Tablas de Roles y Contenido Académico

Estas tablas definen la estructura curricular y el contenido central del sistema.

- **roles:** Contiene los roles de usuario (Estudiante, Profesor, Administrador). Es fundamental para la gestión de permisos en la aplicación.
  - id: Identificador único del rol (Clave Primaria).
  - nombre: Nombre del rol (Único).
- **carreras:** Almacena las carreras que ofrece la institución.
  - id: Identificador único de la carrera (Clave Primaria).
  - nombre: Nombre de la carrera.

- **id\_institucion:** Clave foránea que la relaciona con la tabla instituciones.

- **semestres:** Almacena los semestres académicos.
  - **id:** Identificador único del semestre (Clave Primaria).
  - **nombre:** Nombre del semestre (ej. "Semestre 1", "Semestre de Verano").
- **asignaturas:** Contiene las materias académicas. Cada asignatura pertenece a una carrera.
  - **id:** Identificador único de la asignatura (Clave Primaria).
  - **nombre:** Nombre de la asignatura.
  - **id\_carrera:** Clave foránea que la relaciona con la tabla carreras.
- **asignaturas\_semestre:** Tabla intermedia que resuelve la relación N:M entre asignaturas y semestres.
  - **id\_asignatura:** Clave foránea que la relaciona con la tabla asignaturas (Clave Primaria Parcial).
  - **id\_semestre:** Clave foránea que la relaciona con la tabla semestres (Clave Primaria Parcial).
- **temas:** Centraliza la categorización temática del contenido para los juegos.
  - **id:** Identificador único del tema (Clave Primaria).
  - **nombre:** Nombre del tema.
  - **id\_asignatura:** Clave foránea que la relaciona con la tabla asignaturas.
- **preguntas:** El banco de contenido teórico (para Quiz).
  - **id:** Identificador único de la pregunta (Clave Primaria).
  - **texto:** El enunciado de la pregunta.
  - **respuesta\_correcta:** La respuesta correcta.
  - **id\_asignatura:** Clave foránea que la relaciona con la tabla asignaturas.

- id\_tema: Clave foránea que la relaciona con la tabla temas.

- **conceptos:** Almacena las palabras o frases a adivinar (para Hangman).
  - id: Identificador único del concepto (Clave Primaria).
  - palabra\_concepto: Palabra que se debe adivinar.
  - hint: Pista corta (puede ser generada por IA).
  - fecha\_creacion: Momento en que se creó el concepto.
  - id\_tema: Clave foránea que lo relaciona con la tabla temas.

#### 17.3.1.3 Tablas de Usuarios y Relación

Esta sección maneja la identidad de los usuarios y su matrícula en las asignaturas.

- **usuarios:** Almacena la información de todos los usuarios.
  - id: Identificador único del usuario (Clave Primaria).
  - nombre: Nombre del usuario.
  - apellido: Apellido del usuario.
  - rut: Rut del usuario (Único).
  - id\_rol: Clave foránea que la relaciona con la tabla roles.
  - id\_carrera: Clave foránea que la relaciona con la tabla carreras.
  - password\_hash: Contraseña encriptada.
  - correo: Correo electrónico, usado como identificador único.
  - estado: Indica si la cuenta está activa (Booleano).
  - celular: Número de celular.
  - fecha\_creacion: Fecha en que se creó la cuenta.
  - fecha\_ultimo\_login: Fecha del último acceso del usuario.
- **usuario\_asignatura:** Tabla intermedia que registra la matrícula de un usuario (Profesor o Estudiante) en una asignatura específica.
  - id: Identificador único del registro (Clave Primaria).
  - id\_usuario: Clave foránea que la relaciona con la tabla usuarios.
  - id\_asignatura: Clave foránea que la relaciona con la tabla asignaturas.
  - fecha\_asignacion: Fecha en que se registró la asignación.
  - activo: Indica si la asignación está vigente.

#### 17.3.1.4 Tablas de Auditoría y Cargas

Estas tablas rastrean y auditán los procesos de carga masiva de datos (alumnos, contenido).

- **tipos\_carga:** Almacena los tipos de cargas (ej. "Carga de Usuarios", "Carga de Contenido").
  - id: Identificador único del tipo (Clave Primaria).
  - nombre\_tipo: Nombre del tipo de carga (Único).
- **estados\_carga:** Contiene los estados de una carga masiva (ej. "Completado", "Pendiente", "Fallido").
  - id: Identificador único del estado (Clave Primaria).
  - nombre\_estado: Nombre del estado de la carga (Único).
- **cargas:** Registra el historial de todas las cargas masivas.
  - id: Identificador único de la carga (Clave Primaria).
  - id\_usuario\_carga: Clave foránea que indica el usuario que realizó la carga.
  - fecha\_hora\_carga: Fecha y hora de la carga.
  - nombre\_archivo: Nombre del archivo original.
  - id\_estado: Clave foránea que indica el estado del proceso de carga.
  - detalle\_error: Detalles del error en caso de que la carga falle.
  - id\_tipo\_carga: Clave foránea que indica el tipo de carga realizada.

#### 17.3.1.5 Tablas de Gamificación (Genéricas)

Estas tablas registran la actividad de juego que es común a todos los tipos de juego.

- **juegos:** Registra cada sesión de juego. Es el historial de partidas del usuario.
  - id: Identificador único del juego/sesión (Clave Primaria).
  - id\_usuario: Clave foránea que la relaciona con la tabla usuarios.
  - id\_asignatura: Clave foránea que la relaciona con la tabla asignaturas.
  - nombre\_juego: Nombre del juego que se jugó.

- intentos\_restantes: Número de intentos que quedan en la partida.
- estado\_partida: Estado de la partida (En curso, Ganado, Perdido).
- fecha\_inicio: Momento en que inició el juego.
- fecha\_fin: Momento en que terminó el juego.
- puntaje: Puntaje total obtenido en la sesión.

- **metricas:** Registra cada respuesta individual para análisis de rendimiento, idealmente para juegos de Preguntas (Quiz).
  - id: Identificador único de la métrica (Clave Primaria).
  - id\_usuario: Clave foránea que la relaciona con la tabla usuarios.
  - id\_pregunta: Clave foránea que la relaciona con la tabla preguntas.
  - respuesta\_correcta: Valor booleano (TRUE/FALSE) que indica si la respuesta fue correcta.
  - tiempo\_respuesta\_ms: Tiempo que el usuario tardó en responder.
  - fecha\_hora: Momento en que se registró la respuesta.
- **puntajes:** Almacena el puntaje acumulado de un usuario en una asignatura específica.
  - id: Identificador único del puntaje (Clave Primaria).
  - id\_usuario: Clave foránea que la relaciona con la tabla usuarios.
  - id\_asignatura: Clave foránea que la relaciona con la tabla asignaturas.
  - puntaje: Puntaje total acumulado.
  - fecha\_asignacion: Fecha en que se registró o actualizó el puntaje.

#### 17.3.1.6 Tablas de Gamificación

Estas tablas contienen los datos granulares que solo aplican a juegos específicos, permitiendo la escalabilidad.

- **metricas\_juego\_hangman:** Registra cada acción detallada del usuario en el juego Hangman.
  - id: Identificador único de la métrica (Clave Primaria).
  - id\_juego: Clave foránea que la relaciona con la sesión de juego en la tabla juegos.

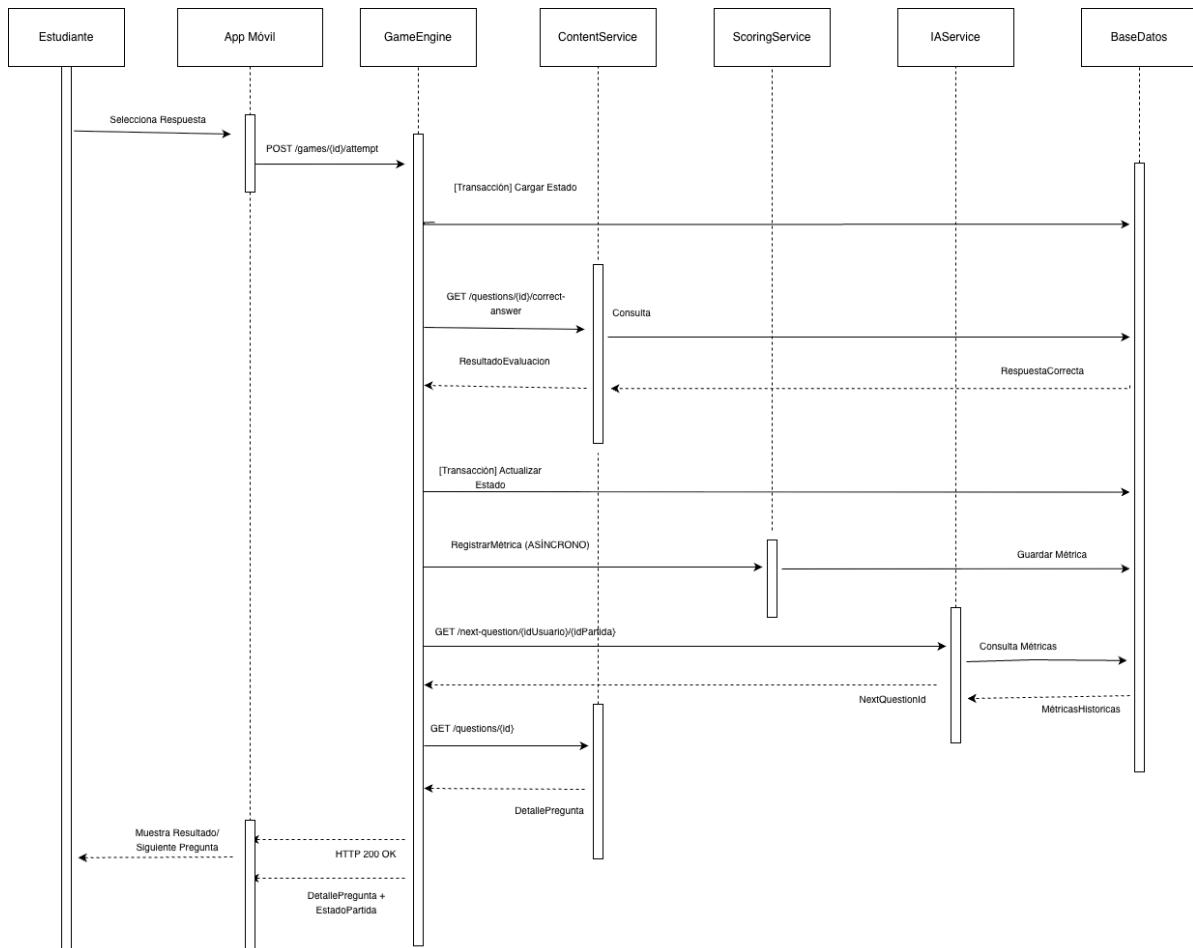
- id\_usuario: Clave foránea que la relaciona con la tabla usuarios.
- id\_concepto: Clave foránea que la relaciona con la palabra que se intentó adivinar.
- letra\_intentada: Letra específica que el usuario ingresó.
- es\_correcta: Indica si el intento fue exitoso (Booleano).
- posicion\_letra: Posición de la letra si fue correcta (NULL si es incorrecta).
- tiempo\_respuesta\_ms: Tiempo que el usuario tardó en ingresar la letra.
- fecha\_hora: Momento del intento.

- **resultados\_juego\_hangman:** Almacena los resultados finales por concepto adivinado dentro de una partida de Hangman.

- id: Identificador único del resultado (Clave Primaria).
- id\_juego: Clave foránea que la relaciona con la sesión de juego en la tabla juegos.
- id\_concepto: Clave foránea que relaciona con el concepto que se intentó resolver.
- adivinado: Indica si el concepto fue adivinado o no (Booleano).
- intentos\_usados: Número de intentos (letras) utilizados en ese concepto.
- tiempo\_total\_ms: Tiempo total dedicado a resolver ese concepto.
- puntaje\_obtenido: Puntaje específico obtenido por resolver el concepto.
- vidas\_restantes: Vidas o intentos restantes al finalizar ese concepto.
- fecha\_hora: Momento de registro del resultado.

