RAPPORT DE PROJET

Application de gestion des prêts d’ouvrages dans une bibliothèque.

Travail par :
-Yessine Helal
-Mohamed Adem Selmi

Introduction:

Les informations circulant dans la bibliothèque sont de nature diverse, elles concernent le lecteur depuis son arrivée jusqu'à son départ de la bibliothèque, et pour le bon fonctionnement de celle-ci, nous avons pensé à mettre en place un système informatique qui facilitera les tâches de gestion de la bibliothèque. L'application à mettre en œuvre permettra une interaction rigoureuse entre le bibliothécaire, le lecteur et les ressources, en assurant ainsi une bonne gestion de la bibliothèque.

Les lecteurs voulant s'informer ou emprunter des ouvrages dans l'année, et les étrangers venant de l'extérieur des locaux de la bibliothèque augmentent d'année en année ce qui engendre cumul d'information. Les responsables trouvent des difficultés dans leurs travaux, le taux d'erreur ne cesse d'accroître et certaines tâches font qu'elles sont pénibles.

À traiter, dont on souligne les problèmes suivants :

-Toutes les procédures sont faites manuellement.

-Mauvais archivage des documents.

-Difficulté et retard lors de la recherche de l'information.

-Perte de temps.

Notre travail vise à réaliser une application réseau pour la gestion d'une bibliothèque qui consiste à gérer les lecteurs, les ouvrages, les emprunts, ainsi que les retours, qui seront acteur. Afin d'atteindre les objectifs, l'application à concevoir offrira les espaces suivants :

Un espace réservé à l'administrateur de l'application qui lui permet la gestion des lecteurs, la gestion des ouvrages, la gestion des emprunts et la mise à jour des données (l'ajout, modification, suppression).

Un espace réservé aux autres utilisateurs de l'application (employés de la bibliothèque par exemple) le réceptionniste…etc. après avoir un compte (login et mot de passe) configuration par l'administrateur et des privilèges pour avoir accès à leurs fonctionnalités.

Besoins fonctionnels

Les besoins fonctionnels se présentent en sept (6) grandes parties :

1. Authentification :

- Cette interface permet aux utilisateurs (administrateurs, bibliothécaires et lecteurs) d'accéder à leur espace personnel après une authentification sécurisée (login et mot de passe, ou autre méthode d'identification).

2. Effectuer une recherche :

- Permet aux utilisateurs de rechercher des documents (livres, articles, ressources numériques) en saisissant un mot-clé, un titre, un auteur, ou une catégorie.

- Fonctionnalités avancées : filtres par genre, disponibilité, langue, etc.

3. Gestion des utilisateurs (adhérents) :

- Permet aux bibliothécaires ou administrateurs d'ajouter, modifier ou supprimer des comptes d'adhérents.

- Gestion des profils (coordonnées, historique d'emprunts, abonnements).

4. Gestion des documents :

- Permet au personnel de la bibliothèque d'ajouter, modifier ou supprimer des références dans le catalogue.

- Gestion des métadonnées (auteur,date de publication, etc.).

5. Gestion des emprunts et retours :

- Permet d'enregistrer les prêts et les retours de documents.

- Calcul automatique des dates de retour et gestion des retards (amendes si nécessaire).

6. Gestion des réservations :

- Permet aux utilisateurs de réserver des documents actuellement indisponibles.
- Notification automatique lorsque le document est de nouveau accessible.

7. Rapports et statistiques (optionnel) :

- Génération de rapports sur les prêts, les retours, les documents populaires, etc.
- Statistiques d'utilisation pour améliorer la gestion des ressources.

Ces fonctionnalités couvrent les besoins essentiels d'un système moderne de gestion de bibliothèque, tout en restant modulables selon les spécificités de l'établissement.

Besoins non fonctionnels :

Les besoins non fonctionnels sont essentiels car ils influencent indirectement les résultats et l'efficacité des utilisateurs. Pour une application de gestion de bibliothèque, les exigences suivantes doivent être respectées :

1.Fiabilité :
Le système doit fonctionner de manière stable et cohérente, avec un minimum d'erreurs, pour garantir une expérience satisfaisante aux utilisateurs (bibliothécaires et lecteurs).

2.Gestion des erreurs :
Toutes les ambiguïtés ou problèmes doivent être signalés par des messages d'erreur clairs et explicites, guidant efficacement l'utilisateur dans la résolution du problème.

