



Rapport SAÉ 5 — Sprint 3

Diego TRIVINO

Mathias FRIES

Alexandre KESSELER

Joshua BROCHOT

Table des matières

1	Introduction	3
2	Conception de la base de données	3
2.1	Objectif	3
2.2	Modèle conceptuel	3
2.3	Associations	3
2.4	Justification de la conception	3
3	Initialisation du projet	3
3.1	Architecture et organisation du projet	4
3.2	Outils et environnement de développement	4
3.3	Documentation et génération de code	4
3.4	Conteneurisation et maintenance	4
3.5	Difficultés rencontrées	4
4	Développement de la page principale	4
4.1	Réalisation des maquettes	4
4.2	Travail d'équipe et itérations visuelles	5
4.3	Développement de l'application avec React Native et Expo	5
4.4	Captures d'écran de l'application	5
5	Gestion de la caméra	6
6	Conclusion	7
7	Liens utiles	8

1 Introduction

L'objectif de cette itération était de renforcer l'intégration entre les différentes composantes du projet. Au cours de cette semaine, les travaux ont principalement porté sur la finalisation de l'initialisation du projet, la gestion du flux caméra, ainsi que la création d'une première maquette — comprenant les pages d'accueil, de caméra et de connexion. En parallèle, nous avons également conçu le schéma de la base de données, une étape essentielle pour structurer et optimiser les futures interactions entre l'application et le serveur.

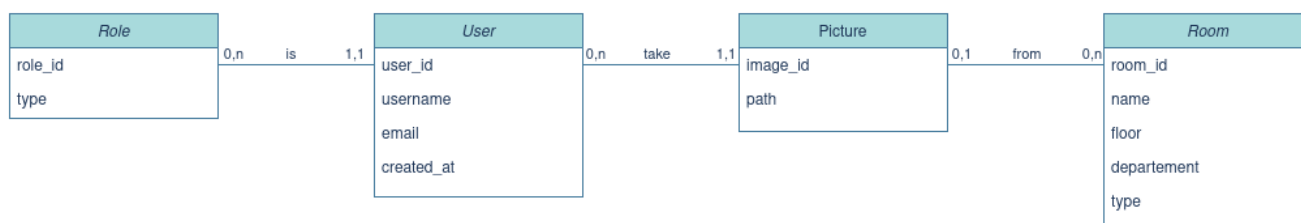
2 Conception de la base de données

2.1 Objectif

La base de données a été conçue pour organiser les informations liées à la reconnaissance des salles, tout en gérant les utilisateurs selon leur rôle dans le système. L'objectif est d'assurer une structure claire, cohérente et extensible pour le stockage des utilisateurs, des rôles, des salles et des images.

2.2 Modèle conceptuel

Le modèle repose désormais sur quatre entités principales : **User**, **Role**, **Room** et **Picture**.



L'entité **Role** définit les différents types d'utilisateurs du système. **User** représente les personnes utilisant l'application. **Room** représente les salles à reconnaître. Et enfin **Picture** pour stocker les images capturées ou utilisées pour l'apprentissage.

2.3 Associations

- **Role – User** : un rôle peut être attribué à plusieurs utilisateurs (1,N).
- **User – Picture** : un utilisateur peut être associé à plusieurs images (1,N).
- **Room – Picture** : une salle peut être liée à plusieurs images (1,N).

2.4 Justification de la conception

Cette conception repose sur quatre principes clés :

- **Hiérarchisation des utilisateurs** grâce à l'entité **Role**.
- **Traçabilité complète** entre les utilisateurs, les salles et les images.
- **Évolutivité** : la structure permet d'ajouter de nouvelles entités (ex. logs, détections, parcours 3D) sans rupture du modèle.

3 Initialisation du projet

Au cours de cette semaine, nous avons finalisé l'initialisation du projet. Le **backend** a été mis en place en utilisant **FastAPI**, conformément aux conclusions de l'étude de marché réalisée la semaine précédente pour le choix des technologies.

3.1 Architecture et organisation du projet

Nous avons opté pour une **architecture hexagonale par entité**, afin d'assurer :

- une meilleure **qualité du code**,
- une **évolutivité** facilitée,
- une **maintenabilité** renforcée à long terme.

3.2 Outils et environnement de développement

Plusieurs outils ont été intégrés pour faciliter la phase de développement :

- **Alembic** (basé sur **SQLAlchemy**) pour le *versionnage de la base de données*.
- Un **système de jeux de données de test statiques**, créés directement dans le code de l'application.
- Un **système de profils développeur**, permettant à chaque membre d'avoir :
 - une base de données dédiée,
 - un rechargement automatique du schéma et des données de test à chaque redémarrage,
 - une isolation complète entre les environnements de développement.

3.3 Documentation et génération de code

Nous avons également intégré une **documentation OpenAPI**, basée sur des fichiers **YAML**, dans le but de :

- générer automatiquement les **DTO** et **contrôleurs** de l'API,
- maintenir une documentation toujours **à jour et cohérente**,
- assurer un **haut niveau de qualité** dans la conception de l'API.

