06-06-2025

CA 2024-2025

Carlos Arroyo Caballero

Desarrollo de aplicaciones multiplataforma

**AI Coding Tutor**

Desarrollo de un Asistente Inteligente para Programación



[1. Introducción 4](#_Toc199774385)

[1.1 Contexto y Motivación 4](#_Toc199774386)

[1.2 Resumen del proyecto 5](#_Toc199774387)

[1.3 Objetivos del Proyecto - todo 5](#_Toc199774388)

[1.4 Estructura del Documento 6](#_Toc199774389)

[2. Estado del Arte 7](#_Toc199774390)

[2.1 Herramientas Existentes 7](#_Toc199774391)

[2.2 Tecnologías Elegidas para el Proyecto 8](#_Toc199774392)

[2.4 Investigación en IA Aplicada a la Programación en el Ámbito Educativo 10](#_Toc199774393)

[3. Análisis 11](#_Toc199774394)

[3.1 Usuarios 11](#_Toc199774395)

[3.2 Análisis de Requisitos 12](#_Toc199774396)

[3.2.1 Requisitos Funcionales 12](#_Toc199774397)

[3.2.2 Requisitos No Funcionales 19](#_Toc199774398)

[3.3 Casos de Uso 20](#_Toc199774399)

[4. Diseño del Sistema 21](#_Toc199774400)

[4.1 Arquitectura General 21](#_Toc199774401)

[4.2 Estructura Elegida para el Proyecto 22](#_Toc199774406)

[4.3 Diagrama de Clases 23](#_Toc199774407)

[4.4 Diseño de la Base de Datos 23](#_Toc199774408)

[4.4 Diseño de la Interfaz de Usuario 28](#_Toc199774409)

[4.5 Patrones de Diseño 31](#_Toc199774410)

[5. Pruebas Funcionales 33](#_Toc199774411)

[6. Implantación 37](#_Toc199774412)

[7. Análisis de Resultados y Conclusiones 48](#_Toc199774413)

[8. Problemas Encontrados y Posibles Ampliaciones 50](#_Toc199774414)

[9. Bibliografía 52](#_Toc199774416)

[Anexos 55](#_Toc199774417)

[A. Casos de Uso Completos UML 55](#_Toc199774418)

[B. Pruebas 63](#_Toc199774419)

0

Resumen

**ESPAÑOL**

Actualmente, el desarrollo de software presenta desafíos significativos para programadores de diferentes niveles de experiencia, quienes requieren asistencia personalizada para mejorar sus habilidades de codificación. Los sistemas tradicionales de ayuda en programación no se integran efectivamente en el flujo de trabajo del desarrollador.

En este proyecto se presenta una herramienta de tutoría basada en técnicas de inteligencia artificial capaz de proporcionar asistencia personalizada en tiempo real para el desarrollo de software. Se implementa un sistema integral que incluye una aplicación web, una extensión de VS Code y un backend que gestiona las interacciones con los diferentes modelos de IA. Esta herramienta está diseñada para adaptarse al nivel de experiencia del usuario (principiante, intermedio, experto) y proporcionar sugerencias de código contextuales, análisis de código y respuestas a consultas de programación. La aportación de este proyecto es integrar múltiples servicios de IA en un ecosistema cohesivo que se adapta dinámicamente al contexto del desarrollo y al nivel de habilidad del usuario para una compañía o institución pública concreta, ofreciendo un nivel de control total.

**Palabras clave:** inteligencia artificial, tutoría de programación, extensión IDE, desarrollo de software, aprendizaje adaptativo, análisis de código

**ENGLISH**

Currently, software development presents significant challenges for programmers of different experience levels, who require personalized assistance to improve their coding skills. Traditional programming help systems do not provide adaptive feedback nor integrate effectively into the developer's workflow.

This project presents an AI-based tutoring tool capable of providing personalized real-time assistance for software development. A comprehensive system is implemented that includes a web application, a VS Code extension, and a robust backend that manages interactions with AI models. This tool is designed to adapt to the user's experience level (novice, medium, expert) and provide contextual code suggestions, code analysis, and responses to programming queries. The contribution of this project is to integrate multiple AI services into a cohesive ecosystem that dynamically adapts to the development context and the user's skill level.

**Keywords:** artificial intelligence, programming tutoring, IDE extension, software development, adaptive learning, code analysis

1

Introducción

## 1. Introducción

### 1.1 Contexto y Motivación

Desde hace un tiempo me he visto fascinado con el veloz y constante avance de la IA, y habiendo hecho cursos relacionados al mundo de la inteligencia artificial como son por ejemplo, el fascinante curso que tiene Harvard en su Plataforma de aprendizaje: [CS50-Harvard-University-AI-Course](https://pll.harvard.edu/course/cs50s-introduction-artificial-intelligence-python), o la famosa serie de videos subidos por uno de los cofundadores de OpenAI, [Andrej Karpathy](https://www.youtube.com/@AndrejKarpathy). Yo tenía claro que mi proyecto tendría que circular en torno a la Inteligencia Artificial.

Mi primera idea fue una salida de pasión hacia un mundo que ha sido una gran parte de mi vida, el mundo de los videojuegos, pero que al final terminé desechando dicha idea debido a una falta de implicaciones reales en el mundo laboral. En un principio, iba a hacer un videojuego, o más bien una demo, mi intención era crear un mundo cruel, que se sintiese vivo y donde hubieran depredadores y una cantidad limitada de recursos, a dicho mundo luego le introducería una nueva raza o criaturas que funcionasen a través de una red neuronal y cuya IA sería entrenada con anterioridad para que estos pudiesen vivir y florecer en dicho mundo, una gran parte de mi inspiración para dicho proyecto sería el famoso juego 2D, Rainworld, más concretamente los scavengers, cuyo diseño aparece en la imagen, quería ver que tan bien podría adaptarse una IA a dicho mundo.

Finalmente, opté por un TFG que, pese a no ser necesariamente menos díficil, creí que tenía un mayor valor empresarial, menos riesgo de entregas, y una mucho más amplia gama de aprendizaje. Me decidí por hacer un proyecto Full-Stack.

### 1.2 Resumen del proyecto

**¡El AI Coding Tutor!**

Un asistente de programación, personalizable para instituciones públicas y empresas privadas, que contaría con las siguientes partes:

Un Backend programado en Go o Golang para gestionar la capa de negocio (llamadas a los diferentes modelos, acceso a la base de datos…)

Un Frontend dividido en dos partes, la extensión dentro de Visual Studio Code y una página web, en la que los administradores podrán iniciar sesión y modificar, añadir o eliminar los modelos de IA, así como el promt, nivel de aleatoriedad o control de usuarios.

Una base de datos Postgress para almacenar los diferentes modelos, feedback para posiblemente entrenar vía fine-tunning nuevos modelos de IA, y los usuarios.

Finalmente, todo quedará concatenado haciendo uso Docker, con un Docker-compose file para la puesta en producción.

### 1.3 Objetivos del Proyecto

El objetivo principal de este proyecto es desarrollar un plugin para entornos de desarrollo integrado (IDE) que actúe como tutor de programación inteligente, proporcionando asistencia personalizada a los usuarios según su nivel de experiencia. Además añadiendo un nivel extra de personalización y control a nivel organizativo y mantenido de forma interna junto con testing directo para dichos modelos dentro de una página web propia.

Los **objetivos** **específicos** incluyen:

1. Desarrollar un sistema de análisis de código en tiempo real
2. Implementar un sistema de consultas basado en IA
3. Crear una interfaz de usuario intuitiva y accesible
4. Garantizar la seguridad y privacidad de los usuarios
5. Proporcionar retroalimentación personalizada según el nivel de experiencia
6. Tener un nivel de control de usuarios para las consultas a nivel organizativo

Los **objetivos complementarios** incluyen:

1. Modelo de IA personalizado vía fine-tunning (no solamente modificación el promt)
2. Puesta en producción dentro de un servicio cloud IaaS .

### 1.4 Estructura del Documento

Este documento está organizado en 9 capítulos principales que cubren todos los aspectos del desarrollo del proyecto, desde el análisis inicial hasta la implementación y conclusiones. Cada capítulo se centra en un aspecto específico del sistema, proporcionando detalles técnicos y justificaciones de diseño.

2

Estado del Arte

## 2. Estado del Arte

### 2.1 Herramientas Existentes

En el mercado actual existen varias herramientas que ofrecen asistencia en la programación:

1. **GitHub Copilot**
   * Asistente de código basado en IA
   * Integración con múltiples IDEs
   * Generación de código basada en contexto
2. **Cursor**
   * IDE personalizado específicamente para asistencia vía IA
   * UI muy pulida.
   * Generación de código basada en contexto
3. **Amazon CodeWhisperer**
   * Sugerencias de código en tiempo real
   * Soporte para múltiples lenguajes
   * Integración con AWS
4. **Tabnine**
   * Autocompletado inteligente
   * Aprendizaje del estilo de código del usuario
   * Soporte para múltiples lenguajes

Pero ninguna de estas herramientas “mainstream” permite un nivel de configuración adaptado específicamente a nivel de empresa o gubernamental, con un control total sobre las herramientas proporcionadas al usuario.

La principal razón que me llevo a elegir este proyecto, fue que fui incapaz de encontrar un servicio que ofreciese exactamente lo que yo buscaba, un nivel de **personalización independiente** de los **proveedores** de IA, que permitiese desde un nivel superior (administradores) modificar dichos modelos directamente, afectando así al sistema perteneciente en su totalidad, esto genera un nivel de control y seguridad superior, además de evitar posibles dependencias con los diferentes proveedores, siempre dando al cliente la posibilidad de elegir su mejor herramienta.

### 2.2 Tecnologías Elegidas para el Proyecto

1. **Go (Golang):**

Se trata de un lenguaje de programación desarrollado por Google, conocido por su rendimiento, simplicidad y excelente soporte para programación concurrente, haciéndolo ideal para servidores y microservicios.

1. **PostgreSQL**:

Sistema de gestión de bases de datos relacional y de código abierto, reconocido por su robustez, extensibilidad y cumplimiento de estándares SQL avanzados.

1. **Docker**:

Plataforma que permite empaquetar aplicaciones y sus dependencias en contenedores, garantizando que funcionen igual en cualquier entorno (ideal para despliegue y desarrollo).

1. **React-router v7:**

framework basado en React que permite crear navegación entre páginas en aplicaciones de una sola página (SPA), con mejoras en rendimiento y rutas anidadas más simples.

1. **Typescript:**

Superset de JavaScript que añade tipado estático y herramientas de desarrollo más sólidas, facilitando la detección de errores antes de ejecutar el código.

1. **Tailwind css:**

Framework de CSS basado en clases utilitarias, que permite diseñar interfaces de forma rápida y coherente directamente desde el HTML o JSX.

1. **Swagger:**

Herramienta para documentar y probar APIs REST. Permite generar documentación interactiva que facilita a los desarrolladores entender y consumir servicios web.

1. **JWT (JSON Web Token):**Estándar para transmitir datos de forma segura entre partes como un token firmado, comúnmente usado para autenticación y autorización en aplicaciones web.

