Gestion de Bibliothèque

1. Contexte du Projet

Dans le cadre de l'amélioration de l'expérience utilisateur en bibliothèque, le projet vise à développer une borne en libre-service permettant aux adhérents d'effectuer eux-mêmes des opérations courantes telles que la consultation du catalogue, l'emprunt et le retour de livres, ainsi que la gestion de leur compte personnel. L'application a pour objectif de simplifier la gestion des flux de livres et d'offrir une interface intuitive, accessible via une borne interactive.

2. Objectifs et Périmètre de l'Application

L'application doit permettre de :

- **Consulter le catalogue de livres** : Afficher la liste des livres disponibles avec leurs informations (titre, auteur, ISBN, disponibilité).
- **Gérer les emprunts et retours** : Permettre à l'utilisateur d'emprunter un livre s'il est disponible et de procéder à son retour en renseignant l'ISBN.
- **Accéder à l'information utilisateur** : Consulter et modifier les informations de son compte (nom, prénom, email, mot de passe) et visualiser l'historique de ses emprunts.

- **Fonctionnalités spécifiques au bibliothécaire** : Accéder à la liste complète des adhérents et consulter l'historique des emprunts et retours pour un suivi global.

3. Architecture Technique et Description Détaillée

Environnement de Développement

- **Langage** : Java
- **IDE** : VScode
- **Base de Données** : MySQL (accès via mysql-connector)
- **Framework UI** : JavaFX, structuré autour du pattern MVC (Modèle-Vue-Contrôleur)

Architecture MVC

L'application est organisée selon le modèle MVC, ce qui permet de séparer les préoccupations et d'assurer une meilleure maintenabilité :

- **Modèle (Model)**

  Contient les classes métier (par exemple, *Livre*, *Utilisateur*, *Auteur*, *Historique*, *Emprunt*) ainsi que la classe centrale `Model` qui gère l'accès à la base de données, l'exécution des requêtes SQL et la manipulation des données. C'est également dans cette couche que s'effectuent les mises à jour liées aux emprunts, retours et modifications de comptes.

- **Vue**

  Composée des interfaces graphiques créées avec JavaFX et WindowBuilder. Chaque vue correspond à une page de l'application (Accueil, Login, DétailLivre, Emprunt, Retour, etc.) et est conçue pour être simple, intuitive et responsive.

- **Contrôleur (Controller)**

  Chaque vue dispose d'un contrôleur dédié (par exemple, `LoginController`, `AccueilController`, `DetailLivreController`, etc.) qui gère la logique d'interaction, les validations d'entrées et la navigation entre les pages. Les contrôleurs

communiquent avec le modèle pour récupérer ou mettre à jour les données et avec un utilitaire (`SceneManager`) pour gérer le changement de vues.

Base de Données et Modèle de Données

La base de données MySQL contient plusieurs tables (utilisateur, livre, auteur, historique, etc.) qui structurent l'information nécessaire au bon fonctionnement de la borne. Les principaux éléments du modèle sont :

- **Utilisateur/Adhérent** : Représenté par la classe `Utilisateur`, qui contient les informations personnelles (nom, prénom, email) et le rôle (adherent ou bibliothécaire).
- **Livre** : La classe `Livre` regroupe les informations relatives aux livres (titre, ISBN, disponibilité) et référence un auteur.
- **Auteur** : La classe `Auteur` stocke les informations sur l'auteur du livre (nom, prénom, date de naissance, description).
- **Historique et Emprunt** : L'historique des emprunts et retours est géré via la table « historique » et représenté par la classe `Historique`. La classe `Emprunt` permet de visualiser les emprunts en cours et passés avec les dates d'emprunt et de retour.

4. Logique Métier et Flux Fonctionnels

La logique métier est implémentée principalement dans la classe `Model` et répartie entre les différents contrôleurs. Voici quelques points clés :

Authentification et Gestion de Session

- **Connexion** : L'utilisateur entre son email et son mot de passe dans le formulaire de connexion. Le contrôleur `LoginController` valide ces informations en interrogeant la base via `Model.verifierConnexion()`. En cas de succès, une instance de `SessionUtilisateur` est mise à jour et l'utilisateur est redirigé en fonction de son rôle (bibliothécaire ou adhérent).*Voir LoginController et SessionUtilisateur*

Consultation du Catalogue

- **Affichage et Recherche** : La vue d'accueil affiche la liste des livres. L'utilisateur peut rechercher ou filtrer les livres par titre, auteur ou disponibilité.

