🚗 Detección de Somnolencia en Conductores – API Flask

Este proyecto utiliza **visión artificial** para detectar si un conductor está **dormido, somnoliento o activo** a través de una API basada en Flask.
El sistema analiza imágenes en tiempo real y actualiza el estado automáticamente en el navegador sin necesidad de recargar la página.

📌 Características

- ☑ Detección de somnolencia basada en la posición de los párpados
- ☑ API REST en Flask
- ☑ Interfaz web en HTML, CSS y JavaScript
- ☑ Captura automática de imágenes y actualización en tiempo real
- ☑ Compatible con navegadores modernos

📂 Estructura del Proyecto

```
project/
│
├── app/                    # Carpeta principal de la aplicación
│   ├── __init__.py         # Inicialización del paquete Flask
│   ├── routes.py           # Rutas de la API
│   ├── utils.py            # Funciones auxiliares
│   ├── static/             # Archivos estáticos (CSS, JS)
│   │   ├── styles.css      # Estilos CSS
│   │   ├── script.js       # Lógica en frontend (captura y envío de imágenes)
│   ├── templates/          # Plantillas HTML
│   │   ├── index.html      # Página principal
│
├── shape_predictor_68_face_landmarks.dat  # Modelo de landmarks de Dlib
│
├── requirements.txt        # Dependencias del proyecto
│
├── app.py                  # Archivo principal para ejecutar la app Flask
│
└── README.md               # Descripción del proyecto
```

🚀 Instalación y Uso

1️⃣ **Clonar el repositorio**

```
git clone https://github.com/JohanSteeven/deteccion_insomnio.git
cd deteccion_insomnio
```

2 Instalar dependencias

Asegúrate de tener Python 3 instalado y ejecuta:

```
pip install -r requirements.txt
```

3 Ejecutar la aplicación

```
python app.py
```

Esto iniciará el servidor en `http://127.0.0.1:5000/`.

4 Abrir la interfaz en el navegador

Accede a `http://127.0.0.1:5000/` y verás la cámara en vivo.
El estado cambiará automáticamente en función de la detección.

Tecnologías Utilizadas

- **Python** – Backend y procesamiento de imágenes
- **Flask** – Framework para la API
- **OpenCV** – Captura y procesamiento de imágenes
- **Dlib** – Detección de puntos faciales
- **HTML, CSS, JavaScript** – Interfaz web
- **Fetch API** – Envío de imágenes a la API

API Endpoints

Método	Ruta	Descripción
`GET`	`/`	Devuelve la interfaz web
`POST`	`/detect`	Recibe una imagen y devuelve el estado (`Active`, `Drowsy`, `Sleeping`)

Ejemplo de solicitud POST

```
curl -X POST -F "image=@captura.jpg" http://127.0.0.1:5000/detect
```

📥 Respuesta esperada

```
{
    "status": "Despierto :)"
}
```

⚠ Notas

1. **Debe descargarse el archivo** `shape_predictor_68_face_landmarks.dat` y colocarse en la raíz del proyecto.

📌 Mejoras Futuras

- Implementar un modo de entrenamiento para mejorar la precisión
- Hacer que la API funcione con múltiples cámaras

👨‍💻 Autor

📍 Desarrollado por el grupo 4 correspondiente a la materia de construcción de software – Escuela Politécnica Nacional

---