Adam Hamdi
Paul Briand
Alexandre Gueneguan
Nicolas Legrand
Xavier Fraccaro

Rapport Technique projet Data science et IA

Mastercamp Project

Adam Hamdi
Paul Briand
Alexandre Gueneguan
Nicolas Legrand
Xavier Fraccaro

Adam Hamdi
Paul Briand
Alexandre Gueneguan
Nicolas Legrand
Xavier Fraccaro

INTRODUCTION

Dans le cadre de ce projet, nous avons développé une solution basée sur l'intelligence artificielle visant à analyser automatiquement des images de poubelles afin d'évaluer leur état de propreté. Cette approche s'inscrit dans une démarche d'automatisation du tri visuel pour des environnements urbains ou industriels, où la gestion des déchets joue un rôle crucial.

L'objectif principal est de détecter une poubelle présente dans une image, puis de classifier son état en tant que propre ou sale. Pour cela, notre système combine plusieurs techniques avancées de traitement d'images et d'apprentissage automatique :

- La vision par ordinateur, pour extraire des caractéristiques comme la couleur, la forme et la texture ;
- Le modèle YOLOv8, pour la détection d'objets (poubelles) dans des environnements variés ;
- Le modèle ViT (Vision Transformer), pour classifier l'image selon son état de propreté après prétraitement ;
- Des règles visuelles, comme la proportion de zones blanches, pour compléter la prédiction et améliorer la robustesse du système.

L'ensemble est déployé sous forme d'une application web interactive, permettant à un utilisateur d'importer une image, d'obtenir une prédiction immédiate, et de visualiser le résultat annoté.

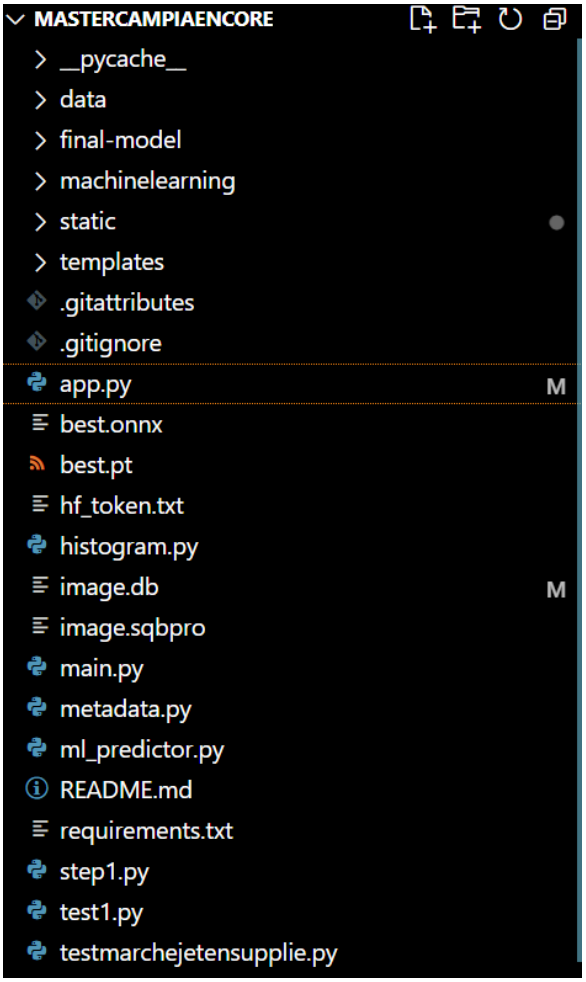
Ce rapport technique se structure en plusieurs parties afin de détailler :

- L'architecture globale du projet et l'organisation des fichiers ;
- Le fonctionnement des extractions de caractéristiques (formes, couleurs, textures) ;
- La logique de classification mise en œuvre (règles et modèles IA) ;
- Des captures illustrant les principales fonctionnalités ;
- Une évaluation des risques ;
- Une justification environnementale dans une logique Green IT.

