PROJET BIBLIOTHEQUE PAR KEVIN

Epreuve E5 BTS SIO

Création d'une application pour la bibliothèque

Cahier des charges

Document rédigée par Kevin Ye le 04/12/2023

Dernière mise à jour le 29-03-2024

PROJET AP2 JAVA

PROJET BIBLIOTHEQUE PAR KEVIN

Sommaire

PROJET AP2 JAVA

PROJET BIBLIOTHEQUE PAR KEVIN

Mise en situation

La bibliothèque municipale permet à ses adhérents d'emprunter des livres.
Chaque adhérent peut emprunter 5 livres maximum.
Les livres de la bibliothèque sont classés par auteur.

Toute la gestion de la bibliothèque est manuelle.

Le responsable voudrait acquérir une borne automatique pour alléger le travail des bibliothécaires qui devra permettre de retirer ou déposer un livre.

La borne est livrée sans logiciel adéquat, en tant que développeur, on vous demande de proposer une solution temporaire pour faire une première évaluation de cette solution innovante.

PROJET AP2 JAVA

1. Rédaction du cahier des charges

1.1. Contexte

Une bibliothèque municipale souhaite avoir un programme permettant la gestion de sa bibliothèque, les adhérents pourront emprunter des livres et rendre le travail des bibliothécaires plus léger lors de l'emprunt et du retour des livres.
Ce projet devra donc être rendu au plus tard le 22 avril 2024.

1.2. Besoins fonctionnels

Le programme sera accessible que par les bibliothécaires en interagissant avec une base de données pour la gestion des livres de la bibliothèque.

1.3. Ressources matérielles

Afin de pouvoir réaliser le projet, on doit pouvoir disposer de plusieurs ressources matérielles qui doivent être mises à disposition. Nous avons besoins des matériaux suivantes :

- Ordinateurs
- Souris
- Claviers
- Écran (PC portable)

1.4. Technologies utilisées

Les logiciels dont nous aurons besoins sont les suivantes :

- IDE : Visual Studio Code
- MAMP : MySQL (Base de données)

- Serveur APACHE (Utilisation du PHP)
- GitHub (plateforme de développement)
- Trello/Gira (Gestion projet)
- Mocodo (MCD)

Les sources documentaires qui aideront por la mission (noter que ceux utiliser) :

- Documentation PHP (site officiel)
- Open Classrooms
- Stack Overflow (forum en EN)
- Developpez.net etc.
- Youtube
- https://dev.mysql.com/downloads/connector/j/5.1.html (liens VScode java → BDD MySQL)
- https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html (Lien pour JFrame)

