**Reto de Python**

**Integrantes:**

**Desarrollador:** Mario Daniel Esquivel Sandoval

**Investigador:** Mario Isaí Esquivel Sandoval

**Prompts**

Quiero me ayudes a descubrir herramientas para el desarrollo de un programa con las siguientes características: -acceso a la cámara de una laptop -capture imágenes a través de la cámara -integración con una IA que reconozca si hay una persona en la imagen

**Herramientas para desarrollo de un programa de detección de personas con cámara**

Puedo ayudarte a identificar herramientas para desarrollar un programa con las características que mencionas. Tu proyecto necesitará:

1. Acceso a la cámara de una laptop
2. Captura de imágenes
3. Integración con IA para reconocimiento de personas

**Opciones de lenguajes de programación**

Estos lenguajes son particularmente adecuados para tu proyecto:

* **Python**: Excelente opción por su facilidad de uso y abundantes bibliotecas para visión por computadora e IA
* **JavaScript**: Buena alternativa si prefieres desarrollo web
* **C++**: Opción más avanzada para máximo rendimiento

**Bibliotecas para acceso y captura de cámara**

**Python**

* **OpenCV**: Biblioteca potente y popular para procesamiento de imágenes y visión por computadora
* **Pillow**: Para manipulación básica de imágenes
* **VideoCapture**: Biblioteca específica para captura de video

**JavaScript**

* **WebRTC**: API para acceso a cámara en navegadores web
* **MediaDevices API**: Para capturar video desde navegadores

**Soluciones de IA para detección de personas**

**APIs de servicios en la nube**

* **Google Cloud Vision API**: Detección de personas en imágenes
* **Azure Computer Vision**: Servicio de Microsoft para análisis de imágenes
* **Amazon Rekognition**: Servicio de AWS para reconocimiento de imágenes

**Bibliotecas locales**

* **TensorFlow** con modelos preentrenados como SSD MobileNet
* **PyTorch** con modelos como YOLO o Faster R-CNN
* **MediaPipe**: Solución de Google para análisis de video en tiempo real

**Ejemplo básico de implementación en Python**

Si eliges Python (que recomendaría para comenzar), un enfoque básico sería:

1. Usar OpenCV para acceder a la cámara y capturar imágenes
2. Implementar un modelo preentrenado de TensorFlow o usar MediaPipe para detección de personas
3. Desarrollar la interfaz de usuario según tus necesidades

¿Te gustaría que profundice en alguna de estas herramientas específicas o prefieres ejemplos de código para alguna parte concreta del proyecto?

el programa estará desarrollado en python y usaremos OpenCV, usando estas herramientas, podrías hacer un programa que acceda a la cámara y capture una imagen

Claro, puedo ayudarte a crear un programa básico en Python con OpenCV que acceda a la cámara y capture una imagen. Aquí tienes un código funcionando:

Captura de Imagen con OpenCV

Código

import cv2 import os from datetime import datetime def capturar\_imagen(): """ Accede a la cámara, muestra video en vivo y captura una imagen cuando el usuario presiona la tecla 'espacio'. """ # Inicializar la cámara (0 es genera

Este programa en Python utiliza OpenCV para:

1. Acceder a la cámara web de tu laptop
2. Mostrar un video en vivo de lo que está capturando la cámara
3. Capturar una imagen cuando presionas la tecla 'espacio'
4. Guardar la imagen capturada con un nombre basado en la fecha y hora actual

**Cómo funciona el código:**

* cv2.VideoCapture(0) abre la cámara predeterminada (generalmente la integrada en tu laptop)
* El bucle principal captura continuamente frames de video y los muestra
* Cuando presionas la tecla 'espacio', guarda el frame actual como una imagen JPG
* Las imágenes se guardan en una carpeta llamada 'capturas' que se crea automáticamente
* Puedes salir del programa presionando 'q'

