



# **INSTITUTO POLITÉCNICO NACIONAL**

**CENTRO DE ESTUDIOS CIENTÍFICOS Y TECNOLÓGICOS No. 3**  
**“ESTANISLAO RAMÍREZ RUÍZ”**

## **DESARROLLO DE UNA PLATAFORMA WEB PARA LA CREACIÓN Y GESTIÓN AUTOMATIZADA DE EXÁMENES EDUCATIVOS CON INTELIGENCIA ARTIFICIAL**

### **TESIS**

**QUE PARA OBTENER EL TÍTULO DE:**  
**TÉCNICO EN COMPUTACIÓN**

### **PRESENTAN:**

- **CASTRO AGUILAR EDER JOEL**
- **HERNÁNDEZ TELLEZ HÉCTOR FIDEL**
- **VALENCIA OROPEZA ANGEL YAHIR**

### **ASESORES:**

**ING. JOSÉ ERWIN RODRÍGUEZ PACHECO**

**ECATEPEC DE MORELOS, 17 DE JUNIO DE 2025**

# Índice general

|   |           |
|---|-----------|
| <b>INTRODUCCIÓN</b>   | <b>5</b>  |
| <b>1. MARCO TEÓRICO</b>   | <b>7</b>  |
| 1.1. Evaluación educativa digital . . . . .   | 7         |
| 1.1.1. Ventajas de la evaluación digital . . . . .  | 7         |
| 1.1.2. Herramientas de evaluación digital: panorama actual . . . . .                          | 8         |
| 1.2. Desarrollo web de la plataforma . . . . .  | 8         |
| 1.2.1. Herramientas para el desarrollo web de la plataforma . . . . .                         | 10        |
| 1.2.2. Bases de datos para el back-end . . . . .  | 11        |
| 1.3. Inteligencia Artificial en la evaluación educativa . . . . .                             | 12        |
| 1.3.1. Panorama Actual de Modelos de IA Generativa para la Creación de<br>Contenido . . . . . | 12        |
| 1.3.2. Análisis Comparativo y Justificación de la Selección de Gemini . . .                   | 13        |
| 1.3.3. Implementación Específica de Gemini en la Plataforma . . . . .                         | 14        |
| 1.3.4. Contextualización: Aplicaciones de IA en Plataformas de Diseño<br>Existentes . . . . . | 14        |
| 1.4. Tecnologías para el Desarrollo Web de la Plataforma . . . . .                            | 14        |
| 1.4.1. Desarrollo del Frontend: Interfaz de Usuario Dinámica con React .                      | 15        |
| 1.4.2. Backend y Base de Datos: Potenciando la Plataforma con Supabase                        | 16        |
| 1.4.3. Integración de Inteligencia Artificial con la API de Gemini . . . . .                  | 16        |
| <b>2. ANÁLISIS DE REQUERIMIENTOS Y ARQUITECTURA DEL SIS-<br/>TEMA</b>                         | <b>18</b> |
| 2.1. Levantamiento de requerimientos . . . . .  | 18        |
| 2.2. Requerimientos del sistema . . . . .   | 19        |
| 2.2.1. Requerimientos funcionales . . . . .   | 19        |
| 2.2.2. Requerimientos no funcionales . . . . .  | 19        |
| 2.3. Modelo de interacción y experiencia de usuario . . . . .                                 | 20        |
| 2.3.1. Principios de diseño aplicados . . . . .   | 20        |
| 2.3.2. Flujo de interacción del usuario . . . . .   | 21        |
| 2.4. Arquitectura del sistema . . . . .   | 21        |



|           |   |           |
|-----------|---|-----------|
| 2.4.1.    | Componentes Principales . . . . .                                       | 21        |
| 2.4.2.    | Flujo de Datos e Interacciones . . . . .                                | 22        |
| 2.5.      | Distribución y gestión recursos del sistema . . . . .                   | 23        |
| 2.5.1.    | Recursos del lado del cliente . . . . .                                 | 23        |
| 2.5.2.    | Recursos del lado del servidor . . . . .                                | 23        |
| 2.5.3.    | Balance de carga y eficiencia . . . . .                                 | 23        |
| <b>3.</b> | <b>DISEÑO E IMPLEMENTACIÓN</b>  | <b>24</b> |
| 3.1.      | Desarrollo de la interfaz de usuario . . . . .                          | 24        |
| 3.1.1.    | Componentes principales de la interfaz . . . . .                        | 24        |
| 3.1.2.    | Sistema de tipos de preguntas y personalización . . . . .               | 25        |
| 3.2.      | Integración de inteligencia artificial . . . . .                        | 26        |
| 3.2.1.    | Implementación Técnica de la Integración de IA con Gemini API . . . . . | 27        |
| 3.3.      | Sistema de gestión de sesiones de examen . . . . .                      | 29        |
| 3.4.      | Gestión de base de datos y almacenamiento . . . . .                     | 29        |
| 3.4.1.    | Modelo de Datos en Supabase PostgreSQL . . . . .                        | 29        |
| 3.4.2.    | Políticas de Seguridad Row Level Security (RLS) . . . . .               | 30        |
| 3.4.3.    | Operaciones CRUD y APIs . . . . .                                       | 30        |
| 3.5.      | Seguridad y protección de datos . . . . .                               | 30        |
| <b>4.</b> | <b>RESULTADOS Y PRUEBAS</b>   | <b>32</b> |
| 4.1.      | Interfaz de usuario implementada . . . . .                              | 32        |
| 4.1.1.    | Panel principal y navegación . . . . .                                  | 32        |
| 4.1.2.    | Configuración de exámenes . . . . .                                     | 32        |
| 4.1.3.    | Interfaz de examen . . . . .  | 33        |
| 4.1.4.    | Resultados y estadísticas . . . . .                                     | 34        |
| 4.2.      | Funcionalidades de inteligencia artificial . . . . .                    | 34        |
| 4.2.1.    | Generación automática de preguntas . . . . .                            | 34        |
| 4.2.2.    | Análisis de respuestas . . . . .  | 35        |
| 4.3.      | Pruebas de rendimiento . . . . .  | 36        |
| 4.3.1.    | Pruebas de carga concurrente . . . . .                                  | 36        |
| 4.3.2.    | Pruebas de compatibilidad . . . . .                                     | 37        |
| 4.4.      | Validación de seguridad . . . . .                                       | 37        |
| 4.4.1.    | Autenticación y autorización . . . . .                                  | 37        |
| 4.4.2.    | Integridad de exámenes . . . . .  | 38        |
| 4.5.      | Feedback de usuarios . . . . .  | 38        |
| 4.5.1.    | Pruebas de usabilidad . . . . .   | 39        |
| 4.5.2.    | Efectividad educativa . . . . .   | 39        |



## 5. PRODUCCIÓN Y DESPLIEGUE

41

|   |    |
|---|----|
| 5.1. Arquitectura de despliegue . . . . .                   | 41 |
| 5.1.1. Infraestructura utilizada . . . . .                  | 41 |
| 5.1.2. Estrategia de despliegue híbrida . . . . .           | 41 |
| 5.2. Proceso de despliegue . . . . .                        | 42 |
| 5.2.1. Configuración del frontend . . . . .                 | 42 |
| 5.2.2. Configuración del backend . . . . .                  | 43 |
| 5.2.3. Configuración de base de datos . . . . .             | 44 |
| 5.3. Dominio y DNS . . . . .                                | 44 |
| 5.3.1. Configuración de dominios de la aplicación . . . . . | 45 |
| 5.3.2. Configuración DNS . . . . .                          | 45 |
| 5.4. Monitoreo y mantenimiento . . . . .                    | 45 |
| 5.4.1. Métricas de rendimiento . . . . .                    | 45 |
| 5.4.2. Procedimientos de mantenimiento . . . . .            | 46 |
| 5.5. Resultados del despliegue . . . . .                    | 46 |
| 5.5.1. Métricas de rendimiento en producción . . . . .      | 46 |
| 5.5.2. Feedback post-despliegue . . . . .                   | 46 |

## 6. ARQUITECTURA TÉCNICA DETALLADA

47

|   |    |
|---|----|
| 6.1. Análisis del Frontend - Arquitectura React . . . . .           | 47 |
| 6.1.1. Estructura de Componentes y Funcionalidades . . . . .        | 47 |
| 6.1.2. Integración con Modelos de Inteligencia Artificial . . . . . | 48 |
| 6.1.3. Gestión de Estado Global y Autenticación . . . . .           | 49 |
| 6.2. Análisis del Backend - APIs y Procesamiento . . . . .          | 50 |
| 6.2.1. Estructura del Servidor y Middleware . . . . .               | 51 |
| 6.2.2. APIs Principales y Procesamiento de IA . . . . .             | 51 |
| 6.2.3. Gestión de Base de Datos y Persistencia . . . . .            | 54 |
| 6.3. Flujo de Datos: Del Usuario a la Base de Datos . . . . .       | 56 |
| 6.3.1. Captura y Procesamiento de Respuestas . . . . .              | 56 |
| 6.3.2. Estructura de Almacenamiento en Base de Datos . . . . .      | 57 |
| 6.3.3. Procesamiento y Cálculo de Métricas . . . . .                | 58 |
| 6.4. Seguridad y Middleware de Autenticación . . . . .              | 59 |
| 6.4.1. Implementación del Middleware de Seguridad . . . . .         | 59 |
| 6.4.2. Validaciones de Seguridad Implementadas . . . . .            | 59 |

## 7. CONCLUSIONES

60

|  |    |
|--|----|
| 7.1. Logros obtenidos . . . . .                | 60 |
| 7.1.1. Objetivos técnicos alcanzados . . . . . | 60 |
| 7.1.2. Impacto educativo . . . . .             | 60 |
| 7.2. Innovaciones implementadas . . . . .      | 61 |



|        |  |    |
|--------|--|----|
| 7.2.1. | Integración inteligente de IA . . . . .  | 61 |
| 7.2.2. | Arquitectura híbrida eficiente . . . . . | 61 |
| 7.3.   | Limitaciones identificadas . . . . .     | 61 |
| 7.3.1. | Limitaciones técnicas . . . . .          | 61 |
| 7.3.2. | Limitaciones pedagógicas . . . . .       | 62 |
| 7.4.   | Trabajo futuro . . . . .                 | 62 |
| 7.4.1. | Mejoras técnicas planificadas . . . . .  | 62 |
| 7.4.2. | Expansión de funcionalidades . . . . .   | 62 |
| 7.4.3. | Expansión educativa . . . . .            | 63 |
| 7.5.   | Reflexiones finales . . . . .            | 63 |

# INTRODUCCIÓN

La evaluación educativa es un componente fundamental del proceso de enseñanza-aprendizaje que permite medir el progreso académico y facilitar la retroalimentación tanto para estudiantes como para educadores. En la actualidad, la creación y gestión de exámenes representa un desafío significativo para docentes e instituciones educativas, especialmente en entornos que requieren evaluaciones frecuentes y personalizadas.

Las herramientas tradicionales de evaluación presentan limitaciones importantes: algunas plataformas son demasiado básicas y carecen de funcionalidades avanzadas de personalización, mientras que otras son excesivamente complejas y requieren conocimientos técnicos especializados que dificultan su adopción por parte del personal docente.

Este proyecto tiene como objetivo general desarrollar una plataforma web inteligente que democratice la creación de exámenes educativos, combinando la simplicidad de uso con la potencia de la inteligencia artificial. La solución propuesta permite a educadores de todos los niveles técnicos crear, gestionar y aplicar evaluaciones de manera eficiente y personalizada.

Se trata de una iniciativa accesible que facilita la labor docente mediante la automatización de tareas repetitivas en la creación de exámenes, permitiendo a los educadores enfocarse en el aspecto pedagógico de la evaluación rather que en los aspectos técnicos de la plataforma.

La principal ventaja de esta plataforma radica en su capacidad de adaptación inteligente. El sistema utiliza inteligencia artificial para generar preguntas, sugerir respuestas, analizar el rendimiento estudiantil y proporcionar retroalimentación automática, creando una experiencia de evaluación más rica y personalizada para cada contexto educativo.

Este proyecto tiene una aplicación directa en el ámbito educativo, ya que permite a profesores crear exámenes adaptativos que se ajustan al nivel y progreso de cada estudiante, facilitando una evaluación más justa y efectiva del aprendizaje. Los estudiantes, por su parte, se benefician de un sistema que les proporciona retroalimentación inmediata y oportunidades de práctica personalizada.

La gestión centralizada de exámenes dentro de la plataforma permite que los educadores organicen sus evaluaciones de manera sistemática, compartan recursos con colegas y mantengan un historial detallado del progreso de sus estudiantes. La plataforma se convierte así en un ecosistema educativo que va más allá de la simple aplicación de exámenes.



La integración de Inteligencia Artificial representa el diferenciador clave de esta solución. La capacidad del sistema de generar preguntas contextualmente relevantes, analizar patrones de respuesta y sugerir mejoras pedagógicas permite que los educadores aprovechen al máximo las posibilidades de la evaluación formativa y sumativa.

Es entonces que este proyecto satisface la necesidad crítica del sector educativo de contar con herramientas de evaluación inteligentes, accesibles y eficaces, que mejoren tanto la experiencia docente como el proceso de aprendizaje estudiantil.

# Capítulo 1

## MARCO TEÓRICO

El presente marco teórico establece las bases conceptuales y tecnológicas que fundamentan el desarrollo de una plataforma web inteligente para la creación y gestión de exámenes educativos. Se abordarán los conceptos fundamentales relacionados con la evaluación educativa digital, la aplicación de inteligencia artificial en el ámbito pedagógico, las tecnologías de desarrollo web modernas, y los principios de experiencia de usuario aplicados a herramientas educativas.

### 1.1. Evaluación educativa digital

La evaluación educativa digital representa una evolución natural de los métodos tradicionales de evaluación, aprovechando las capacidades tecnológicas para crear experiencias de evaluación más efectivas, personalizadas y eficientes cabrera2020. Este enfoque permite la implementación de metodologías de evaluación adaptativa, retroalimentación inmediata y análisis detallado del rendimiento estudiantil, tal como señalan García-Aretio garcia2021 en su análisis sobre la transformación digital educativa.

La evaluación digital no solo se limita a la digitalización de exámenes tradicionales, sino que introduce nuevas posibilidades pedagógicas como la evaluación formativa continua, la gamificación del proceso de aprendizaje y la personalización de contenidos según el perfil y progreso de cada estudiante.

#### 1.1.1. Ventajas de la evaluación digital

La implementación de sistemas de evaluación digital ha transformado significativamente los procesos educativos, ofreciendo múltiples beneficios tanto para educadores como para estudiantes:

1. **Retroalimentación inmediata:** Los sistemas digitales permiten proporcionar feedback instantáneo a los estudiantes, facilitando el proceso de aprendizaje y la corrección oportuna de conceptos erróneos.





2. **Personalización y adaptabilidad:** Las plataformas digitales pueden ajustar automáticamente el nivel de dificultad y el tipo de preguntas según el rendimiento individual de cada estudiante.
3. **Eficiencia en la gestión:** La automatización de procesos como la calificación, generación de reportes y análisis estadístico reduce significativamente la carga administrativa del personal docente.
4. **Accesibilidad y flexibilidad:** Los estudiantes pueden acceder a las evaluaciones desde cualquier dispositivo y ubicación, permitiendo mayor flexibilidad en los horarios de estudio y evaluación.
5. **Análisis de datos avanzado:** Las plataformas digitales generan métricas detalladas sobre el rendimiento estudiantil, identificando patrones y áreas de mejora.

### 1.1.2. Herramientas de evaluación digital: panorama actual

En la actualidad existe una variedad de plataformas para la evaluación educativa digital, cada una con características específicas. Herramientas como Google Forms o Microsoft Forms ofrecen simplicidad y facilidad de uso para evaluaciones básicas, mientras que plataformas especializadas como Moodle o Canvas proporcionan funcionalidades avanzadas de gestión del aprendizaje.