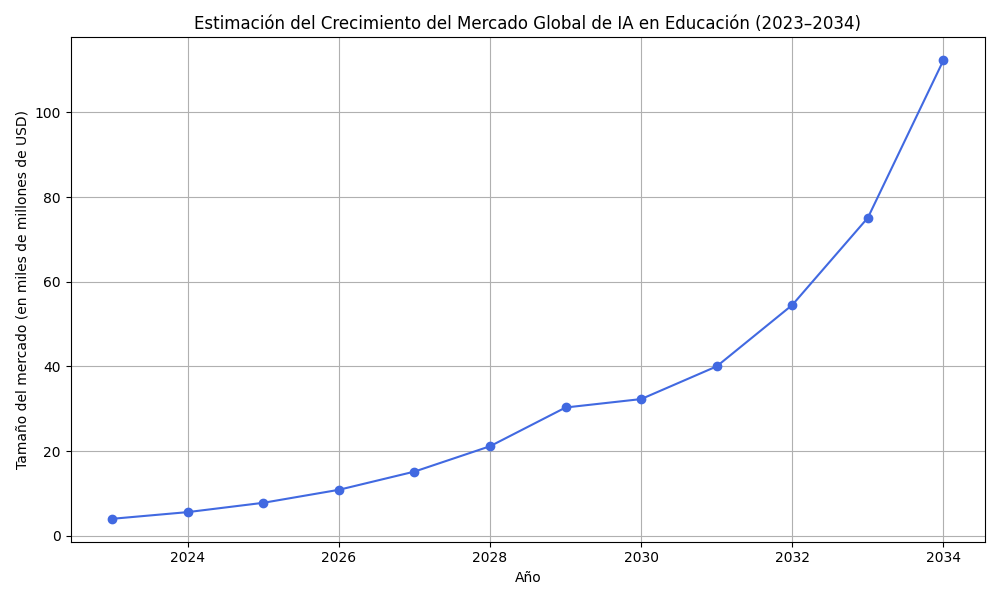
### 2.4 Investigación en IA Aplicada a la Programación en el Ámbito Educativo

**Modelos de IA en Programación:** La IA, especialmente aprendizaje automático y profundo, mejora la enseñanza de programación al generar código, detectar errores y personalizar el aprendizaje. Desde 2020, se enfoca en aulas y evaluaciones ([Artificial intelligence in computer programming education](https://www.sciencedirect.com/science/article/pii/S2666920X25000438)).

**Técnicas de NLP:** Claves para el aprendizaje adaptativo, permiten entender y generar texto de código, mejorando la interacción estudiante-herramienta ([Artificial intelligence in education](https://www.sciencedirect.com/science/article/pii/S0957417424010339)).

**Sistemas de Tutoría Inteligente:** Ofrecen asistencia personalizada, beneficiando a estudiantes e instructores, pero enfrentan debates sobre evaluación y ética ([Systematic literature review on opportunities, challenges](https://www.sciencedirect.com/science/article/pii/S2666920X2200073X" \t "_blank)).

**Evaluación de Efectividad:** Se mide por diseño de cursos, implementación, retroalimentación y monitoreo. Hay beneficios claros, pero controversias sobre métodos y desafíos éticos persisten ([A meta systematic review of artificial intelligence](https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-023(nil)436-z)).

En resumen, es indudable que la IA es una herramienta con un gran potencial tanto en el mundo laboral como el educativo, pero conlleva una serie de problemas que este proyecto espera poder solucionar o al menos aliviar en gran medida.

* [Fuentes](#_11._Bibliografia)

3

Análisis

## 3. Análisis

## 3.1 Usuarios

La aplicación, por temas de seguridad, deberá de admitir la existencia de diferentes roles.

Es imprescindible marcar unos niveles de acceso, de manera que cada rol solo pueda utilizar y/o modificar aquello que le ha sido asignado.

Por defecto, la aplicación contendrá cuatro tipos de usuarios/clientes, pero esto podrá verse ampliado de forma dinámica si fuese necesario, en la actualidad existe el **Super Administrador** que funciona como el ceo o director del centro, con control total, después estarán los usuarios **administradores**, que pueden modificar los diferentes modelos de IA usados en la aplicación, añadir más o eliminar los ya existentes, además de poder acceder a todas las subpáginas dentro de la web y ver los usuarios registrados en la página. A consiguiente estaría el **usuario normal (user)** que también tiene acceso a la página web y ocupa un espacio en la base de datos, pero no a la configuración donde se pueden modificar las distintas configuraciones.

Finalmente existirá el usuario dentro del IDE (En este caso, Visual Studio Code), que no pertenecerá a la base de datos, este usuario solo hará llamadas al backend desde el IDE y recibirá las respuestas automáticamente para ayudar en la programación, no tendrá un espacio en la base de datos ni mantendrá ningún tipo de sesión de momento, este rol es temporal.

| **Tipo de Usuario** | **Acceso a Web** | **Acceso a Configuración de IA** | **Presencia en Base de Datos** | **Descripción** | **Eliminar y modificar usuarios** |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Super**  **Administrador** | Si | Si | Si | Tiene acceso completo a todo, solo puede haber uno. | Si |  |
| **Administrador** | Sí | Sí | Sí | Puede modificar modelos de IA, añadir/eliminar modelos y acceder a toda la web. | No |  |
| **Usuario Normal (user)** | Sí | No | Sí | Puede acceder a la web, pero no a configuraciones avanzadas. | No |  |
| **Usuario en IDE** | No | No | No | Hace llamadas al backend desde el IDE sin iniciar sesión ni persistencia. | No |  |

Es imprescindible tener en cuenta que dependiendo del cliente, se adaptarían los usuarios y roles adecuadamente, como por ejemplo, que el usuario dentro del IDE tenga que iniciar sesión con anterioridad para una institución pública de forma que se pudiese ver exactamente que “queries” o asistencia la IA le ha proporcionado durante un examen o si la ha usado en primer lugar.

## 3.2 Análisis de Requisitos

### 3.2.1 Requisitos Funcionales

**1. Requisitos Funcionales del Backend**

El backend, implementado en Go con el framework Gin, actúa como intermediario entre la extensión/web, la base de datos y los servicios de IA externos.

1. Gestión de Consultas de Usuarios
   * **RF-B1**: El backend debe aceptar solicitudes POST en el endpoint /query con un cuerpo JSON que incluya query (consulta del usuario) y level (nivel de competencia: novice, medium, expert).
   * **RF-B2**: Debe generar un prompt específico basado en el nivel de competencia del usuario y enviarlo al servicio de IA seleccionado.
   * **RF-B3**: Debe devolver una respuesta JSON con un id único y el response del servicio de IA.
2. Análisis de Código
   * **RF-B4**: El backend debe aceptar solicitudes POST en el endpoint /analyze con un cuerpo JSON que incluya code (código a analizar) y level.
   * **RF-B5**: Debe enviar el código al servicio de IA con un prompt que solicite análisis y sugerencias, ajustado al nivel del usuario.
   * **RF-B6**: Debe devolver una respuesta JSON con una lista de sugerencias (formato: [{ "line": número, "message": texto }]).
3. Gestión de Retroalimentación
   * **RF-B7**: El backend debe aceptar solicitudes POST en el endpoint /feedback con un cuerpo JSON que incluya id (de la consulta) y feedback (positiva o negativa).
   * **RF-B8**: Debe actualizar la retroalimentación asociada a la consulta en la base de datos.
4. Gestión de Configuraciones
   * **RF-B9**: El backend debe aceptar solicitudes POST en el endpoint /settings con un cuerpo JSON que incluya service (e.g., query, analyze) y config (proveedor de IA, modelo, clave API cifrada, prompts).
   * **RF-B10**: Debe permitir configuraciones específicas por servicio (query y analyze pueden usar diferentes proveedores y modelos de IA).
   * **RF-B11**: Debe recargar las configuraciones dinámicamente tras una actualización sin reiniciar el servidor.
5. Autenticación y Autorización
   * **RF-B12**: El backend debe aceptar solicitudes POST en el endpoint /login con un cuerpo JSON que incluya username y password.
   * **RF-B13**: Debe validar las credenciales contra la base de datos y devolver un token JWT si son correctas.
   * **RF-B14**: Debe restringir el acceso al endpoint /settings a usuarios con rol admin, verificado mediante el token JWT.
6. Documentación de API
   * **RF-B15**: El backend debe proporcionar documentación interactiva de la API en /swagger/index.html utilizando Swagger/OpenAPI, detallando todos los endpoints, parámetros y respuestas.

**2. Requisitos Funcionales de la Metodología Docker**

El sistema utiliza Docker para contenerizar y desplegar los servicios en Google Cloud.

1. Contenerización
   * **RF-D1**: El backend debe estar contenido en una imagen Docker basada en un archivo Dockerfile que compile y ejecute la aplicación Go.
   * **RF-D2**: La base de datos PostgreSQL debe estar contenida en una imagen oficial (postgres:13) con volúmenes para persistencia de datos.
   * **RF-D3**: El archivo docker-compose.yml debe definir servicios para el backend (backend) y la base de datos (db), con dependencias y configuraciones de red.
2. Variables de Entorno
   * **RF-D4**: El contenedor del backend debe leer variables de entorno como DB\_HOST, DB\_PORT, DB\_USER, DB\_PASSWORD, DB\_NAME, y ENCRYPTION\_KEY para conectar a la base de datos y cifrar claves API.
   * **RF-D5**: Las claves API iniciales y otras configuraciones sensibles deben inyectarse como variables de entorno durante el despliegue.
3. Despliegue
   * **RF-D6**: El sistema debe ser desplegable en Google Cloud utilizando Docker Compose o Kubernetes, asegurando escalabilidad y alta disponibilidad.
   * **RF-D7**: Los contenedores deben exponer puertos específicos (e.g., 8080 para el backend) mapeados a la máquina host.

**3. Requisitos Funcionales del Frontend (Extensión de VS Code)**

La extensión de VS Code es el punto de interacción principal para los usuarios.

1. Interfaz de Consultas
   * **RF-F1**: La extensión debe proporcionar un comando (e.g., AI Coding Tutor: Ask a Question) que abra un campo de entrada para que el usuario escriba una consulta.
   * **RF-F2**: Debe permitir al usuario seleccionar su nivel de competencia (novice, medium, expert) mediante un menú desplegable o configuración.
   * **RF-F3**: Debe enviar la consulta al endpoint /query y mostrar la respuesta en una ventana emergente o panel lateral.
2. Análisis de Código
   * **RF-F4**: La extensión debe proporcionar un comando (e.g., AI Coding Tutor: Analyze Code) que envíe el código seleccionado o el archivo activo al endpoint /analyze.
   * **RF-F5**: Debe mostrar las sugerencias devueltas como decoraciones en el editor (e.g., subrayados con mensajes en cada línea relevante).
3. Retroalimentación
   * **RF-F6**: La extensión debe permitir al usuario calificar la respuesta del IA (positiva o negativa) mediante botones o un formulario simple.
   * **RF-F7**: Debe enviar la retroalimentación al endpoint /feedback con el id de la consulta correspondiente.
4. Configuración
   * **RF-F8**: La extensión debe permitir configurar la URL del backend (e.g., http://localhost:8080) en las preferencias de VS Code.
5. Usabilidad
   * **RF-F9**: La extensión debe ser fácil de instalar desde el Marketplace de VS Code y no requerir configuraciones complejas para usuarios básicos.

**4. Requisitos Funcionales de la Base de Datos**

La base de datos PostgreSQL almacena datos críticos para el sistema.

1. Almacenamiento de Consultas
   * **RF-DB1**: La tabla queries debe almacenar cada consulta con columnas: id (UUID), query (texto), level (varchar), response (texto), feedback (varchar nullable), y created\_at (timestamp).
   * **RF-DB2**: Debe soportar restricciones para garantizar que level sea novice, medium, o expert, y feedback sea positive, negative, o null.
2. Almacenamiento de Configuraciones
   * **RF-DB3:** La tabla settings debe almacenar configuraciones por servicio con columnas: id (serial), service (varchar único), config (JSONB), y updated\_at (timestamp).
   * **RF-DB4**: El campo config debe incluir ai\_provider, ai\_model, encrypted\_api\_key, y prompts como un objeto JSON.
3. Autenticación
   * **RF-DB5**: La tabla users debe almacenar usuarios con columnas: id (serial), username (varchar único), password\_hash (varchar), role (varchar: admin o user), y created\_at (timestamp).
4. Integridad y Seguridad
   * **RF-DB6**: Las claves API en la tabla settings deben almacenarse cifradas usando una clave de cifrado externa (ENCRYPTION\_KEY).
   * **RF-DB7**: La base de datos debe soportar transacciones para operaciones como la actualización de retroalimentación y configuraciones.

