

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE
CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS
Département INFORMATIQUE

Spécialité informatique

Projet Algorithmique et programmation Java

Conception d'un Système de Gestion d'une Bibliothèque



08/03/2023

Réalisé par :

Assia Ghelis
Nke Belmond Florian
Mohamed SLIMANI

Encadré par :

Tatiana Aubonnet

Table des matières

I.	Introduction	3
II.	Etude du projet	4
1.	Présentation du projet	4
2.	Analyse du besoin.....	4
3.	Organisation du projet.....	4
4.	Conception	6
III.	Réalisation du projet	9
5.	Outils de développement	9
6.	Présentation de quelques interfaces développées	9
IV.	Conclusion et Perspectives	15
	Bibliographie et Webographie.....	16
	Annexe	18

I. Introduction

Le projet Java voit le jour en **1991**, dans le secret d'une équipe de Sun Microsystem. **Treize ingénieurs** ont cherché à concevoir **un langage applicable** à de petits appareils électriques (en code embarqué). Celui-ci a évolué au fil du temps et est aujourd'hui l'un des langages les plus utilisés principalement parce qu'il est **la base** de la plupart des applications **en réseau**. On recense actuellement **9 millions de développeurs Java** dans le monde qui utilisent cette technologie afin de développer et déployer efficacement des applications et des services en tout genre. Que ce soit des ordinateurs portables aux centres de données, des consoles de jeux aux superordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente **dans tous les domaines**.

De ce fait, mes camarades et moi avons décidé d'expérimenter l'efficacité de ce langage, en développant un système de gestion de bibliothèque. Ce système d'information a pour but de faciliter le travail des bibliothécaires au quotidien.

La réalisation de ce système de gestion bibliothèque s'est fait en deux étapes, d'abord l'étude du projet étape dans laquelle on présente le projet, l'organisation des étapes du projet, sa conception et enfin la réalisation du projet dans laquelle on présente les choix des solutions, des fonctionnalités, le développement du programme et **les tests effectués**. Ces étapes seront présentées dans la suite de notre rapport d'étude.

II. Etude du projet

Dans cette partie, nous allons présenter le projet, nous ferons ensuite une analyse des besoins avant de passer à son organisation et à sa conception.

1. Présentation du projet

Dans le cadre de la réalisation de notre projet académique java, nous avons opté pour le développement d'un système de gestion d'une bibliothèque. Ce système nous permet de mettre en pratique les notions de programmation java acquises pendant notre formation mais également les notions en conception de système d'information, leurs complexités et l'analyse du besoin. Le système de gestion d'une bibliothèque aura pour objectif de faciliter le travail des bibliothécaires au quotidien en automatisant certaines tâches telles que : l'emprunt, la gestion des adhérents et des sanctions. Un utilisateur pourra donc emprunter un livre en demandant la création d'un compte par le biais de la bibliothécaire. Si l'utilisateur est reconnu par le système et a déjà un emprunt en cours, le système invalide l'emprunt.

2. Analyse du besoin

Dans l'optique de moderniser leur bibliothèque et réduire la charge de travail de leur employé, la bibliothèque **marguerite** a émis le besoin de développer un système de gestion qui prendra en main de nombreuses tâches (la gestion des emprunts, des adhérents et des actions), celui-ci devra être ergonomique, veiller au respect des données personnelles, des interfaces simples avec des options essentiel tels que : l'ajout, la modification, l'actualisation et la suppression pour un livre par exemple.

La bibliothécaire pourra accéder à l'interface du système devra s'identifier en entrant un code spécial.

La complexité de ce projet, nous a permis de faire une révision complète des notions abordées en cours en programmation java et conception de système.

3. Organisation du projet

La réalisation de notre projet s'est faite en 03 phases à savoir:

La première phase a été une phase de choix de sujet et de constitution de notre équipe projet, durant cette phase nous avons décidé de travailler en trinôme d'une part et d'autre part nous avons eu de nombreuses idées de projet néanmoins il a fallu faire des choix en prenant en compte notre niveau de programmation en java, mais également le temps qui nous était dédié, après de nombreuse entretien avec notre encadreur, nous avons opté pour un système de gestion de bibliothèque.

La seconde phase a été une phase de planification, ici nous avons dû analyser les besoins matériels et logiciels donc nous aurions besoin, faire une analyse de la complexité de conception et réalisation du système, attribuer les tâches de manière équitable entre les

membres de l'équipe et établir un planning prévisionnel de l'ensemble des tâches à accomplir avant notre deadline.

La troisième phase a été la phase de réalisation du système, nous avons dans cette phase défini les différentes fonctionnalités, les acteurs pouvant intervenir sur le système, son architecture interne et de développement du code source.



Figure 1 : Planning prévisionnel du projet.

4. Conception

Dans la phase de conception nous avons fait usage des outils de conception, les logiciels utilisés pour la réalisation de l'ensemble des diagrammes ont été :

- **StarUML**, logiciel de modélisation UML (Unified Modeling Language) open source qui nous a été présenté au cours de notre formation, a été utilisé pour la réalisation du DU et du diagramme de classes.

- **Draw io**, logiciel utilisé pour la réalisation de modèles de tout type, il a été utilisé ici pour la réalisation du MCD.