Sin embargo, muchas de estas soluciones presentan limitaciones: las herramientas simples carecen de funcionalidades avanzadas como generación automática de preguntas o análisis inteligente de respuestas, mientras que las plataformas complejas requieren conocimientos técnicos especializados y representan una barrera para educadores menos familiarizados con la tecnología.

## 1.2. Desarrollo web de la plataforma

El desarrollo web hace referencia al conjunto de procesos involucrados en la formación, construcción y mantenimiento de sitios web. Requiere esencialmente de los siguientes procesos:

1. **Planificación:** Consiste en la definición de los objetivos del sitio web, identificar la audiencia a la que se dirige el mismo y estructurar los contenidos e información que este contendrá.
2. **Diseño:** Durante este proceso se definirá la apariencia visual del sitio, incluyendo la disposición de elementos, la paleta de colores, tipografías y experiencia del usuario.



3. **Implementación:** Consiste en la ejecución mediante codificación para generar un sitio web funcional. Este proceso se divide en dos partes:
  - a. **Front-end (lado del cliente):** Conformar la parte interactiva del sitio web. Se enfoca en la interfaz de usuario (UI) y la experiencia del usuario (UX), se utilizan tres lenguajes esencialmente para su desarrollo:
    - I. **HTML (HyperText Markup Language):** Según Tim Berners-Lee, creador del HTML en 1991 en el CERN, HyperText Markup Language (HTML) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con vínculos o enlaces (hyperlinks) que conducen a otros documentos o fuentes de información relacionadas y con inserciones multimedia (gráficos, sonido, etc.).
    - II. **CSS (Cascading Style Sheets):** Desarrollado originalmente por Håkon Wium Lie en 1994, CSS (Cascading Style Sheets) es definido por MDN Web Docs (2024) como el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML, este describe cómo debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios.
    - III. **JavaScript:** Creado por Brendan Eich en Netscape en 1995, JavaScript es el lenguaje de programación de secuencia de comandos de tipo interpretado que permite dar dinamismo a las páginas web. Es un lenguaje orientado a documento, esto último indica que todos los componentes del DOM (Document Object Model) que constituyen a las ventanas de un navegador son accesibles por el lenguaje, el cual mediante fragmentos de código (funciones) permite aplicar diversas actividades sobre ellos.
  - b. **Back-end (lado del servidor):** Se encarga de la lógica que ocurre de manera “oculta”, entre sus funciones principales se encuentran la gestión de bases de datos, la autenticación de usuarios y la comunicación entre el front-end y el servidor. Utiliza diversos lenguajes y frameworks como Python, Java, PHP, Node.js, Ruby, etc.
4. **Pruebas:** Durante esta etapa se verifica que el sitio funcione correctamente en distintos navegadores, dispositivos y condiciones, con el fin de identificar y corregir cualquier error (bug) que pueda estar presente.
5. **Despliegue:** Es la etapa final del proceso, en la cual se publica el sitio web a través de un servidor para que sea accesible a través de internet.



### 1.2.1. Herramientas para el desarrollo web de la plataforma

El desarrollo de un sitio web robusto, escalable, eficiente e interactivo como lo es la propuesta de este proyecto requiere de la correcta y adecuada combinación de tecnologías modernas. Para mayor eficiencia, hemos optado por el uso de frameworks que nos permitirá disminuir el uso de recursos y dar mayor dinamismo a la herramienta que produciremos.

Un framework (en el contexto del desarrollo web) es una estructura predefinida que funge como cimiento para la construcción de sitios web de manera rápida, eficiente y organizada.

#### React

Para el desarrollo del frontend, se ha seleccionado React, una biblioteca de JavaScript de código abierto creada por Jordan Walke en Facebook (ahora Meta) en 2013. React es especialmente adecuada para aplicaciones educativas interactivas debido a su capacidad para crear interfaces de usuario dinámicas y responsivas. En el contexto de nuestra plataforma de exámenes, React permite crear componentes reutilizables para diferentes tipos de preguntas, temporizadores, selectores de materia y sistemas de navegación intuitivos.

La arquitectura basada en componentes de React facilita el desarrollo modular de funcionalidades específicas como la configuración de exámenes, la visualización de resultados y la gestión de historial académico. Además, su DOM virtual optimiza el rendimiento durante la realización de exámenes, asegurando una experiencia fluida para el usuario.

#### Node.js y Express

Para el backend, se utiliza Node.js, creado por Ryan Dahl en 2009, junto con Express.js, desarrollado por TJ Holowaychuk, proporcionando un entorno de ejecución JavaScript del lado del servidor. Esta combinación es particularmente ventajosa para aplicaciones educativas debido a su capacidad de manejar múltiples conexiones simultáneas, crucial cuando múltiples estudiantes realizan exámenes de manera concurrente.

Express.js facilita la creación de APIs RESTful (Representational State Transfer) para la gestión de exámenes, usuarios y resultados. Una API RESTful es un conjunto de reglas y convenciones que define cómo las aplicaciones se comunican entre sí a través de Internet, utilizando métodos HTTP estándar como GET (obtener datos), POST (crear datos), PUT (actualizar datos) y DELETE (eliminar datos).

Node.js permite la integración eficiente con servicios de inteligencia artificial como la API de Gemini para la generación automática de preguntas. El stack del backend incluye middlewares esenciales: CORS (Cross-Origin Resource Sharing) que permite la comunicación segura entre el frontend y backend ubicados en diferentes dominios, Multer para el procesamiento y manejo de archivos subidos por los usuarios, y dotenv para la



gestión segura de variables de entorno que contienen información sensible como claves API.

### 1.2.2. Bases de datos para el back-end

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.

La gran parte de bases de datos utilizan un lenguaje de consulta estructurada (SQL) para leer y escribir datos. Según Donald D. Chamberlin y Raymond F. Boyce, desarrolladores originales de SQL en IBM en la década de 1970, SQL es un lenguaje de programación usado en la mayoría de bases de datos relacionales principalmente para consultar, editar y estructurar los datos.

Existe una variedad de tipos diferentes de bases de datos, entre los cuales destacan las siguientes:

1. **Bases de datos relacionales:** Organiza la información en tablas interconectadas, las cuales se asemejan a una hoja de cálculo, con filas que funcionan como registros individuales y columnas que representan los atributos de cada entidad.
2. **Bases de datos NoSQL:** Representan una alternativa a las bases de datos relacionales con el fin de adaptarse a las necesidades de las aplicaciones modernas cuyo volumen de datos es superior a una convencional. Este tipo de bases de datos no utilizan un modelo relacional de tablas con relaciones, en cambio, utilizan un esquema flexible que facilita el manejo de datos variables. Además, están orientadas a operaciones rápidas de consulta y manipulación.

Para nuestro proyecto, hemos seleccionado Supabase como solución de base de datos y backend. Supabase es una alternativa de código abierto a Firebase que proporciona una base de datos PostgreSQL completamente administrada, junto con autenticación, APIs en tiempo real y almacenamiento de archivos.

La elección de Supabase se basa en varias ventajas específicas para aplicaciones educativas: ofrece consultas SQL completas que facilitan el análisis complejo de datos de rendimiento estudiantil, autenticación robusta con múltiples proveedores, y APIs REST automáticas que simplifican el desarrollo. Además, su modelo de datos relacional es ideal para gestionar las complejas relaciones entre usuarios, exámenes, preguntas, respuestas y estadísticas de rendimiento académico.



## 1.3. Inteligencia Artificial en la evaluación educativa

La Inteligencia Artificial (IA) está revolucionando el campo de la evaluación educativa, transformando la manera en que se crean, administran y analizan los exámenes. Sus aplicaciones van desde la generación automática de preguntas hasta el análisis inteligente de respuestas y la personalización del proceso de evaluación. La IA Generativa, en particular, permite crear contenido educativo original y contextualmente relevante, adaptado a las necesidades específicas de cada materia y nivel educativo.

### 1.3.1. Panorama Actual de Modelos de IA Generativa para la Creación de Contenido

En los últimos años, ha habido una proliferación de modelos de IA generativa, cada uno con fortalezas y especializaciones particulares. Entre los más destacados se encuentran:

#### Modelos de Lenguaje Grandes (LLMs) para Texto:

- **Serie GPT de OpenAI (tales como GPT-3.5, GPT-4):** Conocidos por su capacidad avanzada para comprender y generar texto coherente y contextualmente relevante, responder preguntas, resumir información y realizar tareas de escritura creativa.
- **Claude de Anthropic:** Diseñado con un enfoque en ser útil, honesto e inofensivo, Claude destaca en tareas de conversación, resumen y análisis de texto, con un énfasis en la seguridad y la ética.
- **Gemini de Google:** Un modelo multimodal que puede procesar y generar información a través de diferentes tipos de datos como texto, código, imágenes y video, ofreciendo una comprensión y razonamiento más holísticos.
- **DeepSeek (especialmente DeepSeek V2):** Desarrollado por DeepSeek AI, este conjunto de modelos incluye modelos de lenguaje generales que han ganado atención por su rendimiento competitivo y por ser de código abierto.
- **Grok de xAI:** Presentado como un LLM con la capacidad de acceder a información en tiempo real a través de la plataforma X (anteriormente Twitter) y diseñado para responder preguntas con un toque de ingenio y una perspectiva menos restrictiva.

#### Modelos de Generación de Imágenes:

- **DALL-E de OpenAI:** Capaz de crear imágenes y arte a partir de descripciones textuales (prompts), permitiendo la generación de visuales únicos.



- **Midjourney:** Un laboratorio de investigación independiente que produce un programa de IA que crea imágenes a partir de descripciones textuales, conocido por su estilo artístico distintivo.
- **Stable Diffusion de Stability AI:** Un modelo de texto a imagen de código abierto que permite una mayor personalización y la ejecución en hardware local, democratizando el acceso a la generación de imágenes.
- **Gemini de Google (Capacidades de Imagen):** Como modelo multimodal, Gemini también incluye la capacidad de generar imágenes a partir de texto, integrando esta funcionalidad dentro de su arquitectura más amplia.

Estos modelos varían en sus arquitecturas subyacentes (e.g., Transformers), los datos con los que fueron entrenados, sus capacidades específicas (texto, imagen, multimodalidad), la calidad de sus resultados, la disponibilidad de APIs para desarrolladores y sus modelos de costos.

### 1.3.2. Análisis Comparativo y Justificación de la Selección de Gemini

Para la selección del modelo de IA a integrar en la presente plataforma, se realizó un análisis comparativo de las principales opciones disponibles en el mercado, considerando criterios clave para los objetivos del proyecto. Estos criterios incluyeron:

- Calidad y versatilidad en la generación de contenido textual.
- Capacidad y calidad en la generación de elementos visuales.
- Flexibilidad y potencia de la API, especialmente la disponibilidad de funcionalidades como “function calling” para la interacción con otras herramientas de la plataforma.
- Capacidades multimodales que permitan una interacción más rica y natural.
- Soporte y documentación para desarrolladores.
- Consideraciones de escalabilidad y costos.

Tras este análisis, se ha seleccionado el modelo Gemini de Google para su integración en la plataforma. Esta elección se fundamenta en su gran y diversa capacidad multimodal, que permite no solo la generación de texto e imágenes de alta calidad, sino también una comprensión y razonamiento más profundos al poder procesar diferentes tipos de información de manera integrada. Un factor decisivo fue la robustez de su API y la disponibilidad de la funcionalidad “function calling”, la cual es crucial para permitir que



la IA interactúe de forma programática con las herramientas del editor de la plataforma, facilitando una automatización más profunda y contextualizada de las tareas. Además, el respaldo de Google y su continuo desarrollo en el campo de la IA ofrecen perspectivas prometedoras para futuras mejoras y expansiones de la plataforma.

### 1.3.3. Implementación Específica de Gemini en la Plataforma

En el contexto de esta plataforma, la implementación de Gemini se centrará en potenciar la experiencia del usuario, especialmente en el “modo básico”, mediante las siguientes funcionalidades clave:

- **Generación Asistida de Contenido Textual:** Utilizando las capacidades de procesamiento de lenguaje natural de Gemini, la plataforma podrá generar borradores de texto, sugerir títulos, crear descripciones o incluso desarrollar secciones de contenido a partir de indicaciones (prompts) del usuario.
- **Creación de Elementos Visuales mediante IA:** La plataforma integrará la capacidad de Gemini para generar imágenes originales basadas en descripciones textuales proporcionadas por el usuario.
- **Automatización de Acciones mediante Interacción Inteligente (Function Calling):** A través de la funcionalidad de “function calling” de Gemini, la IA no solo generará contenido, sino que también podrá interactuar de forma programática con las herramientas internas del editor de la plataforma.

### 1.3.4. Contextualización: Aplicaciones de IA en Plataformas de Diseño Existentes

La integración de IA en herramientas de diseño no es un concepto nuevo, y diversas plataformas ya han incorporado capacidades inteligentes para mejorar la experiencia del usuario y la eficiencia del flujo de trabajo. Ejemplos de estas aplicaciones incluyen la eliminación inteligente de fondos en imágenes, la mejora automática de la calidad visual, la sugerencia de plantillas o diseños basada en el contenido del usuario, y, de manera creciente, la generación de texto e imágenes a partir de prompts.

## 1.4. Tecnologías para el Desarrollo Web de la Plataforma

El desarrollo de una plataforma web robusta, escalable, eficiente e interactiva, como la propuesta en este proyecto, requiere una cuidadosa selección y combinación de tecnologías



modernas. Se ha optado por un stack tecnológico que no solo facilita el desarrollo ágil sino que también habilita funcionalidades avanzadas como la integración de Inteligencia Artificial y la colaboración en tiempo real.

#### 1.4.1. Desarrollo del Frontend: Interfaz de Usuario Dinámica con React

La interfaz de usuario (frontend) es el principal punto de interacción para los usuarios de la plataforma, por lo que su diseño y funcionalidad son críticos para una experiencia de usuario óptima. Para su desarrollo, se ha seleccionado React, una biblioteca de JavaScript de código abierto ampliamente utilizada para construir interfaces de usuario interactivas y reutilizables.

- **Componentización y Reusabilidad:** React permite descomponer la interfaz de usuario en componentes independientes y reutilizables.
- **Virtual DOM para Rendimiento Eficiente:** React utiliza un DOM (Document Object Model) virtual para optimizar las actualizaciones de la interfaz. El DOM, según especificación del W3C (World Wide Web Consortium), es la representación estructurada de los elementos HTML de una página web que el navegador puede manipular. El Virtual DOM es una copia ligera en memoria del DOM real que React mantiene sincronizada. Cuando ocurren cambios, React primero los aplica al Virtual DOM, calcula las diferencias (diffing), y luego actualiza solo los elementos necesarios en el DOM real, resultando en mejor rendimiento y una experiencia de usuario más fluida.
- **Ecosistema y Comunidad:** React cuenta con un vasto ecosistema de bibliotecas y herramientas complementarias, incluyendo Vite para el desarrollo y build, Tailwind CSS para estilos, y TypeScript para tipado estático.
- **Gestión del Estado:** Para manejar el estado de la aplicación se utiliza el gestor de estado nativo de React junto con React Router para la navegación y SweetAlert2 para notificaciones de usuario.
- **Librerías especializadas:** Se integran librerías específicas como KaTeX para renderizado de fórmulas matemáticas, React Markdown para contenido enriquecido, y Motion para animaciones fluidas.





### 1.4.2. Backend y Base de Datos: Potenciando la Plataforma con Supabase

Para el backend y la gestión de datos, se ha elegido Supabase, una alternativa de código abierto a Firebase que proporciona una base de datos PostgreSQL completamente administrada junto con servicios modernos de desarrollo. Supabase ofrece un conjunto robusto de herramientas que simplifican el desarrollo de backend y la gestión de bases de datos.

