Université Abdelmalek Essaadi

École Nationale des Sciences Appliquées Al-Hoceima

Raport:

Application d'analyse d'images sur Azure (iVision)

Module: Cloud Computing

Réalisé par :
AYOUB GORRY
YOUSSEF ELKAHLAOUI
ANASS ESSAFI

Encadré par :
Pr. Hayat Routaib

Résumé

Ce projet présente Cloud Vision Pro, une application web innovante qui intègre l'intelligence artificielle et les services cloud pour l'analyse avancée d'images. En utilisant Azure Computer Vision et des technologies web modernes, l'application offre des capacités de traitement d'images multilingues, avec une architecture cloud robuste et des fonctionnalités de surveillance en temps réel.

Liste des Figures

TABLE DES MATIERES

I. Introduction

L'évolution des technologies de vision par ordinateur a permis des avancées significatives dans la manière dont les machines perçoivent et interprètent le monde visuel. L'importance de l'analyse automatisée d'images réside dans sa capacité à extraire des informations pertinentes à partir de données visuelles, facilitant des applications dans des domaines tels que la médecine, la surveillance, et l'automatisation industrielle. Cependant, des défis persistent, notamment en ce qui concerne la précision, la gestion de données volumineuses et la gestion des biais algorithmiques. Le projet réalisé avec Azure Computer Vision vise à surmonter ces défis en utilisant des outils de traitement d'images avancés pour extraire des informations spécifiques à partir d'images, dans un cadre académique. Ce projet explore des solutions innovantes pour améliorer l'efficacité des processus d'analyse d'images, tout en contribuant à la recherche en intelligence artificielle et en machine learning.

Les objectifs du projet sont de développer une plateforme d'analyse d'images accessible, permettant aux utilisateurs de soumettre facilement des images pour un traitement automatisé et d'obtenir des résultats précis et pertinents. Cette plateforme intègrera des services cloud avancés, notamment ceux proposés par Azure Computer Vision, pour offrir des capacités de traitement d'images à grande échelle et des fonctionnalités d'IA telles que la reconnaissance d'objets, le traitement de texte et l'analyse de contenu visuel. En outre, le projet vise à créer une solution multilingue, afin de rendre l'analyse d'images accessible à un public diversifié, et évolutive, capable de s'adapter aux besoins croissants et aux technologies émergentes dans le domaine de la vision par ordinateur.

La problématique de recherche du projet se concentre sur deux questions principales :

Comment peut-on créer une application d'analyse d'images flexible et performante ?

Quelles sont les meilleures pratiques d'intégration des services cloud ?

II. Méthodologie

1. Architecture Technique

L'architecture du projet repose sur un modèle **trois tiers** (Client, Serveur, Cloud), permettant une séparation claire des responsabilités et une gestion efficace des ressources.

- **Client** : L'interface utilisateur, qui permet aux utilisateurs de télécharger des images, de voir les résultats d'analyse et d'interagir avec l'application.
- **Serveur** : Une application Flask (Python) qui gère la logique métier, le traitement des images et l'intégration avec les services cloud.
- **Cloud** : Les services cloud sont utilisés pour le traitement d'images et la gestion des données, en particulier **Azure Computer Vision** pour l'analyse d'images et **Azure Monitor** pour la surveillance des performances.

Technologies utilisées :

1. **Backend** : Flask (Python), qui fournit une structure légère pour gérer les requêtes HTTP et exécuter la logique du serveur.
2. **Services Cloud** : Azure Computer Vision pour l'analyse d'images, et Azure Monitor pour la surveillance des performances et la gestion des journaux.
3. **Bibliothèques** :
    - **deep-translator** : Utilisé pour la traduction multilingue des résultats d'analyse.
    - **PIL (Pillow)** : Bibliothèque pour la manipulation d'images (redimensionnement, filtrage, etc.).
    - **requests** : Permet la communication entre l'application backend et les services externes via des requêtes HTTP.

2. Composants Principaux

1. **Module d'analyse d'images** : Ce module utilise Azure Computer Vision pour effectuer des tâches telles que la détection d'objets, l'extraction de texte à partir d'images, et l'analyse des couleurs présentes dans les images.
2. **Système de traduction dynamique** : Un mécanisme intégré qui permet de traduire automatiquement les résultats d'analyse dans la langue de l'utilisateur en utilisant **deep-translator**.
3. **Mécanisme de journalisation et surveillance** : Grâce à **Azure Monitor**, ce module suit les performances de l'application et enregistre les événements importants pour un suivi continu et une gestion proactive des erreurs.

