UNIVERSITÉ DE LORRAINE | IUT Metz Informatique

Département Informatique

SAÉ 5 — Développement avancé

Rapport SAÉ 5 — Sprint 2

Diego TRIVINO

Mathias FRIES

Alexandre KESSELER

Joshua BROCHOT

Table des matières

1 Introduction

Dans le cadre de notre projet d'application intégrant un modèle de reconnaissance d'images basé sur l'intelligence artificielle, nous avons conçu une architecture alliant performance, modularité et compatibilité avec les environnements mobiles. L'objectif principal de ce projet est de permettre l'identification automatique d'éléments visuels — en l'occurrence, des salles de cours — à partir de photos ou de vidéos capturées en temps réel ou importées par l'utilisateur.

1.1 Méthodologie de développement : approche Agile Scrum

Le développement du projet s'appuie sur la méthode **Agile Scrum**, afin de favoriser la réactivité et le travail en équipe. Le projet est découpé en **sprints hebdomadaires** se terminant chaque **vendredi à 18h**, avec à la fin de chaque sprint une version utilisable et testée de l'application.

Une **réunion d'équipe hebdomadaire** permet de faire le point sur l'avancement, d'ajuster les priorités et de planifier les étapes suivantes. Cette organisation assure une progression régulière du projet et facilite la prise en compte des ajustements nécessaires au fil du développement.

2 Choix des technologies

Dans le cadre de notre projet de reconnaissance d'objets basée sur l'intelligence artificielle, nous avons cette semaine finalisé le choix des principales technologies qui composeront notre architecture applicative. Cette sélection a été guidée par des critères de performance, de simplicité d'intégration avec l'IA, ainsi que par la compatibilité avec un déploiement sur appareils mobiles.

2.1 Frontend

Pour le développement du **frontend**, nous avons choisi **React Native**, un framework open-source qui permet de créer des applications mobiles multiplateformes (iOS et Android) à partir d'un seul et même code source. Ce choix s'explique par la flexibilité qu'offre React Native en matière d'interface utilisateur, mais aussi par la large communauté et la richesse de son écosystème.

2.2 Backend

Plusieurs technologies backend ont été envisagées pour assurer la communication entre l'application mobile, les modèles d'IA, et la base de données. Notre choix final s'est porté sur **FastAPI**, mais ce n'est qu'après une analyse comparative incluant d'autres frameworks tels que **Spring Boot** et **Symfony**.

Symfony, framework PHP mature et très utilisé dans le développement web, a aussi été considéré. Bien qu'il offre une architecture solide, une communauté active et de nombreux bundles utiles, il s'est révélé moins adapté à nos besoins. Le principal frein réside dans le fait que **PHP n'est pas un langage utilisé dans le domaine de l'intelligence artificielle**, ce qui aurait nécessité d'externaliser les traitements IA dans un service distinct en Python, complexifiant inutilement l'architecture.

Nous avons également envisagé **Spring Boot**, reconnu pour sa robustesse, sa structure claire, et sa pertinence pour les projets de grande envergure. Cependant, son écosystème Java le rend plus complexe à manipuler pour des traitements IA majoritairement réalisés en Python. Sa mise en place est plus lourde, et la gestion des fichiers y est moins directe, ce qui a freiné son adoption dans notre contexte de prototypage rapide.

FastAPI, un framework Python moderne et asynchrone, s'est rapidement imposé comme le candidat idéal pour notre projet. Il se distingue par sa simplicité de mise en œuvre, sa performance élevée, et sa compatibilité naturelle avec les bibliothèques d'intelligence artificielle comme **PyTorch**, **TensorFlow**, ou encore **OpenCV**.

2.3 Base de Données

Enfin, pour le stockage des données, nous avons retenu **PostgreSQL**, une base de données relationnelle robuste, open-source, et largement éprouvée. Un des critères déterminants a été son support natif des **UUID**

v4 (Universal Unique Identifier), que nous utiliserons pour identifier de manière unique et sécurisée les objets, les utilisateurs, ou les sessions de traitement. Cette fonctionnalité est particulièrement utile dans un contexte distribué, où plusieurs clients mobiles peuvent interagir avec le backend en parallèle.