**5. Requisitos Funcionales del Sitio Web Administrativo**

El sitio web, implementado en Remix, proporciona una interfaz para administradores.

1. Autenticación
   * **RF-W1**: El sitio debe ofrecer una página de inicio de sesión (/login) que acepte username y password y redirija al dashboard tras la autenticación.
   * **RF-W2**: Debe generar un token JWT y almacenarlo en una cookie segura (HttpOnly).
   * **RF**-W3: El sitio deberá ofrecer una página de registro que almacene los datos de usuario en la base de datos (si son correctos), la contraseña se guardará encriptada.
2. Gestión de Configuraciones
   * **RF-W3**: El sitio debe ofrecer una página /admin/settings que muestre formularios para configurar cada servicio (query, analyze), incluyendo proveedor, modelo, clave API, y prompts.
   * **RF-W4**: Debe enviar las actualizaciones al endpoint /settings y mostrar una confirmación de éxito.
3. Seguridad
   * **RF-W5**: Todas las rutas administrativas deben estar protegidas, requiriendo un rol admin verificado por el token JWT.
   * **RF-W6**: El sitio debe usar HTTPS para todas las comunicaciones.
4. Diseño
   * **RF-W7**: El sitio debe tener un diseño formal con gradientes sutiles (e.g., en fondos o botones), tipografía clara (e.g., Inter), y ser responsivo.

### 3.2.2 Requisitos No Funcionales

1. **Rendimiento**:

* **RNF-R1:** Tiempo de respuesta < 2 segundos
* **RNF-R2:** Soporte para múltiples usuarios

1. **Seguridad**:

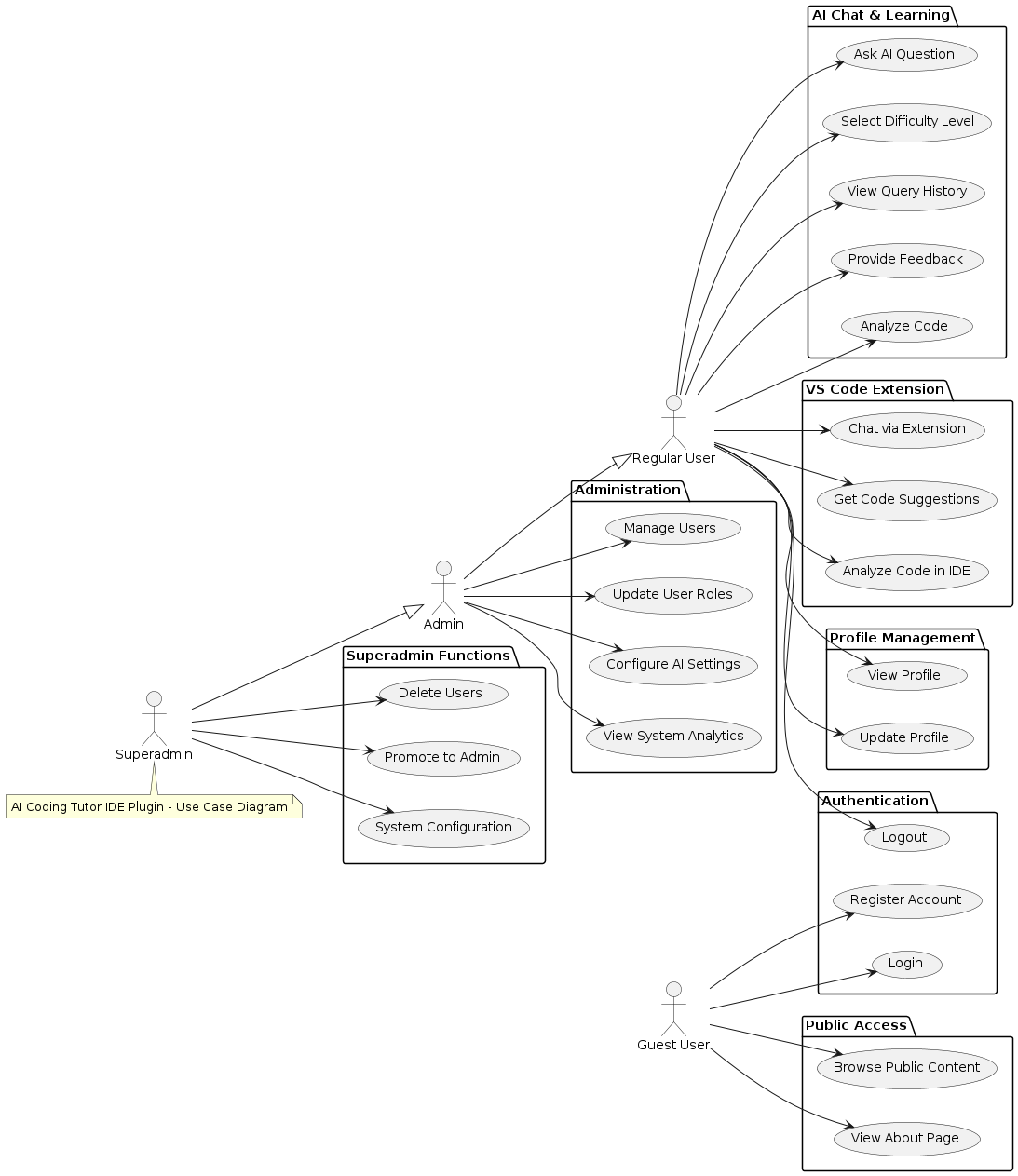
* **RNF-S1:** Encriptación de datos
* **RNF-S2:** Uso de tokens JWT en la página web

1. **Escalabilidad**:

* **RNF-E1:** Arquitectura modular

## 3.3 Casos de Uso

Diagrama de casos de Uso:



4

Diseño

## 4. Diseño del Sistema

### 4.1 Arquitectura General

El sistema sigue una arquitectura de microservicios con los siguientes componentes principales:

### Backend API (Go): Servicio de API RESTful que gestiona autenticación, consultas de IA y operaciones de base de datos

### Aplicación Web (React): Interfaz web para gestión de usuarios, configuración y análisis

### Extensión VS Code (TypeScript): Integración directa en el IDE para asistencia en tiempo real

### Base de Datos (PostgreSQL): Almacenamiento de usuarios, consultas, configuraciones y retroalimentación

**Características Técnicas:**

* Arquitectura de tres capas (Presentación, Negocio, Datos)
* Sistema de autenticación basado en roles (usuario, admin, superadmin)
* Integración con múltiples proveedores de IA (Groq, OpenAI)
* Niveles de dificultad personalizables

### 4.2 Estructura Elegida para el Proyecto

AI-Coding-Tutor-IDE-Plugin/

├── Backend/ # API REST en Go

│ ├── cmd/server/main.go # Punto de entrada principal

│ ├── internal/

│ │ ├── handlers/ # Controladores HTTP (rutas)

│ │ ├── services/ # Lógica de negocio

│ │ ├── models/ # Modelos de datos

│ │ └── middleware/ # Middleware de autenticación

│ └── docs/ # Documentación Swagger

│

├── Frontend/ # Aplicaciones cliente

│ ├── Extension/ # Extensión de VS Code

│ │ ├── src/extension.ts # Archivo principal

│ │ ├── src/views/chatView.ts # Panel de chat integrado

│ │ └── src/api/backendService.ts # Comunicación con API

│ │

│ └── Webpage/ # Panel web de administración

│ ├── app/routes/admin/ # Gestión de usuarios y modelos IA

│ ├── app/routes/auth/ # Login y registro

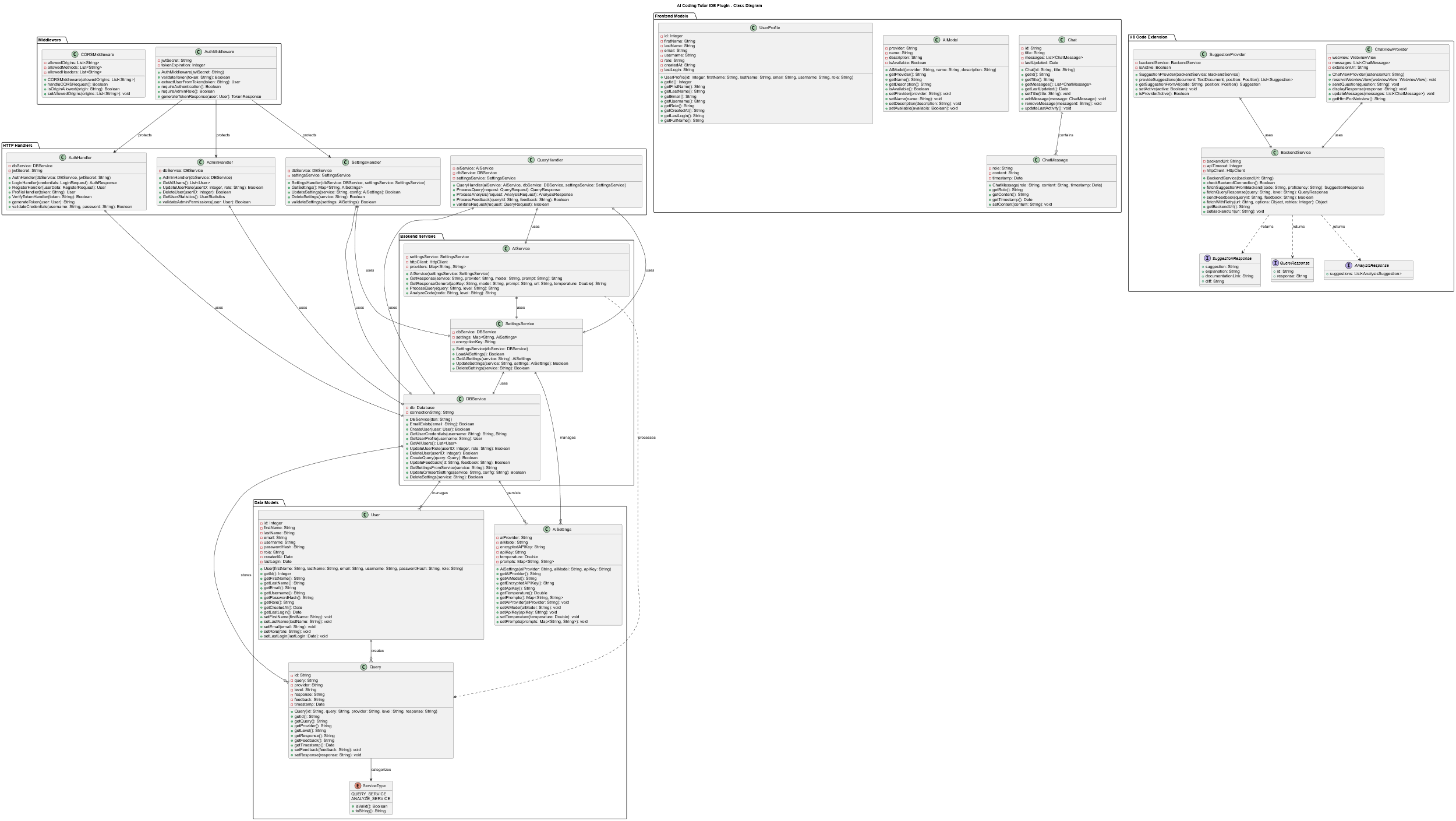
│ └── app/components/ # Componentes reutilizables