Notre diagramme de cas d'utilisation nous a permis de présenter les différentes fonctionnalités du système et les acteurs pouvant intervenir dessus :

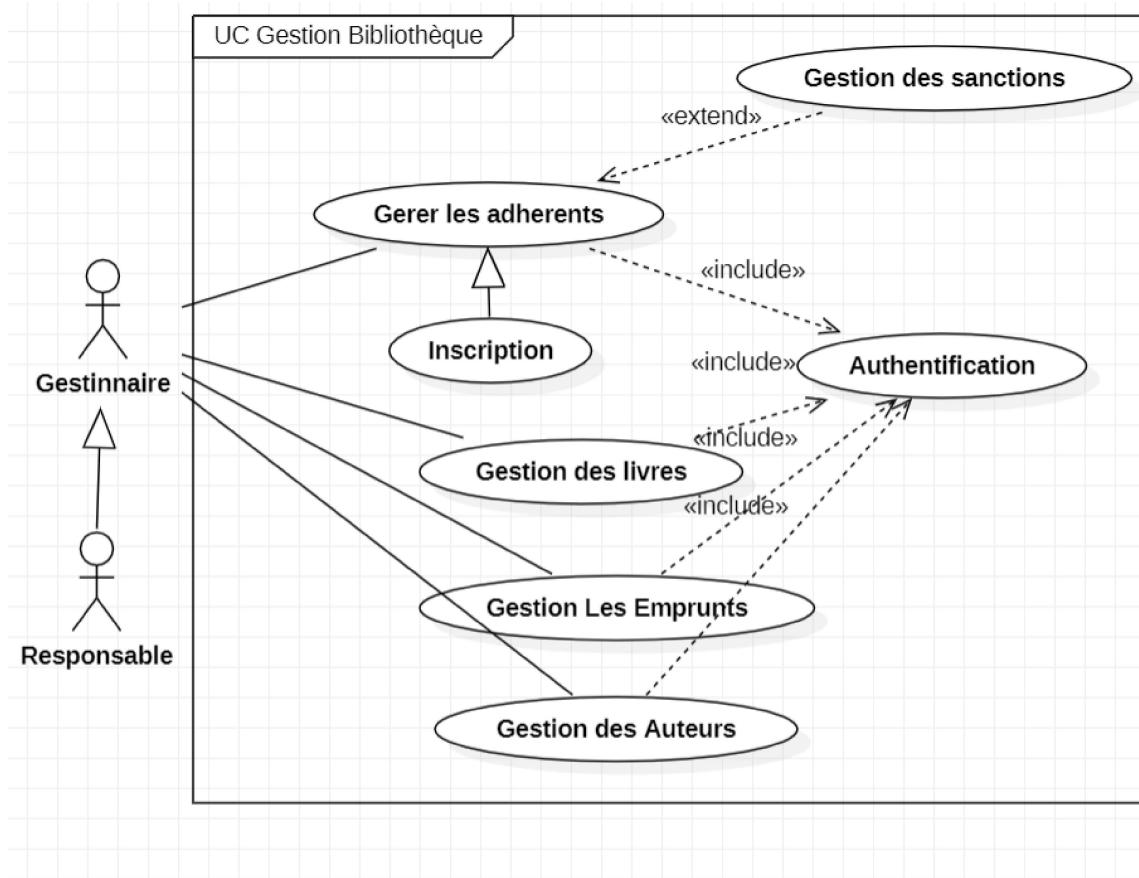


Figure 2 : Diagramme de cas d'utilisation.

Les principaux acteurs intervenant dans le système seront le bibliothécaire et le responsable de la bibliothèque. Pour ce qui est des fonctionnalités, le système pourra :

- Gérer les emprunts des livres et les sanctions, vérifier que l'utilisateur n'a pas dépassé le nombre d'emprunts permis et le sanctionner en cas de non-respect des délais de restitution.

- Gérer les adhésions principalement les inscriptions, au cas où un utilisateur ne veut pas juste consulter, mais aussi effectuer un emprunt.
- La mise à jour des livres.

Afin de modéliser l'organisation des données et identifier les principales entités types et les associations qui les relient, nous avons après une analyse du besoin réalisé un modèle conceptuel des données:

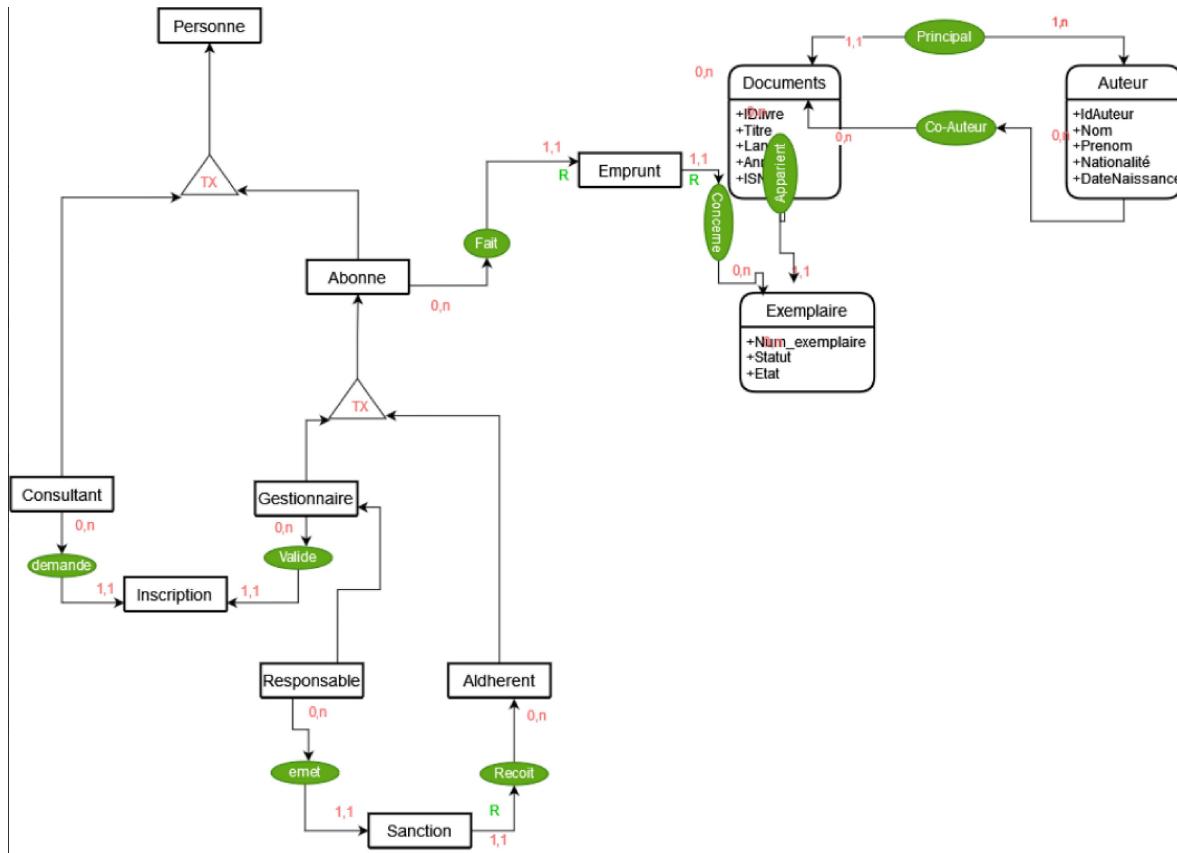


Figure 3 : Modèle conceptuel de données.

Cette étape est essentielle avant le passage au développement, car il permet d'avoir une vision claire de la gestion des données dans notre système, les contraintes permettant d'éviter la redondance d'informations.

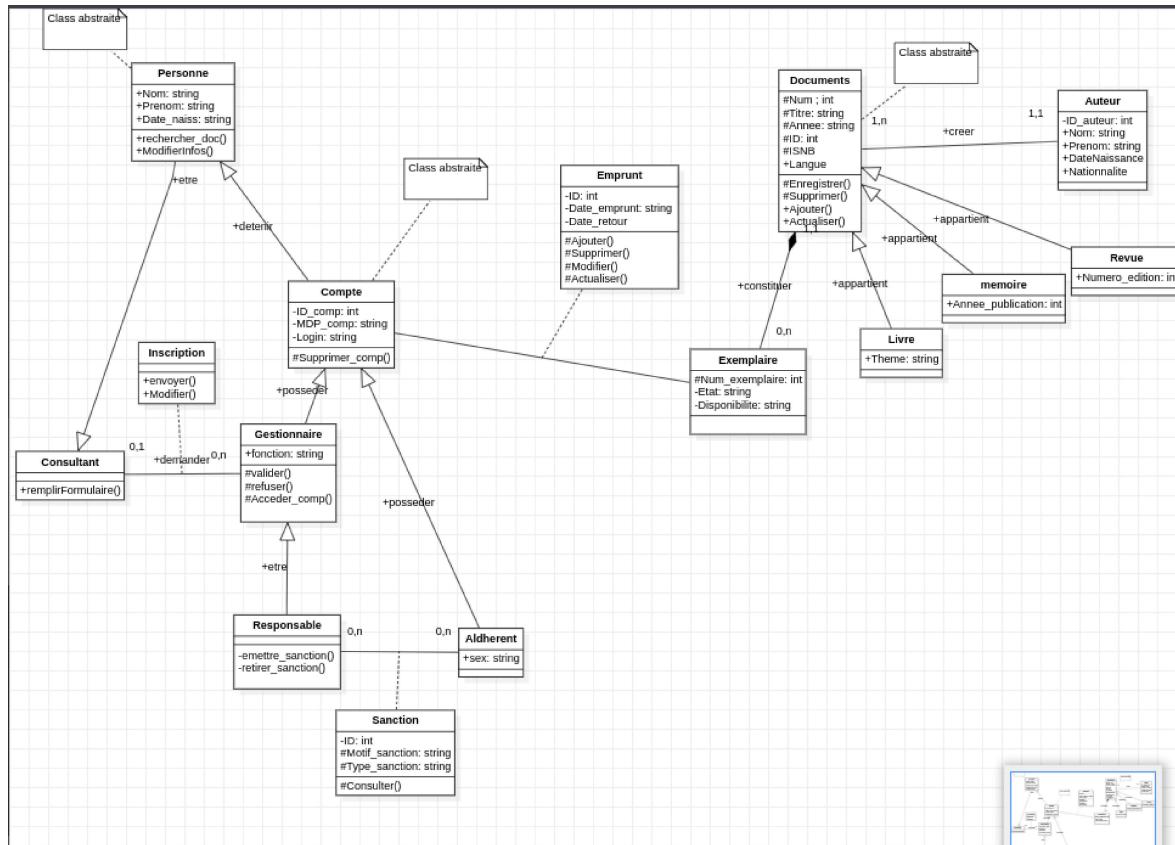


Figure 4 : Diagramme de classe.

