

Application de Gestion de Bibliothèque

Realiser par: Assaad Taoussi  
**G4IIR-10**

**Resume du projet (Abstract) :**

Le projet "Application de Gestion de Bibliothèque" vise à simplifier et moderniser la gestion des bibliothèques grâce à une solution logicielle performante. Développée en Java en adoptant le modèle architectural MVC, cette application offre une approche structurée et modulaire pour gérer les opérations essentielles telles que l'ajout, la modification, et la suppression des livres, la gestion des utilisateurs inscrits, ainsi que le suivi des emprunts et des retours.

L'application repose sur des fichiers CSV pour la persistance des données, garantissant une gestion efficace et durable des informations. L'interface utilisateur, conçue avec Swing, se distingue par sa convivialité et son ergonomie, permettant une interaction fluide et intuitive avec les différentes fonctionnalités.

Grâce à cette solution, les bibliothèques peuvent automatiser leurs processus, réduire les erreurs humaines et améliorer significativement leur productivité. Ce projet met également en lumière les bonnes pratiques de développement logiciel, notamment l'utilisation des diagrammes UML pour la conception, les tests rigoureux pour garantir la qualité, et l'organisation claire du code pour une meilleure maintenabilité.

Les résultats obtenus démontrent la capacité de cette application à répondre aux besoins réels des utilisateurs tout en offrant des perspectives d’évolution intéressantes, comme l’intégration d’une base de données relationnelle ou l’ajout de fonctionnalités avancées telles que l’authentification et l’exportation des données.

**Table de Matière :**

[I. Introduction 4](#_Toc19534)

[II. Analyse et Conception 5](#_Toc12255)

[III. Fonctionnalité Implémentée 9](#_Toc756)

[IV. Structure du projet 10](#_Toc13065)

[V. Methodologie de travail 14](#_Toc32443)

[VI. Tests et résultats 16](#_Toc16388)

[VII. Conclusion 18](#_Toc9851)

[VIII. Annexes 19](#_Toc28110)

1. **Introduction**
2. **Contexte du projet**

Dans un monde de plus en plus axé sur la numérisation, les bibliothèques continuent de jouer un rôle essentiel en tant que centres de savoir et de ressources éducatives. Cependant, la gestion traditionnelle des bibliothèques, basée sur des registres papier ou des systèmes informatiques désuets, présente plusieurs défis, tels que des erreurs humaines, une faible efficacité, et des difficultés à retrouver rapidement des informations. Ces contraintes impactent non seulement les gestionnaires, mais également les utilisateurs, qui souhaitent un accès rapide et fiable aux livres et aux services.

Ce projet s'inscrit dans une démarche de modernisation et d’optimisation de la gestion des bibliothèques. Il propose une application logicielle capable d’automatiser les processus clés, tout en assurant une traçabilité efficace des opérations. Grâce à cette solution, les bibliothèques peuvent améliorer leur fonctionnement, offrir un meilleur service aux utilisateurs, et répondre aux attentes d’une société toujours plus connectée.

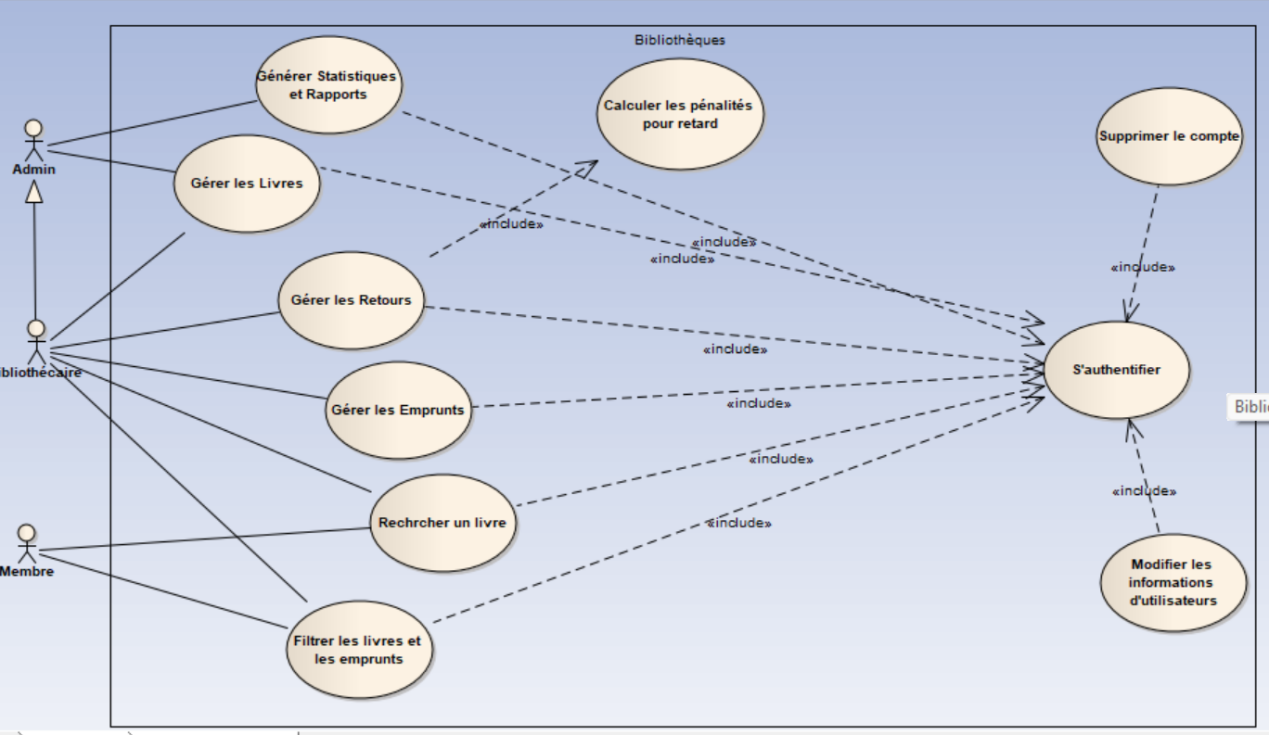
1. **Objectifs**

L’objectif principal de ce projet est de développer une application conviviale et fonctionnelle pour faciliter la gestion des bibliothèques. Plus spécifiquement, les objectifs sont les suivants :