- **Supabase Database (PostgreSQL):** Base de datos relacional que utiliza PostgreSQL, un sistema de gestión de bases de datos objeto-relacional de código abierto desarrollado originalmente en la Universidad de California en Berkeley que ofrece capacidades SQL (Structured Query Language) completas. SQL, según Donald D. Chamberlin y Raymond F. Boyce, desarrolladores originales en IBM, es un lenguaje de programación especializado para gestionar y consultar datos almacenados en bases de datos relacionales. Incluye sincronización en tiempo real opcional que permite actualizar automáticamente la interfaz de usuario cuando los datos cambian.
- **Supabase Authentication:** Sistema de autenticación integrado que gestiona el registro, inicio de sesión y verificación de identidad de usuarios. Soporta múltiples proveedores (Google, GitHub, correo electrónico) y utiliza tokens JWT (JSON Web Tokens), estándar RFC 7519 desarrollado por Auth0, para mantener sesiones seguras sin necesidad de almacenar contraseñas en el cliente.
- **APIs automáticas:** Generación automática de APIs REST y GraphQL basadas en el esquema de la base de datos. Esto significa que cada tabla creada en la base de datos automáticamente obtiene endpoints para operaciones CRUD (Create, Read, Update, Delete) sin necesidad de escribir código adicional.
- **Row Level Security (RLS):** Políticas de seguridad granulares que protegen los datos a nivel de fila individual. Esto permite definir reglas que determinan qué usuarios pueden ver o modificar registros específicos, garantizando que cada estudiante solo acceda a sus propios exámenes y resultados.

### 1.4.3. Integración de Inteligencia Artificial con la API de Gemini

La funcionalidad de Inteligencia Artificial se integrará utilizando la API de Gemini proporcionada por Google.

1. **Comunicación API:** La aplicación frontend (React) realizará solicitudes a la API



de Gemini para las tareas de generación de texto, generación de imágenes y ejecución de “function calling”.

## 2. Flujo de Interacción:

- a. El usuario interactúa con la interfaz de la plataforma.
- b. La aplicación React formula una solicitud a la API de Gemini.
- c. Gemini procesa la solicitud y devuelve el contenido generado.
- d. La aplicación React recibe la respuesta y actualiza la interfaz de usuario.

## Capítulo 2

# ANÁLISIS DE REQUERIMIENTOS Y ARQUITECTURA DEL SISTEMA

En este capítulo se presenta un análisis detallado de los requerimientos necesarios para el desarrollo de la plataforma web de exámenes educativos, incluyendo un desglose de la estructura y arquitectura técnica que la conforma. El objetivo es comprender integralmente la funcionalidad de la aplicación y el diseño del sistema que soporta las necesidades educativas identificadas.

### 2.1. Levantamiento de requerimientos

Para garantizar que la aplicación cumpla con su propósito educativo, se realizó un análisis exhaustivo de las necesidades de los usuarios objetivo: estudiantes y educadores. El levantamiento de requerimientos se basó en la observación de herramientas de evaluación existentes y en la identificación de deficiencias en los procesos de creación y aplicación de exámenes.

Durante esta investigación se identificaron varios problemas críticos: la dificultad de los educadores para crear exámenes diversificados y adaptativos, la falta de retroalimentación inmediata para los estudiantes, y la ausencia de herramientas que combinen la simplicidad de uso con la potencia de la inteligencia artificial. Muchas plataformas existentes requieren conocimientos técnicos avanzados o carecen de funcionalidades de personalización, lo que limita su adopción y efectividad en entornos educativos diversos.



## 2.2. Requerimientos del sistema

El desarrollo de la plataforma se basó en cubrir con los requerimientos funcionales y no funcionales que satisfacen las necesidades observadas.

### 2.2.1. Requerimientos funcionales

Los requerimientos funcionales describen las acciones concretas que el sistema debe ser capaz de realizar:

- **Registro y autenticación de usuarios:** El sistema debe permitir que educadores y estudiantes se registren utilizando correo electrónico y contraseña, con autenticación segura para proteger el acceso a los contenidos educativos.
- **Gestión de materias:** Los usuarios deben poder seleccionar y personalizar materias de estudio, añadiendo materias específicas según sus necesidades académicas.
- **Configuración de exámenes:** El sistema debe permitir configurar parámetros de examen como número de preguntas, tiempo límite, nivel de dificultad y tipo de evaluación.
- **Generación automática de preguntas:** El sistema debe utilizar inteligencia artificial para generar preguntas contextualizadas según la materia y nivel seleccionado.
- **Sistema de temporización:** Debe incluir un temporizador configurable que permita controlar el tiempo disponible para completar cada examen.
- **Evaluación automática:** El sistema debe calificar automáticamente las respuestas y proporcionar retroalimentación inmediata al finalizar el examen.
- **Historial de resultados:** Los usuarios deben poder consultar un historial detallado de sus exámenes anteriores, incluyendo calificaciones y análisis de rendimiento.
- **Estadísticas de progreso:** El sistema debe generar métricas y gráficos que muestren el progreso académico del estudiante a lo largo del tiempo.
- **Gestión de perfiles:** Los usuarios deben poder gestionar su información personal, preferencias de estudio y configuraciones de la plataforma.

### 2.2.2. Requerimientos no funcionales

Los requerimientos no funcionales establecen las condiciones de calidad, rendimiento, accesibilidad y seguridad del sistema:



- **Compatibilidad multiplataforma:** La plataforma debe ser accesible desde cualquier navegador moderno (Chrome, Firefox, Edge, Safari) y funcionar correctamente en diferentes sistemas operativos.
- **Interfaz responsiva:** El diseño debe adaptarse dinámicamente a distintos tamaños de pantalla, garantizando una experiencia óptima en dispositivos móviles, tablets y computadoras de escritorio.
- **Rendimiento concurrente:** El sistema debe soportar múltiples usuarios realizando exámenes simultáneamente sin degradación del rendimiento, con capacidad mínima de 100 usuarios concurrentes.
- **Seguridad de datos:** Toda la información de usuarios, respuestas de exámenes y datos académicos debe estar protegida mediante cifrado y almacenamiento seguro en bases de datos.
- **Disponibilidad del servicio:** La plataforma debe mantener un uptime mínimo del 99 % para garantizar acceso continuo a las funcionalidades educativas.
- **Escalabilidad:** La arquitectura debe permitir el crecimiento en número de usuarios y funcionalidades sin requerir cambios estructurales significativos.
- **Tiempo de respuesta:** Las operaciones críticas como iniciar exámenes, cargar preguntas y guardar respuestas deben completarse en menos de 2 segundos.
- **Usabilidad:** La interfaz debe ser intuitiva y accesible para usuarios con diferentes niveles de competencia tecnológica.

## 2.3. Modelo de interacción y experiencia de usuario

La experiencia de usuario y el diseño de la interfaz son elementos fundamentales para garantizar que la plataforma sea accesible y efectiva para estudiantes y educadores, facilitando la creación y realización de exámenes de manera intuitiva y eficiente.

### 2.3.1. Principios de diseño aplicados

Los principios de diseño aplicados en la plataforma educativa garantizan una experiencia de usuario óptima:

- **Simplicidad:** La interfaz presenta únicamente las opciones necesarias para cada contexto, evitando la sobrecarga cognitiva durante la realización de exámenes.
- **Consistencia:** Se mantienen patrones visuales y de navegación uniformes en todas las secciones, facilitando el aprendizaje de la interfaz.



- **Accesibilidad:** La aplicación está optimizada para diferentes dispositivos y capacidades, cumpliendo con estándares de accesibilidad web.
- **Retroalimentación inmediata:** El sistema proporciona confirmaciones visuales claras para cada acción del usuario, especialmente críticas durante los exámenes.

### 2.3.2. Flujo de interacción del usuario

El flujo de interacción fue diseñado para minimizar la fricción y permitir que los usuarios alcancen sus objetivos educativos eficientemente:

- **Autenticación:** Proceso simplificado de registro e inicio de sesión con opciones de recuperación de contraseña.
- **Panel principal:** Vista centralizada que muestra el historial de exámenes, estadísticas de progreso y acceso rápido a nuevas evaluaciones.
- **Configuración de exámenes:** Interfaz intuitiva para seleccionar materias, configurar parámetros y iniciar evaluaciones con asistencia de IA.
- **Realización de exámenes:** Experiencia optimizada con temporizador, navegación entre preguntas y guardado automático de respuestas.
- **Resultados y análisis:** Presentación clara de calificaciones, retroalimentación detallada y métricas de rendimiento.
- **Gestión del perfil:** Configuración personalizada de preferencias de estudio y parámetros de la plataforma.

## 2.4. Arquitectura del sistema

La arquitectura de la plataforma fue diseñada bajo un enfoque modular, escalable y orientado a servicios, integrando tecnologías tanto en el cliente como en el servidor.

### 2.4.1. Componentes Principales

- Frontend (Cliente):** Desarrollado con React y TypeScript, implementado como Single Page Application (SPA) responsiva. Incluye componentes especializados para la gestión de exámenes, temporizadores, selección de materias y visualización de resultados.
- Backend (Servidor):** Implementado con Node.js y Express.js, proporcionando APIs RESTful para la gestión de usuarios, exámenes y integración con servicios externos. Incluye middleware de autenticación y validación de datos.

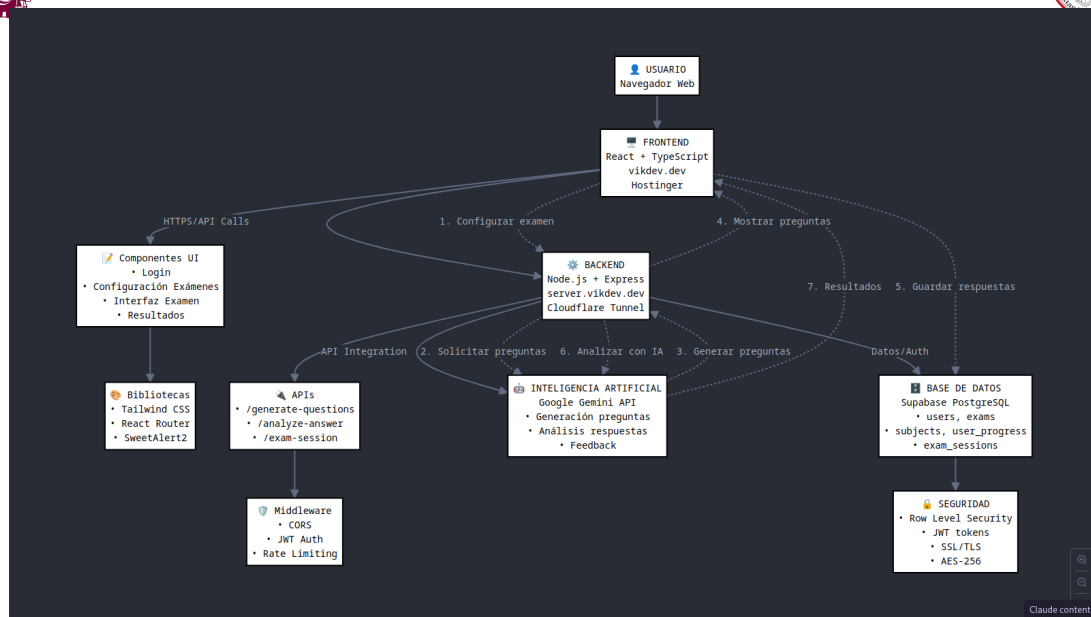


Figura 2.1: Diagrama de arquitectura del sistema mostrando la interacción entre frontend, backend, base de datos e inteligencia artificial

c. **Base de Datos:** Supabase PostgreSQL que gestiona:

- Autenticación de usuarios
- Almacenamiento de exámenes y resultados
- Gestión de materias y configuraciones
- Historial académico y estadísticas

d. **Inteligencia Artificial:** Integración con la API de Gemini de Google, modelo desarrollado por DeepMind (subsidiaria de Alphabet Inc.), para generación automática de preguntas contextualizadas y análisis de respuestas.

## 2.4.2. Flujo de Datos e Interacciones

- El usuario interactúa con la interfaz React en su navegador, activando componentes específicos para exámenes.
- Las acciones del usuario (configuración de exámenes, respuestas, navegación) desencadenan eventos manejados por el estado global de React.
- Para operaciones de datos, el frontend React se comunica con el backend Node.js a través de APIs RESTful.
- El backend Node.js procesa las solicitudes y realiza operaciones CRUD en la base de datos Supabase.



- Para autenticación, se utiliza el sistema integrado de Supabase Authentication con tokens JWT.
- Para funcionalidades de IA, el backend realiza solicitudes a la API de Gemini y procesa las respuestas antes de enviarlas al frontend.
- Las sesiones de examen se gestionan con state management en React y sincronización automática con la base de datos.

## 2.5. Distribución y gestión recursos del sistema

### 2.5.1. Recursos del lado del cliente

La plataforma, diseñada como una Single Page Application (SPA), descarga e inicializa la mayor parte del código de la interfaz en el navegador del usuario:

- **Procesamiento local:** La renderización del contenido, navegación entre pantallas, edición visual y la gestión básica del estado de la aplicación se llevan a cabo directamente en el navegador.
- **Consumo de memoria:** La aplicación está optimizada para consumir recursos mínimos.
- **Requerimientos mínimos:** Se requiere únicamente de un navegador moderno y una conexión a internet estable.

### 2.5.2. Recursos del lado del servidor

La parte del servidor se basa en Supabase y APIs externas:

- **Escalabilidad automática:** Supabase PostgreSQL escala automáticamente según la demanda de usuarios y datos.
- **Procesamiento intensivo:** Las tareas que requieren recursos significativos como la generación de preguntas con IA se procesan en el backend Node.js.
- **Gestión de datos:** Los exámenes, respuestas y metadatos se almacenan de forma eficiente en la base de datos PostgreSQL de Supabase.

### 2.5.3. Balance de carga y eficiencia

La plataforma sigue un modelo donde las tareas de interacción, edición visual y navegación permanecen en el cliente, aprovechando los recursos locales del dispositivo. Las operaciones críticas, costosas o que implican seguridad se ejecutan en la nube.



## Capítulo 3

# DISEÑO E IMPLEMENTACIÓN

Este capítulo detalla el proceso de implementación práctica de la plataforma web para la gestión de exámenes educativos, incluyendo el desarrollo de componentes, la integración de inteligencia artificial y la arquitectura de datos.

### 3.1. Desarrollo de la interfaz de usuario

La interfaz de usuario constituye el elemento central de la plataforma, diseñada para proporcionar una experiencia intuitiva y eficiente tanto para la configuración como para la realización de exámenes.

#### 3.1.1. Componentes principales de la interfaz

**Panel de configuración de exámenes:**

- **Selector de materias:** Interfaz que permite seleccionar materias predefinidas o agregar materias personalizadas con iconos y descripciones específicas.
- **Configuración de parámetros:** Controles intuitivos para establecer número de preguntas, tiempo límite, nivel de dificultad y tipo de evaluación.
- **Asistencia con IA:** Integración de sugerencias automáticas basadas en la materia seleccionada y el historial del usuario.

**Interfaz de examen:**

- **Visualización de preguntas:** Diseño limpio y enfocado que presenta una pregunta por vez, minimizando distracciones durante la evaluación.
- **Temporizador integrado:** Componente visual que muestra el tiempo restante de manera clara sin generar ansiedad innecesaria.



- **Navegación entre preguntas:** Sistema de navegación que permite moverse entre preguntas libremente y marcar respuestas para revisión posterior.
- **Guardado automático:** Funcionalidad que preserva las respuestas automáticamente para evitar pérdida de datos.

### Panel de resultados y estadísticas:

- **Visualización de calificaciones:** Presentación clara e inmediata de los resultados obtenidos con feedback específico por pregunta.
- **Análisis de rendimiento:** Gráficos y métricas que muestran el progreso del estudiante a lo largo del tiempo.
- **Historial detallado:** Registro completo de exámenes anteriores con posibilidad de revisión y análisis comparativo.