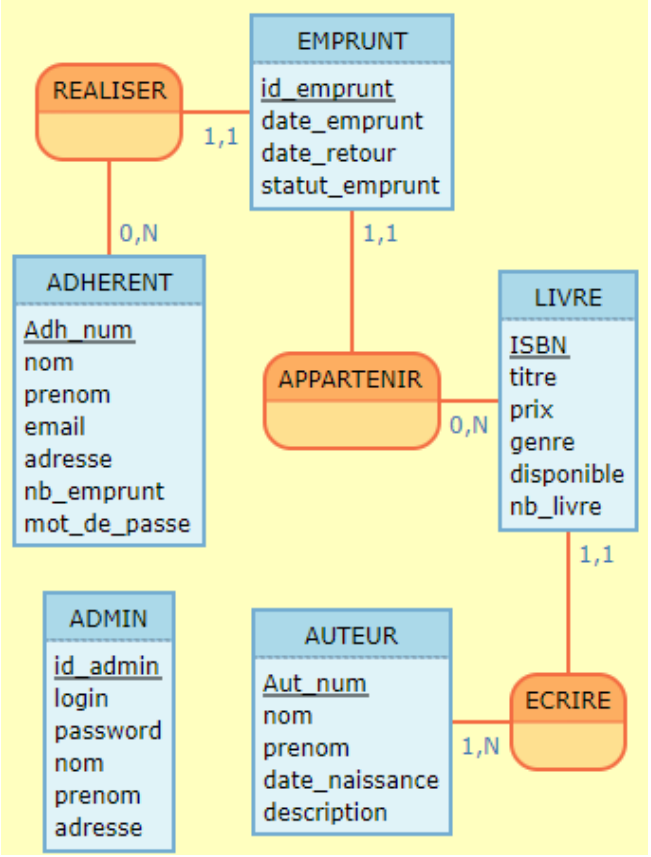
2. Conception de base de données

2.1. Dictionnaire des données

Entité ou association	Libellé de l'attribut	Type de l'attribut
Adherent	Adh_num	int(4)
	nom	varchar(255)
	prenom	varchar(255)
	email	varchar(255)
	adresse	varchar(255)
	nb_emprunt	int(4)
	mot_de_passe	varchar(100)
Admin	id_admin	int(4)
	login	varchar(255)
	password	varchar(100)
	nom	varchar(255)
	prenom	varchar(255)
	adresse	varchar(255)
Auteur	Aut_num	int(4)
	nom	varchar(255)
	prenom	varchar(255)
	date_naissance	date
	description	varchar(42)
Emprunt	id_emprunt	int(4)
	date_emprunt	date
	date_retour	date
	statut_emprunt	tinyint(1)
	Adh_num	int(4)
	ISBN	varchar(20)
Livre	ISBN	varchar(20)
	titre	varchar(255)
	prix	float
	genre	varchar(255)
	disponible	tinyint(1)
	nb_livre	int(4)
	Aut_num	int(4)

2.2. MCD (modèle de conceptuel de données)

Pour mieux construire la base de données nous allons la conceptualiser pour permettre d'avoir une bonne base de données. Nous utiliserons Mocodo afin de mieux visualiser la base de données.



Code dans la console Mocodo

REALISER, 0N ADHERENT, 11 EMPRUNT
EMPRUNT: id_emprunt, date_emprunt, date_retour, statut_emprunt
:

ADHERENT: Adh_num, nom, prenom, email, adresse, nb_emprunt, mot_de_passe
APPARTENIR, 11 EMPRUNT, 0N LIVRE
LIVRE: ISBN, titre, prix, genre, disponible, nb_livre

ADMIN: id_admin, login, password, nom, prenom, adresse
AUTEUR: Aut_num, nom, prenom, date_naissance, description
ECRIRE, 11 LIVRE, 1N AUTEUR

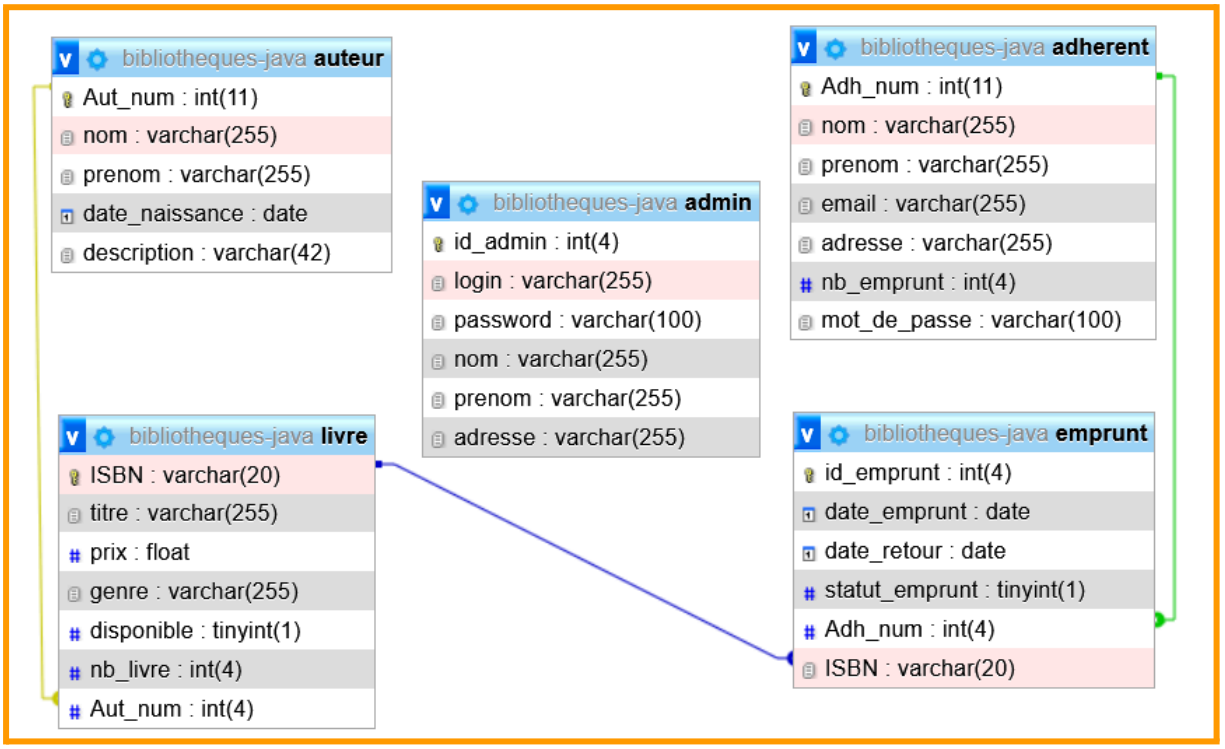
2.3. MLD (modèle de logique de données)

