

Gestion d'une Bibliothèque En ligne

Enseigné par :

Mariem Bousaid

Salwa Zammali

Créé par :

Amri Rihem

Satouri Ranim

GLSI2 GROUPE2



Sommaire :

I-Introduction	3
II. Modèle Conceptuel de Données	4
III-Fonctionnalités de l'Application	5
IV-Captures d'écran	7
1-les Méthodes.....	7
2-les exception.....	27
3-les Interfaces.....	29
4-les présentation visuelles.....	53
V-Conclusion	69

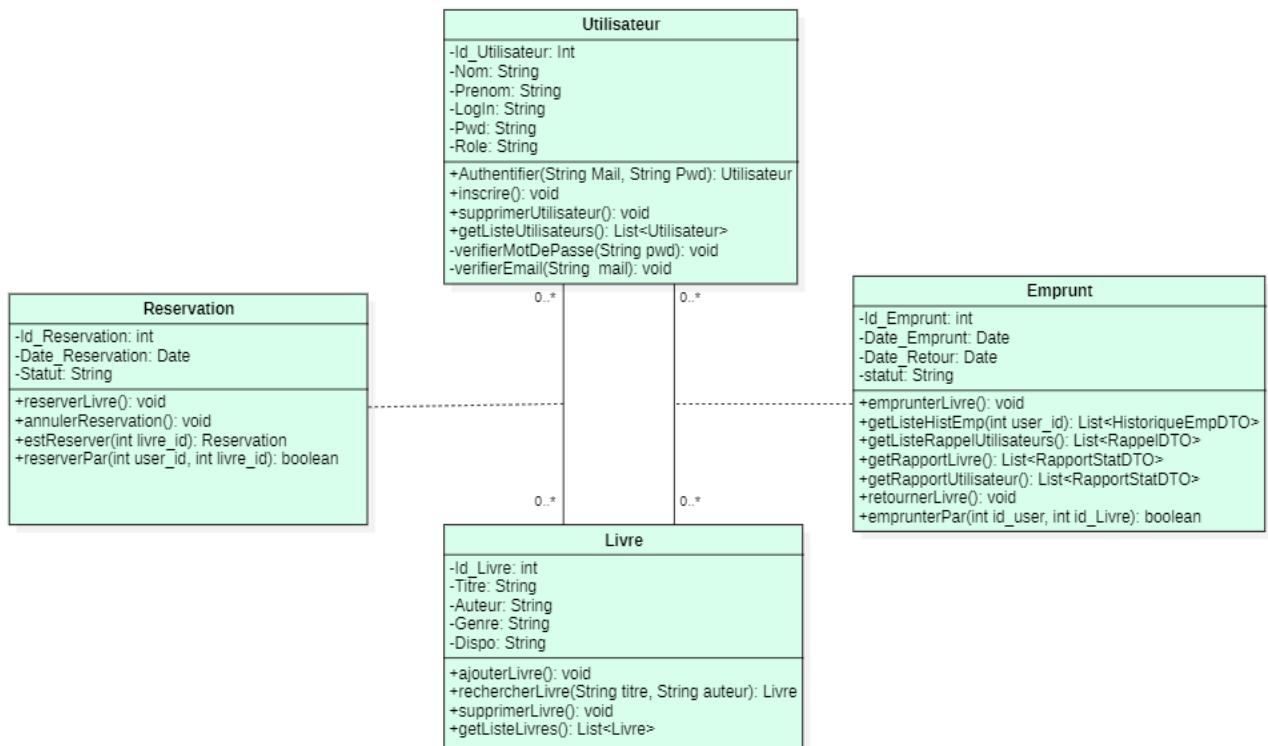
I-Introduction :

Ce projet vise à moderniser le système actuel d'emprunt et de retour des livres dans les bibliothèques en créant une application efficace en Java, garantissant ainsi un processus plus rapide et plus fluide. L'idée centrale derrière cette application est d'offrir aux utilisateurs (étudiants, enseignants) ainsi qu'aux bibliothécaires une interface conviviale tout en assurant l'automatisation de différentes tâches relatives à la gestion des livres.

II. Modèle Conceptuel de Données :

La base de données est conçue avec les entités suivantes : Utilisateur, Livre, Emprunt, et Réservation. Pour assurer une gestion adéquate des utilisateurs, des livres, des emprunts en cours et des réservations, ces entités recueillent toutes les informations nécessaires.

1) Diagram de class



Utilisateur
<ul style="list-style-type: none"> -Id_Utilisateur: Int -Nom: String -Prenom: String -Login: String -Pwd: String -Role: String <ul style="list-style-type: none"> +Authentifier(String Mail, String Pwd): Utilisateur +inscrire(): void +supprimerUtilisateur(): void +getlisteUtilisateurs(): List<Utilisateur> -verifierMotDePasse(String pwd): void -verifierEmail(String mail): void

Livre
<ul style="list-style-type: none"> -Id_Livre: int -Titre: String -Auteur: String -Genre: String -Dispo: String <ul style="list-style-type: none"> +ajouterLivre(): void +rechercherLivre(String titre, String auteur): Livre +supprimerLivre(): void +getlisteLivres(): List<Livre>

Emprunt
<ul style="list-style-type: none"> -Id_Emprunt: int -Date_Emprunt: Date -Date_Retour: Date -statut: String <ul style="list-style-type: none"> +emprunterLivre(): void +getlisteHistEmp(int user_id): List<HistoriqueEmpDTO> +getListeRappelUtilisateurs(): List<RappeIDTO> +getRapportLivre(): List<RapportStatDTO> +getRapportUtilisateur(): List<RapportStatDTO> +retournerLivre(): void +emprunterPar(int id_user, int id_Livre): boolean

Reservation
<ul style="list-style-type: none"> -Id_Reservation: int -Date_Reservation: Date -Statut: String <ul style="list-style-type: none"> +reserverLivre(): void +annulerReservation(): void +estReserver(int livre_id): Reservation +reserverPar(int user_id, int livre_id): boolean

III-Fonctionnalités de l'Application :

L'application offre plusieurs fonctionnalités :

-Authentification pour tous les utilisateurs.

-Consultation du catalogue de livres.

-Recherche de livres par les étudiants et enseignants.

-Gestion des emprunts et retours de livres.

-Réservation en ligne de livres indisponibles.

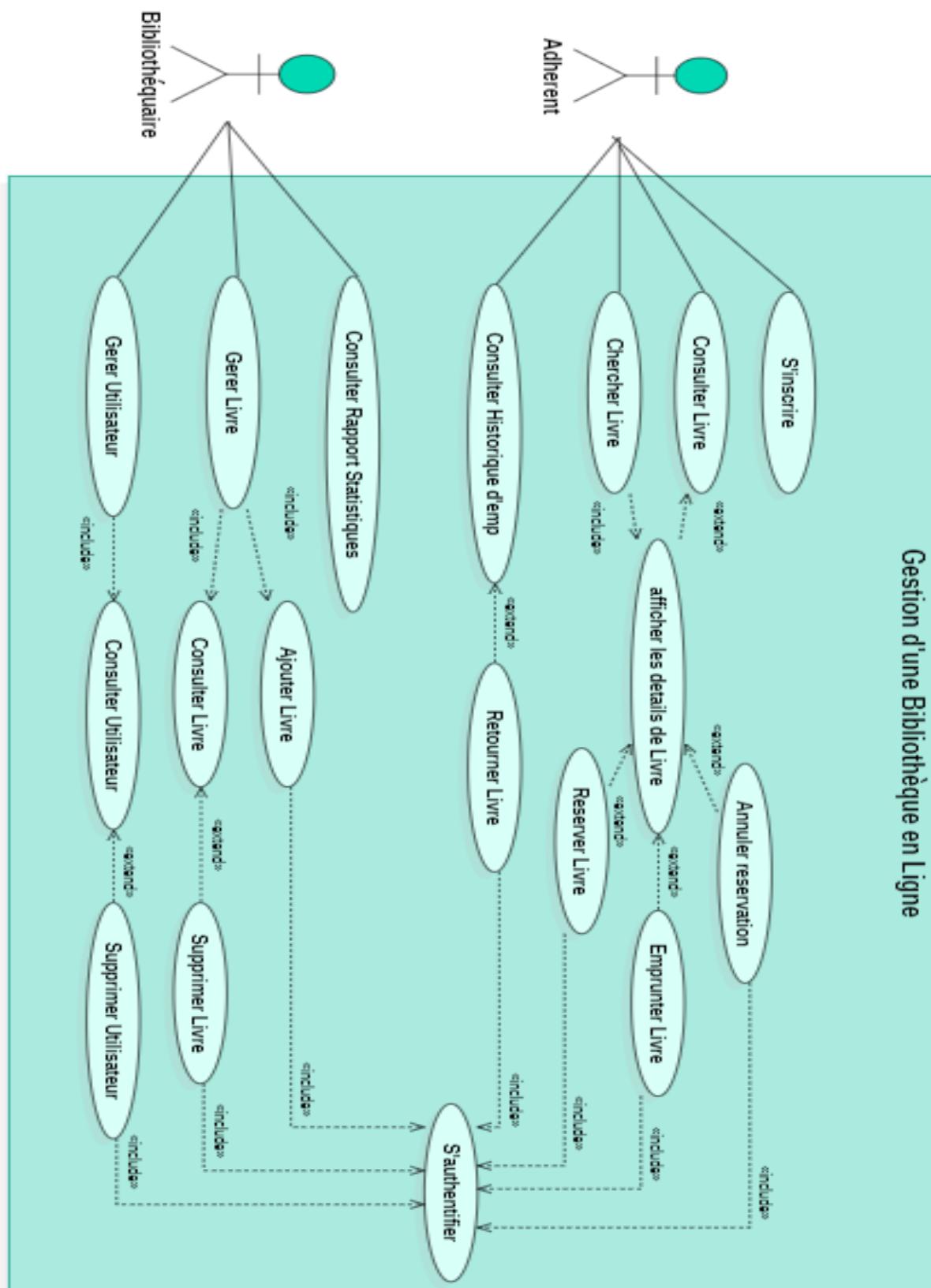
-Consultation de l'historique des emprunts pour un utilisateur donné.

-Notification par e-mail pour les rappels de retour (réservé aux bibliothécaires).

-Génération de rapports statistiques sur les livres les plus empruntés et les utilisateurs les plus assidus (réservé aux bibliothécaires).

Gestion d'une Bibliothèque en Ligne

2) Diagram de cas d'utilisation

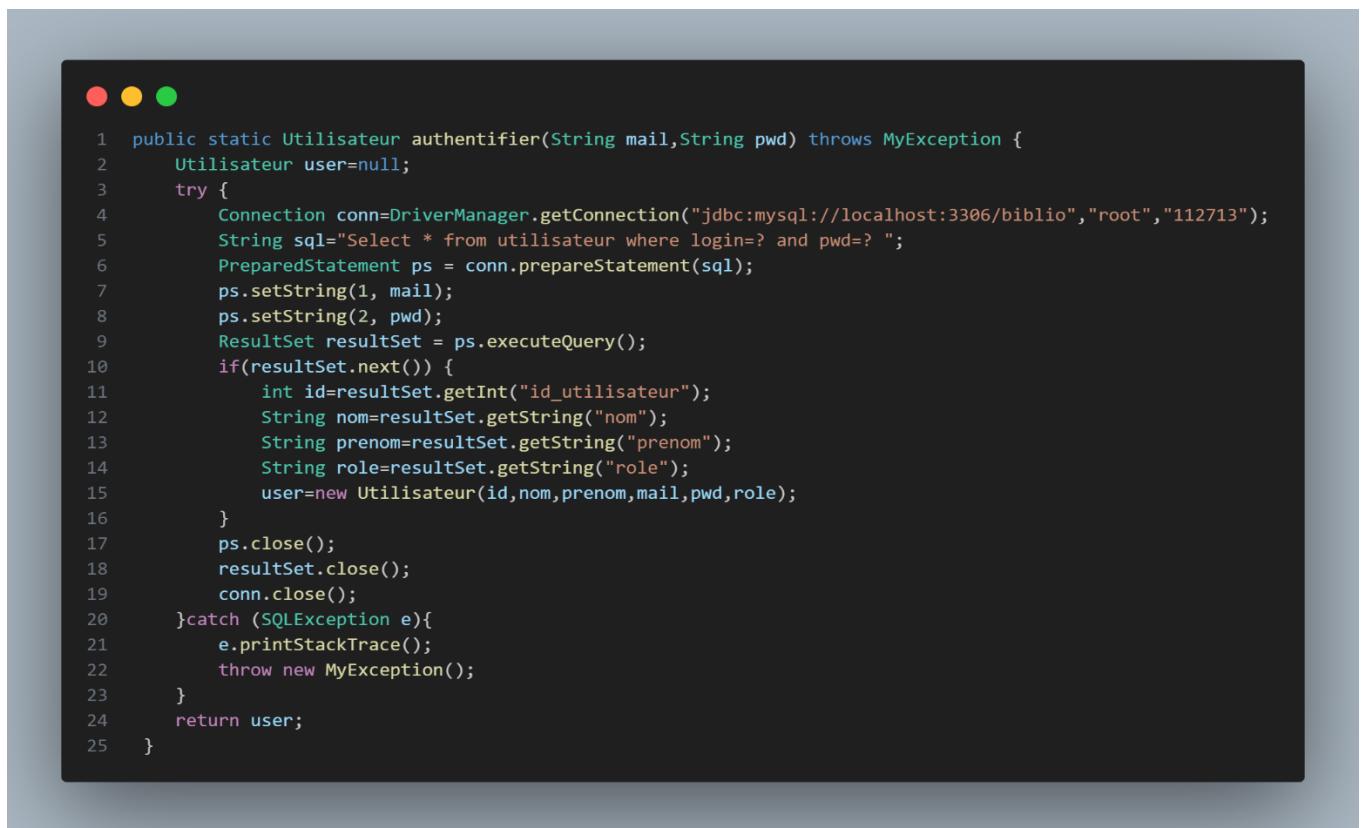


IV-Captures d'écran :

1)Les Méthodes :

Class Utilisateur :

Méthode authentifier :



```
1 public static Utilisateur authentifier(String mail,String pwd) throws MyException {
2     Utilisateur user=null;
3     try {
4         Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
5         String sql="Select * from utilisateur where login=? and pwd=? ";
6         PreparedStatement ps = conn.prepareStatement(sql);
7         ps.setString(1, mail);
8         ps.setString(2, pwd);
9         ResultSet resultSet = ps.executeQuery();
10        if(resultSet.next()) {
11            int id=resultSet.getInt("id_utilisateur");
12            String nom=resultSet.getString("nom");
13            String prenom=resultSet.getString("prenom");
14            String role=resultSet.getString("role");
15            user=new Utilisateur(id,nom,prenom,mail,pwd,role);
16        }
17        ps.close();
18        resultSet.close();
19        conn.close();
20    }catch (SQLException e){
21        e.printStackTrace();
22        throw new MyException();
23    }
24    return user;
25 }
```

Cet Méthode réalise l'authentification d'un utilisateur en interrogeant une base de données MySQL. Il prend en entrée une adresse e-mail (mail) et un mot de passe (pwd), puis exécute une requête SQL pour rechercher un utilisateur correspondant dans la table "utilisateur". Si une correspondance est trouvée, les informations de l'utilisateur sont récupérées et utilisées pour créer un objet de la classe Utilisateur. En cas d'échec de connexion à la base de données ou d'erreur SQL, une exception personnalisée de type MyException est levée .

