

## **Compte rendu N°4 : SAE 5.01 – Développement Avancé**

*Sujet : Création d'une application de reconnaissance par IA d'objets du monde réel en temps réel*

*Thème : Matériel Scolaire (stylos, cahiers, règle, etc...)*

### **I) Avancement (07/10/25) :**

Cette semaine, le travail a principalement porté sur la stabilisation de l'environnement Flutter et la résolution de problèmes techniques rencontrés lors du lancement du projet sur Android Studio.

Après la mise en place du modèle YOLOv8 et du serveur Flask, il a été nécessaire d'effectuer plusieurs vérifications sur la partie mobile pour assurer une bonne compatibilité entre Flutter, Gradle et les dépendances du projet.

Des tests ont été réalisés sur l'émulateur Android (Pixel 7), mais des erreurs de compilation liées à la version de Java et à certaines dépendances Flutter ont temporairement bloqué le déploiement de l'application.

### **II) Partie interface utilisateur (Thomas, Said, Alexandre) :**

Au cours de cette étape, nous avons finalisé et intégré la partie interface utilisateur du projet. L'ensemble des éléments développés sur la branche « feature/UserInterface » a été testé puis fusionné dans la branche principale (main).

Cette partie a notamment consisté à mettre en place la structure visuelle de l'application, à harmoniser le design et à assurer la cohérence entre les différentes pages. L'objectif était d'obtenir une interface claire, fonctionnelle et agréable à utiliser, tout en respectant les contraintes définies dans le cahier des charges.

### **III) Partie Intelligence Artificielle (Lilian) :**

Cette semaine, l'accent a été mis sur la maintenance technique et la préparation à l'intégration entre Flutter et le backend Flask.

Plusieurs ajustements ont été réalisés :

- Nettoyage complet du projet avec les commandes `flutter clean` et `gradlew clean`.
- Analyse et tentative de correction de l'erreur Gradle indiquant une incompatibilité entre Java 11 et Gradle 8, nécessitant une migration vers Java 17.
- Vérification des dépendances (camera, google\_mlkit\_object\_detection, etc.) et mise à jour du fichier `pubspec.yaml`.

- Tests de lancement de l'application sur l'émulateur Android et contrôle du bon fonctionnement du lien entre le code Flutter et les modules de détection.

Ces étapes ont permis d'identifier les causes des blocages de build et de préparer un environnement plus stable pour les prochaines phases.

#### **IV) Prochaines étapes :**

- Mettre à jour la configuration Java (migration vers Java 17) pour corriger définitivement les erreurs de build.
- Relancer les tests de l'application Flutter avec le modèle YOLOv8 intégré.
- Vérifier la communication entre Flutter et le serveur Flask pour la détection d'objets en temps réel.
- Commencer les premiers tests sur un appareil Android physique.