Le contrôleur `AccueilController` utilise des listes observables et des filtres JavaFX pour mettre à jour dynamiquement l'affichage.

Emprunt d'un Livre

- **Processus d'Emprunt** :
  1. L'utilisateur sélectionne un livre dans le catalogue et ouvre la page de détail.
  2. Si le livre est disponible, il peut lancer la procédure d'emprunt via `DetailLivreController` qui redirige vers `EmpruntController`.
  3. Dans la fenêtre d'emprunt, l'utilisateur doit saisir son email. Ce dernier est vérifié (format et existence dans la base).
  4. Si l'email correspond à un adhérent, la méthode `Model.emprunterLivre()` met à jour la base pour associer le livre à l'adhérent et enregistre l'opération dans l'historique.
  5. La vue se rafraîchit pour indiquer que le livre n'est plus disponible.

Retour d'un Livre

- **Processus de Retour** :
  1. L'utilisateur saisit l'ISBN du livre à retourner dans la vue dédiée (contrôlée par `RetourController`).
  2. La méthode `Model.retournerLivre()` vérifie que l'ISBN correspond bien à un livre emprunté et met à jour la base en réinitialisant l'assignation de l'adhérent.
  3. L'historique est mis à jour via `Model.enregistrerRetour()` et une notification de succès est affichée.

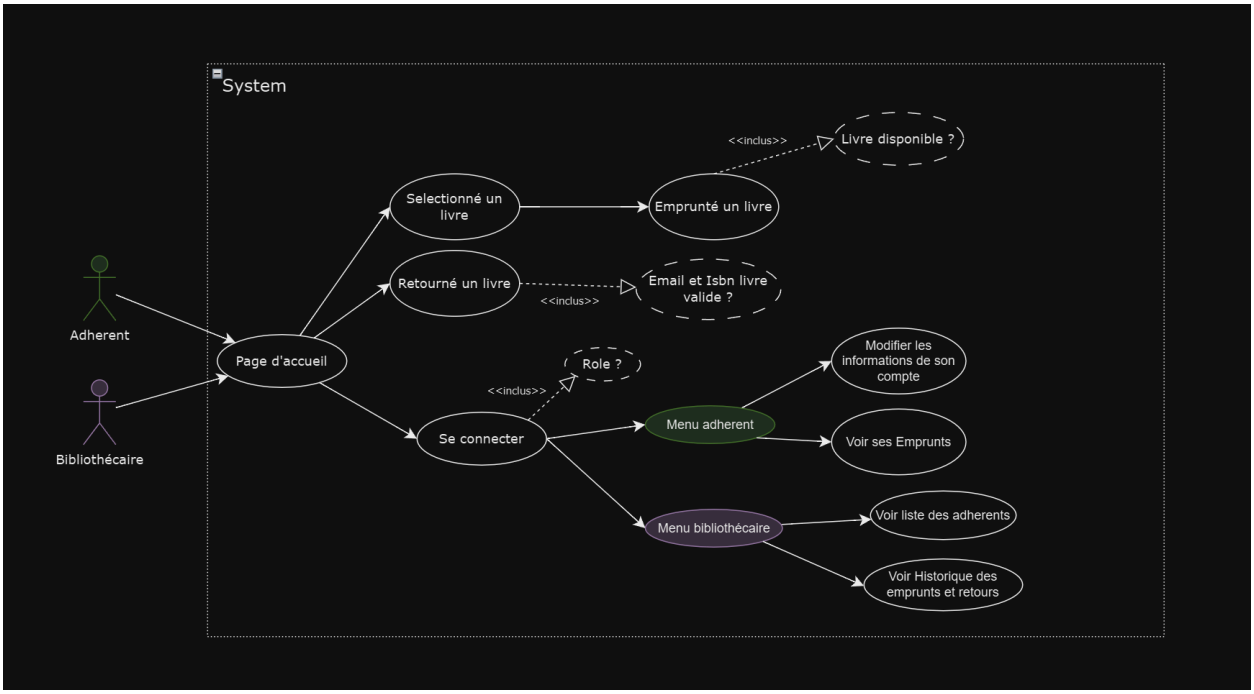
Gestion du Compte Utilisateur

- **Modification du Compte** : L'utilisateur peut modifier ses informations personnelles et, le cas échéant, mettre à jour son mot de passe. Le contrôleur `ModifierCompteController` vérifie la validité des données (vérification du mot de passe actuel, cohérence des nouveaux mots de passe, etc.) avant de lancer la mise à jour en base via `Model.updateUtilisateur()`.