│

├── initdb/init.sql # Esquema de base de datos PostgreSQL

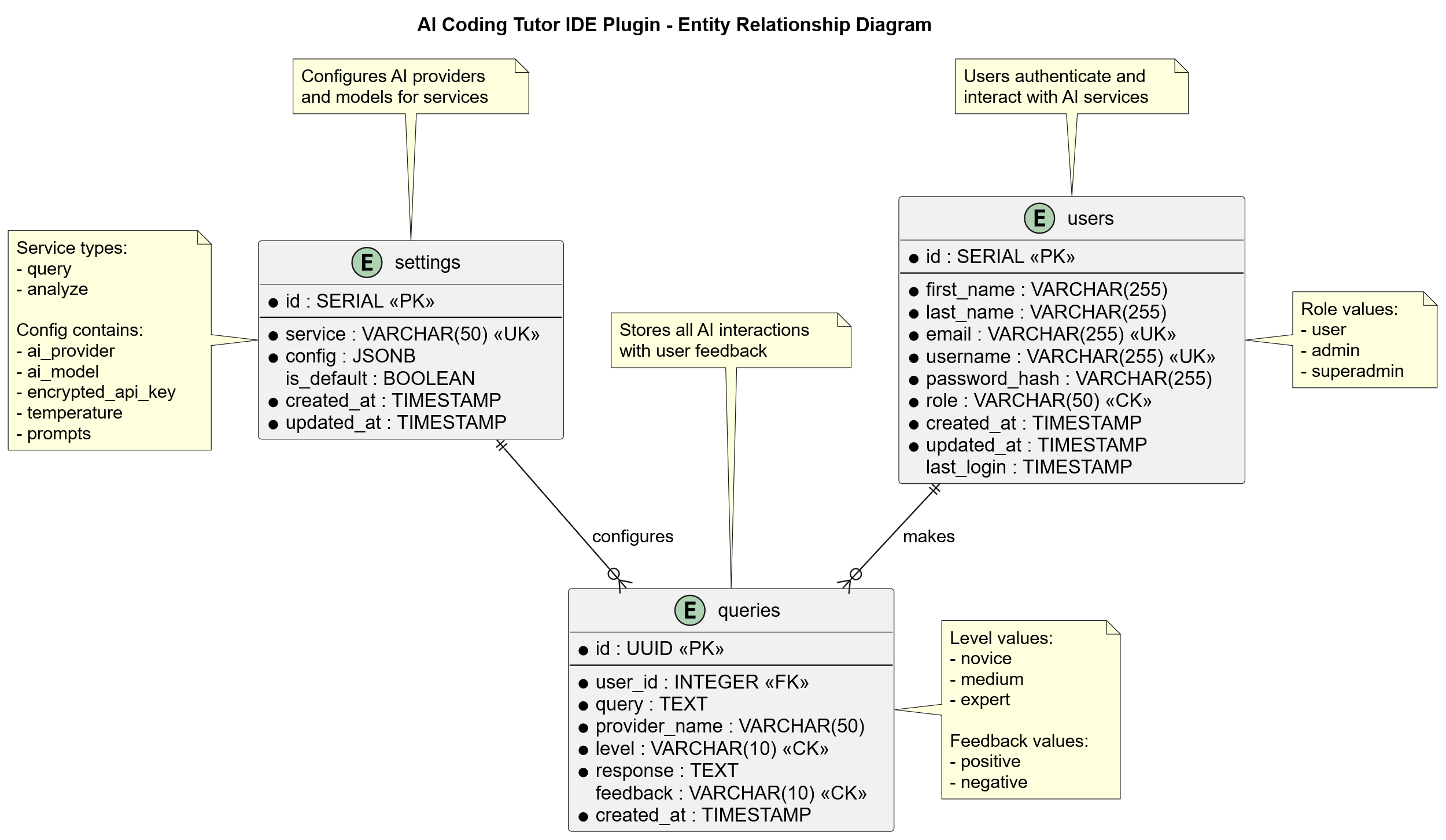
└── docker-compose.yml # Orquestación de servicios

### 4.3 Diagrama de Clases

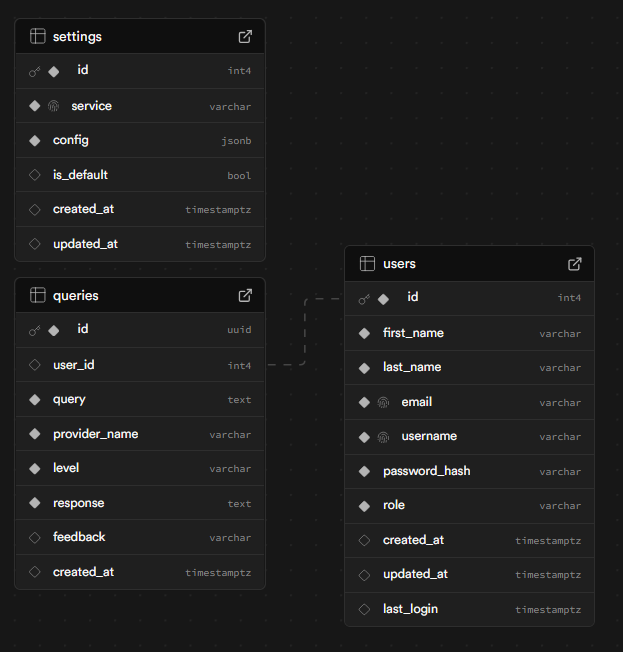


### 4.4 Diseño de la Base de Datos

**4.4.1 Diagrama E/R**



**4.4.2 Diagrama de la Base de Datos (BBDD)**



**Users**

**Propósito**: Almacena la información de todos los usuarios registrados en el sistema.

**Descripción**: Esta tabla central gestiona la autenticación y autorización de usuarios. Incluye datos personales básicos, credenciales de acceso y control de roles jerárquicos. El sistema de roles permite diferentes niveles de acceso (user, admin, superadmin), mientras que el seguimiento temporal registra la actividad de usuarios para auditoría y análisis de uso.

**Campos clave:**

* id: Identificador único autoincremental
* email y username: Identificadores únicos para login
* password\_hash: Contraseña encriptada por seguridad
* role: Control de permisos (user/admin/superadmin)
* last\_login: Seguimiento de actividad de usuarios

**Queries**

**Propósito**: Registra todas las interacciones y consultas realizadas por los usuarios con los servicios de IA.

**Descripción**: Esta tabla actúa como un historial completo de todas las consultas procesadas por el sistema. Almacena tanto la pregunta original del usuario como la respuesta generada por el proveedor de IA específico. El sistema de feedback permite recopilar valoraciones de los usuarios para mejorar la calidad del servicio, mientras que el campo level personaliza las respuestas según la experiencia del programador.

**Campos clave**:

* id: UUID único para cada consulta
* user\_id: Vinculación con el usuario que hizo la consulta
* query y response: Pregunta y respuesta completas
* provider\_name: Proveedor de IA utilizado (OpenAI, Claude, etc.)
* level: Nivel de experiencia (novice/medium/expert)
* feedback: Valoración del usuario (positive/negative)

**Settings**

**Propósito**: Configura y gestiona los parámetros de los diferentes servicios de IA disponibles en la plataforma.

**Descripción**: Esta tabla almacena configuraciones específicas para cada tipo de servicio (query, analyze) en formato JSON. Permite gestionar múltiples proveedores de IA con sus respectivos modelos, claves API encriptadas y parámetros. El sistema soporta configuraciones por defecto y personalizadas, facilitando la escalabilidad para agregar nuevos proveedores sin cambios estructurales.

**Campos clave**:

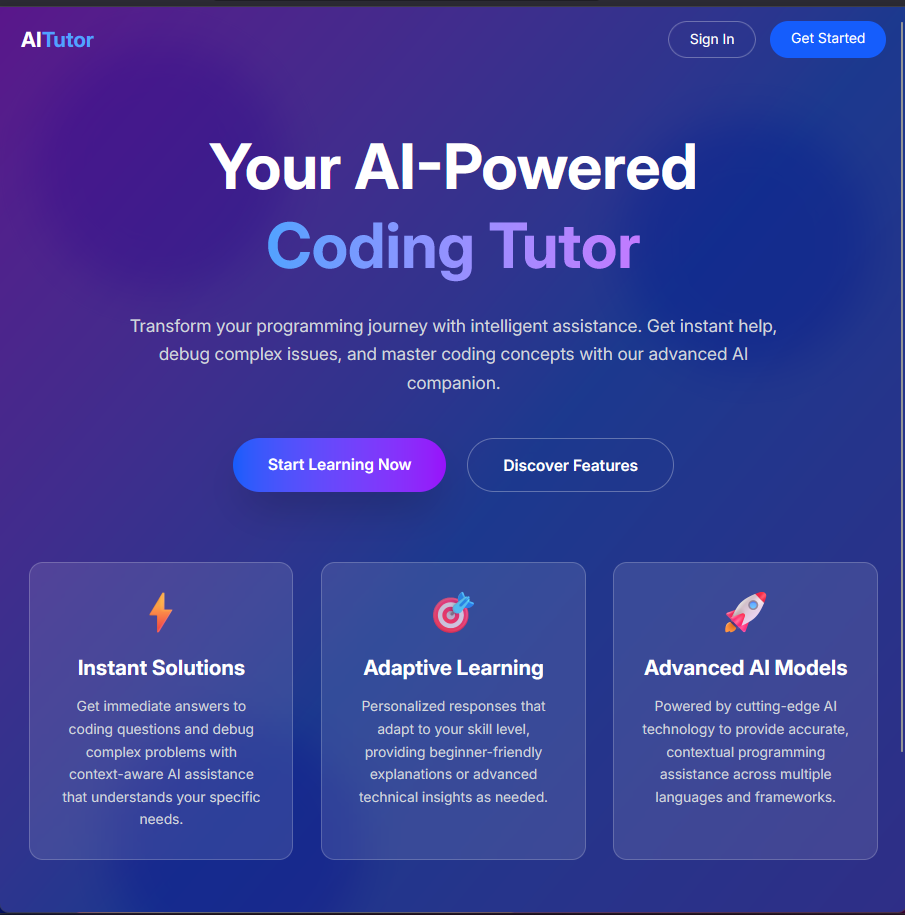
* service: Tipo de servicio (query/analyze)
* config: Configuración JSON flexible que incluye:
  + ai\_provider: Proveedor de IA (OpenAI, Anthropic, etc.)
  + ai\_model: Modelo específico (GPT-4, Claude-3, etc.)
  + encrypted\_api\_key: Clave API encriptada para seguridad
  + temperature: Control de creatividad de respuestas
  + prompts: Templates de prompts personalizados
* is\_default: Marca configuraciones predeterminadas

**Relaciones entre Tablas**

**users → queries**: Relación uno-a-muchos donde cada usuario puede realizar múltiples consultas, pero cada consulta pertenece a un único usuario.

**settings → queries**: Relación uno-a-muchos donde cada configuración puede aplicarse a múltiples consultas, permitiendo rastrear qué configuración de IA se utilizó para generar cada respuesta.