Méthode Incrire :

```
● ● ●
1  public void inscrire() throws MailException , MyException ,PwdException {
2      try {
3          Utilisateur.verifierEmail(login);
4          Utilisateur.verifierMotDePasse(pwd);
5          Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
6          String sql="SELECT LOGIN FROM UTILISATEUR WHERE LOGIN=?";
7          PreparedStatement ps=conn.prepareStatement(sql);
8          ps.setString(1, login);
9          ResultSet rs=ps.executeQuery();
10         if(rs.next()) {
11             throw new MailException("EMail Exist Deja");
12         }else {
13             String sql2="INSERT INTO UTILISATEUR(nom,prenom,login, pwd,role) values(?,?,?,?,?)";
14             PreparedStatement ps2=conn.prepareStatement(sql2);
15             ps2.setString(1, nom);
16             ps2.setString(2, prenom);
17             ps2.setString(3, login);
18             ps2.setString(4, pwd);
19             ps2.setString(5, role);
20             ps2.executeUpdate();
21             ps2.close();
22         }
23         ps.close();
24         rs.close();
25         conn.close();
26     }catch(SQLException e){
27         e.printStackTrace();
28         throw new MyException();
29     }
30 }
```

La méthode vérifie d'abord si L'Email est valide si non , une exception personnalisée de type MailException est levée puis il vérifie le mot de passe et il déclenche une Exception personnalisée PwdException s'il est invalide , Ensuite il teste si l'adresse e-mail (login) existe déjà dans la table "utilisateur". Si c'est le cas, une exception personnalisée de type MailException est levée. Sinon, les informations de l'utilisateur sont insérées dans la base de données, et en cas d'erreur SQL, une exception de type MyException est déclenchée.

Méthode VerifierEmail:

```
● ● ●  
1 private static void verifierEmail(String email) throws MailException {  
2     int atIndex = email.indexOf("@");  
3     int dotIndex = email.lastIndexOf(".");  
4     if(!(atIndex > 0 && dotIndex > atIndex + 1 && dotIndex < email.length() - 1)) {  
5         throw new MailException("Email Invalid !");  
6     }  
7 }
```

La méthode `verifierEmail` examine la position de "@" et "." dans l'adresse e-mail fournie pour que l'email soit de la forme 'aaaa@aaa.aaa'. Si l'une des conditions n'est pas satisfaite, une exception de type `MailException` est levée pour signaler une adresse e-mail invalide.

Méthode VerifierMotDePasse:

```
● ● ●  
1 private static void verifierMotDePasse(String motDePasse) throws PwdException {  
2     if (motDePasse.length() < 8) {  
3         throw new PwdException("Le mot de passe doit avoir une longueur d'au moins 8 caractères.");  
4     }  
5     boolean Lettres = false;  
6     boolean Chiffres = false;  
7     for (char caractere : motDePasse.toCharArray()) {  
8         if (Character.isLetter(caractere)) {  
9             Lettres = true;  
10        }else if (Character.isDigit(caractere)) {  
11            Chiffres = true;  
12        }  
13    }  
14    if (!(Lettres && Chiffres)) {  
15        throw new PwdException("Le mot de passe doit contenir à la fois des lettres et des chiffres.");  
16    }  
17 }
```

La méthode VerifierMotDePasse examine la longueur du mot de passe (≥ 8) et parcourt chaque caractère pour vérifier la présence à la fois de lettres et de chiffres. Si l'un des conditions n'est pas satisfaite, une exception de type PwdException est levée pour signaler un mot de passe invalide.

Méthode getListeUtilisateurs :

```
● ● ●
1  public static List<Utilisateur> getListeUtilisateurs() throws MyException {
2      List<Utilisateur> listeUser = new ArrayList<>();
3      try {
4          Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/ biblio", "root", "112713");
5          String sql = "SELECT * FROM utilisateur WHERE role= ? or role =?";
6          PreparedStatement ps = conn .prepareStatement(sql);
7          ps.setString(1,"etudiant");
8          ps.setString(2,"enseignant");
9
10         ResultSet resultSet = ps.executeQuery();
11         while (resultSet.next()) {
12             int id=resultSet.getInt("id_Utilisateur");
13             String nom = resultSet.getString("nom");
14             String prenom = resultSet.getString("prenom");
15             String role=resultSet.getString("role");
16             Utilisateur user=new Utilisateur(id,nom,prenom,role);
17             listeUser.add(user);
18         }
19         ps.close();
20         conn.close();
21         resultSet.close();
22     }catch(SQLException e) {
23         e.printStackTrace();
24         throw new MyException();
25     }
26     return listeUser;
27 }
```

La méthode exécute une requête SQL Select pour sélectionner les utilisateurs ayant le rôle "étudiant" ou "enseignant", puis crée des objets Utilisateur avec les données récupérées et les ajoute à une liste. En cas d'échec de la connexion ou d'erreur SQL, une exception de type MyException est levée.

Class Livre :

Méthode RechercherLivre :

```
● ○ ●
1  public static Livre RechercherLivre(String title,String auteur) throws MyException {
2      Livre livre=null;
3      try {
4          Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
5          String sql="Select * from livre where titre=? and auteur=?";
6          PreparedStatement ps = conn.prepareStatement(sql);
7          ps.setString(1, title);
8          ps.setString(2, auteur);
9          ResultSet resultSet = ps.executeQuery();
10         if(resultSet.next()) {
11             String Genre=resultSet.getString("genre");
12             int id=resultSet.getInt("id_livre");
13             String dispo=resultSet.getString("dispo");
14             livre=new Livre(id,title,auteur,Genre,dispo);
15         }
16         ps.close();
17         resultSet.close();
18         conn.close();
19     }catch (SQLException e){
20         e.printStackTrace();
21         throw new MyException();
22     }
23     return livre;
24 }
```

La méthode établit une requête SQL Select pour rechercher un livre avec le titre et l'auteur fournis, puis crée un objet Livre avec les données récupérées s'il existe. En cas d'échec de la connexion ou d'erreur SQL, une exception de type MyException est levée.

Méthode SupprimerLivre:

```
● ● ●
1 public void supprimerLivre() throws MyException{
2     try {
3         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/ biblio", "root", "112713");
4         String sql="DELETE FROM livre WHERE Id_Livre= ?";
5         PreparedStatement ps = conn.prepareStatement(sql);
6         ps.setInt(1, id);
7         ps.executeUpdate();
8         ps.close();
9         conn.close();
10    }catch (SQLException e){
11        e.printStackTrace();
12        throw new MyException();
13    }
14 }
```

La méthode établit une requête SQL DELETE pour supprimer le livre correspondant à l'identifiant fourni, puis ferme les ressources de base de données. En cas d'échec de la connexion ou d'erreur SQL, une exception de type MyException est levée.

Méthode AjouterLivre:

```
● ● ●
1 public void AjouterLivre() throws MyException{
2     try {
3         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/ Biblio", "root", "112713");
4         String sql="Insert INTO Livre (Titre, Auteur,Genre,dispo) values(?, ?, ?, 'Disponible')";
5         PreparedStatement ps = conn.prepareStatement(sql);
6         ps.setString(1, Titre);
7         ps.setString(2, Auteur);
8         ps.setString(3, Genre);
9         ps.executeUpdate();
10        ps.close();
11        conn.close();
12    }catch (SQLException e){
13        e.printStackTrace();
14        throw new MyException();
15    }
16 }
```

La méthode établit une requête SQL INSERT pour ajouter un nouveau livre avec les informations fournies. En cas d'échec de la connexion ou d'erreur SQL, une exception de type MyException est levée.

Méthode `getlisteLivres`:

```
 1 public static List<Livre> getlisteLivres() throws MyException {  
 2     List<Livre> listeLivres = new ArrayList<>();  
 3  
 4     try {  
 5         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio", "root", "112713");  
 6         String sql = "SELECT * FROM livre";  
 7         Statement ps = conn.createStatement();  
 8         ResultSet resultSet = ps.executeQuery(sql);  
 9  
10        while (resultSet.next()) {  
11            String titre = resultSet.getString("titre");  
12            String auteur = resultSet.getString("auteur");  
13            Livre livre = new Livre(titre, auteur);  
14            listeLivres.add(livre);  
15        }  
16        ps.close();  
17        resultSet.close();  
18        conn.close();  
19    }catch(SQLException e) {  
20        e.printStackTrace();  
21        throw new MyException();  
22    }  
23    return listeLivres;  
24 }
```

La méthode une requête SQL pour sélectionner tous les livres, puis crée des objets *Livre* avec les données récupérées et les ajoute à une liste. En cas d'échec de la connexion ou d'erreur SQL, une exception de type *MyException* est levée. La liste des livres est ensuite renvoyée.

Class Emprunt :

Méthode EmprunterLivre:

```
● ○ ●

1 public void emprunterLivre() throws MyException {
2     try {
3         Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
4         String sql="insert into emprunt(id_utilisateur,id_livre,date_emprunt,date_retour,statut)+"+
5             " values(?,?,current_date(),DATE_ADD(current_date(), INTERVAL 15 DAY),'En_Cours')";
6         String sql2="update livre set dispo='Indisponible' where id_livre=?";
7         PreparedStatement ps = conn.prepareStatement(sql);
8         PreparedStatement ps2 = conn.prepareStatement(sql2);
9         ps.setInt(1, user_id);
10        ps.setInt(2, book_id);
11        ps2.setInt(1,book_id);
12        ps.executeUpdate();
13        ps2.executeUpdate();
14        ps.close();
15        ps2.close();
16        conn.close();
17    }catch (SQLException e){
18        e.printStackTrace();
19        throw new MyException();
20    }
21 }
```

La méthode établit deux requêtes SQL : une pour insérer un nouvel enregistrement d'emprunt avec l'ID de l'utilisateur, l'ID du livre et les dates d'emprunt et de retour, et une autre pour mettre à jour le statut de disponibilité du livre. En cas d'échec de la connexion ou d'erreur SQL, une exception de type MyException est levée.

Méthode EmprunterPar:

```
 1 public static boolean emprunterPar(int user_id ,int book_id) throws MyException{  
 2     boolean empPar=false;  
 3     try {  
 4         Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");  
 5         String sql="Select * from emprunt where id_utilisateur=? and id_livre=? and statut='En_Cours' ";  
 6         PreparedStatement ps = conn.prepareStatement(sql);  
 7         ps.setInt(1, user_id);  
 8         ps.setInt(2, book_id);  
 9         ResultSet rs=ps.executeQuery();  
10         if(rs.next()) {  
11             empPar=true;  
12         }  
13         ps.close();  
14         rs.close();  
15         conn.close();  
16     }catch(SQLException e) {  
17         e.printStackTrace();  
18         throw new MyException();  
19     }  
20     return empPar;  
21 }  
22 }
```

La méthode établit une requête SQL pour vérifier si l'utilisateur avec l'ID user_id a déjà emprunté le livre avec l'ID book_id et que le statut de l'emprunt est "en_cours". Si tel est le cas, la variable empPar est définie sur true. En cas d'échec de la connexion ou d'erreur SQL, une exception de type MyException est levée. La méthode renvoie ensuite la valeur de empPar .

Le but de cette Méthode est de savoir si l'utilisateur est celui qui a Emprunté le livre pour que on ne lui affiche pas le bouton <Emprunter> encore une fois .