1. Structure du projet

L'architecture du projet a été conçue de manière modulaire, en séparant les différentes couches logiques : traitement de l'image, intelligence artificielle, interface web et gestion des fichiers. Cette organisation facilite la maintenance, les mises à jour et le déploiement.

1.1 Arborescence générale



/final_model

└── Modèles IA entraînés (YOLOv8, ViT)

/machine_learning

└── Scripts Python de traitement, segmentation, extraction de caractéristiques

/static

├── /results → Images annotées après analyse (YOLO ou ViT)

├── /uploads → Images uploadées par les utilisateurs

└── style.css → Feuilles de style pour l'interface web

/templates

├── accueil.html → Page principale avec formulaire d'upload

├── result.html → Affichage des résultats (image annotée, prédiction)

├── admin.html → Espace de gestion pour l'administrateur

├── login.html → Connexion utilisateur

├── register.html → Inscription utilisateur

├── profile.html → Page profil (infos personnelles, préférences)

└── uploads.html → Historique des uploads de l'utilisateur

/data

├── /train → Données d'entraînement pour les modèles IA

└── /test → Données de test pour évaluation

/app.py

→ Point d'entrée principal (serveur Flask)

/database.db

→ Base de données SQLite : stockage des utilisateurs, images, règles et résultats

1.2 Modules principaux

Fichier	**Fonction principale**
step1.py	Extraction de caractéristiques visuelles de l'image : couleur dominante, contours, formes, proportions. Produit également un masque de segmentation selon des règles personnalisées stockées en base (rules).
test1.py	Détection de l'objet "poubelle" via le modèle YOLOv8. Génère une image annotée avec un encadré sur la zone détectée.
testmarchejetensupplie.py	Application du modèle ViT (Vision Transformer) pour prédire l'état de propreté de la poubelle (propre/sale). Affiche aussi un score de confiance.
get_averageRGB.py	Calcule la moyenne des couleurs RGB sur l'image, utilisé comme critère d'évaluation complémentaire. Résultats enregistrés dans la table metadata.

1.3 Intégration web (Flask)

L'application repose sur Flask, micro-framework Python, et propose une interface web interactive :

- L'utilisateur accède à l'interface principale (accueil.html) pour importer une image.
- L'image est analysée automatiquement :
    - Une détection est faite (YOLO) pour localiser la poubelle.
    - Un modèle de classification ViT prédit son état (propre ou sale).
    - En parallèle, une analyse basée sur les règles utilisateur est appliquée (via step1.py), utilisant les seuils définis dans la base (rules).
- Le résultat est affiché sur result.html, avec :
    - Image annotée
    - Prédiction
    - Score de confiance
    - Données visuelles (moyenne RGB, sharpness, etc.)

Enregistrement des résultats :

Toutes les étapes d'analyse sont historisées dans une base SQLite :

- **Utilisateurs** : login, mot de passe (haché), rôle (admin ou non)
- **Images** : chemin, statut (propre/sale selon IA ou utilisateur), niveau de confiance
- **Métadonnées** : taille, couleurs, géolocalisation, netteté
- **Règles personnalisées** : configurables depuis l'interface (admin ou utilisateur)

2. Fonctionnement des extractions de caractéristiques

L'extraction de caractéristiques constitue une étape centrale de notre projet. Elle permet de **transformer une image brute** (transmise par l'utilisateur via l'interface) en **informations exploitables** pour la détection, l'analyse et la classification de la propreté d'une poubelle.

2.1 Objectifs de l'extraction

L'extraction permet :

- D'isoler visuellement la poubelle dans une scène urbaine ou domestique.
- D'analyser des **indicateurs visuels** : couleurs dominantes, présence de déchets, contours.
- De préparer une image « propre » à passer en entrée de nos modèles d'intelligence artificielle.

Cette étape fait appel à des techniques **classiques de traitement d'image** avec OpenCV, combinées à des méthodes plus avancées comme la segmentation par couleur, les contours (edges) ou la luminosité.