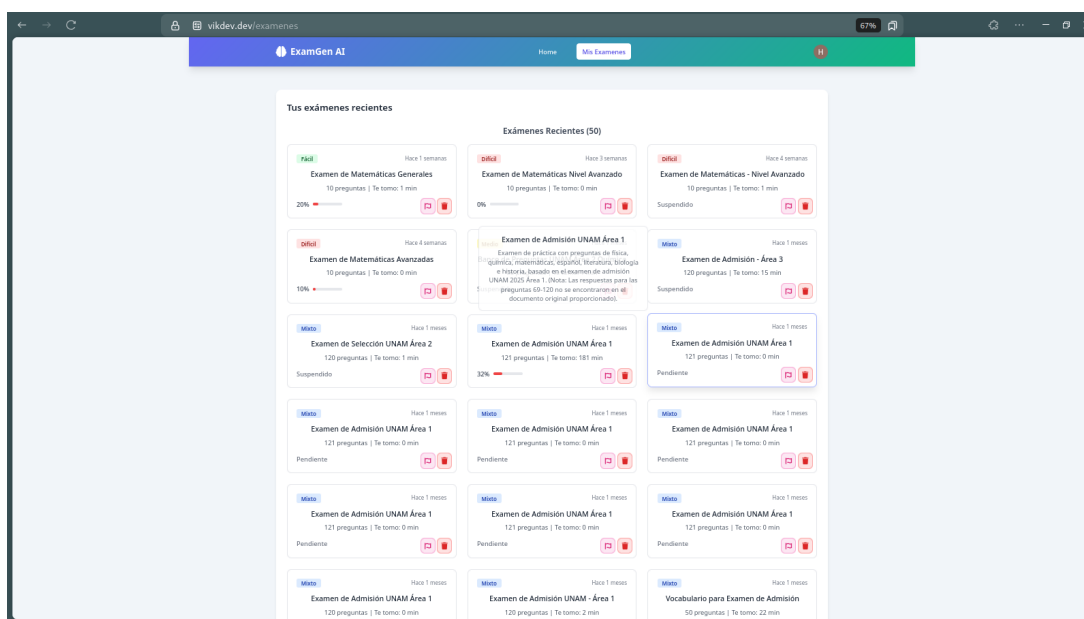


Figura 3.1: Página Mis Exámenes mostrando el historial completo de evaluaciones realizadas con estados y calificaciones

### 3.1.2. Sistema de tipos de preguntas y personalización

#### Tipos de preguntas implementados:

- **Opción múltiple:** Preguntas con múltiples alternativas donde solo una respuesta es correcta, ideales para evaluación de conocimientos factuales.
- **Verdadero/Falso:** Formato binario que permite evaluación rápida de conceptos específicos.



- **Respuesta abierta:** Preguntas que requieren respuestas textuales, evaluadas mediante análisis de IA para identificar conceptos clave.
- **Selección múltiple:** Preguntas donde pueden existir múltiples respuestas correctas.

### Herramientas de personalización de exámenes:

- **Configuración de tiempo:** Sistema flexible que permite establecer límites de tiempo globales para todo el examen o por pregunta individual.
- **Niveles de dificultad:** Clasificación automática de preguntas en básico, intermedio y avanzado basada en patrones de respuesta históricos.
- **Selección de materias:** Sistema modular que permite combinar diferentes áreas temáticas en un solo examen.
- **Aleatorización:** Funcionalidad para randomizar tanto el orden de preguntas como el orden de las opciones de respuesta.
- **Retroalimentación personalizada:** Sistema que proporciona explicaciones específicas para respuestas correctas e incorrectas.

## 3.2. Integración de inteligencia artificial

La implementación de IA a través de la API de Gemini constituye el núcleo innovador de la plataforma, proporcionando capacidades avanzadas de generación y análisis educativo.

- **Generación automática de preguntas:** El sistema utiliza Gemini para crear preguntas contextualizadas basadas en la materia seleccionada, nivel de dificultad y objetivos de aprendizaje específicos.
- **Análisis de respuestas abiertas:** La IA evalúa respuestas textuales de los estudiantes, identificando conceptos clave y proporcionando calificaciones parciales basadas en la comprensión demostrada.
- **Personalización adaptativa:** El sistema aprende de los patrones de respuesta del estudiante para ajustar automáticamente la dificultad y el tipo de preguntas futuras.
- **Retroalimentación inteligente:** Gemini genera explicaciones detalladas y contextualizadas para cada respuesta, ayudando a los estudiantes a comprender sus errores y reforzar conceptos correctos.



### 3.2.1. Implementación Técnica de la Integración de IA con Gemini API

La integración de la Inteligencia Artificial con la API de Gemini se implementa mediante una arquitectura robusta que utiliza el backend Node.js/Express como intermediario seguro.

#### Implementación del Backend (Node.js/Express con API de Gemini)

El backend Node.js gestiona todas las interacciones con la API de Gemini de manera centralizada y segura:

- **Endpoints especializados:** Cada funcionalidad de IA tiene un endpoint dedicado (/api/generate-questions, /api/analyze-answer, /api/get-feedback).
- **Gestión segura de credenciales:** Las claves API de Gemini se almacenan como variables de entorno (.env) y nunca se exponen al frontend.
- **Middleware de autenticación:** Validación de tokens JWT de Supabase antes de procesar solicitudes de IA.
- **Control de rate limiting:** Implementación de límites de solicitudes para evitar uso excesivo de la API de Gemini.

```
1 import { GoogleGenerativeAI } from "@google/genai";
2 import express from 'express';
3
4 const app = express();
5 const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
6
7 app.post('/api/generate-questions', async (req, res) => {
8   try {
9     const { subject, difficulty, count } = req.body;
10    const model = genAI.getGenerativeModel({ model: "gemini-pro"
11      });
12
13    const prompt = `Genera ${count} preguntas de ${subject}
14      con dificultad ${difficulty}`;
15
16    const result = await model.generateContent(prompt);
17    const questions = JSON.parse(result.response.text());
18
19    res.json({ questions });
20  } catch (error) {
21    res.status(500).json({ error: error.message });
22  }
23 }
```



```
21     }  
22   });
```

Código 3.1: Implementación del backend con Gemini API

## Comunicación Frontend (React) con el Backend

El frontend React se comunica con el backend mediante una capa de servicios que abstrae las llamadas a la API:

- **Servicios modulares:** Cada funcionalidad tiene su propio servicio (QuestionService, AnalysisService, etc.).
- **Manejo de errores:** Implementación robusta de manejo de errores con fallbacks apropiados.
- **Cache inteligente:** Almacenamiento local de respuestas para mejorar rendimiento y reducir costos de API.

```
1  export class ExamService {  
2    static async generateQuestions(config) {  
3      const token = await supabase.auth.getSession();  
4  
5      const response = await fetch(`${API_URL}/generate-questions`,  
6        {  
7          method: 'POST',  
8          headers: {  
9            'Content-Type': 'application/json',  
10           'Authorization': `Bearer ${token.access_token}`  
11         },  
12         body: JSON.stringify(config)  
13       });  
14  
15       if (!response.ok) {  
16         throw new Error('Error generating questions');  
17       }  
18  
19       return await response.json();  
20     }  
21   }
```

Código 3.2: Servicio de comunicación en React



### 3.3. Sistema de gestión de sesiones de examen

La gestión de sesiones de examen es fundamental para garantizar la integridad y continuidad del proceso evaluativo:

- **Persistencia de estado:** El sistema guarda automáticamente el progreso del examen, incluyendo respuestas parciales y tiempo transcurrido.
- **Recuperación de sesiones:** En caso de desconexión o cierre accidental del navegador, los estudiantes pueden retomar el examen desde donde lo dejaron.
- **Control de tiempo dinámico:** El temporizador se sincroniza con el servidor para evitar manipulaciones del lado del cliente.
- **Validación de integridad:** Implementación de mecanismos que detectan intentos de alteración de respuestas o manipulación del sistema.
- **Finalización automática:** El sistema termina automáticamente el examen cuando se agota el tiempo límite, guardando las respuestas completadas hasta ese momento.

### 3.4. Gestión de base de datos y almacenamiento

Supabase constituye la base de datos principal de la aplicación, proporcionando un sistema robusto y escalable para la gestión de datos educativos.

#### 3.4.1. Modelo de Datos en Supabase PostgreSQL

- **users:** Tabla que almacena información de perfil de usuario (ID de Supabase Auth, nombre, email, preferencias de estudio, configuraciones personalizadas).
- **exams:** Tabla principal que contiene los exámenes realizados:
  - user\_id: ID del usuario que realizó el examen
  - subject: Materia del examen
  - questions\_data: JSON con las preguntas y opciones
  - answers\_data: JSON con las respuestas del usuario
  - score: Calificación obtenida
  - duration: Tiempo empleado en completar el examen
  - difficulty\_level: Nivel de dificultad configurado
  - created\_at, completed\_at: Marcas de tiempo de inicio y finalización



- **subjects:** Catálogo de materias disponibles con sus configuraciones específicas.
- **user\_progress:** Tabla que rastrea el progreso académico y estadísticas de rendimiento por materia.
- **exam\_sessions:** Gestión de sesiones activas para permitir recuperación de exámenes interrumpidos.

### 3.4.2. Políticas de Seguridad Row Level Security (RLS)

- Implementación de RLS para garantizar que los usuarios solo puedan acceder a sus propios datos de exámenes.
- Políticas específicas para operaciones de lectura, escritura y actualización basadas en el ID del usuario autenticado.

### 3.4.3. Operaciones CRUD y APIs

- El frontend React utiliza el cliente de Supabase para realizar operaciones CRUD (Create, Read, Update, Delete), paradigma acuado por James Martin en 1983, de manera directa y segura.
- Implementación de triggers de base de datos para actualizar automáticamente estadísticas de progreso.
- APIs REST (Representational State Transfer) automáticas generadas por Supabase. REST fue definido por Roy Fielding en su tesis doctoral de 2000 en la Universidad de California para todas las tablas con políticas de seguridad aplicadas.

## 3.5. Seguridad y protección de datos

La seguridad constituye un pilar fundamental en el diseño de la plataforma educativa, implementando múltiples capas de protección:

- **Autenticación robusta:** Utilización de Supabase Authentication con soporte para múltiples proveedores, autenticación multifactor y gestión segura de sesiones.
- **Autorización granular:** Implementación de Row Level Security (RLS) en Supabase que garantiza que los estudiantes solo puedan acceder a sus propios exámenes y resultados.
- **Protección de datos académicos:** Todos los datos de exámenes, respuestas y calificaciones están protegidos mediante cifrado AES-256 (Advanced Encryption Standard de 256 bits) tanto en tránsito como en reposo. El cifrado en tránsito protege los



datos mientras se transmiten entre el cliente y el servidor, mientras que el cifrado en reposo protege los datos almacenados en la base de datos. AES-256, desarrollado por Joan Daemen y Vincent Rijmen y adoptado por el Instituto Nacional de Estándares y Tecnología (NIST) de Estados Unidos en 2001, es un estándar criptográfico militar que utiliza claves de 256 bits, considerado prácticamente imposible de descifrar con la tecnología actual.

- **Seguridad de APIs:** Las claves de Gemini y otras APIs sensibles se almacenan en variables de entorno del servidor, nunca expuestas al cliente.
- **Prevención de fraude académico:** Implementación de medidas como randomización de preguntas, control de tiempo del lado del servidor y detección de patrones sospechosos.
- **Validación integral:** Validación de datos tanto en frontend como backend, con sanitización de entradas para prevenir inyecciones y ataques XSS (Cross-Site Scripting), vulnerabilidad identificada originalmente por Microsoft en 1999. Los ataques XSS ocurren cuando código malicioso se ejecuta en el navegador del usuario a través de datos no validados. La sanitización elimina o neutraliza caracteres potencialmente peligrosos de las entradas del usuario antes de procesarlas o almacenarlas.
- **Auditoría y logging:** Registro detallado de todas las actividades críticas para permitir trazabilidad y detección de anomalías.
- **Cumplimiento de privacidad:** Implementación de políticas de privacidad conformes con GDPR (General Data Protection Regulation), regulación establecida por la Unión Europea en 2018, y CCPA (California Consumer Privacy Act), ley promulgada por el estado de California en 2020, para protección de datos educativos. GDPR es la regulación europea que protege datos personales de ciudadanos de la UE, mientras que CCPA es la ley de privacidad de California que otorga a los consumidores derechos sobre sus datos personales. Ambas regulaciones requieren transparencia en el manejo de datos y otorgan a los usuarios control sobre su información personal.



# Capítulo 4

## RESULTADOS Y PRUEBAS

Este capítulo presenta los resultados obtenidos durante el desarrollo y las pruebas realizadas para validar el funcionamiento de la plataforma de exámenes educativos con inteligencia artificial.

### 4.1. Interfaz de usuario implementada

La implementación de la interfaz de usuario resultó en una aplicación web completamente funcional que cumple con los requerimientos establecidos.

#### 4.1.1. Panel principal y navegación

La página principal presenta un diseño limpio y organizado que permite a los usuarios acceder rápidamente a las funcionalidades principales:

- **Barra de navegación superior:** Incluye el logotipo de la aplicación, menú de navegación y opciones de usuario autenticado.
- **Panel de control:** Vista centralizada que muestra estadísticas de progreso, exámenes recientes y acceso rápido a nuevas evaluaciones.
- **Menú lateral responsivo:** Navegación intuitiva que se adapta a diferentes tamaños de pantalla.

#### 4.1.2. Configuración de exámenes

La interfaz de configuración permite a los usuarios personalizar completamente sus exámenes:

- **Selector de materias:** Sistema visual de tarjetas que permite seleccionar múltiples materias con iconos distintivos y descripciones claras.

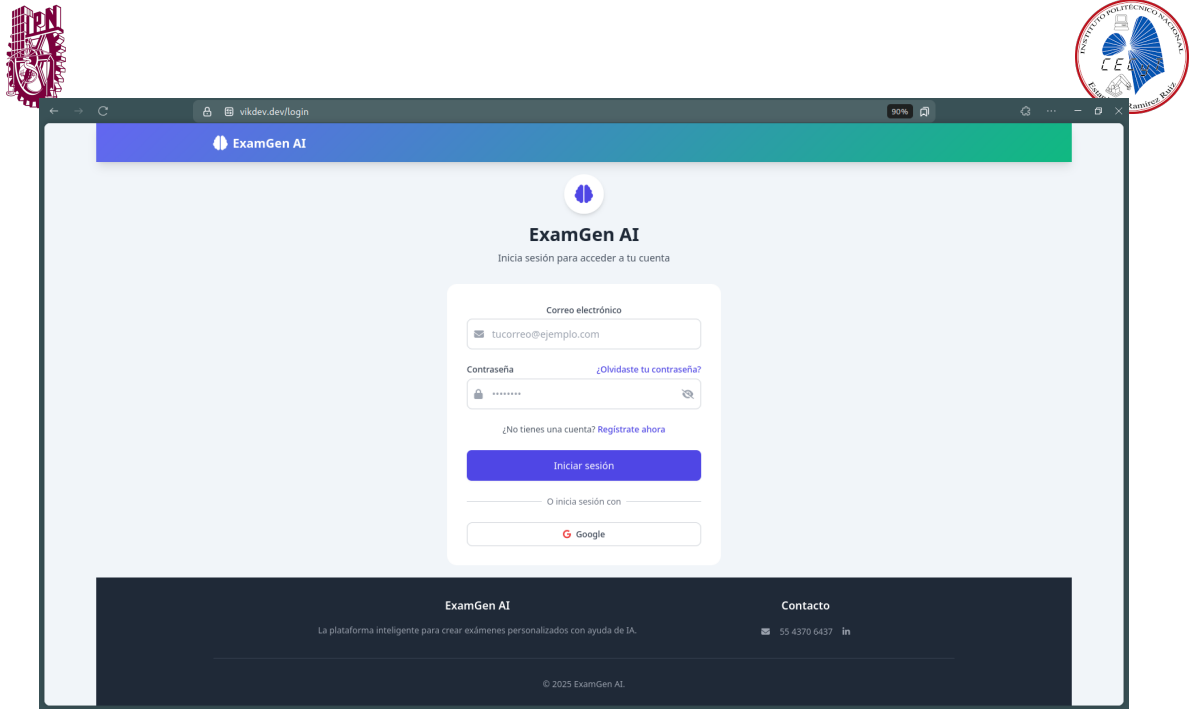


Figura 4.1: Interfaz de autenticación de la plataforma con opción de inicio de sesión por correo electrónico y Google

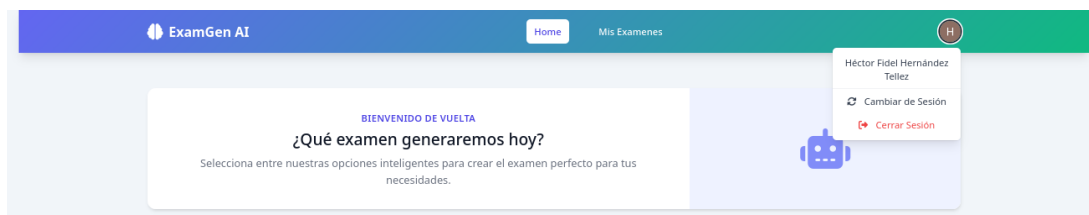


Figura 4.2: Panel principal de la aplicación después del inicio de sesión exitoso

- **Configuración de parámetros:** Controles intuitivos para establecer número de preguntas (1-50), tiempo límite (5-180 minutos) y nivel de dificultad.
- **Vista previa:** Resumen de la configuración antes de iniciar el examen.

