**Explicación completa del Backend Proyecto ESP32-CAM + IA**

**Qué es este backend**

El backend es el cerebro del proyecto:

* Recibe frames de video enviados por la ESP32-CAM.
* Procesa cada frame para detectar las señas (simulado ahora con una función de prueba).
* Envía el resultado de la traducción en tiempo real a los clientes conectados.
* Gestiona el modo práctica: envía desafíos, recibe la respuesta del usuario y devuelve feedback (correcto o incorrecto).
* se construyó con Python y FastAPI, usando WebSockets para comunicación en tiempo real.

Arquitectura del backend

|  |  |
| --- | --- |
| Modulo | Función |
| main.py | Punto de entrada del servidor. Inicializa FastAPI y define endpoints WebSocket. |
| ws\_manager.py | Administra todas las conexiones WebSocket, tanto de la ESP32 como de los clientes web/móvil. |
| esp\_handler.py | Maneja la recepción de frames desde la ESP32, decodifica las imágenes JPEG y envía los frames al motor de inferencia. |
| client\_handler.py | Envía las traducciones a los clientes conectados (web o móvil). |
| frames.py | Funciones para procesar y convertir los frames JPEG en un formato que la IA pueda leer. |
| ia\_stub.py | Simula la inferencia de IA; se puede reemplazar por un modelo real (CNN, LSTM, Transformer, etc.). |
| utils.py | Funciones auxiliares: logs, control de repetición de letras, etc. |
| inference.py | Módulo de inferencia principal. Actualmente simula la detección de gestos de la mano, devolviendo letras o palabras aleatorias con un nivel de confianza. En el futuro, se conectará a un modelo real de visión por computadora. |

**Qué hace cada parte en el flujo**

A. Recepción de frames (ESP32-CAM)

* La ESP32 se conecta por WiFi al backend usando WebSocket.
* Envía los frames JPEG continuamente.
* El backend recibe los frames y los decodifica usando OpenCV.

B. Inferencia IA (simulada)

* Los frames decodificados se envían al motor de IA.
* Actualmente está simulado con una función que devuelve letras o frases aleatorias.
* La función controla repetición: si la letra no cambia, no la vuelve a enviar varias veces seguidas.

C. Envío al cliente (app o web)

* Los clientes también se conectan por WebSocket.
* Cada vez que se obtiene una “traducción” del frame, se envía al cliente.
* En modo práctica, el backend compara la traducción con el desafío y envía feedback.

**Requisitos de instalación**

A. Python

Versión recomendada: 3.10 o superior

Verificar instalación: python –version

**Crear entorno virtual (opcional pero recomendado)**

python -m venv venv

# Activar en Windows

venv\Scripts\activate

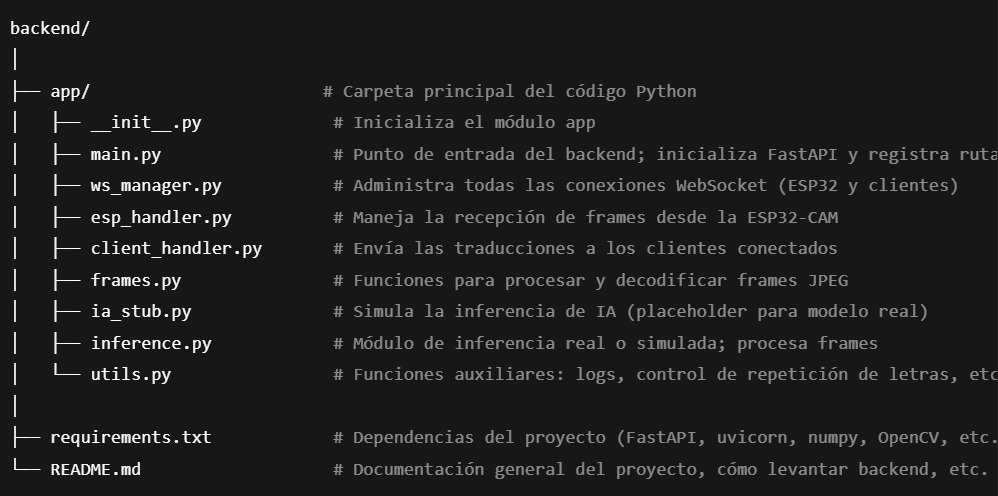
**Instalar dependencias**

pip install --upgrade pip

pip install fastapi uvicorn[standard] opencv-python-headless numpy

**Opcional si después se agrega IA real:** pip install mediapipe

**Archivos del proyecto**

****

**Cómo levantar el backend**

Ejecutar: python -m uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

--reload → reinicia automáticamente si cambias código.

--host 0.0.0.0 → permite que otros dispositivos de la red local se conecten (ESP32, móvil).

--port 8000 → puerto por defecto, se puede cambiar si hace falta.

**Verificar que está corriendo:** <http://localhost:8000/>

**Se debería ver:** {"status": "running", "msg": "Backend activo"}

**Conectar la ESP32-CAM**

Obtener la IP local de tu PC en la red WiFi: ipconfig

**Configurar el sketch de Arduino IDE:**

const char\* websocket\_host = "10.232.112.82"; // tu IP local

const uint16\_t websocket\_port = 8000;

Subir sketch a ESP32-CAM y abrir monitor serie.

Deberías ver conexión exitosa y logs de envío de frames.