AKYUREK Eren

BTS SIO SLAM

SESSION 2026

PAPPE

Liens :

StockTracker : https://github.com/Eren-Akyurek/StockTracker_GSB

Portfolio : https://eren-akyurek.github.io/portfolio/

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2025**

ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle (recto)

Épreuve E5 - Conception et développement d'applications (option SLAM)

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE	**N° réalisation : 2**
Nom, prénom : AKYUREK Eren	**N° candidat : 02249605006**
Épreuve ponctuelle ☒ **Contrôle en cours de formation** ☐	**Date :** 05 / 03 / 2026

Organisation support de la réalisation professionnelle

La réalisation s'inscrit dans le cadre d'un Projet Personnalisé Encadré (PPE) dispensé par le centre de formation. Il concerne une entreprise pharmaceutique fictive, nommée Laboratoire Galaxy Swiss Bourdin (GSB). L'objectif de ce projet spécifique était de concevoir une application mobile destinée à optimiser la gestion des stocks et le suivi de la relation client pour les collaborateurs en déplacement.

Intitulé de la réalisation professionnelle

Développement de l'application mobile "StockTracker GSB" : Gestion d'inventaire en temps réel et création de tickets SAV/Ventes.

Période de réalisation : Jan. 2026 / Mars 2026. **Lieu :** Dans le cadre de la formation

Modalité : ☒ **Seul(e)** ☐ **En équipe**

Compétences travaillées

☒ Concevoir et développer une solution applicative
☒ Assurer la maintenance corrective ou évolutive d'une solution applicative
☒ Gérer les données

Conditions de réalisation[5] (ressources fournies, résultats attendus)

Contexte de l'entreprise fictive GSB (laboratoire pharmaceutique) et cahier des charges exprimant le besoin d'un outil mobile pour les commerciaux et logisticiens en déplacement. Spécifications fonctionnelles concernant la gestion d'inventaire, le portefeuille client et le système de tickets.

Description des ressources documentaires, matérielles et logicielles utilisées[6]

Environnement de développement (IDE) :
Cursor / Visual Studio Code.

Client Mobile (Front-end) :
Framework Flutter (langage Dart) utilisant Material Design 3 pour l'UI/UX.
Utilisation des packages *Provider* (gestion d'état), *Dio* (requêtes HTTP) et *Flutter Secure Storage* (stockage chiffré du token).

Serveur et API REST (Back-end) :
Langage Python avec le framework FastAPI et le serveur Uvicorn.
Validation des données assurée par Pydantic. Sécurisation via Bcrypt (passlib) et PyJWT.

Base de données :
SGBD MySQL (hébergé en local via XAMPP), manipulé via l'ORM SQLAlchemy et le driver PyMySQL.
Outils complémentaires : Git et GitHub pour la gestion des versions.

[5] En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

[6] Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

Modalités d'accès aux productions[7] et à leur documentation[8]

Repository Github : https://github.com/Eren-Akyurek/StockTracker_GSB

Portfolio professionnel : Consultable en ligne à l'adresse suivante : https://Eren-Akyurek.github.io/portfolio/

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2025**

ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle (verso, éventuellement pages suivantes)

Épreuve E5 - Conception et développement d'applications (option SLAM)

[7] Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.* ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

[8] Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

Contexte et objectifs du projet :

Dans le prolongement de la modernisation du système d'information du laboratoire GSB, le projet "StockTracker" consiste en la création d'une application mobile.
Elle est destinée aux commerciaux et logisticiens en déplacement pour leur permettre de consulter les stocks de médicaments en temps réel, de gérer leur portefeuille client et de créer des tickets de suivi (SAV, Ventes).

Architecture Technique (Découplée) :
L'application repose sur une architecture Client-Serveur :

- Back-end (API RESTful) : Développé en Python avec le framework FastAPI et Uvicorn. La manipulation de la base de données MySQL se fait via l'ORM SQLAlchemy.
- Front-end (Client Mobile) : Développé avec le framework Flutter (Dart) en utilisant le pattern de gestion d'état "Provider" et la bibliothèque "Dio" pour les requêtes HTTP.

Fonctionnalités clés développées (Interface Utilisateur) :

- Tableau de bord dynamique : Affichage asynchrone des statistiques clés (clients actifs, tickets ouverts, produits en alerte) dès l'ouverture de l'application.
- Catalogue et Alertes : Consultation des produits avec implémentation de règles métiers (ex: affichage en rouge avec icône d'avertissement si la quantité en stock est inférieure ou égale au seuil d'alerte).
- Système de Tickets (CRUD) : Interface permettant aux commerciaux de lire et créer de nouveaux tickets depuis une fenêtre modale native.