## 17.4 Modelado Técnico

### 17.4.1 Diagrama de Secuencia



DiagramadeSecuencia.png

El Diagrama de Secuencia adjunto modela el flujo de trabajo distribuido para la función central del sistema: Procesar un Intento de Respuesta durante una partida adaptativa

### Fases de la Interacción

#### 1. Inicio y Orquestación de la Partida (:Estudiante → :GameEngine)

- Acción del Usuario: La secuencia comienza cuando el Estudiante selecciona una respuesta, enviándola a la App Móvil.
- Petición al Motor: La AppMovil reenvía la acción al microservicio principal, el :GameEngine, a través del endpoint `POST /games/{id}/attempt`.

- Carga de Estado: El Game Engine inicia una (Transacción) Cargar Estado con la Base Datos para recuperar la información actual de la partida (ej. la pregunta actual y los intentos restantes).

## 2. Evaluación de la Respuesta (Validación)

- Consulta de Evaluación: El Game Engine, como orquestador, delega la tarea de validación al :ContentService (Paso GET /questions/{id}/correct-answer).
- Búsqueda en BD: El :ContentService consulta la Base Datos para obtener la Respuesta Correcta.
- Retorno y Actualización: El ContentService devuelve el Resultado Evaluacion al Game Engine. Inmediatamente, el Game Engine realiza una segunda (Transacción) Actualizar Estado en la Base Datos para registrar el resultado y decrementar los intentos restantes, si es necesario.

## 3. Registro Asíncrono de Métricas (Microservicio de Puntuación)

- Delegación Asíncrona: El Game Engine envía un mensaje de Registrar Métrica (Asíncrono) al :Scoring Service. El uso de una flecha simple (sin retorno inmediato) indica que el Game Engine no espera la respuesta, sino que simplemente envía la tarea a una cola de mensajes.
- Persistencia: El Scoring Service (activado por el mensaje asíncrono) se encarga de guardar el registro detallado (Guardar Métrica) en la Base de Datos. Este proceso fuera de línea evita que las tareas de cálculo de puntaje demoren la experiencia de juego del usuario.

## 4. Selección Adaptativa de la Siguiente Pregunta (IA Service)

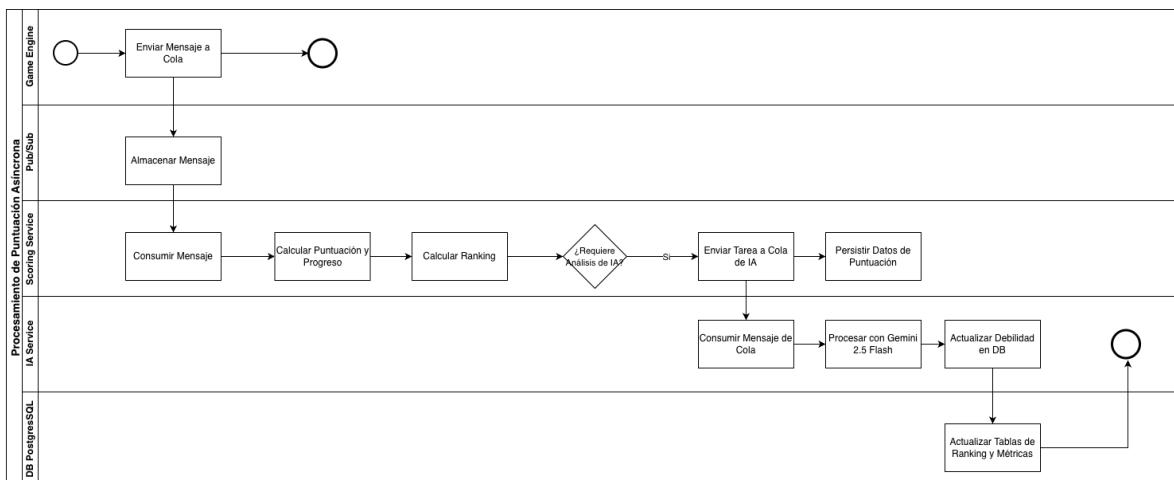
- Lógica Adaptativa: El :Game Engine consulta al :Service (Inteligencia Artificial) (Paso GET /next-question/{idUsuario}/{idPartida}) para determinar la próxima pregunta más adecuada para el estudiante.
- Análisis: El Service consulta la :Base Datos (Consulta Métricas / Métricas Históricas) para analizar el rendimiento pasado y reciente del usuario.
- Selección: El Service devuelve el Next Question Id al Game Engine.
- Obtención de Contenido: El Game Engine utiliza esta ID para solicitar los detalles de la pregunta (texto, opciones) al ContentService.

## 5. Respuesta Final al Estudiante

- Respuesta al Cliente: El Game Engine finaliza su ejecución enviando la respuesta HTTP 200 OK a la AppMóvil. Esta respuesta contiene el resultado del intento actual, la nueva pregunta (Detalle Pregunta), y el Estado Partida actualizado.
- Presentación: La App Móvil recibe la información, desactiva su barra de espera y finaliza la secuencia mostrando el Muestra Resultado / Siguiente Pregunta al Estudiante.

#### 17.4.2 Diagrama de Actividad: Flujo Asíncrono

El Diagrama de Actividad modela el proceso de Actualización de Puntuación Asíncrona.



DiagramaActividad.png

El proceso se define en los siguientes carriles que operan de forma desacoplada:

1. Delegación del Game Engine: El flujo inicia y termina rápidamente en el carril del Game Engine con la acción Enviar Mensaje a Cola. Esto transforma una tarea que podría tomar segundos en una operación de milisegundos, liberando el hilo del usuario para continuar con la sesión de juego.
2. Consumo y Lógica de Puntuación (Scoring Service):
  - El Scoring Service es el primer consumidor de la cola de mensajes. Su responsabilidad principal es el cálculo numérico inmediato (Calcular Puntuación y Progreso y Calcular Ranking).
  - Este servicio toma la decisión crítica a través del Nodo de Decisión (¿Requiere Análisis de IA?). Si la lógica numérica es suficiente, persiste

directamente en la base de datos. Si se necesita un análisis de refuerzo, delega esa tarea al servicio de IA.

### 3. Análisis de Refuerzo (AI Service):

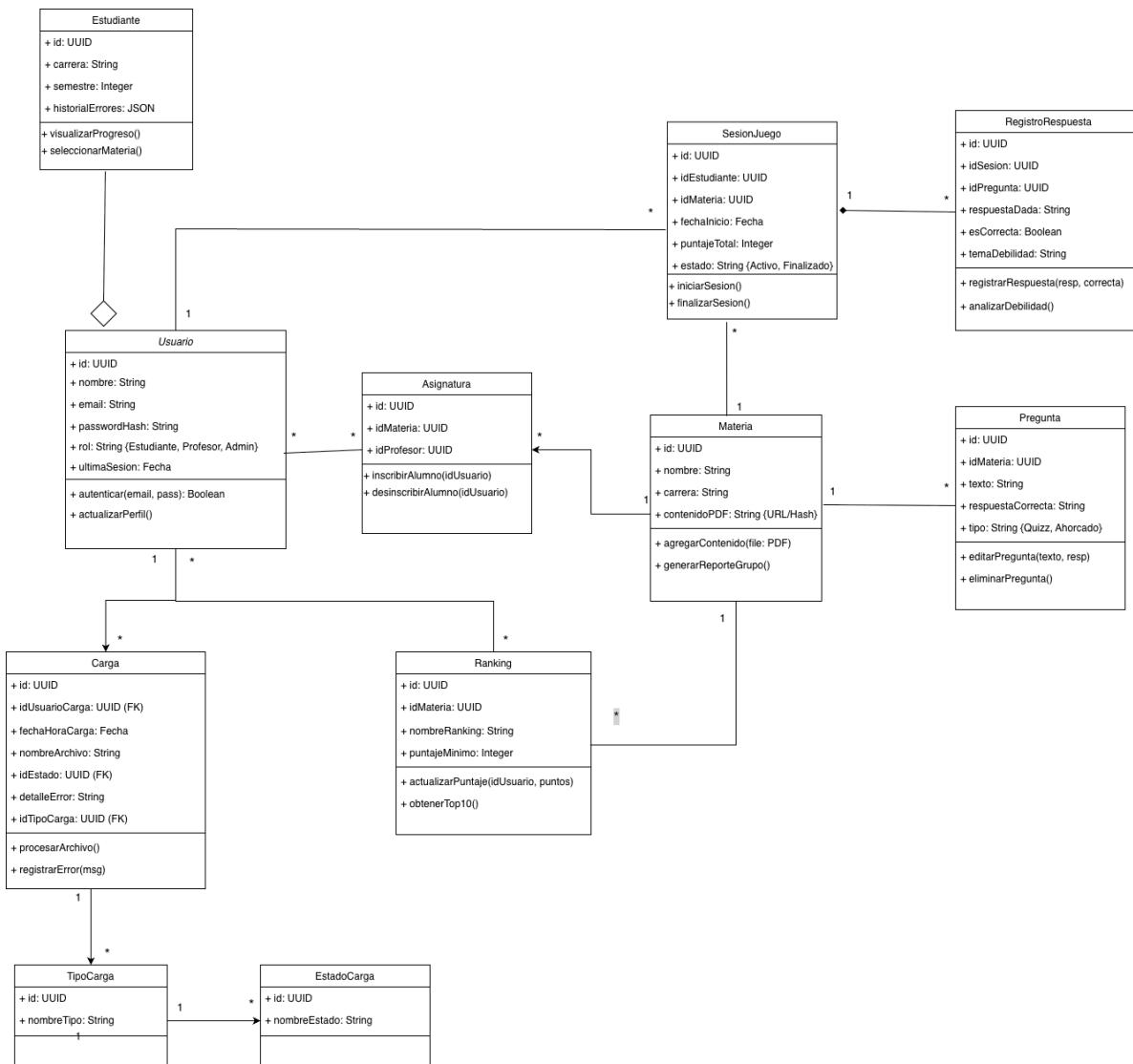
- El AI Service opera como un trabajador que solo se activa si la tarea es enviada por el Scoring Service.
- Su rol es ejecutar la operación más costosa: Procesar con Gemini 2.5 Flash. Este análisis genera el dato de Actualizar Debilidad en DB, que será utilizado por el Game Engine para adaptar la próxima pregunta del estudiante.

### 4. Persistencia (DB Relacional):

- La persistencia de los resultados del cálculo y el ranking ocurre en el carril de DB Relacional (PostgreSQL).
- Tanto el Scoring Service como el AI Service acceden a la base de datos para registrar sus métricas y el historial de debilidades. Esto mantiene la información del ranking y las métricas de forma transaccional y consistente, listas para ser consultadas por el usuario o el profesor.

## 17.4.3 Vistas 4+1

### 17.4.3.1 Vista Lógica: Diagrama de Clases



El modelo se organiza en torno a tres dominios principales:

#### 1. Dominio de Usuarios y Roles:

- La clase Usuario utiliza una relación de generalización/herencia para extender las responsabilidades específicas del Estudiante.
- La clase Estudiante es crítica, ya que contiene el atributo historialErrores: JSON. Este atributo es esencial para el refuerzo adaptativo, ya que el sistema lo consulta para personalizar la próxima pregunta.