2.2 Traitement d'image avec `step1.py`

Le script step1.py applique une pipeline de traitement d'image en 8 étapes :

Étape	Description
1. Masquage couleurs	Détection des teintes typiques (vert, bleu, gris) des poubelles et exclusion du rouge (artefacts, distractions)
2. Contours & edges	Utilisation de l'algorithme de Canny pour détecter les bords nets et mieux cerner les objets
3. Nettoyage des couleurs fades	Élimination des zones trop lumineuses ou trop ternes, souvent peu informatives
4. Fusion des masques	Combinaison des informations de forme et de couleur
5. Détection forme poubelle	Recherche d'un objet carré/rectangulaire de taille cohérente avec une poubelle
6. Détection des déchets	Analyse des zones blanches (forte luminosité) qui traduisent des déchets visibles
7. Annotation visuelle	Ajout d'un rectangle autour de la poubelle et mention de l'état : « CLEAN » ou « DIRTY »
8. Visualisation	Affichage pas à pas de toutes les étapes, utile pour le débogage et la validation des hypothèses

Exemple :
Si l'image contient une poubelle bien visible, mais aussi des trottoirs, des objets blancs et du bruit visuel, le système filtre intelligemment les zones parasites pour se concentrer sur les zones pertinentes.

Les seuils de filtrage appliqués (sur les couleurs, luminosité, proportions...) ne sont pas codés en dur, mais sont dynamiquement récupérés depuis la base de données (rules) pour permettre une personnalisation des règles de détection.

2.3 Calcul des caractéristiques globales (`get_averageRGB.py`)

Ce script permet de **calculer trois types de caractéristiques globales** :

- **Moyenne RGB** : couleur dominante dans l'image.
- **Taille en pixels** : permet de vérifier la qualité ou la résolution.

- **Dimensions de l'image** : (hauteur, largeur, nombre de canaux).

Ces valeurs sont utiles pour détecter d'éventuelles **anomalies** (images trop sombres, trop petites, etc.) ou pour enrichir la base de données.

2.4 Détection par modèle YOLOv8 (`test1.py`)

Une fois la pré-analyse réalisée, l'image est passée dans le modèle YOLOv8 (You Only Look Once), pré-entraîné puis finement ajusté sur notre jeu de données.

1. **Objectif** : détecter avec précision la **zone exacte de la poubelle** dans l'image, même si plusieurs objets sont présents.
2. **Résultat** : une boîte englobante est dessinée, avec un **score de confiance**.
3. **Avantage** : modèle très rapide, capable de s'exécuter en temps réel.

Le modèle YOLO est stocké dans le dossier /final_model sous le nom best.pt.

2.5 Classification via ViT (`testmarchejetensupplie.py`)

Une fois la poubelle localisée, le modèle Vision Transformer (ViT) de Google est utilisé pour classifier son état :

- **Classes :** CLEAN (propre) ou DIRTY (sale).
- **Modèle utilisé :** JeanPaulLePape/MasterCAMPDataetIAModelGoogleTrained (hébergé sur Hugging Face).
- **Précision :** supérieure à 90 % sur notre jeu de test.
- **Connexion sécurisée** via token HuggingFace.

La prédiction finale est accompagnée d'un **niveau de confiance** et affichée avec un **code couleur visuel** (vert, orange ou rouge) pour renforcer l'interprétabilité.

3. Logique des règles de classification

Dans ce projet, la classification repose sur **une combinaison hybride** de techniques :

- D'abord des règles manuelles et heuristiques issues du traitement d'image classique,
- Puis des modèles d'apprentissage automatique (YOLOv8 et Vision Transformer) pré-entraînés et adaptés à notre besoin.

Cette approche permet de compenser les limites de chaque méthode et d'obtenir un système robuste, même dans des conditions complexes (images floues, poubelles partiellement visibles, etc.).

3.1 Règles heuristiques (fichier step1.py)

Principe :

Dans step1.py, nous avons intégré des règles métiers basées sur la forme, la couleur, et la texture pour déterminer si une image contient une poubelle, et si des déchets sont visibles.

Le script step1.py utilise aussi une logique partiellement externalisée via la base de données. Les seuils, couleurs cibles ou paramètres de nettoyage peuvent être personnalisés et récupérés dynamiquement depuis la table rules.