Sécurité et protection des données :

- Authentification par Token : Utilisation de JSON Web Tokens (JWT) générés par l'API.
- Stockage sécurisé : Le token est conservé de manière chiffrée sur le téléphone de l'utilisateur (flutter_secure_storage).
- Protection des routes : Un intercepteur HTTP est configuré côté Flutter pour injecter automatiquement le token dans les en-têtes de chaque requête sortante vers l'API.
- Confidentialité : Hachage des mots de passe en base de données avec l'algorithme Bcrypt. Implémentation du "Soft Delete" (suppression logique) pour conserver l'historique des utilisateurs et clients.

Productions réalisées (Livrables) :

- Le code source de l'API REST (Python) et de l'application mobile (Flutter).
- Le script SQL de la base de données relationnelle (stocktracker_db).
- Un dossier documentaire (PAPPE) contenant l'architecture technique et les principales vues de l'application.

Sommaire

1. Présentation du Projet

- 1.1. Contexte GSB et Objectifs
- 1.2. Acteurs et Rôles

2. Architecture et Modélisation

- 2.1. Architecture technique (Stack Full-Stack)
- 2.2. Modèle Conceptuel de Données (MCD)

3. Manuel Utilisateur (Interfaces clés)

- 3.1. Écran de connexion et Tableau de bord
- 3.2. Catalogue produits et alertes de stock
- 3.3. Système de gestion des tickets

4. Réalisation Technique et Sécurité

- 4.1. Sécurité et Authentification (JWT)
- 4.2. Logique métier et Appels API

1. Présentation du Projet

1.1. Contexte GSB et Objectifs

L'application mobile "StockTracker" a été développée pour le laboratoire pharmaceutique Galaxy Swiss Bourdin (GSB).
L'objectif principal de ce projet est de fournir aux équipes sur le terrain (commerciaux et logisticiens) un outil mobile performant et connecté pour :

- Consulter les stocks de médicaments en temps réel lors de leurs déplacements.
- Gérer le portefeuille client.
- Créer et suivre des tickets d'intervention (SAV, Ventes).

1.2. Acteurs et Rôles

Le système gère trois profils d'utilisateurs distincts :

- Administrateur : Accès complet au système et à la gestion des comptes.
- Commercial : Création de tickets de vente et consultation du catalogue.
- Logisticien : Suivi des niveaux de stock et gestion des alertes.

2. Architecture et Modélisation

2.1. Architecture technique (Stack Full-Stack)

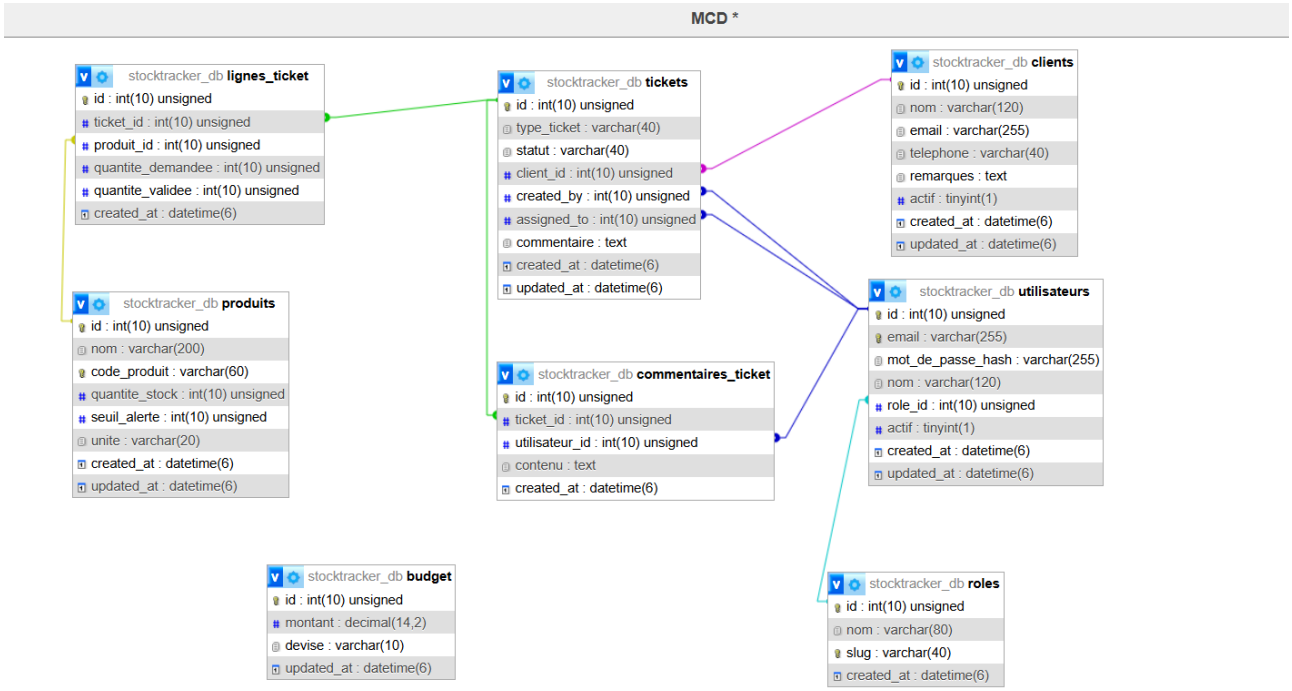
Afin de garantir la flexibilité et la sécurité des échanges de données, l'application repose sur une architecture Client-Serveur strictement découplée via une API RESTful :

