Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ FÉLIX HOUPHOUËT-BOIGNY

UFR DE MATHEMATIQUES ET INFORMATIQUE

RÉPUBLIQUE DE CÔTE D'IVOIRE

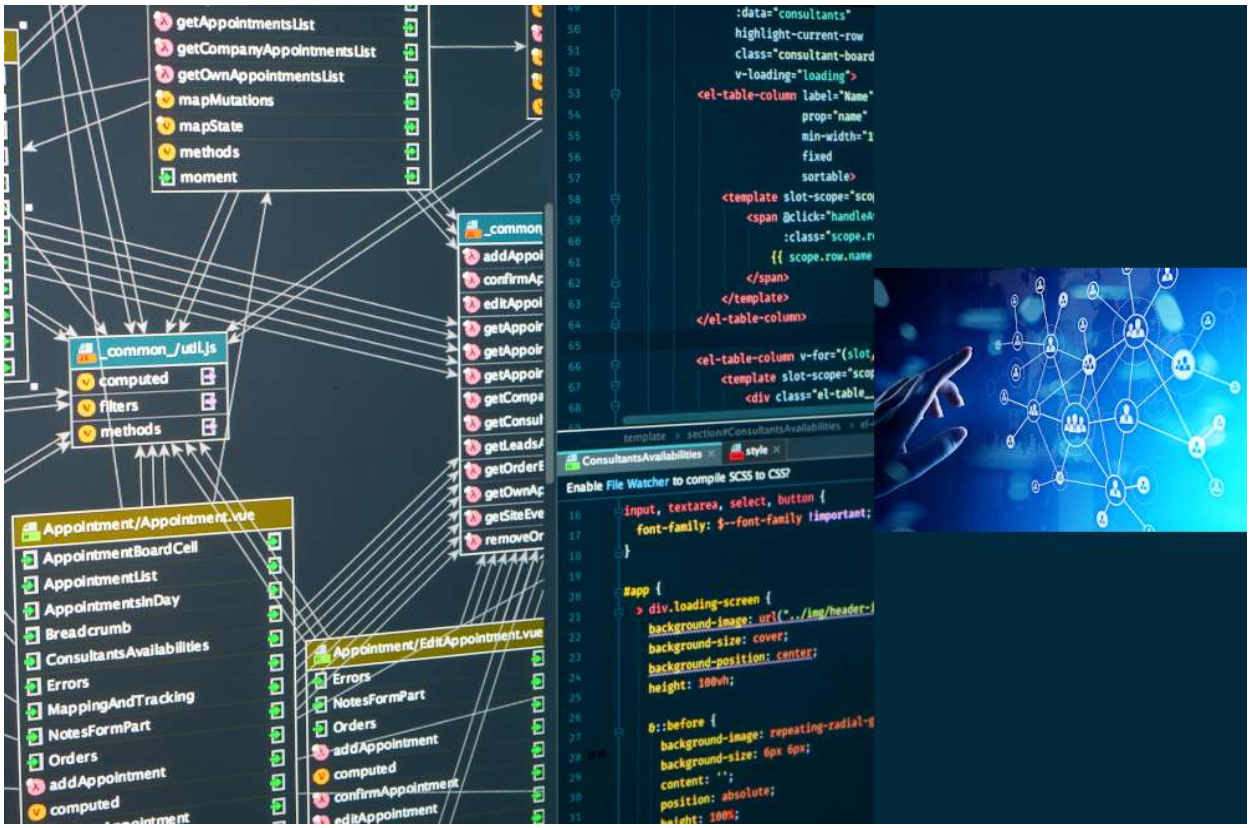
Union – Discipline - Travail

Année Universitaire 2025 – 2026

Licence 3 – Réseaux Informatique Sécurité et Télécommunications

PROJET DE FIN DE MODULE DE GENIE LOGICIEL

MODELISATION D'UN SYSTEME DE GESTION DE BIBLIOTHEQUE



Produit par :

AGBENONZAN Kossivi Jacques Junior

Encadreur : Dr GBAME

Sommaire

Introduction

De nos jours, le logiciel occupe une place incontournable dans le domaine des technologies. Dans le cadre de notre formation à l'Université Félix Houphouët-Boigny, le projet de fin de module en génie logiciel a pour objectif de mettre en pratique les concepts étudiés en modélisation et en développement d'applications.