Pour mieux construire la base de données nous allons la conceptualiser nous utiliserons le MLD est le suivant :

- **ADHERENT** (Adh_num, nom, prenom, email, adresse, nb_emprunt, mot_de_passe)
- **ADMIN** (id_admin, login, password, nom, prenom, adresse)
- **AUTEUR** (Aut_num, nom, prenom, date_naissance, description)
- **EMPRUNT** (id_emprunt, date_emprunt, date_retour, statut_emprunt, *#Adh_num*, *#ISBN*)
- **LIVRE** (ISBN, titre, prix, genre, disponible, nb_livre, *#Aut_num*)

2.4. MRD (modèle de relationnel de données) ou MPD (modèle de physique de données)

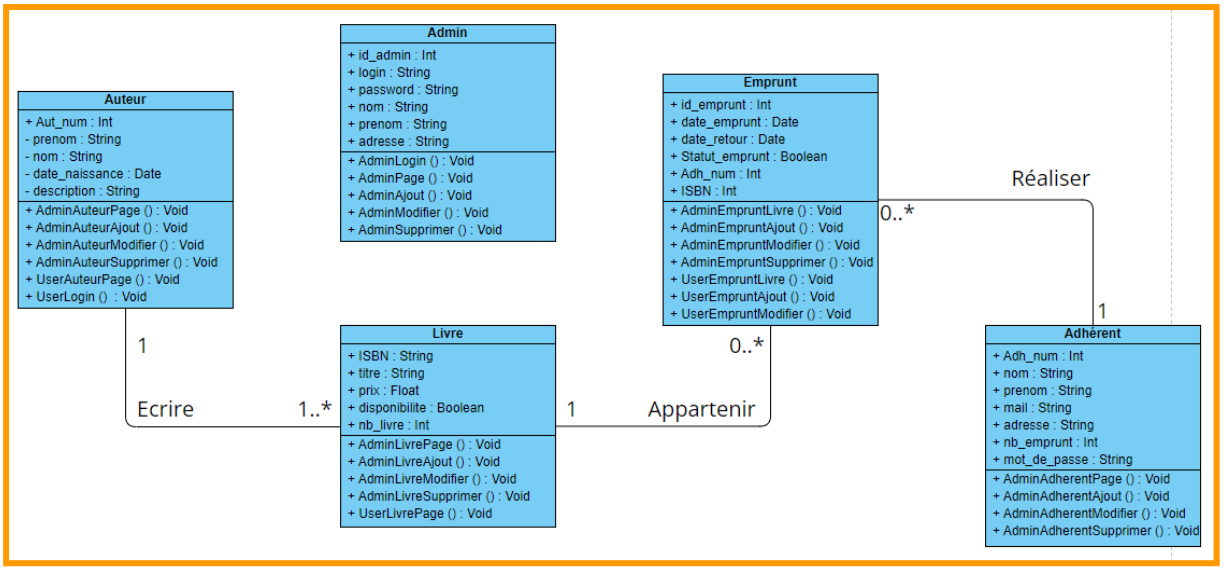
Pour mieux construire la base de données nous allons la conceptualiser nous utiliserons le MPD est le suivant :