3. Fonctionnalités Techniques

- **Détection d'objets** : L'application identifie et localise les objets présents dans une image, à l'aide des capacités d'Azure Computer Vision.

- **Extraction de texte** : Le système utilise la reconnaissance optique de caractères (OCR) pour extraire le texte contenu dans les images, ce qui peut être utile dans des applications comme la lecture de documents ou l'analyse de panneaux.
- **Analyse des couleurs** : Le système analyse les couleurs dominantes dans l'image pour fournir des insights sur la composition visuelle de l'image.
- **Traduction multilingue** : Les résultats de l'analyse sont traduits en plusieurs langues, permettant ainsi aux utilisateurs de comprendre les résultats dans leur langue préférée.
- **Gestion de l'historique des analyses** : L'application conserve un historique des analyses précédentes, permettant aux utilisateurs de revoir leurs résultats et d'effectuer des comparaisons dans le temps.

III. Implémentation

1. Configuration de l'Environnement:

Dépendances (requirements.txt):

Le fichier `requirements.txt` est utilisé pour spécifier les dépendances (bibliothèques et packages) dont votre application a besoin pour fonctionner correctement. Ce fichier permet à d'autres utilisateurs de facilement installer toutes les dépendances requises avec la commande suivante :

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

Le fichier `requirements.txt` contient une liste des bibliothèques Python nécessaires, avec leurs versions spécifiques si nécessaire. Par exemple :

```
Flask==2.3.3

azure-cognitiveservices-vision-computervision==0.9.0

python-dotenv==1.0.0

Pillow==10.0.0

requests==2.31.0
```

Chaque ligne du fichier spécifie une dépendance et, optionnellement, sa version. Cela permet de garantir que votre application fonctionne de manière cohérente sur différentes machines.

Configuration des variables d'environnement:

Les variables d'environnement sont essentielles pour :

- ✓ Sécuriser les informations sensibles (clés API, mots de passe)
- ✓ Configurer l'application sans modifier le code source
- ✓ Permettre différentes configurations entre développement et production

➢ Fichier `.env` dans le projet:

Examinons le contenu du fichier `.env` :

```
# Clés Azure Computer Vision

AZURE_KEY=votre_clé_api_vision

AZURE_ENDPOINT=https://votre_region.cognitiveservices.azure.com/

# Chaîne de connexion Azure Monitor (optionnel)

APPLICATIONINSIGHTS_CONNECTION_STRING=votre_chaine_de_connexion_monitor

# Autres configurations

DEBUG=False

FLASK_ENV=production
```

➢ Configuration dans le code :

Dans `app.py`, l'initialisation des variables d'environnement se fait avec `python-dotenv` :

```
# Chargement des variables d'environnement

from dotenv import load_dotenv

import os



# Charge les variables du fichier .env

load_dotenv()
```

Initialisation des Services Azure:

➢ Configuration du Logging:

```
# Configuration logging

import logging

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(name)s -
%(levelname)s - %(message)s')
```

```
logger = logging.getLogger(__name__)
```

- **Imports pour Azure Services:**

```
# Imports Azure Computer Vision

from azure.cognitiveservices.vision.computervision import
ComputerVisionClient

from azure.cognitiveservices.vision.computervision.models import
VisualFeatureTypes, OperationStatusCodes

from msrest.authentication import CognitiveServicesCredentials
```

- **Initialisation du Client Computer Vision:**

```
# Récupération des identifiants Azure Computer Vision

azure_key = os.getenv('AZURE_KEY')

azure_endpoint = os.getenv('AZURE_ENDPOINT')



# Création du client Computer Vision

computer_vision_client = ComputerVisionClient(

    endpoint=azure_endpoint,

    credentials=CognitiveServicesCredentials(azure_key)

)
```

Monitoring:

- **Configurer Azure Monitor dans Grafana**

1. Connectez-vous à Grafana.
2. Allez dans Configuration > Data Sources > Add data source.
3. Sélectionnez Azure Monitor.
4. Remplissez les informations demandées :
5. Tenant ID : Disponible dans Azure AD > App registrations > [Votre Application].
6. Client ID : Disponible dans les détails de l'application Azure AD.
7. Client Secret : Le secret que vous avez créé plus tôt.
8. Subscription ID : Trouvable dans le portail Azure sous Subscriptions.