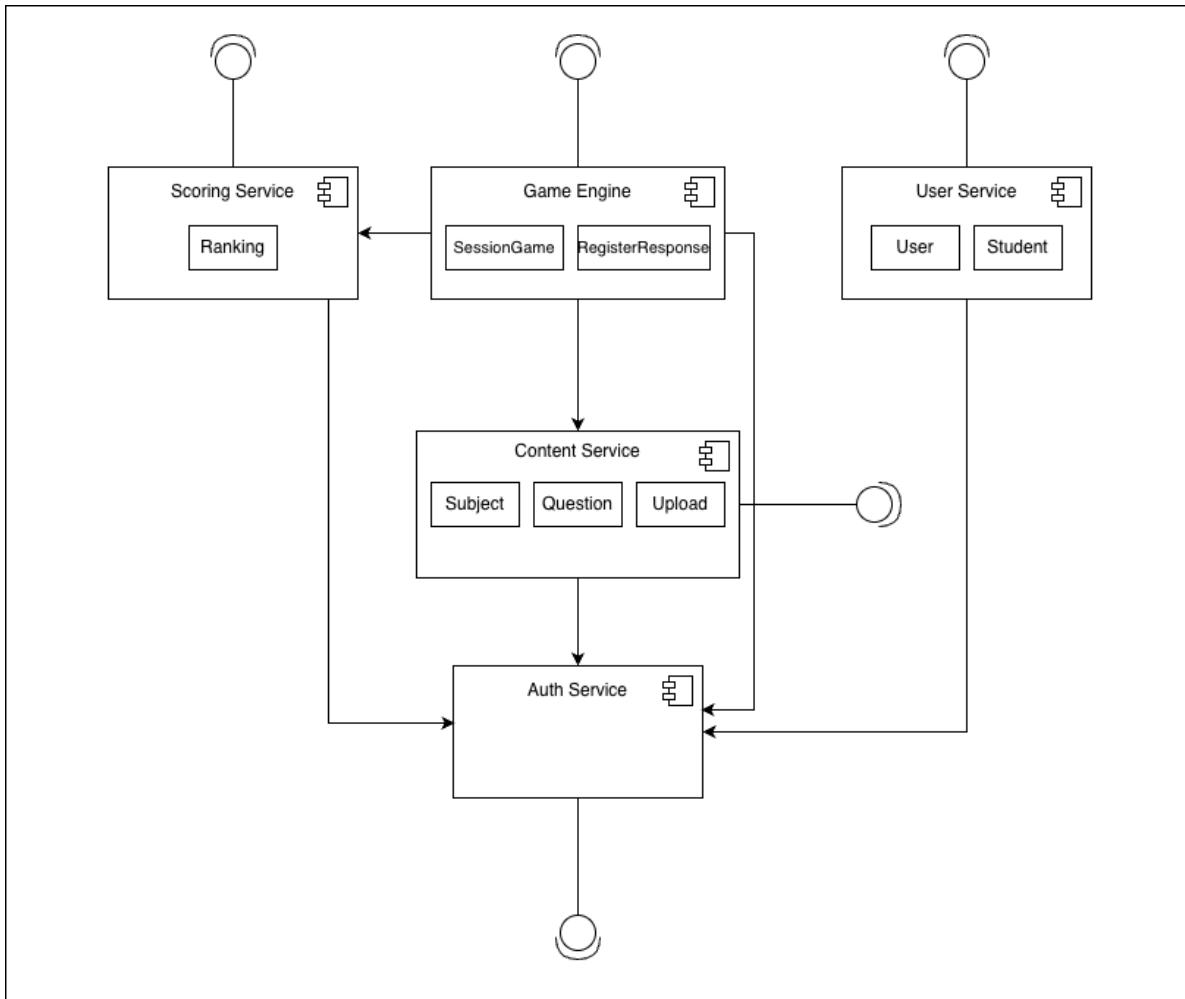
#### 2. Dominio de Contenido y Juego:

- La clase Materia establece una relación con Asignatura, creando el contexto educativo.
- La clase Pregunta contiene atributos clave como tipo: String y respuestaCorrecta: String, que aseguran la variedad de formatos de juego.
- La clase SesionJuego actúa como una entidad transaccional, registrando el estado del juego (estado: String (Activo, Finalizado)) y gestionando el ciclo de vida de la partida.

### 3. Dominio de Persistencia y Analítica:

- La clase RegistroRespuesta almacena la respuesta del usuario (respuestaDada: String) y el resultado (esCorrecta: Boolean), sirviendo como el mensaje de entrada para el procesamiento asíncrono.
- La clase Ranking existe para modelar el puntaje acumulado y las operaciones para obtener el Top 10 (obtenerTop10()), cuyos datos son actualizados por el Scoring Service.
- La clase Carga gestiona el proceso de integración de contenido masivo (procesarArchivo()), vinculándose a las tablas de control de estado (TipoCarga y EstadoCarga).

## 17.4.3.2 Vista de Desarrollo : Diagrama de Componentes



El sistema se divide en los siguientes cinco microservicios atómicos, cada uno con responsabilidades exclusivas:

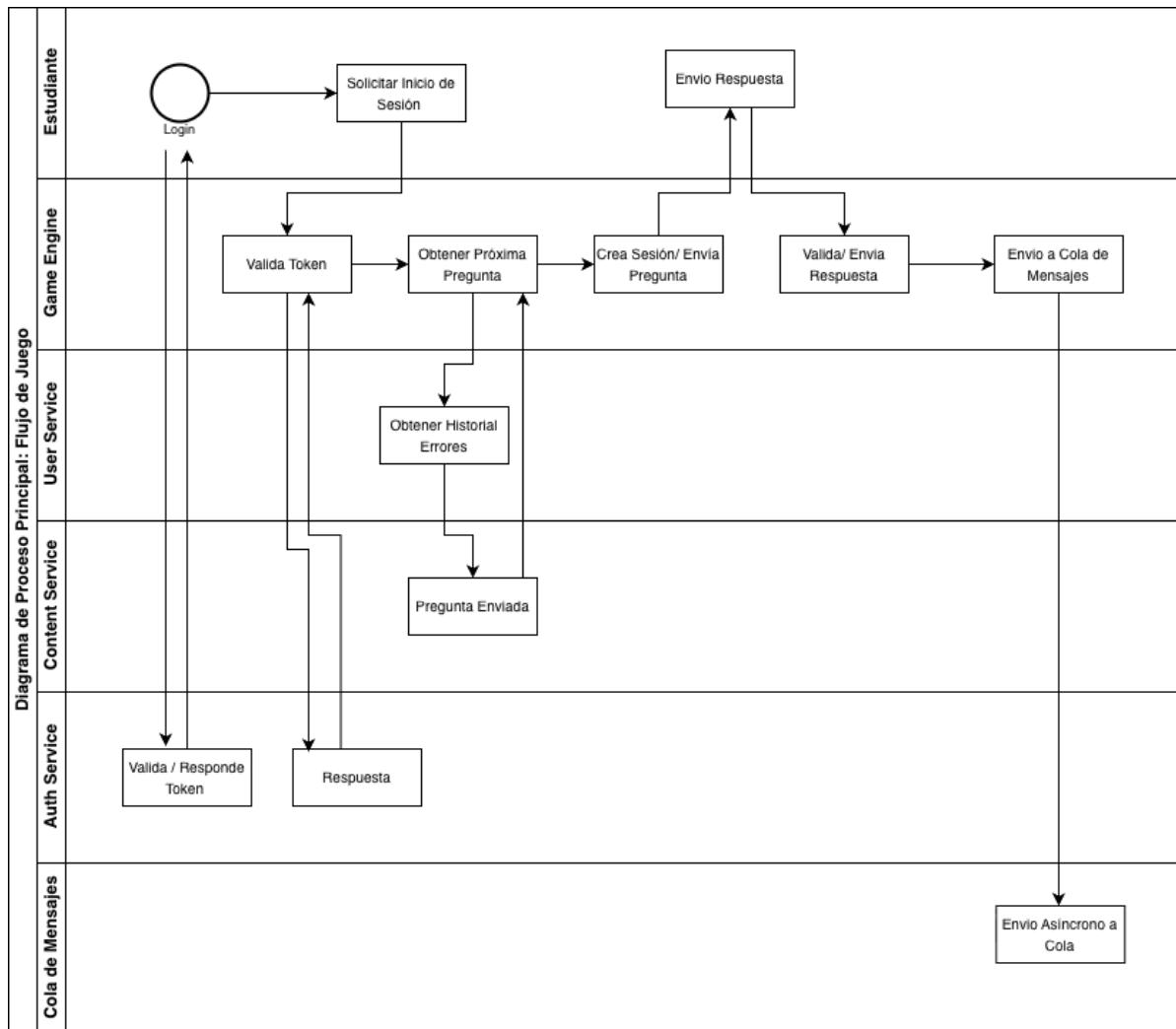
Componente (Microservicio)	Rol Principal	Artefactos Clave (Clases)	Requisitos que Soporta
Game Engine	Motor Central del Juego. Gestiona el ciclo de vida de la sesión de juego, el flujo síncrono y la delegación de	SessionGame, RegisterResponse	RF-G.01 (Flujo de Juego), RNF-01 (Desacoplamiento).

	tareas pesadas.		
Auth Service	Capa de Seguridad. Maneja la autenticación y la autorización (Generación/Validación de JWT).	(Lógica de Hashing y JWT)	RF-E.01 (Seguridad).
User Service	Gestión de Perfil. Almacena y gestiona los datos maestros del usuario y el historial de errores.	User, Student	RF-E.02 (Datos para Adaptatividad).
Content Service	Gestión de Contenido. Provee las preguntas y gestiona el proceso de carga de contenido masivo.	Subject, Question, Upload	RF-P.04 (Carga de Contenido).
Scoring Service	Cálculo de Rendimiento. Responsable del cálculo de puntuaciones, rankings y persistencia de métricas.	Ranking	RF-P.03 (Rankings), RNF-01 (Cálculo Asíncrono).

Las flechas en el diagrama representan las llamadas API síncronas entre los microservicios, mostrando el flujo de la sesión de juego:

1. Dependencia de Autenticación: Casi todos los servicios (Game Engine, Content Service, Scoring Service, User Service) dependen del Auth Service para validar el token JWT en cada petición.
2. Flujo Síncrono de Juego: El Game Engine es el orquestador principal:
  - Depende del User Service para obtener el historial de errores del estudiante.
  - Depende del Content Service para obtener la pregunta específica basada en esa lógica.
3. Flujo Asíncrono: Aunque no se muestra explícitamente la cola en este diagrama, el Game Engine tiene una relación directa con el Scoring Service (a través del artefacto RegisterResponse), donde se delega la tarea de registro de respuesta y cálculo de ranking.

### 17.4.3.3 Vista de Procesos:



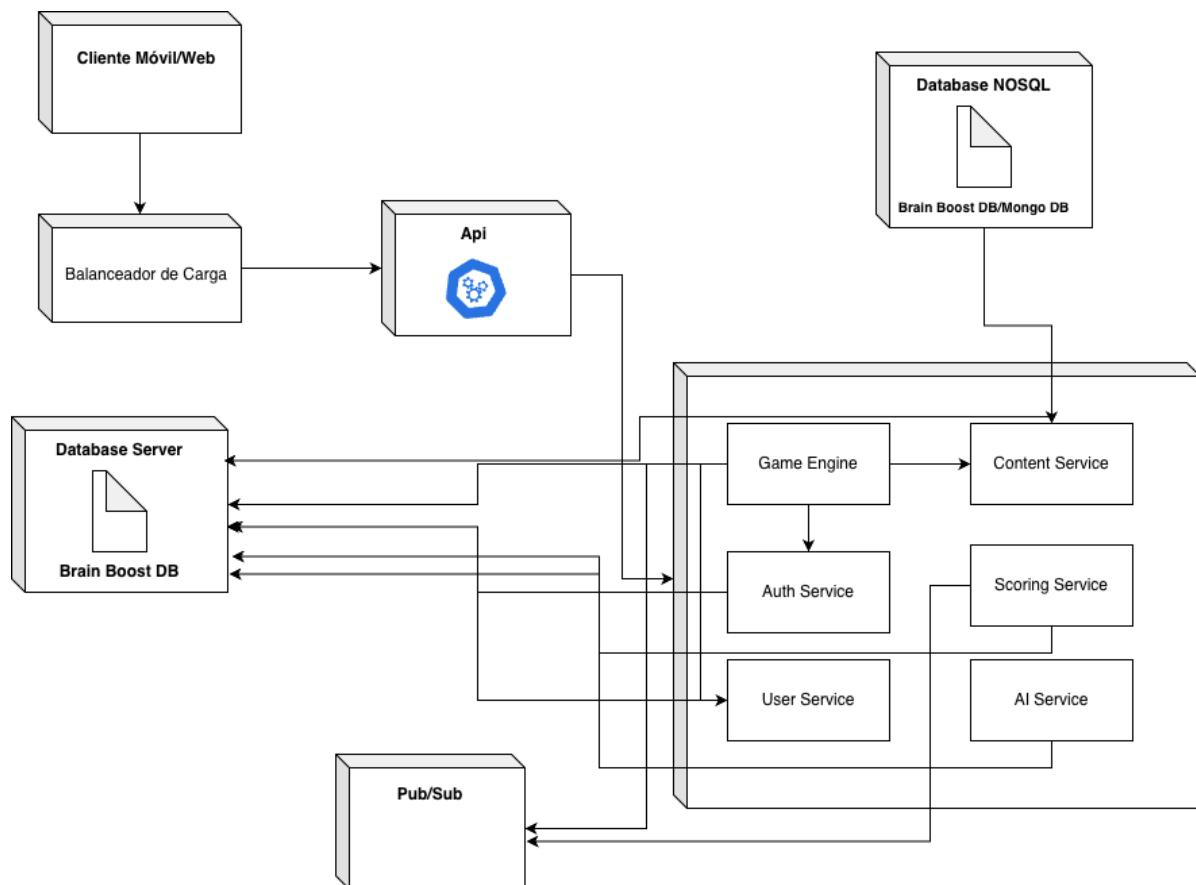
### Proceso Principal: Flujo de Juego Adaptativo (Síncrono)

Este flujo modela la interacción del usuario para obtener y responder una pregunta, el cual debe ser rápido y adaptativo.