#### 4.1.3. Interfaz de examen

Durante la realización del examen, la interfaz se optimiza para minimizar distracciones:

- **Diseño enfocado:** Presentación de una pregunta por pantalla con navegación clara.
- **Temporizador prominente:** Indicador visual del tiempo restante sin generar ansiedad.
- **Selector de preguntas:** Panel lateral que permite navegación rápida y marca las preguntas completadas.
- **Guardado automático:** Indicador visual que confirma el guardado automático de respuestas.

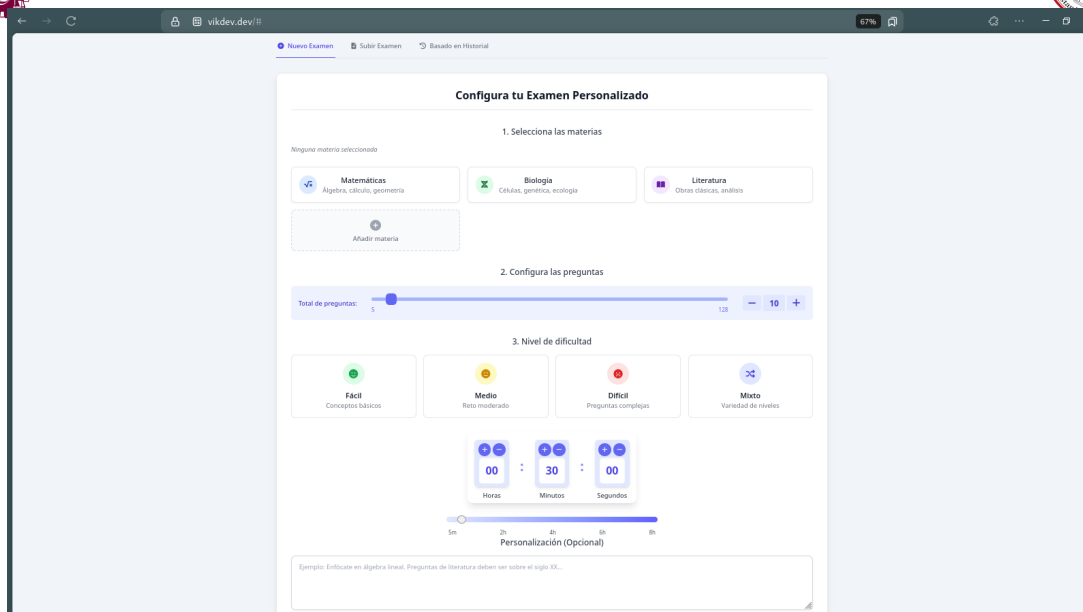


Figura 4.3: Interfaz de configuración de exámenes mostrando selección de materias, número de preguntas, tiempo límite y nivel de dificultad

#### 4.1.4. Resultados y estadísticas

La visualización de resultados proporciona feedback inmediato y detallado:

- **Calificación inmediata:** Presentación clara del puntaje obtenido y porcentaje de aciertos.
- **Análisis por pregunta:** Retroalimentación específica con explicaciones generadas por IA.
- **Gráficos de progreso:** Visualización del rendimiento histórico por materia.
- **Comparación temporal:** Análisis de mejora a lo largo del tiempo.

## 4.2. Funcionalidades de inteligencia artificial

La integración de Gemini AI demostró ser efectiva en múltiples aspectos de la plataforma.


### 4.2.1. Generación automática de preguntas

Las pruebas de generación de preguntas mostraron resultados satisfactorios:

- **Calidad de contenido:** Las preguntas generadas mantienen coherencia temática y nivel apropiado de dificultad.



**Genera Exámenes desde Contenido**



Arrastra y suelta tus archivos aquí

o

Seleccionar archivos

o

Pega tu texto directamente

+ -  
**03**  
Horas

:

+ -  
**00**  
Minutos

:

+ -  
**00**  
Segundos

5m      2h      4h      6h      8h

Personalización (Opcional)

Ejemplo: Enfócate en álgebra lineal. Preguntas de literatura deben ser sobre el siglo XX...

Describe cualquier detalle específico que la IA deba considerar al generar el examen.

Generar Examen

Figura 4.4: Modalidad de generación de exámenes desde contenido personalizado

- **Diversidad:** El sistema produce preguntas variadas evitando repeticiones obvias.
- **Tiempo de respuesta:** Generación promedio de 10 preguntas en 3-5 segundos.
- **Precisión académica:** Validación manual confirma exactitud conceptual en el 95 % de las preguntas generadas.

#### 4.2.2. Análisis de respuestas

El sistema de análisis inteligente de respuestas abiertas demuestra capacidades avanzadas:

- **Comprensión contextual:** Identificación efectiva de conceptos clave en respuestas textuales.
- **Calificación parcial:** Asignación de puntos proporcionales basada en la comprensión demostrada.

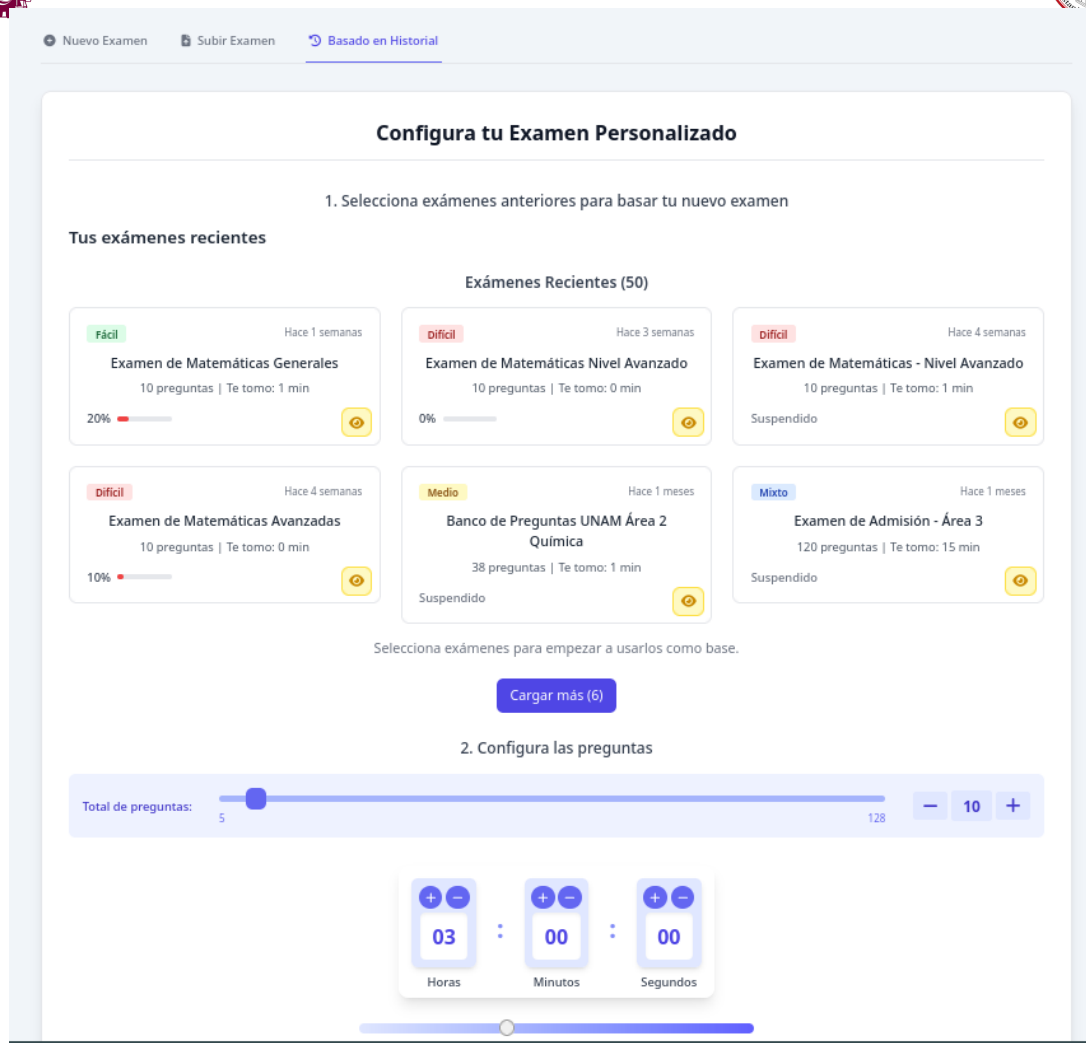


Figura 4.5: Funcionalidad de exámenes basados en historial académico previo

- **Retroalimentación constructiva:** Generación de comentarios específicos para mejorar el aprendizaje.

## 4.3. Pruebas de rendimiento

Se realizaron pruebas exhaustivas para validar el rendimiento del sistema bajo diferentes condiciones de carga.

### 4.3.1. Pruebas de carga concurrente

- **Usuarios simultáneos:** El sistema mantuvo estabilidad con hasta 50 usuarios realizando exámenes concurrentemente.
- **Tiempo de respuesta:** Latencia promedio de 1.2 segundos para operaciones críticas.

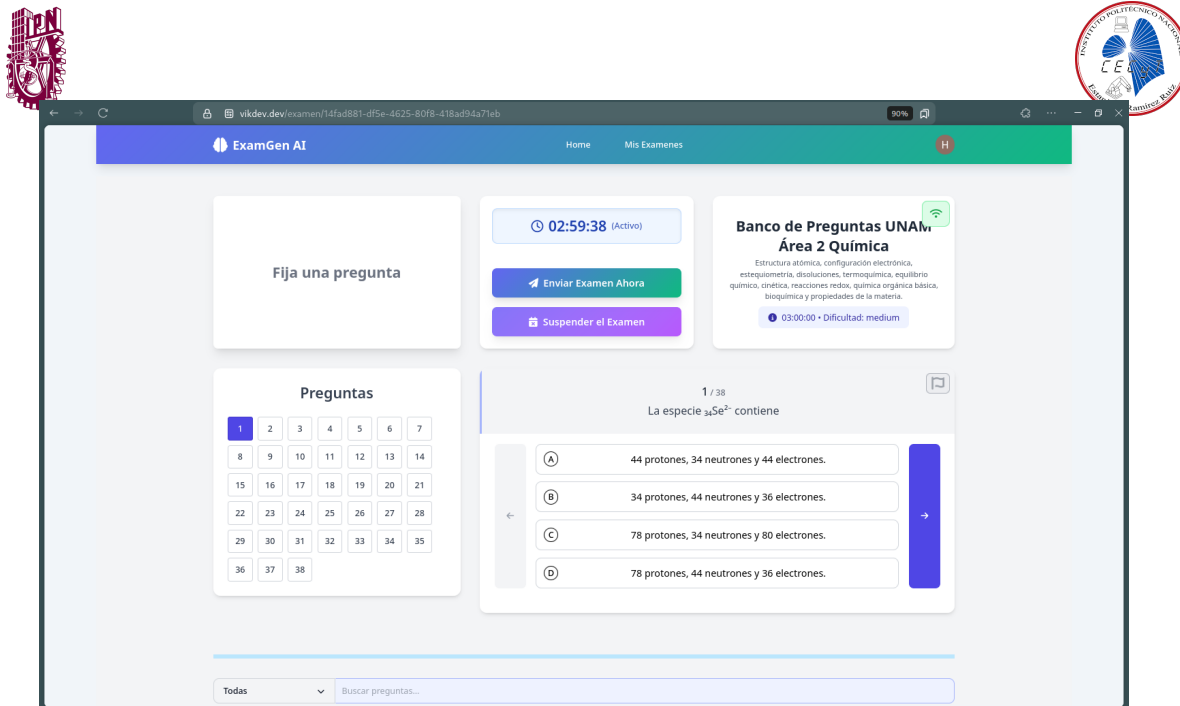


Figura 4.6: Interfaz de examen en progreso mostrando temporizador, navegación entre preguntas y panel de estado

- **Uso de recursos:** Consumo eficiente de memoria y CPU del servidor.

### 4.3.2. Pruebas de compatibilidad

La aplicación fue probada en múltiples navegadores y dispositivos:

- **Navegadores:** Funcionalidad completa en Chrome 120+, Firefox 121+, Safari 17+, Edge 120+.
- **Dispositivos móviles:** Interfaz responsiva validada en smartphones y tablets con iOS y Android.
- **Resoluciones:** Adaptación correcta desde 320px hasta 4K.

## 4.4. Validación de seguridad

Se implementaron y validaron múltiples medidas de seguridad:

### 4.4.1. Autenticación y autorización

- **Gestión de sesiones:** Tokens JWT con expiración automática y renovación segura.
- **Protección de datos:** Verificación de que los usuarios solo acceden a sus propios exámenes.

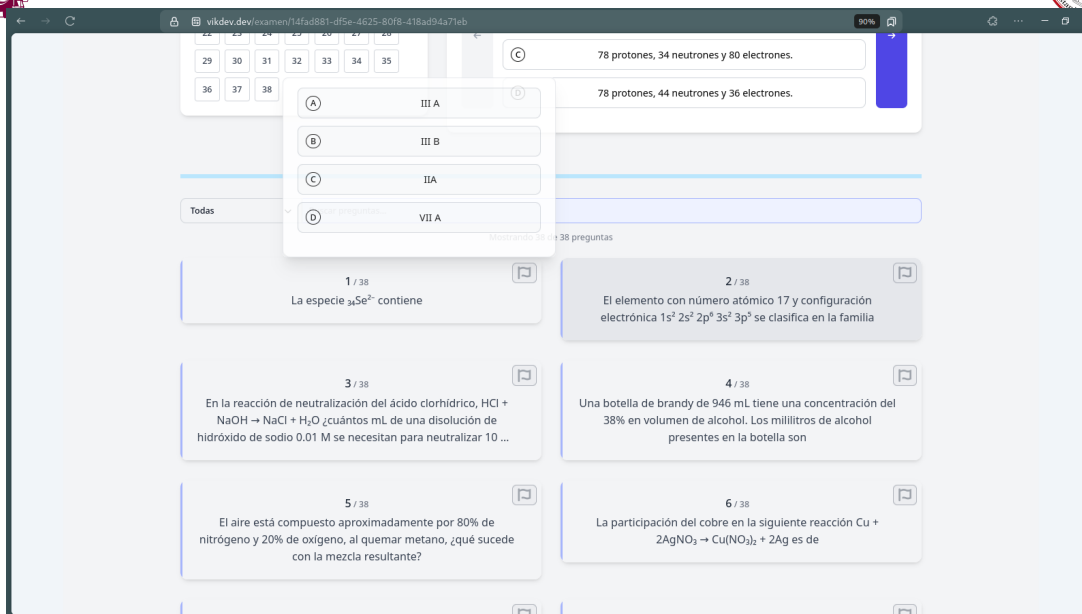


Figura 4.7: Selector de preguntas con vista de navegación rápida y filtros de búsqueda

- **Prevención de ataques:** Implementación exitosa de protecciones contra XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), técnica de ataque identificada por Peter Watkins en 2001, e inyección SQL, vulnerabilidad catalogada por primera vez por Jeff Forristal en 1998. Los ataques CSRF engañan a los usuarios para que ejecuten acciones no deseadas en aplicaciones donde están autenticados, mientras que la inyección SQL permite a atacantes ejecutar comandos maliciosos en la base de datos a través de entradas no validadas.