3.Ergonomie et interface intuitive : L'application doit être conçue pour une utilisation facile et intuitive, avec :

- Une navigation fluide entre les différentes sections
- Une hiérarchie visuelle claire (couleurs, typographie, espacement)
- Une accessibilité adaptée à tous les utilisateurs

4. Sécurité des données : Le système doit garantir :

- La confidentialité des données personnelles des utilisateurs (adhérents)
- La protection des historiques d'emprunt
- Un système d'authentification sécurisé pour les différents niveaux d'accès (administrateurs, bibliothécaires, usagers)

5. Maintenance et évolutivité : L'architecture doit être :

- Modulaire pour faciliter les mises à jour
- Documentée pour permettre une maintenance aisée
- Conforme aux standards pour une éventuelle intégration avec d'autres systèmes

6. Compatibilité et portabilité : L'application doit être :
   - Accessible sur différentes plateformes (web, mobile)
   - Compatible avec les principaux systèmes d'exploitation
   - Adaptée à différents types de dispositifs (ordinateurs, tablettes)

7. Interopérabilité : Capacité à échanger des données avec :
   - D'autres systèmes de gestion de bibliothèque
   - Des plateformes de livres numériques
   - Des systèmes de paiement pour les amendes

Répartition des tâches :

Yessine Helal :

- Interface de login
- Coté admin (supprimer user, ajouter livre, supprimer livre...)
- Procédure PL/SQL (get_user_borrowed_books)
- Diagramme entité-association

Mohamed Adem Selmi :

- Interface de création d'un compte
- Coté utilisateur (emprunter un livre, retourner un livre, gestion de pénalité...)
- Création de la base de données
- Diagramme cas utilisation

Les requêtes SQL Utilisé en Java:

```
}else if(e.getSource() == login){
    Conn c=new Conn();
    String u=cinTextField.getText();
    String password=String.valueOf(passTextField.getPassword());
    String q1="select * from admn where username = '"+u+"'";
    String q2="select * from usr where username = '"+u+"'";
    try{
        if(clt.isSelected()){                          //si le bouton radio client est selectionne
            ResultSet rs= c.s.executeQuery(q2);        //requete sql pour verifier si l'utilisateur existe dans la table utilisateur
            if(rs.next()){
                if(rs.getString(columnLabel:"mdp").equals(password)){ //verifier si le mot de passe est correct
                    setVisible(b:false);
                    new UserPage(u).setVisible(b:true); //ouvrir la fenetre d'accueil de l'utilisateur
                }
                else{JOptionPane.showMessageDialog(parentComponent:null,message:"Password is incorrect");}
            }else{
                JOptionPane.showMessageDialog(parentComponent:null,message:"this user does not exist");
            }

        }else if(adm.isSelected()){                  //si le bouton radio admin est selectionne
            ResultSet rs= c.s.executeQuery(q1);      //requete sql pour verifier si l'utilisateur existe dans la table admin
            if(rs.next()){
                setVisible(b:false);  //fermer la fenetre de connexion
                new PageAdmin(u).setVisible(b:true); //ouvrir la fenetre d'accueil de l'admin
            }else{
                JOptionPane.showMessageDialog(parentComponent:null,message:"Incorrect Username or Password");
            }
        }

    }catch(Exception ex){
        System.out.println(e);
    }
```

Les requêtes SQL Utilisé en Java:

```
}else if(e.getSource()==show){
    table= new DefaultTableModel();
    table.addColumn(columnName:"ID");
    table.addColumn(columnName:"Title");
    table.addColumn(columnName:"Author");
    table.addColumn(columnName:"Available");        //creer un tableau pourt afficher les livres
    try{
        Conn c=new Conn();
        ResultSet rs=c.s.executeQuery(sql:"select id, nom, author,available from book"); //requete sql pour recuperer les livres
        while(rs.next()){
            Object row[] ={
                rs.getInt(columnLabel:"id"),
                rs.getString(columnLabel:"nom"),
                rs.getString(columnLabel:"author"),
                rs.getInt(columnLabel:"available")
            };
            table.addRow(row); //ajouter le livre dans le tableau
        }
        }catch(Exception ex){
            System.out.println(ex);
        }
        bookTable = new JTable(table);
        JScrollPane scrollPane = new JScrollPane(bookTable);
        scrollPane.setBounds(x:150, y:350, width:500, height:100);
        add(scrollPane, BorderLayout.CENTER);
```

```
}else if(e.getSource()== delete){
    String u=usernameTextField.getText();
    try{
       if(u.equals(anObject:"")){
          JOptionPane.showMessageDialog(parentComponent:null,message:"Username is required");
        }else if(u.length()>20){
           JOptionPane.showMessageDialog(parentComponent:null,message:"Username can't be longer than 20c haracters");
        }
        else{
          Conn c = new Conn();
          if(!c.s.executeQuery("select * from usr where username ='"+u+"'").next()){
           JOptionPane.showMessageDialog(parentComponent:null,message:"User doesn't exist");
          }
          else if(!c.s.executeQuery("select * from usr where username ='"+u+"' and nlivre =0").next()){
           JOptionPane.showMessageDialog(parentComponent:null,message:"User a des livres a retourner \n On ne peut pas le supprimer");
          }else{
           String q = "delete from usr where username= '"+u+"'";
           c.s.executeUpdate(q);
           JOptionPane.showMessageDialog(parentComponent:null,message:"User deleted successfully!");
          }
       }
    }catch(Exception ex){
       System.out.println(ex);
    }
    userTable = new JTable(table);
    JScrollPane scrollPane = new JScrollPane(userTable);
    scrollPane.setBounds(x:150, y:350, width:500, height:100);
    add(scrollPane, BorderLayout.CENTER);
```

