Proyecto TalaTrivia

Introducción

El presente documento describe el diseño técnico y funcional de TalaTrivia, un sistema backend orientado a gestionar un juego de preguntas y respuestas enfocado en temas de recursos humanos. La plataforma permite administrar bases de datos de usuarios, preguntas y trivias, además de procesar la participación de los jugadores y generar rankings de desempeño.

El objetivo principal de este documento es presentar una visión estructurada de la solución, definiendo los componentes de la arquitectura, los modelos de datos utilizados, los flujos operativos y las decisiones técnicas que respaldan la implementación del sistema. Con esta información, se busca proveer una referencia clara para desarrolladores, evaluadores técnicos requieran extender, mantener o integrar la API con otros sistemas.

Alcance del proyecto

El alcance del proyecto TalaTrivia se centra exclusivamente en el desarrollo del backend del sistema, contemplando la creación de una API REST que permite gestionar todas las funcionalidades relacionadas con las trivias. Este alcance incluye:

- Estructura de datos necesaria para registrar usuarios, preguntas, trivias y participaciones.
- Exposición de endpoints para administrar contenido, jugar trivias y obtener resultados.
- Implementación de lógica para validar respuestas y calcular puntajes.
- Persistencia de información en una base de datos PostgreSQL.
- Generación de un ranking por trivia en función del desempeño de los usuarios.
- Uso de Docker para facilitar la ejecución del entorno de base de datos.

El alcance no incluye aspectos como:

- Interfaces gráficas o frontend.
- Autenticación o roles de usuario.
- Gestión avanzada de seguridad, caching o despliegue en la nube.
- Motor de recomendación de preguntas o analítica avanzada.

Estas exclusiones permiten delimitar la responsabilidad del proyecto, asegurando un desarrollo claro, ordenado y enfocado en los requerimientos centrales.

Objetivos del Proyecto

Objetivo General

Desarrollar una **API backend robusta, modular y escalable** para gestionar el flujo completo del juego TalaTrivia, permitiendo administrar usuarios, preguntas, trivias, participaciones y ranking, garantizando una experiencia fluida, confiable y fácilmente integrable con cualquier cliente o frontend.

Objetivos Específicos

1. Gestión de Usuarios

- Implementar endpoints para crear y listar usuarios.
- Garantizar que cada usuario cuente con un identificador único, nombre y correo electrónico.
- Preparar la base para futuras extensiones como autenticación y perfiles.

2. Gestión de Preguntas

- Crear un módulo para registrar preguntas con:
  - Enunciado
  - Opciones de respuesta (JSON)
  - Respuesta correcta

- Nivel de dificultad (fácil, medio, difícil)
- Estandarizar puntajes según dificultad:
  - Fácil → 1 punto
  - Medio → 2 puntos
  - Difícil → 3 puntos

- Proveer endpoints para crear, listar y consultar preguntas.

3. Gestión de Trivias

- Desarrollar una entidad “Trivia” que permita:
  - Crear trivias con nombre y descripción.
  - Asociar un conjunto de preguntas seleccionadas previamente.
  - Asignar usuarios participantes.

- Implementar endpoints para la creación y visualización de una trivia y sus elementos asociados.

4. Participación en Trivias

- Permitir que cada usuario:

    - Consulte las trivias asignadas para jugar.
    - Acceda a las preguntas sin revelar las respuestas correctas.
    - Envíe sus respuestas mediante la API.

- Implementar la lógica de validación de respuestas:
    - Comparar respuestas enviadas con las respuestas correctas.
    - Calcular el puntaje total según dificultad.

- Registrar cada participación con fecha/hora, respuestas y puntaje.

5. Sistema de Ranking

- Implementar un módulo para generar rankings por trivia.
- Ordenar usuarios según el puntaje obtenido.
- Mostrar información relevante como:
    - Nombre del usuario
    - Puntaje final
    - Fecha/hora del intento (created_at)

- Proveer un endpoint para consumir esta información.

6. Diseño Técnico y Buenas Prácticas

- Implementar el proyecto siguiendo una arquitectura modular basada en:
    - Routers (endpoints)
    - Models (SQLAlchemy)
    - Schemas (Pydantic)
    - Services (lógica de negocio)

- Utilizar PostgreSQL como base de datos relacional.
- Ejecutar la base mediante Docker, facilitando la portabilidad y despliegue.
- Garantizar validaciones, manejo de errores y consistencia en toda la API.

Arquitectura del Proyecto

La arquitectura de *TalaTrivia* está diseñada para garantizar modularidad, mantenibilidad y claridad en la separación de responsabilidades. Para ello, se utiliza una arquitectura desacoplada y mantenible organizada mediante routers, servicios y modelos, lo que permite estructurar el backend de forma ordenada y escalable, y facilita su eventual evolución a componentes independientes sin necesidad de reescribir la aplicación completa.