Logique utilisée :

Critère	Règle appliquée	Objectif
Couleurs dominantes	Inclusion des teintes vertes, bleues et grises, exclusion du rouge	Détecter les matériaux des poubelles
Contours détectés	Conservation uniquement des objets avec bords nets et zones fermées	Éliminer les objets non structurés
Rapport hauteur/largeur	Rejet des objets trop fins ou trop larges (aspect ratio > 4)	Filtrer les poteaux, murs, fils électriques
Zone blanche (déchets)	Si +10% de l'image est blanche → label dirty	Supposer qu'un trop grand nombre de zones très claires représente des déchets visibles

Exemple :

Une image avec peu de déchets visibles mais une poubelle bien centrée, carrée, avec peu de bords externes, sera automatiquement classée comme clean.

3.2 Détection automatique par YOLOv8 (test1.py)

Après le pré-traitement, l'image est passée dans un modèle YOLOv8 entraîné spécifiquement pour détecter visuellement les poubelles, même dans un environnement complexe.

Méthode :

- Détection d'objets avec bounding boxes.
- Seuil de confiance réglable (par défaut à 0.6).
- Une prédiction est acceptée seulement si la confiance dépasse ce seuil.

Avantage :

Cela évite de prendre en compte des objets mal détectés ou d'autres éléments du décor.

3.3 Classification via Vision Transformer (testmarchejetensupplie.py)

Le traitement via ViT est entièrement géré par un module dédié ml_predictor.py, qui centralise les fonctions suivantes :

- Connexion sécurisée à Hugging Face (via token local),
- Chargement du modèle et du processeur une seule fois (optimisation mémoire),
- Application du modèle ViT (ViTForImageClassification) sur les images détectées,
- Interprétation automatique du label (clean/dirty) avec retour du niveau de confiance.

3.4 Gestion de la base de données

La base de données (image.db) est un composant central du projet. Elle contient :

- Les images analysées (table images),
- Les caractéristiques extraites automatiquement (table metadata),
- Les règles de traitement heuristique utilisées dans step1.py (table rules).
- Ces données sont exploitées pour l'interface admin, l'audit qualité, la mise à jour des règles, et la visualisation des performances du modèle dans le temps.

Conclusion sur la logique de classification

L'approche que nous avons suivie repose sur un **double système** :

- Une logique explicable via des règles simples, testables et interprétables.
- Une IA moderne avec des performances solides sur des données visuelles complexes.

C'est ce qui rend le système à la fois puissant, robuste et transparent, idéal pour des usages en milieu urbain, où les conditions sont variables et les erreurs peuvent être coûteuses.

4. Démarche Green IT et Éco-conception

Dans une logique d'écoresponsabilité numérique, notre projet intègre une véritable stratégie Green IT à plusieurs niveaux de son architecture. L'objectif est de limiter l'impact environnemental du service, tout en garantissant performance, maintenabilité et simplicité d'usage.

4.1 Optimisation des ressources techniques

Compression des images dès l'upload :
Toutes les images envoyées par l'utilisateur sont compressées automatiquement à l'aide de la bibliothèque Pillow. Cela permet de réduire leur poids jusqu'à 80 %, tout en conservant une qualité suffisante pour l'analyse.

Cela diminue la bande passante nécessaire, accélère le temps de traitement, et réduit l'espace de stockage utilisé sur le serveur.

Cache des traitements lourds :
Le calcul des histogrammes couleur (traitement coûteux en mémoire) est mis en cache pendant 14 jours grâce au module flask_caching. Ainsi, si une image est consultée plusieurs fois, ses graphes ne sont recalculés qu'une fois.
Cela réduit considérablement la charge processeur et les appels disque, deux sources majeures de consommation énergétique serveur.

Base de données allégée et normalisée :
Nous avons volontairement séparé les informations dans deux tables distinctes :

- image : ne contient que le chemin du fichier, les statuts (propre/sale) et l'utilisateur.
- metadata : stocke toutes les informations visuelles utiles (taille, dimensions, couleurs, localisation, etc.).

