Proyecto Integrador - Ruta Básica

Orbis IA Detector (MVP)

Objetivo general

Desarrollar un MVP que permita **detectar contenido generado por Inteligencia Artificial en las entregas de los coders**, utilizando la API de OpenAI, almacenando los resultados en una base de datos relacional y brindando al Team Leader (TL) una interfaz para consultar los análisis, priorizar entregas sospechosas y registrar observaciones mediante feedback.

Objetivos específicos

- **Simular el flujo de Moodle** permitiendo que un estudiante registre su nombre, el nombre del entrenamiento y suba un archivo como entrega.
- **Procesar automáticamente los archivos subidos** (PDF inicialmente, con posibilidad de expandirse a otros formatos).
- **Analizar el contenido con OpenAI** mediante un prompt estandarizado que devuelva:
  - porcentaje de IA detectada,
  - un valor booleano (true/false) según el umbral del 50%,
  - una explicación breve (≤200 caracteres).
- **Almacenar los resultados en la base de datos**, relacionando cada entrega con su análisis.
- **Implementar un panel para el TL** que muestre la lista de entregas, resaltando si presentan IA detectada.

- **Proveer un detalle de análisis en modal**, donde el TL pueda visualizar el porcentaje, explicación y registrar feedback.
- **Permitir la gestión de feedbacks**, incluyendo la creación, consulta y eliminación de observaciones asociadas a un análisis.
- **Establecer una arquitectura escalable** que facilite futuras mejoras como filtros por módulo/semana/coder e integración directa con Moodle.

Planteamiento del problema

En los entornos de educación, los coders cuentan con múltiples herramientas basadas en Inteligencia Artificial que pueden generar textos de manera automática. Esto representa un reto para los Team Leaders (TL), ya que dificulta evaluar de forma objetiva la autoría y el esfuerzo real de cada entrega.

Actualmente, las plataformas de gestión de aprendizaje como **Moodle** permiten la entrega de archivos, pero **no ofrecen un sistema automatizado que identifique contenido generado por IA**. Esto obliga a los TL a revisar manualmente cada documento, incrementando el tiempo de evaluación y reduciendo la capacidad de detectar casos sospechosos.

Además, al no existir una **trazabilidad clara** (entrega → análisis → retroalimentación), se pierde la oportunidad de tener un registro sistemático de los casos detectados y del feedback otorgado a los estudiantes.

Por lo tanto, se hace necesario **diseñar e implementar una solución mínima viable (MVP)** que:

- Automatice el análisis de las entregas de los coders.
- Detecte el nivel de participación de IA mediante un porcentaje y una explicación breve.
- Almacene los resultados en una base de datos relacional.
- Proporcione a los TL una interfaz sencilla para visualizar los casos sospechosos y añadir feedback.

Alcance del proyecto

Dentro del alcance (MVP)

- **Simulación de Moodle**: página para que el estudiante registre su nombre, el nombre del entrenamiento y suba un archivo (PDF como formato inicial).

- **Procesamiento de entregas**: almacenamiento automático del archivo en el servidor y registro en la base de datos.
- **Análisis con IA**: integración con la API de OpenAI mediante un prompt estandarizado que retorna:
  - porcentaje de contenido generado por IA,
  - indicador booleano (true/false según el umbral del 50%),
  - explicación breve (máx. 200 caracteres).
- **Gestión de resultados en base de datos**: persistencia en tablas relacionales (**Submissions, Analysis, Feedback**) con relaciones claras entre cada entrega, análisis y retroalimentación.
- **Panel para Team Leaders (TL)**:
  - Tabla con lista de entregas y estado de detección de IA.
  - Modal con detalle del análisis (porcentaje, explicación, datos del estudiante, feedback).
  - Funcionalidad para agregar, visualizar y eliminar feedback asociado a un análisis.

Fuera del alcance (para futuras versiones)

- **Filtros avanzados por módulo y semana** (ya que no se almacenan aún esos datos en cada entrega).
- **Sistema de login y roles** (actualmente se simula el flujo de Coder y TL sin autenticación).
- **Soporte para múltiples formatos de archivo** (por ahora solo PDF).
- **Integración real con Moodle** (se trabaja únicamente con simulación).