L'une des difficultés majeures rencontrées dans cette partie a été d'analyser les besoins fonctionnels de notre système, mais également d'effectuer des choix qui nous permettent d'avoir un système efficace, pas très complexe au vu du temps dont nous disposions, toutefois nous permettant de monter en compétence.

III. Réalisation du projet

Après la phase de conception précédente, nous avons passé au développement de notre application.

5. Outils de développement

Nous avons commencé par créer notre base de données à l'aide du Logiciel SQL, que nous avons appelée Biblio et qui est constituée des Tables suivantes:

- Table auteur, livre, adhérent, emprunt, dans lesquels nous allons pouvoir nous connecter, ajouter, modifier et supprimer ou faire des recherches sur les auteurs, les livres, les adhérents .

Ensuite grâce au logiciel Java et l'environnement Netbeans, nous avons réalisé une interface graphique principale avec des fonctionnalités pour faciliter les opérations de gestion.



Figure 5 : Environnements de développement (Netbeans, phpMyadmin).

Le choix de ces environnements s'est fait car nous nous sommes sentis plus familier mais aussi l'accessibilité de ces environnements a été un facteur majeur de choix.

6. Présentation de quelques interfaces développées

Les parties du code source jugées pertinentes de notre projet seront présentées dans cette partie, parmi toutes les fonctionnalités que nous avons implémentées, nous en avons sélectionné quelques-unes pour vous les présenter.

L'interface principale comporte plusieurs fonctionnalités qui permettent à la gestionnaire de naviguer. Grâce aux différents boutons que nous avons conçus, elle peut accéder à la classe Auteur, Livre, Adhérent, ou bien la Classe Emprunt et ainsi faire un ajout, mettre à jour, supprimer ou bien faire une recherche.



Figure 6 : Interface principale.

D'autres fonctionnalités ont également été implémentées:

- Une table pour chaque classe dans lesquelles le gestionnaire peut visualiser les différentes opérations qu'elle réalise (ajout, mise à jour, suppression ou recherche) grâce à l'événement que nous avons programmé sur la souris).
- En cliquant sur le bouton **Auteur** la gestionnaire sera automatiquement redirigée vers l'interface Auteur.

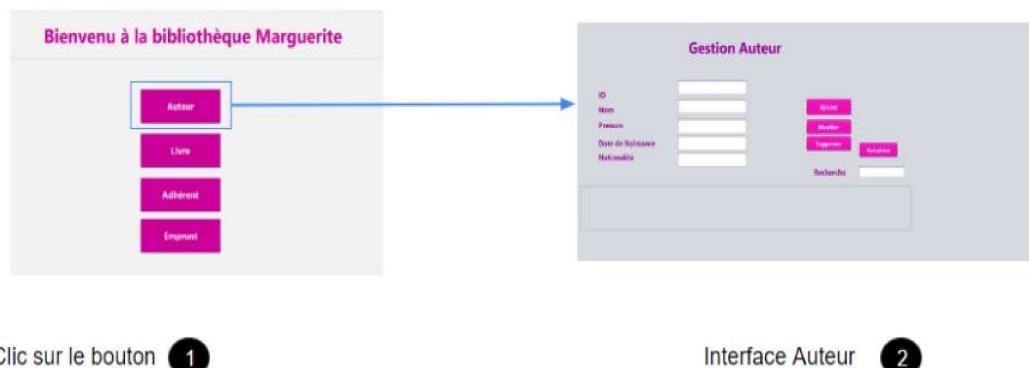


Figure 7 : Accéder à la classe auteur.

L'interface auteur, à partir de cette interface, la gestionnaire peut se connecter à la base de données **Biblio** dans la table auteur, elle peut ajouter, mettre à jour, supprimer ou faire des recherches, le résultat de l'opération sélectionnée sera affiché dans une table propre à chaque opération. Le code de ses opérations est décrit ci-dessous:

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/*
 *
 * @author 33699
 */
public class Auteur extends javax.swing.JFrame {
    Connection con;
    PreparedStatement st;
    ResultSet rs;
    String dbUrl="jdbc:mysql://localhost:3306/mabD?serverTimezone=Europe/Paris&useSSL=false";
    //Profil/mot de passe
    String user = "biblio";
    String password = "javal23";
    //connexion à la base
}

```

Figure 8 : Connection à la base de données classe auteur.

En appuyant sur le **bouton Ajouter**, qui nous permet d'ajouter un nouveau Auteur, voici le code qui sera exécuté.

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName( className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection( url:dbUrl,user,password);
        String sql = "insert into auteur(id,nom, prenom, naissance,nationalite) value (?, ?,?, ?, ?)";
        st=con.prepareStatement(sql);
        // st.setString(1,idAuteur1.getText());
        st.setString( parameterIndex: 2, x: nomAuteur.getText());
        st.setString( parameterIndex: 3, x: prenomAuteur.getText());
        st.setString( parameterIndex: 4, x: naissanceAuteur.getText());
        st.setString( parameterIndex: 5, x: nationaliteAuteur.getText());
        st.executeUpdate(sql);
        con.close();

        JOptionPane.showMessageDialog( parentComponent: null, message: "Enregistrement réussi");
        TableAuteur();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Figure 9 : Code du bouton d'ajout d'un auteur.

L'**Interface Livre**, en cliquant sur le bouton **Livre** à partir du menu principal. La gestionnaire sera automatiquement redirigée vers l'interface Livre, à partir duquel elle peut se connecter à la base de données Biblio dans la table livre.
Elle peut ajouter, mettre à jour, supprimer ou faire des recherches, le résultat de l'opération sélectionnée sera affiché dans une table propre à chaque opération. Le code de ses opérations est décrit ci-dessous:

Figure 10 : Interface Livre.

```
public class Livre extends javax.swing.JFrame {
    Connection con;
    PreparedStatement st;
    ResultSet rs;
    String dbUrl="jdbc:mysql://localhost:3306/maBD?serverTimezone=Europe/Paris&useSSL=false";
    //Profil/mot de passe
    String user = "biblio";
    String password = "java123";
    /**
     * Creates new form Auteur
     */
    public Livre() {
        initComponents();
        TableLivre();
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
}
```

Figure 11 : Connection à la base de données classe Livre.

Le bouton ajouter, permet à la gestionnaire d'ajouter un nouveau livre dans la table livre, pour cela voici le code que nous avons développé pour cette opération.



```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName(className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection(url:dbUrl,user,password);
        String sql = "insert into livre(idlivre,titre, langue, année, isbn, auteur, dateparrution, dateachat) value (?,?,?,?,?,?,?,?)";
        st=con.prepareStatement(sql);
        st.setString(parameterIndex: 1, x:idlivre.getText());
        st.setString(parameterIndex: 2, x:titrelivre.getText());
        st.setString(parameterIndex: 3, x:languelivre.getText());
        st.setString(parameterIndex: 4, x:annéelivre.getText());
        st.setString(parameterIndex: 5, x:isbnlivre.getText());
        st.setString(parameterIndex: 6, x:auteurlivre.getText());
        st.setString(parameterIndex: 7, x:dateparrutionlivre.getText());
        st.setString(parameterIndex: 8, x:dateachatlivre.getText());
        st.executeUpdate(sql);
        con.close();
        JOptionPane.showMessageDialog(parentComponent:null, message:"Enregistrement réussi");
        TableLivre();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Figure 12 : Code bouton d'ajout d'un livre.

La table livre, cette table permet d'afficher la liste des livres que nous allons insérer dans notre base de données.



```

public void TableLivre(){
String [] livre ={ "IDLIVRE", "TITRE", "LANGUE", "ANNEE", "ISBN", "AUTEUR", "DATEPARRUTION", "DATECHAT"};
String [] afficherl =new String [8];
DefaultTableModel model = DefaultTableModel( object: null, livre);
String sql= "select* from livre";

try {

Class.forName( className:"com.mysql.cj.jdbc.Driver");
con=DriverManager.getConnection( url:dbUrl,user,password);
rs=st.executeQuery();
while (rs.next()){

afficherl[0]= rs.getString( columnName: "idlivre");
afficherl[1]= rs.getString( columnName: "titrelivre");
afficherl[2]= rs.getString( columnName: "languelivre");
afficherl[3]= rs.getString( columnName: "annéelivre");
afficherl[4]= rs.getString( columnName: "isbnlivre");
afficherl[5]= rs.getString( columnName: "auteurlivre");
afficherl[6]= rs.getString( columnName: "dateparrutionlivre");
afficherl[7]= rs.getString( columnName: "dateachatlivre");
model.addRow( rowData:afficherl);

}

jTable2.setModel( dataModel: model);
con.close();
}

```

Figure 13 : Code de la table livre.

Le Bouton Modifier, Celui-ci permet au gestionnaire de supprimer un livre dans la table livre, pour cela, voici le code que nous avons développé pour cette opération.

```

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName(className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection(url, user, password);
        String sql = "Update livre set titre =?,langue=?, année=?, isbn=?, auteur=?, dateparrution =?,dateachat=?";
        st=con.prepareStatement(sql);
        st.setString(parameterIndex: 8, x:idlivre.getText());
        st.setString(parameterIndex: 1, x:titrelivre.getText());
        st.setString(parameterIndex: 2, x:languelivre.getText());
        st.setString(parameterIndex: 3, x:annéelivre.getText());
        st.setString(parameterIndex: 4, x:isbnlivre.getText());
        st.setString(parameterIndex: 5, x:auteurlivre.getText());
        st.setString(parameterIndex: 6, x:dateparrutionlivre.getText());
        st.setString(parameterIndex: 7, x:dateachatlivre.getText());
        st.executeUpdate(sql);
        con.close();

        JOptionPane.showMessageDialog(parentComponent: null, message: "Modification réussie");
        TableLivre();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Figure 14 : Code du bouton modifier de la classe livre.

Le bouton supprimer permet à la gestionnaire de supprimer un livre dans la table livre, pour cela, voici le code que nous avons développé pour cette opération.