Les requêtes SQL Utilisé en Java:

```java
}else if(e.getSource()==show){
    String titre=usernameTextField.getText();
    table= new DefaultTableModel();
    table.addColumn(columnName:"Username");
    table.addColumn(columnName:"Nom");
    table.addColumn(columnName:"Prenom");
    table.addColumn(columnName:"Password");
    table.addColumn(columnName:"Fin de penalite");
    try{
        if(titre.equals(anObject:"")){
           JOptionPane.showMessageDialog(parentComponent:null,message:"Username is required");
        }else{
         Conn c=new Conn();
         ResultSet rs=c.s.executeQuery("select username, nom, prenom, mdp,finpenalty from usr where username= '"+titre+"'");
         while(rs.next()){
             Object row[] ={
                 rs.getString(columnLabel:"username"),
                 rs.getString(columnLabel:"nom"),
                 rs.getString(columnLabel:"prenom"),
                 rs.getString(columnLabel:"mdp"),
                 rs.getString(columnLabel:"finpenalty")
             };
             table.addRow(row);}                }
         }catch(Exception ex){
             System.out.println(ex);
         }
         bookTable = new JTable(table);
         JScrollPane scrollPane = new JScrollPane(bookTable);
         scrollPane.setBounds(x:150, y:350, width:500, height:100);
         add(scrollPane, BorderLayout.CENTER);
        }
```

Les requêtes SQL Utilisé en Java:

```
}else if(e.getSource()== delete){
    String titre=titrenameTextField.getText();
    String auteur=auteurTextField.getText();
    try{
       if(titre.equals(anObject:"")){
           JOptionPane.showMessageDialog(parentComponent:null,message:"Title is required");
       }else{
           Conn c = new Conn();
           if(!c.s.executeQuery("select * from book where nom ='"+titre+"' and author = '"+auteur+"'").next()){ //verifier si le livre existe
               JOptionPane.showMessageDialog(parentComponent:null,message:"book doesn't exist");
            JOptionPane.showMessageDialog(parentComponent:null,message:"book doesn't exist");
           }
           else if(c.s.executeQuery("select * from book where nom = '"+titre+"' and author = '"+auteur+"' and available=0 ").next()){ //verifier si le livre est disponible
            JOptionPane.showMessageDialog(parentComponent:null,message:"Ce livre est actuellement emprunté \n On ne peut pas le supprimer");
           }else{
            String q = "delete from book where nom= '"+titre+"' and author = '"+auteur+"'"; //requete de suppression du livre
            c.s.executeUpdate(q);
            JOptionPane.showMessageDialog(parentComponent:null,message:"book deleted successfully!");
           }
       }

    }catch(Exception ex){
        System.out.println(ex);
    }
}else if(e.getSource()==show){
    table= new DefaultTableModel();
    table.addColumn(columnName:"ID");
    table.addColumn(columnName:"Title");
    table.addColumn(columnName:"Author");
    table.addColumn(columnName:"Available");
    try{
        Conn c=new Conn();
        ResultSet rs=c.s.executeQuery(sql:"select id, nom, author,available from book where available=true"); //requete sql pour recuperer les livres
        while(rs.next()){
            Object row[] ={
                rs.getInt(columnLabel:"id"),
                rs.getString(columnLabel:"nom"),
                rs.getString(columnLabel:"author"),
                rs.getInt(columnLabel:"available")
            };
            table.addRow(row);  //ajouter le livre dans le tableau
       }
        }catch(Exception ex){
            System.out.println(ex);
        }
        bookTable = new JTable(table);
        JScrollPane scrollPane = new JScrollPane(bookTable);
        scrollPane.setBounds(x:150, y:350, width:500, height:100);
        add(scrollPane, BorderLayout.CENTER);

}
```

```
}else if(e.getSource()== modifier){
    String titre=titrenameTextField.getText();
    String auteur=auteurTextField.getText();
    String genre=genreTextField.getText();
    String status="1";
    try{
       if(titre.equals(anObject:"")){
           JOptionPane.showMessageDialog(parentComponent:null,message:"Title is required");
       }else{
           Conn c = new Conn();
           String q1 = "insert into book (nom,author,available,genre) values('"+titre+"','"+auteur+"','"+status+"','"+genre+"')";
           c.s.executeUpdate(q1);      //requete d'insertion du livre dans la base de données
           JOptionPane.showMessageDialog(parentComponent:null, message:"Book added successfully.");
       }

    }catch(Exception ex){
       System.out.println(ex);
    }
```