Flujo general

1. **Coder** sube archivo → (Frontend Coder)
2. **Backend** recibe → guarda en *Submissions*
3. Backend llama a **OpenAI API** → guarda respuesta en *Analysis*
4. **TL** accede al panel → consulta lista de análisis (fetch a BD)
5. Al abrir un análisis: se muestra modal con detalle + feedback editable

Historias de usuario

Coder

- Como **Coder**, quiero **registrar mi nombre y el entrenamiento** al subir un archivo, para que quede identificado en la base de datos
- Como **Coder**, quiero **subir un archivo PDF como entrega**, para que pueda ser analizado automáticamente por la IA.

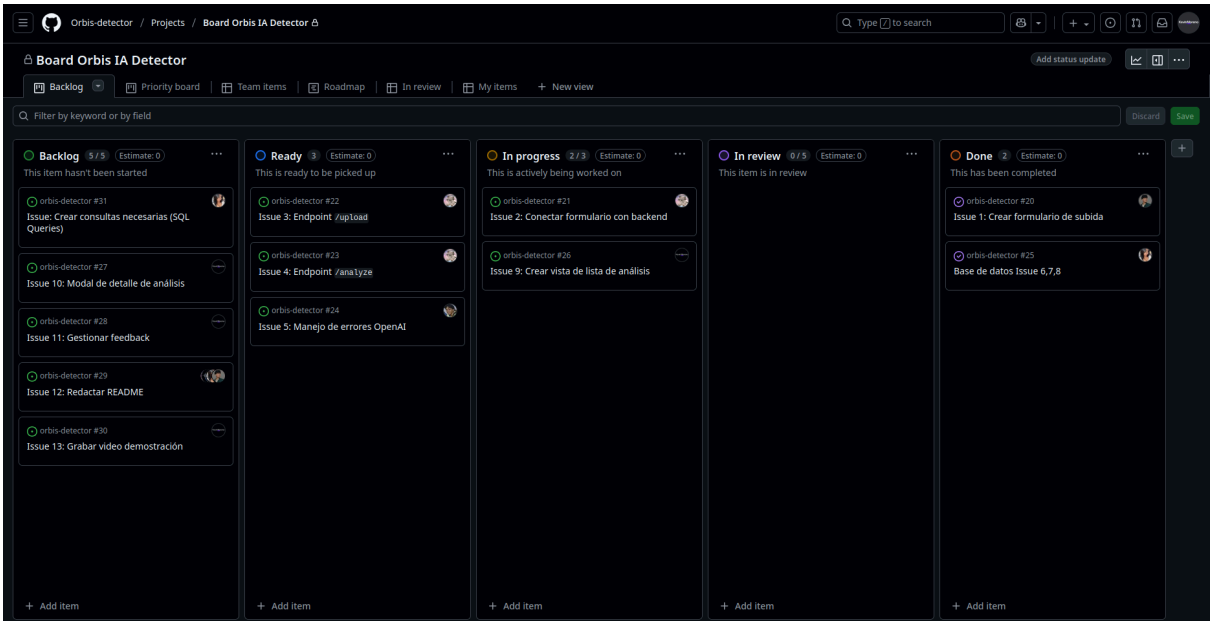
Sistema

- Como **sistema**, debo **almacenar el archivo en el servidor y registrar sus datos** en la tabla *Submissions*, para garantizar trazabilidad de las entregas.
- Como **sistema**, debo **enviar el archivo a la API de OpenAI con un prompt definido**, para obtener el porcentaje de IA detectada, el booleano y una explicación breve.
- Como **sistema**, debo **guardar el resultado del análisis** (IA %, IA detectada, explicación) en la tabla *Analysis*, asociado al `submission_id`.

Team Leader (TL)

- Como **TL**, quiero **ver una lista de todas las entregas analizadas**, para identificar cuáles contienen IA detectada.
- Como **TL**, quiero **abrir un detalle de cada análisis en un modal**, para visualizar el porcentaje, explicación y datos del coder.
- Como **TL**, quiero **gestionar feedback sobre un análisis** (crear, actualizar o eliminar), para registrar y mantener observaciones claras sobre el trabajo del estudiante.

Evidencia de aplicación de la metodología SCRUM



Tablero de trabajo **GitHub Projects** como tablero Kanban para gestionar el backlog.

Herramienta utilizada

Se implementó **GitHub Projects** como tablero Kanban para gestionar el backlog y aplicar los principios de la metodología SCRUM.

Flujo de trabajo definido

Se trabajó con las siguientes columnas:

- **Backlog:** Tareas identificadas, aún no iniciadas.
- **Ready:** Tareas priorizadas, listas para iniciar.
- **In Progress:** Tareas en desarrollo.
- **In Review:** Tareas terminadas, en revisión.
- **Done:** Tareas finalizadas y aprobadas.