3. Conception du fonctionnement du site

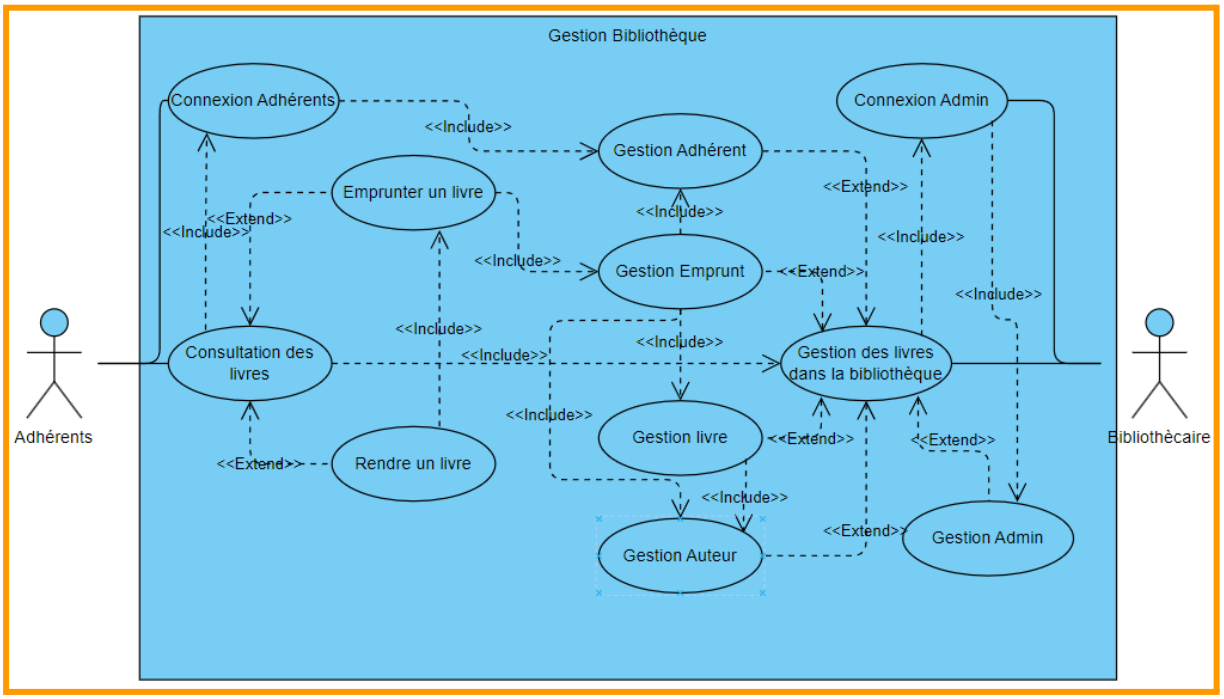
3.1. Diagramme de classes

Pour pouvoir mieux visualiser les différentes fonctions à faire sur Java, nous avons le diagramme de classes suivant :



3.2. Diagramme de cas d'utilisation

Pour mieux définir le contexte dans laquelle le logiciel sera utiliser pour répondre aux attentes :



4. Gestion du projet

4.1. Listes des règles de gestion

Les règles de gestion par rapport à ce qu'on veut faire (nombre de livre dans la bdd, livre limiter 5 par personne, le livre doit être rendu avant une certaines date, l'adhérent doit être ajouter dans la base de données avant d'emprunter)

A ne pas oublier lors de la saisie des données les règles suivantes :

- Pour la table Adhèrent pas de restriction
- Pour la table Auteur pas de restriction
- Pour la table Livre, nous devons absolument créer la table Auteur pour pouvoir remplir tous les champs
- Pour la table Emprunt, toutes les tables doivent être déjà créées avant l'emprunt car nous avons besoin de l'adhérent, le livre (pas d'auteur = pas de livre) et l'auteur pour pouvoir remplir tous les champs.

4.2. Diagramme de Gantt (gestion de projet)

Diagramme de Gantt du projet E5 Java sur les bibliothèques.

cf(Annexe)

GANTT CHART | Projet E5 Java

PROJECT TITRE	Projet AP2 Bibliothèque	COMPANY NAME	Kilox's Dev
PROJECT MANAGER	Kevin	DATE	29/01/2024