Les requêtes SQL Utilisé en Java:

```
    q1="select * from book where available =1"; // requette qui remettre les livres disponible dans la bibiothèque
    q2="select * from historique join book on livre=id where usr = '"+ username+"' and dateretour is null";
    //requette pour les livres actuellement emprunté par l'utilisateur
    q3="select * from historique join book on livre=id where usr = '"+ username+"'";
    //requette pour tous livres actuellement emprunté par l'utilisateur
    condition="";
    // condition initialement vide, elle serai utilisé pour le filtrage
```

```
cuteQuery("select nom,author,count(*) as c from historique join book on livre =id where usr= '"+username +"' group by id order by c desc limit 1;")
          une requette qui remettre le livre emprunté le plus de fois par l'utlisateur

xt("Votre livre préferé: "+r.getString(columnIndex:1)+" par "+r.getString(columnIndex:2));

xt(text:"Que serait votre premier livre?");

sql:"select nom,author,count(*) as c from historique join book on livre =id group by id order by c desc limit 1;");
    une requette qui remettre le livre le plus emprunté par tous les utilisateurs

"Le livre le plus populaire dans la bibliothèque: "+r.getString(columnIndex:1)+" par "+r.getString(columnIndex:2));}
```

```
int n=Integer.parseInt(id.getText());
// les requette utilisé pour verifier la demande d'un livre
if(c.s.executeQuery("select * from usr where nlivre =5 and username = '"+username+"'").next()){
    ans.setText(t:"Vous avez déja le nombre maximal de livres (5)");

}
else if(c.s.executeQuery("select * from usr where username = '"+username+"' and finpenalty >= CURDATE() ").next()){
    ans.setText(t:"Vous etes maintenant pénalisé !!");
}
else if(c.s.executeQuery("select * from historique where dateretour is null and datepret + interval 30 day < CURDATE() and usr ='"+userr
    ans.setText(t:"Vous avez un livre qui a dépasser 30 jours \n Il faut le retourner avant demander d'autre livres");
}
else if(c.s.executeQuery("select * from book where id = '"+n+"' and available =1").next()){
    c.s.executeUpdate("insert into historique values ('"+username+"',"+n+",CURDATE(),null)");
    // inserer dans la table historique la transaction avec date de pret est la date actuelle, date de retour est null
    c.s.executeUpdate("Update book set available = 0 where id ="+n);
    // changer la disponibilité du livre
    c.s.executeUpdate("Update usr set nlivre = nlivre+1 where username = '"+username+"'");
    // incrementer l'attribut nombre de livre de l'utilisateur
    ans.setForeground(new java.awt.Color(r:0, g:128, b:0));;
    ans.setText(t:"Transaction succesif!");
}
```

```java
ResultSet r=c.s.executeQuery(sql:"select count(distinct author) as c from book");
//requette pour avoir le nombre d'auteurs
r.next();
String[] s= new String[r.getInt(columnLabel:"c")+1];
s[0]="Any";
r=c.s.executeQuery(sql:"select distinct author from book");
// requette qui remet la liste des auteurs
int i=1;
while(r.next()){
    s[i]=r.getString(columnLabel:"author");
    i++;
}
author.setModel(new javax.swing.DefaultComboBoxModel<>(s));
r=c.s.executeQuery(sql:"select count(distinct genre) as c from book");
// de meme pour le genre
r.next();
s= new String[r.getInt(columnLabel:"c")+1];
s[0]="Any";
r=c.s.executeQuery(sql:"select distinct genre from book");
i=1;
while(r.next()){
    s[i]=r.getString(columnLabel:"genre");
    i++;
}
genre.setModel(new javax.swing.DefaultComboBoxModel<>(s));
```

```java
// les requettes utilisé pour verifier et retourner un livre
ResultSet r=c.s.executeQuery("Select datepret from historique where usr = '"+username+"' and livre= "+i+" and dateretour is null");
if(r.next()){
    LocalDate d=LocalDate.now().plusDays(-30);
    String d1=r.getString(columnLabel:"datepret");
    String d2=d.toString();
    c.s.executeUpdate("update book set available = 1 where id="+i); // rendre le livre disponible
    c.s.executeUpdate("update usr set nlivre = nlivre-1 where username = '"+username+"'"); // décrementer l'attribut de nombre de livre du user
    c.s.executeUpdate("update historique set dateretour = CURDATE() where usr = '"+username+"' and livre= "+i+" and dateretour is null");
    // changer la date de retour (initialement nulle) avec la date actuelle
    if (d2.compareTo(d1)>0){
        // d2 contient date de retour moins 30 jours on compare cette valeur avec date de pret pour verifier si l'utilisateur doit etre penalisé
        ans.setText(t:"Livre retourné après la date limite \n Pour cela vous serez penalisé");
        r=c.s.executeQuery("select finpenalty from usr where username = '"+username+"'");
        r.next();
        d1=r.getString(columnIndex:1);
        d2=LocalDate.now().toString();
        if(d1==null || d2.compareTo(d1)>0){
            // la penalité est un ban du user d'emprunté des livres pour 3 mois
            // si l'utilisateur n'est pas en periode de penalité la date de fin de penalité est la date actuelle + 3 mois
            c.s.executeUpdate("update usr set finpenalty = CURDATE() + interval 3 MONTH where username = '"+username+"'");
        }
        else{
            // si user déja penalisé on incremente la date de fin de penalité par 3 mois
            c.s.executeUpdate("update usr set finpenalty = finpenalty + interval 3 MONTH where username = '"+username+"'");
        }
```