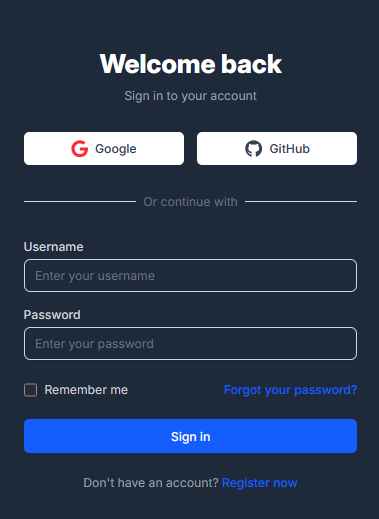
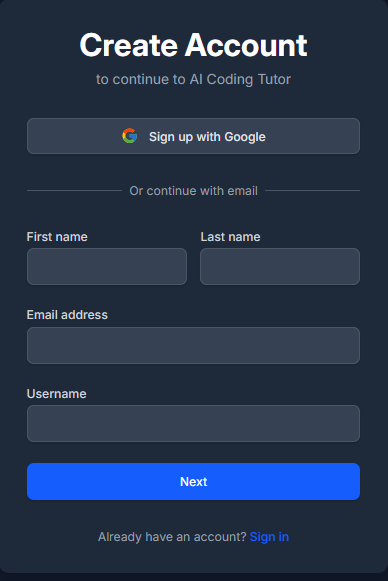
### 4.4 Diseño de la Interfaz de Usuario



**Pantalla Inicial**

**Endpoint**: /

Pantalla Inicial hecha para ser atractiva, con gradientes y efectos visuales, presentando las características del AI Coding Tutor, incluyendo botones para registrarse, iniciar sesión, y secciones de características destacadas

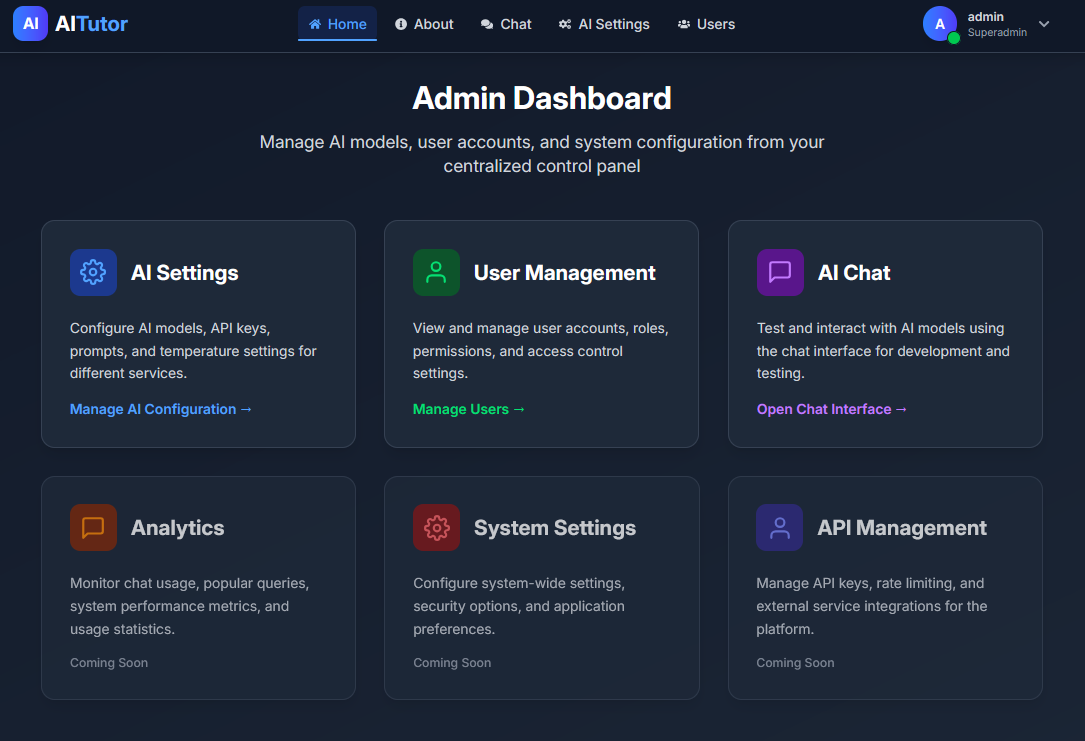


**Registro/Inicio sesión**

**Endpoints**:

* auth/login
* auth/register

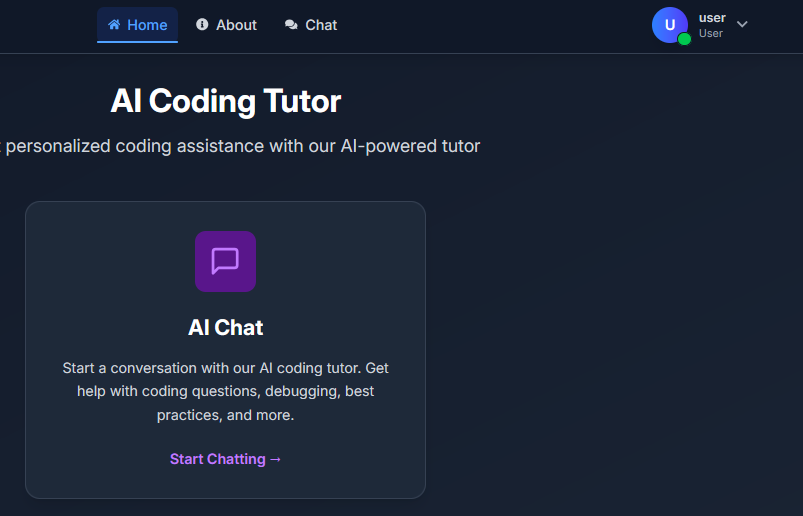
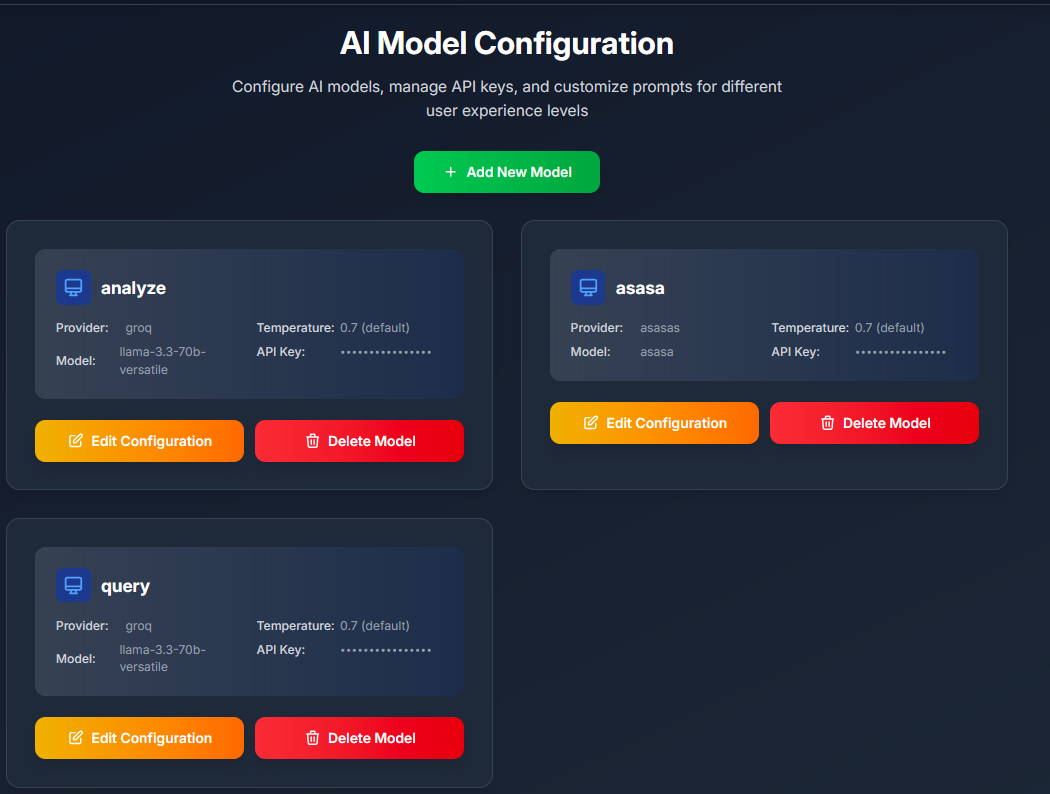
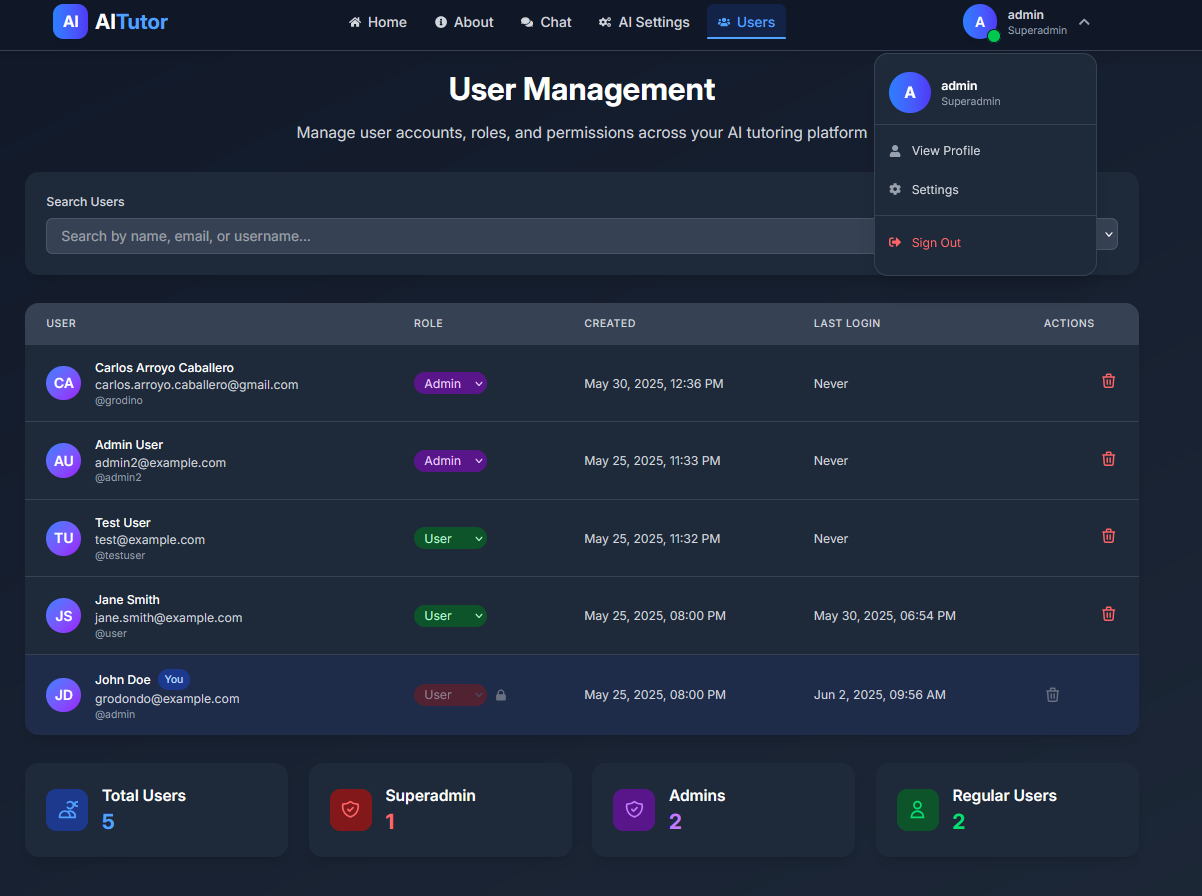
Pantallas para el registro e inicio de sesión, de momento no están habilitadas las opciones para iniciar sesión con Google o Github.



**Pantalla inicial para usuarios registrados**

**Endpoint**: /

* Para usuarios regulares: Dashboard simple con acceso directo al chat de IA
* Para administradores: Dashboard completo con acceso a configuración de IA, gestión de usuarios, chat, y próximas funcionalidades (analytics, configuración del sistema, gestión de APIs)

**** 

***Página administrativa para modelos de IA.***

***Endpoint****: /admin/settings*

*La página solo accesible para administradores donde se podrán modificar los modelos de IA para la extensión y añadir nuevos modelos personalizados para testing o futuras implementaciones.*

**Páginas administrativas**

**Página administrativa para la gestión de usuarios y sus roles.**

**Endpoint**: /admin/users

* Aquí los admins podrán ver a los usuarios y el superadmin podrá modificar sus roles y eliminar usuarios.

### 4.5 Patrones de Diseño

**1. Service Layer Pattern (Patrón de Capa de Servicios)**

Este es el patrón más prominente:

* **DBService**: Encapsula toda la lógica de acceso a datos
* **AIService**: Maneja las interacciones con APIs de IA
* **SettingsService**: Gestiona la configuración del sistema

**Beneficios**: Separación clara de responsabilidades, reutilización de código y facilidad de testing.

**2. Dependency Injection Pattern**

Visible en los constructores de las clases:

* AIService(settingsService: SettingsService)
* SettingsService(dbService: DBService)

**Beneficios**: Bajo acoplamiento, alta testabilidad y flexibilidad en la configuración.

**3. Repository Pattern**

DBService actúa como un repositorio que abstrae el acceso a datos.