* **Automatisation des tâches** : Réduire la charge de travail liée aux processus manuels tels que la gestion des livres, des utilisateurs, des emprunts, et des retours.
* **Fiabilité et traçabilité des données** : Garantir la persistance et l’intégrité des informations grâce à l’utilisation de fichiers CSV pour stocker les données.
* **Interface utilisateur intuitive** : Fournir une interface graphique claire et simple d’utilisation, développée avec Swing, afin d’assurer une interaction fluide pour les gestionnaires et les utilisateurs.
* **Modularité et extensibilité** : Concevoir une architecture basée sur le modèle MVC, permettant une maintenance facile et la possibilité d’ajouter des fonctionnalités supplémentaires à l’avenir.
* **Formation et apprentissage** : Mettre en pratique les connaissances en programmation Java, en conception UML, et en structuration de projets, tout en acquérant une expérience concrète dans le développement logiciel.

1. **Analyse et Conception**
2. **Diagrammes UML**

* **Diagramme Use Case :**

****

**Acteurs Principaux :**

**Admin** :

* 1. Supervise les activités globales du système.
  2. Génère des statistiques et des rapports.
  3. Supprime les comptes des utilisateurs.

**Bibliothécaire** :

* 1. Gère les livres, les emprunts et les retours.
  2. Filtre et recherche les informations sur les livres et les emprunts.
  3. Peut effectuer des actions similaires à celles de l’admin mais avec des droits plus restreints.

**Membre** :

* 1. Rechercher des livres.
  2. Filtrer les livres et emprunts selon les critères.

**Cas d'utilisation :**

**Générer Statistiques et Rapports** :

* 1. Acteur : Admin.
  2. Description : Permet de produire des rapports concernant l'état des emprunts, les pénalités, et les statistiques globales de la bibliothèque.

**Gérer les Livres** :

* 1. Acteur : Bibliothécaire.
  2. Description : Ajouter, modifier ou supprimer des livres de la bibliothèque.

**Gérer les Retours** :

* 1. Acteur : Bibliothécaire.
  2. Description : Enregistrer les retours de livres et vérifier si des pénalités doivent être appliquées.

**Gérer les Emprunts** :

* 1. Acteur : Bibliothécaire.
  2. Description : Suivre les emprunts de livres effectués par les membres et gérer les extensions éventuelles.

**Rechercher un Livre** :

* 1. Acteurs : Bibliothécaire, Membre.
  2. Description : Trouver des livres en fonction de critères comme le titre, l’auteur ou le genre.

**Filtrer les Livres et les Emprunts** :

* 1. Acteurs : Bibliothécaire, Membre.
  2. Description : Appliquer des filtres pour afficher uniquement les informations pertinentes sur les livres ou les emprunts.

**Calculer les Pénalités pour Retard** :

* 1. Acteurs : Bibliothécaire.
  2. Description : Automatiser le calcul des pénalités en fonction des retards de retour.

**S'authentifier** :

* 1. Acteurs : Admin, Bibliothécaire.
  2. Description : Chaque utilisateur doit s'authentifier pour accéder à ses fonctionnalités spécifiques.

**Modifier les Informations d'Utilisateurs** :

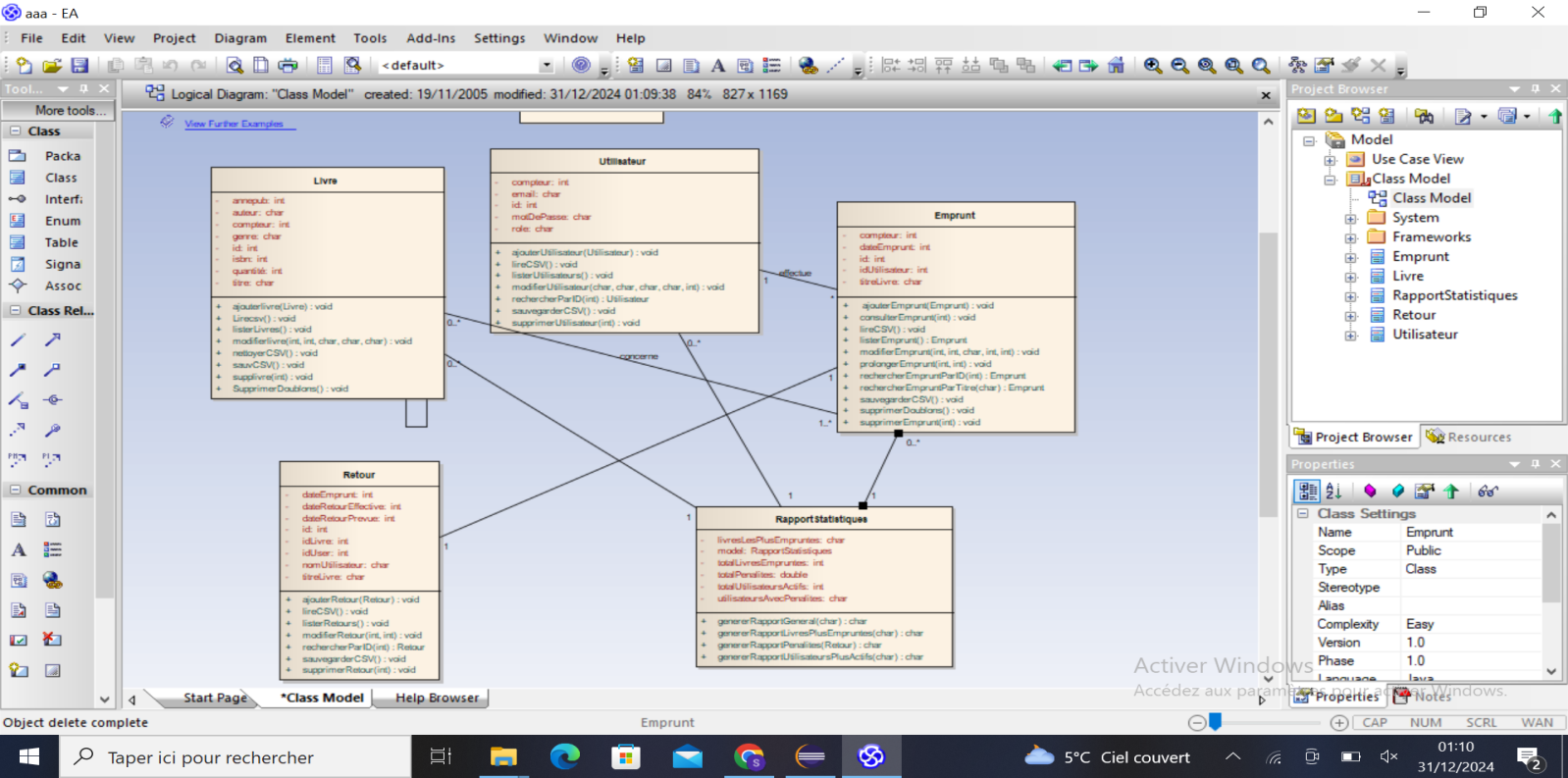
* 1. Acteur : Admin.
  2. Description : Permet à l'admin de mettre à jour les informations des utilisateurs.