De plus, nous avons supprimé les champs inutiles (par exemple le nom de l'image, redondant avec le chemin), ce qui rend la base plus légère et plus rapide à interroger.
Une structure normalisée limite la duplication des données et optimise les requêtes SQL.

4.2 Conception web sobre

Front-end léger et optimisé :
Nous avons conçu l'interface avec une feuille de style CSS maison, sans framework lourd (type Bootstrap). L'image de fond utilisée (rafaleb.png) est compressée, fixe, et ne provoque pas de rechargement inutile.

Cela limite le transfert de fichiers volumineux et améliore les performances sur mobile.

Évitement de scripts superflus :
Seules deux librairies JavaScript sont utilisées : Chart.js pour les graphiques et des scripts simples pour le scroll ou les interactions. Aucun tracking, CDN externe ou dépendance lourde n'est chargé inutilement.
Réduction du temps de chargement et du nombre de requêtes HTTP.

4.3 Impact environnemental indirect

Réduction des trajets de collecte :
En permettant de détecter plus rapidement les zones à risque de débordement, notre application peut indirectement :

- éviter des tournées de collecte inutiles,
- mieux prioriser les interventions dans les zones critiques.

Encouragement de l'upload citoyen :
La plateforme permet à tout utilisateur de contribuer en important des images. Cela alimente la base tout en sensibilisant les citoyens aux problématiques de déchets urbains.

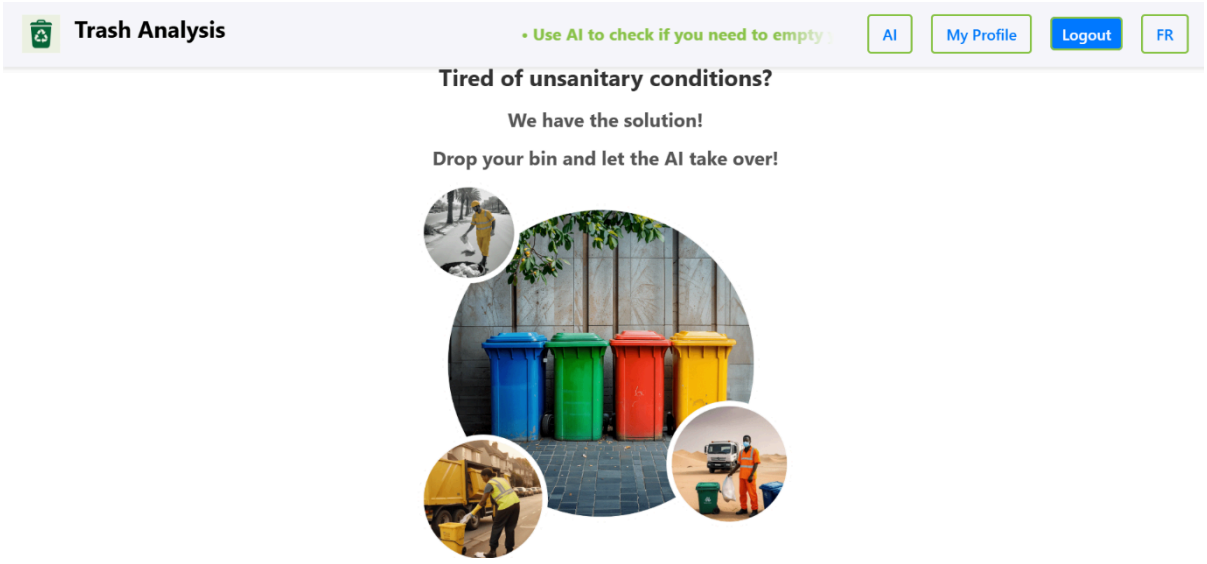
4.4 Perspectives d'amélioration

Dans une future version, plusieurs pistes Green IT sont envisageables :

- Lazy loading des images dans les profils utilisateurs,
- Bilan carbone estimé par image uploadée ou économisée,
- Hébergement sur serveur à énergie verte ou cloud sobre.