- Back-end (Serveur / API) :
    - Langage / Framework : Python avec FastAPI et le serveur ASGI Uvicorn.
    - Base de données : MySQL (hébergé en local via XAMPP) avec la base stocktracker_db.
    - ORM et Validation : SQLAlchemy (driver PyMySQL) et Pydantic.

- Front-end (Client Mobile) :
    - Technologie : Framework Flutter (langage Dart).
    - UI/UX : Material Design 3.
    - Packages clés : dio (requêtes HTTP) et provider (gestion d'état).

2.2. Modèle Conceptuel de Données (MCD)

La base de données relationnelle est structurée autour de 5 entités principales : roles, utilisateurs, clients, produits et tickets. Une règle métier importante a été implémentée : le "Soft Delete" (suppression logique via une colonne actif) pour les utilisateurs et les clients, évitant ainsi la perte d'historique.



3. Manuel Utilisateur (Interfaces clés)

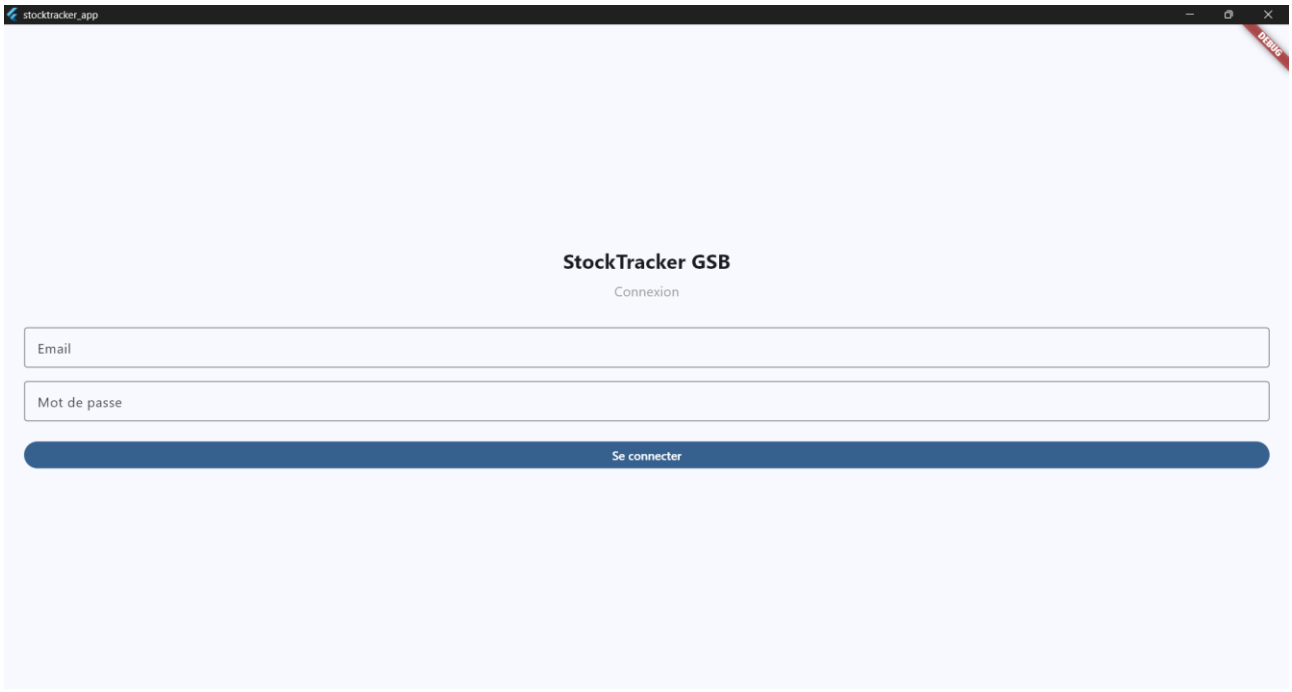
3.1. Écran de connexion et Tableau de bord

L'accès à l'application est restreint. Dès le lancement, un écran de connexion (LoginView) demande les identifiants de l'utilisateur pour récupérer un jeton de sécurité (JWT).
Une fois authentifié, l'utilisateur atterrit sur un tableau de bord dynamique (DashboardView).