Chaque étudiant doit, pour le projet choisi, élaborer les diagrammes UML (cas d'utilisation, séquence et classes) afin de représenter le système dans son ensemble, puis proposer une application fonctionnelle avec ses différentes interfaces (back-end et front-end) en utilisant le langage **Python**.

Nous avons retenu le **Projet 1 : Modélisation d'un Système de Gestion de Bibliothèque**.

- **Description** : Concevoir un modèle UML complet permettant de gérer une bibliothèque (emprunts, retours, réservations, gestion des usagers et des ouvrages).
- **Objectifs** : Réaliser les diagrammes de classes, cas d'utilisation et séquences pour décrire le fonctionnement, puis développer une application académique illustrant ces fonctionnalités.
- **Compétences mobilisées** : UML (diagrammes de classes, séquences, cas d'utilisation), modélisation orientée objet, gestion de base de données.
- **Outils utilisés** : UML, Python (Flask), SQLite, HTML/CSS.

1. Analyse et Modélisation UML

1.1. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet de représenter les interactions principales entre les acteurs du système et les fonctionnalités offertes par l'application de gestion de bibliothèque.

Acteurs

Deux acteurs principaux interviennent dans le système :

- **Usager (Adhérent)** : il correspond à l'utilisateur standard de la bibliothèque. Ses actions incluent l'inscription, la connexion, la consultation du catalogue, la demande d'emprunt, la réservation d'ouvrages, l'annulation de réservation et le suivi de son tableau de bord personnel.
- **Administrateur** : il supervise l'ensemble du système. Ses responsabilités incluent la gestion des ouvrages, la validation ou le rejet des demandes d'emprunt, l'enregistrement des retours, la clôture des réservations et la suspension ou la levée de suspension des usagers.

1.2. Principaux cas d'utilisation

Les cas d'utilisation identifiés couvrent les fonctionnalités essentielles du système :

- **Inscription et authentification** : création de compte, connexion et déconnexion.
- **Gestion des emprunts** : demande d'emprunt par l'usager, validation ou rejet par l'administrateur, enregistrement du retour.
- **Gestion des réservations** : réservation d'un ouvrage, annulation par l'usager, clôture par l'administrateur.
- **Gestion des suspensions** : suspension manuelle ou automatique d'un usager, levée de suspension par l'administrateur.
- **Consultation du catalogue** : recherche multi-critères et affichage des ouvrages disponibles.
- **Tableaux de bord** : suivi personnel pour l'usager et tableau global pour l'administrateur.

