

AIAppTest: Integración Web y Móvil con IA

AIAppTest es un proyecto integral que combina una aplicación móvil Android (cliente) con un backend web en Python (servidor). Su objetivo es permitir a los usuarios interactuar con una IA de forma sencilla y segura.



Componentes Clave del Proyecto

El sistema se divide en dos partes, cada uno con un rol específico en la arquitectura general, asegurando una separación clara de responsabilidades.

AIAppTest-AndroidApp

Aplicación cliente desarrollada en Kotlin y Jetpack Compose.

AIAppTest-WebApp

Backend en Python (Flask) con MySQL y la API de OpenAI.



Estructura de la Aplicación Android

La aplicación móvil está diseñada para una experiencia de usuario fluida, con

- **templates/**: ChatUI.kt, LoginUI.kt (pantallas principales con Jetpack Compose)
- **utils/**: AuthHandler.kt, IChatHandler.kt, TokenManager.kt (manejadores de lógica)
- **MainActivity.kt**: Actividad principal que inicia la interfaz de usuario.

```
me.idkbemja.aichatapp
├── templates
│   ├── ChatUI.kt
│   └── LoginUI.kt
├── ui.theme
├── utils
│   ├── AuthHandler.kt
│   ├── IChatHandler
│   └── TokenManager
└── MainActivity
```

Incluye funciones como envío con Enter, scroll automático en el chat y almacenamiento seguro del JWT.



Estructura del Backend Web

El backend gestiona la lógica del servidor, la base de datos y la integración con la inteligencia artificial, utilizar

- **main.py:** Inicializa la aplicación y crea tablas de la base de datos.
- **app/controllers/:** api_controller.py, app_controller.py (endpoints REST y lógica de rutas).
- **app/models/:** Users.py (modelo de usuario).
- **app/utils/:** jwt.py, OpenAIUtil.py, mysqlhandler.py, mysqlconnection.py, config/env_config.py (u

```

└─ app
  └─ controllers
    ├── api_controller.py
    └── app_controller.py
  └─ models
    └── Users.py
  └─ static
    ├── css
    └── js
        └── app.js
  └─ templates
    └── index.html
  └─ utils
    └─ config
        ├── env_config.py
        ├── jwt.py
        ├── mysqlconnection.py
        ├── mysqlhandler.py
        └── OpenAIUtil.py
```

Los endpoints principales son: /api/register, /api/login, /api/ask, /api/logout, /api/refresh-token.



Tecnologías Utilizadas

El proyecto AIAppTest se basa en un conjunto robusto de tecnologías para garantizar rendimiento, seguridad y escalabilidad.



Móvil

Kotlin, Jetpack Compose, Coroutines, Material3, OkHttp.



Base de Datos

MySQL para la gestión de datos.



Servidor

Python (Flask), flask-bcrypt, pymysql, python-dotenv, openai.



Autenticación

JWT con tabla `jwt_blacklist` para invalidar tokens.

Flujo Funcional: Registro y Login

El proceso de registro y autenticación es fundamental para la seguridad y personalización de la experiencia del usuario.