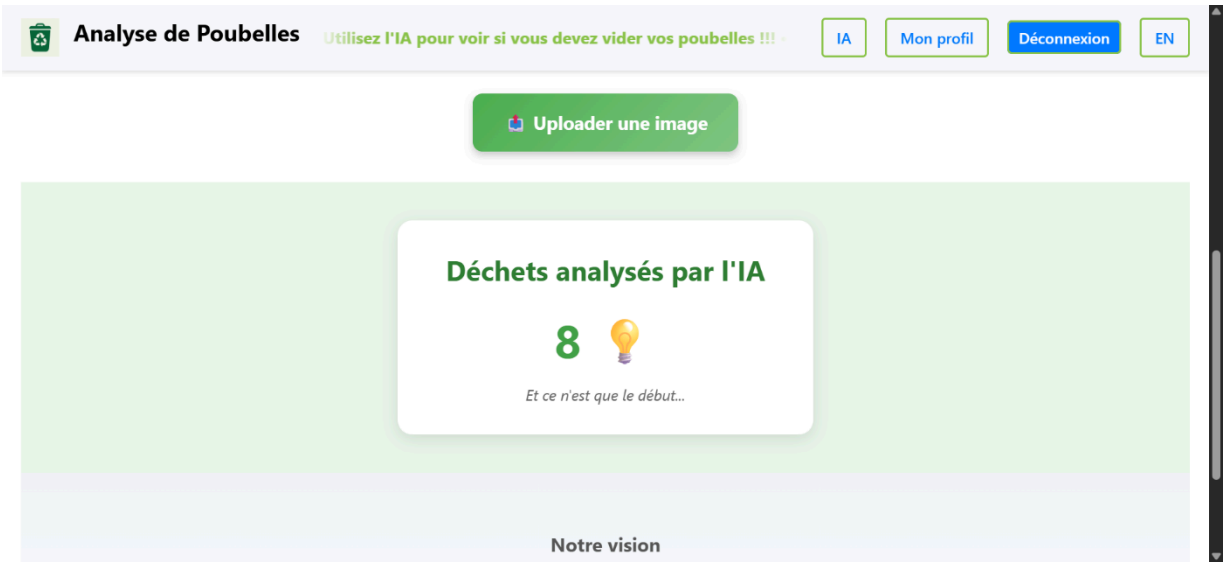
5. Captures d'écran & démonstration des fonctionnalités