Backlog

Epic 1 – Frontend Coder (Simulación Moodle)

Feature: Página de subida de archivo

- Task: Crear formulario con campos `name_coder`, `name_training` y selector de archivo.
- Task: Validar que se suba un archivo válido (PDF).

- Task: Enviar datos y archivo al backend.

Epic 2 – Backend (Procesamiento y análisis)

Feature: Endpoint `/upload`

- Task: Recibir archivo y datos del coder.
- Task: Guardar archivo en el servidor y registrar en tabla *Submissions*.

Feature: Endpoint `/analyze`

- Task: Implementar integración con API de OpenAI.
- Task: Procesar respuesta JSON y guardar en *Analysis*.
- Task: Manejar errores (API caída, archivo inválido, texto vacío).

Epic 3 – Base de datos

Feature: Modelo relacional de 3 tablas

- Task: Crear tabla *Submissions*.
- Task: Crear tabla *Analysis*.
- Task: Crear tabla *Feedback*.
- Task: Definir relaciones y llaves foráneas.
- Task: Definir consultas de datos

Epic 4 – Frontend TL (Panel de gestión)

Feature: Vista principal de análisis

- Task: Crear tabla con lista de entregas (coder, entrenamiento, IA detectada).
- Task: Implementar consumo de `GET /analysis`.

Feature: Modal de detalle

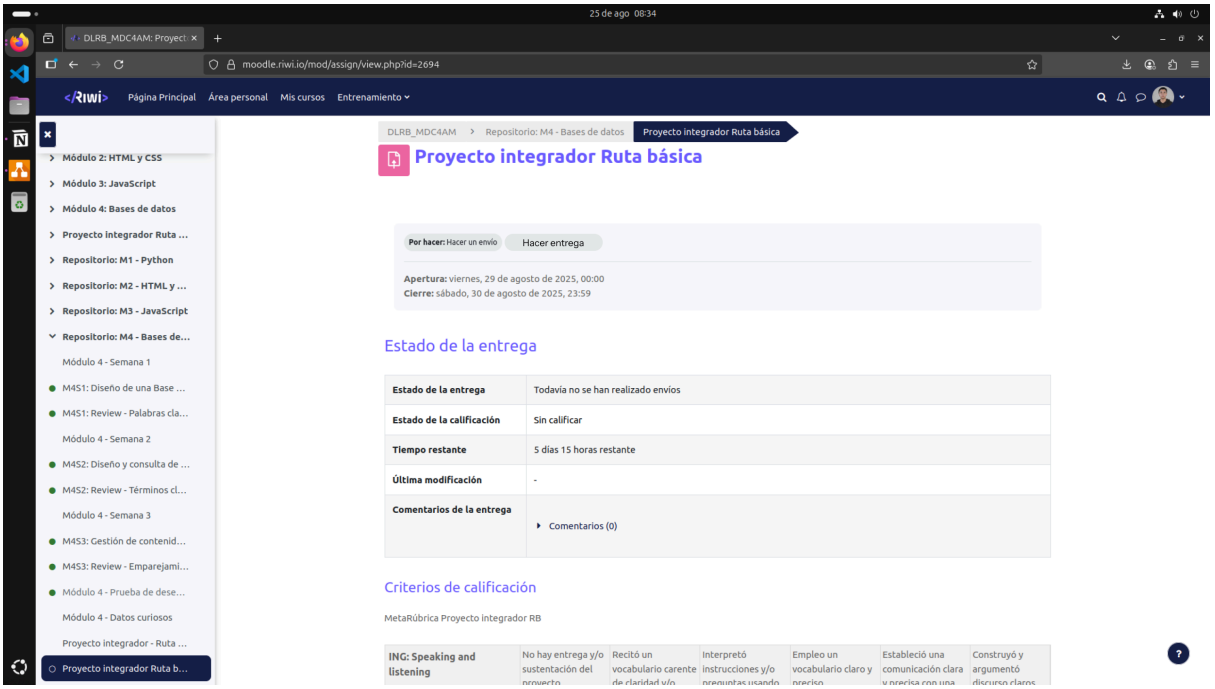
- Task: Mostrar % IA, explicación, coder y entrenamiento.
- Task: Crear textarea para feedback.
- Task: Conectar con `POST /feedback` para guardar feedback.
- Task: Conectar con `DELETE /feedback/:id` para eliminar feedback.