1. Actores Involucrados: Estudiante, Game Engine, User Service, Content Service, Auth Service.
2. Inicio: La sesión comienza con el Estudiante solicitando la siguiente pregunta.
3. Validación de Identidad: El Game Engine inicia el proceso validando el Token del usuario contra el Auth Service.

4. Punto de Adaptabilidad: Para asegurar la personalización, el Game Engine consulta al User Service para obtener el Historial de Errores del estudiante antes de solicitar contenido.
5. Obtención de Pregunta: El Game Engine utiliza el historial de errores para solicitar la Próxima Pregunta al Content Service.
6. Final Síncrono: Una vez que el Estudiante envía su respuesta, el Game Engine la valida y, antes de liberar la respuesta de *feedback* al usuario, delega el registro y el cálculo de puntuación a la Cola de Mensajes.

#### 17.4.3.4 Vista Física: Diagrama de Despliegue



El despliegue se organiza en capas claras para gestionar el tráfico y la seguridad:

1. Capa de Acceso y Enrutamiento:
  - Cliente Móvil/Web: El punto de origen del tráfico.

- Balanceador de Carga: Distribuye el tráfico entrante uniformemente hacia el API Gateway, garantizando la Escalabilidad Horizontal (RNF-04) y la tolerancia a fallos.
- API (API Gateway): Actúa como el *entry point* y *proxy* inverso para la aplicación, manejando la validación inicial de *tokens* y el enrutamiento de peticiones a los microservicios internos.

## 2. Clúster de Microservicios:

- Un nodo principal encapsula todos los microservicios del *backend*: Game Engine, Auth Service, User Service, Content Service, Scoring Service y AI Service.
- Esta agrupación en un clúster permite el *autoscaling* y la gestión centralizada del ciclo de vida de los contenedores.

## 3. Capa de Persistencia y Comunicación Asíncrona:

- Database Server (Brain Boost DB): Aloja la Base de Datos Relacional (PostgreSQL), utilizada para los datos transaccionales, perfiles de usuario, contenido de preguntas y los rankings/métricas.
- Database NOSQL (MongoDB): Utilizada específicamente por el Content Service para el almacenamiento o procesamiento de archivos de carga de contenido masivo, aprovechando su flexibilidad de esquema.
- Pub/Sub (Cola de Mensajes): Un nodo desacoplado que facilita la comunicación asíncrona, donde el Game Engine delega las tareas de cálculo de puntuación y análisis de IA, cumpliendo con el requisito de rendimiento.

El diagrama muestra las siguientes relaciones clave:

- Conexión Interna: Las peticiones del API Gateway son enrutadas a los microservicios. Las flechas bidireccionales entre los microservicios y la Database Server confirman la persistencia de datos transaccionales (como sesiones, usuarios y rankings).
- Flujo Asíncrono: La conexión del Game Engine y el API Gateway (para la gestión de mensajes de contenido y notificaciones) hacia el Pub/Sub demuestra el mecanismo de concurrencia adoptado.
- Conexión Específica: La Database NOSQL solo está conectada al Content Service, validando la decisión de usarla únicamente para los archivos de carga masiva, según el requisito inicial.

#### 17.4.3.5 Vista de Escenarios: Diagrama de Casos de Uso



## 1. Actores y Sus Funciones Principales

Administrador: El Administrador es responsable de la gestión de alto nivel del sistema y las cuentas de usuario.

- Gestión de Cuentas: Capacidad para crear, modificar o desactivar cuentas de Profesor y Administrador.
- Gestión de Contenido: Supervisión general del contenido disponible en la plataforma.
- Autenticarse: Iniciar sesión en el sistema.

Profesor: El Profesor es el usuario que inyecta valor al sistema al cargar el material didáctico y gestionar su uso.

- Registrar Usuarios: Capacidad para subir masivamente la matrícula de alumnos a sus secciones.
- Cargar Contenido: Subir material fuente (PDFs, documentos) al sistema. Este caso de uso se extiende al IA Service a través de la relación de asociación para la Generación de Preguntas.
- Gestión de Preguntas: Revisar, editar o aprobar las preguntas y conceptos generados automáticamente por la IA antes de asignarlos a los alumnos.
- Autenticarse: Iniciar sesión en el sistema.

Estudiante: El Estudiante es el consumidor final del sistema, centrado en la interacción y la recepción de *feedback*.

- Jugar Partida: Caso de uso central que orquesta la interacción del alumno con los juegos (Quiz o Hangman).
  - <<include>> Cálculo de Puntaje: Cada acción dentro del juego (acierto o error) activa el Cálculo de Puntaje en el Scoring Service.
  - <<include>> Registrar Partida: Al finalizar o cerrar el juego, el sistema registra todos los datos de la partida, permitiendo el análisis posterior.
- Ver Puntaje Personal: Revisar sus métricas individuales de rendimiento.
- Ver Puntaje Global / Ver Ranking: Acceder a la información comparativa de su rendimiento con el resto del curso o sede.
- Autenticarse: Iniciar sesión en la aplicación móvil.

## 2. Servicios Internos

- IA Service: Responsable de la Generación de Preguntas y conceptos a partir del contenido cargado por el Profesor.
- Content Service: Almacena, organiza y entrega el contenido (preguntas y conceptos) solicitado por el Game Engine para que el Estudiante pueda Jugar Partida.
- Scoring Service: Colabora con el Game Engine para realizar el Cálculo de Puntaje de forma asíncrona, asegurando que la lógica de juego no se ralentice por procesos complejos.

## 18. Decisiones Técnicas y Tecnologías Aplicadas

### 18.1 Tecnologías, Frameworks y Librerías:

Categoría	Tecnología / Librería	Justificación e Impacto
Lenguaje Backend	Kotlin (con Java) y Spring Boot	Elegido por su concisión, seguridad (Kotlin) y el ecosistema robusto de Spring Boot, ideal para la creación de microservicios escalables y modulares.
Bases de Datos	PostgreSQL (Relacional)	Utilizado para datos estructurados y transaccionales donde la integridad y las relaciones son cruciales (Principio ACID).
Bases de Datos	MongoDB (No Relacional)	Utilizado estratégicamente para datos flexibles, de alto

		volumen o logs, permitiendo esquemas dinámicos.
Infraestructura	GCP (Google Cloud Platform) & Docker	Garantiza la Escalabilidad y la Mantenibilidad. Docker empaqueta cada microservicio, asegurando ambientes idénticos entre desarrollo y producción.
Inteligencia Artificial	Google Gemini 2.5 Flash	Elegido por su equilibrio rendimiento/latencia y gestión eficiente de la cuota free-tier. La compatibilidad total con el SDK de Java google-genai (v2.5+) fue crítica para evitar errores y asegurar la integración técnica.
Comunicación	JWT (JSON Web Token)	Estándar de la industria para la autenticación sin estado entre microservicios (Token Stateless), clave para la Seguridad.

## 18.2 Patrones de Diseño Implementados:

Hemos implementado patrones de arquitectura y de desarrollo específicos para garantizar el cumplimiento de los requisitos:

Patrón Implementado	Descripción / Ubicación	Justificación
Patrón Arquitectónico: Microservicios	Todo el sistema está segmentado en servicios atómicos y dedicados (Auth, Game Engine, Scoring, Content, User).	Escalabilidad: Permite escalar servicios individuales sin afectar a otros.
Patrón de Comunicación: Asincronía (Cola de Mensajes)	Uso de Pub/Sub (o similar) entre el Game Engine y el Scoring/AI Service.	Rendimiento (RNF-01): Desacopla la ruta crítica del usuario de las operaciones pesadas, asegurando una respuesta inmediata.
Patrón de Desarrollo: Inyección de Dependencias	Implementado a través del framework Spring Boot en cada microservicio.	Mantenibilidad y Testabilidad: Facilita el desarrollo modular y simplifica la creación de mocks y pruebas unitarias.
Patrón de Desarrollo: Backend For Frontend	El API Gateway actúa como un <i>BFF</i> ligero, orquestando peticiones y añadiendo una capa de seguridad (validación de JWT).	Seguridad y Rendimiento: Controla y optimiza el tráfico, exponiendo solo lo necesario al cliente móvil.

### 18.3 Principales Decisiones Técnicas y su Impacto en la Solución

Decisión Clave	Impacto en la Solución	Trazabilidad Arquitectónica
Inclusión del Game Engine	Modularidad y Cohesión: Centraliza toda la lógica de sesión de juego y el flujo adaptativo, liberando al gateway de responsabilidades de negocio.	Vista de Desarrollo, Flujo Síncrono
Inclusión del Scoring Service	Aislamiento de Cómputo: Aísla el cálculo de ranking y progreso, permitiendo que esta lógica funcione de manera asíncrona y paralela.	Vista de Desarrollo, Flujo Asíncrono
Estrategia de Fallback de Cuotas de IA	Continuidad del Servicio: Implementación de un fallback a contenido estático al detectar error HTTP 429, garantizando la estabilidad bajo la restricción del Free-Tier.	Buenas Prácticas de Resiliencia
Eliminación de Redundancia de Ranking	Consistencia de Datos: El ranking se calcula dinámicamente (Scoring Service), eliminando una tabla materializada y el riesgo de inconsistencia.	Vista Lógica, Decisiones al Modelo de Datos

Buenas Prácticas de Desarrollo Adoptadas:

Práctica de Desarrollo	Detalle de Implementación	Impacto en la Solución
Seguridad	Autenticación con Hashing y JWT	El Auth Service utiliza hashing de contraseñas y genera JSON Web Tokens cifrados para validar la identidad en cada petición, reforzando la seguridad.
Seguridad	Restricción de Acceso a DB	Se eliminó el acceso remoto a PostgreSQL y se restringió el acceso al puerto 5432 a un rango de IP específico, mitigando ataques de red.
Mantenibilidad	Contratos API	El Auth Service define formalmente sus endpoints (DTOs), permitiendo que el desarrollo frontend (móvil) y backend se realicen en paralelo.
Mantenibilidad	Diseño Stateless	Todos los microservicios son diseñados para ser sin estado (excepto las sesiones temporales gestionadas por el Game Engine), lo que facilita su réplica y el escalado horizontal.
Resiliencia	Mecanismo de Fallback de IA	Ante fallos de la API externa (Cuota)

		Excedida), el sistema recurre a contenido local estático, asegurando la operatividad.
--	--	---

## 19. Desarrollo y Soluciones Implementadas

Esta sección documenta la implementación completa de la infraestructura *backend* de Brain Boost. La arquitectura de microservicios diseñada en el Modelo 4+1 ha sido codificada en Kotlin con Spring Boot, y todos los módulos críticos de seguridad, contenido, lógica de juego están operativos, listos para la fase de integración y las pruebas que validan la lógica de personalización del contenido.