- Optimisation des performances : Lors de l'initialisation, l'application utilise la méthode Future.wait de Dart pour lancer des requêtes asynchrones en parallèle vers l'API.

- Statistiques : Affichage de "Cards" résumant le nombre de clients actifs, de tickets ouverts et de produits en alerte de stock.

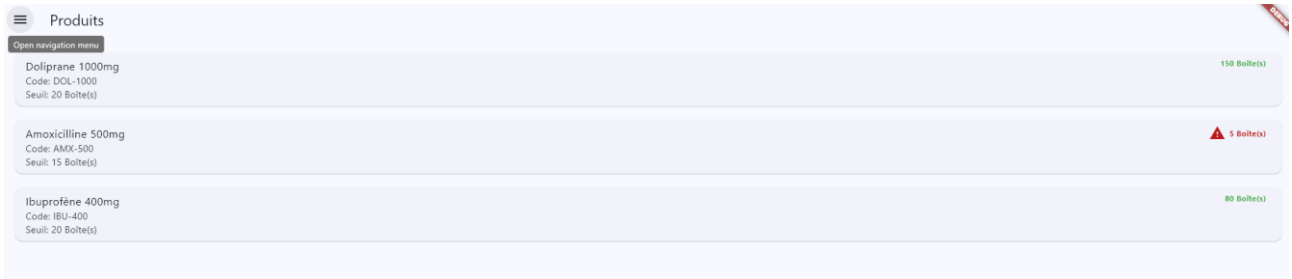
- Navigation : Un menu latéral (Drawer) géré par le package Provider permet de naviguer de manière fluide sans recharger l'application entière.



3.2. Catalogue produits et alertes de stock

L'écran des produits (ProductsView) liste l'inventaire complet récupéré depuis la base de données via l'API. L'affichage gère les états de chargement, d'erreur et de liste vide grâce au widget FutureBuilder.

- Règle métier visuelle : Si la quantite_stock d'un médicament est inférieure ou égale à son seuil_alerte, la quantité s'affiche automatiquement en rouge accompagnée d'une icône d'avertissement (Icons.warning) pour alerter le logisticien.



3.3. Système de gestion des tickets (CRUD)
Les commerciaux peuvent créer et suivre des tickets d'intervention directement sur le terrain.

- Création : Un clic sur le bouton d'action flottant (FloatingActionButton) ouvre une fenêtre modale native (showModalBottomSheet) contenant le formulaire de saisie.
- Typage : L'utilisateur sélectionne le type de ticket ("Vente", "SAV", ou "Autre") via un menu déroulant.

- Traitement : À la validation, une requête HTTP POST est envoyée à l'API, la modale se ferme, et la liste se rafraîchit automatiquement. Le ticket prend le statut "Nouveau" par défaut.

4. Réalisation Technique et Sécurité

4.1. Sécurité et Authentification (JWT)

La sécurisation des échanges entre l'application Flutter et l'API Python a été un enjeu majeur du développement.

- Hachage des mots de passe : Côté serveur, l'algorithme Bcrypt (via la bibliothèque passlib) est utilisé pour ne jamais stocker de mots de passe en clair dans la base MySQL.
- Authentification par Token (JWT) : Les sessions sont gérées via des JSON Web Tokens signés par PyJWT.
- Stockage chiffré : Côté client (Flutter), le token reçu est conservé dans un espace sécurisé du téléphone grâce au package flutter_secure_storage.

4.2. Logique métier et Appels API (Intercepteur HTTP)

Pour optimiser le code et garantir la sécurité de chaque requête vers l'API, une classe dédiée ApiClient a été configurée avec le package dio.

- Intercepteur HTTP : Un "Interceptor" lit automatiquement le token JWT stocké et l'injecte dans l'en-tête HTTP (Authorization: Bearer <token>) de chaque requête sortante.
- Gestion d'expiration : Si le token expire et que l'API renvoie une erreur 401 Unauthorized, l'intercepteur intercepte cette erreur, déclenche une déconnexion automatique, purge le stockage local et redirige l'utilisateur vers l'écran de Login (LoginView).