```

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName(className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection(url, user, password);

        String sql = "Delete from livre where idlivre=?";
        st=con.prepareStatement(sql);
        st.setString(parameterIndex: 1, x:idlivre.getText());
        st.executeUpdate(sql);
        con.close();
        JOptionPane.showMessageDialog(parentComponent: null, message: "Suppression réussie");
        TableLivre();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Figure 15 : Code du bouton supprimer de la classe livre.

IV. Conclusion et Perspectives

En conclusion, notre projet a porté sur la conception d'un système de gestion d'une bibliothèque qui a pour but de faciliter le travail des bibliothécaires au quotidien d'une part et de nous permettre de nous perfectionner dans la programmation java d'autre part. Nous avons fait un travail d'analyse du besoin, puis nous sommes passés à la phase de conception avec les différents modèles conceptuels, enfin nous avons réalisé notre système avec les fonctionnalités nécessaires en implémentant des interfaces graphiques et le code nécessaires.

Toutefois au vu du temps dont nous disposions, nous n'avons pas pu réaliser certaines fonctionnalités (comme l'authentification de la gestionnaire par exemple), celles-ci pourront être réalisées dans une étude faisant suite à notre projet, elle serait une perspective intéressante.

Ce projet nous a permis non seulement de répondre au besoin de la bibliothèque Marguerites, mais aussi de mettre en valeurs nos connaissances théoriques acquises au cours de cette années.

Bibliographie et Webographie

[1] Cours d'introduction Algorithmique et programmation java, T. Aubonnet, Professeur au CNAM – Paris.

[2] Cours Type de logiciels systèmes d'information Méthodologies et technologies de système d'information, S. Si-said Cherfi, Professeur au CNAM – Paris.

[3] Cours Le Langage UML, Faten Atigui, Maître de conférences CNAM – Paris.

[4] Servet et Programmation Web, Lemoine Frédéric, Professeur au CNAM – Paris.

<https://openclassrooms.com/fr/courses/2434016-developpez-des-sites-web-avec-java-ee/2438671-envoyer-des-fichiers>

Liste des figures

Figure 1 : Planning prévisionnel du projet.....	5
Figure 2 : Diagramme de cas d'utilisation.....	6
Figure 3 : Modèle conceptuel de données.....	7
Figure 4 : Diagramme de classe.....	8
Figure 5 : Environnements de développement (Netbeans, phpMyadmin).....	9
Figure 6 : Interface principale.....	10
Figure 7 : Accéder à la classe auteur.....	10
Figure 8 : Connection à la base de données classe auteur.....	11
Figure 9 : Code du bouton d'ajout d'un auteur.....	11
Figure 10 : Interface Livre.....	12
Figure 11 : Connection à la base de données classe Livre.....	12
Figure 12 : Code bouton d'ajout d'un livre.....	13
Figure 13 : Code de la table livre.....	13
Figure 14 : Code du bouton modifier de la classe livre.....	14
Figure 15 : Code du bouton supprimer de la classe livre.....	15

Annexe

1. Présentation de quelques interfaces développées

La **Table Recherche** permet d'afficher à la gestionnaire les auteurs sur lesquels elle fait ses recherches.

```
public void TableLivreRecherche(String valeur){
    String [] livre={"IDLIVRE","TITRE","LANGUE","ANNEE","ISBN","AUTEUR","DATEPARRUTION","DATEACHAT"};
    String [] afficherl = new String [8];
    DefaultTableModel model = DefaultTableModel( object: null, livre);
    String sql= "select* from livre where titre like'%" +valeur+"%'";

    try {
        Class.forName( className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection( url:dbUrl,user,password);
        rs=st.executeQuery();
        while ( rs.next()){

            afficherl[0]= rs.getString( columnLabel:"idlivre");
            afficherl[1]= rs.getString( columnLabel:"titre");
            afficherl[2]= rs.getString( columnLabel:"languelivre ");
            afficherl[3]= rs.getString( columnLabel:"annéelivre");
            afficherl[4]= rs.getString( columnLabel:"isbnlivre");
            afficherl[5]= rs.getString( columnLabel:" auteurlivre");
            afficherl[6]= rs.getString( columnLabel:" dateparrutionlivre");
            afficherl[7]= rs.getString( columnLabel:" dateachatlivre");

            model.addRow( rowData:afficherl);

        }
        jTable2.setModel( dataModel: model);
        con.close();
    }
}
```

L'Interface **Adhérents**, à Partir de cette interface, la gestionnaire peut se connecter à la base de données Biblio dans la table adhérent, pour soit ajouter, modifier, supprimer ou faire une recherche sur un adhérent à partir des différents boutons. Voir le code de chaque bouton. Et le résultat sera également affiché dans une Table propre à chaque opération.

```
public class Adherent extends javax.swing.JFrame {
    Connection con;
    PreparedStatement st;
    ResultSet rs;
    String user = "biblio";
    String password = "javal23";
    String dbUrl="jdbc:mysql://localhost:3306/maBD?serverTimezone=Europe/Paris&useSSL=false";
    /**
     * Creates new form Auteur
     */
}
```