3 Liste des fonctionnalités

Cette section décrit les principales fonctionnalités envisagées par le groupe pour l'application mobile de reconnaissance des salles de l'IUT de Metz. Le développement de l'application n'ayant pas encore débuté, ces éléments représentent les fonctionnalités prévues dans la phase de conception afin de définir clairement les objectifs et les besoins du futur système.

3.1 Page d'accueil

La page d'accueil constitue le point d'entrée principal de l'application. Elle propose une interface simple et intuitive permettant à l'utilisateur d'accéder rapidement aux principales fonctionnalités : caméra, import de photos ou de vidéos, etc. Grâce à cette page, l'utilisateur peut lancer facilement une reconnaissance de salle.

3.2 Caméra

Cette fonctionnalité permet à l'utilisateur de capturer une photo ou une vidéo en temps réel d'une salle, afin que l'application puisse analyser l'image et identifier le nom de la salle (exemple : `F33`).

Le module caméra inclut :

- un bouton de capture ;
- une zone pour importer une photo ou une vidéo ;
- une possibilité de zoom ;
- un retour visuel indiquant la probabilité de la prédiction.

3.3 Import des photos et vidéos

L'utilisateur peut sélectionner une photo ou une vidéo existante depuis la galerie de son smartphone. Cette fonctionnalité permet de tester la reconnaissance sur des médias déjà enregistrés, comme des images prises précédemment ou partagées par d'autres utilisateurs.

Le module d'import gère également le redimensionnement et la préparation de l'image avant son envoi au modèle de prédiction.

3.4 Barre d'outils (Toolbar)

La barre d'outils permet une navigation rapide entre les différentes sections de l'application. Elle contient notamment :

- un accès direct à la caméra (fonctionnalité principale) ;
- les icônes de navigation (Accueil, Historique, Profil, Connexion, etc.) ;
- le nom ou le logo de l'application.

3.5 Historique

La fonctionnalité d'historique permet de consulter les prédictions précédemment réalisées par l'application. Chaque entrée de l'historique comprend :

- l'image analysée ;
- la prédiction obtenue ;
- la date et l'heure ;
- le taux de certitude du modèle.

3.6 Authentification

Le système d'authentification assure une gestion sécurisée des utilisateurs. Il permet la connexion via un identifiant (adresse e-mail, mot de passe ou autre méthode). Chaque utilisateur dispose ainsi d'un espace personnel contenant son historique, ses corrections et ses préférences.

Cette étape contribue également à l'amélioration continue du modèle grâce aux retours des utilisateurs.

3.7 Correction de la prédiction

En cas d'erreur de reconnaissance, l'utilisateur peut corriger manuellement la prédiction. Cette correction est enregistrée dans la base de données afin d'améliorer le modèle d'apprentissage.

Ce mécanisme de *feedback* est essentiel au processus de réentraînement du modèle et à l'amélioration de la précision globale de l'application. Un système complémentaire permettra aux administrateurs de réaliser des tests et d'apporter leurs propres corrections.

3.8 Solutions proposées

— Pour pallier l'absence de séparation claire entre client et serveur, nous avons mis en place une API REST reposant sur une architecture MVC côté backend.

— Pour renforcer la sécurité, un système d'authentification basé sur JWT a été implémenté, en remplacement de l'ancien mécanisme reposant sur un simple cookie utilisateur.

— Afin d'améliorer la maintenabilité du code côté client, tous les appels à l'API ont été centralisés dans un fichier unique.

— Chaque page dispose désormais de son propre fichier de script, ce qui permet une meilleure organisation du code JavaScript.

— Des composants réutilisables ont été créés pour le header et le footer, afin de limiter la duplication du code HTML.

4 Initialisation du dépôt GitHub

Le dépôt du projet contient à la fois le *backend* et le *frontend*. Il est structuré autour de plusieurs branches :

— `main` : branche stable contenant les versions validées,

— `dev` : branche de développement principal,

— `test` : branche destinée aux tests avant intégration,

— branches de fonctionnalités (*feature branches*) : créées pour chaque ajout ou modification spécifique.