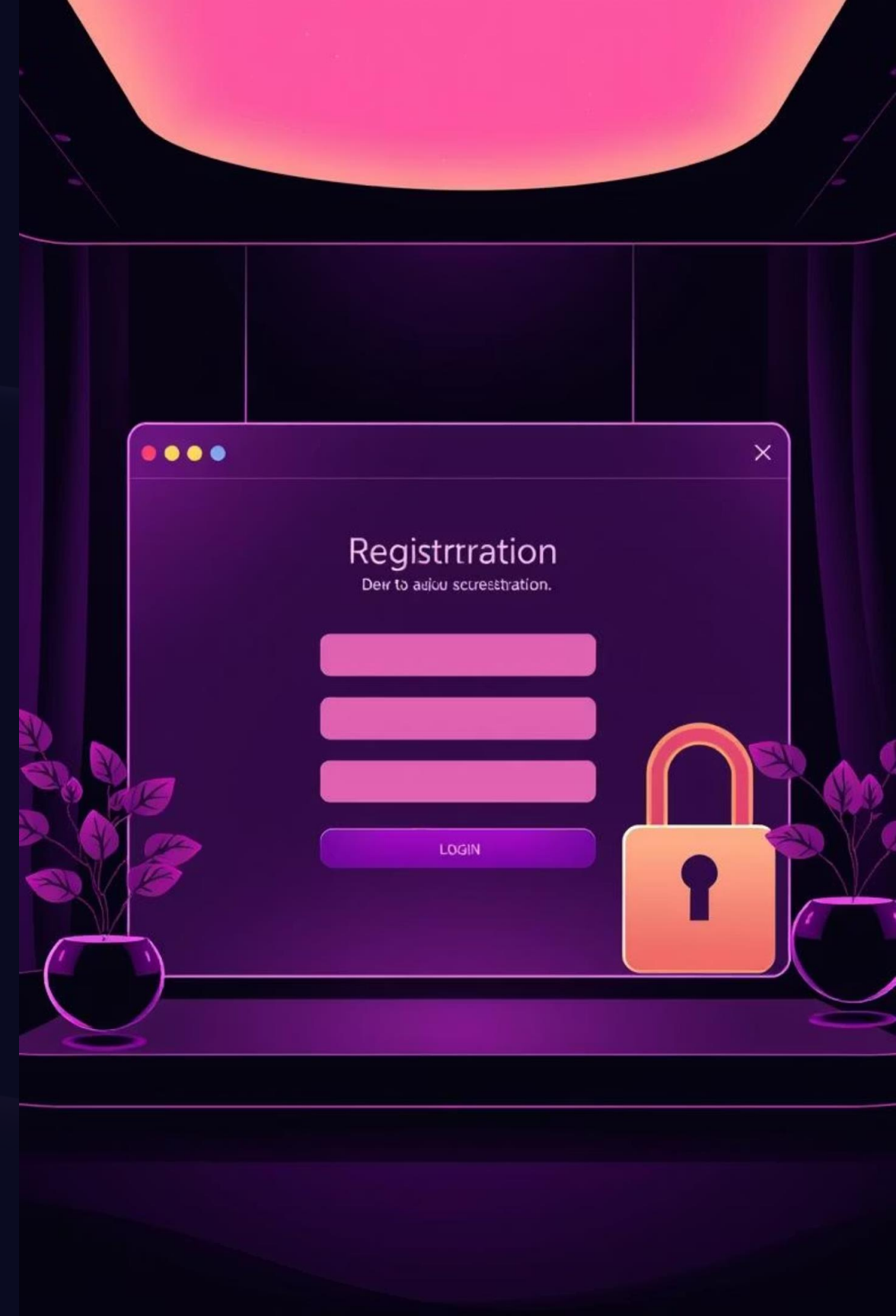
Registro (POST /api/register)

Crea un usuario con nombre, contraseña hashada y token IA personal.



Login (POST /api/login)

Verifica credenciales y devuelve un JWT para acceder a endpoints protegidos.



Flujo Funcional: Interacción con IA

Una vez autenticado, el usuario puede interactuar con la IA a través de endpoints específicos, gestionando las consultas y la validez de la sesión.

01

Enviar Pregunta (POST /api/ask)

Requiere JWT. El backend usa el token IA del usuario para consultar OpenAI y retorna la respuesta.