La API ha sido desarrollada con FastAPI, un framework que favorece la construcción de servicios ligeros, altamente organizados y con un fuerte enfoque en tipado, validación y rendimiento. Aunque TalaTrivia se implementa como un servicio monolítico, su estructura modular refleja principios comunes en arquitecturas orientadas a microservicios, tales como:

- Separación clara entre capas.
- Independencia entre recursos.
- Routers dedicados para cada dominio funcional
- Modelos y esquemas reutilizables
- Posibilidad de escalar o dividir componentes en el futuro.

Este enfoque permite que, si el sistema creciera, podría evolucionar hacia una arquitectura distribuida más compleja, sin necesidad de modificar significativamente el diseño inicial.

Componentes principales de la arquitectura

El sistema está compuesto por los siguientes elementos:

1. Cliente / Frontend

   Consume los endpoints de la API mediante HTTP/JSON desde herramientas como Swagger UI, Postman o un cliente web externo.

2. API Backend – FastAPI

   Gestiona:

   - Endpoints REST
   - Validación de datos (Pydantic)
   - Lógica de negocio (servicios)
   - Orquestación de modelos y operaciones de persistencia.

   Los routers representan áreas funcionales independientes:

   - /users → gestión de usuarios
   - /questions → preguntas
   - /trivias → trivias y participación
   - /ranking → puntajes y resultados

3. Capa de Persistencia – SQLAlchemy

   Se encarga del mapeo objeto-relacional entre modelos Python y tablas de PostgreSQL. Los modelos integrados en esta aplicación son los siguientes:

   - Modelo: User

Atributo	Tipo de dato	Descripción
id	Integer (PK)	Identificador único del usuario.
name	String	Nombre completo del usuario.
email	String	Correo electrónico del usuario; debe ser único.

   - Modelo: Question

Atributo	Tipo de dato	Descripción
id	Integer (PK)	Identificador único de la pregunta.
text	String	Enunciado de la pregunta.
difficulty	Enum/String	Nivel de dificultad: easy, medium, hard.
options	JSON	Conjunto de opciones de respuesta disponibles.
correct_option	String	Opción correcta entre las disponibles.

   - Modelo: Trivia

Atributo	Tipo de dato	Descripción
id	Integer (PK)	Identificador único de la trivia.
name	String	Nombre o título de la trivia.
description	String	Descripción general del contenido o propósito.

   - Modelo: TriviaQuestion

Atributo	Tipo de dato	Descripción
id	Integer (PK)	Identificador único del registro.
trivia_id	Integer (FK)	Referencia a la trivia a la que pertenece la pregunta.
question_id	Integer (FK)	Referencia a la pregunta incluida en la trivia.
order	Integer	Posición de la pregunta dentro de la trivia.

   - Modelo: TriviaAssignment

Atributo	Tipo de dato	Descripción
id	Integer (PK)	Identificador único del registro.
trivia_id	Integer (FK)	Trivia asignada al usuario.
user_id	Integer (FK)	Usuario asignado a la trivia.

- Modelo: Participation

Atributo	Tipo de dato	Descripción
id	Integer (PK)	Identificador único de la participación.
trivia_id	Integer (FK)	ID de la trivia en la que participa el usuario.
user_id	Integer (FK)	ID del usuario que responde la trivia.
answers	JSON	Respuestas enviadas por el usuario (pregunta → respuesta).
score	Integer	Puntaje obtenido según las respuestas correctas.
created_at	Timestamp	Fecha y hora del intento; útil para ranking

4. Base de Datos – PostgreSQL

   Almacena entidades del dominio y aprovecha el tipo JSONB para guardar opciones de preguntas y respuestas enviadas.

5. Infraestructura – Docker

   Se utiliza Docker para ejecutar PostgreSQL en un contenedor aislado, lo que facilita la portabilidad y el despliegue del entorno de desarrollo sin necesidad de instalar la base de datos de forma manual en el sistema operativo del usuario.

Interacción entre componentes

El flujo de trabajo entre los componentes es el siguiente:

1. El cliente envía solicitudes HTTP a la API.
2. FastAPI procesa la entrada y ejecuta la lógica de negocio.
3. SQLAlchemy realiza operaciones sobre PostgreSQL.
4. La respuesta es devuelta al cliente en formato JSON.

Este ciclo se repite para cada operación del sistema, manteniendo las capas separadas y permitiendo una fácil depuración y mantenimiento.

Diagramas de la solución

Para complementar la arquitectura, se agregan los siguientes diagramas:

- **Diagrama de Arquitectura**
  Representa las capas principales y su interacción.