Fonctions Spécifiques pour le Bibliothécaire

- **Gestion des Adhérents et de l'Historique** :
  - Le bibliothécaire peut accéder à une vue listant tous les adhérents (`ListeAdherentsController`) et consulter leur dernier emprunt ou retour.
  - Une autre vue permet de consulter l'historique complet des opérations d'emprunt et de retour (`HistoriqueController`), facilitant ainsi la gestion administrative.

5. Cas d'Utilisation



Le diagramme de cas d'utilisation met en évidence les principaux scénarios fonctionnels de l'application. Voici une description synthétique de quelques cas d'utilisation :

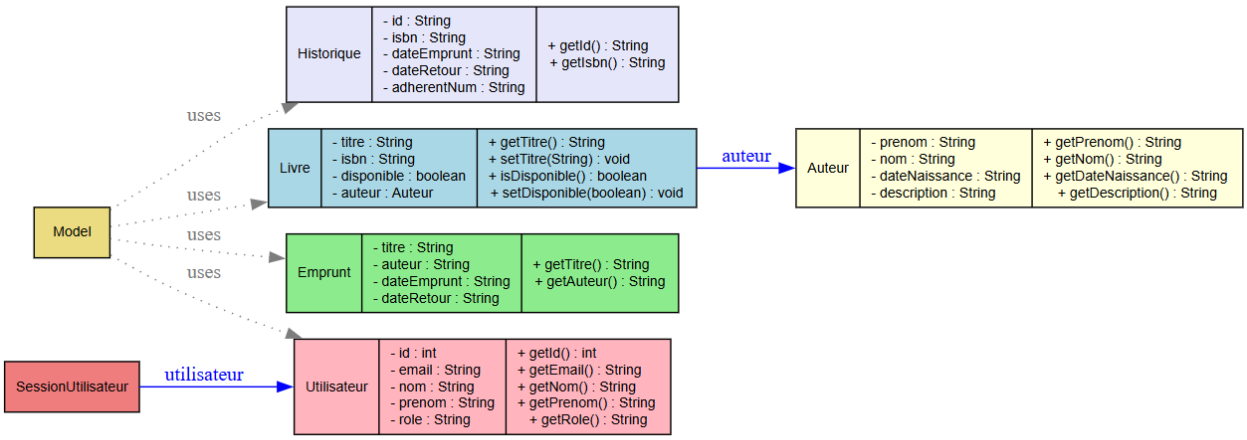
- **Consulter le Catalogue**
  - Acteur : Adhérent, Bibliothécaire
  - Description : L'utilisateur accède à la liste des livres, effectue des recherches ou des filtres (par titre, auteur, disponibilité).

    - Flux principal : Affichage initial → Saisie du terme de recherche → Mise à jour dynamique de la liste.
- **Emprunter un Livre**
    - Acteur : Adhérent
    - Description : Sélection d'un livre dans le catalogue, vérification de sa disponibilité, saisie de l'email de l'adhérent et enregistrement de l'emprunt.
    - Flux alternatif : Message d'erreur en cas de livre déjà emprunté ou email invalide.
- **Retourner un Livre**
    - Acteur : Adhérent
    - Description : Saisie de l'ISBN, vérification de la validité de l'ISBN et mise à jour de l'état du livre dans la base.
    - Flux alternatif : Notification d'erreur si le livre n'est pas reconnu comme emprunté.
- **Modifier son Compte**
    - Acteur : Adhérent
    - Description : Consultation et modification des informations personnelles avec vérification du mot de passe actuel pour autoriser la mise à jour.
    - Flux alternatif : Affichage d'un message d'erreur en cas de non-concordance des mots de passe.
- **Gestion par le Bibliothécaire**
    - Acteur : Bibliothécaire
    - Description : Consultation de la liste des adhérents et accès à l'historique complet des opérations d'emprunt/retour pour assurer le suivi administratif.

