Documentation présentation portes ouvertes

Résumé

Dans le cadre de notre projet de **portes ouvertes**, nous avons réalisé, avec mes collègues de 3e année en informatique, un **système de détection de personnes** basé sur l'intelligence artificielle. Ce document a pour but d'expliquer **brièvement le fonctionnement** de notre projet ainsi que les **outils techniques** utilisés. Merci de votre attention et bonne lecture !

Fonctionnement

Le fonctionnement de notre détecteur de personnes se déroule en plusieurs étapes :

1. **Information et conformité légale.** Un document est affiché à l'entrée pour informer les visiteurs de **l'utilisation et du traitement des données** collectées, conformément à la nLPD.
2. **Capture du flux vidéo.** Les différentes **caméras disposées dans la salle** envoient un flux vidéo constant grâce à **OBS (logiciel de streaming)** et au plugin **DistroAV (implémantation de la technologie NDI qui permet de transmettre de la vidéo et du son via le réseau)**.
3. **Analyse et traitement de l'image.** Le **Mac** reçoit le flux vidéo et l'analyse à l'aide des algorithmes **YOLO** (détection d'objets) et **DeepSORT** (suivi et identification).
4. **Stockage des données.** Une **base de données** enregistre les différents **ID détectés**, permettant ensuite d'effectuer des calculs (comme le **nombre de personnes** présentes ou le **temps de présence**).
5. **Affichage du flux filtré.** Le **Mac** affiche le flux vidéo filtré à travers une application qui **combine les calculs et le flux vidéo**, offrant une visualisation en temps réel des résultats.

Limites

Lors de la conception de notre projet, nous avons dû surmonter plusieurs obstacles :

1. **Respect de la législation (nLPD)**
   - Il nous était impossible d'utiliser la **reconnaissance faciale** ou toute donnée biométrique sans recueillir le consentement écrit de chaque visiteur.
2. **Puissance de calcul limitée**
   - Le matériel utilisé est suffisant, mais nous avons atteint la **limite de performances raisonnables** pour un budget restreint.
3. **Manque de temps**
   - Nous avons manqué de temps pour **affiner les réglages** et optimiser entièrement notre système.

Charte de sensibilisation- Détection de personnes par l'IA

Souriez vous êtes filmés !!

Objectif

détecter et reconnaître des personnes à partir de leur silhouette (corps,vetements).

(aucune données biométrique)

Engagement

- **Aucune reconnaissance faciale**
- **Données temporaire et anonymes**
- **Respect loi (nLPD) et éthique**

Gestion des données

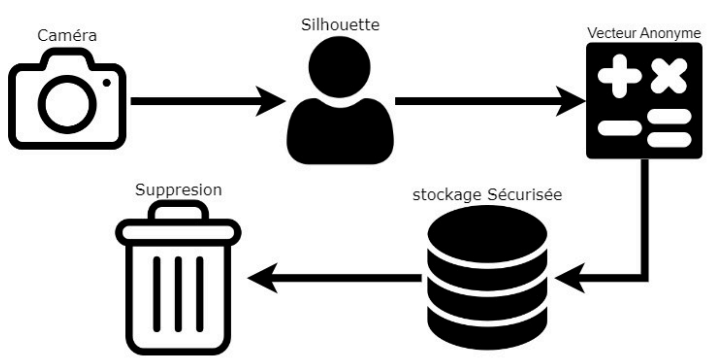
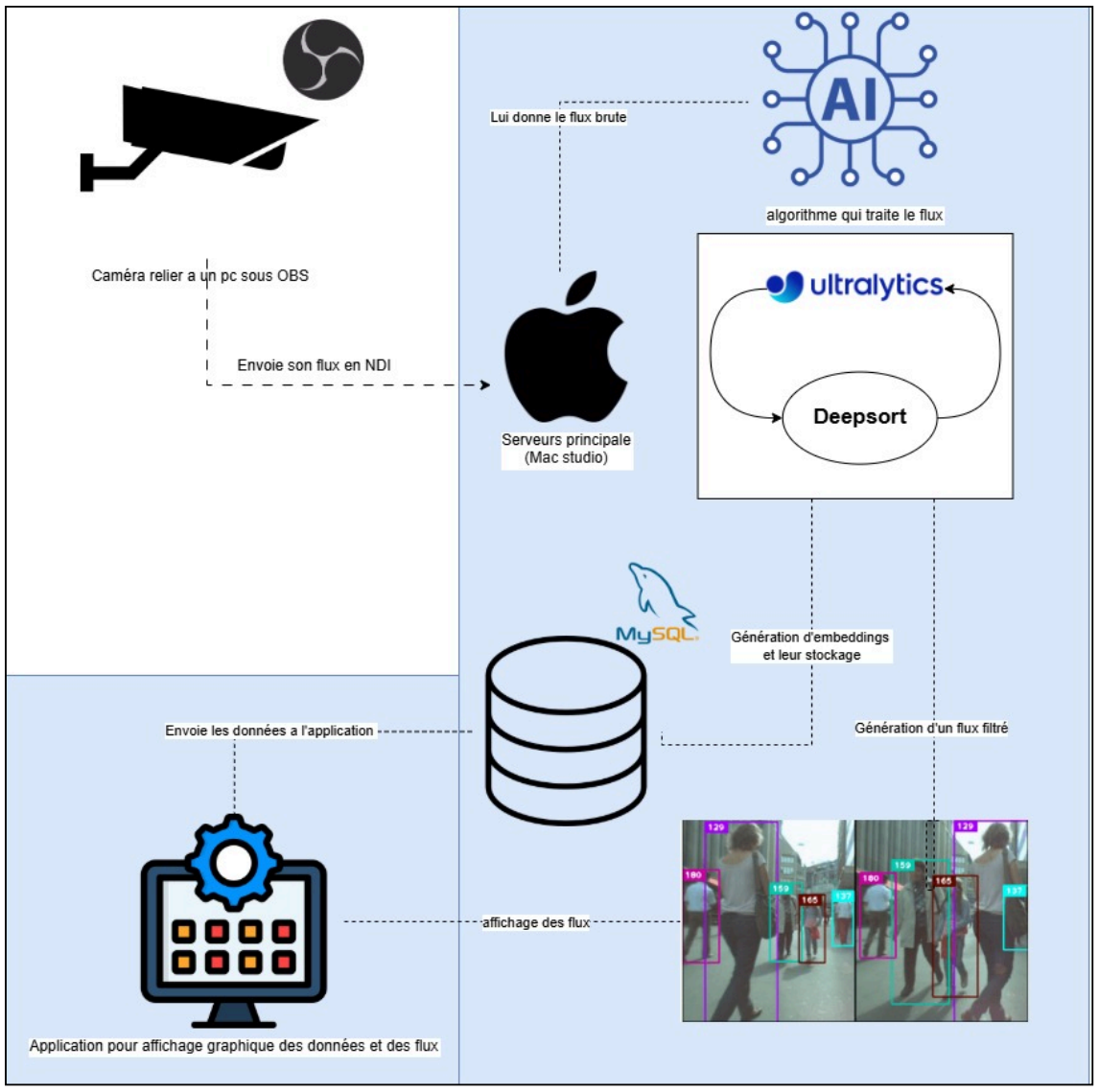