### 19.1 Funcionalidades Desarrolladas y Cumplimiento de Requerimientos

El desarrollo abarcó la implementación de los cinco servicios CORE, garantizando el cumplimiento de los requerimientos funcionales (RF) y no funcionales (RNF) definidos:

Microservicio	Funcionalidad Operativa	Requerimiento Validado
Auth Service	Hashing, generación/validación de JWT, Carga masiva de usuarios.	RF-E.01 (Seguridad): Autenticación y gestión de acceso seguros.
Game Engine	Flujo Síncrono Operativo: Gestiona la partida, el estado actual del usuario y	RF-G.01 (Flujo de Juego) y RNF-01 (Rendimiento): Operatividad de la ruta

	el mecanismo de solicitud de contenido personalizado.	crítica de usuario.
Content Service	Endpoint de subida de archivos (teoría) y Módulo de Contenido Estático (banco de preguntas de fallback).	RF-P.04 (Carga de Contenido) y RF-E.02 (Adaptabilidad): Provisión de contenido base y alternativo.
Scoring Service	Lógica de cálculo de ranking y registro del Historial de Respuestas del usuario para su análisis.	RF-A.02 (Scoring): Cálculo de puntaje y registro de la data fuente para la personalización asíncrona.
IA Service	Proxy de conexión y Generación Dinámica de Contenido con Gemini 2.5 Flash, incluyendo Mecanismo de Fallback.	RF-E.02 (Adaptabilidad): Motor de generación de contenido y resiliencia (RNF-03).

La base del sistema es la correcta segregación de responsabilidades, la cual se implementó para optimizar la escalabilidad y el rendimiento.

#### 19.1.1 Auth Service

Este servicio centraliza la seguridad, implementando JWT para el control de acceso. Utiliza PostgreSQL para la persistencia de usuarios, garantizando la integridad transaccional de los datos de cuenta. Su principal implementación, la Carga Masiva de Usuarios, confirma la capacidad de iniciar el sistema con grandes volúmenes de estudiantes.

#### 19.1.2 Game Engine

Es el único servicio que debe responder en tiempo real al usuario. Su rol es orquestar la partida, delegando las tareas pesadas de scoring y análisis al flujo asíncrono. Esta delegación es clave para cumplir con el Bajo Tiempo de Respuesta, ya que su única función síncrona es presentar la pregunta y registrar la respuesta inicial.

### 19.1.3 Content Service

Su arquitectura está diseñada para la Persistencia Híbrida.

- El contenido teórico es almacenado en MongoDB para aprovechar su manejo flexible de documentos no estructurados.
- Además, gestiona el Banco de Contenido Estático (alojado en PostgreSQL), sirviendo como la fuente de datos de contingencia para el *fallback* del IA Service.

### 19.1.4 IA Service

Este servicio encapsula la complejidad de la conexión externa. Actúa como *proxy* para la API de Gemini 2.5 Flash, traduciendo las solicitudes del Game Engine al formato del LLM. Su implementación principal es el Mecanismo de Fallback, que asegura la Tolerancia a Fallos al garantizar que, incluso con errores de cuota en la API externa, el sistema pueda continuar operando con contenido del Content Service.

### 19.1.5 Scoring Service

Este servicio opera fuera de la ruta crítica síncrona. Su función es recibir los eventos de respuesta del Game Engine y realizar el procesamiento intensivo de datos (*scoring*, actualización de *ranking* y, lo más importante, el registro detallado del Historial de Respuestas del estudiante en PostgreSQL). Los datos persistidos en este servicio son la fuente de entrada principal para el futuro módulo de refuerzo adaptativo.

## 19.2 Retos Enfrentados y Soluciones Aplicadas

El desafío de mayor criticidad fue lograr que el sistema de *scoring* y la generación de contenido no afectaran la fluidez del juego, garantizando la latencia mínima

Reto Enfrentado	Impacto Potencial	Solución Aplicada
Latencia del AI Service	Bloqueo del <i>Game Engine</i> y	Implementación de IA

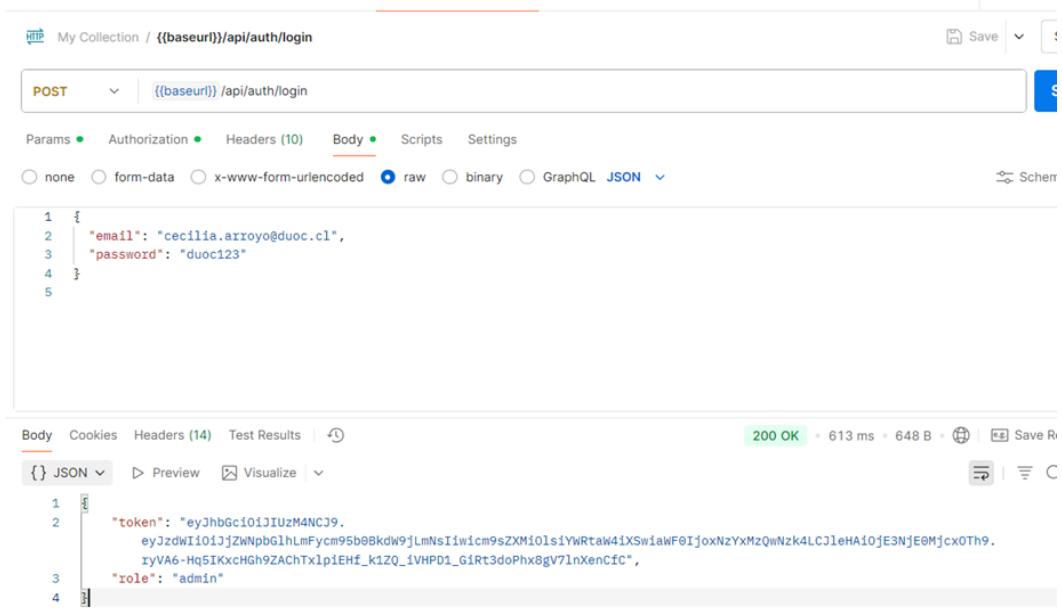
	alto tiempo de respuesta .	Service como Proxy con TTL bajo y un Mecanismo de Fallback al contenido estático, desviando las peticiones fallidas en lugar de esperar una respuesta.
Concurrencia de Scoring	Sobrecarga de la DB Relacional por las operaciones de escritura del Scoring Service.	Diseño de la Vista de Procesos Asíncronos: el Game Engine delega la tarea a un <i>endpoint</i> no bloqueante, permitiendo que el <i>Scoring</i> se ejecute fuera de la ruta crítica de usuario.
Gestión de Carga de Contenido	Rigidez y dificultad para almacenar archivos no estructurados.	Adopción de la Estrategia de Persistencia Híbrida, utilizando MongoDB específicamente para el Content Service, optimizando el almacenamiento y la recuperación de archivos fuente.

## 19.2. Evidencia de Funcionamiento

Las siguientes evidencias demuestran la operatividad de los principales microservicios.

### 19.2.1 Registro y Login Exitoso con JWT:

Muestra la respuesta del *endpoint* de autenticación, confirmando la generación del JSON Web Token (JWT), base de la seguridad en el sistema.



The screenshot shows a Postman collection named "My Collection" with a single POST request to `({baseurl})/api/auth/login`. The request body is a JSON object:

```
1 {  
2   "email": "cecilia.arroyo@duoc.cl",  
3   "password": "duoc123"  
4 }  
5
```

The response status is 200 OK, with a response time of 613 ms and a response size of 648 B. The response body is a JSON object containing a JWT token and a role:

```
{ } JSON ▾ ▶ Preview ▾ Visualize ▾  
1 {  
2   "token": "eyJhbGciOiJIUzI1NiJ9.  
         eyJzdWIiOiJjZWNPbGhLmFycm95b0BkdW9jLmNsIiwicm9sZXMiOlIsIYnRtaW4iXSwiaWF0IjoxNzYxMzQwNzk4LCJleHAiOjE3NjE0MjcxOTh9.  
         ryVA6-Hq5IKxcHGH9ZACHTxlpEHf_k1ZQ_ivHPD1_GiRt3doPhx8gV7lnXenCfc",  
3   "role": "admin"  
4 }
```

### 19.2.2 Creación de Usuario y Persistencia en PostgreSQL

Muestra la solicitud de creación de un nuevo usuario por API, seguida de la verificación directa en la tabla de usuarios de PostgreSQL, confirmando que la información (incluyendo el *hash* de la contraseña) se almacena correctamente.

HTTP My Collection / {{baseurl}}/api/auth/login

POST  Send

Params • Authorization • Headers (11) Body • Scripts Settings Cookies

Auth Type: Bearer Token

Token:  

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Body Cookies Headers (14) Test Results | 

200 OK 613 ms 648 B Save Response    

{ } JSON ▶ Preview Visualize | 

```

1
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJjZWNpbGlhLmFycm95b0BkdW9jLmNsIiwicm9sZXMiolsiYwRtaW4iXSwiaWF0IjoxNzYxMzQwNzk4LCJleHAiOjE3NjE0MjcxOTh9.eyJVA6-Hg5IKxcHgh9ZAChTx1piEHf_k1ZQ_1VHPD1_GiRt3doPhx8gv7lnXenCfc",
3   "role": "admin"
4 }
```

Overview New Environment POST {{baseurl}}/api/auth/login + New Environment

HTTP My Collection / {{baseurl}}/api/auth/login Save Share

POST  Send

Params • Authorization • Headers (11) Body • Scripts Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

Schema Beautify

Body Cookies Headers (14) Test Results | 

201 Created 268 ms 611 B Save Response    

{ } JSON ▶ Preview Visualize | 

```

1 {
2   "name": "Claudio",
3   "lastName": "Valencia",
4   "email": "cvalencia@duoc.cl",
5   "phone": "999999",
6   "password": "duoc123",
7   "role": "profesor",
8   "degreeName": "Ingeniería en Informática"
9 }
```

Body Cookies Headers (14) Test Results | 

201 Created 268 ms 611 B Save Response    

{ } JSON ▶ Preview Visualize | 

```

1 {
2   "id": "bfe52199-c564-48b8-aa1c-198953fdcba5",
3   "name": "Claudio",
4   "lastName": "Valencia",
5   "email": "cvalencia@duoc.cl",
6   "role": {
7     "name": "profesor",
8     "id": "ec9b4bcc-9373-4d2a-879d-0e8f24caeae5d"
9   }
10 }
```

Query    Query History    Scratch Pad

```

1
2  SELECT * FROM usuarios;

```

Data Output    Messages    Notifications

	<b>id</b> [PK] uuid	<b>nombre</b> character varying (100)	<b>apellido</b> character varying (100)	<b>id_rol</b> uuid	<b>id_carrera</b> uuid
1	63a890fe-58ec-44a5-99ba-800af6d27c36	Cecilia	Arroyo	5084555d-2083-42e5-b974-20c3af65c2...	c9bf0ed1
2	b12b868c-fac5-49b3-bacb-54c98b3faf86	Profe	Demo	ec9b4bcc-9373-4d2a-879d-0e8f24cae5d	c9bf0ed1
3	f7cb2690-e48a-4f5f-8b36-07338ff9c697	Cristian	Espinosa	ec9b4bcc-9373-4d2a-879d-0e8f24cae5d	c9bf0ed1
4	bfe52199-c564-48b8-aa1c-198953fdcba5	Claudio	Valencia	ec9b4bcc-9373-4d2a-879d-0e8f24cae5d	c9bf0ed1

```

@Service 2 Usages  Ignacio-leon-m
open class FileUploadService(
    private val authService: AuthService
) {
    open fun processExcelFile(file: MultipartFile): List<RegisterRequestDto> {
        val students = mutableListOf<RegisterRequestDto>()
        file.inputStream.use { input ->
            val workbook = XSSFWorkbook( stream = input)
            val sheet = workbook.getSheetAt( index = 0 ) // first sheet
            for (row in sheet.drop( n = 1 )) { // drop(1) to skip header row
                val name = row.getCell( p0 = 0 )?.stringCellValue ?: ""
                val lastName = row.getCell( p0 = 1 )?.stringCellValue ?: ""
                val email = row.getCell( p0 = 2 )?.stringCellValue ?: ""
                val phone = row.getCell( p0 = 3 )?.stringCellValue ?: ""
                val password = name.lowercase() + "1234" // Default password
                students.add(RegisterRequestDto(name, lastName, email, phone, password))
            }
        }
        return students
    }
}

```

### 19.2.3 Carga Masiva de Usuarios:

Esta captura valida la funcionalidad administrativa crítica, donde el `servicio` procesa una lista de usuarios (desde un archivo xlsx) e inserta múltiples registros en la base de datos de manera eficiente.