02

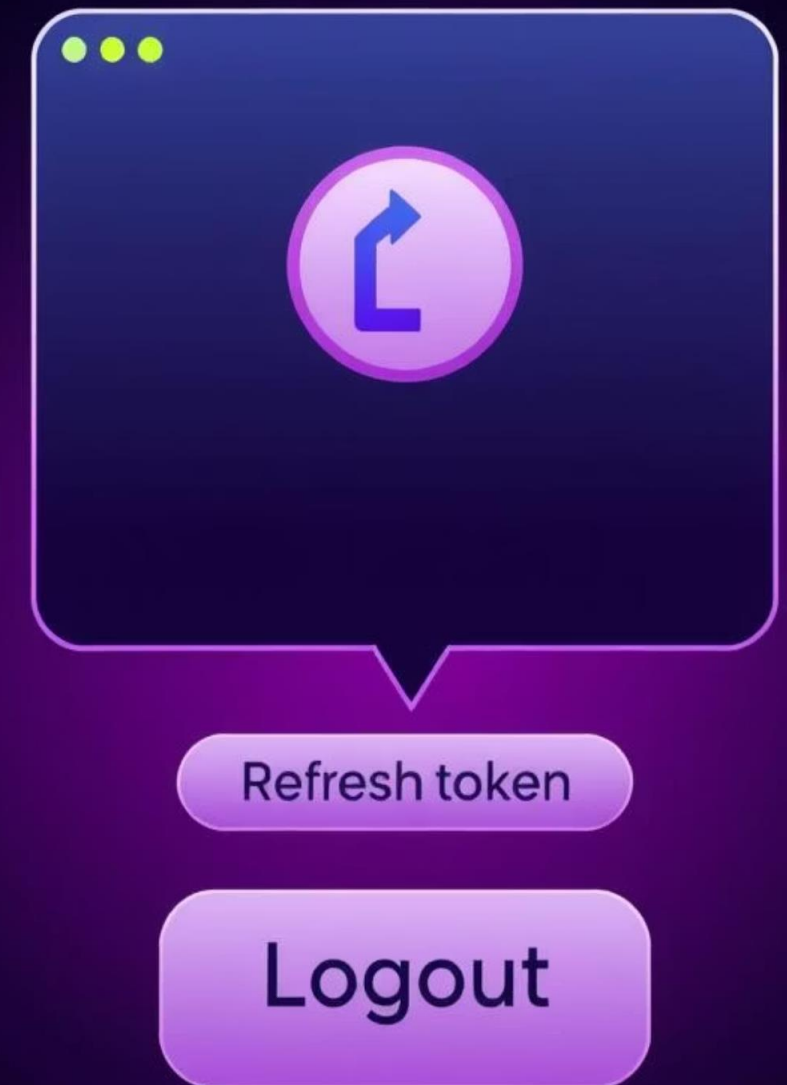
Logout (POST /api/logout)

Añade el JWT actual a la jwt_blacklist, invalidándolo.

03

Refresh Token (POST /api/refresh-token)

Renueva el JWT antes de su expiración para extender la sesión.