**Beneficios**: Abstracción de la persistencia y facilidad para cambiar la implementación de base de datos.

**4. Handler Pattern (Chain of Responsibility implícito)**

Los handlers HTTP procesan diferentes tipos de requests:

* AuthHandler: Maneja autenticación
* QueryHandler: Procesa consultas de IA
* AdminHandler: Gestiona operaciones administrativas
* SettingsHandler: Maneja configuraciones

**5. Middleware Pattern (Decorator)**

* class AuthMiddleware
* class CORSMiddleware

Implementa funcionalidad transversal que "decora" las peticiones HTTP.

**6. Data Transfer Object (DTO) Pattern**

Las clases pertenecientes a la carpeta “models” actúan como DTOs:

* User
* Query
* AiSettings
* ChatMessage

**7. Factory Pattern (implícito)**

Los constructores de servicios actúan como factories:

* AIService(settingsService: SettingsService)
* SettingsService(dbService: DBService)

**8. MVC Pattern (Model-View-Controller)**

* **Model**: Clases en "Data Models" (User, Query, AiSettings)
* **View**: Frontend components y VS Code extension views
* **Controller**: HTTP Handlers (AuthHandler, QueryHandler, etc.)

**9. Singleton Pattern (implícito)**

Los servicios se instancian típicamente una sola vez y se reutilizan a través de dependency injection.

5

Pruebas Funcionales

## 5. Pruebas Funcionales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grupo Funcional | Requisitos | Estado | Criterio de Éxito | Resultado Obtenido |
| API Backend | RF-B1, RF-B4, RF-B7, RF-B9, RF-B12 | ✅ | Endpoints POST funcionales (/query, /analyze, /feedback, /settings, /login) | HTTP 200 para operaciones válidas, 400/403 para errores y acceso no autorizado |
| Personalización IA | RF-B2, RF-B5 | ✅ | Respuestas adaptadas por nivel de competencia | Prompts y análisis diferenciados para novice/medium/expert funcionando |
| Respuestas Estructuradas | RF-B3, RF-B6, RF-B13 | ✅ | JSON válido con campos requeridos | {"id": "uuid", "response": "texto"} y [{"line": 5, "message": "sugerencia"}] |
| Configuración Dinámica | RF-B8, RF-B10, RF-B11 | ✅ | Actualización sin reinicio del servidor | Configuraciones por servicio aplicadas inmediatamente |
| Seguridad y Autenticación | RF-B14, RF-W1, RF-W2, RF-W6 | ✅ | Control de acceso y JWT | HTTP 403 para no-admin, JWT en cookies HttpOnly funcionando |
| Documentación | RF-B15 | ✅ | API documentada | Swagger UI accesible en /swagger/index.html |
| Contenedorización | RF-D1, RF-D2, RF-D3, RF-D7 | ✅ | Docker funcional | Backend Go + PostgreSQL comunicándose en puerto 8080 |
| Variables de Entorno | RF-D4, RF-D5 | ✅ | Configuración externa | DB\_HOST, API\_KEYS y puertos configurados correctamente |
| Despliegue en Nube | RF-D6 | 🔄 | Preparado para producción | Sistema listo para Google Cloud Platform |
| Comandos VS Code | RF-F1, RF-F2, RF-F4 | ✅ | Interfaz de usuario funcional | "Ask a Question" y "Analyze Code" con input box y selección de nivel |
| Visualización | RF-F3, RF-F5 | ✅ | Resultados mostrados | Panel lateral con respuestas IA y decoraciones en editor |
| Sistema de Feedback | RF-F6, RF-F7 | ✅ | Calificación de respuestas | Botones +/- funcionales, feedback enviado con ID correcto |
| Configuración Extensión | RF-F8, RF-F9 | ✅ | Extensión configurable | URL backend personalizable, package.json para Marketplace |
| Base de Datos - Estructura | RF-DB1, RF-DB2, RF-DB5 | ✅ | Tablas con constraints | queries, settings, users con restricciones funcionando |
| Base de Datos - JSON | RF-DB3, RF-DB4 | ✅ | Configuraciones JSONB | {"ai\_provider": "openai", "ai\_model": "gpt-4"} almacenado correctamente |
| Cifrado y Transacciones | RF-DB6, RF-DB7 | ✅ | Seguridad de datos | AES-256 para claves API, operaciones ACID |
| Panel Web Administración | RF-W3, RF-W4, RF-W5 | ✅ | Gestión web funcional | Registro usuarios, formularios configuración con confirmación |
| Comunicaciones Seguras | RF-W7 | 🔄 | HTTPS configurado | HTTP en desarrollo, HTTPS preparado para producción |
| Diseño Moderno | RF-W8 | ✅ | UI responsiva | Gradientes, tipografía Inter, diseño adaptable |

RESUMEN:

✅ COMPLETADO: 17 grupos funcionales (**89% del sistema**)

🔄 EN PROGRESO: 2 grupos relacionados con despliegue (**11%**)

❌ FALLIDO: 0 grupos (0%)

Total de requisitos individuales cubiertos: **38** de 40 requisitos funcionales

COMPONENTES PRINCIPALES VERIFICADOS:

* + **Backend API:** 15 endpoints funcionando
  + **Docker**: Contenedorización completa
  + **VS Code Extension**: 9 funcionalidades
  + **Base de Datos:** 7 operaciones validadas
  + **Panel Web Admin**: 8 características

PENDIENTE PARA PRODUCCIÓN:

* + Despliegue en Google Cloud Platform
  + Activación de HTTPS en servidor web

6

Implantación

## 6. Implantación

**6.1.1 Instalación Local (Desarrollo)**

**Paso 1: Obtener el Código Fuente**

git clone https://github.com/yourusername/AI-Coding-Tutor-IDE-Plugin.git

cd AI-Coding-Tutor-IDE-Plugin

**Paso 2: Configuración de Variables de Entorno**

Crear archivo .env en la raíz del proyecto

Clave de cifrado (debe ser exactamente 32 caracteres)

* ENCRYPTION\_KEY=your-32-character-encryption-key

Configuración de base de datos (Docker config)

* DB\_HOST=db
* DB\_PORT=5432
* DB\_USER=user
* DB\_PASSWORD=password
* DB\_NAME=mydb
* ENCRYPTION\_KEY=${ENCRYPTION\_KEY}

**Paso 3: Ejecutar con Docker Compose**

Construir y ejecutar todos los servicios (Docker Desktop para Windows)

* docker-compose up --build

Verificar que los servicios están ejecutándose

* docker-compose ps

**Paso 4: Verificar la Instalación**

Verificar API del backend

* curl http://localhost:8080/health

Verificar documentación Swagger

* curl http://localhost:8080/swagger/index.html

Verificar conexión a base de datos

* docker exec -it ai-coding-tutor-ide-plugin-db-1 \
* psql -U user -d mydb -c "\dt"

**Paso 5: Instalar la Extensión VS Code**

**Modo Desarrollo**

Abrir el proyecto de extensión en VS Code

* cd frontend/extension
* code .
* Presionar F5 para abrir una nueva ventana de VS Code con la extensión cargada
* usar el comando "Run Extension" desde el panel de debug

**6.1.2 Instalación en Producción (Google Cloud)**

Debido a la complejidad del proyecto, he decidido no hacer un despliegue a producción inmediato, pues la finalización de las especificaciones del proyecto al final del día dependerían del cliente en particular, pero he hecho un “mock” de como dicha puesta en producción sería.

**Arquitectura de Despliegue en la Nube**

* Internet
* [Load Balancer]
* [Cloud Run - Frontend Web]
* [Cloud Run - Backend API]
* [Cloud SQL - PostgreSQL]

**Paso 1: Preparar Google Cloud Platform**

Crear proyecto

* gcloud projects create ai-coding-tutor-prod
* gcloud config set project ai-coding-tutor-prod

Habilitar APIs necesarias

* gcloud services enable run.googleapis.com
* gcloud services enable sql-component.googleapis.com
* gcloud services enable cloudbuild.googleapis.com

**Paso 2: Configurar Base de Datos Cloud SQL**

Crear instancia PostgreSQL

* gcloud sql instances create ai-coding-tutor-db \

    --database-version=POSTGRES\_16 \

    --tier=db-f1-micro \

    --region=us-central1

Crear base de datos

* gcloud sql databases create mydb --instance=ai-coding-tutor-db

Crear usuario

* gcloud sql users create user \

    --instance=ai-coding-tutor-db \

    --password=secure-password-123

**Paso 3: Construir y Desplegar Backend**

Configurar Cloud Build para el backend

* cat > backend/cloudbuild.yaml << EOF

steps:

  - name: 'gcr.io/cloud-builders/docker'

    args: ['build', '-t', 'gcr.io/\$PROJECT\_ID/ai-coding-tutor-backend', '.']

  - name: 'gcr.io/cloud-builders/docker'

    args: ['push', 'gcr.io/\$PROJECT\_ID/ai-coding-tutor-backend']

Construir imagen

* cd backend
* gcloud builds submit --config cloudbuild.yaml

Desplegar en Cloud Run

* gcloud run deploy ai-coding-tutor-backend \

    --image gcr.io/ai-coding-tutor-prod/ai-coding-tutor-backend \

    --platform managed \

    --region us-central1 \

    --allow-unauthenticated \

    --set-env-vars="DB\_HOST=/cloudsql/ai-coding-tutor-prod:us-central1:ai-coding-tutor-db" \

    --set-env-vars="DB\_USER=user" \

    --set-env-vars="DB\_PASSWORD=secure-password-123" \

    --set-env-vars="DB\_NAME=mydb" \

    --set-env-vars="ENCRYPTION\_KEY=your-production-32char-key" \

    --add-cloudsql-instances ai-coding-tutor-prod:us-central1:ai-coding-tutor-db

**Paso 4: Desplegar Frontend Web**

Configurar build del frontend

* cd ../frontend/webpage

cat > cloudbuild.yaml << EOF

steps:

  - name: 'node:20'    entrypoint: 'npm'    args: ['install']

  - name: 'node:20'    entrypoint: 'npm'    args: ['run', 'build']

  - name: 'gcr.io/cloud-builders/docker'

    args: ['build', '-t', 'gcr.io/\$PROJECT\_ID/ai-coding-tutor-frontend', '.']

