**Université Ouaga 1 Pr Joseph Ki ZERBO**

……………………………………

**Institut Burkinabè des Arts et Métiers**

……………………………

Système de Gestion de Bibliothèque

Fait par : **KONFE Adama**

**SAWADOGO Abdel Kaled**

**SAWADOGO Nathanael**

Enseignant : **Dr OUATTARA Dimitri**

Table des matières

[1. Introduction 3](#_Toc200782292)

[2. Présentation générale 3](#_Toc200782293)

[3. Méthodologie de développement 3](#_Toc200782294)

[4. Description technique 4](#_Toc200782295)

[5. Fonctionnalités principales 5](#_Toc200782296)

[6. Procédure d'installation et d'utilisation 5](#_Toc200782297)

[7. Description de la base de données 6](#_Toc200782298)

[8. Architecture détaillée 6](#_Toc200782299)

[9. Tests et validation 7](#_Toc200782300)

[10. Points forts 7](#_Toc200782301)

[11. Limites et perspectives d'amélioration 8](#_Toc200782302)

[12. Sécurité et bonnes pratiques 8](#_Toc200782303)

[13. Conclusion 8](#_Toc200782304)

# 1. Introduction

Ce projet consiste en la réalisation d'une application de gestion de bibliothèque permettant de gérer les livres, les adhérents et les emprunts. L'application est développée en Java avec une interface graphique Swing et utilise une base de données MySQL/MariaDB pour le stockage des données. Elle suit l'architecture MVC (Modèle-Vue-Contrôleur) pour une meilleure organisation et maintenabilité du code.

# 2. Présentation générale

* **Nom du projet** : GestionBibliotheque
* **Langage principale** : Java
* **Bibliothèque graphique** : Swing
* **Base de données** : MySQL ou MariaDB
* **Outil de gestion de dépendances** : Maven
* **Architecture** : MVC (Modèle-Vue-Contrôleur)

# 3. Méthodologie de développement

Le développement s’est basé sur une approche incrémentale avec plusieurs itérations. Chaque fonctionnalité a été développée, testée puis intégrée selon les principes de l’architecture MVC. Cette méthode a permis de garantir une évolution progressive et fiable du projet.

**GestionBibliotheque/**

**├── pom.xml**

**├── src/**

**│ ├── main/java/com/example/bibliotheque/**

**│ │ ├── Main.java**

**│ │ ├── DBConnection.java**

**│ │ ├── model/ (Livre, Adherent, Emprunt + DAO)**

**│ │ ├── controller/ (Controllers MVC)**

**│ │ └── view/ (Interfaces Swing)**

**│ └── test/**

**└── target/**

# 4. Description technique

## a. Modèle

Les classes **Livres, Adherent, Emprunt** représentent les entités du domaine. Chaque entité possède un **DAO** (Data Access Objet) dédié pour gérer les opérations CRUD (ex : LivreDAO, AdherentDAO, EmpruntDAO).

## b. Vue

L’interface est construite en **Swing**, avec un panneau par entité (LivrePanel, AdherentPanel, etc.) et une fenêtre de connexion (LoginDialog).

## c. Contrôleur

Chaque entité dispose d’un contrôleur (LivreController, etc.) jouant le rôle d’intermédiaire entre les actions de l'utilisateur et la logique du modèle.

## d. Connexion à la base de données

La classe DBConnection gère l'accès à la base via JDBC :

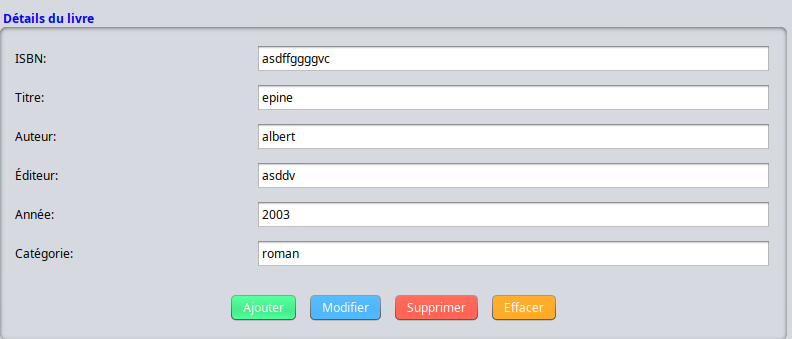
**private static final String URL = "jdbc:mysql://127.0.0.1:3306/bibliotheque";**