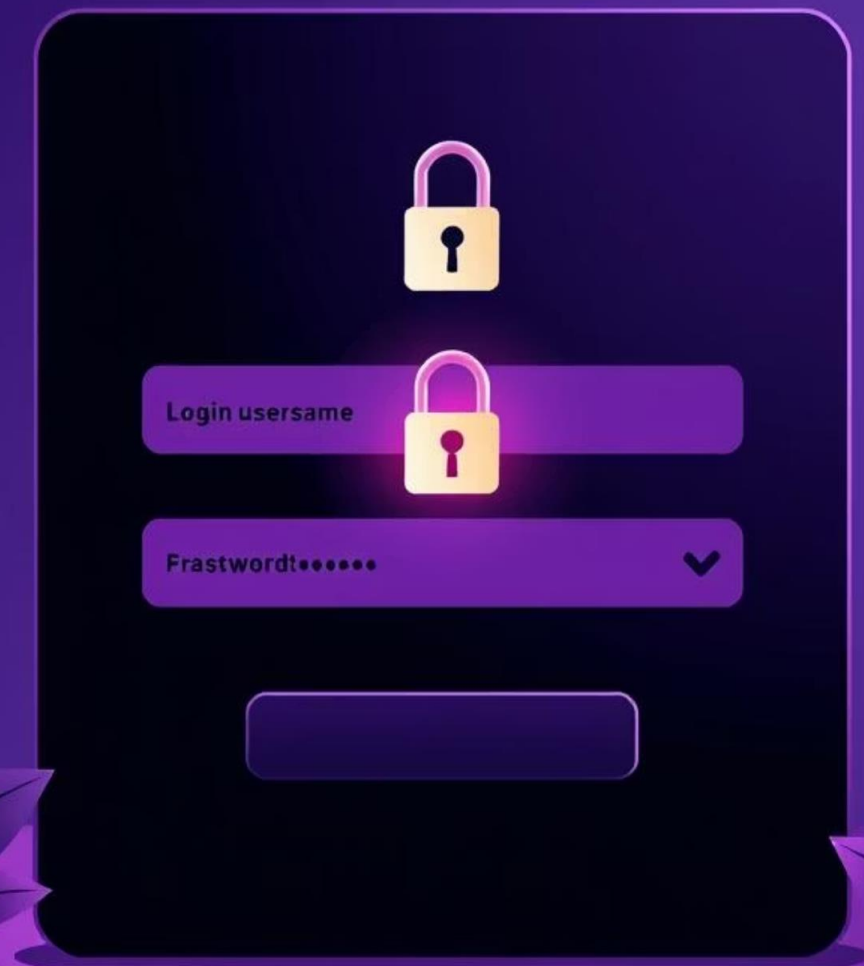
Ejemplo de Interacción: Login

Este ejemplo ilustra cómo un cliente (móvil o curl) realizaría una petición de login al backend.

```
curl -X POST https://aichattest.urrahost.app/api/login -H "Content-Type: application/json" -d '{"username":"alice", "password":"miPasswordSeguro" }'
```

Respuesta esperada:

```
{ "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...", "username": "alice", "token_ia": "sk-xxxxx" }
```





Ejemplo de Interacción: Enviar Pregunta

Una vez autenticado, el cliente utiliza el JWT para enviar preguntas a la IA y recibir respuestas.

```
curl -X POST https://aichattest.urrahost.app/api/ask -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1Ni..." -d eyJhbGciOiJIUzI1Ni..." -d '{"question": "¿Cuál es la capital de Chile?"}'
```

Respuesta esperada:

```
{ "response": "La capital de Chile es Santiago.", "metadata": { "model_used": "gpt-4-like", "usage_ms": 1234 } }
```

Urra Hosting: Alojamiento Web en Chile

Este proyecto se aloja en Urra Hosting, un hosting orientado a desarrolladores y pymes en Chile donde la aplicación utiliza las siguientes urls y endpoints.

Aplicación Web:

La aplicación web tiene un formulario de registro donde pide usuario, contraseña y token IA otorgado por Github Marketplace (Lugar para utilizar playgrounds con IA).

- **URL:** <https://aichattest.urrahost.app>

Aplicación Android

La aplicación android apunta a los endpoints especificos para cada llamada.

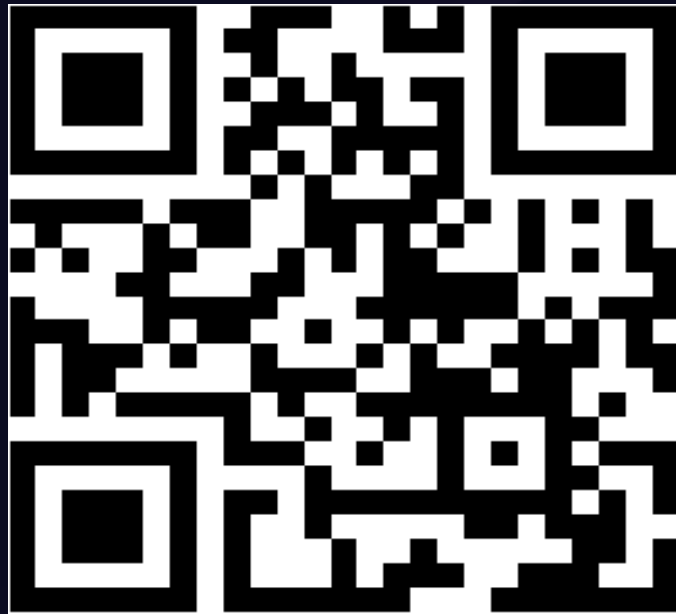
- POST: <https://aichattest.urrahost.app/api/login>
- POST: <https://aichattest.urrahost.app/api/ask>
- POST: <https://aichattest.urrahost.app/api/logout>



QR del Hosting



Demostración Página Web



QR Página de Registro

El Token IA se obtiene en github marketplace.