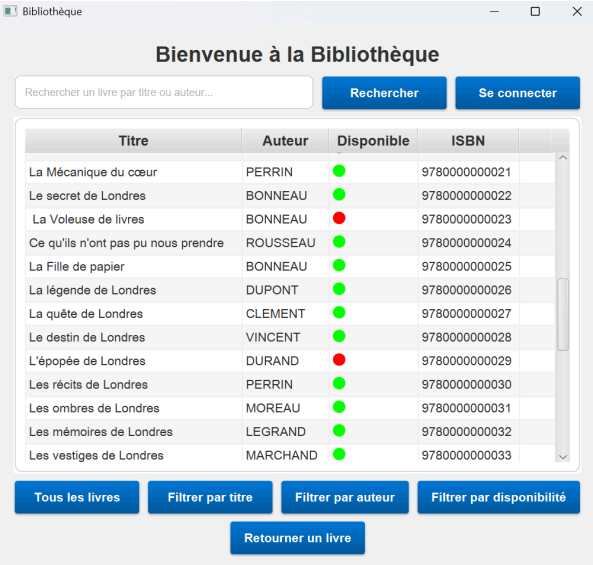
6. Informations Supplémentaires

Pour compléter ce document, il pourrait être utile d'ajouter :

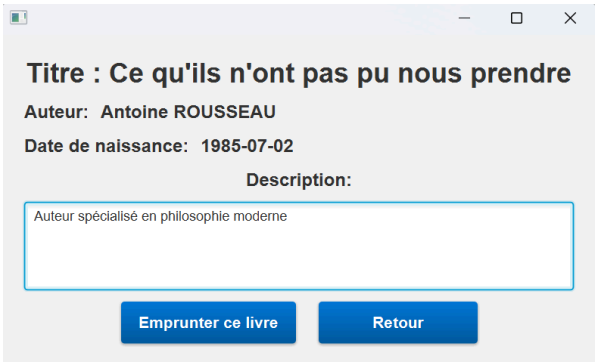
- **Diagrammes UML des classes Model**.



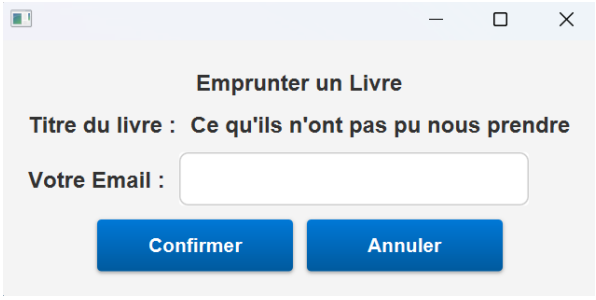
- **Maquettes des Interfaces** : (Accueil, Login, DétailLivre, etc.).