**Supprimer le Compte** :

* 1. Acteur : Admin.
  2. Description : Supprimer un compte utilisateur ou membre en cas de nécessité.

**Relations entre les cas d'utilisation :**

* Les cas d'utilisation comme **"Gérer les Emprunts"**, **"Gérer les Retours"**, **"Générer des Statistiques"**, et **"Calculer les Pénalités pour Retard"** incluent des actions liées à **"S'authentifier"**, car elles nécessitent une identification préalable.
* **"Modifier les Informations d'Utilisateurs"** et **"Supprimer le Compte"** sont spécifiques à l'admin, ce qui reflète un niveau d'autorisation plus élevé.
* **Diagramme de classe :**

****

Le diagramme de classes présenté modélise le système de gestion d'une bibliothèque. Il met en évidence les différentes entités et leurs relations dans le cadre du projet. Voici une description des principales classes :

**Classe Livre** :

* 1. Attributs : Représentent les propriétés d'un livre, telles que le titre, l'auteur, le genre, l'année, le stock, et l'état.
  2. Méthodes : Permettent d'ajouter, modifier, supprimer ou rechercher des livres, ainsi que de sauvegarder et de charger les données à partir de fichiers CSV.

**Classe Utilisateur** :

* 1. Attributs : Contiennent des informations sur les utilisateurs, telles que l'identifiant, l'e-mail, le mot de passe, et leur rôle (par exemple, administrateur ou lecteur).
  2. Méthodes : Gèrent les opérations liées aux utilisateurs, comme l'ajout, la modification, la suppression ou la recherche d'un utilisateur.

**Classe Emprunt** :

* 1. Attributs : Incluent des informations sur les emprunts, comme la date d'emprunt, la date de retour, et les identifiants des livres et des utilisateurs associés.
  2. Méthodes : Permettent de gérer les emprunts, notamment leur création, leur modification, leur prolongation ou leur suppression, ainsi que de rechercher des emprunts spécifiques.

**Classe Retour** :

* 1. Attributs : Décrivent les retours de livres avec des informations telles que la date de retour prévue, la date de retour effective, et les identifiants des livres et des utilisateurs concernés.
  2. Méthodes : Gèrent les retours, incluant l'ajout, la modification, et la recherche de retours.

**Classe RapportStatistiques** :

* 1. Attributs : Fournissent des données agrégées pour analyser le système, comme le nombre total d'emprunts, les utilisateurs ayant des pénalités, ou les livres les plus empruntés.
  2. Méthodes : Génèrent des rapports sous divers formats pour les administrateurs, couvrant les statistiques générales, les emprunts, les retours, et les pénalités.

**Relations entre les classes** :

* La classe Livre est associée à Emprunt et Retour, reflétant le fait qu'un livre peut être emprunté ou retourné.
* La classe Utilisateur interagit avec Emprunt et Retour, car les utilisateurs effectuent ces actions.
* La classe RapportStatistiques est utilisée pour analyser les données provenant des emprunts, des retours, et des utilisateurs.

1. **Modele MVC**

Le modèle MVC (Modèle-Vue-Contrôleur) a été adopté dans ce projet pour structurer l'application de gestion de bibliothèque. Ce design pattern permet de séparer les responsabilités en trois composantes distinctes :

**Modèle (Model)** :  
Cette couche gère les données, la logique métier et les règles d'accès. Les classes comme (Livre,LivreModel, Utilisateur,UtilisateurModel, Emprunt,EmpruntModel, Retour et RetourModel) et les interfaces comme (LivreModelInterface,UtilisateurModelInterface,EmpruntModelInterface et RetourModelInterface) font partie de cette composante. Elles encapsulent les données et fournissent des méthodes pour effectuer des opérations telles que l'ajout, la modification ou la suppression d'entités.

**Vue (View)** :  
La vue est responsable de l'affichage des informations à l'utilisateur. Dans ce projet, elle est implémentée avec la bibliothèque Swing pour fournir une interface graphique conviviale. Les classes associées (LivreView, UtilisateurView, EmpruntView et RetourView) gèrent les fenêtres et composants graphiques, tels que les formulaires pour ajouter des livres ou visualiser les emprunts.

**Contrôleur (Controller)** :  
Le contrôleur agit comme un intermédiaire entre le modèle et la vue. Il capture les actions de l'utilisateur via l'interface graphique (par exemple, cliquer sur un bouton) et les traduit en opérations à exécuter sur le modèle. Les classes contrôleurs (comme LivreController ou EmpruntController) contiennent la logique pour orchestrer les interactions entre les différentes couches.