POST {{baseUrl}} /api/users/upload

Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> file	File <input type="button" value="Browse..."/> ExcelAlumnos.xlsx		
Key	Text <input type="button" value="Browse..."/> Value	Description	

Response History

#### 19.2.4 Carga de Contenido Teórico y Persistencia en MongoDB

Esta evidencia valida el uso de MongoDB dentro de la estrategia de Persistencia Híbrida. Muestra el *endpoint* del Content Service que recibe un documento teórico por parte del profesor y confirma que dicho archivo es almacenado exitosamente en la base de datos NoSQL.

The screenshot shows the MongoDB Compass interface with the 'ai\_summaries' collection selected. A red box highlights the top document in the list:

```

{
  "_id": "68fb0f0469174fbfd438772",
  "userId": Binary.createFromBase64("pUtsWP6QqGM2fNL2CoC6mQ==", 3),
  "title": "articles-16135_recuso_1.pdf",
  "mimeType": "application/pdf",
  "sizeBytes": 97991,
  "sh1": "6dfa7e32a9a6ffb49a89edec678fff66e7c6bc0",
  "summary": "Este documento presenta estadísticas de prestadores institucionales ac...",
  "createdAt": "2025-10-24T08:58:01.178+00:00",
  "_class": "org.duocuc.capstonebackend.nosql.AiSummaryDoc"
}

```

Below it are two more documents, each with identical data:

```

{
  "_id": "68fb0f0469174fbfd438772",
  "userId": Binary.createFromBase64("pUtsWP6QqGM2fNL2CoC6mQ==", 3),
  "title": "articles-16135_recuso_1.pdf",
  "mimeType": "application/pdf",
  "sizeBytes": 97991,
  "sh1": "6dfa7e32a9a6ffb49a89edec678fff66e7c6bc0",
  "summary": "Este documento presenta estadísticas de prestadores institucionales de s...",
  "createdAt": "2025-10-22T01:17:55.448+00:00",
  "_class": "org.duocuc.capstonebackend.nosql.AiSummaryDoc"
}

{
  "_id": "68fb0f0469174fbfd438772",
  "userId": Binary.createFromBase64("pUtsWP6QqGM2fNL2CoC6mQ==", 3),
  "title": "articles-16135_recuso_1.pdf",
  "mimeType": "application/pdf",
  "sizeBytes": 97991,
  "sh1": "6dfa7e32a9a6ffb49a89edec678fff66e7c6bc0",
  "summary": "Este documento presenta estadísticas de Prestadores Institucionales Ac...",
  "createdAt": "2025-10-24T21:34:46.847+00:00",
  "_class": "org.duocuc.capstonebackend.nosql.AiSummaryDoc"
}

```

## 19.2.5 Generación de Contenido por Consumo de API

Muestra la solicitud al *endpoint* del IA Service, donde se pasa un texto y la respuesta del servicio devuelve las cinco preguntas generadas dinámicamente, listas para ser usadas por el Game Engine

HTTP My Collection / {{baseUrl}}/api/auth/login

POST {{baseUrl}}/api/ai/pdf/quiz?numQuestions=5

Params • Authorization • Headers (11) Body • Scripts Settings Cookies

None form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/>	file	File articles-16135_recurso_1.pdf		
	Key	Text Value	Description	

Body Cookies Headers (14) Test Results

200 OK 4.99 s 1.55 KB Save Response

{ } JSON ▾ Preview Visualize

```

1 {
2   "questions": [
3     {
4       "question": "¿Cuál es el período que abarca el documento de estadísticas de prestadores institucionales acreditados?",
5       "options": [
6         "Del 1 de enero de 2010 al 31 de diciembre de 2016",
7         "Del 24 de diciembre de 2009 al 30 de junio de 2017",
8         "Del 1 de julio de 2010 al 30 de junio de 2017",
9         "Del 24 de diciembre de 2008 al 30 de junio de 2016"
10      ],
11      "answerIndex": 1
12    },
13  ]
14 }
  
```

"questions": "Preguntas Prestadores Institucionales Acreditados realizadas el 20 de junio de 2017"

## 19.2.6 Inicio de Partida

Muestra el *endpoint* que inicia una nueva instancia del juego del Ahorcado, devolviendo el gameId y el estado inicial de la partida.

POST {{baseUrl}}/api/hangman/start

Docs Params Authorization • Headers (10) Body • Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL JSON ▾

```

1 {
2   "userId": "6dd612d3-861a-445b-92b7-f50d1bcb7d9e",
3   "subjectId": "e50f3ae9-4e85-42dc-bf88-736b8324b102",
4   "topicId": "61cfbb4b-371a-4655-8adf-455564c3a54a",
5   "conceptCount": 10
6 }
  
```

## 20. Pruebas y Validación del Sistema

El proceso de pruebas y validación se desarrolló de forma iterativa y paralela al avance de funcionalidades clave, siguiendo la metodología Scrum. Esto permitió la validación temprana de la arquitectura de microservicios y los flujos críticos del sistema. El objetivo principal fue asegurar la estabilidad técnica y la aceptación funcional del producto.

### 20.1 Tipos de Pruebas Realizadas

Se ejecutaron cuatro tipos de pruebas para validar la calidad, el rendimiento y la usabilidad del sistema:

#### 2.1.1 Pruebas Unitarias

Estas pruebas se enfocaron en los componentes de *software* que contienen lógica directa, asegurando que las unidades individuales funcionen correctamente.

- **Validación de Sesión (JWT):** Se verificó que el SessionManager recuperara y almacenara el JSON Web Token (JWT) de forma correcta tras el inicio de sesión, asegurando la integridad del protocolo de autenticación.
- **Integridad de Datos (DTOs):** Se comprobó el *parsing* correcto de los Objetos de Transferencia de Datos (DTOs) en llamadas a la API , confirmando que los datos de entrada y salida fueran mapeados correctamente.
- **Lógica de Juego:** Se validó la correcta transición de estados del Hangman Game State al enviar letras válidas, confirmando que el *core* del juego funciona sin errores lógicos.

#### 2.1.2 Pruebas de Integración

Las pruebas de integración validaron la comunicación entre los microservicios y los componentes de la aplicación.

- **Flujo de Acceso:** Se verificó que, tras un Login exitoso, el token permitiera consultar correctamente la lista de Asignaturas, confirmando la comunicación entre la capa de autenticación y el Content Service.

- Flujo Crítico de IA (PDF → Quiz): Se probó el flujo completo: 1) Selección de PDF por el Profesor, 2) Activación del IA Service para la generación de conceptos, y 3) Recepción y persistencia de las preguntas, resultando en la generación de un JSON con preguntas completas y válidas.
- Flujo de Juego: Se confirmó la correcta orquestación de la partida, desde el ingreso del Alumno al juego hasta la visualización de preguntas filtradas por la asignatura inscrita.

#### 2.1.3 Pruebas de Rendimiento

Las pruebas de rendimiento se centraron en medir la velocidad de respuesta de las operaciones más críticas del sistema, especialmente aquellas que dependen de la red y la IA.

Operación	Tiempo Promedio	Observaciones
Login (JWT)	180 – 250 ms	Buen desempeño, la variación depende del tiempo de validación del token y la latencia de red.
Carga de Asignaturas	250 – 350 ms	Tiempo aceptable para la consulta inicial tras el <i>login</i> .
Generación IA (Quiz / Conceptos)	4 – 8 segundos	El proceso más exigente incluye subida del PDF, su lectura y el procesamiento por la API de Google Gemini.
Carga de Preguntas (Modo Alumno)	130 – 200 ms	La consulta de preguntas desde el repositorio es rápida, lo que garantiza una buena experiencia de juego.

#### 2.1.4 Pruebas de Aceptación (UAT)

Se realizaron pruebas con usuarios piloto (simulando roles de Profesor, Alumno y Administrador) para validar la usabilidad y la satisfacción del sistema:

- Satisfacción General: La mayoría de los flujos críticos (ver lista de asignaturas, crear cuenta, usar Hangman) obtuvieron un puntaje de 5 sobre 5.
- Resultados de la IA: El Profesor validó el flujo de subir PDF y generar conceptos, reportando una alta precisión, aunque se detectaron 2 conceptos repetidos en el *output* debido a la estructura de tablas dentro del PDF fuente.
- Flujo de Juego: El Alumno encontró el flujo del juego Hangman completo y estable (5/5). En la respuesta de *Quiz*, se recomendó agregar sonidos/animaciones (4/5) para enriquecer la retroalimentación inmediata.

#### 2.1.5 Principales Errores Detectados y Correcciones

La fase de pruebas permitió la detección y corrección oportuna de errores clave que mejoraron la estabilidad del sistema:

1. Botón “Generar” Inactivo: Se corrigió un error en la capa de UI (*Compose*) donde el botón de generar contenido no respondía; se ajustó el manejo de estados (*mutableStateOf*) para habilitarlo solo con archivos válidos.
2. Error 400 en Backend: Se detectó un *Missing parameter* (*subjectId*) en una llamada *backend* que se resolvió con la documentación del *endpoint* y el ajuste correspondiente en la solicitud del *frontend*.
3. Lógica de Juego Congelada: El juego Hangman presentaba *bugs* de actualización; se identificó y corrigió un DTO incorrecto (*parsing* y *enums*), lo que estabilizó la transición de estados en el juego.
4. Manejo de Fallas: Se implementaron bloques try/catch a nivel de la aplicación para manejar archivos PDF corruptos, evitando el cierre inesperado (crashing) de la *app*.

## 21. Despliegue y Entorno de Ejecución

### 21.1 Configuración del Entorno (Stack Tecnológico)

El backend de Brain Boost se implementó utilizando un Stack Moderno y Escalable diseñado para soportar la arquitectura de microservicios y la alta concurrencia:

Categoría	Tecnología/Componente	Versión/Uso
Lenguaje Principal	Kotlin y Java	Kotlin 1.9.25 como lenguaje principal y JDK 21 como <i>runtime</i> (con optimizaciones para <i>Virtual Threads</i> ).
Framework	Spring Boot	3.5.5. Utilizado para construir los microservicios REST (GameEngine, ContentService, ScoringService), incluyendo Spring Security para la autenticación JWT.
Base de Datos	Neon PostgreSQL	PostgreSQL 17.5 Serverless (Neon) para datos transaccionales críticos. Se utiliza Flyway 11.7.2 para el versionado y migración automatizada del esquema.
Persistencia	Spring Data JPA	ORM para el mapeo de entidades, con Hibernate para la gestión de datos.

Inteligencia Artificial	Google Gemini SDK	Modelo <code>gemini-2.0-flash</code> para la generación dinámica de preguntas y el procesamiento de documentos (con Apache Tika).
Build y Contenedorización	Gradle y Docker	Gradle 8.14.3 (Kotlin DSL) para la construcción y Docker Multi-Stage para crear imágenes ligeras de ejecución.

## 21.2 Arquitectura y Entorno de Despliegue

El proyecto opera bajo una Arquitectura de Microservicios Desacoplados desplegada en una plataforma *cloud* con capacidades de *auto-scaling*.

El entorno de ejecución sigue el siguiente esquema de despliegue:

- Plataforma Cloud: El *backend* (compilado en un JAR ejecutable de 127MB) se despliega en Render.com, utilizando la infraestructura de Google Cloud Platform (GCP) como un contenedor Docker.
- Estrategia de Persistencia: Los microservicios (Game Engine, Scoring Service, etc.) se conectan a la base de datos Neon PostgreSQL, la cual está configurada con conexión *pooling* (HikariCP) para optimizar el rendimiento de las transacciones concurrentes.
- Despliegue Continuo (CI/CD): El sistema está configurado para Auto-Deploy; cualquier *push* a la rama main del repositorio Git activa automáticamente el proceso de *build* y despliegue del nuevo contenedor en Render.com.
- Diferenciación de Ambientes: Se utilizan Spring Profiles (dev y prod) para diferenciar configuraciones. El ambiente de Producción (prod) inhabilita el *logging* SQL para optimizar rendimiento y utiliza variables de entorno para manejar credenciales secretas (JWT, Gemini API Key).