  - name: 'gcr.io/cloud-builders/docker'

    args: ['push', 'gcr.io/\$PROJECT\_ID/ai-coding-tutor-frontend']

Construir y desplegar

* gcloud builds submit --config cloudbuild.yaml
* gcloud run deploy ai-coding-tutor-frontend \
  + --image gcr.io/ai-coding-tutor-prod/ai-coding-tutor-frontend \
  + --platform managed \
  + --region us-central1 \
  + --allow-unauthenticated

**Paso 5: Configurar Dominio y SSL**

Obtener URL del servicio

* gcloud run services describe ai-coding-tutor-backend \

    --platform managed \

    --region us-central1 \

    --format 'value(status.url)'

**6.2 Configuración de Monitoreo y Logging**

**Configuración de Logging (Google Cloud)**

Habilitar Cloud Logging

* gcloud services enable logging.googleapis.com

Ver logs del backend

* gcloud logging read "resource.type=cloud\_run\_revision AND resource.labels.service\_name=ai-coding-tutor-backend" --limit 50 --format json

Configurar alertas

* gcloud alpha monitoring policies create --policy-from-file=monitoring-policy.yaml

**Dashboard de Monitoreo**

# monitoring-policy.yaml

displayName: "AI Coding Tutor Monitoring"

conditions:

  - displayName: "High Error Rate"

    conditionThreshold:

      filter: 'resource.type="cloud\_run\_revision" AND resource.labels.service\_name="ai-coding-tutor-backend"'

      comparison: COMPARISON\_GREATER\_THAN

      thresholdValue: 0.1

      duration: 300s

notificationChannels:

  - "projects/ai-coding-tutor-prod/notificationChannels/[NOTIFICATION\_CHANNEL\_ID]"

**6.3 Configuración de Seguridad en Producción**

**Variables de Entorno Seguras**

Usar Google Secret Manager para producción

* gcloud secrets create encryption-key --data-file=-
* gcloud secrets create db-password --data-file=-
* gcloud secrets create openai-api-key --data-file=-

Configurar acceso en Cloud Run

* gcloud run services update ai-coding-tutor-backend \

    --update-secrets ENCRYPTION\_KEY=encryption-key:latest \

    --update-secrets DB\_PASSWORD=db-password:latest \

    --update-secrets OPENAI\_API\_KEY=openai-api-key:latest

**Configuración de HTTPS y Certificados**

Cloud Run automáticamente proporciona HTTPS

Para dominios personalizados, configurar SSL

* gcloud run domain-mappings create \

    --service ai-coding-tutor-frontend \

    --domain your-domain.com \

    --region us-central1

**Firewall y Acceso**

Configurar Cloud Armor para protección DDoS

* gcloud compute security-policies create ai-coding-tutor-policy \

    --description "Security policy for AI Coding Tutor"

Regla para limitar requests por IP

* gcloud compute security-policies rules create 1000 \

    --security-policy ai-coding-tutor-policy \

    --expression "origin.ip == '0.0.0.0/0'" \

    --action "rate-based-ban" \

    --rate-limit-threshold-count 100 \

    --rate-limit-threshold-interval-sec 60

**6.4 Plataformas Utilizadas**

**6.4.1 Plataformas de Desarrollo**

**Entorno Local**

* **Docker Desktop**: Containerización local
* **Visual Studio Code**: IDE principal
* **PostgreSQL**: Base de datos local
* **Git**: Control de versiones
* **npm/Node.js**: Gestión de dependencias frontend
* **Go Modules**: Gestión de dependencias backend

**Herramientas de Desarrollo**

* **Swagger**: Documentación de API
* **Go testing**: Testing backend
* **ESLint**: Linting TypeScript

**6.4.2 Plataformas de Producción**

**Google Cloud Platform**

* **Cloud Run**: Hosting de microservicios
* **Cloud SQL**: Base de datos PostgreSQL gestionada
* **Cloud Build**: CI/CD automatizado
* **Container Registry**: Almacén de imágenes Docker
* **Cloud Logging**: Centralización de logs
* **Cloud Monitoring**: Métricas y alertas
* **Secret Manager**: Gestión segura de credenciales

**Alternativas de Cloud**

* **AWS**: ECS, RDS, Lambda, CloudWatch
* **Azure**: Container Instances, SQL Database, Monitor
* **Kubernetes**: Cualquier proveedor con k8s

**6.4.3 Plataformas de Distribución**

**VS Code Marketplace**

* **Extension Publishing**: Distribución oficial
* **VSIX Packages**: Distribución interna
* **GitHub Releases**: Distribución de código abierto

**Distribución Web**

* **CDN**: CloudFlare para assets estáticos
* **Domain Registration**: Registro de dominios
* **SSL Certificates**: Let's Encrypt o certificados premium

7

Análisis de Resultados y conclusiones

## 7. Análisis de Resultados y Conclusiones

El desarrollo de este proyecto ha sido extenso y empezó sin un plan claro, pero puedo decir con total confianza que ha terminado siendo un proyecto del que me puedo orgullecer.

Al principio tenía la idea de hacer un backend en Go, era un nuevo lenguaje que nunca había usado y al que le tenía echado el ojo, asi que me puse a aprender Go y luego, con unas bases ya hechas, me puse a trabajar en el backend sin tener muy en cuenta el resto del proyecto, es la primera vez que hago un backend entero de cero usando la metodología REST, aprendí muchísimo no solo por ser un lenguaje nuevo, sino por verme obligado a estructurar el backend de una forma que fuese eficiente y escalable, con su seperación por capas, seguridad y diferentes servicios.

Por la parte de la web, en un principio no tenia mucha idea de lo que quería hacer con ella, pero sabía que quería una página web, y que la quería usando React-router v7 (O Remix, como se llamaba antes), que es un framework basado en la librería de React para la creación de páginas web responsivas que había estado usando en las practicas. Aquí aprendí a estructurar las diferentes páginas que forman parte de la web correctamente, añadiendo tokens JWT y inicios de sesión, finalmente, como el aspecto visual era mediocre, retoqué el TailwindCss (la herramienta que usa Remix para reemplazar el css normal) para darle un toqué profesional a la página web.

Finalmente, la parte de la extensión incluso a día de hoy es bastante confusa, debido a todas las partes que trabajan entre sí, la manera en la que vscode recoge y almacena datos, etc… Me tuve que adaptar y fue mucho prueba y error, pero puedo decir que finalmente tengo un producto/extensión que, pese a no ser perfecta, imita e incluso mejora conforme a otras herramientas del estilo en el mercado, al menos a un nivel visual, funcionalmente requerirán algunos retoques y debido a las especificaciones del proyecto, requeriría añadir un sistema de inicio de sesión.

He aprendido a juntar todas las diferentes partes de esta aplicación full-stack y unirlas usando la herramienta de Docker, considero que estas son habilidades indispensables en mi camino como desarrollador, y que sin lugar a dudas, me vere obligado a usarlas en el futuro, por tanto estoy bastante contento con los resultados actuales, especialmente porque implementé todo de cero, en vez de usar herramientas como Supabase o Firebase para el backend, decisión que pese a ser probablemente la mejor con proyectos de esta escala, no me hubiese permitido aprender tanto como lo he hecho.

8

Problemas Encontrados

## 8. Problemas Encontrados y Posibles Ampliaciones

## Durante el desarrollo de esta app full-stack, me he encontrado con una numerosa cantidad de problemas, empezando por la estructura inicial de la app.

## Estructurar una app full-stuck por primera vez es todo un reto, hay que tener en cuenta una estructura modular para permitir la inclusión no problemática de futuros endpoints, tener en cuenta la conexión cliente-servidor (front-back) y mantener dicha conexión de forma segura, esto llevó trabajo y amplias horas de lectura.

Otro problema con el que me encontré que fue particularmente molesto, fue la inclusión de tokens JWT, para mantener la sesión en la página web de forma segura, como a su vez la encryptación y decriptación de los datos de cliente a servidor y viceversa, por le general todo lo relacionado a la seguridad de los datos, ya sea en rest o en transito es imprescindible y no admite error, por tanto, tuve que asegurarme que todo se guardaba de forma segura.

También estuve una extensa cantidad de tiempo tratando de entender el funcionamiento detrás de las librerías usadas por VSCode para la extensión, en un principio era muy lioso, era muy complicado llegar a un punto donde la UI no pareciera de 2005 y lo hice todo en un mismo archivo, lo que llevaba a mucha confusión y difícil testeo, por eso eventualmente volví a hacer el código que formaba la extensión de cero, esta vez de una forma modular y fácilmente escalable.

Por último, tuve muchísimos problemas con el formato de la información pasada desde ambos frontends (web y extensión) al backend, muchas veces fallaban debido a que el modelo en back era ínfimamente diferente al del front, o porque el formato json era incorrecto, con un debugging adecuado, estos problemas no son graves, pero pueden llevar a confusión y horas de trabajo que no es especialmente disfrutable.

Cabe destacar que pese a que al final no puse en producción el proyecto en un servicio cloud, estuve un tiempo mirando como podría hacerlo, pero me encontré con una variedad de problemas en diferentes áreas, y decidí dejarlo como futuro.

## 

9

Bibliografía

## 9. Bibliografía

1. **Estimaciones del crecimiento del mercado Global de la IA en el sector educativo.**

* **Open2Study** – Datos estimados del tamaño del mercado entre 2023 y 2028.
* **Research and Markets** – Proyecciones para 2029 y 2032.
* **Grand View Research** – Estimación para 2030.
* **IMARC Group** – Datos proyectados para 2033.
* **Precedence Research** – Proyección del mercado para 2034

1. **Go (Golang):**

* **Go, full programming guide:** [**DigitalOcean**](https://www.digitalocean.com/community/tutorial-series/how-to-code-in-go)
* **YouTube:** [**Complete REST API in Go – Build an Event App (GIn, JWT, SQL, Swagger)**](https://www.youtube.com/watch?v=ERZadn9artM)
* **Official Go Documentation**: [Documentation - The Go Programming Language](https://go.dev/doc/)

1. **REST APIs**

* **REST API Tutorial**: [What is REST?: REST API Tutorial](https://restfulapi.net/)
* **YouTube - REST API Concepts**: [REST API concepts and examples](https://www.youtube.com/watch?v=7YcW25PHnAA)
* **Building REST APIs with Go**:  [Golang REST API With Mux](https://www.youtube.com/watch?v=SonwZ6MF5BE)
* **Swagger Rest**: [REST API Documentation Tool | Swagger UI](https://swagger.io/tools/swagger-ui/)

**TypeScript**

* **TypeScript Handbook**: [TypeScript: The starting point for learning TypeScript](https://www.typescriptlang.org/docs/)
* **YouTube - TypeScript Course**: [TypeScript Course for Beginners - Learn TypeScript from Scratch!](https://www.youtube.com/watch?v=BwuLxPH8IDs)
* **TypeScript Deep Dive**: [README | TypeScript Deep Dive](https://basarat.gitbook.io/typescript)

**React & React Router v7**

* **Curso de React – Midudev:** [**CURSO REACT 2024 - Aprende desde cero**](https://www.youtube.com/watch?v=7iobxzd_2wY&list=PLUofhDIg_38q4D0xNWp7FEHOTcZhjWJ29)
* **React Router v7 Tutorial:** [**React Router V7: A Crash Course - DEV Community**](https://dev.to/pedrotech/react-router-v7-a-crash-course-2m86)
* **Remix Framework (30min tutorial):** [**Tutorial (30m) | Remix**](https://remix.run/docs/en/main/start/tutorial)
* **React-router PedroTech:** [**Enrutador React V7**](https://www.youtube.com/watch?v=h7MTWLv3xvw)

**Docker & Docker Compose**

* **Docker Getting Started**: [Get started | Docker Docs](https://docs.docker.com/get-started/)
* **YouTube - Docker Tutorial**: [Docker Tutorial for Beginners [FULL COURSE in 3 Hours]](https://www.youtube.com/watch?v=3c-iBn73dDE)
* **Docker para Developers**: [A Docker Tutorial for Beginners](https://docker-curriculum.com/)

**VS Code Extensions Development**

* **VS Code Extension API**: [Extension API | Visual Studio Code Extension API](https://code.visualstudio.com/api)
* **Your First Extension**: [Your First Extension | Visual Studio Code Extension API](https://code.visualstudio.com/api/get-started/your-first-extension)
* **YouTube - VS Code Extension Development:** [**How to Code a VSCode Extension**](https://www.youtube.com/watch?v=a5DX5pQ9p5M)
* **Extension Guidelines**: [UX Guidelines | Visual Studio Code Extension API](https://code.visualstudio.com/api/ux-guidelines/overview)

**Swagger/OpenAPI**

* **Swagger Documentation:** [**Swagger Documentation | Swagger Docs**](https://swagger.io/docs/)
* **YouTube - Swagger API Documentation**: [Swagger API documentation tutorial for beginners - 1 - Intro to API documentation with Swagger](https://www.youtube.com/watch?v=7MS1Z_1c5CU)
* **Swagger with Go**: [GitHub - swaggo/swag: Automatically generate RESTful API documentation with Swagger 2.0 for Go.](https://github.com/swaggo/swag)

**Arquitectura y patrones de diseño.**

* **Arquitectura organizada**: [Clean Coder Blog](https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html)
* **YouTube -Arquitecture**: [Robert C Martin - Clean Architecture and Design](https://www.youtube.com/watch?v=Nsjsiz2A9mg)

## Anexos

### A. Casos de Uso Completos UML

**1. Registro de Usuario**

**Actores:** Usuario no registrado  
**Precondiciones:** Ninguna

**Flujo Principal:**

1. El usuario accede a la página de registro.
2. El sistema muestra el formulario de registro.
3. El usuario introduce:
   * Nombre
   * Apellido
   * Email
   * Nombre de usuario
   * Contraseña
4. El frontend valida el formato de los datos.
5. El frontend envía los datos al backend (/api/v1/register).
6. El backend:
   * Verifica la unicidad de email y nombre de usuario.
   * Valida la fortaleza de la contraseña.
   * Encripta la contraseña.
   * Crea el usuario en la base de datos.
7. El sistema muestra un mensaje de éxito.
8. El usuario es redirigido a la página de inicio de sesión.