Méthode RetournerLivre :

```
● ● ●
1  public void retournerLivre() throws MyException {
2      try {
3          Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
4          String sql="update emprunt set date_retour=current_date() , statut='Terminé' where id_emprunt=?";
5          PreparedStatement ps = conn.prepareStatement(sql);
6          ps.setInt(1, emp_id);
7          ps.executeUpdate();
8          Reservation res=Reservation.estReserver(book_id);
9          if(res==null) {
10              String sql2="update livre set dispo='Disponible' where id_livre=?";
11              PreparedStatement ps2 = conn.prepareStatement(sql2);
12              ps2.setInt(1,book_id);
13              ps2.executeUpdate();
14              ps2.close();
15          }else {
16              String sql3="insert into emprunt(id_utilisateur,id_livre,date_emprunt,date_retour,statut)+"+
17                  "values(?,?,current_date(),DATE_ADD(current_date(), INTERVAL 15 DAY),'En_Cours')";
18              PreparedStatement ps3 = conn.prepareStatement(sql3);
19              ps3.setInt(1,res.GetUserId());
20              ps3.setInt(2,book_id);
21              ps3.executeUpdate();
22              String sql2="update reservation set statut='Terminé' where id_reservation=?";
23              PreparedStatement ps2 = conn.prepareStatement(sql2);
24              ps2.setInt(1,res.GetResId());
25              ps2.executeUpdate();
26              ps3.close();
27              ps2.close();
28          }
29          ps.close();
30          conn.close();
31      }catch (SQLException e){
32          e.printStackTrace();
33          throw new MyException();
34      }
35  }
```

la méthode retourner livre exécute une requête SQL pour la mise a jour de statut de l'emprunt(Terminé) et la date retour(Current_Date) puis elle vérifie si le livre est réservé en appelant la méthode estReserver de la class réservation si ce n'est pas le cas une mise a jour sur la disponibilité de livre est effectuée(Disponible) sinon une instance <res> est retourné de la première réservation effectuée sur ce livre alors ce dernier devient l'emprunteur et par suite l'insertion dans la table emprunt par le nouvel emprunt effectué, et une mise à jour de statut = 'Terminé' pour la réservation .

Méthode *GetListeHistEmp*:

```
● ● ●
1 public static List<HistoriqueEmpDTO> getListeHistEmp(int user_id) throws MyException {
2     List<HistoriqueEmpDTO> liste = new ArrayList<>();
3     try {
4         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio", "root", "112713");
5         String sql = "SELECT id_emprunt,titre,auteur,date_emprunt,date_retour,statut FROM emprunt e join livre l"+
6         " where e.id_livre=l.id_livre and id_utilisateur=?";
7         PreparedStatement ps = conn.prepareStatement(sql);
8         ps.setInt(1, user_id);
9         ResultSet resultSet = ps.executeQuery();
10
11        while (resultSet.next()) {
12            int id_emprunt = resultSet.getInt("id_emprunt");
13            String titre=resultSet.getString("titre");
14            String auteur=resultSet.getString("auteur");
15            String date_emprunt = resultSet.getString("Date_emprunt");
16            String date_retour = resultSet.getString("Date_retour");
17            String statut=resultSet.getString("statut");
18            HistoriqueEmpDTO object=new HistoriqueEmpDTO(id_emprunt,titre,auteur,date_emprunt,date_retour,statut);
19            liste.add(object);
20        }
21        ps.close();
22        resultSet.close();
23        conn.close();
24    }catch(SQLException e) {
25        e.printStackTrace();
26        throw new MyException();
27    }
28    return liste;
29 }
```

Cette méthode récupère les informations d'historique des emprunts (ID, titre, auteur, date d'emprunt, date de retour, statut) des livres empruntés par l'utilisateur avec l'ID user_id. Les résultats sont encapsulés dans des objets HistoriqueEmpDTO qui sont ajoutés à une liste, laquelle est ensuite renvoyée. En cas d'échec de connexion ou d'erreur SQL, une exception de type MyException est déclenchée.

Méthode *GetRapportUtilisateur*:

```
 1  public static List<RapportStatDTO> getRapportUtilisateur() throws MyException{
 2      List<RapportStatDTO> liste = new ArrayList<>();
 3      try {
 4          Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/ biblio", "root", "112713");
 5          String sql = "SELECT count(e.Id_Utilisateur) as nb,Nom,Prenom FROM Emprunt e,Utilisateur u "+
 6          "where e. Id_Utilisateur=u. Id_Utilisateur group by e.Id_Utilisateur order by nb DESC";
 7          PreparedStatement ps = conn .prepareStatement(sql);
 8          ResultSet resultSet = ps.executeQuery();
 9          while (resultSet.next()) {
10              int nb=resultSet.getInt("nb");
11              String nom = resultSet.getString("Nom");
12              String prenom = resultSet.getString("Prenom");
13              RapportStatDTO object=new RapportStatDTO(nb,nom,prenom);
14              liste.add(object);
15          }
16          resultSet.close();
17          ps.close();
18          conn.close();
19      } catch (SQLException e) {
20          e.printStackTrace();
21          throw new MyException();
22      }
23      return liste;
24  }
```

Cette méthode récupère le nombre d'emprunts par utilisateur, et crée des objets RapportStatDTO contenant ces données (nombre d'emprunts, nom, prénom). Les résultats sont ordonnés par nombre d'emprunts décroissant et stockés dans une liste, qui est ensuite renvoyée. En cas d'échec de connexion ou d'erreur SQL, une exception de type MyException est générée.

Méthode *GetRapportLivre*:

```
● ● ●
1 public static List<RapportStatDTO> getRapportLivre()throws MyException{
2     List<RapportStatDTO> liste = new ArrayList<>();
3     try {
4         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio", "root", "112713");
5         String sql = "SELECT count(e.Id_Livre) as nb,Titre,Auteur FROM Emprunt e,Livre l where e.Id_Livre=l.Id_Livre"+
6             "group by e.Id_Livre order by nb DESC ";
7         PreparedStatement ps = conn.prepareStatement(sql);
8         ResultSet resultSet = ps.executeQuery();
9         while (resultSet.next()) {
10             int nb=resultSet.getInt("nb");
11             String titre = resultSet.getString("Titre");
12             String auteur = resultSet.getString("Auteur");
13             RapportStatDTO object=new RapportStatDTO(nb,titre,auteur);
14             liste.add(object);
15         }
16         resultSet.close();
17         ps.close();
18         conn.close();
19     } catch (SQLException e) {
20         e.printStackTrace();
21     }
22     return liste;
23 }
```

Cette méthode récupère le nombre d'emprunts par livre, et crée des objets *RapportStatDTO* contenant ces données (*nombre d'emprunts, titre, auteur*). Les résultats sont ordonnés par nombre d'emprunts décroissant et stockés dans une liste, qui est ensuite renvoyée. En cas d'échec de connexion ou d'erreur SQL, une exception de type *MyException* est générée.

Méthode *GetListeRappelUtilisateurs*:

```
● ● ●
1  public static List<RappelDTO> getListeRappelUtilisateurs() throws MyException{
2      List<RappelDTO> liste = new ArrayList<>();
3      try {
4          Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio", "root", "112713");
5          String sql = "SELECT e.Id_Utilisateur ,e.Id_Livre,Nom,Prenom,Titre,auteur ,Date_Emprunt,Date_Retour"+
6          " FROM Emprunt e,Utilisateur u, Livre l where ( e. Id_Utilisateur=u. Id_Utilisateur and e.Id_Livre=l.Id_Livre"+
7          "and Date_Retour-curdate()=1 and Date_Emprunt-curdate()=1 and statut='En_Cours')";
8          PreparedStatement ps = conn .prepareStatement(sql);
9          ResultSet resultSet = ps.executeQuery();
10         while (resultSet.next()) {
11             String nom = resultSet.getString("Nom");
12             String prenom = resultSet.getString("Prenom");
13             String titre = resultSet.getString("Titre");
14             String auteur = resultSet.getString("auteur");
15             String date_emprunt = resultSet.getString("Date_Emprunt");
16             String date_retour = resultSet.getString("Date_Retour");
17             RappelDTO object=new RappelDTO(nom,prenom,titre,auteur,date_emprunt,date_retour);
18             liste.add(object);
19         }
20         resultSet.close();
21         ps.close();
22         conn.close();
23     }catch (SQLException e) {
24         e.printStackTrace();
25         throw new MyException();
26     }
27 }
28 return liste;
29 }
```

Cette méthode récupère la liste des emprunts qui l'ont reste un jour pour la date de retour , et crée des objets RappelDTO contenant ces données (nom, prénom,titre,auteur,date_emprunt,date_retour) stockés dans une liste, qui est ensuite renvoyée. En cas d'échec de connexion ou d'erreur SQL, une exception de type MyException est générée.

DTO :

Un DTO (Data Transfer Object) en Java est une classe qui transporte des données entre différentes parties d'une application. Elle est utilisée pour encapsuler un ensemble de données et permettre leur transfert facile entre les différentes couches de l'application

```
1 public class HistoriqueEmpDTO {  
2     private int id_emprunt;  
3     private String titre;  
4     private String auteur;  
5     private String date_emprunt;  
6     private String date_retour;  
7     private String statut;  
8  
9     public HistoriqueEmpDTO(int id_emprunt, String titre, String auteur, String date_emprunt, String date_retour, String statut) {  
10        this.id_emprunt = id_emprunt;  
11        this.titre = titre;  
12        this.auteur = auteur;  
13        this.date_emprunt = date_emprunt;  
14        this.date_retour = date_retour;  
15        this.statut = statut;  
16    }  
17    public int getId_emprunt() {  
18        return id_emprunt;  
19    }  
20  
21    public String getTitre() {  
22        return titre;  
23    }  
24  
25    public String getAuteur() {  
26        return auteur;  
27    }  
28  
29    public String getDate_emprunt() {  
30        return date_emprunt;  
31    }  
32  
33    public String getDate_retour() {  
34        return date_retour;  
35    }  
36  
37    public String getStatut() {  
38        return statut;  
39    }  
40 }
```

```
1 public class RappelDTO {
2     private String nom;
3     private String prenom;
4     private String titre;
5     private String auteur;
6     private String date_emprunt;
7     private String date_retour;
8
9     public RappelDTO(String nom, String prenom, String titre, String auteur, String date_emprunt, String date_retour) {
10         this.nom = nom;
11         this.prenom = prenom;
12         this.titre = titre;
13         this.auteur = auteur;
14         this.date_emprunt = date_emprunt;
15         this.date_retour = date_retour;
16     }
17     public String getNom() {
18         return nom;
19     }
20
21     public String getPrenom() {
22         return prenom;
23     }
24
25     public String getTitre() {
26         return titre;
27     }
28
29     public String getAuteur() {
30         return auteur;
31     }
32
33     public String getDate_emprunt() {
34         return date_emprunt;
35     }
36
37     public String getDate_retour() {
38         return date_retour;
39     }
40 }
```

```
1 public class RapportStatDTO {
2     private int nbEmp;
3     private String chaine1;
4     private String chaine2;
5
6     public RapportStatDTO(int nbEmp, String chaine1, String chaine2) {
7         this.nbEmp = nbEmp;
8         this.chaine1 = chaine1;
9         this.chaine2 = chaine2;
10    }
11    public int getNbEmp() {
12        return nbEmp;
13    }
14    public String getChaine1() {
15        return chaine1;
16    }
17    public String getChaine2() {
18        return chaine2;
19    }
20 }
```

Class Reservation :

Méthode ReserverLivre:

```
● ● ●
1 public void reserverLivre() throws MyException {
2     try {
3         Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
4         String sql="insert into reservation(id_utilisateur,id_livre,date_reservation,statut) values(?,?,NOW(),'En_Cours')";
5         PreparedStatement ps = conn.prepareStatement(sql);
6         ps.setInt(1, user_id);
7         ps.setInt(2, book_id);
8         ps.executeUpdate();
9         ps.close();
10        conn.close();
11    }catch (SQLException e){
12        e.printStackTrace();
13        throw new MyException();
14    }
15 }
```

Cette méthode insère un enregistrement de réservation pour un utilisateur (user_id) et un livre (book_id) avec la date actuelle. En cas de succès, l'enregistrement est ajouté à la table "reservation" avec le statut "en_cours". En cas d'échec de connexion ou d'erreur SQL, une exception de type MyException est levée.

Méthode *EstReserver* :

```
● ● ●
1 public static Reservation estReserver(int book_id) throws MyException{
2     Reservation res=null;
3     try {
4         Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
5         String sql="Select id_reservation,id_utilisateur,id_livre,date_reservation,statut"+
6             "from reservation where id_livre=? and statut='En_Cours' ORDER BY date_reservation ASC LIMIT 1";
7         PreparedStatement ps = conn.prepareStatement(sql);
8         ps.setInt(1, book_id);
9         ResultSet rs=ps.executeQuery();
10
11        if(rs.next()) {
12            int res_id=rs.getInt("id_reservation");
13            int user_id=rs.getInt("id_utilisateur");
14            String date_reservation=rs.getString("date_reservation");
15            res=new Reservation(res_id,user_id,book_id,date_reservation,"En_Cours");
16        }
17        ps.close();
18        rs.close();
19        conn.close();
20    }catch(SQLException e) {
21        e.printStackTrace();
22        throw new MyException();
23    }
24    return res;
25 }
```

Cette méthode exécute une requête SQL pour récupérer la première réservation en cours pour le livre avec l'ID book_id. Si une réservation est trouvée, elle crée un objet Reservation avec les informations associées. En cas d'échec de connexion ou d'erreur SQL, une exception de type MyException est levée. La méthode retourne l'objet Reservation ou null s'il n'y a pas de réservation en cours.