### 21.2.1 Instrucciones Básicas para Ejecutar el Sistema

Para ejecutar el sistema en un ambiente de desarrollo local, se requieren mínimamente el JDK 21 y Git.

#### A. Requisitos Previos

- Instalar el Java Development Kit (JDK) versión 21 (ej. Eclipse Temurin).
- Clonar el repositorio desde GitHub: `git clone <URL_REPO>`
- Navegar a la carpeta del `backend`: `cd capstone_grupo_3/capstone-backend-code/CapstoneBackend`

#### B. Ejecución en Desarrollo Local (Perfil dev)

El proyecto utiliza el Gradle Wrapper (`./gradlew`) para la ejecución sin necesidad de instalar Gradle globalmente:

Tarea	Comando de Ejecución	Descripción
Compilación	<code>./gradlew build</code>	Compila el código, verifica dependencias y genera el JAR.
Ejecución	<code>./gradlew bootRun</code>	Inicia la aplicación en el puerto 8080. El perfil dev habilita <i>logging</i> detallado y <i>tokens</i> JWT de larga duración (24 horas).
Acceso API	<code>http://localhost:8080/actuator/health</code>	Verifica que la aplicación esté activa (status: UP).

#### C. Testing Automatizado

Para asegurar la calidad y el *runtime* estable de los microservicios:

- Pruebas Unitarias: Ejecutar `./gradlew test` (Incluye cobertura JaCoCo).
- Pruebas Funcionales: Utilizar Postman o cURL para verificar flujos críticos (Login, Carga de Archivos, Inicio de Juego Hangman) contra `http://localhost:8080`.

#### D. Despliegue en Producción (Perfil prod)

El despliegue se gestiona automáticamente mediante Render.com.

1. Asegurar que las Variables de Entorno (ej. GEMINI\_API\_KEY, DATABASE\_URL) estén configuradas como secretos en el Dashboard de Render.
2. Realizar *push* a la rama de producción: git push origin main.
3. Render.com utiliza el Dockerfile para construir la imagen y lanza el servicio con el perfil prod activo, optimizando la memoria y la seguridad.

### Documentación Complementaria

#### Especificación de API

Esta documentación describe los contratos de interfaz de los microservicios, esenciales para la comunicación con el frontend y la interacción interna del backend.

#### Especificación de Endpoints

##### 1. Auth Service: Identidad y Seguridad

Este servicio es la autoridad de identidad. Se encarga de procesar el registro y el inicio de sesión, validando credenciales y generando el JSON Web Token (JWT), que se usa para la autorización del usuario.

Endpoint	Método HTTP	Descripción	Requerimiento de Seguridad
/api/auth/login	POST	Autentica al usuario (Estudiante o Profesor) mediante username y password. Si el acceso es exitoso,	N/A

		retorna un JWT que debe ser incluido en el header Authorization de todas las peticiones posteriores.	
--	--	--	--

## 2. User Service: Gestión del Ciclo de Vida de Cuentas

El User Service gestiona el ciclo de vida de las cuentas, la asignación de roles y la gestión de la carga masiva.

Endpoint	Método HTTP	Descripción	Requerimiento de Seguridad
/api/auth/register	POST	Crea una nueva cuenta de usuario en el sistema. Genera un <i>hash</i> seguro para la contraseña antes de persistir los datos en PostgreSQL.	N/A
/api/users/upload-excel	POST	Permite al Profesor (rol validado por JWT) cargar un archivo (ej., CSV/Excel) con datos de múltiples alumnos. El servicio aplica <i>hashing</i> a las contraseñas temporales y crea	Requiere JWT (Rol: Profesor)

		las cuentas en masa.	
--	--	----------------------	--

### 3. IA Service : Generación de Contenido

Actúa como un *proxy* seguro y es el motor de la personalización y generación de contenido dinámico.

- Arquitectura: Gestiona el recurso mediante Cache Query (consultando primero el Caché para mitigar la latencia y los costos de la API externa) y analiza el historial de errores del usuario para solicitar preguntas dinámicas a la API de Google Gemini.

Endpoint	Método HTTP	Descripción	Requerimiento de Seguridad
/api/ai/pdf/resumen	POST	Recibe un archivo adjunto (PDF/Texto). Utiliza el modelo de IA para procesar el contenido y generar un título y un resumen conciso para el profesor.	Requiere JWT (Rol: Profesor)
/api/ai/pdf/quiz?numQuestions=5	POST	Recibe un archivo adjunto y genera un número específico de preguntas (numQuestions) junto con sus opciones de respuesta, transformando el texto en un cuestionario estructurado.	Requiere JWT (Rol: Profesor)

### 4. Game Engine: Orquestación y Lógica Transaccional

Es el orquestador de la experiencia de juego. Es responsable de la lógica transaccional de la partida, gestionando el estado y asegurando que las métricas se registren de forma asíncrona.

- Arquitectura: Recibe las peticiones de inicio de juego, interactúa con el Content Service para obtener preguntas, y al finalizar la partida, envía el resultado detallado al Scoring Service para el cálculo del puntaje.

Endpoint	Método HTTP	Descripción	Requerimiento de Seguridad
/start	POST	Inicia una nueva instancia de juego, inicializa el estado (intentos_restantes, estado_partida) y devuelve el gameld.	Requiere JWT (Rol: Estudiante)
/games/{gameld}/concepts/submit	POST	Procesa la respuesta o acción del usuario durante la partida. Registra la acción en la tabla metricas y avanza el estado del juego.	Requiere JWT (Rol: Estudiante)
/games/{gameld}/attempt	POST	Procesa y evalúa un intento de respuesta del alumno. Actualiza el estado de la partida, decrementa los intentos restantes y devuelve el estado actual del juego.	Requiere JWT (Rol: Estudiante)

/games/{gameId}/end	POST	Cierra la partida. Registra los resultados finales y delegan el resultado detallado al Scoring Service (Proceso Asíncrono).	Requiere JWT (Rol: Estudiante)
---------------------	------	---	-----------------------------------

## Manual Técnico

### Registro de Versiones

Para asegurar la colaboración eficiente entre los miembros del equipo y el seguimiento de los cambios en la arquitectura de microservicios, se implementó Git como sistema de control de versiones distribuido, alojado en GitHub. El modelo de trabajo se basó en una adaptación simplificada de Gitflow, asegurando la trazabilidad de cada cambio a una tarea específica del proyecto:

1. Rama Principal (main): Contiene la versión estable y lista para ser desplegada. Las *releases* se etiquetan (*tag*) a partir de esta rama.
2. Rama de Desarrollo (develop): Es la rama central de integración. Todos los *features* completados son fusionados aquí una vez que han pasado las pruebas y la revisión de código.
3. Convención de Ramas de Funcionalidad (feature/\*): Cada funcionalidad o historia de usuario fue desarrollada en su propia rama, utilizando la convención *feature/CG3-numero-de-historia*. Estas ramas fueron fusionadas a develop solo después de una *Pull Request* (PR) y la aprobación del código por pares.

### Bitácora de Implementación (Hitos por Sprint)

La siguiente tabla detalla los principales hitos de *merge* que marcan la finalización de la infraestructura *core* del proyecto:

Versión	Fecha de Finalización	Módulos Afectados	Descripción de Cambios / Hito
---------	-----------------------	-------------------	-------------------------------

v1.0.0	14 de Septiembre	Auth Service, User Service	Hito: Base Tecnológica Funcional. Finalización del flujo de autenticación, implementación de Hashing/JWT y conexión exitosa a PostgreSQL.
v1.1.0	29 de Septiembre	IA Service, Content Service	Hito: Integración IA y Persistencia Híbrida. Implementación del <i>proxy</i> a Gemini, <i>endpoint</i> de Carga Masiva (User Service) y configuración de MongoDB.
v1.2.0	13 de Octubre	Game Engine, Content Service	Hito: Contenerización y Staging. Dockerización de los microservicios core. Desarrollo de la estructura de <i>endpoints</i> del Game Engine y refactorización del Content Service.
v1.3.0	10 de Noviembre	Game Engine, Scoring Service	Hito: Lógica de Juego y Asincronía. Implementación de la lógica del juego del Ahorcado, desarrollo del Scoring Service y el mecanismo de delegación asíncrona de resultados.
v1.4.0 (Final)	18 de Noviembre	Todos los Servicios	Release Final. Corrección de <i>bugs</i> menores, validación del mecanismo de <i>fallback</i> y preparación de la infraestructura para la entrega final.

## Aspectos Éticos, Legales y de Seguridad

### Marco Legal y Cumplimiento Normativo

El proyecto Brain Boost se compromete a cumplir con la legislación chilena vigente en materia de protección de datos:

#### Ley 19.628 sobre Protección de Datos Personales

Se implementan medidas para asegurar el cumplimiento estricto de la Ley 19.628, que regula el tratamiento de datos personales en Chile.

- Finalidad Específica: Los datos de los estudiantes (rendimiento, historial de errores, métricas de juego) son recopilados y utilizados exclusivamente con el propósito de optimizar el proceso de aprendizaje y ofrecer refuerzo adaptativo.
- Consentimiento Informado: Se establece la obligación de que la institución o el usuario otorgue el consentimiento explícito para el tratamiento de los datos, informando sobre la finalidad de su uso (mejora académica).
- Principio de Proporcionalidad: Solo se recopilan los datos estrictamente necesarios para la función de personalización adaptativa, evitando la recolección excesiva o irrelevante.

### Mecanismos de Autenticación, Cifrado y Resguardo

La seguridad del sistema está implementada a nivel de código y arquitectura, garantizando la protección de la información sensible:

#### Autenticación y Autorización

- Hashing de Contraseñas: Se utiliza un algoritmo de hashing robusto dentro del Auth Service para almacenar las contraseñas, asegurando que estas nunca se guarden en texto plano.
- JSON Web Token (JWT): El control de acceso a los microservicios se realiza mediante JWT, garantizando que solo los usuarios autenticados y autorizados puedan acceder a los endpoints del Game Engine o el Content Service.

- Seguridad a Nivel de API: Todos los endpoints críticos de la API están protegidos por firewalls y la validación de tokens, restringiendo el acceso solo a las operaciones permitidas para el rol (estudiante, profesor, administrador).

### Cifrado y Resguardo de Información

- Cifrado en Tránsito (TLS/HTTPS): Se utiliza el protocolo TLS/HTTPS para cifrar toda la comunicación entre el frontend y los microservicios, protegiendo las credenciales y los datos de respuesta de la partida contra interceptaciones maliciosas.
- Aislamiento de Microservicios: La arquitectura distribuida aísla las bases de datos. Si un servicio es comprometido, el radio de impacto se limita a su dominio de datos específico, protegiendo los datos críticos de otros servicios.
- Persistencia Segura: Las bases de datos (PostgreSQL y MongoDB) se configuran con acceso restringido solo para los microservicios relevantes, utilizando credenciales y métodos de conexión seguros.

### Uso Responsable de Datos e Inteligencia Artificial

El proyecto aborda los desafíos éticos asociados al manejo de datos de rendimiento y el uso de modelos de IA generativa (Gemini 2.5 Flash).

#### Ética en el Uso de Datos

- No Discriminación y Equidad: El algoritmo de adaptatividad se centra en la debilidad conceptual y no en características demográficas, promoviendo un uso justo de los datos para la mejora académica.
- Auditoría y Transparencia: El registro detallado en el Scoring Service del historial de respuestas permite la trazabilidad de la ruta de aprendizaje de cada estudiante, facilitando auditorías sobre el funcionamiento del motor adaptativo.
- Análisis Anónimo: Cualquier benchmarking o reporte de rendimiento a nivel de institución se realizará utilizando datos pseudonimizados o agregados, eliminando la vinculación directa con el dato personal.

#### Uso Responsable de Inteligencia Artificial

- Principio de Mínima Data: El IA Service actúa como una capa de protección. En las solicitudes a la API de Gemini, nunca se envía información personal identificable

del estudiante. Solo se transmiten los conceptos académicos o la debilidad conceptual.