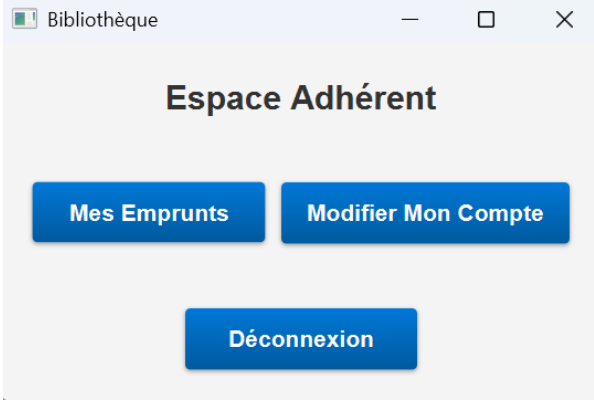
Page d'accueil



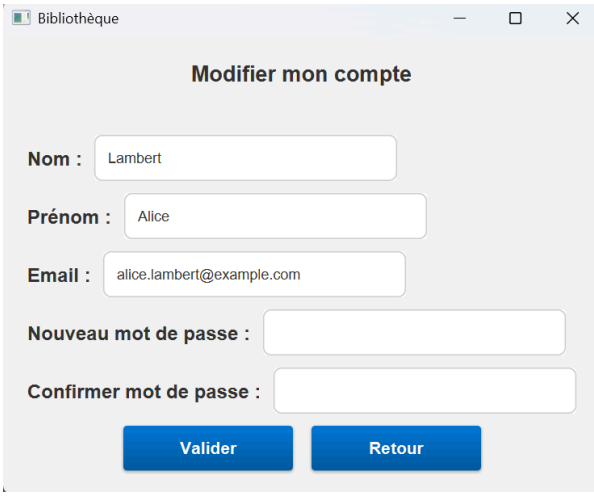
Page de détail d'un livre



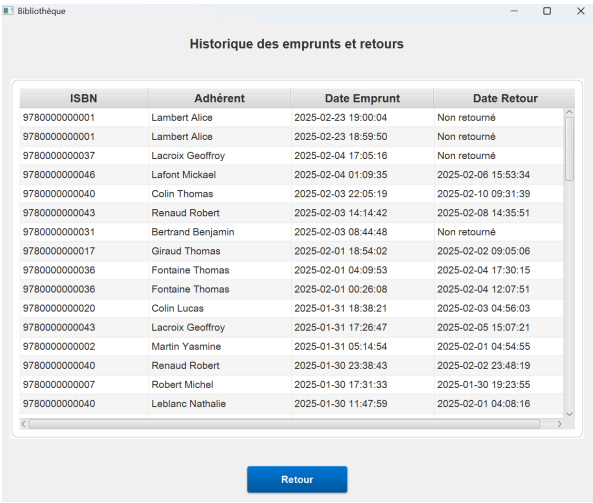
Page d'emprunt



Page adhérent

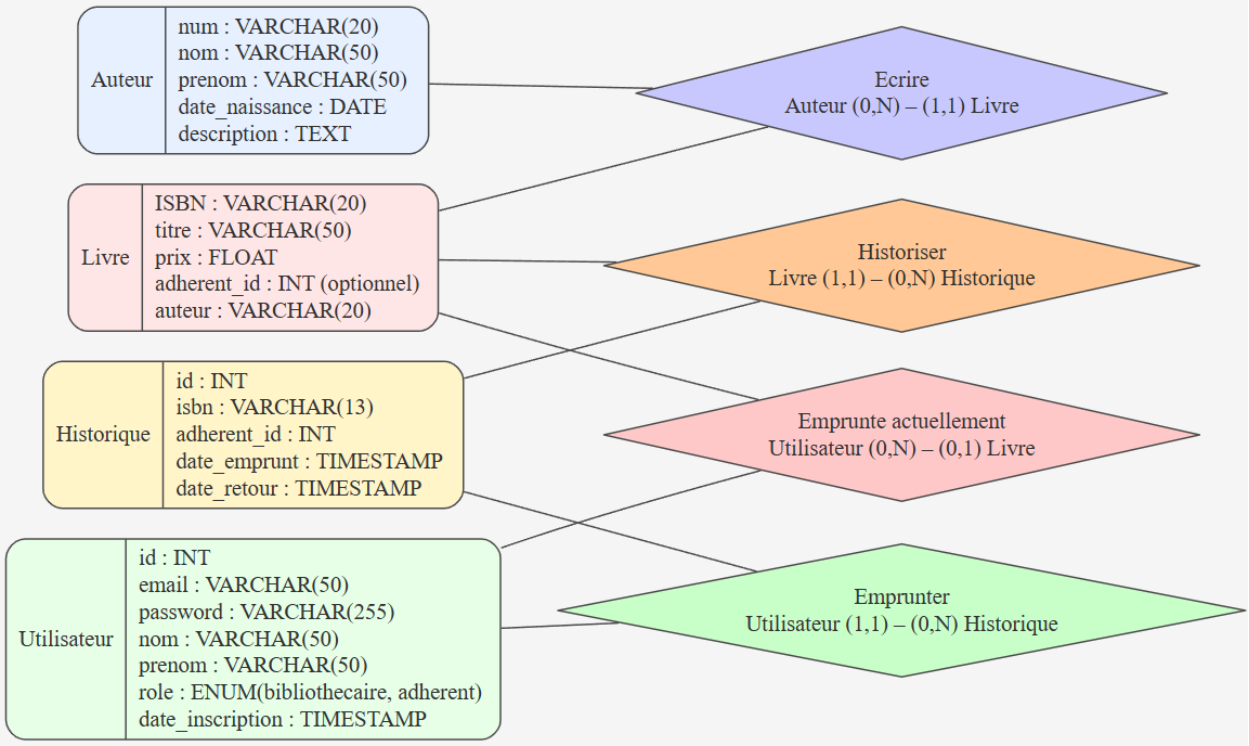


Page de Modification du compte



Page d'historique d'emprunt et de retour

- **Modèle conceptuel des données (MCD)** :



7. Conclusion

L'application met en œuvre des concepts de programmation orientée objet et de développement en Java en utilisant une architecture MVC, facilitant ainsi la séparation des responsabilités entre la logique métier, l'interface utilisateur et l'accès aux données. Elle offre une solution complète pour la gestion des emprunts et retours de livres en bibliothèque, en proposant des fonctionnalités adaptées aux adhérents comme aux bibliothécaires.