Méthode ReserverPar :

```
● ● ●  
1  public static boolean reserverPar(int user_id ,int book_id) throws MyException {  
2      boolean resPar=false;  
3      try {  
4          Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");  
5          String sql="Select * from reservation where id_utilisateur=? and id_livre=? and statut='En_Cours' ";  
6          PreparedStatement ps = conn.prepareStatement(sql);  
7          ps.setInt(1, user_id);  
8          ps.setInt(2, book_id);  
9          ResultSet rs=ps.executeQuery();  
10         if(rs.next()) {  
11             resPar=true;;  
12         }  
13         ps.close();  
14         rs.close();  
15         conn.close();  
16     }catch(SQLException e) {  
17         e.printStackTrace();  
18         throw new MyException();  
19     }  
20     return resPar;  
21 }
```

La méthode `reserverPar` est conçue pour déterminer si un utilisateur (identifié par `user_id`) a déjà une réservation en cours pour un livre spécifique (identifié par `book_id`). Cette fonctionnalité est réalisée par l'exécution d'une requête SQL dans la base de données afin de rechercher un enregistrement correspondant dans la table des réservations. La valeur de retour `resPar` est définie sur `true` si une réservation en cours est trouvée pour cet utilisateur et ce livre, sinon elle reste à `false`. Cette méthode est utilisée avant de permettre à un utilisateur de créer une nouvelle réservation pour un livre.

Méthode AnnulerReservation:

```
● ● ●
1  public void annulerReservation() throws MyException {
2      try {
3          Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/biblio","root","112713");
4          String sql="delete from reservation where id_utilisateur=? and id_livre=? and statut='En_Cours'";
5          PreparedStatement ps = conn.prepareStatement(sql);
6          ps.setInt(1, user_id);
7          ps.setInt(2, book_id);
8          ps.executeUpdate();
9          ps.close();
10         conn.close();
11     }catch(SQLException e) {
12         e.printStackTrace();
13         throw new MyException();
14     }
15 }
```

Cette méthode annule une réservation en cours pour un utilisateur spécifique (`user_id`) et un livre spécifique (`book_id`). En cas d'échec de connexion ou d'erreur SQL, une exception de type `MyException` est levée.

2) Les Exceptions :

Classe MyException :

```
● ● ●  
1 public class MyException extends Exception{  
2     public MyException() {  
3         super("Ooops !Une erreur est survenue. Veuillez réessayer ultérieurement.");  
4     }  
5 }
```

La classe MyException offre une exception générique sans message spécifique, mais elle fournit un message par défaut indiquant qu'une erreur générale est survenue "ooops ! Une erreur est survenue. Veuillez réessayer ultérieurement."

Classe MailException :

```
● ● ●  
1 public class MailException extends Exception {  
2  
3     public MailException(String msg) {  
4         super(msg);  
5     }  
6 }
```

la classe MailException est conçue pour traiter des erreurs spécifiques liées aux opérations sur les adresses mail. Elle possède un constructeur qui prend un message explicatif en paramètre et le transmet à la classe parente (Exception) pour une gestion appropriée.

Classe PwdException :

```
● ● ●  
1 public class PwdException extends Exception {  
2     public PwdException(String msg) {  
3         super(msg);  
4     }  
5  
6 }
```

La classe PwdException est utilisée pour signaler des erreurs liées aux mots de passe. Elle possède un constructeur qui prend un message explicatif en paramètre et le transmet à la classe parente (Exception) pour une gestion appropriée.

3) Les Interfaces :

Swing est une bibliothèque graphique en Java utilisée pour créer des interfaces graphiques utilisateur (GUI). Elle fait partie du package javax.swing et offre des composants graphiques, des gestionnaires de disposition, des événements, et d'autres fonctionnalités pour développer des applications GUI

Authentification :

```
1  public class LogIn{
2      public LogIn() {
3          JFrame frame = new JFrame("S'authentifier");
4          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5          frame.setSize(800, 400);
6          frame.getContentPane().setBackground(Color.PINK);
7
8          JLabel welcomeLabel = new JLabel("Bienvenue dans notre bibliothèque");
9          welcomeLabel.setFont(welcomeLabel.getFont().deriveFont(30.0f));
10         welcomeLabel.setForeground(Color.BLUE);
11
12         JLabel mailLabel = new JLabel("Email:");
13         JTextField mailField = new JTextField(20);
14         JLabel pwdLabel = new JLabel("Mot de Passe:");
15         JPasswordField pwdField = new JPasswordField(20);
16         pwdField.setEchoChar('*');
17         JButton loginButton = new JButton("Se Connecter");
18         JButton SigninButton = new JButton("S'inscrire");
19
20         loginButton.addActionListener(new ActionListener() {
21             @Override
22             public void actionPerformed(ActionEvent e) {
23                 String mail = mailField.getText().trim();
24                 char[] password = pwdField.getPassword();
25                 String pwd = new String(password);
26                 if(! mail.isEmpty() & ! pwd.isEmpty()) {
27                     try {
28                         Utilisateur user=Utilisateur.authentifier(mail, pwd);
29                         if(user==null) {
30                             JOptionPane.showMessageDialog(frame, "E-mail ou mot de passe incorrect.");
31                         } else {
32                             if(! user.GetRole().equals("biblio")) {
33                                 frame.dispose();
34                                 new UserAccueil(user.GetId());
35                             }else {
36                                 frame.dispose();
37                                 new BiblioAccueil();
38                             }
39                         }
40                     }catch (MyException ex) {
41                         JOptionPane.showMessageDialog(frame, ex.getMessage(), "Erreur d'authentification", JOptionPane.ERROR_MESSAGE);
42                     }
43                 }else {
44                     JOptionPane.showMessageDialog(frame, "Veuillez remplir tous les champs.", "Champs Incomplets", JOptionPane.WARNING_MESSAGE);
45                 }
46             }
47         });
48     });
49 }
```



```
1      SignInButton.addActionListener(new ActionListener() {
2          @Override
3              public void actionPerformed(ActionEvent e) {
4                  frame.dispose();
5                  new Inscription();
6              }
7          });
8
9      frame.setLayout(new GridBagLayout());
10     GridBagConstraints gbc = new GridBagConstraints();
11     gbc.insets = new Insets(15, 20, 15, 20);
12
13     gbc.gridx = 0;
14     gbc.gridy = 0;
15     gbc.gridwidth = 2;
16     gbc.anchor = GridBagConstraints.PAGE_START;
17     frame.add(welcomeLabel, gbc);
18
19     gbc.gridwidth = 1;
20     gbc.anchor = GridBagConstraints.CENTER;
21
22     gbc.gridx = 0;
23     gbc.gridy = 1;
24     frame.add(mailLabel, gbc);
25     mailLabel.setPreferredSize(new Dimension(40, 30));
26
27     gbc.gridx = 1;
28     gbc.gridy = 1;
29     frame.add(mailField, gbc);
30     mailField.setPreferredSize(new Dimension(200, 30));
31
32     gbc.gridx = 0;
33     gbc.gridy = 2;
34     frame.add(pwdLabel, gbc);
35     pwdLabel.setPreferredSize(new Dimension(130, 30));
36
37     gbc.gridx = 1;
38     gbc.gridy = 2;
39     frame.add(pwdField, gbc);
40     pwdField.setPreferredSize(new Dimension(200, 30));
41
42     gbc.gridx = 0;
43     gbc.gridy = 3;
44     frame.add(SignInButton, gbc);
45     SignInButton.setPreferredSize(new Dimension(222, 30));
46
47     gbc.gridx = 1;
48     gbc.gridy = 3;
49     frame.add(loginButton, gbc);
50     loginButton.setPreferredSize(new Dimension(222, 30));
51
52     frame.setLocationRelativeTo(null);
53     frame.setLocationRelativeTo(null);
54     frame.setVisible(true);
55 }
56 }
```

La classe LogIn définit une fenêtre de connexion avec des champs pour l'e-mail et le mot de passe, ainsi que des boutons pour se connecter ou s'inscrire. Les composants tels que les labels, les champs de texte et les boutons sont configurés dans la fenêtre, et des gestionnaires d'événements sont attachés aux boutons pour gérer les actions de l'utilisateur. La disposition des composants est réalisée avec le gestionnaire GridBagLayout .

Inscription :

```
● ● ●  
1  public class Inscription {  
2      public Inscription() {  
3          JFrame frame = new JFrame("Inscription");  
4          frame.setSize(800, 400);  
5          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
6          frame.getContentPane().setBackground(Color.PINK);  
7  
8          JLabel nom = new JLabel("nom");  
9          JLabel prenom = new JLabel("prenom:");  
10         JLabel login= new JLabel("login");  
11         JLabel pwd = new JLabel("pwd");  
12  
13         JTextField nomfield = new JTextField(15);  
14         JTextField prenomfield = new JTextField(15);  
15         JTextField loginfield= new JTextField(20);  
16         JTextField pwdfield = new JTextField(20);  
17  
18         JButton inscrire =new JButton("S'inscrire");  
19  
20         frame.setLayout(new GridLayout(6, 2));  
21  
22         JRadioButton etudiantRadioButton = new JRadioButton("Etudiant");  
23         JRadioButton enseignantRadioButton = new JRadioButton("Enseignant");  
24  
25         ButtonGroup buttonGroup = new ButtonGroup();  
26         buttonGroup.add(etudiantRadioButton);  
27         buttonGroup.add(enseignantRadioButton);  
28  
29         final JRadioButton[] roleSelected = {null};  
30  
31         ActionListener radioButtonListener = new ActionListener() {  
32             @Override  
33             public void actionPerformed(ActionEvent e) {  
34                 if (etudiantRadioButton.isSelected()) {  
35                     roleSelected[0] = etudiantRadioButton;  
36                 } else if (enseignantRadioButton.isSelected()) {  
37                     roleSelected[0] = enseignantRadioButton;  
38                 }  
39             }  
40         };  
41         etudiantRadioButton.addActionListener(radioButtonListener);  
42         enseignantRadioButton.addActionListener(radioButtonListener);
```

```
1     inscrire.addActionListener(new ActionListener() {
2         @Override
3         public void actionPerformed(ActionEvent e) {
4             String nom=nomField.getText().trim();
5             String prenom=prenomField.getText().trim();
6             String login=loginField.getText().trim();
7             String pwd=pwdField.getText();
8             String role="";
9             if(!(nom.isEmpty() | prenom.isEmpty() | login.isEmpty() | pwd.isEmpty() | roleSelected == null)) {
10                 role=roleSelected[0].getText();
11                 Utilisateur user=new Utilisateur(nom,prenom,login,pwd,role);
12                 try {
13                     user.inscrire();
14                     JOptionPane.showMessageDialog(frame, "Nous avons bien recus votre inscription ,vous pouvez dès maintenant vous connecter !");
15                     frame.dispose();
16                     new LogIn();
17                 }catch (MailException | MyException | PwdException m ){
18                     JOptionPane.showMessageDialog(frame, m.getMessage(),"Erreur d'Inscription",JOptionPane.ERROR_MESSAGE);
19                 }
20             }else {
21                 JOptionPane.showMessageDialog(frame, "Veuillez remplir tous les champs.", "Champs Incomplets", JOptionPane.WARNING_MESSAGE);
22             }
23         }
24     );
25     JButton retour =new JButton("Retour");
26     retour.addActionListener(new ActionListener() {
27         @Override
28         public void actionPerformed(ActionEvent e) {
29             frame.dispose();
30             new LogIn();
31         }
32     });
33     frame.add(nom);
34     frame.add(nomField );
35     frame.add(prenom);
36     frame.add(prenomField );
37     frame.add(login);
38     frame.add(loginField );
39     frame.add(pwd);
40     frame.add(pwdField);
41     frame.add(etudiantRadioButton);
42     frame.add(enseignantRadioButton);
43     frame.add(retour);
44     frame.add(inscrire);
45     frame.setLocationRelativeTo(null);
46     frame.setVisible(true);
47 }
48 }
```

User_Accueil:

```
1 public class UserAccueil {
2     public UserAccueil(int user_id) {
3         JFrame frame = new JFrame("Accueil");
4         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5         frame.setSize(800, 400);
6         JButton chercher_Button = new JButton("Rechercher un livre");
7         JButton consult_livre_Button = new JButton("Consulter le Catalogue des Livres");
8         JButton hist_Button = new JButton("Consulter l'historique des empruntes");
9
10        frame.setLayout(new GridLayout(3,1));
11
12        consult_livre_Button.addActionListener(new ActionListener() {
13            @Override
14            public void actionPerformed(ActionEvent e) {
15                frame.dispose();
16                new ConsulterCatalogue(user_id);
17            }
18        });
19        chercher_Button.addActionListener(new ActionListener() {
20            @Override
21            public void actionPerformed(ActionEvent e) {
22                frame.dispose();
23                new ChercherLivre(user_id);
24            }
25        });
26        hist_Button.addActionListener(new ActionListener() {
27            @Override
28            public void actionPerformed(ActionEvent e) {
29                frame.dispose();
30                new ConsulterHistoEmprunt(user_id);
31            }
32        });
33        frame.add(chercher_Button);
34        frame.add(consult_livre_Button);
35        frame.add(hist_Button);
36        frame.setLocationRelativeTo(null);
37        frame.setVisible(true);
38    }
39 }
```