L'utilisation de l'architecture MVC a permis de rendre l'application modulaire, facilitant ainsi son évolution et sa maintenance. Chaque composante est indépendante, ce qui améliore la lisibilité et réduit les interférences entre les différentes parties du code.

1. **Persistance des données**

La persistance des données dans ce projet de gestion de bibliothèque est assurée par l'utilisation de fichiers CSV. Cette approche permet de stocker de manière simple et efficace les informations concernant les livres, les utilisateurs, ainsi que les emprunts et retours. Chaque entité est représentée par un fichier CSV spécifique, garantissant ainsi une séparation claire des données. Par exemple, le fichier "Livre.csv" contient les informations des livres, tandis que "utilisateurs.csv" enregistre les données des utilisateurs. Lorsqu'un utilisateur effectue une opération, telle qu'un emprunt ou un retour, les informations sont mises à jour dans ces fichiers, assurant la sauvegarde des données entre les sessions d'utilisation de l'application. Cette méthode, bien que simple, est suffisante pour un projet de cette envergure et permet de garantir la persistance des données sans recourir à une base de données complexe.

1. **Fonctionnalité Implémentée**
2. **Gestion des livres**

La gestion des livres permet de manipuler le catalogue de la bibliothèque en proposant des fonctionnalités telles que l'ajout de nouveaux ouvrages, la suppression de ceux qui ne sont plus nécessaires, la modification des informations existantes et la recherche rapide d’un livre en fonction de critères spécifiques. Ces options assurent une organisation structurée et un suivi détaillé des livres disponibles.

1. **Gestions des utilisateurs**

Cette fonctionnalité facilite l’administration des utilisateurs de la bibliothèque. Elle permet l'enregistrement de nouveaux profils utilisateurs, la modification de leurs informations personnelles et la suppression des comptes inactifs ou obsolètes. Cela garantit une base de données utilisateur à jour, essentielle pour une gestion fluide des interactions.

1. **Gestion des Emprunts**

L’application prend en charge la gestion des emprunts en offrant la possibilité d’enregistrer de nouvelles transactions d’emprunt, de prolonger les délais si nécessaire et de rechercher des informations sur les emprunts en cours ou passés. Cette fonctionnalité est cruciale pour assurer le suivi des prêts et éviter les litiges.

1. **Gestion de Retours**

La gestion des retours est conçue pour enregistrer efficacement les livres retournés à la bibliothèque tout en calculant automatiquement les pénalités en cas de retard. Cela garantit une transparence dans les transactions et encourage le respect des délais par les utilisateurs.

1. **Interface Graphique**

L'application dispose d'une interface utilisateur conviviale et intuitive, développée avec Swing. Elle offre une navigation simple et des interactions fluides, permettant aux utilisateurs de réaliser facilement toutes les opérations. Des captures d’écran sont incluses pour illustrer l'apparence et le fonctionnement de l'application.

1. **Structure du projet**

Le projet suit l'architecture **Modèle-Vue-Contrôleur (MVC)**, une approche qui sépare les responsabilités en trois couches distinctes afin de garantir la modularité, la maintenabilité et la simplicité du développement. La structure du projet est organisée autour de trois principaux packages : **model**, **view**, et **controller**, chacun ayant des rôles bien définis :

**1. Package** model/ **:**

Ce package regroupe toutes les classes liées aux entités métier et à leur logique métier. Il est structuré de manière à séparer les interfaces des implémentations concrètes, ce qui facilite la flexibilité et l’évolutivité du projet. Voici les classes principales :

* **Livre** : Représente les livres de la bibliothèque avec des attributs comme le titre, l'auteur, le genre, etc.
* **LivreModelInterface** : Déclare les méthodes pour gérer les livres (ajout, suppression, modification).
* **LivreModel** : Implémente les méthodes de l'interface LivreModelInterface, assurant la gestion des livres.
* **Emprunt** : Représente les emprunts effectués par les utilisateurs, incluant des détails sur la durée et les utilisateurs.
* **EmpruntModelInterface** : Définit les méthodes nécessaires à la gestion des emprunts.
* **EmpruntModel** : Implémente les méthodes de gestion des emprunts, comme l'ajout et la prolongation.
* **Utilisateur** : Représente les utilisateurs de la bibliothèque (nom, ID, contact).
* **UtilisateurModelInterface** : Définit les méthodes pour la gestion des utilisateurs.
* **UtilisateurModel** : Implémente les méthodes pour gérer les utilisateurs (création, modification, suppression).
* **Retour** : Gère les informations sur les retours de livres, incluant les pénalités éventuelles.
* **RetourModelInterface** : Définit les méthodes pour gérer les retours et calculer les pénalités.
* **RetourModel** : Implémente les méthodes pour la gestion des retours et le calcul des pénalités.
* **RapportStatistiques** : Représente les rapports et statistiques de la bibliothèque, comme les livres les plus empruntes ou les utilisateurs actifs.
* **RapportStatistiquesModelInterface** : Définit les méthodes pour extraire les données statistiques nécessaires.
* **RapportStatistiquesModel** : Implémente les calculs et l'extraction des données statistiques pour générer des rapports.

Ce découpage garantit une organisation claire entre les données et leur traitement logique, facilitant les évolutions futures du projet.

**2. Package** view/ **:**

Le package **view** regroupe les classes responsables de l'interface utilisateur. Ces classes permettent une interaction intuitive avec les fonctionnalités du système. Voici les principales classes de ce package :

* **LivreView** : Gère l'affichage et les interactions pour la gestion des livres (ajout, suppression, modification, recherche).
* **UtilisateurFrame** : Permet la gestion des utilisateurs, avec des fonctionnalités pour créer, modifier ou supprimer des profils.
* **EmpruntView** : Interface pour gérer les emprunts et afficher les emprunts en cours ou prolongés.
* **RetourView** : Interface dédiée à l'enregistrement des retours et au calcul des pénalités éventuelles.
* **RapportStatistiquesView** : Permet de visualiser les rapports et statistiques de la bibliothèque, comme les livres les plus empruntés ou les pénalités totales générées. Cette vue peut inclure des graphiques et tableaux synthétiques.
* **MainView** : Point d’entrée principal de l’application, regroupant toutes les fonctionnalités sous forme d’onglets ou de menus interactifs.