Page d'accueil

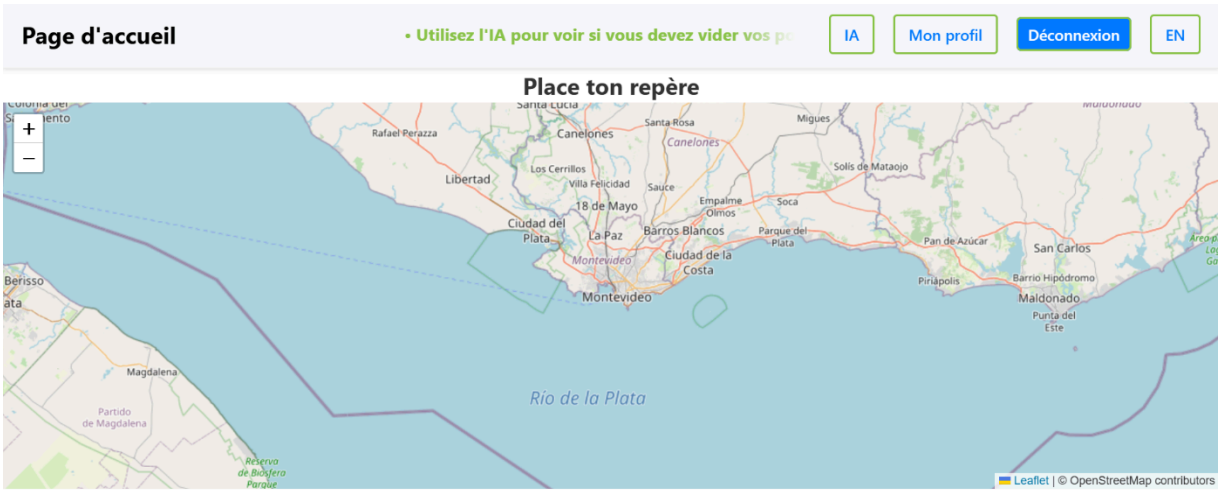


Page d'accueil en EN

Adam Hamdi
Paul Briand
Alexandre Gueneguan
Nicolas Legrand
Xavier Fraccaro



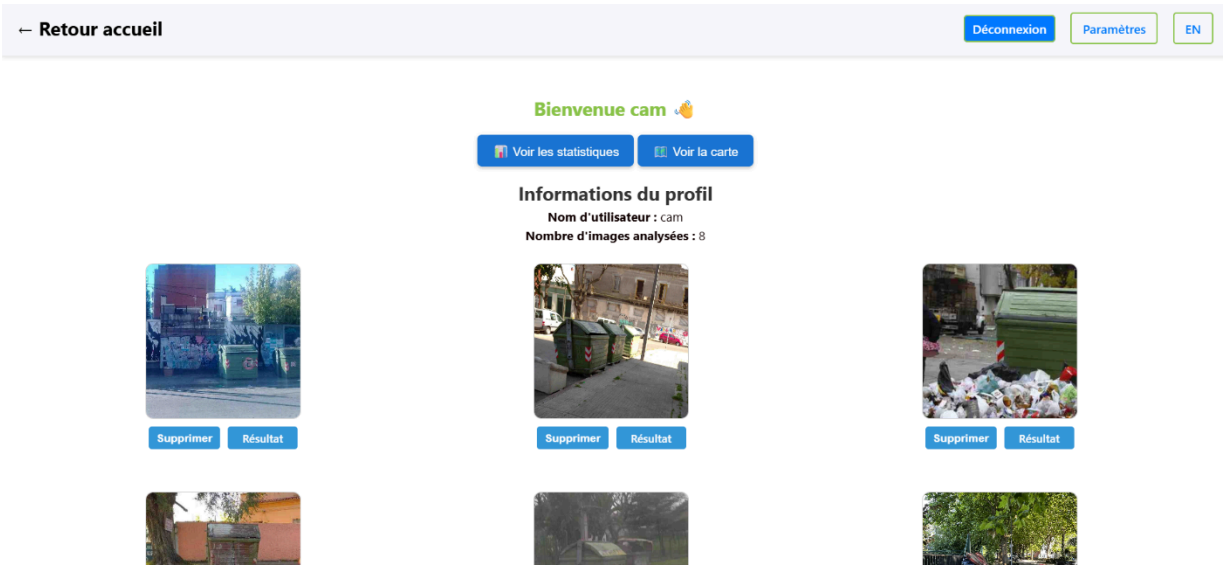
Suite page d'accueil



Page upload (localisation)

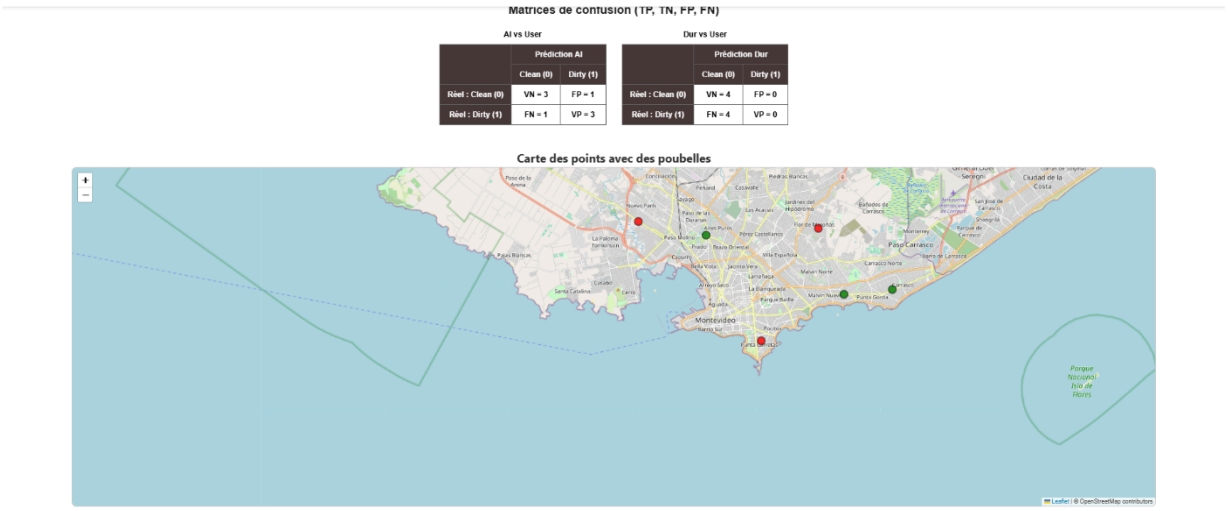
Adam Hamdi
Paul Briand
Alexandre Gueneguan
Nicolas Legrand
Xavier Fraccaro