Cependant, nous rencontrons actuellement quelques problèmes concernant la génération automatique des contrôleurs. Cette partie devra donc être retravaillée ultérieurement, bien que ces ajustements ne soient pas critiques pour la suite du développement.

3.4 Conteneurisation et maintenance

Le backend a été **conteneurisé** afin de simplifier le déploiement et l'intégration continue. Plusieurs problèmes ont été corrigés au cours de cette étape, notamment dans le fichier `Makefile`.

3.5 Difficultés rencontrées

L'initialisation du backend s'est révélée plus complexe que prévu, principalement à cause du choix d'une technologie encore peu maîtrisée par l'équipe. De ce fait, la mise en place des pipelines **CI/CD** n'a pas pu être finalisée au cours de ce sprint, bien que leur conception ait été amorcée. Cette tâche sera reportée au prochain sprint.

4 Développement de la page principale

4.1 Réalisation des maquettes

Avant de commencer l'implémentation de l'application, on a réalisé des maquettes d'écrans avec l'outil Figma afin de définir l'identité visuelle et l'organisation des interfaces. Pour rendre ces maquettes plus réalistes, on a utilisé un *overlay* de téléphone permettant de simuler le rendu final sur un appareil mobile. Ces maquettes avaient également pour objectif de faciliter la discussion au sein de l'équipe projet et d'assurer une vision commune de l'interface utilisateur avant le début du développement.

4.2 Travail d'équipe et itérations visuelles

Les maquettes ont été présentées aux membres de l'équipe afin de recueillir des avis et d'ajuster les choix de conception avant de passer à la phase de développement. Plusieurs retours constructifs ont été pris en compte, notamment :

- repositionnement des éléments de la barre de navigation pour placer le bouton « Caméra » au centre, car il s'agit de la fonctionnalité principale de l'application ;
- ajout d'un bouton « Importer » sur la page d'accueil pour permettre l'import de photos ou vidéos depuis la galerie ;
- amélioration de l'espacement et de l'alignement des éléments pour un meilleur équilibre visuel.

Ces ajustements ont permis de valider collectivement la structure de l'application avant l'implémentation technique.

4.3 Développement de l'application avec React Native et Expo

Après validation des maquettes, on s'est concentré sur l'implémentation de la partie front-end en utilisant **React Native** associé au framework **Expo**. Ce choix technologique permet un développement mobile multiplateforme (Android et iOS) tout en conservant une base de code unique.

On a utilisé l'application **Expo Go** afin de tester rapidement le rendu directement sur un smartphone, ce qui a facilité les ajustements visuels tout au long du développement. Une structure claire de projet a été mise en place, avec séparation des composants, des pages, des styles et des constantes. Cette organisation permet de maintenir un code propre, réutilisable et facilement évolutif.

4.4 Captures d'écran de l'application

Cette section présente une vue d'ensemble des premières interfaces développées. Ces écrans correspondent aux maquettes validées en amont et constituent la base visuelle de l'application.

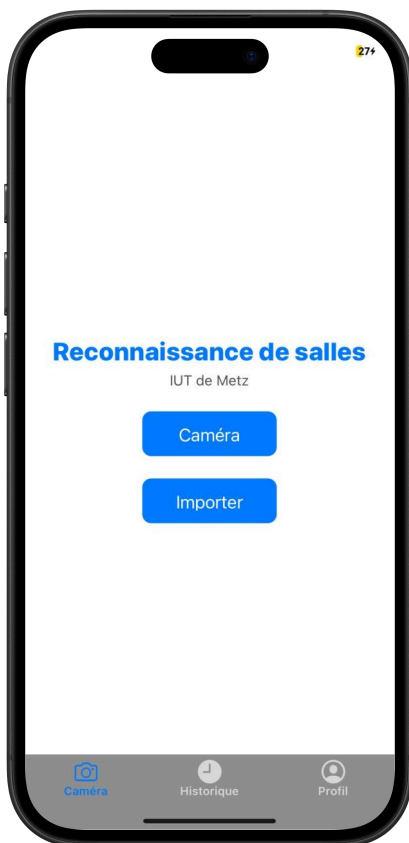


FIGURE 1 – Page d'accueil : accès aux fonctionnalités principales.