NUMÉRO	TITRE DE LA TÂCHE	PROPRIETAIRE DE LA TACHE	DATE DE DÉBUT	DATE LIMITE	DURÉE	TÂCHE TERMINÉE (EN %)
1	**Cahier des charges**				14	
1.1	Rédaction du cahier des charges	Développeur	22/01/24	02/02/24	14	75 %
1.2	Conception de base de données	Développeur	24/1/24	29/01/24	4	100%
1.3	Gestion de projet	Développeur	26/1/24	31/01/24	4	75%
1.4	Ajout des éléments de l'annexe	Développeur	30/1/24	2/2/24	4	50 %
2	**Programmation du projet**				19	
2.1	Mise en place des logins	Développeur	05/02/24	09/02/24	5	0 %
2.2	Mise en place de la partie de l'administrateur	Développeur	12/2/24	21/02/24	8	0 %
2.3	Mise en place de la partie adhérent	Développeur	22/2/24	29/02/24	6	0 %
3	**Rendu du projet**				23	
3.1	Test du code + correction	Développeur	1/3/24	13/03/24	9	0 %
3.2	Révision de code	Développeur	1/3/24	21/03/24	15	0 %
3.3	Dernière vérification et finalisation	Développeur	12/3/24	22/03/24	9	0 %

4.2.1. Sécurité des services

Nous allons mettre en place un code pour éviter l'injection SQL. Ajouter un login pour sécuriser l'accès.
L'algorithme de hachage SHA-256 pour produire un haché sécurisé du mot de passe afin de pouvoir vérifier l'identité de l'utilisateur.

```
159     // Méthode pour hacher le mot de passe
160     public static String hashPassword(String password) {
161         try {
162             MessageDigest md = MessageDigest.getInstance(algorithm:"SHA-256");
163             byte[] messageDigest = md.digest(password.getBytes());
164             BigInteger no = new BigInteger(signum:1, messageDigest);
165             String hashText = no.toString(radix:16);
166             while (hashText.length() < 32) {
167                 hashText = "0" + hashText;
168             }
169             return hashText;
170         } catch (NoSuchAlgorithmException e) {
171             throw new RuntimeException(e);
172         }
173     }
174
```