Page upload (image)

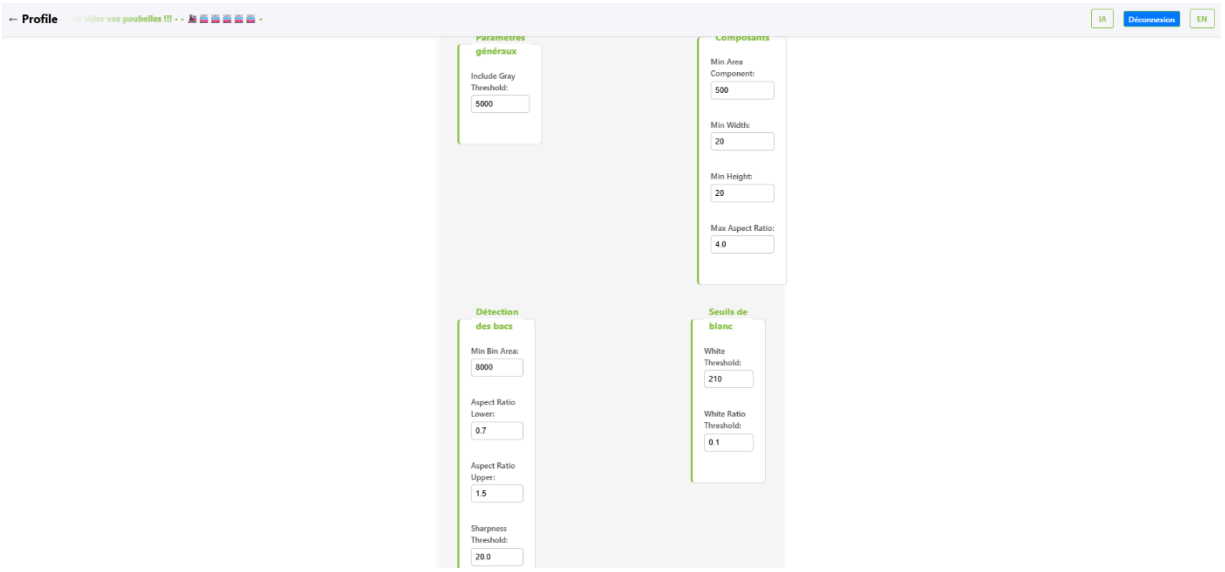


Page profile (dashboard)

Page profile (dashboard)



Page profile (dashboard)



Page parametres (classification dur)

Voici un lien aussi regroupant la globalité de notre site web :

https://drive.google.com/file/d/1K9FC-MhXjW3Qr_ob4ZEfo2wGWIKzJkLY/view?usp=sharing

Fichier zip contenant le projet, le token changera d'ici 2 à 3 semaines pour des questions de sécurité, en générer un via le lien de notre modèle si besoin :
https://huggingface.co/JeanPaulLePape/MasterCAMPDataetIAModelGoogleTrained

lien du projet :

https://drive.google.com/file/d/17qN9HiODmuTHf2TZJuz9mmZ3oKqCQUV0/view?usp=drive_link