AI Registration

Username:

Password:

Token AI:

Register

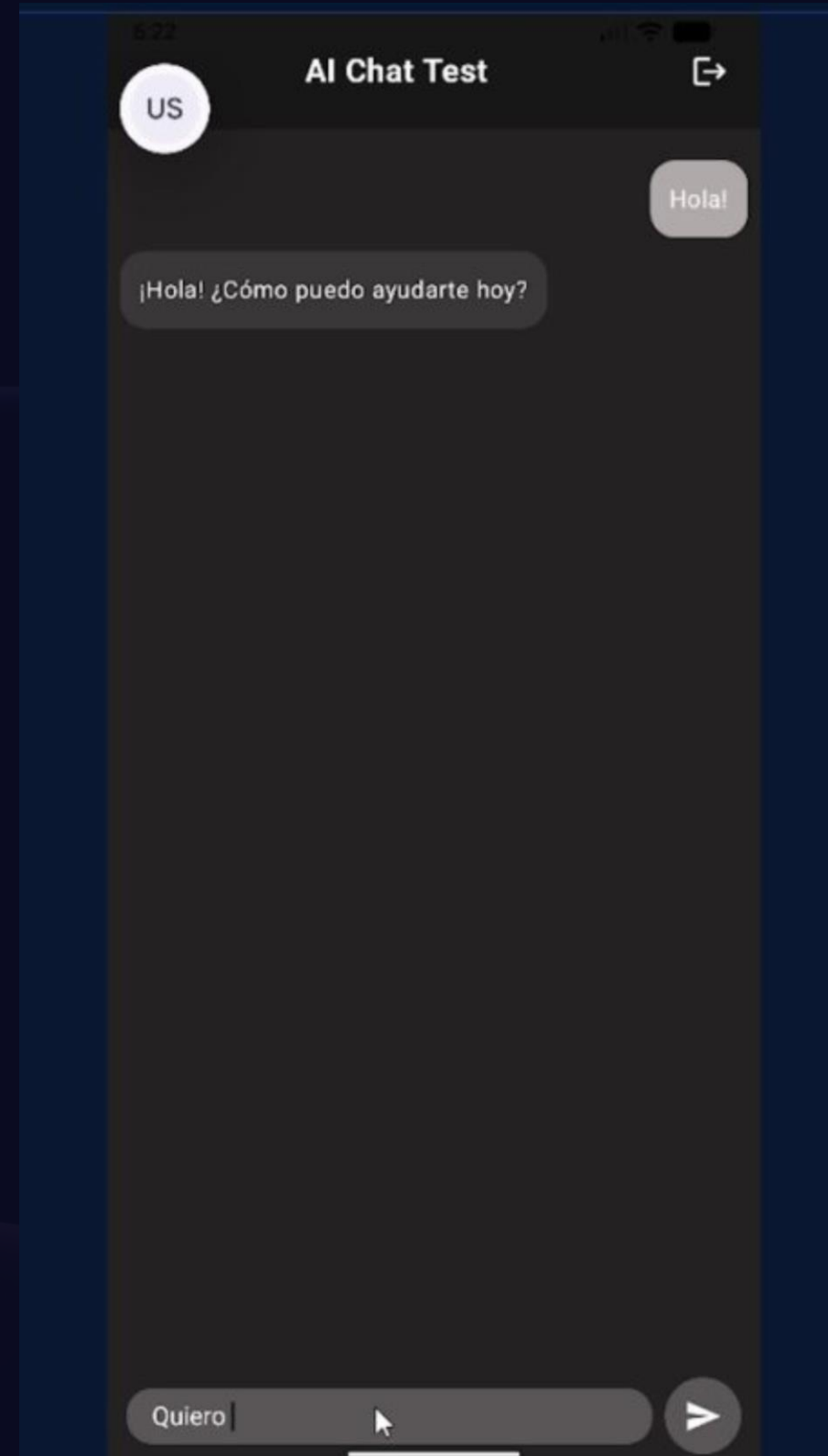
Demostración de funcionamiento



QR Repositorio de
GitHub (App Android)