Biblio_Accueil:

```
1  public class BiblioAccueil {
2      public BiblioAccueil(){
3
4          JFrame frame = new JFrame("Accueil");
5          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6          frame.setSize(800, 400);
7
8          JButton ajouter_livre = new JButton("Ajouter un Livre ");
9          JButton consulter_livre = new JButton("Consulter le Catalogue des livres");
10         JButton supprimer_et_en= new JButton("Consulter le Catalogue des Adhérents");
11         JButton statistiques_etud_ens = new JButton("Statistiques sur les Adhérents les plus assidus");
12         JButton rappelle = new JButton("Envoyer des Rappels de Retour");
13         JButton Statistiques_livre = new JButton("Statistiques sur les livres les plus empruntés");
14
15         frame.setLayout(new GridLayout(3,2));
16
17         consulter_livre.addActionListener(new ActionListener() {
18             @Override
19             public void actionPerformed(ActionEvent e) {
20                 frame.dispose();
21                 new ConsulterLivresBiblio();
22             }
23         });
24         supprimer_et_en.addActionListener(new ActionListener() {
25             @Override
26             public void actionPerformed(ActionEvent e) {
27                 frame.dispose();
28                 new ConsulterAdherentsBiblio();
29             }
30         });
31         statistiques_etud_ens.addActionListener(new ActionListener() {
32             @Override
33             public void actionPerformed(ActionEvent e) {
34                 frame.dispose();
35                 new RapportStatAdherents();
36             }
37         });
38     }
```



```
1      Statistiques_livre.addActionListener(new ActionListener() {
2          @Override
3          public void actionPerformed(ActionEvent e) {
4              frame.dispose();
5              new RapportStatLivres();
6          }
7      );
8      rappelle.addActionListener(new ActionListener() {
9          @Override
10         public void actionPerformed(ActionEvent e) {
11             frame.dispose();
12             new Rappelle();
13         }
14     );
15     ajouter_livre.addActionListener(new ActionListener() {
16         @Override
17         public void actionPerformed(ActionEvent e) {
18             frame.dispose();
19             new AjouterLivre();
20         }
21     );
22     frame.add(supprimer_et_en);
23     frame.add(consuler_livre);
24     frame.add(statistiques_etud_ens);
25     frame.add(Statistiques_livre);
26     frame.add(ajouter_livre);
27     frame.add(rappelle);
28     frame.setLocationRelativeTo(null);
29     frame.setVisible(true);
30 }
31 }
```

Rechercher_livre :

```
● ● ●
```

```
1 public class ChercherLivre {
2     public ChercherLivre(int user_id) {
3         JFrame frame = new JFrame("Rechercher un Livre");
4         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5         frame.getContentPane().setBackground(Color.PINK);
6
7         frame.setSize(800, 400);
8         JLabel titre = new JLabel("Nom de Livre :");
9         JTextField titreField = new JTextField(10);
10        JLabel auteur = new JLabel("Auteur de Livre :");
11        JTextField auteurField = new JTextField(10);
12
13        JButton chercher = new JButton("Rechercher");
14        JButton retourButton = new JButton("Retour a L'acceuil");
15        frame.setLayout(new GridLayout(6,2));
16
17        chercher.addActionListener(new ActionListener() {
18            @Override
19            public void actionPerformed(ActionEvent e) {
20                String title = titreField.getText().trim();
21                String auteur = auteurField.getText().trim();
22                if(! title.isEmpty() & ! auteur.isEmpty()) {
23                    try {
24                        Livre livre=Livre.RechercherLivre(title, auteur);
25                        if(livre==null) {
26                            JOptionPane.showMessageDialog(frame, "Livre inexistant", "Erreur", JOptionPane.ERROR_MESSAGE);
27                        } else {
28                            frame.dispose();
29                            new BookDetails(user_id,livre);
30                        }
31                    }catch (MyException ex) {
32                        JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
33                    }
34                }else {
35                    JOptionPane.showMessageDialog(frame, "Veuillez remplir tous les champs.", "Champs Incomplets", JOptionPane.WARNING_MESSAGE);
36                }
37            }
38        });
39    });
40}
```



```
1     retourButton.addActionListener(new ActionListener() {
2         @Override
3         public void actionPerformed(ActionEvent e) {
4             frame.dispose();
5             new UserAccueil(user_id);
6         }
7     );
8     frame.add( new JLabel());
9     frame.add( new JLabel());
10    frame.add(titre);
11    frame.add(titreField);
12    frame.add( new JLabel());
13    frame.add( new JLabel());
14    frame.add(auteur);
15    frame.add(auteurField);
16    frame.add( new JLabel());
17    frame.add( new JLabel());
18    frame.add(retourButton);
19    frame.add(chercher);
20    frame.setLocationRelativeTo(null);
21    frame.setVisible(true);
22}
23}
```

Consulter_catalogue :

```
● ● ●
```

```
1  public class ConsulterCatalogue {
2      private static int selectedRow = -1;
3      public ConsulterCatalogue(int user_id) {
4          JFrame frame = new JFrame("Consulter le Catalogue des Livres");
5          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6
7          DefaultTableModel model = new DefaultTableModel() {
8              @Override
9              public boolean isCellEditable(int row, int column) {
10                  return false;
11              }
12          };
13          model.addColumn("Titre");
14          model.addColumn("Auteur");
15
16          try {
17              List<Livre> listeLivres = Livre.getListeLivres();
18
19              for (Livre livre : listeLivres) {
20                  model.addRow(new Object[]{livre.GetTitre(), livre.GetAuteur()});
21              }
22
23          } catch (MyException e) {
24              JOptionPane.showMessageDialog(frame, e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
25          }
26          JTable table = new JTable(model);
27
28          JScrollPane scrollPane = new JScrollPane(table);
29
30          ListSelectionModel selectionModel = table.getSelectionModel();
31          selectionModel.addListSelectionListener(new ListSelectionListener() {
32              @Override
33              public void valueChanged(ListSelectionEvent e) {
34                  if (!e.getValueIsAdjusting()) {
35                      selectedRow = table.getSelectedRow();
36                  }
37              }
38          });
39      }
40  }
```

```
1 JButton afficher_Button = new JButton("Afficher les détails de ce livre");
2 afficher_Button.setPreferredSize(new Dimension(210, 30));
3 afficher_Button.addActionListener(new ActionListener() {
4     @Override
5     public void actionPerformed(ActionEvent e) {
6         if (selectedRow != -1) {
7             String titre = (String) table.getValueAt(selectedRow, 0);
8             String auteur = (String) table.getValueAt(selectedRow, 1);
9             try {
10                 Livre livre=Livre.RechercherLivre(titre, auteur);
11                 frame.dispose();
12                 new BookDetails(user_id,livre);
13                 selectedRow = -1;
14             }catch(MyException ex) {
15                 JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
16             }
17         } else {
18             JOptionPane.showMessageDialog(frame, "Veuillez sélectionner une ligne avant de cliquer sur le bouton.");
19         }
20     }
21 });
```

```
1 JButton retour =new JButton("Retour a L'accueil");
2 retour.setPreferredSize(new Dimension(210, 30));
3 retour.addActionListener(new ActionListener() {
4     @Override
5     public void actionPerformed(ActionEvent e) {
6         frame.dispose();
7         new UserAccueil(user_id);
8     }
9 });
10 JPanel buttonPanel = new JPanel();
11 buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.LINE_AXIS));
12 buttonPanel.add(retour);
13 buttonPanel.add(Box.createHorizontalGlue());
14 buttonPanel.add(afficher_Button);
15
16 frame.getContentPane().setLayout(new BorderLayout());
17 frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
18 frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
19 frame.setSize(800, 400);
20 frame.setLocationRelativeTo(null);
21 frame.setVisible(true);
22 }
23 }
```

Lorsqu'un utilisateur sélectionne une ligne dans la table, l'index de la ligne sélectionnée est enregistré dans la variable selectedRow. Deux boutons sont présents en bas de la fenêtre : "Afficher les détails de ce livre" et "Retour à l'accueil".

Le bouton "Afficher les détails de ce livre" est associé à un gestionnaire d'événements qui récupère les informations du livre (titre et auteur) de la ligne sélectionnée, puis ferme la fenêtre actuelle pour ouvrir une nouvelle fenêtre affichant les détails de ce livre.

Consulter_Historique :

```
1 public class ConsulterHistoEmprunt {
2     private static int selectedRow = -1;
3     public ConsulterHistoEmprunt(int user_id) {
4         JFrame frame = new JFrame("Consulter Historique des Emprunts");
5         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6
7         DefaultTableModel model = new DefaultTableModel() {
8             @Override
9             public boolean isCellEditable(int row, int column) {
10                 return false;
11             }
12         };
13         model.addColumn("ID");
14         model.addColumn("Titre");
15         model.addColumn("Auteur");
16         model.addColumn("date Emprunte");
17         model.addColumn("date Retour");
18         model.addColumn("statut Actuel");
19
20         try {
21             List listeHistoriqueEmpDTO= Emprunt.getListeHistEmp(user_id);
22             for (HistoriqueEmpDTO object : listeHistoriqueEmpDTO) {
23                 model.addRow(new Object[]{object.getId_emprunt(),object.getTitre(),object.getAuteur(),object.getDate_emprunt(),object.getDate_retour(),object.getStatut()});
24             }
25         } catch (MyException ex) {
26             JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
27         }
28
29         JTable table = new JTable(model);
30
31         JScrollPane scrollPane = new JScrollPane(table);
32
33         ListSelectionModel selectionModel = table.getSelectionModel();
34         selectionModel.addListSelectionListener(new ListSelectionListener() {
35             @Override
36             public void valueChanged(ListSelectionEvent e) {
37                 if (!e.getValueIsAdjusting()) {
38                     selectedRow = table.getSelectedRow();
39                 }
40             }
41         });
42     }
43 }
```

```

1     JButton retour = new JButton("Retour a L'Accueil");
2     retour.addActionListener(new ActionListener() {
3         @Override
4         public void actionPerformed(ActionEvent e) {
5             frame.dispose();
6             new UserAccueil(user_id);
7         }
8     });
9
10    JPanel buttonPanel = new JPanel();
11    buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.LINE_AXIS));
12    buttonPanel.add(retour);
13    buttonPanel.add(Box.createHorizontalGlue());
14    buttonPanel.add(retourner_Button);
15
16    frame.getContentPane().setLayout(new BorderLayout());
17    frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
18    frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
19
20    frame.setSize(800, 400);
21    frame.setLocationRelativeTo(null);
22    frame.setVisible(true);
23}
24

```

```

1 JButton retourner_Button = new JButton("Retourner ce livre");
2 retourner_Button.addActionListener(new ActionListener() {
3     @Override
4     public void actionPerformed(ActionEvent e) {
5         if (selectedRow != -1) {
6             String statut = (String) table.getValueAt(selectedRow, 5);
7             if(statut.equals("Terminé")) {
8                 JOptionPane.showMessageDialog(frame, "Livre déjà Rotourné !");
9             }else {
10                 int choix = JOptionPane.showConfirmDialog(frame, "Êtes-vous sûr de vouloir retourner ce livre ?", "Confirmation de retour de livre", JOptionPane.YES_NO_OPTION);
11                 if (choix == JOptionPane.YES_OPTION) {
12                     int id_emprunt =(int) table.getValueAt(selectedRow, 0);
13                     String titre = (String) table.getValueAt(selectedRow, 1);
14                     String auteur = (String) table.getValueAt(selectedRow, 2);
15                     try {
16                         Livre livre=Livre.RechercherLivre(titre, auteur);
17                         Emprunt emp=new Emprunt(id_emprunt,user_id,livre.getId());
18                         emp.retournerLivre();
19                         JOptionPane.showMessageDialog(frame, "Retour de livre réussi !", "Succès du retour", JOptionPane.INFORMATION_MESSAGE);
20                         frame.dispose();
21                         new ConsulterEmprunt(user_id);
22                         selectedRow = -1;
23                     }catch (MyException ex) {
24                         JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
25                     }
26                 }
27             }
28         } else {
29             JOptionPane.showMessageDialog(frame, "Veuillez sélectionner une ligne avant de cliquer sur le bouton.");
30         }
31     });

```

Book_Details :

```
1  public class BookDetails {  
2      public BookDetails(int user_id,Livre livre) {  
3          JFrame frame = new JFrame("Afficher les Details d'un Livre");  
4          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
5          frame.setSize(800, 400);  
6          frame.getContentPane().setBackground(Color.PINK);  
7  
8          JLabel label1 = new JLabel("Titre:");  
9          JLabel label2 = new JLabel("Auteur:");  
10         JLabel label3 = new JLabel("Genre:");  
11         JLabel label4 = new JLabel("Disponibilité:");  
12  
13         JTextField textField1 = new JTextField();  
14         JTextField textField2 = new JTextField();  
15         JTextField textField3 = new JTextField();  
16         JTextField textField4 = new JTextField();  
17  
18         frame.setLayout(new GridLayout(6,4));  
19  
20         frame.add(label1);  
21         frame.add(textField1);  
22         frame.add(label2);  
23         frame.add(textField2);  
24         frame.add(label3);  
25         frame.add(textField3);  
26         frame.add(label4);  
27         frame.add(textField4);  
28         JButton emprunt= new JButton("Emprunter");  
29         JButton reserver = new JButton("Reserver");  
30         JButton retour=new JButton("Catalogue");  
31         JButton annuler=new JButton("Annuler Reservation");  
32  
33         textField1.setEditable(false);  
34         textField2.setEditable(false);  
35         textField3.setEditable(false);  
36         textField4.setEditable(false);  
37  
38         textField1.setText(livre.GetTitre());  
39         textField2.setText(livre.GetAuteur());  
40         textField3.setText(livre.GetGenre());  
41         textField4.setText(livre.GetDispo());
```

```

1  if(livre.GetDispo().equals("Disponible")) {
2      frame.add(new JLabel());
3      frame.add(new JLabel());
4      frame.add(retour);
5      frame.add(emprunt);
6  }else {
7      try {
8          if(Emprunt.emprunterPar(user_id,livre.getId())) {
9              frame.add(new JLabel());
10         frame.add(new JLabel());
11         frame.add(retour);
12         frame.add(new JLabel("Vous êtes l'emprunteur de ce livre."));
13     }else if(Reservation.reserverPar(user_id,livre.getId())) {
14         frame.add(new JLabel());
15         frame.add(new JLabel(" Votre demande de réservation est en cours"));
16         frame.add(retour);
17         frame.add(annuler);
18     }else {
19         frame.add(new JLabel());
20         frame.add(new JLabel());
21         frame.add(retour);
22         frame.add(reserver);
23     }
24 }catch (MyException ex) {
25     JOptionPane.showMessageDialog(frame, ex.getMessage());
26 }
27 }
28 emprunt.addActionListener(new ActionListener() {
29     @Override
30     public void actionPerformed(ActionEvent e) {
31         Emprunt emp=new Emprunt(user_id,livre.getId());
32         int choix = JOptionPane.showConfirmDialog(frame, "Êtes-vous sûr de vouloir emprunter ce livre ?", "Demande de Confirmation d'emprunt de livre", JOptionPane.YES_NO_OPTION);
33         if (choix == JOptionPane.YES_OPTION) {
34             try {
35                 emp.emprunterLivre();
36                 JOptionPane.showMessageDialog(frame, "Emprunt réussi !", "Confirmation d'emprunt de livre", JOptionPane.INFORMATION_MESSAGE);
37                 frame.dispose();
38                 livre.SetDispo("Indisponible");//khater kif livre.dispo=disponible wa9t naawd nabaath el livre yab9a dispo
39                 new BookDetails(user_id,livre);
40             }catch (MyException ex){
41                 JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
42             }
43         }
44     }
45 });