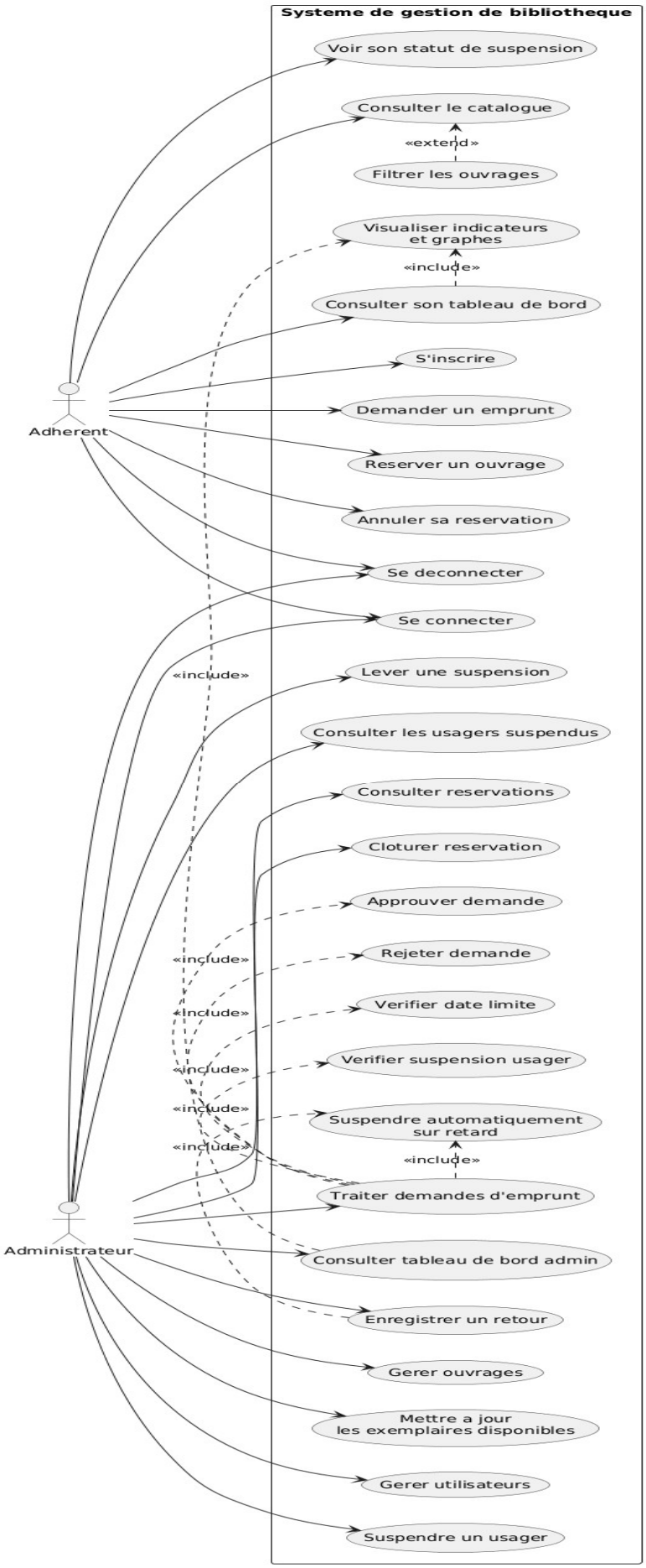


Fig1 - Diagramme de Cas d'utilisation

1.3. Diagramme de classes

Le diagramme de classes UML permet de représenter la structure statique du système de gestion de bibliothèque. Il met en évidence les principales entités manipulées par l'application, leurs attributs, ainsi que les relations entre elles.

Classes principales

- **Utilisateur** : représente les usagers et administrateurs du système.
  - ✓ Attributs : id, nom_complet, email, téléphone, type_utilisateur, rôle, mot_de_passe_hash, est_suspendu, raison_suspension, origine_suspension, date_suspension.
- **Livre** : représente les ouvrages disponibles dans la bibliothèque.
  - ✓ Attributs : id, titre, auteur, isbn, éditeur, année_publication, nombre_exemplaires, exemplaires_disponibles.
- **DemandeEmprunt** : représente une demande d'emprunt effectuée par un usager.
  - ✓ Attributs : id, date_demande, statut.
- **Emprunt** : représente un emprunt validé par l'administrateur.
  - ✓ Attributs : id, date_emprunt, date_limite, date_retour.
- **Réservation** : représente une réservation d'ouvrage effectuée par un usager.
  - ✓ Attributs : id, date_reservation, statut.

Relations

- Un **Utilisateur** peut effectuer plusieurs **Demandes d'emprunt**, plusieurs **Emprunts** et plusieurs **Réservations** (relations 1..*).
- Un **Livre** peut être associé à plusieurs **Demandes d'emprunt**, plusieurs **Emprunts** et plusieurs **Réservations** (relations 1..*).
- Une **Demande d'emprunt** peut aboutir à un **Emprunt** si elle est approuvée.
- Un **Emprunt** peut entraîner une suspension automatique de l'**Utilisateur** en cas de retard.