Le bouton Ajouter, ce bouton permet à la gestionnaire d'ajouter un adhérent dans la table adhérent, pour cela voici le code que nous avons développé pour cette opération.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName(className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection(url:dbUrl,user,password);
        String sql = "insert into adherent(id,nom, prenom, naissance,adresse,email) value (?, ?, ?, ?, ?, ?)";
        st=con.prepareStatement(sql);
        st.setString(parameterIndex:1, x:idadherent.getText());
        st.setString(parameterIndex:2, x:nomadherent.getText());
        st.setString(parameterIndex:3, x:prenomadherent.getText());
        st.setString(parameterIndex:4, x:naissanceadherent.getText());
        st.setString(parameterIndex:5, x:adresseadherent.getText());
        st.setString(parameterIndex:6, x:emailadherent.getText());
        st.executeUpdate(sql);
        con.close();

        JOptionPane.showMessageDialog(parentComponent:null, message:"Enregistrement réussi");
        //TableAdherent();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

La Table Adhérent, cette fonctionnalité permet d'afficher l'ajout d'un nouveau adhérent sur la base de données dans la table adhérent.

```
public void TableAdherent(){
    String [] Adherent ={"ID","Nom","Prenom","Naissance","Adresse", "Email"};
    String [] afficher2 =new String [6];
    //DefaultTableModel model = DefaultTableModel(null,Auteur);
    DefaultTableModel model = DefaultTableModel(object:null,Adherent);
    String sql= "select* from adherent";
    try {

        Class.forName(className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection(url:dbUrl,user,password);
        rs=st.executeQuery();
        while (rs.next()){

            afficher2[0]= rs.getString(columnLabel:"id");
            afficher2[1]= rs.getString(columnLabel:"nom");
            afficher2[2]= rs.getString(columnLabel:"prenom");
            afficher2[3]= rs.getString(columnLabel:"naissance");
            afficher2[4]= rs.getString(columnLabel:"adresse");
            afficher2[5]= rs.getString(columnLabel:"email");
            model.addRow( rowData:afficher2);
        }

        jTable3.setModel(dataModel:model);
        con.close();
    }

    catch (Exception e) {
        e.printStackTrace();
    }
}
```

```

private void jTable3MouseReleased(java.awt.event.MouseEvent evt) {
    int i= jTable3.getSelectedRow();
    DefaultTableModel model = (DefaultTableModel)jTable3.getModel();
    idadherent.setText( t: model.getValueAt( row:i, column:0).toString());
    nomadherent.setText( t: model.getValueAt( row:i, column:1).toString());
    prenomadherent.setText( t: model.getValueAt( row:i, column:2).toString());
    naissanceadherent.setText( t: model.getValueAt( row:i, column:3).toString());
    adresseadherent.setText( t: model.getValueAt( row:i, column:4).toString());
    emailadherent.setText( t: model.getValueAt( row:i, column:5).toString());
}

```

Le Bouton Modifier, ce bouton permet à la gestionnaire de modifier un adhérent dans la table adhérent, pour cela voici le code que nous avons développé pour cette opération.

```

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection( url: dbUrl,user,password);
        String sql = "Update adherent set nom=?, prenom=?, naissance=?,nationalité=? where id=?";
        st=con.prepareStatement(sql);
        st.setString( parameterIndex: 6, x: idadherent.getText());
        st.setString( parameterIndex: 1, x: nomadherent.getText());
        st.setString( parameterIndex: 2, x: prenomadherent.getText());
        st.setString( parameterIndex: 3, x: naissanceadherent.getText());
        st.setString( parameterIndex: 4, x: adresseadherent.getText());
        st.setString( parameterIndex: 5, x: emailadherent.getText());
        st.executeUpdate(sql);
        con.close();

        JOptionPane.showMessageDialog( parentComponent: null, message: "Modification réussie");
        TableAdherent();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Le Bouton Supprimer, ce bouton permet à la gestionnaire de supprimer un adhérent dans la table adhérent, pour cela voici le code que nous avons développé pour cette opération.

```

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    try {

        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection( url: dbUrl,user,password);
        String sql = "Delete from adherent where id=?";
        st=con.prepareStatement(sql);
        st.setString( parameterIndex: 1, x: idadherent.getText());
        st.executeUpdate(sql);
        con.close();
        JOptionPane.showMessageDialog( parentComponent: null, message: "Suppression réussie");
        TableAdherent();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        con=DriverManager.getConnection(url,username,password);
        String sql = "Delete from adherent where id=?";
        st=con.prepareStatement(sql);
        st.setString(1,idadherent.getText());
        st.executeUpdate(sql);
        JOptionPane.showMessageDialog(null, "Suppression réussie");
        Table1=Adherent();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Des tables pour afficher à la gestionnaire, la modification ou la suppression comme dans le cas de l'ajout ont été implémentées (voir le code sources).

L'**Interface Gestion des emprunts**, comme vous voyez grâce aux différents boutons la responsable de gestion peut gérer ses emprunts, en se connectant à la base de données Biblio dans la table emprunt.

Pour chacune des opérations que la gestionnaire souhaite faire, elle commence par se connecter à la base de données, puis soit ajouter soit mettre à jour ou bien supprimer.

