Cahier des Charges : Application iOS complète pour la détection et la gestion d'espèces

Aya Trabelsi et Yasmine Mahdoui

October 28, 2024

Contents

1 Introduction

Ce projet propose une application iOS complète qui intègre toutes les fonctionnalités nécessaires : capture d'image, détection locale d'espèces avec CoreML, synchronisation avec le cloud pour une identification avancée et visualisation des détections directement au sein de l'application. L'objectif est d'avoir une application unique qui réalise les détections, gère les résultats et affiche les informations en temps réel.

2 Objectifs

- Capturer les images et enregistrer la localisation avec les APIs natives (caméra, GPS).
- Détecter rapidement des catégories d'espèces (ex. : chien, chat, arbre, fleur) en local avec CoreML.
- Envoyer les résultats au cloud backend (Jakarta EE) pour identifier l'espèce exacte (ex. : Husky, Siamois).
- Stocker les résultats dans une base de données Azure SQL.
- Offrir un dashboard natif dans l'application pour visualiser les détections en temps réel avec une carte interactive.
- Analyser les données collectées directement depuis l'application.

3 Technologies Utilisées

- SwiftUI : Framework pour développer une interface utilisateur moderne et réactive.
- CoreML : Moteur d'inférence locale sur iOS pour la détection rapide.
- AVFoundation : API pour gérer la caméra et capturer les images.
- CoreLocation : API pour récupérer les coordonnées GPS de l'utilisateur.
- Jakarta EE : Backend cloud pour l'identification précise et la gestion des APIs.
- WildFly : Serveur d'application Jakarta EE rapide et léger.
- Azure SQL : Base de données relationnelle pour centraliser les résultats des détections.
- WebSocket : Assure la synchronisation en temps réel des données.

4 Architecture du Système

1. Capture d'image et GPS avec AVFoundation et CoreLocation : Prend la photo et enregistre la position GPS.
2. Détection locale avec CoreML : Catégorisation de l'objet détecté (chien, chat, arbre, fleur).

3. Envoi des résultats au backend : Transmission au serveur cloud pour une identification fine.
4. Stockage dans Azure SQL : Sauvegarde des informations enrichies dans la base de données.
5. Synchronisation avec WebSocket : Mise à jour des informations en temps réel au sein de l'application.
6. Visualisation native du dashboard : Affichage des résultats avec galerie et carte interactive.
7. Analyse des données directement depuis l'application : Exploration des tendances et statistiques.

5 Fonctionnalités de l'Application iOS

5.1 Capture et Détection

- Accès à la caméra avec AVFoundation : Capture d'images pour analyse.
- Géolocalisation avec CoreLocation : Récupère la position GPS en temps réel.
- Détection rapide avec CoreML : Identifie la catégorie d'objet détecté en local.

5.2 Transmission et Stockage des Données

- Transmission au backend : Envoi des images et résultats au serveur cloud.
- Stockage dans Azure SQL : Sauvegarde des informations de détection enrichies.

5.3 Visualisation et Dashboard Intégré

- Dashboard natif : Affiche les détections sous forme de galerie avec le nom de l'espèce.
- Carte interactive : Visualise les détections sur une carte avec les coordonnées GPS.
- Recherche et filtres dynamiques : Recherche par nom d'espèce, date ou localisation.

5.4 Analyse des Données

- Exploration des tendances : Identifie les espèces les plus détectées.
- Visualisation des statistiques : Affiche la fréquence et les localisations des détections.

6 Backend et Gestion des Données

- API REST : Permet à l'application d'envoyer les résultats au backend pour identification fine.
- WebSocket : Synchronise les informations en temps réel avec l'application.
- Azure SQL : Gère le stockage des résultats enrichis.

7 Livrables

- Application iOS complète avec détection locale, capture d'image, visualisation et analyse.
- Backend Jakarta EE avec API REST et WebSocket déployé sur WildFly.
- Base de données Azure SQL opérationnelle.
- Documentation complète du projet.

8 Échéancier

1. Phase 1 : Planification et conception (1 semaine).
2. Phase 2 : Développement du backend (3 semaines).
3. Phase 3 : Développement de l'application iOS avec CoreML (4 semaines).
4. Phase 4 : Intégration avec Azure SQL et WebSocket (2 semaines).
5. Phase 5 : Tests et validation (2 semaines).
6. Phase 6 : Documentation et livraison (1 semaine).

9 Risques et Contraintes

- Qualité des modèles IA : Les modèles CoreML et cloud doivent être précis.
- Synchronisation des données : Assurer une communication fluide entre l'application et le backend.
- Connectivité réseau : Une connexion est nécessaire pour l'identification fine dans le cloud.

10 Conclusion

Cette application iOS unique regroupe toutes les fonctionnalités nécessaires pour la capture, détection, identification et gestion des espèces. Grâce à CoreML, l'application offre une détection rapide en local, tandis que le backend Jakarta EE assure une identification fine et un stockage centralisé. Le dashboard natif permet de suivre et analyser les résultats en temps réel, offrant une solution performante, complète et intuitive.