```

```

1  reserver.addActionListener(new ActionListener() {
2     @Override
3     public void actionPerformed(ActionEvent e) {
4         int confirmation = JOptionPane.showConfirmDialog(frame, "Voulez-vous réserver ce livre ?", "Confirmation de réservation de livre", JOptionPane.YES_NO_OPTION);
5
6         if (confirmation == JOptionPane.YES_OPTION) {
7             Reservation res=new Reservation(user_id,livre.getId());
8
9             try {
10                 res.reserverLivre();
11                 JOptionPane.showMessageDialog(frame, "Réservation réussie !", "Succès de la réservation", JOptionPane.INFORMATION_MESSAGE);
12                 frame.dispose();
13                 new BookDetails(user_id,livre);
14
15             }catch (MyException ex) {
16                 JOptionPane.showMessageDialog(frame, ex.getMessage());
17             }
18         }
19     }
20 });
21 annuler.addActionListener(new ActionListener() {
22     @Override
23     public void actionPerformed(ActionEvent e) {
24         int confirmation = JOptionPane.showConfirmDialog(frame, "Voulez-vous annuler cette réservation ?", "Confirmation d'annulation de réservation", JOptionPane.YES_NO_OPTION);
25
26         if (confirmation == JOptionPane.YES_OPTION) {
27             Reservation res=new Reservation(user_id,livre.getId());
28             try {
29                 res.annulerReservation();
30                 JOptionPane.showMessageDialog(frame, "Annulation de réservation réussie !", "Succès de l'annulation", JOptionPane.INFORMATION_MESSAGE);
31                 frame.dispose();
32                 new BookDetails(user_id,livre);
33
34             }catch (MyException ex) {
35                 JOptionPane.showMessageDialog(frame, ex.getMessage());
36             }
37         }
38     }
39 });
40 retour.addActionListener(new ActionListener() {
41     @Override
42     public void actionPerformed(ActionEvent e) {
43         frame.dispose();
44         new ConsulterCatalogue(user_id);
45     }
46 });
47
48 frame.setLocationRelativeTo(null);
49
50 frame.setVisible(true);
51 }

```

Ajouter_livre :

```
● ● ●
1 public class AjouterLivre {
2     public AjouterLivre(){
3         JFrame frame = new JFrame("Ajouter un livre");
4         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5         frame.getContentPane().setBackground(Color.PINK);
6
7         JLabel titre = new JLabel("Titre");
8         JLabel auteur = new JLabel("Auteur");
9         JLabel genre = new JLabel("Genre");
10
11        JTextField titrefield = new JTextField(5);
12        JTextField auteurfield = new JTextField(5);
13        JTextField genrefield = new JTextField(5);
14
15        frame.setLayout(new GridLayout(8, 2));
16
17        JButton ajouter = new JButton("Ajouter ce livre ");
18        ajouter.addActionListener(new ActionListener() {
19            @Override
20            public void actionPerformed(ActionEvent e) {
21                String titre=titrefield.getText().trim();
22                String auteur=auteurfield.getText().trim();
23                String genre=genrefield.getText().trim();
24                if((titre.isEmpty() | auteur.isEmpty() | genre.isEmpty())) {
25                    try {
26                        if(Livre.RechercherLivre(titre,auteur)!=null) {
27                            JOptionPane.showMessageDialog(frame, "Le livre est déjà présent dans la bibliothèque.", "Livre Existant", JOptionPane.INFORMATION_MESSAGE);
28                        } else {
29                            Livre livre=new Livre(titre,auteur,genre);
30                            livre.AjouterLivre();
31                            JOptionPane.showMessageDialog(frame, "Livre a été ajouté avec succès.", "Ajout Réussi", JOptionPane.INFORMATION_MESSAGE);
32                            titrefield.setText("");
33                            auteurfield.setText("");
34                            genrefield.setText("");
35                        }
36                    }catch (MyException ex) {
37                        JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur d'ajout",JOptionPane.ERROR_MESSAGE);
38                    }
39                }else {
40                    JOptionPane.showMessageDialog(frame, "Veuillez remplir tous les champs.", "Champs Incomplets", JOptionPane.WARNING_MESSAGE);
41                }
42            }
43        });
44    });
}
```



```
1      JButton retour = new JButton("Retour a L'Accueil");
2      retour.addActionListener(new ActionListener() {
3          @Override
4          public void actionPerformed(ActionEvent e) {
5              frame.dispose();
6              new BiblioAccueil();
7          }
8      );
9      frame.add(new JLabel());
10     frame.add(new JLabel());
11     frame.add(titre);
12     frame.add(titrefield);
13     frame.add(new JLabel());
14     frame.add(new JLabel());
15     frame.add(auteur);
16     frame.add(auteurfield);
17
18     frame.add(new JLabel());
19     frame.add(new JLabel());
20     frame.add(genre);
21     frame.add(genrefield);
22     frame.add(new JLabel());
23     frame.add(new JLabel());
24
25     frame.add(retour);
26     frame.add(ajouter);
27     frame.setSize(800, 400);
28     frame.setLocationRelativeTo(null);
29     frame.setVisible(true);
30 }
31 }
```

Consulter_Adheren(supprimer_utilisateur) :

```
● ● ●
```

```
1 public class ConsulterAdherentsBiblio {
2     private static int selectedRow = -1;
3     public ConsulterAdherentsBiblio() {
4         JFrame frame = new JFrame("Supprimer Utilisateur");
5         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6         frame.getContentPane().setBackground(Color.PINK);
7
8         DefaultTableModel model = new DefaultTableModel() {
9             @Override
10            public boolean isCellEditable(int row, int column) {
11                return false;
12            }
13        };
14        model.addColumn("Id");
15        model.addColumn("Nom");
16        model.addColumn("Prenom");
17        model.addColumn("Role");
18        try {
19            List<Utilisateur> listeUser = Utilisateur.getListeUtilisateurs();
20            for (Utilisateur user : listeUser) {
21                model.addRow(new Object[]{user.getId(), user.getNom(), user.getPrenom(), user.getRole()});
22            }
23        } catch (MyException ex) {
24            JOptionPane.showMessageDialog(frame, ex.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
25        }
26
27        JTable table = new JTable(model);
28
29        JScrollPane scrollPane = new JScrollPane(table);
30        ListSelectionModel selectionModel = table.getSelectionModel();
31        selectionModel.addListSelectionListener(new ListSelectionListener() {
32            @Override
33            public void valueChanged(ListSelectionEvent e) {
34                if (!e.getValueIsAdjusting()) {
35                    selectedRow = table.getSelectedRow();
36                }
37            }
38        });
39    }
```

```
● ● ●
```

```
1 JButton supprimer = new JButton("Supprimer cet Adhérent ");
2 supprimer.addActionListener(new ActionListener() {
3     @Override
4     public void actionPerformed(ActionEvent e) {
5         if (selectedRow != -1) {
6             int choix = JOptionPane.showConfirmDialog(frame, "Êtes-vous sûr de supprimer cet Adhérent ?", "Confirmation de suppression", JOptionPane.YES_NO_OPTION);
7             if (choix == JOptionPane.YES_OPTION) {
8                 try {
9                     int id = (int) table.getValueAt(selectedRow, 0);
10                    String nom=(String) table.getValueAt(selectedRow, 1);
11                    String prenom=(String) table.getValueAt(selectedRow, 2);
12                    String role=(String) table.getValueAt(selectedRow, 3);
13                    Utilisateur user=new Utilisateur(id,nom,prenom,role);
14                    user.supprimerUtilisateur();
15                    JOptionPane.showMessageDialog(null, "Suppression réussie avec succès !", "Suppression", JOptionPane.INFORMATION_MESSAGE);
16                    frame.dispose();
17                    new ConsulterAdherentsBiblio();
18                    selectedRow = -1;
19                }catch (MyException ex) {
20                    JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
21                }
22            }else {
23                JOptionPane.showMessageDialog(frame, "Veuillez sélectionner une ligne avant de cliquer sur le bouton.");
24            }
25        }
26    });
27 });
28 }
```

```
● ● ●
```

```
1 JButton retour = new JButton("Retour a L'Accueil");
2 frame.setLayout(new GridLayout(3,1));
3 retour.addActionListener(new ActionListener() {
4     @Override
5     public void actionPerformed(ActionEvent e) {
6         frame.dispose();
7         new BiblioAccueil();
8     }
9 });
10 JPanel buttonPanel = new JPanel();
11 buttonPanel.add(retour);
12 buttonPanel.add(supprimer);
13
14 frame.getContentPane().setLayout(new BorderLayout());
15 frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
16 frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
17
18 frame.setSize(800, 400);
19 frame.setLocationRelativeTo(null);
20 frame.setVisible(true);
21 }
22 }
23 }
```

Consulter_livre_biblio(supprimer_livre) :

```
● ● ●
```

```
1 public class ConsulterLivresBiblio {
2     private static int selectedRow = -1;
3     public ConsulterLivresBiblio() {
4         JFrame frame = new JFrame("Supprimer livres");
5         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6
7         DefaultTableModel model = new DefaultTableModel() {
8             @Override
9             public boolean isCellEditable(int row, int column) {
10                 return false;
11             }
12         };
13
14         model.addColumn("Titre");
15         model.addColumn("Auteur");
16
17         try {
18             List<Livre> listeLivres = Livre.getListeLivres();
19
20             for (Livre livre : listeLivres) {
21                 model.addRow(new Object[]{livre.GetTitre(), livre.GetAuteur()});
22             }
23         } catch (MyException e) {
24             JOptionPane.showMessageDialog(frame, e.getMessage());
25         }
26
27         JTable table = new JTable(model);
28
29         JScrollPane scrollPane = new JScrollPane(table);
30         ListSelectionModel selectionModel = table.getSelectionModel();
31         selectionModel.addListSelectionListener(new ListSelectionListener() {
32             @Override
33             public void valueChanged(ListSelectionEvent e) {
34                 if (!e.getValueIsAdjusting()) {
35                     selectedRow = table.getSelectedRow();
36                 }
37             }
38         });
39     }
40 }
```

```
● ● ●
```

```
1 JButton supprimer = new JButton("supprimer ce livre ");
2 supprimer.addActionListener(new ActionListener() {
3     @Override
4     public void actionPerformed(ActionEvent e) {
5         if (selectedRow != -1) {
6             int choix = JOptionPane.showConfirmDialog(frame, "Êtes-vous sûr de supprimer ce livre ?", "Confirmation de suppression", JOptionPane.YES_NO_OPTION);
7             if (choix == JOptionPane.YES_OPTION) {
8                 String titre = (String) table.getValueAt(selectedRow, 0);
9                 String auteur = (String) table.getValueAt(selectedRow, 1);
10                try {
11                    Livre livre=Livre.RechercherLivre(titre, auteur);
12                    livre.supprimerLivre();
13                    JOptionPane.showMessageDialog(frame, "Suppression réussie avec succès !", "Suppression", JOptionPane.INFORMATION_MESSAGE);
14                    frame.dispose();
15                    new ConsulterLivresBiblio();
16                    selectedRow = -1;
17                }catch (MyException ex) {
18                    JOptionPane.showMessageDialog(frame, ex.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
19                }
20            }
21        }else {
22            JOptionPane.showMessageDialog(frame, "Veuillez sélectionner une ligne avant de cliquer sur le bouton.");
23        }
24    });
25});
```

```
● ● ●
```

```
1 JButton retour = new JButton("Retour a L'Accueil ");
2 retour.addActionListener(new ActionListener() {
3     @Override
4     public void actionPerformed(ActionEvent e) {
5         frame.dispose();
6         new BiblioAccueil();
7     }
8 });
9 JPanel buttonPanel = new JPanel();
10 buttonPanel.add(retour);
11 buttonPanel.add(supprimer);
12
13 frame.getContentPane().setLayout(new BorderLayout());
14 frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
15 frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
16
17 frame.setSize(800, 400);
18 frame.setLocationRelativeTo(null);
19 frame.setVisible(true);
20 }
21 }
```