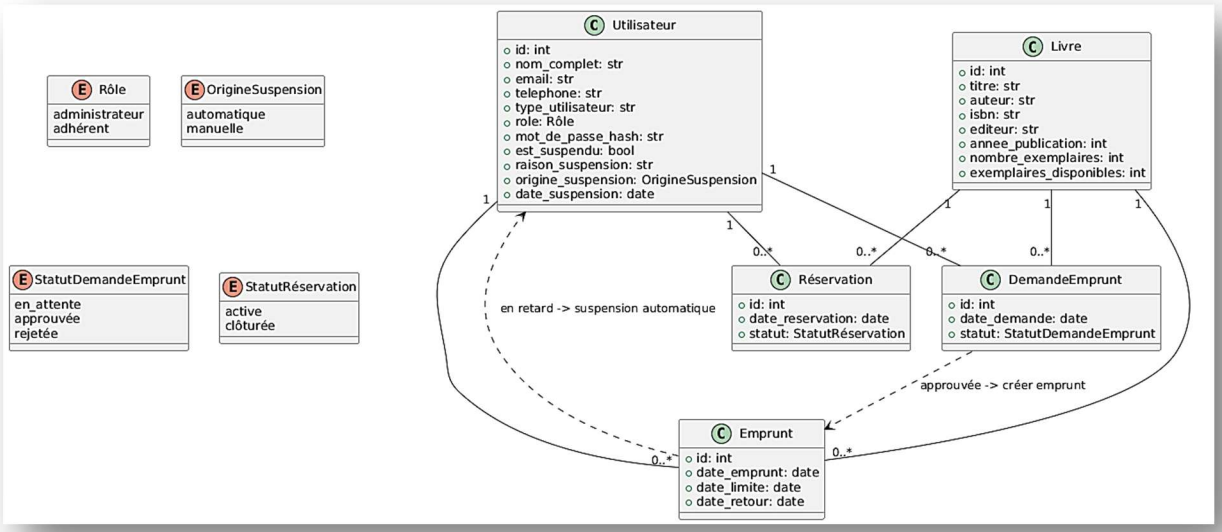


Fig 2 – Diagramme de Classe

1.4. Diagrammes de séquence

Les diagrammes de séquence UML permettent de représenter la dynamique du système, c'est-à-dire l'enchaînement des interactions entre les acteurs, les interfaces et la base de données au cours des différents scénarios métier. Ils illustrent la logique de fonctionnement et les règles de gestion de la bibliothèque.

Scénario 1 : Demande et validation d'emprunt

Ce scénario décrit le processus par lequel un usager formule une demande d'emprunt et l'administrateur valide ou rejette cette demande.

- L'usager choisit un ouvrage et soumet une demande via l'interface.
- L'application vérifie l'état du compte (suspension éventuelle) et l'existence d'une demande en attente.
- Si la demande est valide, elle est enregistrée en base.
- L'administrateur peut ensuite valider la demande en fixant une date limite ou la rejeter.
- En cas de validation, un emprunt est créé et le stock du livre est mis à jour.