QR Repositorio de
GitHub (App Web)



Mejoras

1. Se añadió un sistema de *"TextFormatter"* exclusivo para Titulos, Subtitulos, entre otros.
2. Se Elimino la burbuja de respuesta de la IA y ahora abarca todo el espacio de la pantalla para mejorar su apartado visual.

```
@Composable
fun formatChatText(text: String): AnnotatedString {
    val builder = AnnotatedString.Builder()
    val lines = text.split(...delimiters = "\n")
    for (line in lines) {
        when {
            // Título principal: **Titulo**
            line.matches(Regex(pattern = "^\\s*\\s*(.+)?\\s*\\s*$")) -> {
                val title = Regex(pattern = "^\\s*\\s*(.+)?\\s*\\s*$").find(input = line)?.groupValues?.get(1) ?: line
                builder.pushStyle(style = SpanStyle(fontWeight = FontWeight.Bold, fontSize = 18.sp, color = Color(color = 0xFFB0AAAA)))
                builder.append(title + "\n")
                builder.pop()
            }
            // Subtitulo: ## Subtitulo
            line.startsWith(prefix = "##") -> {
                builder.pushStyle(style = SpanStyle(fontWeight = FontWeight.Bold, fontSize = 16.sp, color = Color(color = 0xFFB0AAAA)))
                builder.append(line.removePrefix(prefix = "## ") + "\n")
                builder.pop()
            }
            // Lista simple: - item
            line.trim().startsWith(prefix = "- ") -> {
                builder.pushStyle(style = SpanStyle(fontSize = 15.sp, color = Color.White))
                builder.append("• " + line.trim().removePrefix(prefix = "- ") + "\n")
                builder.pop()
            }
        }
    }
}
```

AI Chat Test

Ahora un titulo como respuesta tipo ****Titulo**** y luego haz un texto de ejemplo con subtitulo y Titulo, el subtitulo es **##subtitulo** para que entiendas.

Explorando Nuevos Horizontes: El Futuro de la Energía Renovable

La Revolución Energética del Siglo XXI

En las últimas décadas, el mundo ha sido testigo de un cambio significativo en la forma en que producimos y consumimos energía. La necesidad de reducir la dependencia de los combustibles fósiles y mitigar el cambio climático ha impulsado el desarrollo de tecnologías de energía renovable. Desde la energía solar hasta la eólica, estas fuentes están transformando no solo el panorama energético, sino también la economía global.

Uno de los aspectos más fascinantes de esta revolución energética es cómo las innovaciones han permitido que comunidades enteras se vuelvan autosuficientes. Por ejemplo, en regiones remotas, los sistemas de energía solar descentralizados proporcionan electricidad a hogares que antes no tenían acceso a la red eléctrica. Esto no solo mejora la calidad de vida, sino que también fomenta el desarrollo económico en áreas rurales.

A medida que avanzamos hacia un futuro más

Ask Anything





Conclusión: Aprendizajes y Futuro

AIAppTest demostró la conexión práctica entre una interfaz móvil y un backend que orquesta llamadas a modelos de IA.

Autenticación JWT

Manejo seguro de sesiones y acceso a recursos protegidos.

Arquitectura Modular

Separación de capas para facilitar la iteración y evolución del proyecto.

Escalabilidad

Base flexible para añadir nuevas funcionalidades y plataformas.

Este proyecto sienta las bases para futuros desarrollos, permitiendo una adaptación y escalabilidad eficientes.