Ce package garantit une présentation claire et organisée, permettant aux utilisateurs d’interagir avec les données de manière fluide et intuitive.

**3. Package** controller/ **:**

Le package **controller** contient les classes qui orchestrent la communication entre le modèle et la vue. Ces contrôleurs gèrent la logique métier et assurent la synchronisation entre les données et leur affichage. Voici les classes principales :

* **LivreController** : Assure la gestion des actions liées aux livres, en connectant LivreView à LivreModel.
* **UtilisateurController** : Supervise les interactions liées aux utilisateurs entre UtilisateurFrame et UtilisateurModel.
* **EmpruntController** : Coordonne les opérations d'emprunt entre EmpruntView et EmpruntModel.
* **RetourController** : Gère les retours de livres et les pénalités, en connectant RetourView à RetourModel.
* **RapportStatistiquesController** : Contrôle la génération et l’affichage des rapports et statistiques, en connectant RapportStatistiquesView à RapportStatistiquesModel. Ce contrôleur extrait les données nécessaires et les envoie à la vue pour un affichage sous forme de graphiques ou tableaux.

Cette organisation garantit une séparation claire des responsabilités et facilite la maintenance du projet tout en assurant une évolution harmonieuse des fonctionnalités.

**Conclusion**

La structure du projet est conçue pour respecter les principes fondamentaux de l’architecture MVC, assurant une modularité et une séparation claire des responsabilités. L’ajout des classes pour les rapports et statistiques enrichit le système en offrant des fonctionnalités supplémentaires essentielles à la gestion efficace de la bibliothèque. Cette organisation garantit un code maintenable, évolutif et facilement compréhensible pour les développeurs.

1. **Methodologie de travail**

Pour garantir le bon déroulement du projet, une approche méthodique a été adoptée, comprenant les étapes suivantes :

1. Analyse des besoins

Cette étape initiale a consisté à identifier les fonctionnalités requises pour l’application. Cela comprenait une étude approfondie des opérations essentielles telles que la gestion des livres, des utilisateurs, des emprunts et des retours. Les besoins des utilisateurs finaux ont été pris en compte pour concevoir une solution adaptée et intuitive.

1. Conception UML

Une fois les besoins identifiés, des diagrammes UML ont été réalisés pour visualiser la structure et les interactions du système. Ces diagrammes comprenaient :

* **Diagrammes de cas d’utilisation** : Illustrant les principales interactions entre les utilisateurs et le système.
* **Diagrammes de classes** : Définissant la structure des données et leurs relations.

1. Codage

Après la conception, le développement de l'application a été réalisé en Java en suivant l’architecture **MVC** (Modèle-Vue-Contrôleur). Chaque fonctionnalité a été implémentée de manière modulaire pour assurer une organisation claire et un code maintenable.

1. Test et debogage

Une phase de test rigoureuse a été menée pour valider les cas d’utilisation et identifier les éventuelles anomalies. Chaque fonctionnalité a été testée individuellement (tests unitaires) avant de vérifier le fonctionnement global de l’application (tests d’intégration). Les problèmes détectés ont été corrigés pour garantir la fiabilité et la stabilité du système.

Cette méthodologie a permis de structurer efficacement le travail et d'assurer une progression fluide du projet, depuis la conception jusqu'à la réalisation finale.

1. **Tests et résultats**
2. **Cas de Test**

Pour valider les fonctionnalités de l’application, plusieurs cas de test ont été définis. Ces tests couvrent les principaux scénarios d’utilisation et visent à détecter les anomalies potentielles. Voici une description des cas de test :

| **Fonctionnalité** | **Entrée** | **Résultat Attendu** |
| --- | --- | --- |
| Ajouter un livre | Titre : "Java Avancé", Auteur : "J. Doe", ISBN : "12345" | Le livre est ajouté à la liste. |
| Modifier un utilisateur | ID : 1, Nouveau Nom : "Alice Smith" | Les informations de l'utilisateur sont mises à jour. |
| Enregistrer un emprunt | ID Utilisateur : 2, ID Livre : 101 | L'emprunt est enregistré avec succès. |
| Enregistrer un retour | ID Emprunt : 5, Date : "2024-12-01" | Le retour est enregistré et les pénalités calculées. |
| Rechercher un livre | Titre : "Java" | Une liste des livres correspondants est affichée. |

1. **Resultats Observes**

Après l'exécution des tests, les résultats observés ont été consignés. La table suivante compare les résultats attendus et observés :

| **Fonctionnalité** | **Entrée** | **Résultat Attendu** | **Résultat Observé** | **Statut** |
| --- | --- | --- | --- | --- |
| Ajouter un livre | Titre : "Java: A Beginner's Guide", Auteur : "Herbert Schildt", ISBN : "1" | Livre ajouté avec succès | Livre ajouté avec succès | ✅ OK |
| Modifier un utilisateur | ID : 1, Nouveau Nom : "Sara Tahiri" | Informations mises à jour | Informations mises à jour | ✅ OK |
| Enregistrer un emprunt | ID Utilisateur : 2, Titre Livre : Clean Code | Emprunt enregistré | Emprunt enregistré | ✅ OK |
| Enregistrer un retour | ID Emprunt : 1, Date : "2024-03-09" | Retour enregistré et pénalité calculée | Retour enregistré et pénalité correcte | ✅ OK |
| Rechercher un livre | Titre : "Java" | Liste des livres correspondants affichée | Liste des livres correspondants affichée | ✅ OK |

**Conclusion du Test :**

Les résultats observés ont confirmé que toutes les fonctionnalités principales fonctionnent conformément aux attentes. Aucune anomalie critique n'a été détectée, et l'application est opérationnelle dans les scénarios testés. Les tests ont également démontré que l’interface utilisateur est réactive et facile à utiliser.