- **Créer des Dashboards**

1. Une fois la source de données configurée, allez dans Create > Dashboard.

2. Ajoutez des Panels pour visualiser des métriques comme l'utilisation CPU, les disques, la mémoire, ou les logs des ressources Azure.
3. Sélectionnez Azure Monitor Metrics comme source de données dans les panels.

2. Flux de Traitement des Images

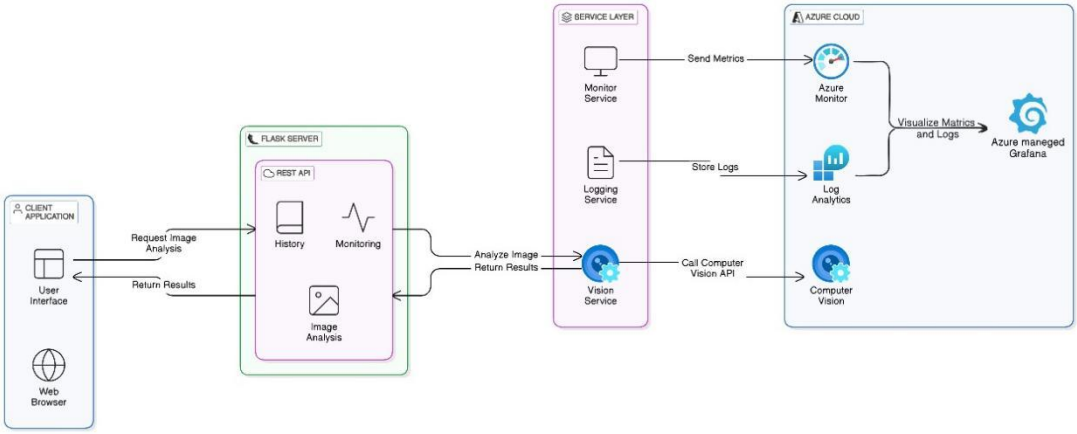

Figure 1: Architecture d'application

Cette figure illustre une architecture intégrée qui combine un serveur Flask, des services cloud Azure, et une application cliente pour fournir une analyse d'image via une API REST. Voici les principaux éléments et leurs interactions :

1. Serveur Flask

Le serveur Flask est au cœur de l'application. Il gère les appels API et les interactions entre les différentes couches de l'infrastructure :

- **REST API** :
  - Gère l'analyse des images via des points de terminaison spécifiques.
  - Offre des fonctionnalités comme l'historique et la surveillance.
- Envoie des requêtes à la couche des services pour traiter les images.

2. Couche de Services

Cette couche joue un rôle intermédiaire essentiel, en orchestrant les appels aux services Azure et en gérant les métriques et les journaux.

- **Vision Service** :
  - Appelle l'API Azure Computer Vision pour analyser les images.
  - Retourne les résultats de l'analyse au serveur Flask.
- **Logging Service** :
  - Stocke les détails des requêtes et réponses pour suivi et audit.

- **Monitor Service** :
    - Envoie des métriques liées aux performances et à l'utilisation à Azure Monitor.

3. Cloud Azure

Les services cloud Azure sont utilisés pour effectuer les tâches principales de traitement et de surveillance :

- **Computer Vision** :
    - Effectue l'analyse des images envoyées par le Vision Service.
- **Azure Monitor** :
    - Surveille les performances des systèmes et collecte des métriques clés.
- **Log Analytics** :
    - Stocke et analyse les journaux pour le suivi à long terme.

4. Application Client

Cette application fournit une interface utilisateur qui permet aux utilisateurs finaux de :

- **Ouvrir l'application** pour interagir avec les fonctionnalités.
- **Afficher les résultats** de l'analyse des images dans un navigateur Web.

Flux de Données

1. L'utilisateur soumet une image via l'application cliente.
2. L'image est envoyée au serveur Flask via la REST API.
3. Le serveur appelle le Vision Service pour traiter l'image.
4. Le Vision Service interagit avec Azure Computer Vision pour analyser l'image.
5. Les résultats et métriques sont renvoyés au serveur Flask.
6. Les journaux et métriques sont stockés dans Azure Monitor et Log Analytics.
7. Les résultats sont renvoyés à l'application cliente pour affichage.