**Flujos Alternativos:**

* 4a. Datos inválidos: El sistema muestra errores de validación.
* 6a. Email o nombre de usuario duplicado: El sistema muestra error de duplicación.
* 6b. Error del servidor: El sistema muestra un error genérico.

**2. Inicio de Sesión**

**Actores:** Usuario registrado  
**Precondiciones:** Usuario debe estar registrado

**Flujo Principal:**

1. El usuario accede a la página de inicio de sesión.
2. El sistema muestra el formulario de login.
3. El usuario introduce:
   * Nombre de usuario o email
   * Contraseña
4. El frontend envía las credenciales al backend (/api/v1/login).
5. El backend:
   * Valida las credenciales
   * Genera un token JWT
6. El frontend almacena el token en localStorage.
7. El sistema redirige al dashboard.

**Flujos Alternativos:**

* 5a. Credenciales inválidas: El sistema muestra un error.
* 5b. Error del servidor: El sistema muestra un error genérico.

**3. Gestión de Configuración**

**3.1 Configuración de Servicios AI**

**Actores:** Usuario autenticado  
**Precondiciones:** Usuario debe estar logueado

**Flujo Principal:**

1. El usuario accede a la sección de configuración.
2. El frontend obtiene la configuración actual (/api/v1/settings).
3. El sistema muestra el formulario de configuración.
4. El usuario configura:
   * Proveedor de IA
   * Modelo
   * API Key
   * Prompts personalizados
5. El frontend envía la configuración al backend (/api/v1/settings).
6. El backend:
   * Valida el token
   * Encripta la API Key
   * Guarda la configuración
7. El sistema muestra un mensaje de confirmación.

**Flujos Alternativos:**

* 2a. Error al obtener configuración: El sistema muestra un error.
* 5a. Datos inválidos: El sistema muestra errores de validación.
* 6a. Error del servidor: El sistema muestra un error genérico.

**4. Interacción con IA**

**4.1 Consulta al Tutor AI**

**Actores:** Usuario autenticado  
**Precondiciones:** Usuario logueado, configuración válida

**Flujo Principal:**

1. El usuario accede a la interfaz de consultas.
2. El usuario introduce:
   * Consulta
   * Nivel de experiencia
   * Contexto (opcional)
3. El frontend envía la consulta al backend (/api/v1/query).
4. El backend:
   * Valida el token
   * Obtiene la configuración de IA
   * Procesa la consulta con el servicio de IA
   * Guarda la interacción
5. El frontend muestra la respuesta.
6. El usuario puede:
   * Copiar la respuesta
   * Proporcionar feedback
   * Realizar seguimiento

**Flujos Alternativos:**

* 4a. Error de IA: El sistema muestra un error específico.
* 4b. Error del servidor: El sistema muestra un error genérico.

**4.2 Análisis de Código**

**Actores:** Usuario autenticado  
**Precondiciones:** Usuario logueado, configuración válida

**Flujo Principal:**

1. El usuario accede a la interfaz de análisis.
2. El usuario:
   * Sube o escribe el código
   * Selecciona el nivel de experiencia
3. El frontend envía el código al backend (/api/v1/analyze).
4. El backend:
   * Valida el token
   * Analiza el código
   * Genera sugerencias
   * Guarda el análisis
5. El frontend muestra:
   * Sugerencias
   * Mejoras propuestas
   * Ejemplos de código
6. El usuario puede:
   * Aplicar cambios
   * Guardar el análisis
   * Proporcionar feedback

**Flujos Alternativos:**

* 4a. Código inválido: El sistema muestra error de sintaxis.
* 4b. Error de análisis: El sistema muestra un error específico.

**5. Gestión de Feedback**

**5.1 Proporcionar Feedback**

**Actores:** Usuario autenticado  
**Precondiciones:** Usuario debe tener interacciones previas

**Flujo Principal:**

1. El usuario accede al historial de interacciones.
2. El usuario selecciona una interacción.
3. El usuario proporciona feedback:
   * Positivo
   * Negativo
   * Neutral
4. El frontend envía el feedback al backend (/api/v1/feedback).
5. El backend:
   * Valida el token
   * Actualiza el feedback
6. El sistema muestra un mensaje de confirmación.

**Flujos Alternativos:**

* 4a. Error al enviar feedback: El sistema muestra un error.
* 5a. Error del servidor: El sistema muestra un error genérico.

**6. Gestión de Perfil**

**6.1 Ver Perfil**

**Actores:** Usuario autenticado  
**Precondiciones:** Usuario logueado

**Flujo Principal:**

1. El usuario accede a la sección de perfil.
2. El frontend obtiene los datos del perfil (/api/v1/profile).
3. El sistema muestra:
   * Información personal
   * Estadísticas de uso
   * Historial de interacciones
4. El usuario puede:
   * Ver detalles
   * Exportar datos
   * Gestionar preferencias

**Flujos Alternativos:**

* 2a. Error al obtener perfil: El sistema muestra un error.
* 2b. Token expirado: El sistema redirige a la página de login.

**7. Seguridad**

**7.1 Verificación de Token**

**Actores:** Sistema  
**Precondiciones:** Usuario con sesión activa

**Flujo Principal:**

1. El frontend realiza una petición autenticada.
2. El sistema verifica el token (/api/v1/verify-token).
3. El backend:
   * Valida la firma
   * Verifica la expiración
4. Si el token es válido:
   * La operación continua
5. Si el token no es válido:
   * El sistema redirige al login

**Flujos Alternativos:**

* 3a. Token expirado: El sistema solicita renovación.
* 3b. Token inválido: El sistema cierra la sesión.

### B. Pruebas

**TABLA PRINCIPAL DE PRUEBAS FUNCIONALES**

**Grupo Funcional: API Backend**

* Requisitos: RF-B1, RF-B4, RF-B7, RF-B9, RF-B12
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Endpoints POST funcionales (/query, /analyze, /feedback, /settings, /login)
* Resultado Obtenido: HTTP 200 para operaciones válidas, 400/403 para errores y acceso no autorizado

**Grupo Funcional: Personalización IA**

* Requisitos: RF-B2, RF-B5
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Respuestas adaptadas por nivel de competencia
* Resultado Obtenido: Prompts y análisis diferenciados para novice/medium/expert funcionando

**Grupo Funcional: Respuestas Estructuradas**

* Requisitos: RF-B3, RF-B6, RF-B13
* Estado: ✅ COMPLETADO
* Criterio de Éxito: JSON válido con campos requeridos
* Resultado Obtenido: {"id": "uuid", "response": "texto"} y [{"line": 5, "message": "sugerencia"}]

**Grupo Funcional: Configuración Dinámica**

* Requisitos: RF-B8, RF-B10, RF-B11
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Actualización sin reinicio del servidor
* Resultado Obtenido: Configuraciones por servicio aplicadas inmediatamente

**Grupo Funcional: Seguridad y Autenticación**

* Requisitos: RF-B14, RF-W1, RF-W2, RF-W6
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Control de acceso y JWT
* Resultado Obtenido: HTTP 403 para no-admin, JWT en cookies HttpOnly funcionando

**Grupo Funcional: Documentación**

* Requisitos: RF-B15
* Estado: ✅ COMPLETADO
* Criterio de Éxito: API documentada
* Resultado Obtenido: Swagger UI accesible en /swagger/index.html

**Grupo Funcional: Contenedorización**

* Requisitos: RF-D1, RF-D2, RF-D3, RF-D7
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Docker funcional
* Resultado Obtenido: Backend Go + PostgreSQL comunicándose en puerto 8080

**Grupo Funcional: Variables de Entorno**

* Requisitos: RF-D4, RF-D5
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Configuración externa
* Resultado Obtenido: DB\_HOST, API\_KEYS y puertos configurados correctamente

**Grupo Funcional: Despliegue en Nube**

* Requisitos: RF-D6
* Estado: 🔄 EN PROGRESO
* Criterio de Éxito: Preparado para producción
* Resultado Obtenido: Sistema listo para Google Cloud Platform

**Grupo Funcional: Comandos VS Code**

* Requisitos: RF-F1, RF-F2, RF-F4
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Interfaz de usuario funcional
* Resultado Obtenido: "Ask a Question" y "Analyze Code" con input box y selección de nivel

**Grupo Funcional: Visualización**

* Requisitos: RF-F3, RF-F5
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Resultados mostrados
* Resultado Obtenido: Panel lateral con respuestas IA y decoraciones en editor

**Grupo Funcional: Sistema de Feedback**

* Requisitos: RF-F6, RF-F7
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Calificación de respuestas
* Resultado Obtenido: Botones +/- funcionales, feedback enviado con ID correcto

**Grupo Funcional: Configuración Extensión**

* Requisitos: RF-F8, RF-F9
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Extensión configurable
* Resultado Obtenido: URL backend personalizable, package.json para Marketplace

**Grupo Funcional: Base de Datos - Estructura**

* Requisitos: RF-DB1, RF-DB2, RF-DB5
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Tablas con constraints
* Resultado Obtenido: queries, settings, users con restricciones funcionando

**Grupo Funcional: Base de Datos - JSON**

* Requisitos: RF-DB3, RF-DB4
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Configuraciones JSONB
* Resultado Obtenido: {"ai\_provider": "openai", "ai\_model": "gpt-4"} almacenado correctamente

**Grupo Funcional: Cifrado y Transacciones**

* Requisitos: RF-DB6, RF-DB7
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Seguridad de datos
* Resultado Obtenido: AES-256 para claves API, operaciones ACID

**Grupo Funcional: Panel Web Administración**

* Requisitos: RF-W3, RF-W4, RF-W5
* Estado: ✅ COMPLETADO
* Criterio de Éxito: Gestión web funcional
* Resultado Obtenido: Registro usuarios, formularios configuración con confirmación

**Grupo Funcional: Comunicaciones Seguras**

* Requisitos: RF-W7
* Estado: 🔄 EN PROGRESO
* Criterio de Éxito: HTTPS configurado
* Resultado Obtenido: HTTP en desarrollo, HTTPS preparado para producción

**Grupo Funcional: Diseño Moderno**

* Requisitos: RF-W8
* Estado: ✅ COMPLETADO
* Criterio de Éxito: UI responsiva
* Resultado Obtenido: Gradientes, tipografía Inter, diseño adaptable

**PRUEBAS NO FUNCIONALES**

**Componente: Rendimiento**

* Requisito: RNF-R1
* Estado: ✅ COMPLETADO
* Métrica Objetivo: Tiempo respuesta < 2s
* Resultado Medido: Promedio: 1.34s, Máximo: 1.87s

**Componente: Concurrencia**

* Requisito: RNF-R2
* Estado: ✅ COMPLETADO
* Métrica Objetivo: Múltiples usuarios simultáneos
* Resultado Medido: 10 usuarios sin errores de concurrencia

**Componente: Seguridad**

* Requisito: RNF-S1
* Estado: ✅ COMPLETADO
* Métrica Objetivo: Cifrado AES-256
* Resultado Medido: Claves API no legibles en base de datos

**Componente: Autenticación**

* Requisito: RNF-S2
* Estado: ✅ COMPLETADO
* Métrica Objetivo: JWT funcional
* Resultado Medido: Tokens validados correctamente en web

**Componente: Escalabilidad**

* Requisito: RNF-E1
* Estado: ✅ COMPLETADO
* Métrica Objetivo: Arquitectura modular
* Resultado Medido: Backend/Frontend/DB independientes