TP final: Computer Vision
Materia: Programación Funcional & Scripting
Integrantes: CARRIZO, GUILLAUMET y LLAMOSAS
Deadline: 11/11/2025

Consigna:

1) Crear un módulo que contenga:
    - setOnnxModel(modelName, umbralConfianza, height, width, classNames)
        - modelName: modelo a ejecutar en formato Onnx
        - umbralConfianza: confianza mínima a tener en cuenta
        - height, width: alto y ancho de la imagen a procesar
        - classNames: nombres de las clases que detecta el modelo
    - run(sourceImagePath | sourceImageBuffer, classes): se retorna un json con los datos de la detección: boundingbox, clase detectada y confianza.
        - sourceImagePath | sourceImageBuffer: la imagen sobre la que se correrá el modelo
        - classes: arreglo números enteros que representan las clases que se quieren tener en cuenta, pueden ser menos que las que el modelo detecta.
2) Crear un módulo que recibirá información de detecciones y las almacenará a un sistema de persistencia a elección, la información mínima que debe almacenarse es:
    - devId: id de dispositivo que envío la imagen originalmente
    - timestamp: momento en el que se capturó, se debe asegurar sincronización de tiempo.
    - frame: cuadro en el cual se detectó algo “relevante”
    - detects: arreglo con las detecciones, cada item debe tener al menos: boundingbox, confianza y clase. (*Es fundamental definir un mensaje estándar para dar soporte a consultas y características como: sincronización con flujo, post procesamiento del frame, etc*)

    Este módulo deberá contener un método query(existen, noExisten) que recibirá dos arreglos (al menos uno de los arreglos debe tener un elemento):
    - existen
    - no Existen

    El método retorna un arreglo con los frames en los cuales se cumpla que las clases detectadas están en “existen” o bien que no están en “noExisten”. *Mejora posible, enviar en la cadena “nombreClase:color”, ejemplo: existen=[“persona”,”sombrero:rojo”] <= esto implicaría un posible postprocesamiento.*
3) Crear un módulo que capture vídeo desde una webcam y lo envíe a un flujo de entrada RTSP MediaMTX, La configuración básica es:
    - URL MeditaMTX: la url del flujo de entrada RTSP, ejemplo: rtsp://localhost:8554/webcam
    - Source device: dispositivo del cual se captura video y audio, ejemplo: /dev/video0
    - FPS: cuadros por segundo que se capturan y transmiten
    - Video Size: tamaño de los cuadros, ejemplo: 640x480

Este mismo módulo debe procesar los cuadros con el método run() del punto 1) y cuando se realicen detecciones “relevantes[1]” se enviarán a 2)

Muy importante: crear un mecanismo que permita vincular los cuadros “relevantes” almacenados en 2) con los “recordings” de MediaMTX

Módulos recomendados:

- onnxruntime-node
- opencv4nodejs
- sharp

Herramientas recomendadas:

- ultralitics (yolo)

Se recomienda instalar MediaMTX usando docker

[1] Relevante puede ser cuando se detecte alguna clase configurada como importante (no necesariamente cualquiera, y que se supere cierto nivel de confianza o que se detecte dentro de cierta área, etc. La definición de “relevante” la define el grupo.