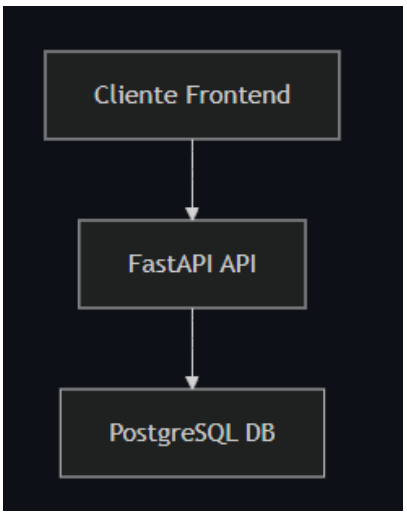

Imagen tomada del Readme

- **Diagrama de Componentes**
  Muestra routers, servicios y modelos, así como su conexión con la base de datos.

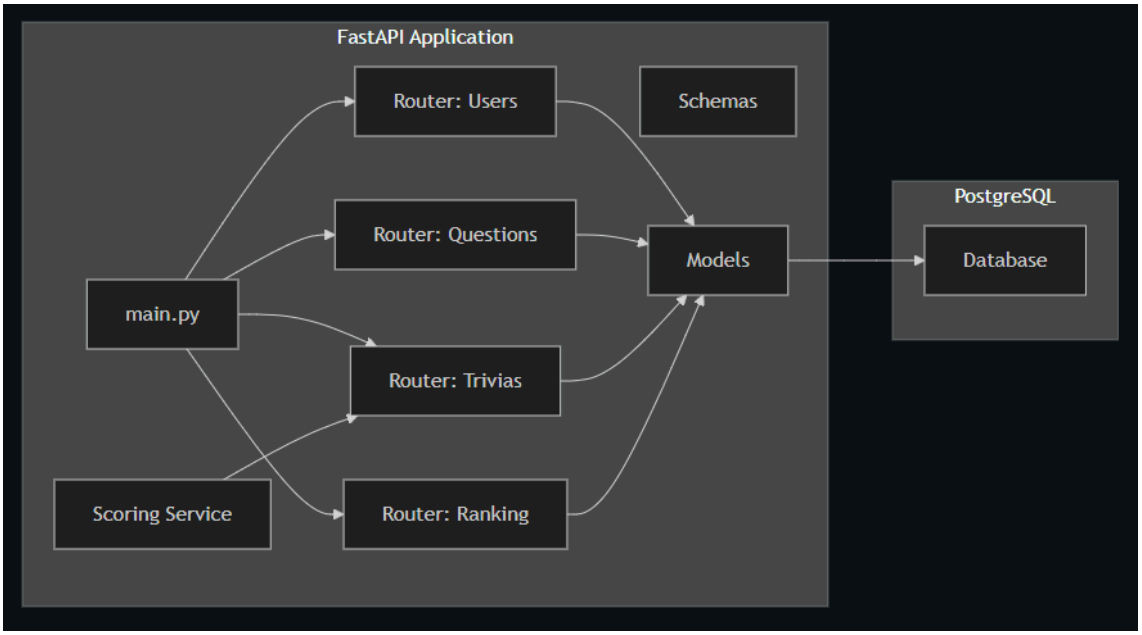

Imagen tomada del Readme

- **Diagrama ER (Entidad–Relación)**
  Representa la estructura de las tablas y sus relaciones.