Schéma de l'infrastructure de notre projet portes ouvertes



Mathys Joly, Dario Chapuis, Timéo Chevalier, Leny Mory

```python
from typing import Optional

import mysql.connector
import dotenv
import os

from mysql.connector.abstracts import MySQLCursorAbstract

dotenv.load_dotenv()
print(".env chargé avec succès.")


class DB:
    def __init__(self):
        self.conn: Optional[mysql.connector.MySQLConnection] = None
        self.cursor: Optional[MySQLCursorAbstract] = None
        self.create_db()
        self.connect_db()

    # créer la base de données
    def create_db(self):
        # Créer un curseur pour exécuter les requêtes
        conn_root = mysql.connector.connect(
            host=os.environ['MYSQL_HOST'],  # ou l'IP de ton serveur MySQL
            user=os.environ['MYSQL_USER'],
            password=os.environ['MYSQL_PASSWORD'],
        )
        cursor_root = conn_root.cursor()

        # créer la base de données
        commands = [
            "CREATE DATABASE IF NOT EXISTS IA_DB",
            "USE IA_DB",
            "DROP TABLE IF EXISTS visites",
            """
            CREATE TABLE visites
            (
                ID          INT AUTO_INCREMENT PRIMARY KEY,
                id_personne INT          not null unique,
                state       varchar(255) not null,
                timestamp   timestamp
            )
            """,
        ]
        for command in commands:
            cursor_root.execute(command)

        # Sauvegarder les changements
        conn_root.commit()
        cursor_root.close()
        conn_root.close()

    def connect_db(self):
        # Connexion à la base de données
        self.conn = mysql.connector.connect(
            host=os.environ['MYSQL_HOST'],  # ou l'IP de ton serveur MySQL
            user=os.environ['MYSQL_USER'],
            password=os.environ['MYSQL_PASSWORD'],
            database=os.environ['MYSQL_DB']
        )
        self.cursor = self.conn.cursor()

    # insérer une les lieux dans la base de données
    def fill_DB(self):
        # Insérer les lieux dans la DB
        self.cursor.execute("INSERT INTO personne (ID_lieux, lieux) VALUES (1, stand)", )
        self.cursor.execute("INSERT INTO personne (ID_lieux, lieux) VALUES (2, porte)", )

        # Sauvegarder les changements
        self.conn.commit()

    def insert_visites(self, personnes):
        self.cursor.executemany("INSERT INTO visites (id_personne, state, timestamp) VALUES (%s,%s,%s)", personnes)
        self.conn.commit()

    def fetch_nb_personnes(self):
        self.cursor.execute("SELECT COUNT(*) as nb_personnes FROM visites")
        return self.cursor.fetchone()


    def fetch_personnes(self, ids):
        placeholders = ', '.join(['%s'] * len(ids))
        query = f"SELECT v.id_personne, TIME(v.timestamp) FROM visites v WHERE v.id_personne IN ({placeholders})"
        self.cursor.execute(query, ids)
        return self.cursor.fetchall()

    def close_db(self):
        # Fermer le curseur et la connexion
        self.cursor.close()
        self.conn.close()

```