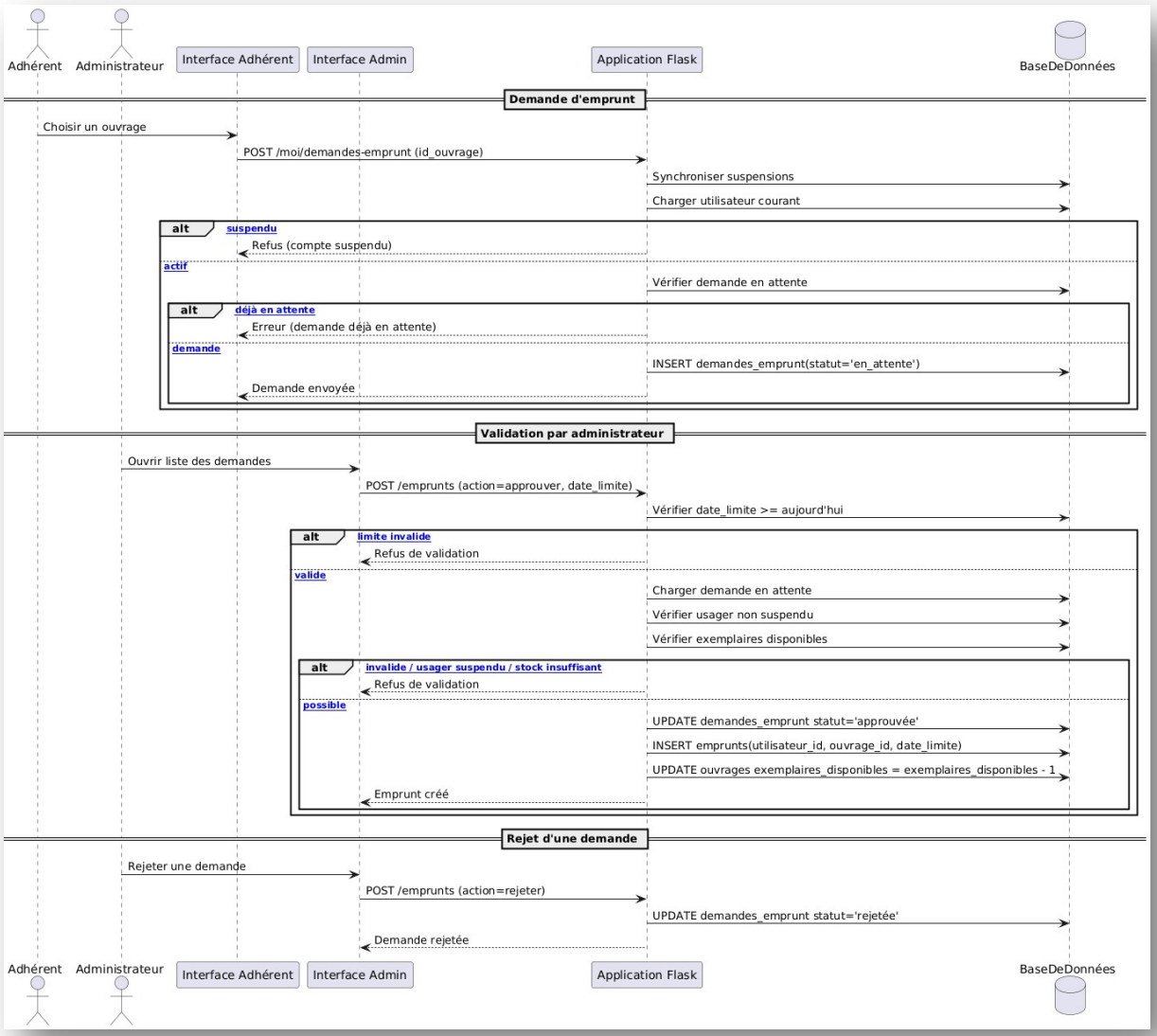


Fig 3 – Diagramme de Séquences pour les emprunts

Scénario 2 : Réservation et annulation

Ce scénario illustre la gestion des réservations d'ouvrages par les usagers et leur clôture par l'administrateur.

- L'usager peut réserver un ouvrage si son compte est actif.
- Il peut également annuler sa réservation.
- L'administrateur peut clôturer une réservation.