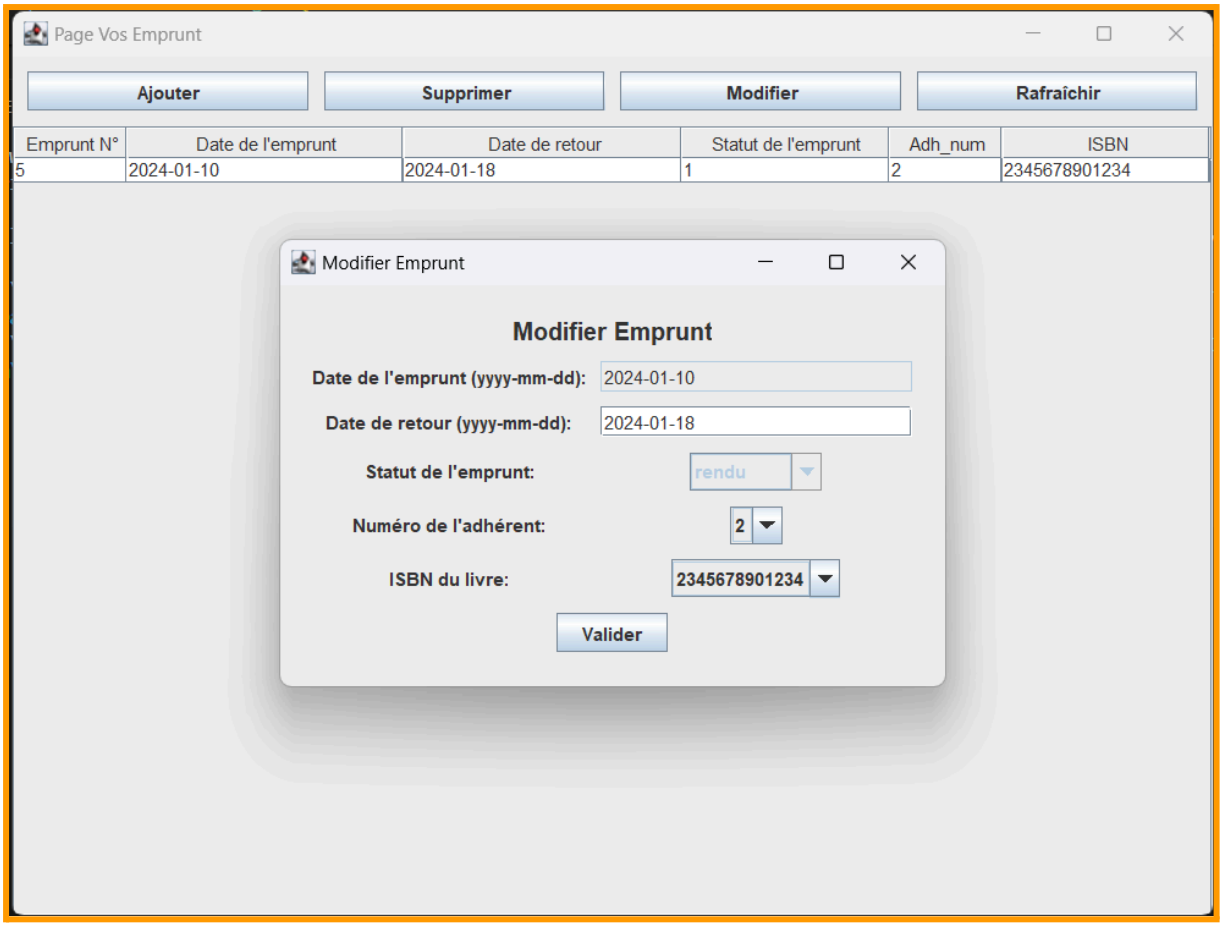
Cette méthode permet de hacher le mot de passe, on l'utilisera pour insérer le mot de passe haché dans la base de données. Pour se connecter on utilisera le hachage pour pouvoir comparer le hachage du mot de passe saisie à celui de la base de données qui est déjà haché.

```java
private void seConnecter() {
    String username = usernameField.getText();
    String adherentId = username;
    char[] passwordChars = passwordField.getPassword();
    String password = new String(passwordChars);

    try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {
        String sql = "SELECT * FROM adherent WHERE Adh_num = ?";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(parameterIndex:1, username);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    String storedPasswordHash = rs.getString(columnLabel:"mot_de_passe"); // Récupérer le mot de passe
                    String hashedPassword = AdminAdherentAjout.hashPassword(password); // Appeler la méthode hashPassw
                    if (storedPasswordHash.equals(hashedPassword)) {
                        // Afficher un message de connexion réussie
                        JOptionPane.showMessageDialog(frame, message:"Connexion réussie !", title:"Succès",
                                JOptionPane.INFORMATION_MESSAGE);

                        // Masquer la fenêtre de connexion
                        frame.setVisible(b:false);

                        // Créer une instance de la page d'accueil et l'initialiser en passant
                        // l'identifiant de l'adhérent
                        AccueilUser accueil = new AccueilUser();
                        accueil.initialize(adherentId); // username est l'identifiant de l'adhérent
                    } else {
                        // Afficher un message d'erreur si le mot de passe est incorrect
                        JOptionPane.showMessageDialog(frame, message:"Numéro utilisateur ou mot de passe incorrect.",
                                title:"Erreur de connexion", JOptionPane.ERROR_MESSAGE);
                    }
                } else {
                    // Afficher un message d'erreur si l'utilisateur n'existe pas
                    JOptionPane.showMessageDialog(frame, message:"Numéro utilisateur ou mot de passe incorrect.",
                            title:"Erreur de connexion", JOptionPane.ERROR_MESSAGE);
```

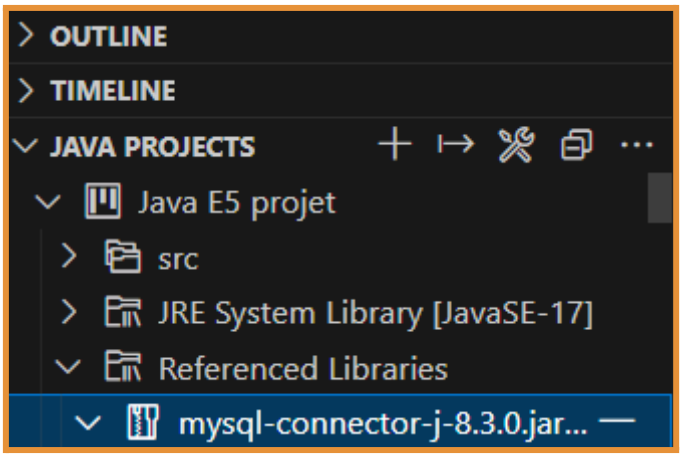
4.2.2. Programmation du projet

Le projet sera codé sur VSCode avec l'extension de java en y ajoutant des modules pour permettre d'ajouter des fonctionnalités.

En y ajoutant un fichier connector MySQL→Java, on pourra créer des interfaces spéciales pour l'affichage comme l'image ci-dessous.