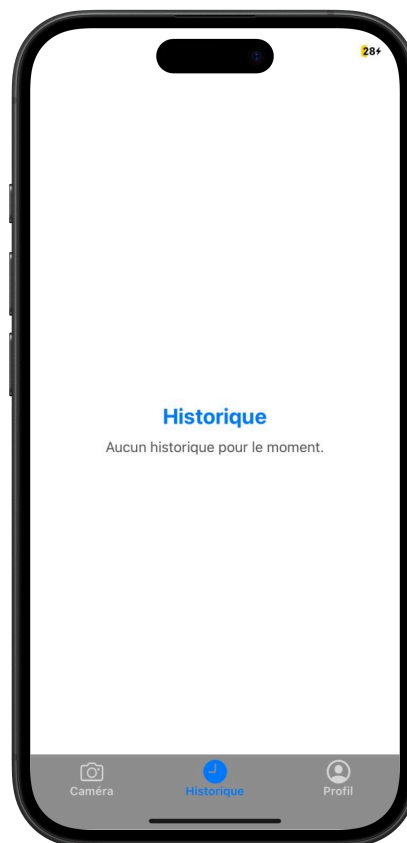


FIGURE 2 – Page Historique listant les reconnaissances passées.



FIGURE 3 – Page Profil dédiée aux informations utilisateur.

5 Gestion de la caméra

Le but de l'application étant la reconnaissance par photographie, il faut intégrer la caméra du téléphone dans l'application. On utilise ainsi le package 'expo-camera'. Ce dernier permet d'intégrer de nombreux contrôles de la caméra tel que la gestion des capteurs, le zoom ou encore le focus.

Le but est d'envoyer une image par seconde au serveur pour la reconnaissance de la salle. Une fois le bouton de reconnaissance enclenché, les images sont envoyées chaque seconde si le capteur camera respecte certaines conditions telles qu'un focus précis ou si l'image précédente est bien envoyée afin d'éviter doublons et blocage d'envoi.

De plus, l'utilisateur doit choisir quand démarre l'identification de salle, d'où la présence du bouton pour démarrer la reconnaissance de la salle (et donc de l'envoi des images). Ces images seront donc envoyées au serveur par le moyen d'un API pour laisser place au modèle d'intelligence artificielle. Et pour le login

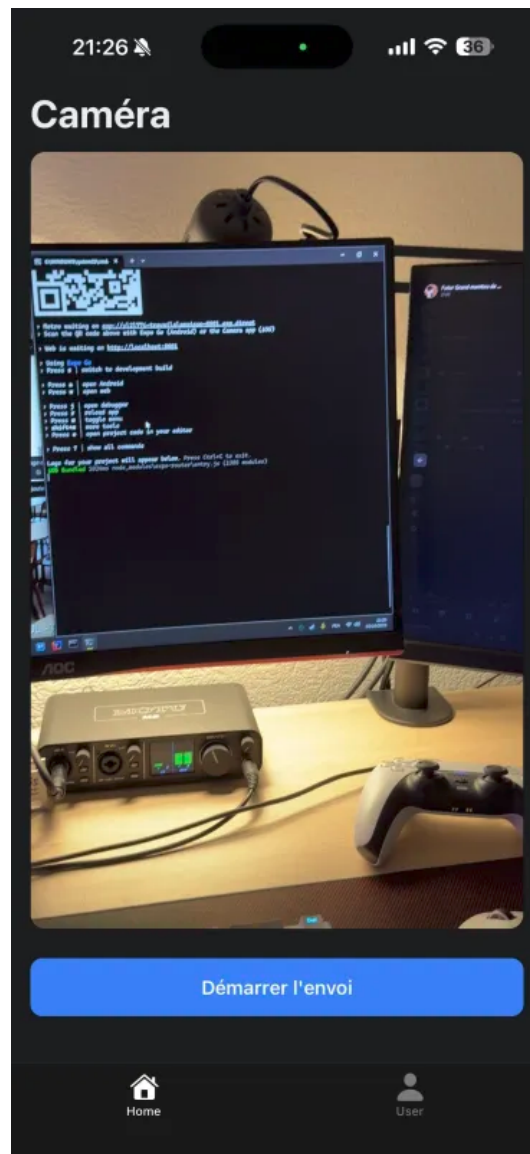


FIGURE 4 – Page de la caméra

6 Conclusion

Ce sprint a permis de rendre le projet pleinement **opérationnel** en finalisant les éléments essentiels de l'initialisation. Nous avons amorcé la mise en place des **pipelines CI/CD**, conçu les premières **maquettes** et intégré la **gestion du flux vidéo de la caméra**.

Un léger **retard** est toutefois à noter concernant la finalisation des pipelines CI/CD, dont la mise en œuvre complète est reportée à la prochaine itération.

Pour la suite, les objectifs sont :

- l'implémentation des **entités principales** et d'un **jeu de données de test**,
- la création de la **structure de navigation** de l'application,
- l'élaboration de **plusieurs chartes graphiques** afin d'explorer différentes identités visuelles,
- l'ajout de la fonctionnalité d'**import de photos depuis la galerie**,
- le développement d'une première version du **panel d'administration**, qui sera approfondie lors du prochain sprint.

Cette prochaine itération marquera également la **première mise en production de l'application**, une fois les nouvelles fonctionnalités testées et validées.

7 Liens utiles

- Repository GitHub du projet

<https://github.com/invader237/sae5>

- Tableau Trello du projet

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