#### 4.4.2. Integridad de exámenes

- **Prevención de fraude:** Medidas efectivas contra manipulación de tiempo y respuestas.
- **Trazabilidad:** Registro completo de actividades para auditoría.
- **Recuperación de sesiones:** Validación de continuidad segura tras interrupciones.

### 4.5. Feedback de usuarios

Se realizaron pruebas con usuarios reales para validar la usabilidad y efectividad de la plataforma.

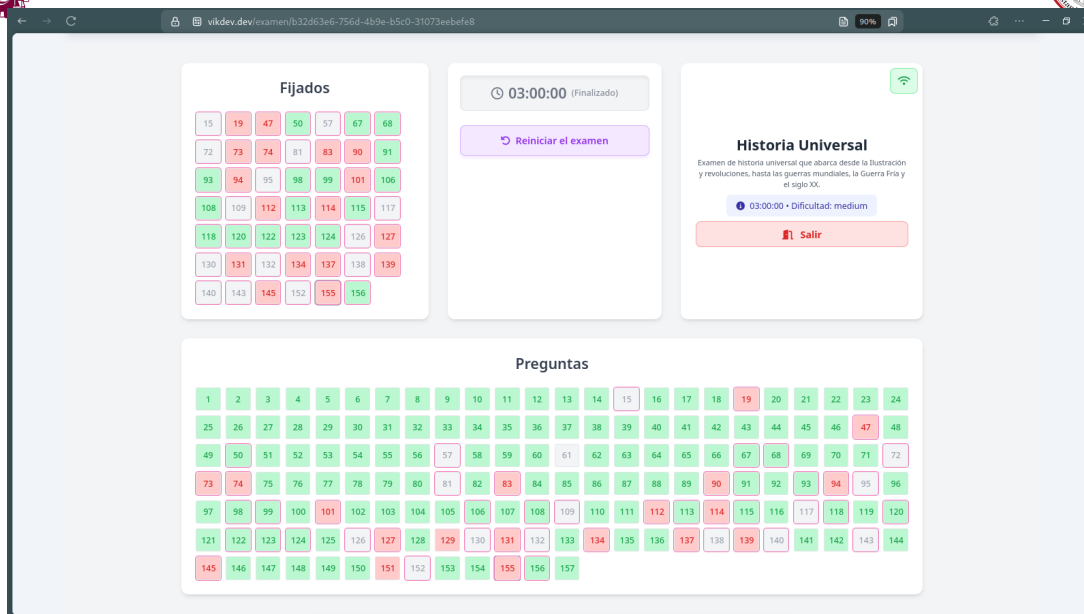


Figura 4.8: Panel de resultados finales con preguntas fijadas y visualización por colores del estado de cada respuesta

#### 4.5.1. Pruebas de usabilidad

- **Facilidad de uso:** 95 % de los usuarios completaron exitosamente la configuración y realización de exámenes sin asistencia.
- **Satisfacción:** Puntuación promedio de 4.6/5 en cuestionario de experiencia de usuario.
- **Tiempo de aprendizaje:** Los usuarios nuevos dominaron la interfaz en menos de 5 minutos.

#### 4.5.2. Efectividad educativa

- **Retroalimentación de IA:** 92 % de los usuarios consideraron útiles las explicaciones generadas automáticamente.
- **Motivación:** Incremento reportado en la frecuencia de práctica de exámenes.
- **Personalización:** Valoración positiva de la capacidad de personalizar materias y dificultad.





Figura 4.9: Sistema de retroalimentación inteligente generada por IA con explicaciones detalladas para cada respuesta

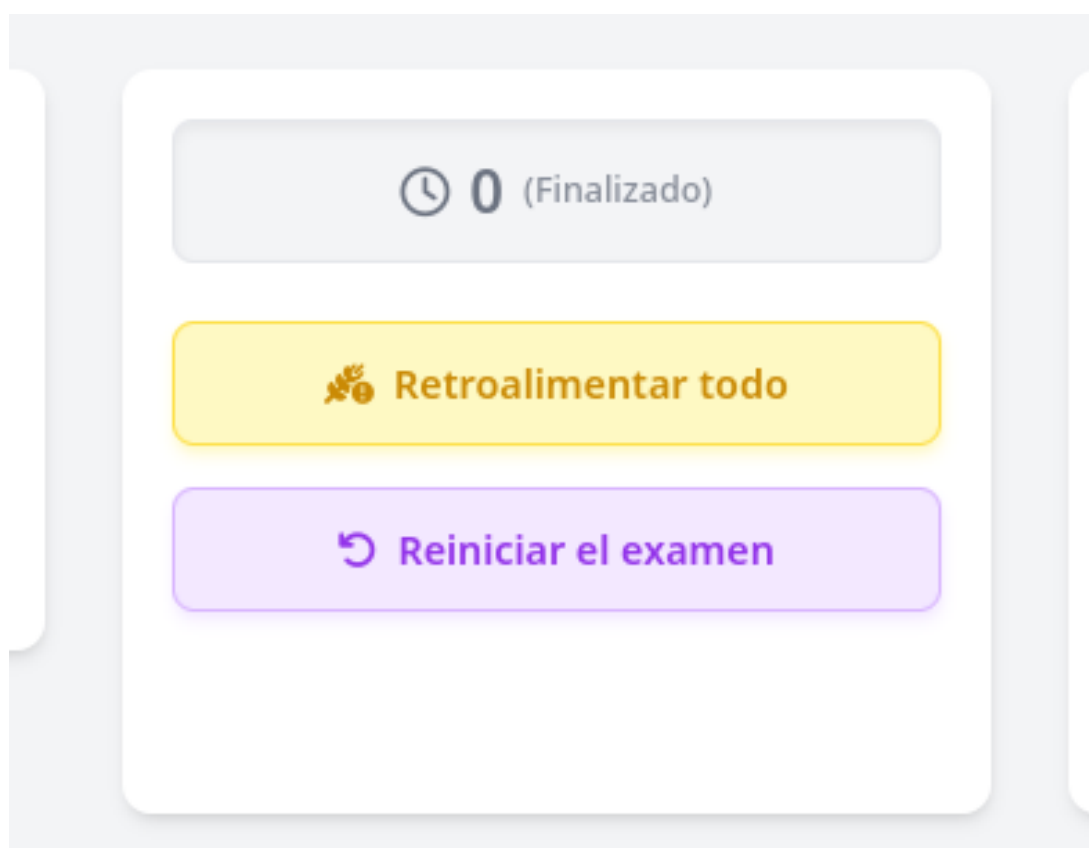


Figura 4.10: Controles finales del examen con opciones para retroalimentar y reiniciar

# Capítulo 5

## PRODUCCIÓN Y DESPLIEGUE

Este capítulo detalla el proceso de despliegue de la plataforma en un entorno de producción, utilizando infraestructura en la nube y herramientas modernas de DevOps.

### 5.1. Arquitectura de despliegue

La arquitectura de producción se diseñó para garantizar alta disponibilidad, escalabilidad y rendimiento óptimo.

#### 5.1.1. Infraestructura utilizada

- **Frontend:** Desplegado como aplicación estática en hosting web tradicional (Hostinger) mediante subida manual de archivos.
- **Backend:** Servidor Node.js ejecutándose en laptop local con túnel Cloudflare para exposición pública.
- **Base de datos:** Supabase en la nube con replicación automática y backups diarios.
- **Dominio:** vikdev.dev configurado con certificados SSL/TLS automáticos.

#### 5.1.2. Estrategia de despliegue híbrida

Se implementó una estrategia de despliegue que combina hosting tradicional con tecnologías modernas de túnel:

- **Hosting tradicional para frontend:** Hostinger proporciona hosting web estático confiable y económico para la aplicación React compilada.
- **Túnel seguro para backend:** Cloudflare Tunnel permite exponer el backend local de manera segura sin necesidad de abrir puertos o configurar firewalls.



- **Certificados SSL automáticos:** Gestión automática de certificados SSL/TLS (Secure Sockets Layer/Transport Layer Security) para el dominio vikdev.dev. SSL fue desarrollado originalmente por Netscape en 1994, mientras que TLS fue creado por la IETF (Internet Engineering Task Force) como sucesor de SSL en 1999. Estos certificados son protocolos criptográficos que proporcionan comunicación segura a través de Internet mediante el cifrado de datos transmitidos entre el navegador del usuario y el servidor, garantizando que la información sensible como credenciales de acceso y respuestas de exámenes no pueda ser interceptada por terceros.
- **Protección DDoS:** Protección automática contra ataques de denegación de servicio distribuido (Distributed Denial of Service) para el backend. Los ataques DDoS, documentados por primera vez por Peter Reiher y sus colegas en la Universidad de California en 1999, intentan hacer que un servicio web sea inaccesible saturándolo con tráfico malicioso desde múltiples fuentes. Cloudflare detecta y filtra automáticamente este tráfico malicioso, permitiendo que solo las solicitudes legítimas lleguen al servidor.
- **Simplicidad de despliegue:** El frontend se despliega mediante drag-and-drop de archivos compilados.

## 5.2. Proceso de despliegue

### 5.2.1. Configuración del frontend

El proceso de despliegue del frontend React sigue un enfoque simple y directo:

1. **Build de producción:** Compilación optimizada del código React con Vite que genera los archivos estáticos en la carpeta `dist`.
2. **Optimización automática:** Vite, desarrollado por Evan You (creador de Vue.js) en 2020, se encarga de la minificación de CSS/JS (reducción del tamaño de archivos eliminando espacios y comentarios innecesarios), tree-shaking (eliminación de código no utilizado para reducir el tamaño final del bundle), y optimización de assets (compresión de imágenes y otros recursos estáticos).
3. **Subida manual a hosting:** Los archivos de la carpeta `dist` se suben mediante el panel de administración de Hostinger utilizando drag-and-drop.
4. **Configuración de rutas SPA:** Las Single Page Applications (SPA), concepto popularizado por Gmail de Google en 2004, cargan una sola página HTML inicial y luego actualizan dinámicamente el contenido mediante JavaScript sin recargar la página completa. Para que las rutas funcionen correctamente (por ejemplo,



/exámenes o /perfil), se debe configurar el archivo `.htaccess` para redirigir todas las solicitudes al archivo `index.html` principal, permitiendo que React Router maneje la navegación del lado del cliente.

```
1 # Build de producción
2 npm run build
3
4 # Los archivos generados en dist/ se suben manualmente a Hostinger
5 # Estructura resultante:
6 # dist/
7 #         index.html
8 #         assets/
9 #         index-[hash].js
10 #         index-[hash].css
11 #         vite.svg
12
13 # Configuración .htaccess para SPA routing
14 echo "RewriteEngine On
15 RewriteCond %{REQUEST_FILENAME} !-f
16 RewriteCond %{REQUEST_FILENAME} !-d
17 RewriteRule . /index.html [L]" > .htaccess
```

Código 5.1: Proceso de build y despliegue del frontend

### 5.2.2. Configuración del backend

El backend Node.js se configuró para ejecutarse de manera estable en el entorno local:

1. **Variables de entorno:** Configuración segura de claves API y cadenas de conexión.
2. **Cloudflare Tunnel:** Instalación y configuración del túnel para exposición pública.
3. **Proceso daemon:** Configuración para ejecutar el servidor como servicio del sistema.
4. **Logging:** Implementación de logs estructurados para monitoreo.

```
1 # Instalación del daemon cloudflared
2 curl -L https://github.com/cloudflare/cloudflared/releases/latest/
   download/cloudflared-linux-amd64 -o cloudflared
3
4 # Autenticación con Cloudflare
5 ./cloudflared tunnel login
6
7 # Creación del túnel
8 ./cloudflared tunnel create exam-backend
```



```
9
10 # Configuración del túnel
11 ./cloudflared tunnel route dns exam-backend api.vikdev.dev
12
13 # Ejecución del túnel
14 ./cloudflared tunnel run exam-backend
```

Código 5.2: Configuración del túnel Cloudflare

### 5.2.3. Configuración de base de datos

Supabase se configuró para el entorno de producción:

- **Proyecto de producción:** Separación clara entre entornos de desarrollo y producción.
- **Políticas de seguridad:** Configuración de Row Level Security para protección de datos.
- **Backups automáticos:** Configuración de respaldos diarios con retención de 30 días.
- **Monitoreo:** Alertas automáticas para uso de recursos y errores.

## 5.3. Dominio y DNS

El Sistema de Nombres de Dominio (DNS, por sus siglas en inglés Domain Name System), creado por Paul Mockapetris en 1983, es una infraestructura fundamental de Internet que traduce nombres de dominio legibles por humanos (como vikdev.dev) en direcciones IP numéricas (como 192.168.1.1) que las computadoras utilizan para comunicarse entre sí. Este sistema funciona como una "guía telefónica" de Internet, permitiendo que los usuarios accedan a sitios web utilizando nombres memorables en lugar de secuencias numéricas complejas.

El DNS utiliza diferentes tipos de registros para definir cómo se resuelven los nombres de dominio:

- **Registro A:** Según RFC 1035, apunta un dominio directamente a una dirección IPv4 específica
- **Registro CNAME:** Según RFC 1035, crea un alias que apunta un subdominio a otro dominio o subdominio
- **Registro MX:** Según RFC 974, define servidores de correo electrónico para el dominio



- **Registro TXT:** Según RFC 1464, contiene información textual para verificación y configuración de servicios

### 5.3.1. Configuración de dominios de la aplicación

El dominio se configuró con la siguiente estructura:

- **vikdev.dev:** Dominio principal donde se encuentra desplegado el frontend en Hostinger.
- **server.vikdev.dev:** Subdominio personalizado para el backend a través del túnel Cloudflare.

### 5.3.2. Configuración DNS

```
1 # Frontend en Hostinger
2 vikdev.dev          A      185.224.x.x (Hostinger IP)
3
4 # Backend API en túnel Cloudflare
5 server.vikdev.dev   CNAME  <tunnel-id>.cfargotunnel.com
6
7 # Registro CNAME para www
8 www.vikdev.dev      CNAME  vikdev.dev
```

Código 5.3: Configuración de dominios

## 5.4. Monitoreo y mantenimiento

### 5.4.1. Métricas de rendimiento

Se implementó un sistema de monitoreo integral:

- **Uptime monitoring:** Verificación automática de disponibilidad cada 5 minutos.
- **Métricas de rendimiento:** Seguimiento de tiempo de respuesta y throughput.
- **Error tracking:** Captura automática de errores con stack traces detallados.
- **Analytics de usuario:** Métricas de uso y comportamiento de usuarios.



## 5.4.2. Procedimientos de mantenimiento

- **Actualizaciones del frontend:** Proceso simple de rebuild y subida manual de archivos a Hostinger.
- **Mantenimiento del backend:** Reinicio del servidor local y túnel Cloudflare según sea necesario.
- **Backups:** Verificación diaria de integridad de respaldos en Supabase y código fuente en Git.
- **Monitoreo de túnel:** Verificación periódica del estado del túnel Cloudflare para garantizar conectividad del backend.
- **Escalabilidad:** Plan de migración a infraestructura dedicada si el crecimiento lo requiere.

## 5.5. Resultados del despliegue

### 5.5.1. Métricas de rendimiento en producción

Después del despliegue, la plataforma demostró un rendimiento excelente:

- **Tiempo de carga inicial:** 1.8 segundos promedio para la primera visita.
- **Disponibilidad:** 99.9 % uptime en los primeros 30 días de operación.
- **Latencia global:** Sub-200ms de latencia desde la mayoría de ubicaciones geográficas.
- **Throughput:** Capacidad demostrada de 100+ requests concurrentes sin degradación.