**Para ejecutar este programa necesitas:**

1. Tener Python instalado
2. Instalar OpenCV con el comando:

pip install opencv-python

1. Guardar el código como un archivo .py (por ejemplo, camara\_captura.py)
2. Ejecutarlo con python camara\_captura.py

¿Te gustaría que modifique algo de este código? Por ejemplo, podría:

* Cambiar el formato de guardado de las imágenes
* Añadir más opciones como ajustar la resolución
* Integrar una funcionalidad básica de detección de personas

Añade la una funcion que analise si hay una persona en la imagen por medio de vision studio

He añadido la funcionalidad para detectar personas en las imágenes utilizando el detector HOG (Histograma de Gradientes Orientados) que viene incluido en OpenCV. Este es un método bien establecido para la detección de personas. Las principales modificaciones son:

1. **Nueva función detectar\_personas()**: Analiza una imagen y detecta si hay personas en ella utilizando el detector HOG de OpenCV.
2. **Modificación de capturar\_imagen()**: Ahora cuando capturas una imagen con la tecla espacio, automáticamente:
   * Guarda la imagen original
   * Analiza la imagen para detectar personas
   * Muestra el resultado en pantalla
   * Guarda una segunda imagen con los resultados del análisis
3. **Nueva función analizar\_imagen\_existente()**: Permite analizar una imagen que ya tienes guardada, sin necesidad de usar la cámara.
4. **Mejoras en el programa principal**: Ahora puedes:
   * Ejecutar el programa normalmente para usar la cámara: python nombre\_script.py
   * O analizar una imagen existente pasándola como argumento: python nombre\_script.py ruta/a/tu/imagen.jpg

El detector marcará con rectángulos verdes a las personas detectadas en la imagen y mostrará un contador con el número de personas encontradas.

¿Te gustaría hacer alguna otra modificación o mejora al código?

Cambia el programa para que funcione usando Azure Vision Studio

He modificado el programa para que utilice Azure Vision Studio en lugar del detector HOG de OpenCV. Las principales modificaciones son:

**1. Cambios en la configuración**

* Añadí las importaciones necesarias (requests y json) para realizar peticiones a la API de Azure.
* Agregué variables de configuración donde deberás poner tu endpoint y clave de API de Azure Vision.

**2. Nueva función detectar\_personas\_azure()**

* Esta función reemplaza a la antigua detectar\_personas().
* Ahora envía la imagen a Azure Computer Vision API para su análisis.
* Procesa los resultados para detectar objetos de tipo "persona".
* Dibuja rectángulos alrededor de las personas detectadas y muestra la confianza de cada detección.

**3. Modificaciones en capturar\_imagen() y analizar\_imagen\_existente()**

* Ahora estas funciones utilizan la nueva función detectar\_personas\_azure().
* Se actualizaron los mensajes para reflejar que se está usando Azure Vision.

**4. Nueva función verificar\_configuracion\_azure()**

* Comprueba si has configurado correctamente tus credenciales de Azure.
* Muestra una advertencia si no las has configurado.

**Cómo usar el programa:**

1. **Paso 1: Configurar Azure Computer Vision**
   * Necesitas crear una cuenta en Azure y un recurso de Computer Vision.
   * Obtén tu endpoint y clave de API.
   * Reemplaza estas líneas en el código:

python

VISION\_ENDPOINT = "TU\_ENDPOINT\_DE\_AZURE\_VISION" *# Por ejemplo: "https://tu-resource.cognitiveservices.azure.com/"*

VISION\_KEY = "TU\_CLAVE\_DE\_AZURE\_VISION"

1. **Paso 2: Usa el programa como antes**
   * Para capturar y analizar con la cámara: python nombre\_script.py
   * Para analizar una imagen existente: python nombre\_script.py ruta/a/tu/imagen.jpg
2. **Ventajas de usar Azure Vision Studio**
   * Detección mucho más precisa y robusta que el detector HOG local.
   * Puede detectar personas en muchas más posiciones y situaciones.
   * Proporciona un nivel de confianza para cada detección.

¿Necesitas alguna otra modificación o tienes dudas sobre cómo configurar Azure Vision Studio?