- Mecanismo de Fallback Ético: El fallback garantiza que la experiencia educativa no dependa al 100% de la IA. Si la IA falla o genera contenido inadecuado, el sistema cambia inmediatamente al banco de contenido estático y validado, manteniendo la calidad educativa.
- Integridad del Contenido: Se establece un proceso de validación de contenido para las preguntas generadas por IA antes de ser presentadas, asegurando la precisión académica y la objetividad, mitigando sesgos inherentes al modelo.

## Conclusiones Técnicas y Aprendizajes

Esta sección resume los logros técnicos más significativos del proyecto, las dificultades clave que se encontraron durante la implementación de la arquitectura de microservicios, y las líneas de mejora futura identificadas.

### Principales Logros Técnicos

El proyecto alcanzó con éxito la implementación de la arquitectura de microservicios de Brain Boost, validando una solución altamente escalable y eficiente..

#### 1. Implementación Exitosa de Persistencia Híbrida

Se logró integrar de manera efectiva dos tipos de bases de datos, resolviendo el requerimiento de gestionar datos transaccionales y documentales de forma óptima:

- PostgreSQL (Relacional): Utilizado por el Auth Service y el Scoring Service para garantizar la integridad transaccional de los datos de usuario y las métricas de rendimiento.
- MongoDB (NoSQL): Utilizado por el Content Service para el almacenamiento flexible y escalable de documentos teóricos, facilitando la gestión de contenido no estructurado.

#### 2. Cumplimiento Riguroso del RNF-01 (Rendimiento)

Mediante el diseño de la Vista de Procesos, se cumplió el requisito de no exceder los 2 segundos de tiempo de respuesta para el usuario:

- El Game Engine fue diseñado para ser un orquestador ligero, delegando el cálculo intensivo de *scoring* y el registro de la data al Proceso Asíncrono a través del Scoring Service. Esto asegura que el *feedback* inmediato al estudiante no se vea afectado por las operaciones de *backend*.

### 3. Integración Segura y Resiliente con IA

Se implementó el IA Service como un *proxy* desacoplado de la lógica de negocio central, logrando dos metas fundamentales:

- Adaptabilidad: Conexión funcional con Gemini 2.5 Flash para la generación dinámica de preguntas de refuerzo.
- Tolerancia a Fallos: Se desarrolló un Mecanismo de Fallback que redirige la solicitud de contenido al Content Service si la API de Google falla o excede la cuota, garantizando la continuidad operativa.

### Dificultades Encontradas y Soluciones Aplicadas

El desarrollo de una arquitectura distribuida presentó desafíos que requirieron decisiones de diseño técnico específicos:

Dificultad Encontrada	Solución Aplicada	Aprendizaje Técnico Clave
Latencia del IA Service	Implementación del Mecanismo de Fallback en el IA Service y uso del patrón Asíncrono en el Game Engine	Desacoplamiento: Las dependencias externas deben ser tratadas como procesos de alta latencia, usando desacoplamiento para no bloquear el hilo principal de la aplicación.
Separación de Datos de Rendimiento	Creación del Scoring Service separado del Game Engine.	Cohesión: El rendimiento es un dominio de negocio distinto del juego , validando la necesidad de la segregación de

		responsabilidades de la Vista Lógica.
--	--	---------------------------------------

## Posibles Mejoras o Líneas Futuras

El proyecto cuenta con una implementación de la infraestructura *core* que permite el desarrollo de las siguientes mejoras en iteraciones futuras:

### 1. Mejoras en el Motor Adaptativo

- Análisis de Debilidad Avanzado: Evolucionar la lógica de refuerzo del Scoring Service de una simple métrica de error a un modelo más sofisticado, como un Modelo de Espaciamiento de la Repetición.
- Generación de Hints Dinámicos: Utilizar el IA Service no solo para generar preguntas, sino también para ofrecer pistas (hints) contextualizadas cuando el estudiante se equivoca en un concepto, mejorando la retroalimentación adaptativa.

### 2. Optimización de Infraestructura y Despliegue

- Implementación de un Cache Distribuido (Redis): Integrar Redis para cachear las respuestas más frecuentes y reducir la carga sobre el Scoring Service y la base de datos, mejorando aún más el rendimiento.
- Observabilidad Completa: Añadir herramientas de monitorización (Datadog y Grafana) para medir las métricas específicas de latencia entre microservicios.

### 3. Desarrollo de la Capa de Visualización (Gamificación)

- Visualización del Ranking Global y por Curso: Desarrollar el endpoint y la interfaz para mostrar el Top 10 global y por asignatura. Aunque el Scoring Service ya calcula y persiste los datos de ranking en PostgreSQL, la presentación de esta métrica es clave para incentivar la gamificación y fue un elemento no alcanzado debido a las restricciones de tiempo.
- Interfaz de Auditoría: Desarrollar una interfaz en el módulo de administración para que el profesorado pueda auditar el resultado de la Carga Masiva de Usuarios, en lugar de depender solo de la respuesta de la API.

## Factibilidad Económica

La evaluación de la factibilidad económica de Brain Boost se realiza a través del análisis de su Flujo de Caja a diez años (2025-2034) y la aplicación de los indicadores financieros Valor Actual Neto (VAN) y Tasa Interna de Retorno (TIR).

### 1. Flujo de Caja

El Flujo de Caja tiene como objetivo proyectar y cuantificar todos los ingresos y egresos de efectivo asociados al proyecto durante su horizonte de vida útil (10 años). Esto permite determinar la capacidad de la inversión para generar liquidez neta en cada periodo.

En Brain Boost, el Flujo de Caja es crucial porque:

- **Viabilidad Financiera:** Permite determinar si el modelo de suscripción anual (Ingresos por Venta) es suficiente para cubrir los altos costos fijos de personal (Ingenieros, PO, Diseñador) y los costos de tecnología especializada (Servicios Cloud, Servicios API de IA).
- **Decisión de Inversión:** Los indicadores clave derivados del FC, como el Valor Actual Neto (VAN) y la Tasa Interna de Retorno (TIR), son las métricas que demuestran la conveniencia o no de llevar a cabo el proyecto, confirmando su rentabilidad en un horizonte de 10 años.
- **Recuperación de Capital:** Muestra en qué momento el proyecto recupera la inversión inicial y el Capital de Trabajo (KT), lo cual ocurre a partir del Año 3, cuando el Flujo de Caja se vuelve positivo.

### Cálculo del Flujo de Caja

Los cálculos se realizan para un período de 10 años (Año 0 a Año 10), utilizando una Tasa de Descuento (Retorno) del 10% como costo de oportunidad o tasa mínima exigida.

Flujo de Caja	0	1	2	3	4	5	6	7	8	9	10
Ingresos por venta		\$49.200.000	\$105.780.000	\$113.713.500	\$122.242.013	\$131.410.163	\$141.265.926	\$151.860.870	\$163.250.435	\$175.494.218	\$188.656.284
Venta de Activo										\$1.732.885	
Costos		-\$81.634.400	-\$84.083.432	-\$86.605.935	-\$89.204.113	-\$91.880.236	-\$95.555.446	-\$99.377.664	-\$103.352.770	-\$107.486.881	-\$111.786.356
Depreciación (-)		-\$386.365	-\$386.365	-\$386.365	-\$386.365	-\$386.365	-\$386.365	-\$386.365	-\$386.365	-\$386.365	-\$386.365
Valor Libro Activo											
Intereses Ptmo.		-\$13.781.482	-\$12.946.195	-\$11.943.850	-\$10.741.035	-\$9.297.658	-\$7.565.606	-\$5.487.143	-\$2.992.987	\$0	\$0
Utilidad Bruta		-\$46.602.247	\$8.364.009	\$14.777.351	\$21.910.500	\$29.845.904	\$37.634.174	\$46.609.699	\$58.251.199	\$67.620.972	\$76.483.564
Impuestos		\$0	-\$2.258.282	-\$3.989.885	-\$5.915.835	-\$8.058.394	-\$10.161.227	-\$12.584.619	-\$15.727.824	-\$18.257.663	-\$20.650.562
Utilidad Neta											
Depreciación (+)											
Valor Libro Activo											
Inversiones		-\$3.599.892									
KT Total		-\$68.028.667	-\$70.069.527	-\$72.171.612	-\$74.336.761	-\$76.566.864	-\$79.629.538	-\$82.814.720	-\$86.127.309	-\$89.572.401	-\$93.155.297
Inv. KT		-\$68.028.667	-\$2.040.860	-\$2.102.086	-\$2.165.148	-\$2.230.103	-\$3.062.675	-\$3.185.182	-\$3.312.589	-\$3.445.092	-\$3.582.896
Valor Deshecho											\$124.336
Prestamo		\$68.907.412									
Amortización Ptmo.		-\$4.176.438	-\$5.011.726	-\$6.014.071	-\$7.216.886	-\$8.660.263	-\$10.392.315	-\$12.470.778	-\$14.964.934	\$0	\$0
Flujo de Caja		-\$2.721.147	-\$52.433.181	-\$621.721	\$2.994.611	\$6.934.041	\$10.450.937	\$14.406.150	\$18.628.078	\$24.499.713	\$46.166.778
Tasa descuento (retorno)		10%									
VAN		\$68.911.554									
TIR		24%									

La tabla del Flujo de Caja muestra todos los componentes del cálculo para el período estipulado:

### Ingresos por Venta

Los ingresos crecen de \$49.200.000 en el Año 1 hasta \$188.656.284 en el Año 10, impulsados por un aumento anual de clientes del 7,5%.

### Costos Operacionales

Parten de \$81.634.400 en el Año 1, e incluyen salarios, servicios cloud y API de IA, creciendo al 3% anual.

### Amortización de Deuda e Intereses

- Préstamo: Se obtiene un financiamiento de \$68.907.412 a una tasa de 20,00% y un plazo de 8 años.
- Intereses Ptmo. (Gasto): Es el costo financiero que disminuye la utilidad bruta (y actúa como escudo fiscal).
- Amortización Ptmo. (Egreso): Es la devolución del principal, un egreso de caja que se realiza hasta el Año 8.

### Inversiones y Capital de Trabajo (KT)

- Inversiones (Año 0): Se registra una inversión inicial de \$3.599.892.
- KT Total: Es el capital de trabajo necesario para la operación, calculado y financiado en el Año 0.

## 2. Valor Actual Neto

El Valor Actual Neto (VAN) tiene como objetivo determinar si el proyecto es rentable hoy, descontando todos los flujos futuros de efectivo a valor presente, utilizando la Tasa de Descuento (10%).

- Criterio de Aceptación: Si el VAN es mayor o igual 0, el proyecto es aceptado ya que recupera la inversión inicial y genera una ganancia adicional superior a la rentabilidad exigida (10%).

### Cálculo del VAN

El cálculo del VAN, utilizando los flujos de caja y la tasa de descuento del 10%, es el siguiente:

Indicador	Valor	Criterio de Decisión
VAN	\$68.911.554	VAN > 0. El proyecto genera un valor significativo por encima del rendimiento exigido. Es rentable.

Un VAN positivo de \$68.9 millones CLP demuestra que, incluso con un modelo de ingresos que contempla 5 meses de desarrollo sin ventas en el primer año, el proyecto Brain Boost crea un valor económico significativo para los inversionistas.

## 3. Tasa Interna de Retorno

La Tasa Interna de Retorno (TIR) es la tasa de descuento que hace que el VAN sea exactamente cero. Su objetivo es medir la rentabilidad intrínseca del proyecto en términos porcentuales anuales.

- Criterio de Aceptación: Si la TIR > Tasa de Descuento (10%), el proyecto es aceptado.

## Cálculo e Interpretación del TIR

El cálculo del TIR para el horizonte de 10 años es:

Indicador	Valor	Criterio de Decisión
Tasa Interna de Retorno (TIR)	24%	TIR (24%) > Tasa de Descuento (10%). La rentabilidad es alta y superior al costo de capital.

La Tasa Interna de Retorno del 24% es más del doble de la Tasa de Descuento exigida del 10%. Esto significa que el proyecto no solo supera las expectativas de rentabilidad mínima, sino que ofrece una excelente capacidad para generar ganancias a lo largo de su vida útil. El proyecto Brain Boost es, por lo tanto, financieramente muy atractivo y robusto.

## Bibliografía y Referencias

## Anexos