Rapport_Satistique_Livre :

```
● ● ●
1  public class RapportStatLivres {
2      public RapportStatLivres() {
3          JFrame frame = new JFrame("Statistiques sur les livres les plus empruntés");
4          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5          frame.setSize(400, 200);
6          DefaultTableModel model = new DefaultTableModel() {
7              @Override
8              public boolean isCellEditable(int row, int column) {
9                  return false;
10             }
11         };
12         model.addColumn("Titre");
13         model.addColumn("Auteur");
14         model.addColumn("Nombre d'Emprunts");
15
16         try {
17             List<RapportStatDTO> listeRapportStatDTO = Emprunt.getRapportLivre();
18             for (RapportStatDTO object : listeRapportStatDTO) {
19                 model.addRow(new Object[]{object.getChaine1(),object.getChaine2(),object.getNbEmp()});
20             }
21         } catch (MyException ex) {
22             JOptionPane.showMessageDialog(frame, ex.getMessage());
23         }
24
25         JTable table = new JTable(model);
26         JScrollPane scrollPane = new JScrollPane(table);
27
28         JButton retour = new JButton("Retour a L'Accueil");
29         retour.addActionListener(new ActionListener() {
30             @Override
31             public void actionPerformed(ActionEvent e) {
32                 frame.dispose();
33                 new BiblioAccueil();
34
35             }
36         });
37         JPanel buttonPanel = new JPanel();
38         buttonPanel.add(retour);
39
40         frame.getContentPane().setLayout(new BorderLayout());
41         frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
42         frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
43         frame.setSize(800, 400);
44         frame.setLocationRelativeTo(null);
45         frame.setVisible(true);
46     }
47 }
```

Rapport_Satistique_d'Adherents :

```
 1  public class RapportStatAdherents {
 2      public RapportStatAdherents() {
 3          JFrame frame = new JFrame("Statistiques sur les utilisateurs les plus assidus");
 4          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 5          frame.setSize(400, 200);
 6          DefaultTableModel model = new DefaultTableModel() {
 7              @Override
 8              public boolean isCellEditable(int row, int column) {
 9                  return false;
10              }
11          };
12          model.addColumn("Nom");
13          model.addColumn("Prenom");
14          model.addColumn("Nombre d'Emprunts");
15
16          try {
17              List<RapportStatDTO> listeRapportStatDTO = Emprunt.getRapportUtilisateur();
18              for (RapportStatDTO object : listeRapportStatDTO) {
19                  model.addRow(new Object[]{object.getChaine1(),object.getChaine2(),object.getNbEmp()});
20              }
21          } catch (MyException ex) {
22              JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur",JOptionPane.ERROR_MESSAGE);
23          }
24
25          JTable table = new JTable(model);
26
27          JScrollPane scrollPane = new JScrollPane(table);
28
29          JButton retour = new JButton("Retour a L'Accueil");
30
31          frame.setLayout(new GridLayout(3,1));
32          retour.addActionListener(new ActionListener() {
33              @Override
34              public void actionPerformed(ActionEvent e) {
35                  frame.dispose();
36                  new BiblioAccueil();
37
38              }
39          });
40          JPanel buttonPanel = new JPanel();
41          buttonPanel.add(retour);
42
43          frame.getContentPane().setLayout(new BorderLayout());
44          frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
45          frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
46          frame.setSize(800, 400);
47          frame.setLocationRelativeTo(null);
48          frame.setVisible(true);
49      }
50  }
```

Rappelle:

```
● ● ●
1 public class Rappelle {
2     public Rappelle(){
3         JFrame frame = new JFrame("rappelle");
4         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5         frame.setSize(800, 400);
6         DefaultTableModel model = new DefaultTableModel() {
7             @Override
8             public boolean isCellEditable(int row, int column) {
9                 return false;
10            }
11        };
12        model.addColumn("Nom");
13        model.addColumn("Prenom");
14        model.addColumn("Titre");
15        model.addColumn("Auteur");
16        model.addColumn("date_emprunte");
17        model.addColumn("date_retour");
18
19    try {
20        List<RappelDTO> listeRappelDTO = Emprunt.getListeRappelUtilisateurs();
21
22        for (RappelDTO object : listeRappelDTO) {
23            model.addRow(new Object[]{object.getNom(),object.getPrenom(),object.getTitre(),object.getAuteur(),object.getDate_emprunte(),object.getDate_retour()});
24        }
25    } catch (MyException ex) {
26        JOptionPane.showMessageDialog(frame, ex.getMessage(),"Erreur", JOptionPane.ERROR_MESSAGE);
27    }
28
29    JTable table = new JTable(model);
30    JScrollPane scrollPane = new JScrollPane(table);
31
32    JButton envoyer = new JButton("envoyer un mail ");
33    envoyer.addActionListener(new ActionListener() {
34        @Override
35        public void actionPerformed(ActionEvent e) {
36
37            int nombreDeLignes = model.getRowCount();
38            if(nombreDeLignes!=0) {
39                JOptionPane.showMessageDialog(frame, nombreDeLignes+" E-mails de rappel envoyés avec succès.", "Rappel par Mail", JOptionPane.INFORMATION_MESSAGE);
40            }
41        }
42    });
43
44    JButton retour = new JButton("Retour a L'Accueil");
45
46    frame.setLayout(new GridLayout(3,1));
47    retour.addActionListener(new ActionListener() {
48        @Override
49        public void actionPerformed(ActionEvent e) {
50            frame.dispose();
51            new BiblioAccueil();
52        }
53    });
54    JPanel buttonPanel = new JPanel();
55    buttonPanel.add(retour);
56    buttonPanel.add(envoyer);
57
58    frame.getContentPane().setLayout(new BorderLayout());
59    frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
60    frame.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
61    frame.setSize(800, 400);
62    frame.setLocationRelativeTo(null);
63    frame.setVisible(true);
64}
65}
66
```

4) Présentation Visuelle et Démonstrations :



Si Email ou Mot de passe ne sont pas valides :



Si Email et Mot de passe sont valides :

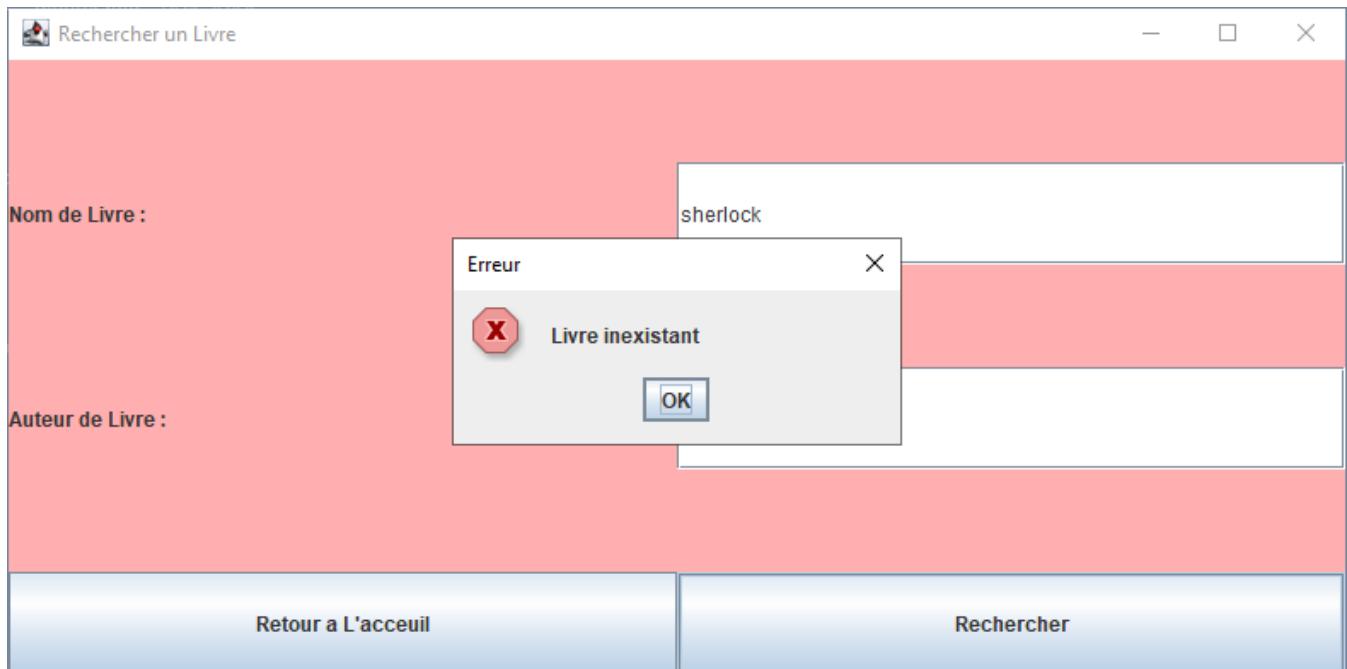
Si L'authentifiant est un étudiant ou un Enseignant :



Si L'adhèrent d'après la page d'accueil clique sur < Rechercher un livre >:



Si le livre n'existe pas :



Si utilisateur d'après la page d'accueil clique sur <Consulter catalogue> :

Consulter le Catalogue des Livres	
Titre	Auteur
Le Comte de Monte-Cristo	Alexandre Dumas
Crime et Châtiment	Fiodor Dostoïevski
Le Petit Prince	Antoine de Saint-Exupéry
Les Misérables	Victor Hugo
Guerre et Paix	Léon Tolstoï
La Trilogie de la Poussière: La Belle Sauvage	Philip Pullman
À la croisée des mondes: Les Royaumes du Nord	Philip Pullman
Le Miroir d'Ambre	Philip Pullman
la poussière La Communauté des esprits	Philip Pullman
Fondation	Isaac Asimov
Neuromancien	William Gibson
En attendant le vote des bêtes sauvages	Yôko Tawada
L'Ève future	Auguste de Villiers de l'Isle-Adam
livre	livre

Retour a L'accueil Afficher les détails de ce livre

Si l'utilisateur clique sur le bouton <rechercher> d'après l'interface 'rechercher un Livre' et le livre existe ou s'il sélectionne une ligne de catalogue et clique sur <afficher les détails de ce livre> d'après l'interface consulter catalogue des livres :

Si le livre n'est ni emprunté ni réservé :

Afficher les Details d'un Livre	
Titre:	Le Petit Prince
Auteur:	Antoine de Saint-Exupéry
Genre:	Conte philosophique
Disponibilité:	Disponible
Catalogue	Emprunter

Si le livre est déjà emprunté :

Afficher les Details d'un Livre	
Titre:	Le Petit Prince
Auteur:	Antoine de Saint-Exupéry
Genre:	Conte philosophique
Disponibilité:	Indisponible
Catalogue	Reserver

Si le livre est emprunte d'après adhérent actuellement connecter :



Afficher les Details d'un Livre

Titre: Le Petit Prince

Auteur: Antoine de Saint-Exupéry

Genre: Conte philosophique

Disponibilité: Indisponible

Catalogue Vous êtes l'emprunteur de ce livre.

Si le livre est déjà réservé d'après l'adhérent actuellement connecter :



Afficher les Details d'un Livre

Titre: Le Petit Prince

Auteur: Antoine de Saint-Exupéry

Genre: Conte philosophique

Disponibilité: Indisponible

Votre demande de reservation est en cours

Catalogue Annuler Reservation

Si l'Adhérent clique sur Button< consulter historique> d'après la page d'accueil :

Consulter Historique des Empruntes					
ID	Titre	Auteur	date Emprunte	date Retour	statut Actuel
32	Le Petit Prince	Antoine de Saint-Exupéry	2023-12-10	2023-12-25	En_Cours
36	Les Misérables	Victor Hugo	2023-12-10	2023-12-10	Terminé
37	L'Ève future	Auguste de Villiers de L'Isle-Adam	2023-12-10	2023-12-25	En_Cours

[Retour a L'Accueil](#) [Retourner ce livre](#)

Si l'Adhérent clique sur Button< Retourner ce livre>:

Si le livre est déjà retourné:

Consulter Historique des Empruntes					
ID	Titre	Auteur	date Emprunte	date Retour	statut Actuel
7	L'Alchimiste	Paulo Coelho	2023-12-10	2023-12-10	Terminé
8	L'Écume des jours	Boris Vian	2023-12-10	2023-12-10	Terminé
9	Le Maître et Marguerite	Mikhail Boulgakov	2023-12-10	2023-12-10	Terminé
18	Les Trois Mousquetaires	Alexandre Dumas	2023-12-10	2023-12-11	En_Cours
22	Moby Dick	Herman Melville	2023-12-10	2023-12-10	Terminé
24	Le Rouge et le Noir	Stendhal	2023-12-10	2023-12-25	En_Cours

Message X

Livre déjà Retourné !

[OK](#)

[Retour a L'Accueil](#) [Retourner ce livre](#)

Si le livre n'est pas encore retourné:

The screenshot shows a web-based library management system. At the top, there is a header bar with a logo and the text "Consulter Historique des Empruntes". Below this is a table listing six books with their details: ID, Title, Author, Borrow Date, Return Date, and Current Status. The books listed are: L'Alchimiste by Paulo Coelho (status Terminé), L'Écume des jours by Boris Vian (status Terminé), Le Maître et Marguerite by Mikhaïl Boulgakov (status Terminé), Les Trois Mousqueta... by Alexandre Dumas (status En_Cours), Moby Dick by Herman Melville (status Terminé), and Le Rouge et le Noir by Stendhal (status En_Cours). A modal dialog box titled "Succès du retour" is displayed in the center, containing the message "Retour de livre réussi!" (Book return successful!) with an "OK" button. At the bottom of the page are two buttons: "Retour a L'Accueil" on the left and "Retourner ce livre" on the right.