1. **Conclusion**

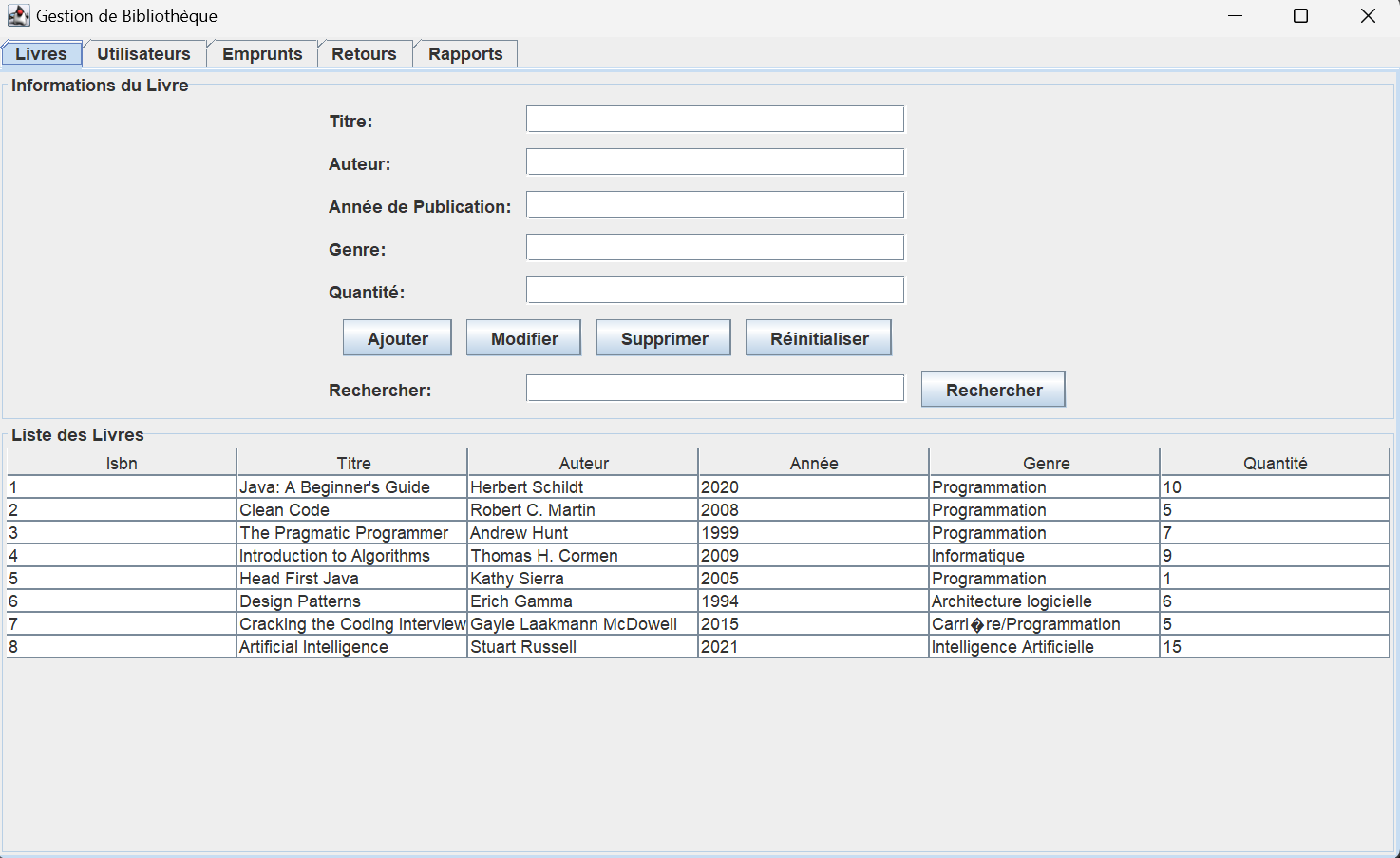
Le projet de gestion de bibliothèque a permis de concevoir et de développer une application fonctionnelle répondant aux besoins essentiels de la gestion des livres, des utilisateurs, des emprunts et des retours. À travers une méthodologie structurée, allant de l'analyse des besoins à la phase de tests, chaque fonctionnalité a été implémentée de manière modulaire et maintenable. L'architecture MVC a été soigneusement adoptée pour assurer une séparation claire entre la logique métier, la présentation et l’interaction avec l’utilisateur, ce qui permet une évolutivité et une gestion simplifiée du code.

Les tests réalisés ont validé les fonctionnalités principales de l’application, confirmant son bon fonctionnement et sa stabilité. Chaque fonctionnalité, qu'il s'agisse de l'ajout de livres, de la gestion des emprunts, ou du calcul des pénalités pour les retours en retard, a été implémentée avec succès, offrant ainsi une solution complète pour la gestion d'une bibliothèque.