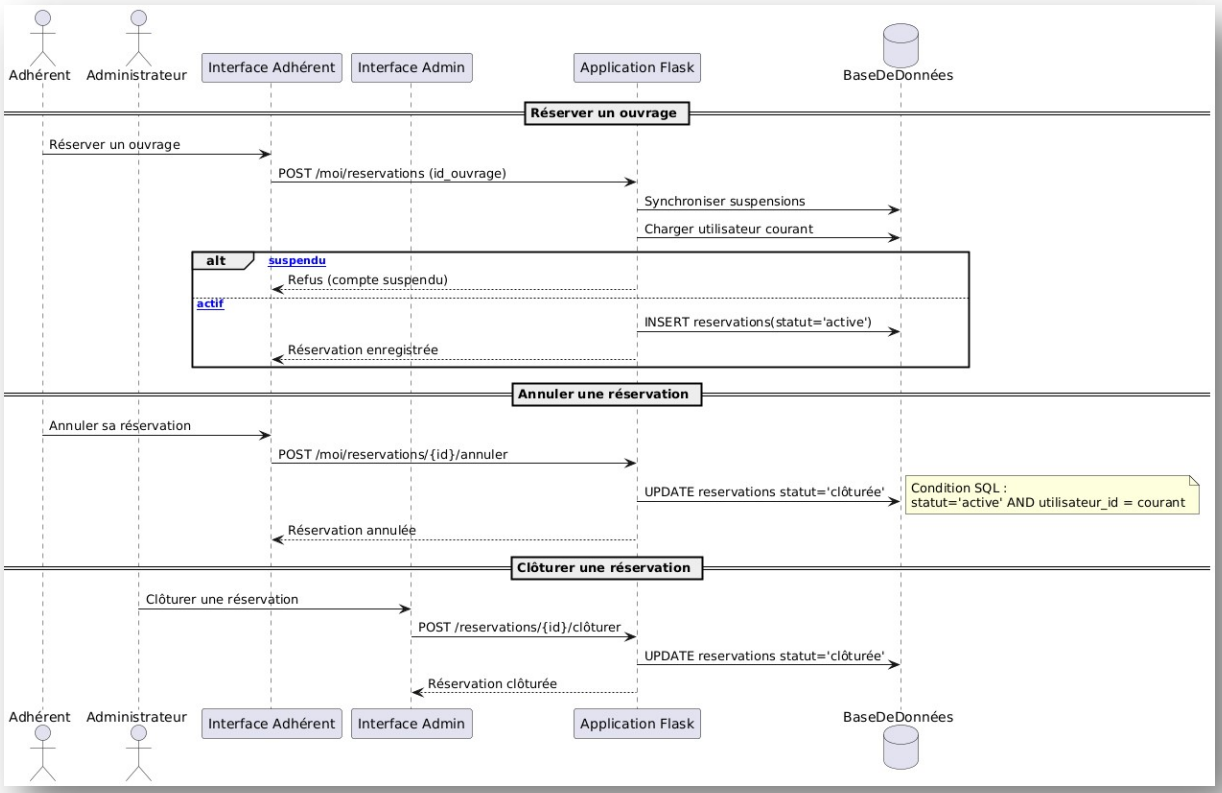


Fig 4 - Diagramme de Séquences pour les Reservations et annulations

Scénario 3 : Retour d'emprunt

Ce scénario décrit l'enregistrement du retour d'un emprunt par l'administrateur.

- L'administrateur sélectionne l'emprunt concerné.
- L'application vérifie que l'emprunt est actif.
- Si valide, la date de retour est enregistrée et le stock du livre est incrémenté.

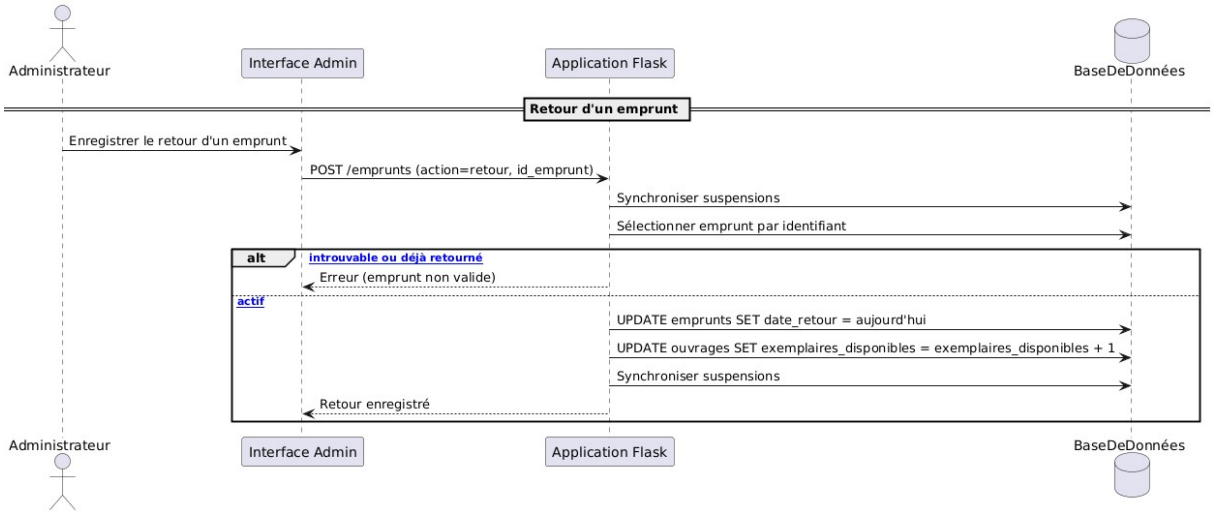


Fig 5 - Diagramme de Séquences pour les Retour d'emprunts

Scénario 4 : Suspension manuelle et automatique

Ce scénario illustre la gestion des suspensions d'usagers.

- L'administrateur peut suspendre ou lever la suspension d'un usager.
- Le système applique automatiquement une suspension en cas de retard d'emprunt.