IV. Résultats et Performances

1. Résultats

La page web où tu peux téléverser les images à traiter:

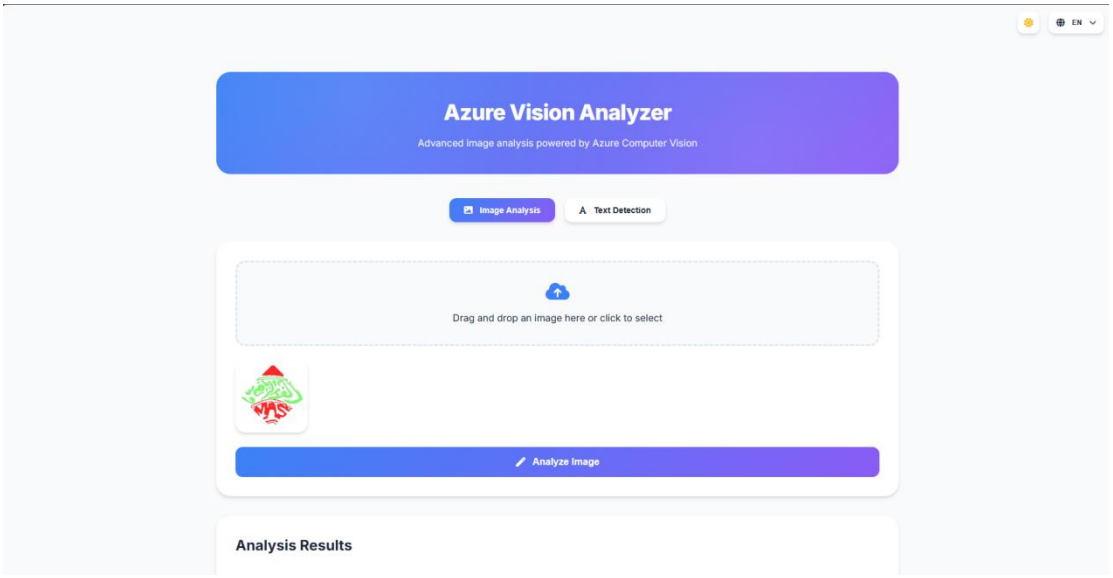

Figure 2: Capture d'écran de la page WEB d'application

Partie d'analyse des image:

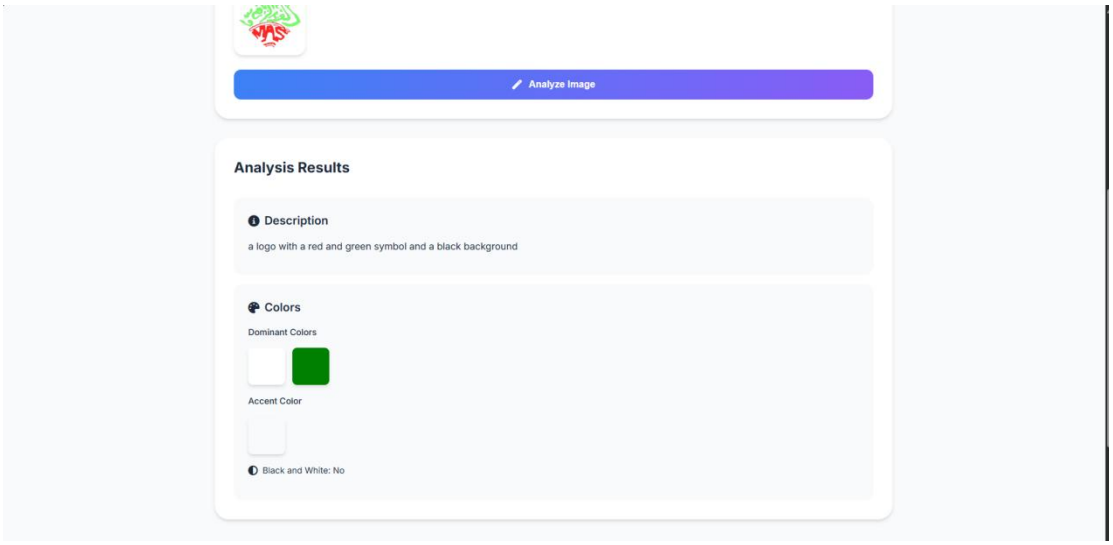

Figure 3: Capture d'écran la partie d'analyse des images

Partie de la detection de texte dans les image :