Connection à la base de données:

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/*
 *
 * @author 33699
 */
public class Emprunt extends javax.swing.JFrame {
    Connection con;
    PreparedStatement st;
    ResultSet rs;
    String dbUrl="jdbc:mysql://localhost:3306/maBD?serverTimezone=Europe/Paris&useSSL=false";
    //Profil/mot de passe
    string user = "biblio";
    string password = "java123";
    /**
     * Creates new form Auteur
     */
    public Emprunt() {
        initComponents();
        TableEmprunt();
    }
}

```

Le Bouton Ajouter, ce bouton permet à la gestionnaire d'ajouter un emprunt dans la table emprunt, pour cela voici le code que nous avons développé pour cette opération.

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName( className:"com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection( url:dbUrl,user,password);
        String sql = "insert into Emprunt(idadherent,debut,fin, ,rendu) value (?,?,?,?,?)";
        st=con.prepareStatement(sql);
        st.setString( parameterIndex: 1, x: idadherent.getText());
        st.setString( parameterIndex: 2, x: debut.getText());
        st.setString( parameterIndex: 3, x: fin.getText());
        st.setString( parameterIndex: 4, x: rendu.getSelectedItem().toString());

        st.executeUpdate(sql);
        con.close();

        JOptionPane.showMessageDialog( parentComponent: null, message: "Enregistrement réussi");
        TableEmprunt();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Table d'affichage de la liste des emprunts

```

public void TableEmprunt(){
String [] Emprunt={"ID","Nom","Prenom","Naissance","Nationalité"};
String [] afficher =new String [4];
//DefaultTableModel model = DefaultTableModel(null,Auteur);
DefaultTableModel model = DefaultTableModel( object: null,Emprunt);
String sql= "select * from adherent";

try {

Class.forName( className:"com.mysql.cj.jdbc.Driver");
con=DriverManager.getConnection( url:dbUrl,user,password);
rset.executequery();
while (rs.next()){

afficher[0]= rs.getString( columnLabel: "idadherent");
afficher[1]= rs.getString( columnLabel: "debut");
afficher[2]= rs.getString( columnLabel: "fin");
afficher[3]= rs.getString( columnLabel: "rendu");

model.addRow( rowData:afficher);

}

jTable4.setModel( dataModel: model);
con.close();
}
catch (Exception e) {
e.printStackTrace();
}
}

```

Le Bouton Modifier, Ce bouton permet à la gestionnaire de mettre à jour un emprunt dans la table emprunt, pour cela voici le code que nous avons développé pour cette opération.

```

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
try {
Class.forName( className:"com.mysql.cj.jdbc.Driver");
con=DriverManager.getConnection( url:dbUrl,user,password);
String sql = "Update emprunt set datdebut=?, datefin=?, rendu=? where idadherent=?";
st=con.prepareStatement(sql);
st.setString( parameterIndex: 4, x: idadherent.getText());
st.setString( parameterIndex: 1, x: debut.getText());
st.setString( parameterIndex: 2, x: fin.getText());
st.setString( parameterIndex: 3, x: rendu.getSelectedItem().toString());
st.executeUpdate(sql);
con.close();

JOptionPane.showMessageDialog( parentComponent: null, message: "Modification réussie");
TableEmprunt();
} catch (Exception e) {
e.printStackTrace();
}
}

```

Le Bouton Supprimer, Ce bouton permet à la gestionnaire de supprimer un emprunt dans la table emprunt, pour cela voici le code que nous avons développé pour cette opération.

```

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        con=DriverManager.getConnection( url: dbUrl, user, password);
        String sql = "Delete from emprunt where rendu=?";
        st=con.prepareStatement(sql);
        st.setString( parameterIndex: 1, x: idadherent.getText());
        st.executeUpdate(sql);
        con.close();
        JOptionPane.showMessageDialog( parentComponent: null, message: "Suppression réussie");
        TableEmprunt();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Le Bouton Recherche, Ce bouton permet à la gestionnaire de faire une recherche sur un adhérent dans la table emprunt, pour cela voici le code que nous avons développé pour cette opération.

```

438 public void EmpruntRecherche(String valeur){
439     String [] Emprunt={"Idadherent","date_debut","date_fin","Rendu"};
440     String [] afficher =new String [4];
441     //DefaultTableModel model = DefaultTableModel(null,Auteur);
442     DefaultTableModel model = DefaultTableModel( object: null, Emprunt);
443     String sql= "select* from adherent where rendu like'%" +valeur+"%'";
444
445     try {
446
447         Class.forName( className: "com.mysql.cj.jdbc.Driver");
448         con=DriverManager.getConnection( url: dbUrl, user, password);
449         rset.executeQuery();
450         while ( rs.next()){
451             afficher[0]= rs.getString( columnLabel: "idadherent");
452             afficher[1]= rs.getString( columnLabel: "debut");
453             afficher[2]= rs.getString( columnLabel: "fin");
454             afficher[3]= rs.getString( columnLabel: "rendu");
455
456             model.addRow( rowData: afficher);
457
458         }
459
460         jTable4.setModel( dataModel: model);
461         con.close();
462     }
463
464     catch (Exception e) {
465         e.printStackTrace();
466     }
}

```