2026/02/01

Principios de Diseño del Sistema

Porque y sus componentes más fundamentales:

1. Introducción general

El propósito de esta iniciativa es dual y bien definido:

1. Entorno de Pruebas y Experimentación**:** El sistema servirá como una plataforma controlada donde los modelos de visión por computadora puedan ser sometidos a pruebas, validación y afinamiento (tuning). Esto permite a los desarrolladores y equipos de investigación evaluar el rendimiento de los modelos en condiciones simuladas antes de una implementación final.

2. Entorno de Producción Final para la Enseñanza**:** Paralelamente, el sistema está concebido para ser un recurso educativo. Al operar en un entorno de producción que simula escenarios reales, facilita la enseñanza práctica de la visión por computadora y el AA.

2. Componentes del sistema:

- Cliente: JavaScript Vanilla

La interfaz está hecha con JavaScript Vanilla, es decir, JavaScript sin frameworks (sin React, sin Vue, etc.). Esto significa que la lógica de la UI, el manejo del DOM y el flujo de interacción se controlan de forma directa, con el objetivo de mantener el cliente liviano, predecible y con el menor costo posible para un sistema que trabaja en tiempo real.

- Contenedor de escritorio: Electron

Esa interfaz corre dentro de Electron, que es una tecnología que permite crear aplicaciones de escritorio usando tecnologías web. En la práctica, Electron actúa como el “cuerpo” del programa: abre la ventana, integra el entorno de escritorio, y permite que la UI en JavaScript se ejecute como una app instalable y offline.

- Captura de imágenes: cámara desde el cliente

Desde el lado del cliente se obtiene la imagen (por ejemplo, frames de cámara). Esa imagen es la materia prima que el sistema envía al backend para ser procesada.

- Backend: Python

La parte que ejecuta la lógica pesada está hecha en Python, principalmente porque es el ecosistema donde más naturalmente se integran librerías y runtimes de modelos de visión por computadora. En este

proyecto, el backend es el responsable de cargar el modelo seleccionado y producir resultados a partir de cada imagen.

- API local: FastAPI

La comunicación entre la app (cliente) y el backend se hace con FastAPI, que es un framework para crear APIs en Python. FastAPI funciona como el “puente” entre ambos mundos: expone endpoints para que el cliente pueda enviar una imagen, seleccionar un modelo, y recibir una respuesta estructurada con los resultados.