Epic 5 – Documentación y Demo

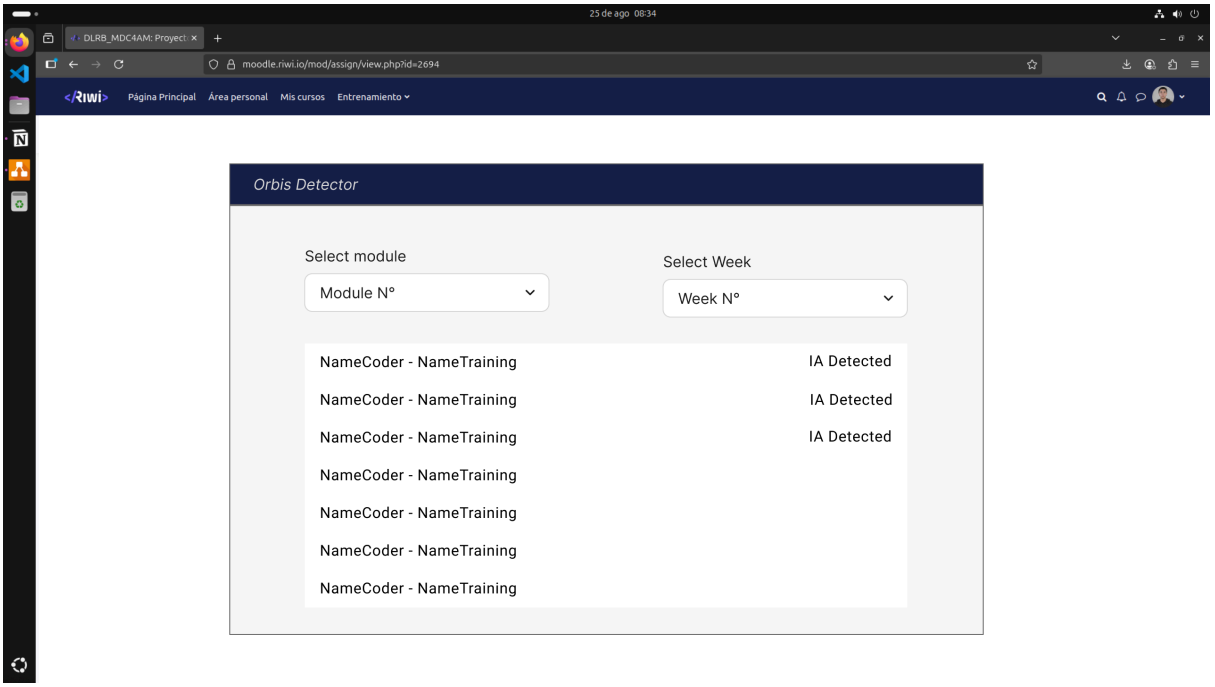
Feature: Documentación del proyecto

- Task: Redactar README con instrucciones de instalación y uso.
- Task: Incluir modelo de datos y endpoints.
- Task: Grabar video corto mostrando el flujo completo (subida → análisis → panel TL).