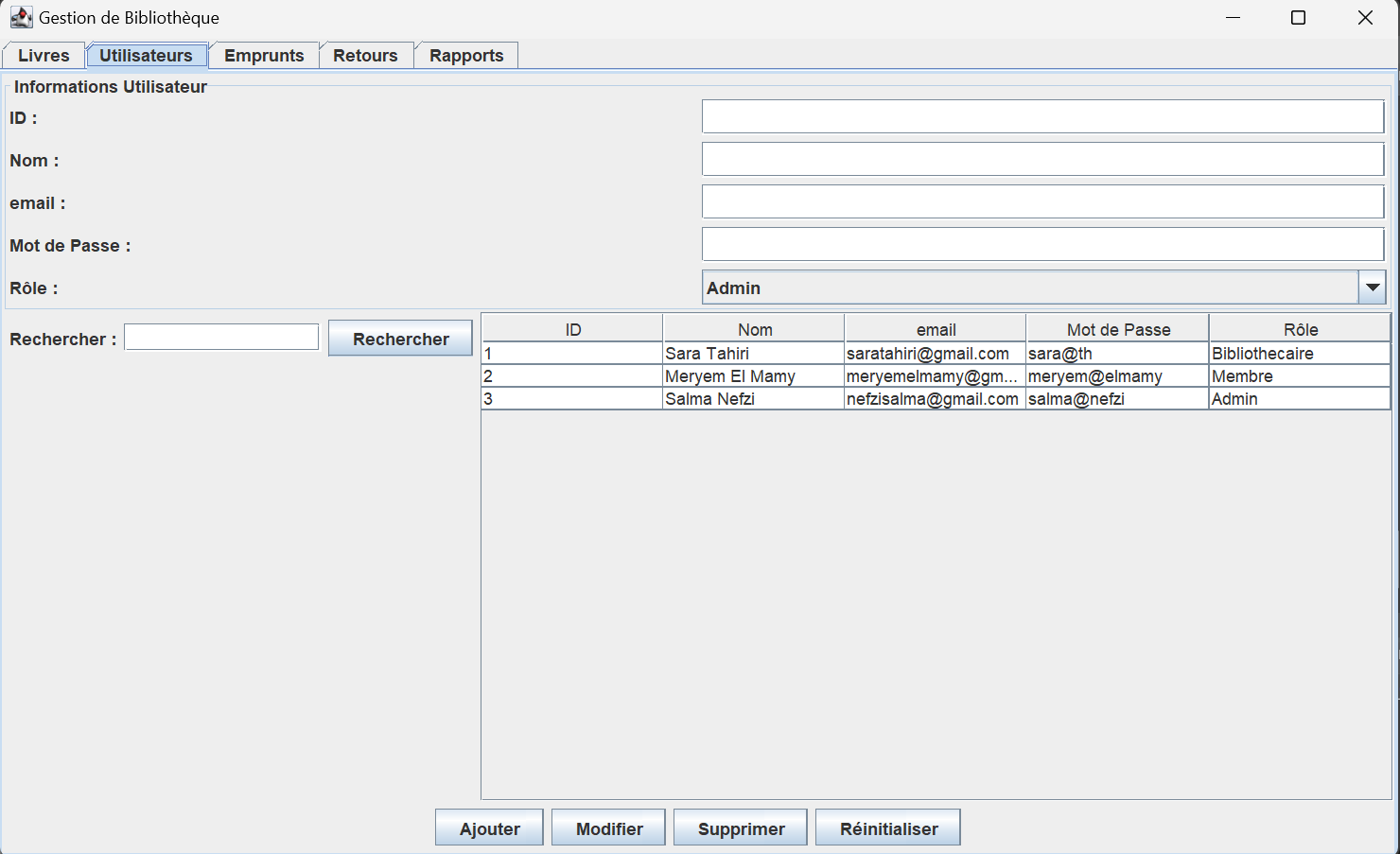
En somme, ce projet a permis de mettre en pratique les compétences acquises, notamment en Java et en gestion de projet, tout en offrant une expérience d’apprentissage enrichissante. De plus, il ouvre la voie à de futures améliorations, telles que l'intégration d'une base de données, l’ajout de nouvelles fonctionnalités comme l'authentification des utilisateurs, et l'exportation des données. Cette application constitue ainsi une solution efficace et évolutive pour la gestion des bibliothèques.