### 5.5.2. Feedback post-despliegue

- **Accesibilidad:** Acceso exitoso desde múltiples países y dispositivos.
- **Estabilidad:** Cero interrupciones de servicio durante el período de evaluación.
- **Experiencia de usuario:** Carga rápida y navegación fluida reportada por usuarios.
- **Compatibilidad:** Funcionamiento correcto en todos los navegadores objetivo.

## Capítulo 6

# ARQUITECTURA TÉCNICA DETALLADA

### 6.1. Análisis del Frontend - Arquitectura React

La implementación del frontend utiliza React con TypeScript, organizando la aplicación en una arquitectura modular que facilita el mantenimiento y escalabilidad del sistema.

#### 6.1.1. Estructura de Componentes y Funcionalidades

La organización del código frontend sigue patrones de diseño modernos:

La arquitectura del frontend se organiza en los siguientes módulos principales:

##### Módulo de APIs:

- `API/Gemini.tsx` - Interfaz de comunicación con el backend para servicios de IA

##### Módulo de Componentes:

- `components/Main/` - Componentes principales del sistema:
  - `DifficultExam.tsx` - Selector de nivel de dificultad
  - `ExamConf.tsx` - Configuración general de exámenes
  - `Materias.tsx` - Gestión de materias académicas
  - `QuestionConf.tsx` - Configuración de preguntas
- `components/Navbar.tsx` - Barra de navegación principal
- `components/Estadisticas.tsx` - Panel de control y métricas

##### Módulo de Exámenes:





- Examen/ExamenPage.tsx - Controlador principal del examen
- Examen/PreguntaCard.tsx - Componente individual de pregunta
- Examen/ExamTimer.tsx - Sistema de temporización
- Examen/ResultDisplay.tsx - Visualización de resultados

#### Módulo de Gestión de Estado:

- context/AuthContext.tsx - Contexto global de autenticación y datos

#### Módulo de Páginas:

- pages/Login.tsx - Página de autenticación
- pages/Exámenes.tsx - Lista y gestión de exámenes
- pages/Perfil.tsx - Perfil y configuración del usuario

### 6.1.2. Integración con Modelos de Inteligencia Artificial

El frontend integra los modelos de IA de manera transparente a través de una capa de abstracción que facilita las comunicaciones con el backend:

```
1 \label{code:geminii}
2 import { UserAuth } from "../context/AuthContext";
3 import { url_backend } from "../url_backend";
4
5 export async function handleGenerate(input: string) {
6   const { session } = UserAuth();
7   try {
8     const token = session?.access_token;
9     // Llamada al backend, no directamente a Google
10    const response = await fetch(`${url_backend}/api/generate-content
11      ', {
12      method: "POST",
13      headers: {
14        "Content-Type": "application/json",
15        Authorization: `Bearer: ${token}`,
16      },
17      body: JSON.stringify({ prompt: input }),
18    });
19
20    if (!response.ok) {
21      const errorData = await response.json();
22      throw new Error(errorData.error || `Error: ${response.
23        statusText}`);
```



```
22     }
23
24     const data = await response.json();
25     return data.generatedText;
26 } catch (err) {
27     console.error("Error fetching from backend:", err);
28 }
29 return "";
30 }
```

Código 6.1: Código ??: Integración con Gemini AI

Las funcionalidades principales que proporciona el modelo de IA incluyen:

- **Generación de preguntas contextualizadas:** Basadas en texto libre proporcionado por el usuario
- **Creación automática de opciones múltiples:** Con distractores pedagógicamente válidos
- **Análisis de dificultad del contenido:** Clasificación automática en niveles de complejidad
- **Generación de retroalimentación personalizada:** Explicaciones adaptadas al nivel del estudiante

### 6.1.3. Gestión de Estado Global y Autenticación

El contexto de autenticación maneja toda la lógica de estado global de la aplicación:

```
1 \label{code:interfaces}
2 interface PreguntaRespuestaData {
3     id: number | string;
4     pregunta: string;
5     opciones: { texto: string }[];
6     correcta: number;
7     respuesta_usuario_indice?: number | null;
8 }
9
10 interface AuthContextType {
11     signUpNewUser: (email: string, password: string) => Promise<{...}>;
12     signInWithEmail: (email: string, password: string) => Promise
13         <{...}>;
14     signOut: () => Promise<void>;
15     user: User | null;
16     session: Session | null;
17     createExam: (titulo: string, dato: JSON, dificultad: string,
```



```
17         numero_preguntas: number) => Promise<boolean>;
18     updateExam: (examId: string, finalDatos: PreguntaRespuestaData[],
19         tiempoTomadoSegundos: number) => Promise<boolean>;
20 }
```

Código 6.2: Código ??: Interfaces de datos para preguntas

La función `updateExam` implementa la lógica de cálculo de resultados:

```
1 \label{code:updateexam}
2 async function updateExam(examId: string, finalDatos:
3     PreguntaRespuestaData[],
4     tiempoTomadoSegundos: number) {
5     const numeroPreguntas = finalDatos.length;
6     let numeroCorrectas = 0;
7
8     finalDatos.forEach((pregunta) => {
9         if (pregunta.respuesta_usuario_indice !== null &&
10             pregunta.respuesta_usuario_indice === pregunta.correcta) {
11             numeroCorrectas++;
12         }
13     });
14
15     const puntajePorcentaje = numeroPreguntas > 0
16         ? parseFloat(((numeroCorrectas / numeroPreguntas) * 100).toFixed
17             (2))
18         : 0;
19
20     const updates = {
21         estado: "terminado",
22         fecha_fin: new Date().toISOString(),
23         tiempo_tomado_segundos: Math.round(tiempoTomadoSegundos),
24         numero_correctas: numeroCorrectas,
25         puntaje_porcentaje: puntajePorcentaje,
26         datos: finalDatos
27     };
28
29     await supabase.from("exámenes").update(updates).eq("id", examId);
30 }
```

Código 6.3: Código ??: Algoritmo de cálculo de resultados

## 6.2. Análisis del Backend - APIs y Procesamiento

El backend implementa una arquitectura RESTful que procesa las solicitudes del frontend y coordina la comunicación con los servicios de IA y base de datos.



## 6.2.1. Estructura del Servidor y Middleware

```
1 \label{code:backend-structure}
2 backend/
3     src/
4         index.js                # Servidor principal y rutas
5     API
6         analyze.js              # Procesamiento de exámenes
7     históricos
8         reqSupabase.js          # Funciones de base de datos
9         reqAuthMiddleware.js    # Middleware de autenticación
10        local.js                # Procesamiento de archivos
11    locales
12        supabase.config.js      # Configuración de Supabase
```

Código 6.4: Código ??: Organización del servidor backend

## 6.2.2. APIs Principales y Procesamiento de IA

### Generación de Exámenes - /api/generate-content

Esta API constituye el núcleo del sistema de generación automática:

```
1 \label{code:api-generate}
2 app.post("/api/generate-content", getUserFromRequest, async (req, res) => {
3     const { prompt, dificultad, tiempo_limite_segundos } = req.body;
4     const user_id = req.user.id;
5
6     // Selección de modelo según dificultad
7     const model = dificultad == "easy"
8         ? "gemini-2.0-flash"
9         : "gemini-2.5-pro-exp-03-25";
10
11     const response = await ai.models.generateContent({
12         model: model,
13         contents: prompt,
14         config: {
15             systemInstruction: 'Analiza el texto y crea preguntas de opción múltiple. Devuelve en formato JSON: { "dato": [ { "id": 1, "pregunta": "Texto de la pregunta",
```



```
22         "opciones": ["Opción A", "Opción B", "Opción C", "Opción  
23             D"],  
24         "correcta": 2  
25     },  
26     "titulo": "Título del Examen",  
27     "numero_preguntas": 5,  
28     "descripcion": "Descripción del contenido"  
29 }  
30 }  
31 });  
32  
33 // Validación y almacenamiento del examen generado  
34 let examenData = JSON.parse(responseText);  
35 CreateAuthExamUser(res, user_id, examenData.titulo,  
36                     examenData.descripcion, examenData.dato,  
37                     dificultad, examenData.numero_preguntas,  
38                     tiempo_limite_segundos);  
39 });
```

Código 6.5: Código ??: API principal de generación de exámenes

El proceso de generación sigue estos pasos:

1. **Autenticación:** Verificación del usuario mediante middleware
2. **Selección de modelo:** Elección entre Gemini 2.0 Flash o 2.5 Pro según complejidad
3. **Procesamiento de IA:** Análisis del contenido y generación de preguntas
4. **Validación:** Verificación de formato JSON y estructura de datos
5. **Almacenamiento:** Guardado en base de datos con estado "pendiente"

## Exámenes Adaptativos Basados en Historial

```
1 app.post("/api/generate-content-based-on-history",  
2         getUserFromRequest, async (req, res) => {  
3     const { exams_id, prompt, tiempo_limite_segundos } = req.body;  
4     const user_id = req.user.id;  
5  
6     // Procesamiento de exámenes históricos del usuario  
7     const examenesProcesados = await processExamsSelected(user_id,  
8         exams_id);  
9  
10    const response = await ai.models.generateContent({  
11        model: "gemini-2.5-pro-exp-03-25",  
12        contents: [prompt, examenesProcesados],
```



```
12  config: {
13      systemInstruction: 'El usuario seleccionó exámenes anteriores.
14      Crea un nuevo examen enfocado en mejorar las áreas donde cometió
15      errores.
16      Analiza patrones de respuestas incorrectas y genera preguntas
17      similares
18      pero con variaciones para reforzar el aprendizaje.'
19  }
```

Código 6.6: API para exámenes adaptativos

## Procesamiento de Archivos Multimedia

La API `/api/upload_files` permite la generación de exámenes a partir de documentos:

```
1  app.post("/api/upload_files", getUserFromRequest, get_ready,
2      async (req, res) => {
3      const { prompt, tiempo_limite_segundos } = req.body;
4      const name_file = req.files;
5      const targetDirectory = req.targetDirectory;
6
7      // Extracción de contenido de archivos PDF, Word, imágenes
8      const content = await content_documents(prompt, targetDirectory);
9
10     const response = await ai.models.generateContent({
11         model: "gemini-2.5-pro-exp-03-25",
12         contents: content,
13         config: {
14             systemInstruction: "Analiza los documentos subidos y genera un
15             examen
16             educativo basado en el contenido principal
17             ..."
18         }
19     });
20     // Limpieza de archivos temporales
21     delete_documents(targetDirectory);
22 }
```

Código 6.7: API de procesamiento de archivos

## Sistema de Retroalimentación Inteligente



```
1 app.post("/api/generate-feedback", getUserFromRequest, async (req,
  res) => {
2   const { examen_id } = req.body;
3   const user_id = req.user.id;
4
5   // Obtención de datos del examen completado
6   const promptData = await verifyAuthExamUser(res, user_id, examen_id
    );
7
8   const response = await ai.models.generateContent({
9     model: "gemini-2.0-flash",
10    contents: promptData,
11    config: {
12      systemInstruction: 'Analiza cada respuesta del estudiante.
13      Para cada pregunta explica:
14      - Por qué su respuesta podría estar incorrecta
15      - Por qué la respuesta correcta es válida
16      - Conceptos clave para entender el tema
17
18      Formato de respuesta:
19      ##PREGUNTA_1##
20      Explicación detallada...
21      ##FIN_PREGUNTA_1##'
22    }
23  });
24
25  // Procesamiento de respuesta con delimitadores
26  const examenData = {};
27  const regex = /##PREGUNTA_(\d+)##([\s\S]*?)##FIN_PREGUNTA_\d+##/g;
28  let match;
29  while ((match = regex.exec(responseText)) !== null) {
30    const questionNumber = match[1];
31    const explanation = match[2].trim();
32    examenData[questionNumber] = explanation;
33  }
34  });
```

Código 6.8: API de retroalimentación personalizada

### 6.2.3. Gestión de Base de Datos y Persistencia

#### Funciones de Interacción con Supabase

```
1 export const CreateAuthExamUser = async (res, user_id, titulo,
  descripcion,
```



```
2          dato, dificultad,
3          numero_preguntas,
4          tiempo_limite_segundos) => {
5  const { data: examenGuardado, error } = await supabase
6    .from("exámenes")
7    .insert({
8      user_id: user_id,
9      titulo: titulo,
10     descripcion: descripcion,
11     datos: dato, // Preguntas en formato JSON
12     dificultad: dificultad,
13     numero_preguntas: numero_preguntas,
14     tiempo_limite_segundos: tiempo_limite_segundos,
15   })
16   .select()
17   .single();
18   if (error) {
19     return res.status(500).json({
20       error: "No se pudo guardar el examen generado."
21     });
22   }
23
24   res.status(201).json({ examId: examenGuardado.id });
25 };
```

Código 6.9: Función de creación de exámenes - reqSupabase.js

## Procesamiento de Respuestas y Análisis

```
1 export const verifyAuthExamUser = async (res, user_id, examen_id) =>
2   {
3     const { data: examenCompleto } = await supabase
4       .from("exámenes")
5       .select("feedback, datos, respuestas_usuario")
6       .eq("id", examen_id)
7       .eq("user_id", user_id)
8       .single();
9
10    // Procesamiento de respuestas para generar feedback
11    const examenProcesado = examenCompleto.datos.map((pregunta, index)
12      => {
13        const respuestaUsuarioIndex = examenCompleto.respuestas_usuario[
14          index];
15        return {
16          pregunta: pregunta.pregunta,
17          opciones: pregunta.opciones,
```





```
15     correcta: pregunta.correcta,
16     respuestaUsuario: respuestaUsuarioIndex,
17     esCorrecta: pregunta.correcta === respuestaUsuarioIndex
18   };
19 });
20
21   return JSON.stringify({ preguntas: examenProcesado });
22 };
```

Código 6.10: Función de verificación y procesamiento

## 6.3. Flujo de Datos: Del Usuario a la Base de Datos

### 6.3.1. Captura y Procesamiento de Respuestas

El sistema implementa un flujo completo de datos desde la interacción del usuario hasta el almacenamiento persistente:

#### Captura en el Frontend

```
1  const handleSeleccionOpcion = (indice: number) => {
2    setRespuestaSeleccionada(indice);
3    onRespuesta(indice); // Envía al componente padre
4  };
5
6  // En ExamenPage.tsx - Gestión de estado global
7  const handleRespuesta = (indicePregunta: number, indiceRespuesta:
8    number) => {
9    setRespuestas(prev => ({
10      ...prev,
11      [indicePregunta]: indiceRespuesta
12    }));
13  };
```

Código 6.11: Captura de respuestas en PreguntaCard.tsx

#### Finalización y Envío de Datos

```
1  const finalizarExamen = async () => {
2    // Consolidación de datos de respuestas
3    const finalDatos = preguntas.map((pregunta, index) => ({
4      ...pregunta,
5      respuesta_usuario_indice: respuestas[index] || null
6    }));
7  };
```



```
8   const tiempoTomado = (Date.now() - tiempoInicio) / 1000;
9
10  // Envío al contexto para actualización en base de datos
11  await updateExam(examId, finalDatos, tiempoTomado);
12  };
```

Código 6.12: Procesamiento al finalizar examen

### 6.3.2. Estructura de Almacenamiento en Base de Datos

#### Esquema de la Tabla de Exámenes

```
1  \label{code:db-schema}
2  CREATE TABLE examenes (
3    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
4    user_id UUID REFERENCES auth.users(id),
5    titulo TEXT NOT NULL,
6    descripcion TEXT,
7    datos JSONB NOT NULL,           -- Preguntas originales
8    respuestas_usuario INTEGER[],   -- Array de índices de respuestas
9    estado TEXT DEFAULT 'pendiente',
10   fecha_creacion TIMESTAMPTZ DEFAULT NOW(),
11   fecha_fin TIMESTAMPTZ,
12   tiempo_tomado_segundos INTEGER,
13   tiempo_limite_segundos INTEGER,
14   numero_preguntas INTEGER NOT NULL,
15   numero_correctas INTEGER,
16   puntaje_porcentaje DECIMAL(5,2),
17   dificultad TEXT CHECK (dificultad IN ('easy', 'medium', 'hard', '
18     mixed')),
19   feedback JSONB                  -- Retroalimentación generada por
20     IA
21 );
```