Figure 4: Capture d'écran la partie detection de texte

La partie d'hestorique des analyse:

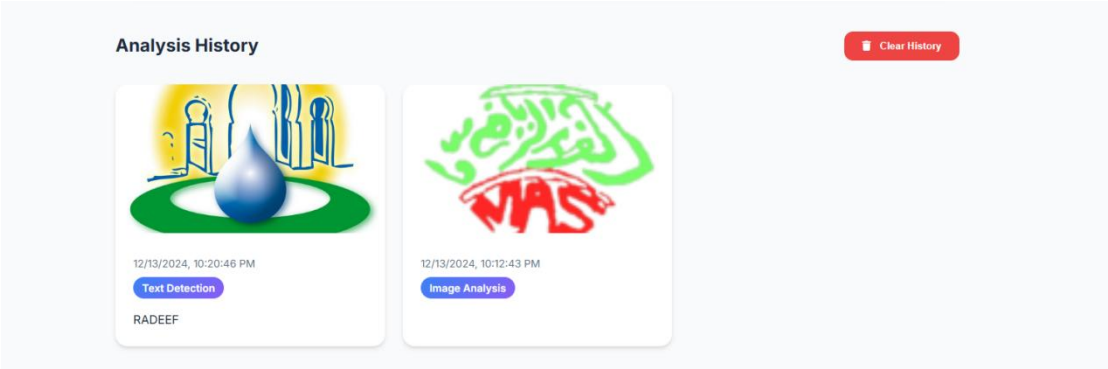

Figure 5: Capture d'écran la partie d'historique des analyses

2. Performances:

Voici des exemples :

- **Azure:**

On peut visualiser l'utilisation de l'application de performance dans Azure Monitor comme suit :