1. **Annexes**

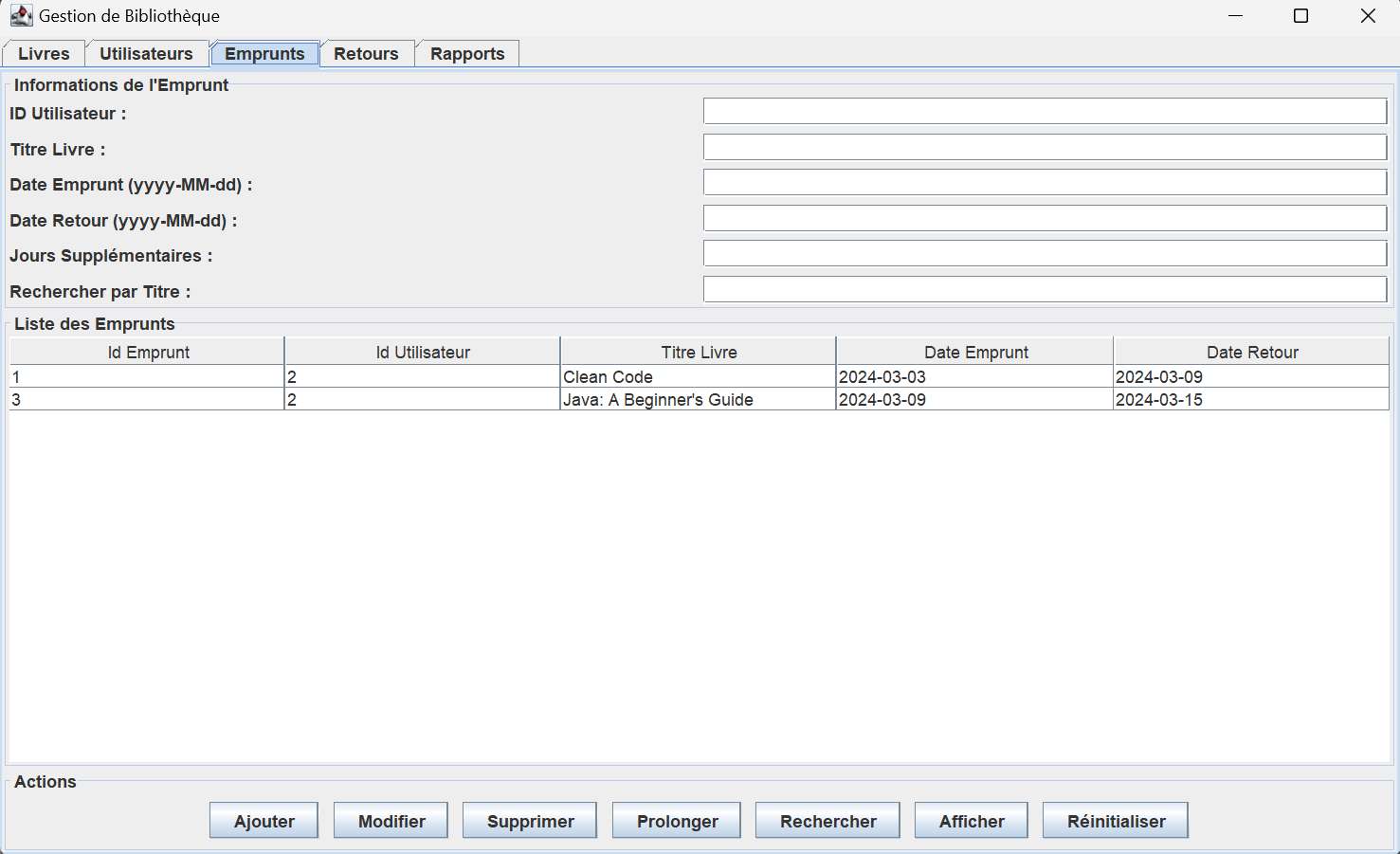
**Partie Livres dans l’interface**

****

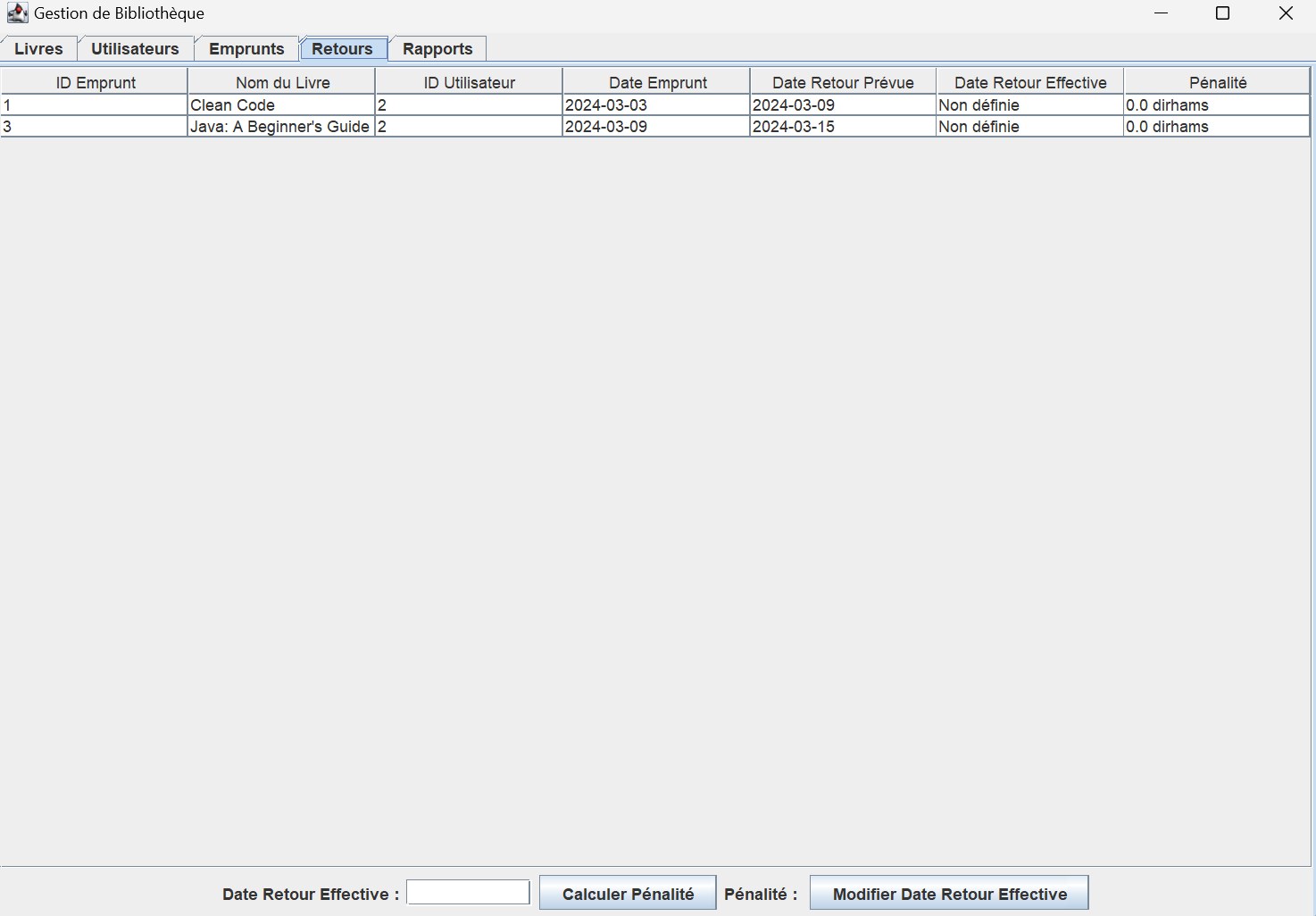
**Partie Utilisateurs dans Interface**

****

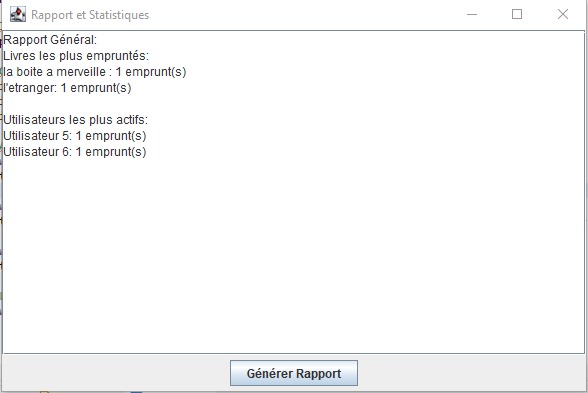
**Partie Emprunts dans l’interface**

****

**Partie Retours dans l’interface**

****

**Partie Rapport et Statistiques**

****