Au niveau de la connexion à la base de données nous utiliserons un fichier connector qu'on va installer directement dans le projet.



Et au niveau du codage nous utiliserons les lignes suivantes pour pouvoir interagir avec la base de données.

```
6       // Information sur la base de données
7       private static final String URL = "jdbc:mysql://localhost:3306/bibliotheques-java";
8       private static final String USERNAME = "root";
9       private static final String PASSWORD = "root";
19          try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {
```

4.2.3. Mise en ligne du projet

Il est difficile de pouvoir le mettre en ligne donc nous allons laisser le projet en local.

4.2.4. Tests du projet

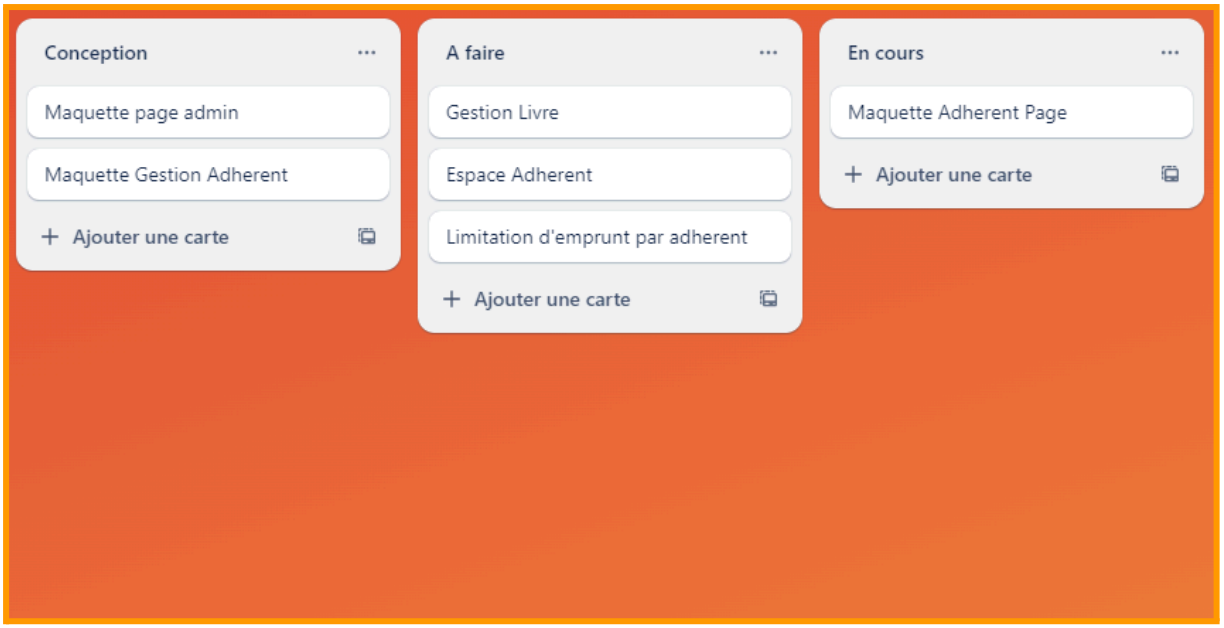
Ajout d'un jeu de test, afin de créer une base de données vide pour que l'utilisateur puisse tester le programme, saisir des données sur la base de données lui-même.