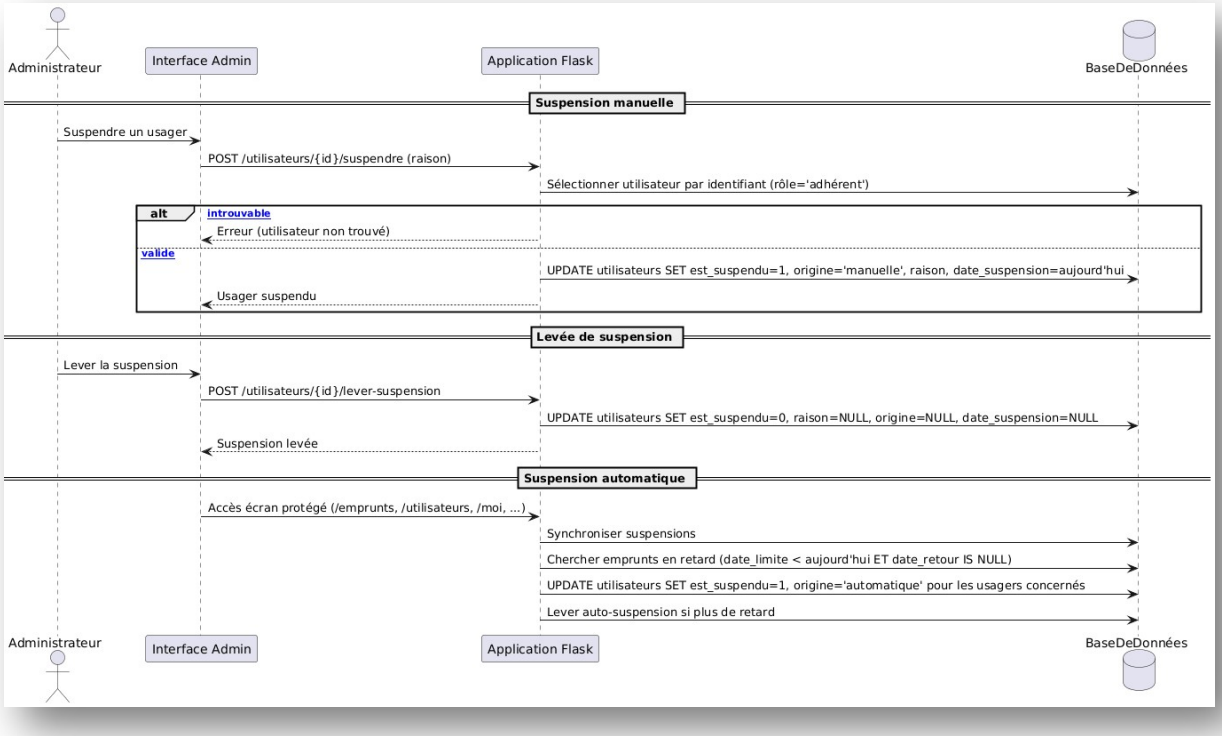


Fig 5 - Diagramme de Séquences pour les Retour d'emprunts

2. Conception de l'application

2.1. Architecture

L'application repose sur une architecture simple et modulaire :

- **Backend** : développé avec **Flask**, il gère les routes, la logique métier et l'authentification des utilisateurs. Les rôles (admin, user) sont pris en compte pour sécuriser les opérations.
- **Frontend** : conçu en **HTML/CSS** avec les templates **Jinja2**, il propose deux interfaces distinctes : une pour l'usager et une pour l'administrateur.
- **Base de données** : gérée par **SQLite** via le fichier library.db. Elle stocke les informations relatives aux usagers, aux ouvrages, aux demandes d'emprunt, aux emprunts et aux réservations.

2.2. Interfaces

- **Usager** :

- Dashboard personnel (indicateurs, alertes, activité récente).
- Consultation du catalogue et recherche multi-critères (titre, auteur, ISBN, éditeur, année).
- Création de demandes d'emprunt et de réservations.
- Suivi des emprunts, demandes et réservations.

- **Administrateur** :
  - Dashboard global (statistiques du système).
  - Gestion des ouvrages (ajout, recherche, mise à jour du stock).
  - Gestion des comptes usagers et administrateurs.
  - Validation ou rejet des demandes d'emprunt avec date limite obligatoire.
  - Gestion des retours et des suspensions (manuelles ou automatiques).