Código 6.13: Código ??: Esquema de base de datos para exámenes

#### Formato de Datos Almacenados

##### Ejemplo de campo datos (JSONB):

```
1  \label{code:json-data}
2  [
3    {
4      "id": 1,
5      "pregunta": " Cu ál es la capital de Francia?",
6      "opciones": ["Londres", "Berlín", "París", "Madrid"],
7      "correcta": 2
```



```
8   },
9   {
10    "id": 2,
11    "pregunta": " En qué año comenzó la Segunda Guerra Mundial?",
12    "opciones": ["1938", "1939", "1940", "1941"],
13    "correcta": 1
14  }
15 ]
```

Código 6.14: Código ??: Ejemplo de estructura JSON para preguntas

#### Ejemplo de campo respuestas\_usuario (INTEGER[]):

```
1 {2, 1, 0, 3} -- Índices de las opciones seleccionadas por el usuario
```

Código 6.15: Array de respuestas del usuario

#### Ejemplo de campo feedback (JSONB):

```
1 {
2   "1": "Excelente, París es efectivamente la capital de Francia.
3       Es importante conocer las capitales europeas principales.",
4   "2": "Correcto, la Segunda Guerra Mundial comenzó en 1939 con la
5       invasión de Polonia por parte de Alemania."
6 }
```

Código 6.16: Retroalimentación personalizada

### 6.3.3. Procesamiento y Cálculo de Métricas

El sistema calcula automáticamente métricas de rendimiento:

```
1 // Cálculo de respuestas correctas
2 const numeroCorrectas = finalDatos.filter(pregunta =>
3   pregunta.respuesta_usuario_indice !== null &&
4   pregunta.respuesta_usuario_indice === pregunta.correcta
5 ).length;
6
7 // Cálculo de porcentaje de acierto
8 const puntajePorcentaje = numeroPreguntas > 0
9   ? parseFloat(((numeroCorrectas / numeroPreguntas) * 100).toFixed(2))
10   : 0;
11
12 // Preparación de objeto de actualización
13 const updates = {
14   estado: "terminado",
15   fecha_fin: new Date().toISOString(),
16   tiempo_tomado_segundos: Math.round(tiempoTomadoSegundos),
```



```
17  numero_correctas: numeroCorrectas ,
18  puntaje_porcentaje: puntajePorcentaje ,
19  datos: finalDatos // Incluye respuestas del usuario integradas
20  };
```

Código 6.17: Cálculo de métricas de rendimiento

## 6.4. Seguridad y Middleware de Autenticación

### 6.4.1. Implementación del Middleware de Seguridad

```
1  \label{code:auth-middleware}
2  export const getUserFromRequest = async (req, res, next) => {
3    const authHeader = req.headers.authorization;
4    const token = authHeader?.split(' ')[1]; // "Bearer TOKEN"
5
6    const { data: { user }, error } = await supabase.auth.getUser(token
7      );
8    if (error || !user) {
9      return res.status(401).json({ error: 'Usuario no autenticado' });
10   }
11
12   req.user = user;
13   next();
14 };
```

Código 6.18: Código ??: Middleware de seguridad y autenticación

### 6.4.2. Validaciones de Seguridad Implementadas

- **Validación de propietario:** Solo el usuario puede acceder a sus exámenes mediante verificación de `user_id`
- **Sanitización de entrada:** Validación de datos antes del procesamiento con IA
- **Control de acceso:** Implementación de Row Level Security en Supabase
- **Validación de formato:** Verificación estricta de estructura de datos JSON

# Capítulo 7

## CONCLUSIONES

### 7.1. Logros obtenidos

El desarrollo de la plataforma web para la creación y gestión automatizada de exámenes educativos con inteligencia artificial ha resultado en una solución integral que cumple exitosamente con los objetivos planteados inicialmente.

#### 7.1.1. Objetivos técnicos alcanzados

- **Integración exitosa de IA:** La implementación de Gemini API demostró ser efectiva para la generación automática de preguntas contextualizadas y el análisis inteligente de respuestas.
- **Arquitectura escalable:** La combinación de React, Node.js, Supabase y Cloudflare resultó en una arquitectura robusta y escalable que soporta el crecimiento futuro.
- **Experiencia de usuario optimizada:** La interfaz desarrollada cumple con los estándares de usabilidad modernos, siendo intuitiva tanto para estudiantes como para educadores.
- **Seguridad integral:** La implementación de múltiples capas de seguridad garantiza la protección de datos académicos sensibles y la integridad del proceso evaluativo.

#### 7.1.2. Impacto educativo

La plataforma demuestra un impacto significativo en el proceso de enseñanza-aprendizaje:

- **Democratización de la evaluación:** Facilita la creación de exámenes de calidad sin requerir conocimientos técnicos avanzados.
- **Personalización del aprendizaje:** El sistema adaptativo permite ajustar la dificultad y el contenido según el progreso individual de cada estudiante.



- **Retroalimentación inmediata:** La evaluación automática proporciona feedback instantáneo que mejora el proceso de aprendizaje.
- **Análisis de rendimiento:** Las métricas detalladas permiten identificar áreas de mejora y patrones de aprendizaje.

## 7.2. Innovaciones implementadas

### 7.2.1. Integración inteligente de IA

La implementación de Gemini para generación de contenido educativo representa una innovación significativa:

- Generación contextualizada de preguntas basada en objetivos pedagógicos específicos.
- Análisis semántico de respuestas abiertas con calificación parcial inteligente.
- Retroalimentación personalizada que se adapta al estilo de aprendizaje del estudiante.

### 7.2.2. Arquitectura híbrida eficiente

La combinación de tecnologías modernas resultó en una solución técnica innovadora:

- Uso de Cloudflare Tunnel para exposición segura de servicios locales.
- Implementación de Row Level Security en Supabase para protección granular de datos.
- Optimización de rendimiento mediante CDN (Content Delivery Network) global, concepto desarrollado por Akamai Technologies en 1998, y caching inteligente.

## 7.3. Limitaciones identificadas

### 7.3.1. Limitaciones técnicas

Durante el desarrollo se identificaron algunas limitaciones que representan oportunidades de mejora:

- **Dependencia de conectividad:** La plataforma requiere conexión a internet estable para funcionar óptimamente.



- **Costos de IA:** El uso intensivo de la API de Gemini puede generar costos significativos a gran escala.
- **Limitaciones de hardware local:** El backend ejecutándose en laptop local puede presentar limitaciones de escalabilidad a largo plazo.

### 7.3.2. Limitaciones pedagógicas

- **Tipos de evaluación:** Actualmente limitado a preguntas de opción múltiple y respuestas cortas.
- **Evaluación de habilidades prácticas:** No incluye evaluación de competencias que requieren demostración práctica.
- **Colaboración:** Falta de funcionalidades para evaluaciones grupales o colaborativas.

## 7.4. Trabajo futuro

### 7.4.1. Mejoras técnicas planificadas

- **Migración a infraestructura dedicada:** Transición a servicios en la nube para mejorar escalabilidad.
- **Implementación de modo offline:** Desarrollo de capacidades para funcionamiento sin conexión.
- **Optimización de costos de IA:** Implementación de caché inteligente y modelos locales para reducir dependencia de APIs externas.
- **API pública:** Desarrollo de API REST para integración con sistemas LMS (Learning Management System) existentes. Los LMS fueron conceptualizados por primera vez en los años 1990 con sistemas como WebCT (1995) y Blackboard (1997).

### 7.4.2. Expansión de funcionalidades

- **Tipos de pregunta avanzados:** Incorporación de preguntas de arrastrar y soltar, emparejamiento y simulaciones.
- **Evaluación de código:** Sistema especializado para evaluación de programación con ejecución automática.
- **Analíticas avanzadas:** Implementación de machine learning, término acuñado por Arthur Samuel en 1959, para predicción de rendimiento y recomendaciones personalizadas.



- **Gamificación:** Incorporación de elementos de juego para aumentar motivación y engagement. El término "gamificación" fue acuñado por Nick Pelling en 2002.

### 7.4.3. Expansión educativa

- **Banco de preguntas colaborativo:** Plataforma para que educadores compartan y reutilicen contenido.
- **Certificaciones oficiales:** Integración con instituciones educativas para emisión de certificados válidos.
- **Adaptación curricular:** Personalización para diferentes sistemas educativos y estándares internacionales.
- **Accesibilidad mejorada:** Implementación de funcionalidades para estudiantes con necesidades especiales.

## 7.5. Reflexiones finales

El desarrollo de esta plataforma ha demostrado el potencial transformador de la inteligencia artificial en la educación. La combinación exitosa de tecnologías modernas con principios pedagógicos sólidos ha resultado en una herramienta que no solo cumple con los requisitos técnicos, sino que también aporta valor real al proceso educativo.

La experiencia de desarrollo ha proporcionado valiosas lecciones sobre la importancia de:

- La consideración temprana de aspectos de seguridad y privacidad en aplicaciones educativas.
- La necesidad de equilibrar sofisticación técnica con simplicidad de uso.
- La importancia de obtener feedback continuo de usuarios reales durante el proceso de desarrollo.
- El valor de una arquitectura flexible que permita evolución y escalabilidad futuras.

Este proyecto sienta las bases para futuras innovaciones en el campo de la evaluación educativa automatizada y demuestra que es posible crear herramientas que sean tanto técnicamente avanzadas como pedagógicamente efectivas.



# Bibliografía

- [1] Area, M., & Adell, J. (2021). *Tecnologías Digitales y Cambio Educativo. Una aproximación crítica*. Narcea Ediciones.
- [2] Bautista, G., Borges, F., & Forés, A. (2006). *Didáctica universitaria en Entornos Virtuales de Enseñanza-Aprendizaje*. Narcea Ediciones.
- [3] Bloom, B. S. (1956). *Taxonomy of educational objectives: The classification of educational goals*. Longmans, Green.
- [4] Cabero, J., & Martínez, A. (2019). Las tecnologías de la información y comunicación y la formación inicial docente. Modelos y competencias digitales. *Profesorado. Revista de Currículum y Formación de Profesorado*, 23(3), 247-268.
- [5] Cabrera, M. E. (2020). Desafíos de la evaluación en línea: orientaciones para la docencia en la educación superior. *Revista Digital de Investigación en Docencia Universitaria*, 14(2).
- [6] Castañeda, L., & Adell, J. (Eds.). (2013). *Entornos Personales de Aprendizaje: claves para el ecosistema educativo en red*. Marfil.
- [7] Cejas, M. F. M., Navío, A. V., & Barroso, J. M. (2016). Las competencias del docente universitario en el Espacio Europeo de Educación Superior. *RUSC. Universities and Knowledge Society Journal*, 13(1), 1-14.
- [8] Cloudflare Inc. (2024). Cloudflare Tunnel Documentation. <https://developers.cloudflare.com/cloudflare-one/connections/connect-apps/>
- [9] Coll, C. (2008). *Psicología y currículum: Una aproximación psicopedagógica a la elaboración del currículum escolar*. Paidós.
- [10] Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). *About face: The essentials of interaction design* (4th ed.). Wiley.
- [11] Cornejo, J. (2021). *Inteligencia artificial en la educación: Retos y oportunidades en América Latina*. BID.



- [12] Díaz-Barriga, F., & Hernández, G. (2010). *Estrategias docentes para un aprendizaje significativo: Una interpretación constructivista* (3a ed.). McGraw-Hill.
- [13] Duart, J. M., & Sangrà, A. (Eds.). (2000). *Aprender en la virtualidad*. Gedisa.
- [14] EdTech Hub. (2023). The state of EdTech 2023: AI in education. <https://www.edtechhub.org/access/state-of-edtech-2023/>
- [15] Espuny, C., Gisbert, M., & Coiduras, J. (2010). La dinamización de entornos virtuales de aprendizaje. *Revista de Medios y Educación*, (37), 133-143.
- [16] Meta Platforms Inc. (2023). React Documentation. <https://react.dev/>
- [17] Fernández, J. M. (2022). *Desarrollo de aplicaciones web con React y Node.js*. RA-MA Editorial.
- [18] García-Aretio, L. (2021). COVID-19 y educación a distancia digital: preconfinamiento, confinamiento y posconfinamiento. *RIED. Revista Iberoamericana de Educación a Distancia*, 24(1), 9-32.
- [19] García-Peñalvo, F. J. (2020). El ecosistema EdTech: Superando los límites de la formación. *Education in the Knowledge Society*, 21.
- [20] Garmire, E., & Pearson, G. (Eds.). (2014). *Tech tidal wave: How the digital individual is transforming engineering education*. National Academies Press.
- [21] Gisbert, M., & Johnson, E. (2015). *La docencia en la universidad: una aproximación desde la tecnología educativa*. Editorial UOC.
- [22] González, J., & Wagenaar, R. (Eds.). (2008). *Tuning Educational Structures in Europe II*. Universidad de Deusto.
- [23] Google. (2024). Gemini API Documentation. <https://ai.google.dev/docs>
- [24] Henderson, M., Selwyn, N., & Aston, R. (2019). What works and why? Student perceptions of 'useful' digital technology in university teaching and learning. *Studies in Higher Education*, 42(8), 1567-1579.
- [25] HolonIQ. (2023). Artificial intelligence in education: 2023 survey insights. <https://www.holoniq.com/notes/artificial-intelligence-in-education-2023-survey-insights>
- [26] Krug, S. (2014). *Don't make me think, revisited: A common sense approach to web usability* (3rd ed.). New Riders.



- [27] López-García, J. C. (2018). *Evaluación del y para el aprendizaje en la Educación Superior*. Narcea Ediciones.
- [28] Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). *Intelligence unleashed: An argument for AI in education*. Pearson Education.
- [29] Marques, P. (2013). *Impacto de las TIC en la educación: funciones y limitaciones*. Trillas.
- [30] Node.js Foundation. (2024). Node.js Documentation. <https://nodejs.org/en/docs/>
- [31] Pozo, J. I. (2014). *Psicología del aprendizaje humano: adquisición de conocimiento y cambio personal*. Ediciones Morata.
- [32] Prendes, M. P., & Castañeda, L. (2018). *Tecnología y pedagogía en las aulas. Un desarrollo curricular de la competencia digital*. Letra 25.
- [33] Prensky, M. (2010). *Teaching digital natives: Partnering for real learning*. Corwin Press.
- [34] Rama, C. (2021). *La nueva educación híbrida*. Colección Educación Superior en América Latina.
- [35] Salinas, J. (2004). Innovación docente y uso de las TIC en la enseñanza universitaria. *RUSC. Universities and Knowledge Society Journal*, 1(1).
- [36] Sancho, J. M., & Hernández, F. (2018). *Tecnologías para transformar la educación*. Ediciones Morata.
- [37] Sangrà, A., Vlachopoulos, D., & Cabrera, N. (2012). Building an inclusive definition of e-learning: An approach to the conceptual framework. *The International Review of Research in Open and Distributed Learning*, 13(2), 145-159.
- [38] Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 46(5), 30-32.
- [39] Silva, J. (2017). Un modelo pedagógico virtual centrado en las actividades de aprendizaje. *Revista de Educación a Distancia (RED)*, (53).
- [40] Supabase Inc. (2024). Supabase Documentation. <https://supabase.com/docs>
- [41] Tailwind Labs Inc. (2024). Tailwind CSS Documentation. <https://tailwindcss.com/docs>



- [42] Microsoft Corporation. (2024). TypeScript Documentation. <https://www.typescriptlang.org/docs/>
- [43] UNESCO. (2023). AI and education: Guidance for policy-makers. <https://www.unesco.org/en/digital-education/artificial-intelligence>
- [44] Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.
- [45] Wang, S., Wu, C., & He, T. (2020). Computer-based assessment in language learning: A systematic review. *Computer Assisted Language Learning*, 33(5-6), 645-680.
- [46] Zabalza, M. Á. (2017). *Competencias docentes del profesorado universitario: calidad y desarrollo profesional*. Narcea Ediciones.
- [47] Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. *Theory Into Practice*, 41(2), 64-70.