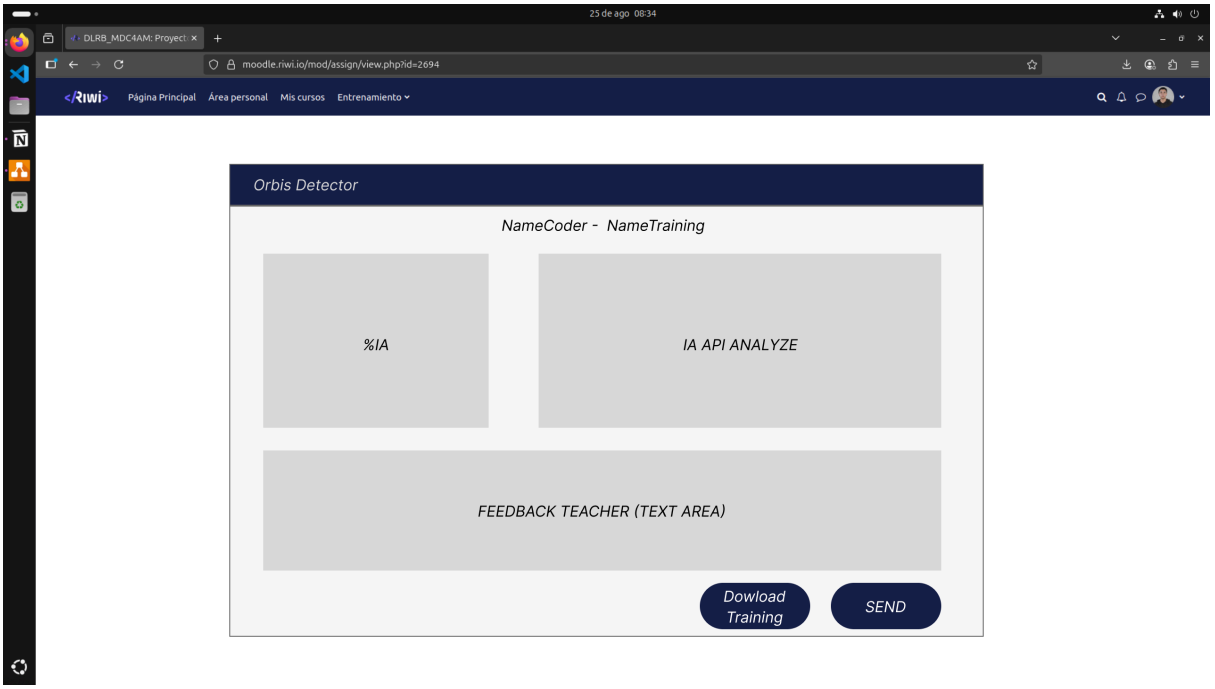
Prototipo visual del proyecto (Coder y TL).



(**Frontend Coder**) Página de subida de archivo.



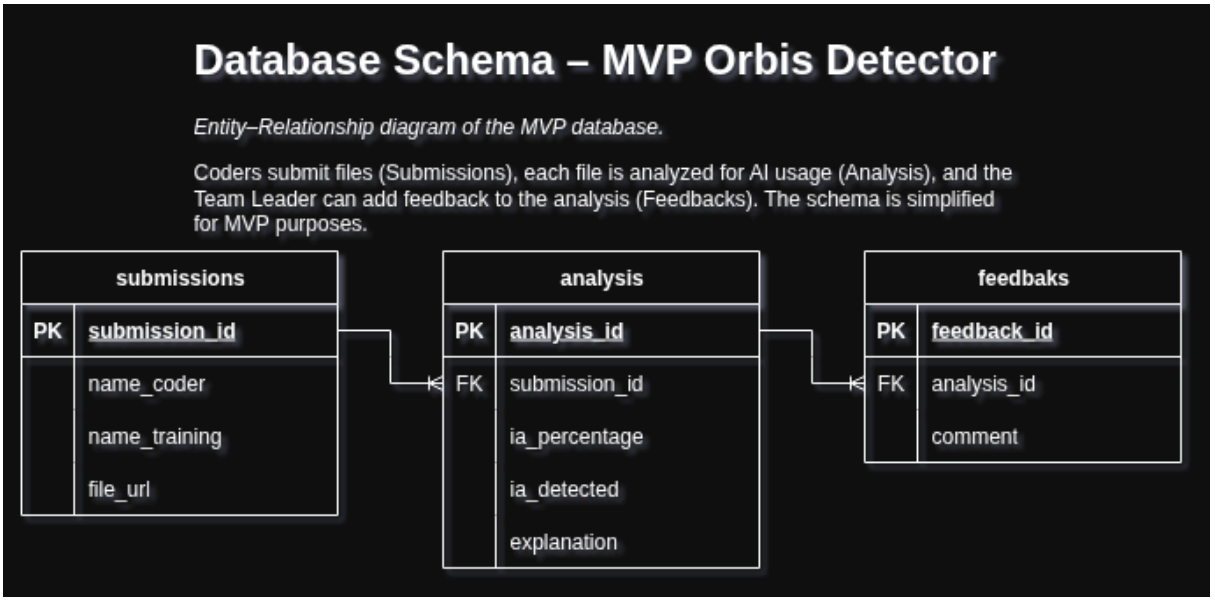
(**Panel TL**) Vista con filtros (módulo, semana), Lista priorizada (IA detected primero).



(**Panel TL**) Detalle con IA %, explicación y campo feedback. Guardar feedback en DB.