Le dépôt est mis à jour régulièrement. Des versions sont publiées chaque semaine sous forme de *releases*, afin d'assurer un suivi régulier de l'évolution du projet. Exceptionnellement, cette semaine, seul un tag de version sera créé, aucune nouvelle fonctionnalité n'ayant été ajoutée.

Une réflexion a également été menée sur les bonnes pratiques à adopter pour maintenir un code clair, cohérent et de qualité.

4.1 Initialisation du projet

Le *frontend* a déjà été initialisé à l'aide d'un projet *Expo React Native*. Les prochaines étapes consistent à mettre en place la base du *backend* et la configuration de la base de données.

4.2 Intégration et déploiement continus

Le projet intègrera des processus d'intégration continue (CI) et de déploiement continu (CD) afin d'automatiser les tests, la construction et le déploiement des différentes parties de l'application.

4.3 Conteneurisation

L'ensemble du projet sera conteneurisé à l'aide de *Docker* et *Docker Compose*. La conteneurisation du *frontend* est déjà opérationnelle. Elle sera prochainement étendue au *backend* et à la base de données, dès que ces composants auront été mis en place.

Cette approche permet de garantir un environnement commun entre tous les développeurs et de simplifier le déploiement sur les différentes plateformes.

4.4 Pipelines CI/CD

Les pipelines d'intégration continue seront prochainement ajoutés. Ils permettront :

— d'exécuter automatiquement les tests de linting et de compilation,

— de valider les demandes de fusion (*pull requests*),

— et de déployer automatiquement l'application sur les environnements de test et de production.

5 Test du modèle de reconnaissance

5.1 Présentation du modèle

Le système repose sur un modèle d'intelligence artificielle pré-entraîné, que nous réentraînons spécifiquement à notre problématique grâce à un jeu de données adapté. Ce réentraînement permet d'améliorer la précision du modèle sur les images des salles ciblées.

5.2 Jeu de données

Le dataset est constitué d'images de différentes salles de cours, chacune associée à son nom correspondant. Actuellement, il contient environ **120 images par salle**.

Les images ont été sélectionnées de manière à mettre en évidence les caractéristiques distinctives de chaque salle, telles que :

- la disposition des tables ;
- le type de tableau ;
- l'éclairage ;
- les éléments décoratifs spécifiques.

Afin d'améliorer la robustesse du modèle, le jeu de données comprend également des images avec et sans présence humaine, afin de réduire les risques d'erreur liés à la détection de personnes.

5.3 Prétraitement des images

Avant l'entraînement, les images sont redimensionnées à une résolution de **250×250 pixels**. Ce redimensionnement permet à la fois :

- de réduire la taille des fichiers pour un gain d'espace de stockage ;
- d'accélérer le traitement par le modèle sans dégrader significativement la qualité de la reconnaissance.

5.4 Processus de reconnaissance

Lorsqu'une image est soumise, le modèle évalue son degré de similarité avec chacune des salles du jeu de données. Pour chaque salle, il calcule un **score de probabilité** correspondant à la correspondance entre l'image analysée et les caractéristiques apprises de la salle.

La salle présentant la probabilité la plus élevée est considérée comme la plus plausible. Cette approche permet d'obtenir une reconnaissance rapide et explicable, tout en offrant la possibilité d'afficher au besoin le pourcentage de confiance associé à chaque prédiction.

6 Conclusion

Ce sprint nous a permis d'initialiser le projet en choisissant le backend le plus adapté à notre projet tout en initialisant la partie frontend. La listes des fonctionnalités a pu etre également établie.

Le prochain sprint sera composé d'ajouts de fonctionnalités comme la récupération du flux caméra ou la création de la page d'accueil dans l'application, ainsi que de la modélisation de la base de données.

7 Liens utiles

— Repository GitHub du projet

https://github.com/invader237/sae5

— Tableau Trello du projet

https://trello.com/b/GN5yM7u6/sae-5