**private static final String USER = "root";**

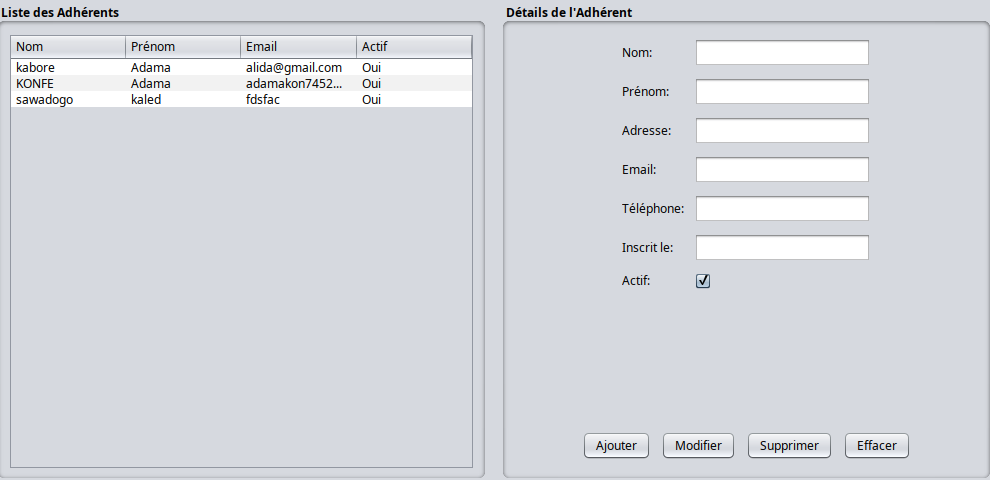
**private static final String PASSWORD = "";**

À adapter selon votre configuration locale.

# **5. Fonctionnalités principales**

* **Gestion des livres** : ajout, modification, suppression, recherche, affichage de la liste.
* **Gestion des adhérents** : ajout, modification, suppression, recherche, affichage.
* **Gestion des emprunts** : enregistrer un emprunt, rendre un livre, voir l'historique, recherche.
* **Authentification** : fenêtre de login (actuellement avec identifiants codés en dur).
* **Interface graphique** : application de bureau moderne avec onglets pour chaque entité.
* **Connexion à la base de données** : via JDBC, paramètres dans DBConnection.java.









# 6. Procédure d'installation et d'utilisation

### Prérequis

- Java 8 ou supérieur

- MariaDB ou MySQL installé et en service

- Maven

### Installation

1. Cloner le projet ou copier les fichiers sources.

2. Importer le script SQL fourni dans MariaDB/MySQL :

Bash : mariadb -u root -p < bibliotheque.sql

3. Adapter les paramètres de connexion dans `DBConnection.java` si besoin.

4. Compiler le projet :

Bash : mvn clean package

5. Lancer l'application :

Bash : java -jar target/GestionBibliotheque-1.0-SNAPSHOT-jar-with-dependencies.jar

### Utilisation

- Se connecter avec l'identifiant par défaut (admin/admin).

- Naviguer entre les onglets pour gérer livres, adhérents, emprunts.



# 7. Description de la base de données

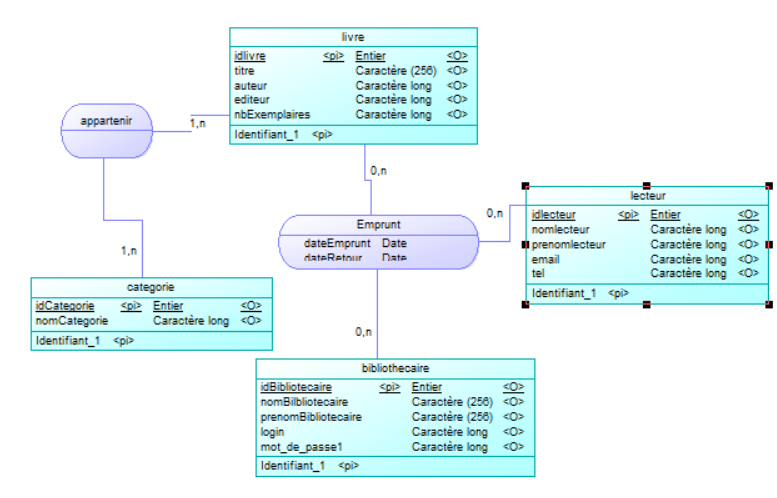
- Tables : livre, adherent, emprunt

- Relations :

- Un emprunt relie un livre et un adhérent.