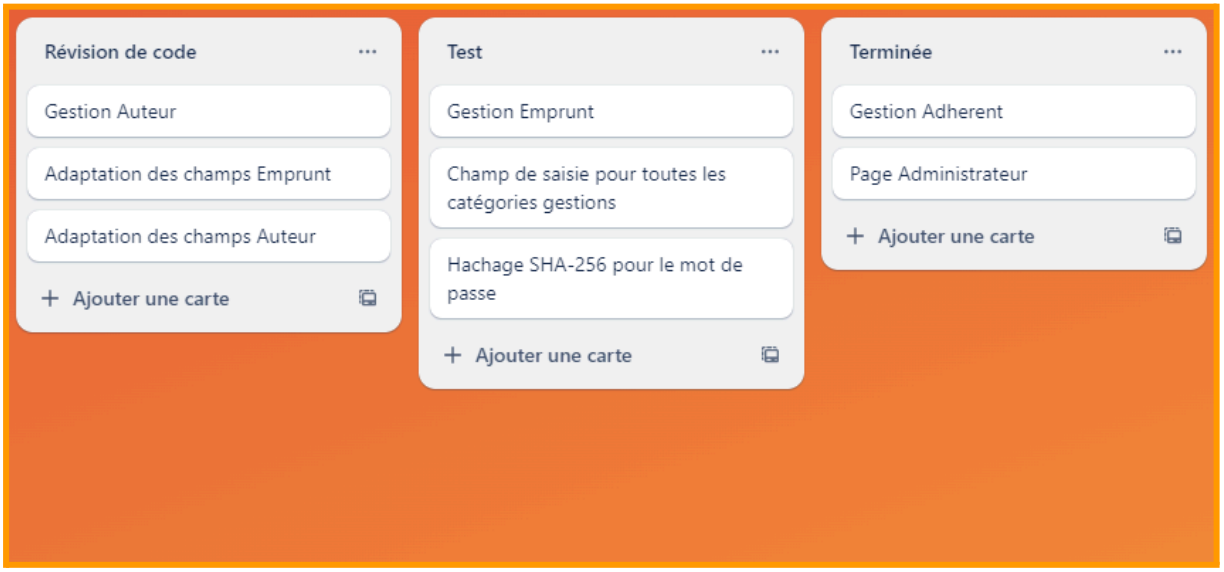
Tester l'affichage, l'ajout, la suppression et la modification des données.

Lors de l'ajout il faut faire attention à l'ordre de saisie car certaines tables doivent avoir des données déjà saisies dans d'autres tables (cf. 4.1).

4.3. Méthode de gestion projet

Nous utiliserons donc un outil de gestion projet se nommant trello pour mettre en place la méthode Agile Kanban tout au long du projet à partir du site web trello.





5. ANNEXE

5.1. Liens du projet

Liens redirigeant vers le GitHub du projet bibliothèques :
https://github.com/kilox-afk/JavaE5Library

Liens redirigeant vers le diagramme de Gantt :
https://docs.google.com/spreadsheets/d/1IrjTRmOENKEVCpN3mrsQrg-1Kc2DhgIIp5BmDXPtOsA/edit?usp=sharing

5.2. Code sql pour la base de données

```
CREATE TABLE ADHERENT (
  PRIMARY KEY AUTO_INCREMENT (Adh_num),
  Adh_num INT(4) NOT NULL,
  nom     VARCHAR(255),
  prenom  VARCHAR(255),
  email   VARCHAR(255),
  adresse  VARCHAR(255),
  nb_emprunt     INT(4),
  mot_de_passe VARCHAR(100)
);

CREATE TABLE ADMIN (
   PRIMARY KEY AUTO_INCREMENT (id_admin),
   id_admin INT(4) NOT NULL,
   login VARCHAR(255),
   password VARCHAR(100),
   nom VARCHAR(255),
   prenom VARCHAR(255),
   adresse VARCHAR(255)
 );

CREATE TABLE AUTEUR (
  PRIMARY KEY AUTO_INCREMENT (Aut_num),
  Aut_num        INT(4) NOT NULL,
  nom            VARCHAR(255),
  prenom         VARCHAR(255),
  date_naissance DATE,
  description    VARCHAR(42)
);

CREATE TABLE EMPRUNT (
  PRIMARY KEY AUTO_INCREMENT (id_emprunt),
  id_emprunt     INT(4) NOT NULL,
  date_emprunt   DATE,
  date_retour    DATE,
  statut_emprunt BOOLEAN,
  Adh_num INT(4) NOT NULL,
  ISBN   VARCHAR(20)
);

CREATE TABLE LIVRE (
  PRIMARY KEY (ISBN),
```

```
 ISBN       VARCHAR(20),
 titre      VARCHAR(255),
 prix       FLOAT(4),
 genre      VARCHAR(255),
 disponible BOOLEAN,
 nb_livre   INT(4),
 Aut_num    INT(4) NOT NULL
);

ALTER TABLE EMPRUNT ADD FOREIGN KEY (Adh_num) REFERENCES ADHERENT
(Adh_num);
ALTER TABLE EMPRUNT ADD FOREIGN KEY (ISBN) REFERENCES LIVRE (ISBN);

ALTER TABLE LIVRE ADD FOREIGN KEY (Aut_num) REFERENCES AUTEUR (Aut_num);
G
```