1) Performance d'application
2) Utilisation

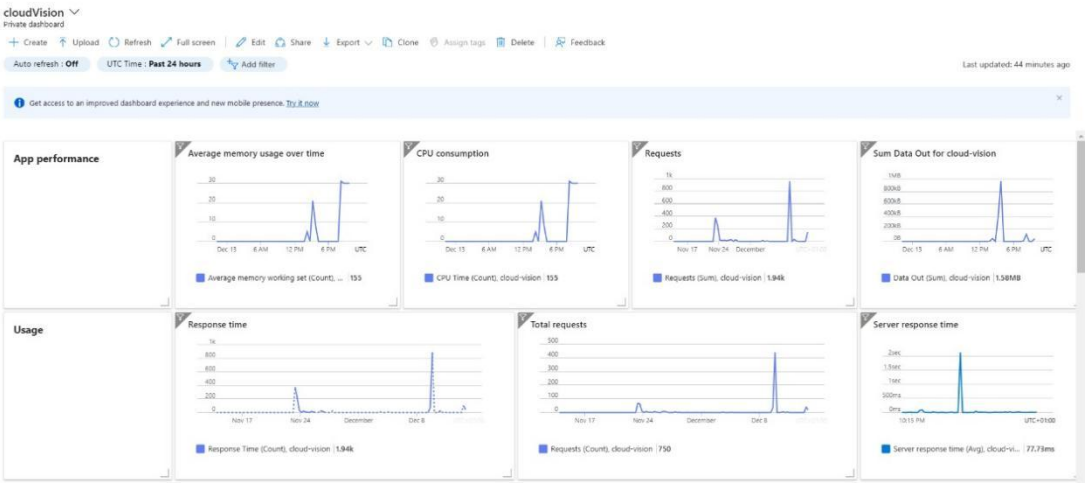

Figure 6: Capture d'ecrant pour Azure Monitor App performance & Usage

3) HTTP codes
4) BandWidth

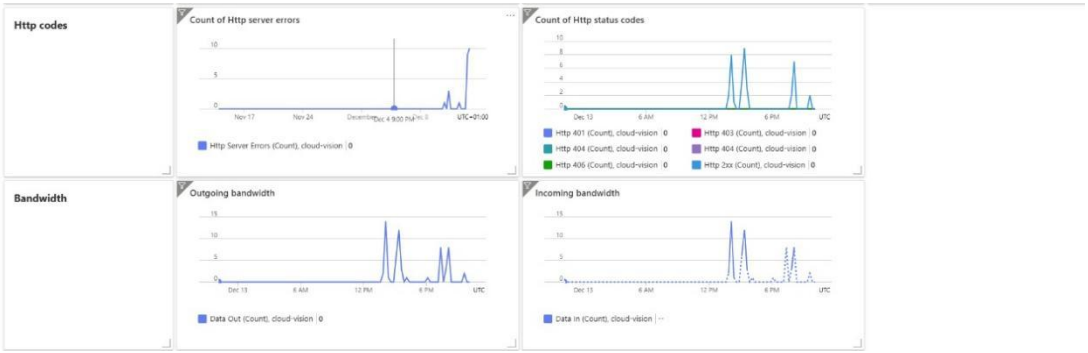

Figure 7: Capture d'ecrant pour Azure Monitor Http codes & BandWidth

- **Graphana:**

De même avec Grafana, mais on peut mieux visualiser les performances de notre application :

1) Titre de ligne

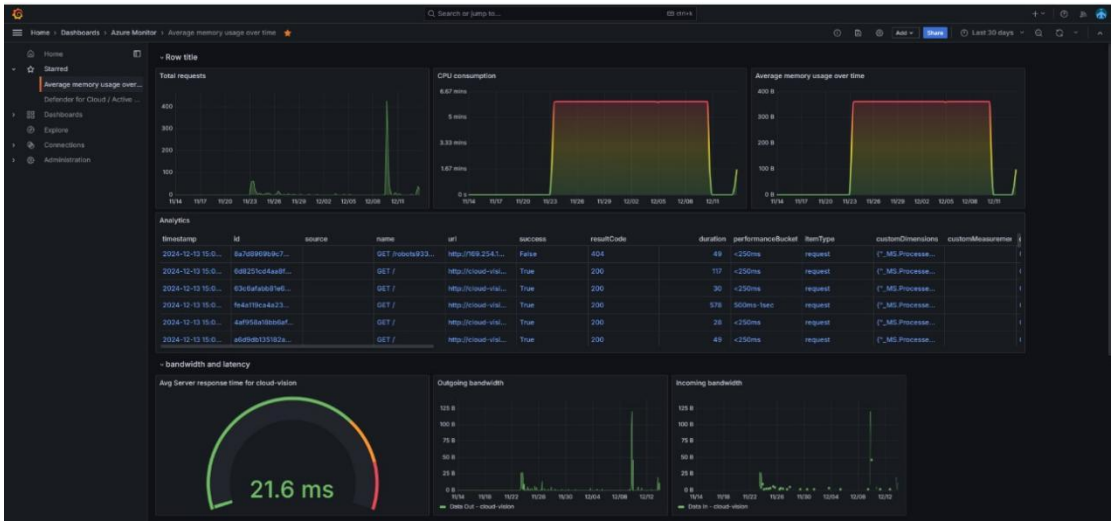

Figure 8: Capture d'ecrant pour Graphana Row title

2) BandWidth & latency

Figure 9: Capture d'ecrant pour Graphana BandWidth & latency

V. Discussion

Avantages de l'Approche

- Modularité : La séparation des responsabilités (API, service, surveillance) facilite la maintenance et l'extension.
- Utilisation d'Azure : Les services cloud garantissent la scalabilité et la robustesse.

- Surveillance Intégrée : L’utilisation d’Azure Monitor et Log Analytics permet un suivi détaillé des performances.

Limitations

- Dépendance à Azure : Forte dépendance aux services Azure, ce qui limite la flexibilité en cas de migration ou de changement de fournisseur.
- Coût élevé : Multiples services Azure utilisés, ce qui peut entraîner un surcoût important pour le projet.
- Performances non optimisées : Absence de cache ou de mécanismes pour accélérer le traitement et réduire les temps de réponse.

Perspectives d'Amélioration

- Implémentation de caches pour réduire les temps de réponse lors des requêtes répétées.
- Authentification utilisateur

VI. Conslusion

Cloud Vision Pro démontre la puissance de l'intégration des services cloud dans le traitement d'images. En combinant Azure Computer Vision, une architecture Flask modulaire et des capacités de traduction dynamique, l'application offre une solution flexible et performante pour l'analyse automatisée d'images.

Références

Repertoire Github:[link]

Application web: [link]