- Contraintes d'intégrité référentielle (clé étrangère).

- Triggers : pour mettre à jour la disponibilité des livres lors des emprunts/rendus.

- Script SQL fourni : à importer pour initialiser la base et insérer des exemples de données.

*Figure : Modèle Conceptuel de Donnés*

# 8. Architecture détaillée

a. Exemple de contrôleur : LivreController

- Gère l'ajout, la modification, la suppression, la recherche et l'affichage des livres.

- Utilise le modèle `LivreDAO` et la vue `LivrePanel`.

- Exemple de méthode :

private void ajouterLivre() {

String isbn = view.isbnField.getText();

Livre livre = new Livre(isbn, ...);

if (model.ajouterLivre(livre)) {

JOptionPane.showMessageDialog(view, "Livre ajouté avec succès !");

loadLivres();

}

}

b. Exemple de vue : EmpruntPanel

- Affiche la liste des emprunts, un formulaire pour ajouter/rendre/supprimer un emprunt, et des outils de recherche.

- Utilise des composants Swing avancés (JTable, JComboBox, JDatePicker).

c. Exemple de DAO : AdherentDAO

- Fournit des méthodes pour ajouter, modifier, supprimer, lister et rechercher des adhérents.

- Utilise la classe `DBConnection` pour accéder à la base de données.

# 9. Tests et validation

### Tests manuels :

- Ajout, modification, suppression de livres, adhérents, emprunts

- Recherche et filtrage

- Vérification de la mise à jour de la disponibilité des livres

-Tests automatiques :

- (À développer dans src/test/java)

-Validation :

- Vérification de la cohérence des données après chaque opération

- Contrôle des messages d'erreur et de la robustesse de l'interface

# 10. Points forts

- Architecture claire (MVC)

- Séparation des responsabilités (modèle, vue, contrôleur)

- Interface utilisateur ergonomique (onglets, formulaires, recherche)

- Gestion des erreurs (messages d'erreur, vérification des champs)

- Connexion à une vraie base de données (MySQL/MariaDB)

- Utilisation de DAO pour l'accès aux données (meilleure maintenabilité)

# 11. Limites et perspectives d'amélioration

- Authentification : actuellement codée en dur, à remplacer par une vérification en base.

- Sécurité : mots de passe en clair dans le code (**DBConnection.java**), à sécuriser.

- Tests unitaires : peu ou pas de tests présents, à développer pour fiabiliser le code.

- Gestion des exceptions : améliorer la robustesse (messages plus clairs, logs).

- Internationalisation : l'interface est en français, mais non prévue pour plusieurs langues.

- Gestion des droits utilisateurs : pas de distinction admin/utilisateur simple.

- Interface web : possibilité d'évolution vers une version web (Spring Boot, Angular, etc.).

# 12. Sécurité et bonnes pratiques

- Ne jamais laisser de mots de passe en clair dans le code source\*\*

- Utiliser des utilisateurs SQL avec des droits limités\*\*

- Prévoir une gestion des sessions et des droits utilisateurs\*\*

- Sauvegarder régulièrement la base de données\*\*

- Tenir à jour les dépendances Maven\*\*

# 13. Conclusion

Ce projet a permis de mettre en œuvre une application de gestion efficace, robuste et maintenable. L’architecture MVC et l’utilisation de DAO ont assuré une bonne organisation du code. Ce projet peut facilement évoluer vers une version plus avancée, intégrant la sécurité, des rôles utilisateurs et une interface web moderne.

Les tests réalisés montrent la fiabilité de l'application pour la gestion courante d'une bibliothèque. Des améliorations sont possibles, notamment sur la sécurité, l'expérience utilisateur et l'ouverture vers d'autres technologies.