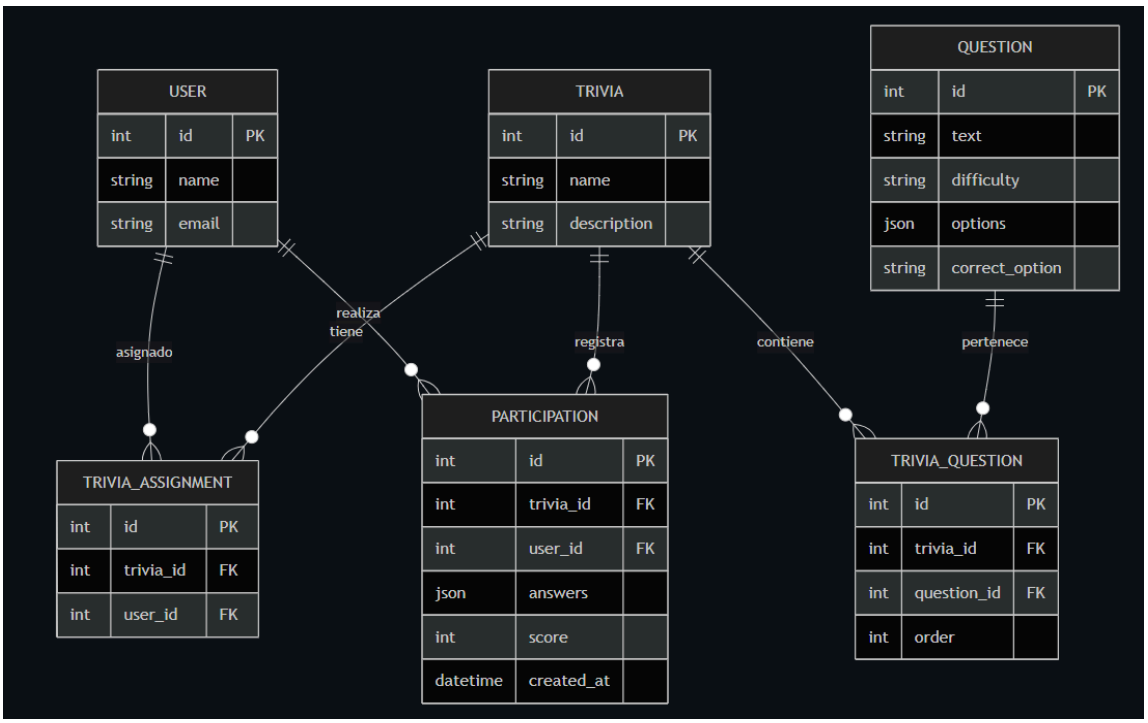

Imagen tomada del Readme

- **Diagrama de Secuencia (flujo Jugar Trivia)**
  Explica el ciclo completo desde la solicitud de la trivia hasta el cálculo del puntaje.

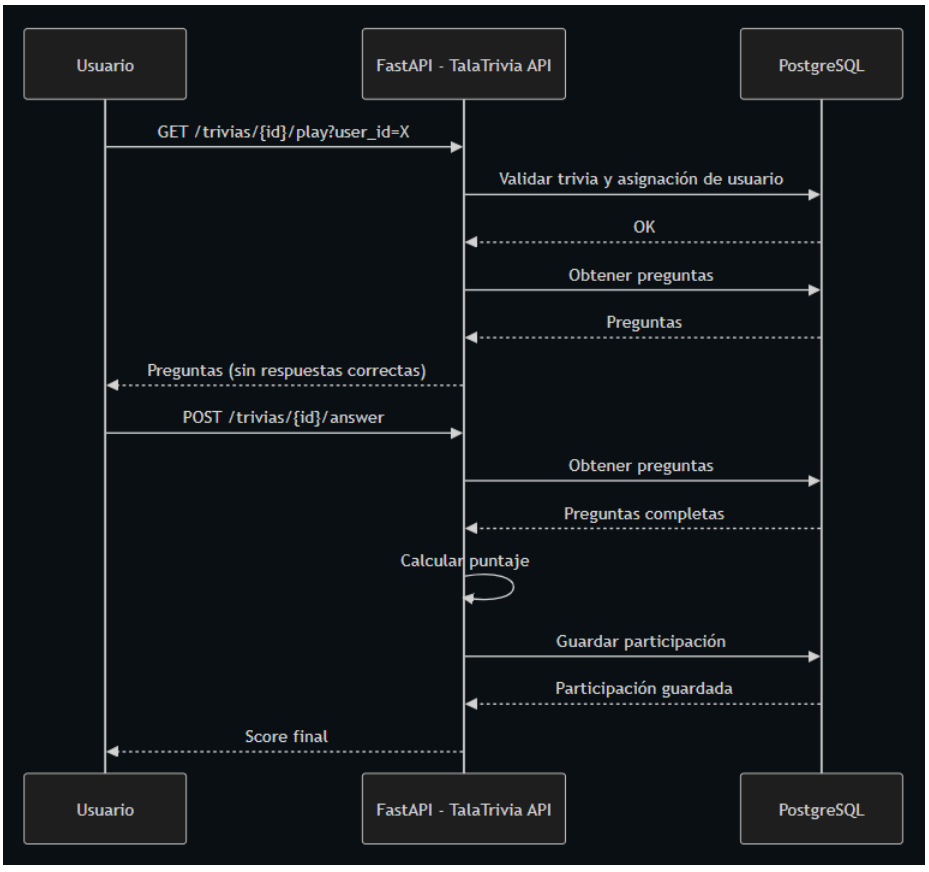

Imagen tomada del Readme

Plan de Pruebas TalaTrivia

1. Crear algunos **usuarios** (`/users/`).
2. Crear varias **preguntas** (`/questions/`).
3. Crear una **trivia** asignando:
    - `question_ids`: IDs de preguntas
    - `user_ids`: IDs de usuarios

4. Para cada usuario:
    - Usa `GET /trivias/{id}/play?user_id=X` para ver la trivia.
    - Usar `POST /trivias/{id}/answer` con distintos `answers` para generar participaciones con distintos puntajes.

5. Finalmente, gatillar: `GET /ranking/{trivia_id}`