Documentación técnica

Modelo relacional



- **Submissions**
  - `submission_id` **(PK)** int
  - `name_coder` VARCHAR(100)
  - `name_training` VARCHAR(100)
  - `file_url` VARCHAR(100)
- **Analysis**
  - `analysis_id` **(PK)** INT
  - `submission_id` **(FK → Submission)** INT
  - `ia_percentage` DECIMAL(3,0)
  - `ia_detected` BOLEAN
  - `explanation` TEXT
- **Feedbacks**
  - `feedback_id` **(PK)** INT
  - `analysis_id` **(FK → Analysis)** INT
  - `comment` TEXT

- Cada **Submission** tiene exactamente **un Analysis**.
- Cada **Analysis** puede tener **cero o un Feedback**.

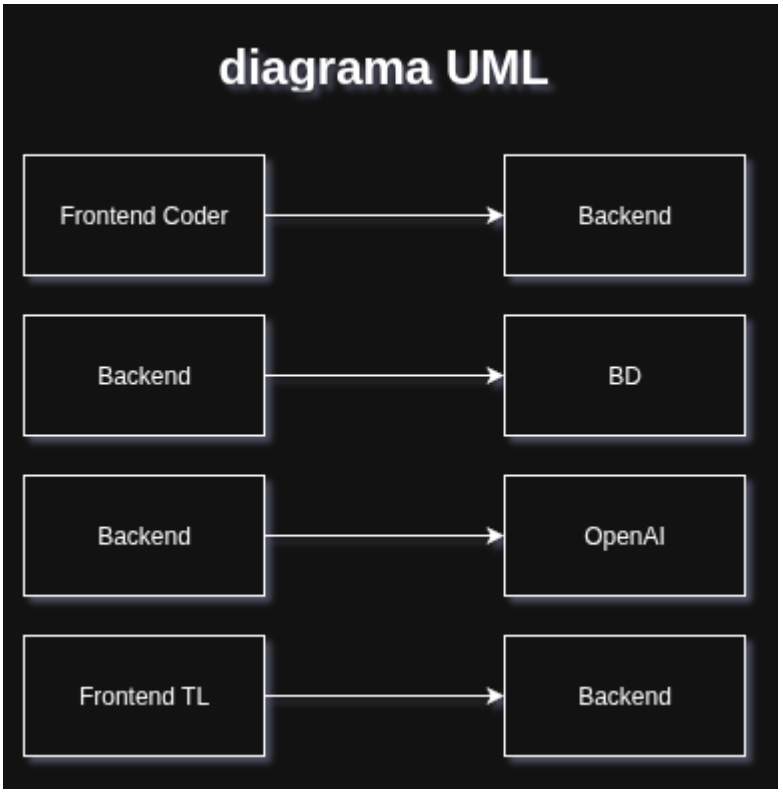
SQL Schema Draft

```
CREATE TABLE Submissions (
    submission_id INT AUTO_INCREMENT PRIMARY KEY,
    name_coder VARCHAR(100) NOT NULL,
    name_training VARCHAR(100) NOT NULL,
    file_url VARCHAR(255) NOT NULL
);

CREATE TABLE Analysis (
    analysis_id INT AUTO_INCREMENT PRIMARY KEY,
    submission_id INT NOT NULL UNIQUE,
    ia_percentage DECIMAL(5,2) NOT NULL,
    ia_detected BOOLEAN DEFAULT 0,
    explanation TEXT,
    FOREIGN KEY (submission_id) REFERENCES Submissions(submission_i
d)
        ON DELETE CASCADE
);

CREATE TABLE Feedbacks (
    feedback_id INT AUTO_INCREMENT PRIMARY KEY,
    analysis_id INT NOT NULL UNIQUE,
    comment TEXT,
    FOREIGN KEY (analysis_id) REFERENCES Analysis(analysis_id)
        ON DELETE CASCADE
);
```

Diagrama de componentes



- El **Frontend Coder** se comunica con el **Backend** (endpoint `/upload`).
- El **Backend** almacena datos en la **BD** y consulta resultados.
- El **Backend** también llama a la **API de OpenAI** para analizar el texto.
- El **Frontend TL** consulta al **Backend** (`/analysis`, `/feedback`) para mostrar resultados y guardar los feedbacks.

Componentes:

- **Frontend Coder (Simulación Moodle):** HTML + CSS + JavaScript.
- **Backend:** Node.js con Express.
- **Base de datos:** MySQL.
- **API externa:** OpenAI (API Key + Prompt).
- **Frontend TL (Panel de gestión):** HTML + CSS + JavaScript (Fetch API).FUNCIONES