3. Implémentation technique

3.1. Structure du projet

- **app.py** : contient la logique Flask (routes, authentification, workflow métier).
- **templates/** : regroupe les vues HTML/Jinja2 pour les interfaces usager et administrateur.
- **static/style.css** : assure le style moderne et responsive de l'application.
- **library.db** : base SQLite locale, créée automatiquement au premier lancement.

3.2. Workflow typique

Le fonctionnement de l'application suit un cycle simple :

a. **Connexion** : l'usager ou l'administrateur s'authentifie.

b. **Action** : une opération est initiée (ex. demande d'emprunt).

c. **Mise à jour DB** : la base de données est modifiée selon la logique métier (insertion, mise à jour, suppression).

d. **Retour interface** : l'utilisateur reçoit une réponse via son tableau de bord ou l'interface correspondante.

4. Limites et perspectives

4.1. Limites actuelles

Bien que l'application réponde aux objectifs pédagogiques fixés, certaines fonctionnalités avancées n'ont pas été intégrées dans cette version académique :

- **Gestion des mots de passe** : absence de mécanisme de réinitialisation par email ou de récupération sécurisée.
- **Déploiement et maintenance** : pas de pipeline de déploiement continu ni de monitoring automatisé.
- **Tests automatisés** : absence de couverture de tests unitaires et fonctionnels pour valider la robustesse du système.
- **Sécurité limitée** : gestion basique de l'authentification sans chiffrement avancé ni protection contre les attaques courantes (CSRF, XSS).

4.2. Perspectives d'amélioration

Dans une optique d'évolution vers une application plus complète et professionnelle, plusieurs pistes d'amélioration peuvent être envisagées :

- **Renforcement de la sécurité** : mise en place de mécanismes de hachage et de salage robustes pour les mots de passe, ajout de fonctionnalités de réinitialisation par email, et intégration de protections contre les attaques web.
- **Tests automatisés** : développement d'une suite de tests unitaires et fonctionnels pour garantir la fiabilité du système et faciliter les mises à jour.
- **Interface utilisateur moderne** : amélioration du design avec des frameworks front-end (Bootstrap, TailwindCSS, ou React) pour une expérience plus fluide et responsive.
- **Déploiement et scalabilité** : mise en place d'un pipeline CI/CD, hébergement sur un serveur cloud (Heroku, AWS, Azure), et optimisation de la base de données pour supporter un plus grand volume d'utilisateurs et d'ouvrages.
- **Fonctionnalités supplémentaires** : notifications par email ou SMS pour les échéances d'emprunt, recommandations personnalisées d'ouvrages, et statistiques avancées pour l'administrateur.

5. Liens et Mot de passe Administrateur par défaut de l'application

Liens : https://gestion-d-une-bibliotheque--juniorKossivi.replit.app

User Administrateur : **admin@bibliotheque.local**

Mot de passe : admin1234

6. Conclusion

Ce projet de **modélisation et de développement d'un système de gestion de bibliothèque** nous a permis de mettre en pratique les concepts fondamentaux du génie logiciel. À travers l'élaboration des diagrammes UML (cas d'utilisation, classes et séquences), nous avons pu traduire les besoins fonctionnels en une représentation claire et structurée, facilitant ainsi la compréhension et la conception du système.

La réalisation de l'application avec **Flask, SQLite et HTML/CSS** a constitué une étape essentielle, reliant la théorie à la pratique. Elle illustre comment une modélisation rigoureuse peut guider efficacement l'implémentation technique et donner naissance à une solution fonctionnelle. Ce travail a mobilisé des compétences variées : modélisation orientée objet, gestion de base de données, conception d'interfaces et logique métier.

Au-delà de l'aspect académique, ce projet démontre l'importance de l'ingénierie logicielle dans la création de systèmes fiables et évolutifs. Il ouvre également des perspectives d'amélioration, notamment en matière de sécurité, de tests automatisés et de modernisation de l'interface utilisateur.

En somme, ce projet nous a permis de consolider nos acquis, de développer une vision intégrée du cycle de vie logiciel et de renforcer notre capacité à concevoir des solutions adaptées aux besoins réels. Il constitue une étape significative dans notre parcours de formation et un socle solide pour nos futures réalisations en informatique.