ID	Titre	Auteur	date Emprunte	date Retour	statut Actuel
7	L'Alchimiste	Paulo Coelho	2023-12-10	2023-12-10	Terminé
8	L'Écume des jours	Boris Vian	2023-12-10	2023-12-10	Terminé
9	Le Maître et Marguerite	Mikhaïl Boulgakov	2023-12-10	2023-12-10	Terminé
18	Les Trois Mousqueta...	Alexandre Dumas	2023-12-10	2023-12-11	En_Cours
22	Moby Dick	Herman Melville	2023-12-10	2023-12-10	Terminé
24	Le Rouge et le Noir	Stendhal	2023-12-10	2023-12-25	En_Cours

Si Utilisateur est un Bibliothécaire :

The screenshot shows a dashboard for library staff. At the top, there is a header bar with a logo and the text "Accueil". Below this are four main functional buttons arranged in a 2x2 grid: "Consulter le Catalogue des Adhérents" (Consult Member Catalogue) in the top-left, "Consulter le Catalogue des livres" (Consult Book Catalogue) in the top-right, "Statistiques sur les Adhérents les plus assidus" (Statistics on the most frequent members) in the bottom-left, and "Statistiques sur les livres les plus empruntés" (Statistics on the most borrowed books) in the bottom-right. In the bottom center, there are two additional buttons: "Ajouter un Livre" (Add a Book) and "Envoyer des Rappels de Retour" (Send Return Reminders).

Si le bibliothécaire clique sur <consulter catalogue des adhérents> :

The screenshot shows a window titled "Supprimer Utilisateur". It contains a table with columns: Id, Nom, Prenom, and Role. The data is as follows:

Id	Nom	Prenom	Role
2	ranim	ranim	Etudiant
3	rihem	rihem	Etudiant
4	karim	karim	Enseignant
6	nimou	nimou	Enseignant
14	ALI	Ali	Etudiant
16	aziz	aziz	Enseignant
17	amal	amal	Enseignant

At the bottom are two buttons: "Retour a L'Accueil" and "Supprimer cet Adhérent".

Si le bibliothécaire clique sur <supprimer cet Adhérent>:

The screenshot shows a window titled "Supprimer Utilisateur". It contains a table with columns: Id, Nom, Prenom, and Role. The data is as follows:

Id	Nom	Prenom	Role
2	ranim	ranim	Etudiant
3	rihem	rihem	Etudiant
4	amal	amal	Enseignant
5	ALI	Ali	Etudiant
6	rayen	rayen	Etudiant
7	aziz	aziz	Etudiant
9	achraf		Etudiant
10	imen		Etudiant

A modal dialog box is displayed in the center, titled "Suppression" with a close button "X". It contains the message "Suppression réussie avec succès !" (Deletion successful) with an "OK" button.

At the bottom are two buttons: "Retour a L'Accueil" and "Supprimer cet Adhérent".

Si le bibliothécaire clique sur <consulter catalogue des livres> d'après la page d'accueil :

Supprimer livres	
Titre	Auteur
Le Comte de Monte-Cristo	Alexandre Dumas
Crime et Châtiment	Fiodor Dostoïevski
Le Petit Prince	Antoine de Saint-Exupéry
Les Misérables	Victor Hugo
Guerre et Paix	Léon Tolstoï
La Trilogie de la Poussière: La Belle Sauvage	Philip Pullman
À la croisée des mondes: Les Royaumes du Nord	Philip Pullman
Le Miroir d'Ambre	Philip Pullman
la poussièrè La Communauté des esprits	Philip Pullman
Fondation	Isaac Asimov
Neuromancien	William Gibson
En attendant le vote des bêtes sauvages	Yōko Tawada
L'Ève future	Auguste de Villiers de l'Isle-Adam
livre	livre

[Retour a L'Accueil](#)
|
[supprimer ce livre](#)

Si le bibliothécaire clique sur <supprimer ce livre>:

Supprimer livres	
Titre	Auteur
L'Alchimiste	Paulo Coelho
Les Fleurs du Mal	Charles Baudelaire
L'Écume des jours	Boris Vian
La Nausée	Jean-Paul Sartre
Cent ans de solitude	Gabriel García Márquez
Le Maître et Marguerite	
Fahrenheit 451	
Le Guépard	
Moby Dick	
Les Liaisons Dangereuses	
Le Comte de Monte-Cristo	
Le Rouge et le Noir	
Voyage au bout de la nuit	
La Recherche du Temps Perdu	Marcel Proust
Les Fourmis	Bernard Werber
Autant en emporte le vent	Margaret Mitchell
La Horde du Contrevent	Alain Damasio
Les Particules élémentaires	Michel Houellebecq
Les Trois Mousquetaires	Alexandre Dumas

i Suppression réussie avec succès !

[OK](#)

[Retour a L'Accueil](#)
|
[supprimer ce livre](#)

Si le bibliothécaire clique sur <Ajouter Livre> d'après la page d'accueil :

The screenshot shows a window titled "Ajouter un livre". It contains three input fields: "Titre" (Title) with the value "Guerre et Paix", "Auteur" (Author) with the value "Léon Tolstoï", and "Genre" (Genre) with the value "Historique". At the bottom, there are two buttons: "Retour a L'Accueil" (Return to Home) and "Ajouter ce livre" (Add this book).

Ajouter un livre	-	□	×
Titre	Guerre et Paix"		
Auteur	Léon Tolstoï		
Genre	Historique		
Retour a L'Accueil	Ajouter ce livre		

*Si le bibliothécaire clique sur <statistique sur les adhérents les plus assidus>
d'après la page d'accueil :*

The screenshot shows a window titled "Statistiques sur les utilisateurs les plus assidus". It displays a table with three columns: "Nom" (Name), "Prenom" (First Name), and "Nombre d'Emprunts" (Number of Loans). The data is as follows:

Statistiques sur les utilisateurs les plus assidus	-	□	×
Nom	Prenom	Nombre d'Emprunts	
ranim	ranim	4	
rihem	rihem	4	
karim	karim	3	
ALi	Ali	3	
aziz	aziz	1	
Retour a L'Accueil			

S'il clique sur <statistique sur les livres les plus empruntés> d'après la page d'accueil :

Statistiques sur les livres les plus empruntés

Titre	Auteur	Nombre d'Emprunts
Crime et Châtiment	Fiodor Dostoïevski	2
Les Misérables	Victor Hugo	2
L'Ève future	Auguste de Villiers de l'Isle-Adam	2
Le Comte de Monte-Cristo	Alexandre Dumas	1
Le Petit Prince	Antoine de Saint-Exupéry	1
La Trilogie de la Poussière: La Belle Sauvage	Philip Pullman	1
À la croisée des mondes: Les Royaumes du ...	Philip Pullman	1
Le Miroir d'Ambre	Philip Pullman	1
Fondation	Isaac Asimov	1
En attendant le vote des bêtes sauvages	Yôko Tawada	1
livre	livre	1
Guerre et Paix"	Léon Tolstoï	1

[Retour a L'Accueil](#)

Si le bibliothécaire clique sur <envoyer des Rappels de Retour> d'après la page d'accueil :

rappelle

Nom	Prenom	Titre	Auteur	date_enprunte	date_retour
ranim	ranim	Neuromancien	William Gibson	2023-12-10	2023-12-11
rihem	rihem	En attendant le vote d...	Yôko Tawada	2023-12-10	2023-12-11
karim	karim	livre	livre	2023-12-10	2023-12-11
nimou	nimou	Le Petit Prince	Antoine de Saint-Exu...	2023-12-10	2023-12-11

[Retour a L'Accueil](#) [envoyer un mail](#)

Si le bibliothécaire clique sur <envoyer un mail>:

The screenshot shows a software application window titled "rappelle". Inside, there is a table of borrowed books with columns: Nom, Prenom, Titre, Auteur, date_enprunte, and date_retour. Four rows are listed: amal, amal, Les Particules éléme..., Michel Houellebecq, 2023-12-11, 2023-12-12; ranim, ranim, Les Particules éléme..., Michel Houellebecq, 2023-12-11, 2023-12-12; ranim, ranim, Les Particules éléme..., Michel Houellebecq, 2023-12-11, 2023-12-12; ALi, Ali, La Horde du Contrev..., Alain Damasio, 2023-12-11, 2023-12-12. Overlaid on the main window is a smaller modal dialog titled "Rappel par Mail" with a message: "4 E-mails de rappel envoyés avec succès." and an "OK" button.

Si l'utilisateur n'a pas de compte alors il clique sur <S'inscrire> :

The screenshot shows a registration form titled "Inscription". The left side has a pink background with fields: nom (rihem), prenom: (amri), login (rihemamri@gmail.com), and pwd (rihem123). The right side has a white background. At the bottom, there are two radio buttons: "Etudiant" (selected) and "Enseignant". Below the radio buttons are two buttons: "Retour" and "S'inscrire".

Si l'e-mail n'est pas valide :

The screenshot shows a web-based registration form titled "Inscription". The form fields include "nom" (value: amri), "prenom:" (value: rihem), "login" (radio button selected), "pwd" (empty), "Etudiant" (radio button selected), and "Enseignant" (radio button unselected). Below the form is a "Retour" button and an "S'inscrire" button. A modal dialog box titled "Erreur d'Inscription" displays the message "Email Invalid !".

Si le mot de passe n'est pas valide :

The screenshot shows the same registration form as the previous one. The "pwd" field is empty. A modal dialog box titled "Erreur d'Inscription" displays the message "Le mot de passe doit avoir une longueur d'au moins 8 caractères.".

Inscription

nom	amri
prenom:	<p>Erre^{ur} d'Inscription</p> <p>Le mot de passe doit contenir à la fois des lettres et des chiffres.</p> <p>OK</p>
login	
pwd	rihemrihem
<input checked="" type="radio"/> Etudiant	<input type="radio"/> Enseignant
Retour S'inscrire	

Si l'e-mail existe déjà :

Inscription

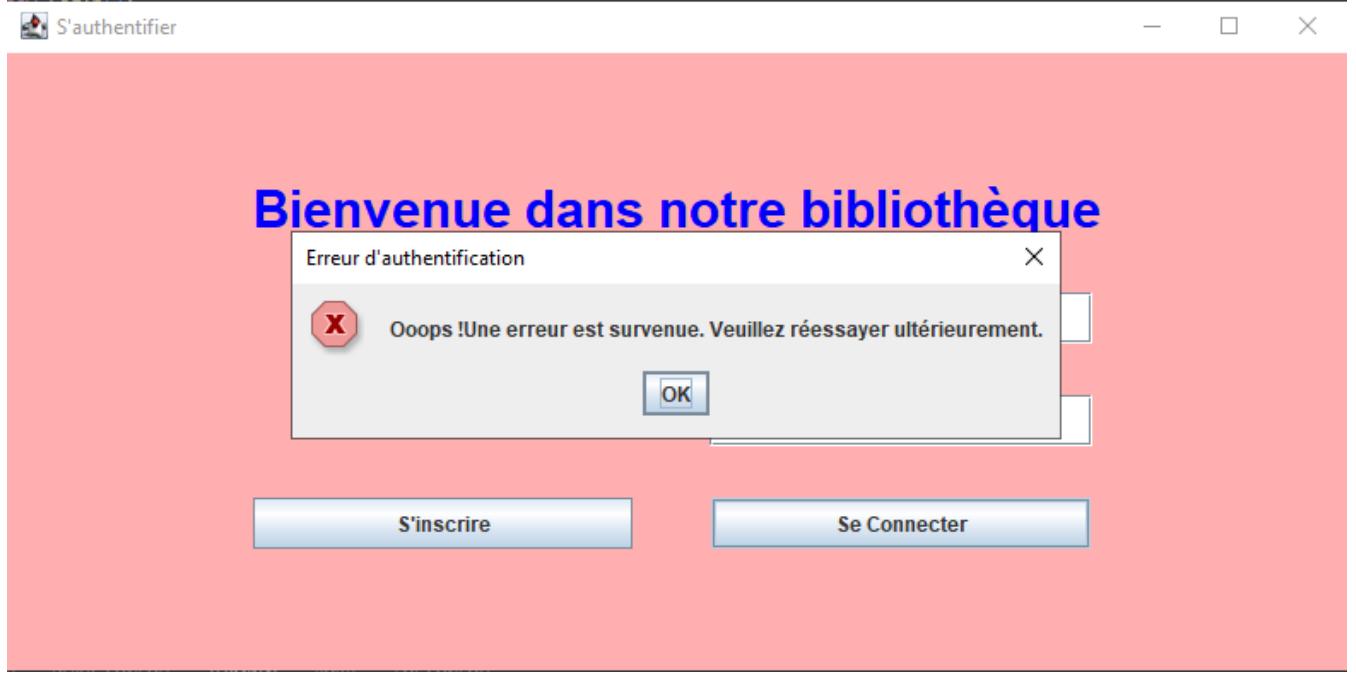
nom	amri
prenom:	rihem
login	
pwd	
<input checked="" type="radio"/> Etudiant	<input type="radio"/> Enseignant
Retour S'inscrire	

Erre^{ur} d'Inscription

EMail Exist Deja

OK

Si il y a un problème de connexion a la base de donnée ou une SQLException :



V- Conclusion :

Grâce au présent travail, nous avons réussi à concevoir et implémenter une application performante en Java permettant de gérer une bibliothèque de manière optimale selon les besoins établis. Avec un modèle conceptuel de données solide, la création d'une base de données performante devient plus facile ; quant au maintien et au développement continu du logiciel, ils sont assurés par un code correctement structuré. Avec son interface conviviale et ses fonctionnalités étendues, cette application se présente comme le choix idéal pour moderniser la gestion des emprunts de livres dans cet établissement.