PL/SQL:

Procédure qui prend en entrée l'identifiant d'un utilisateur de la bibliothèque et renvoie la liste des livres empruntés par cet utilisateur, y compris le titre du livre, la date d'emprunt et la date de retour prévue.

```
use biblio;
DELIMITER $$
CREATE PROCEDURE get_user_borrowed_books(IN user_id varchar(20))
BEGIN
    SELECT id,nom,
           datepret,
           datepret + interval 30 DAY as dateretourprevue
        FROM historique
        JOIN book ON livre = id
        WHERE usr=user_id and dateretour is null;
END $$
DELIMITER ;
call get_user_borrowed_books('jeff');
```

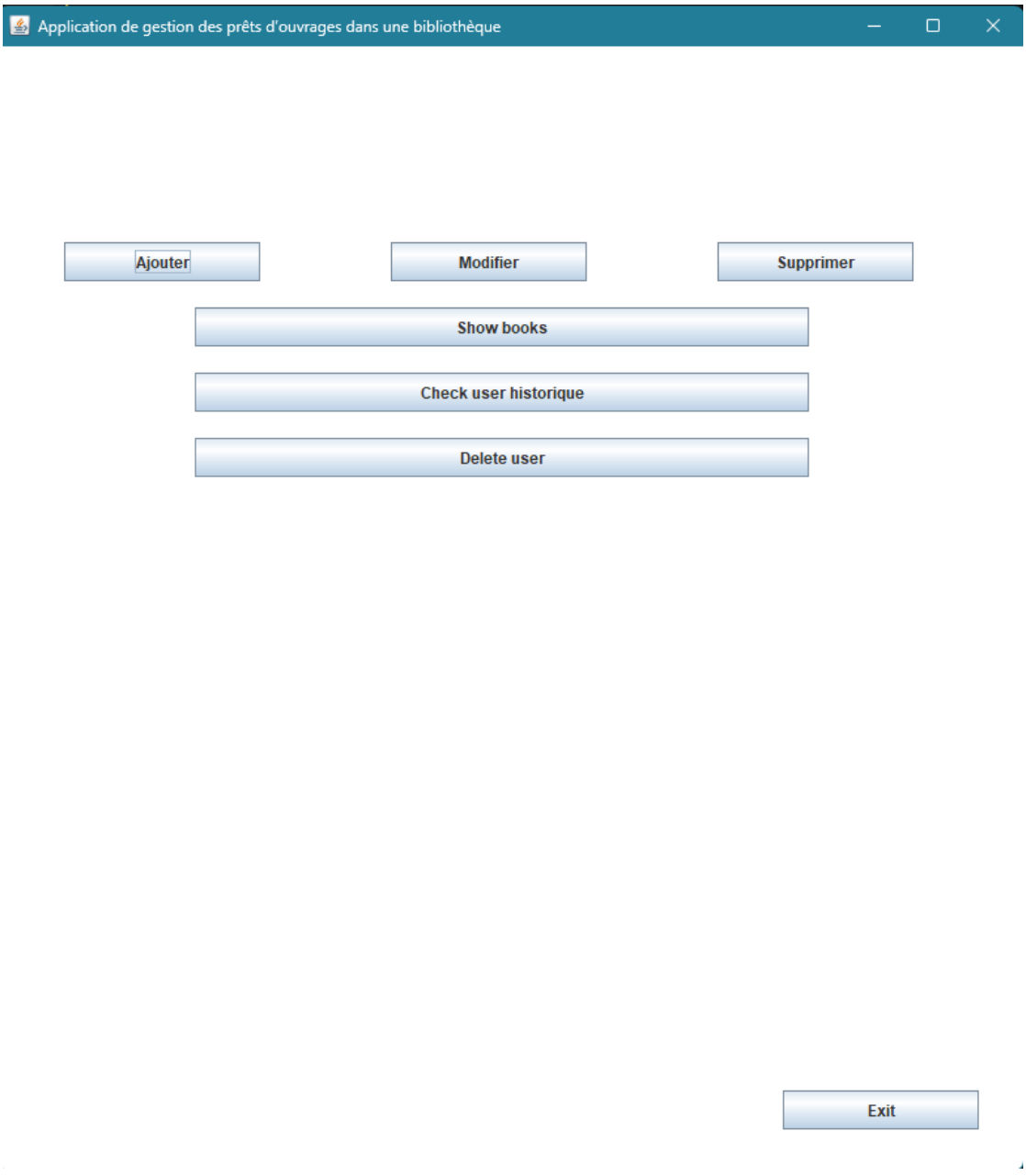
Les Interfaces Java:

Login:

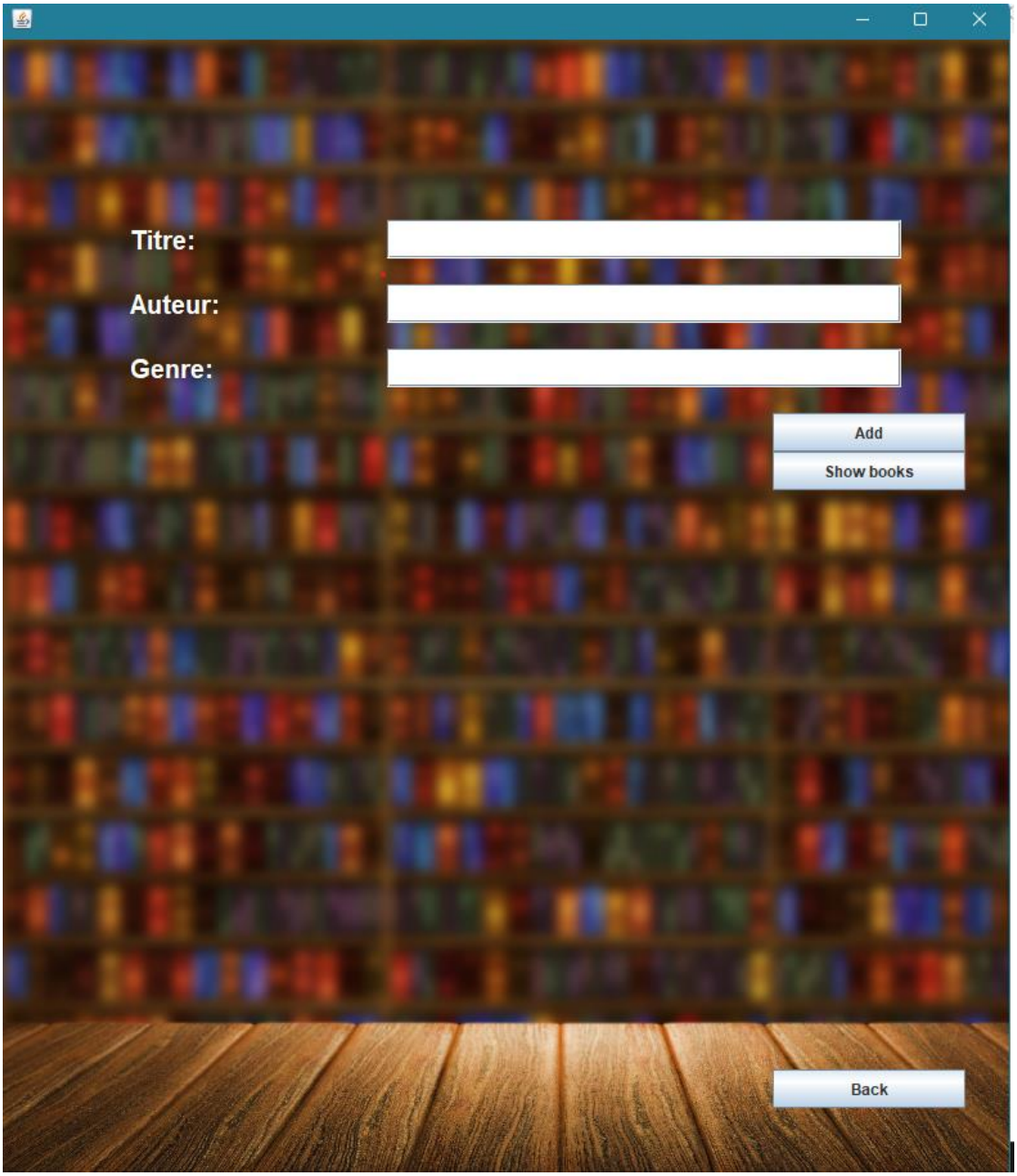
Sign Up:


Welcome to FST Library
Prénom :
Nom:
Hillary
Clinton
Enter votre date de naissance
Il faut avoir au moin 13 ans
nai 2020
Choisir username:
jeff
username utilisé
Mot-de-passe:
eee
Verifier mot-de-passe
••
les mots de passe ne sont pas les memes
Créer

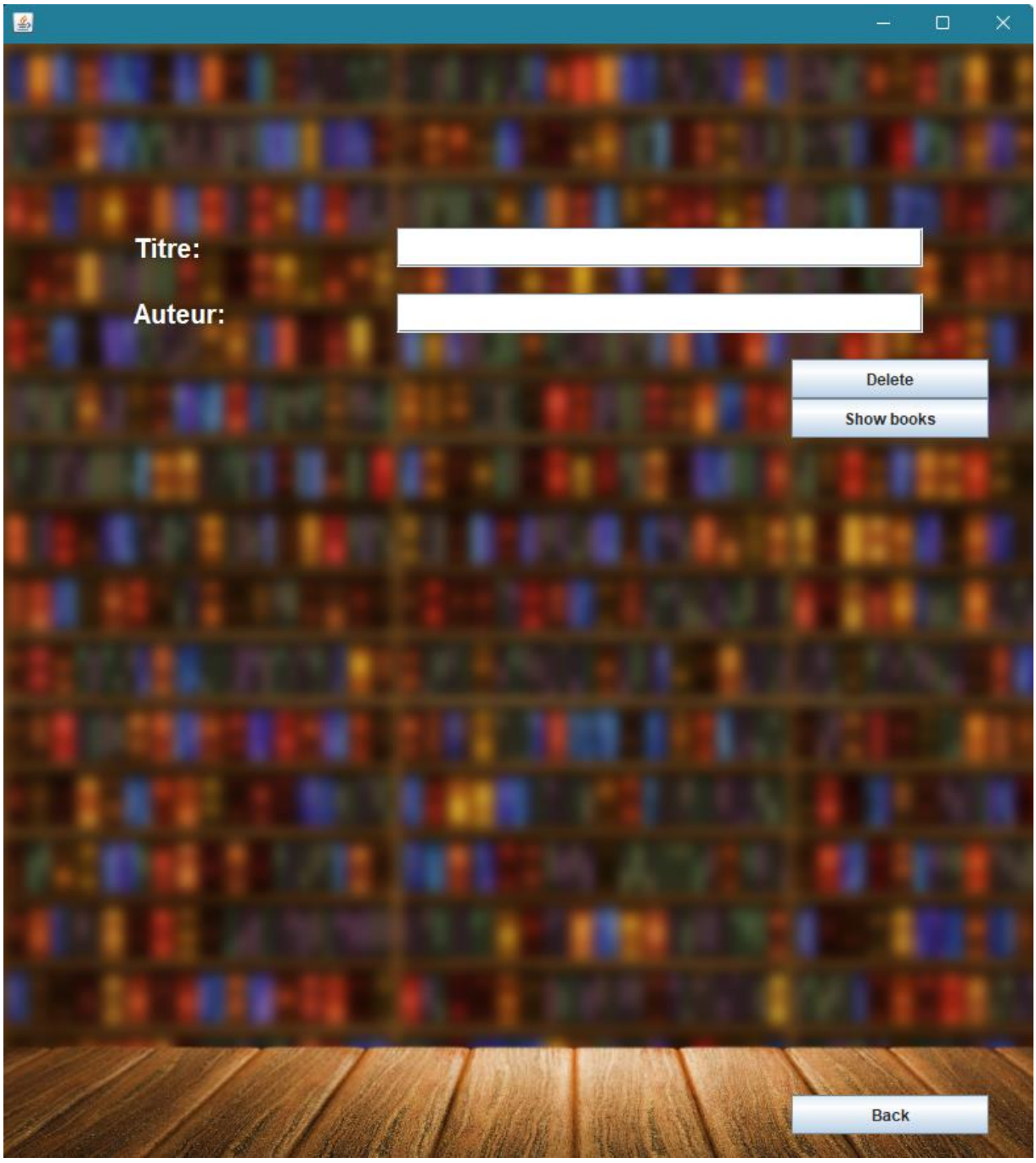
Dashboard Admin:



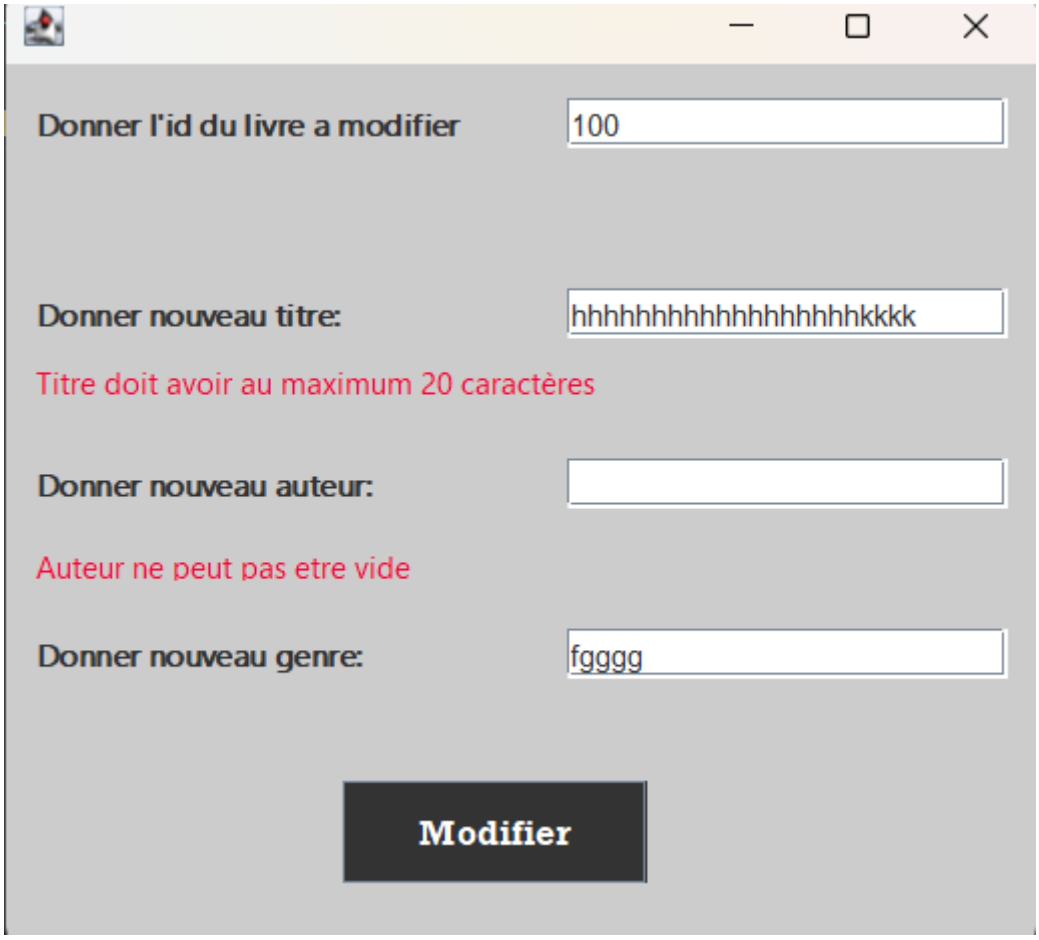
Ajouter un livre:



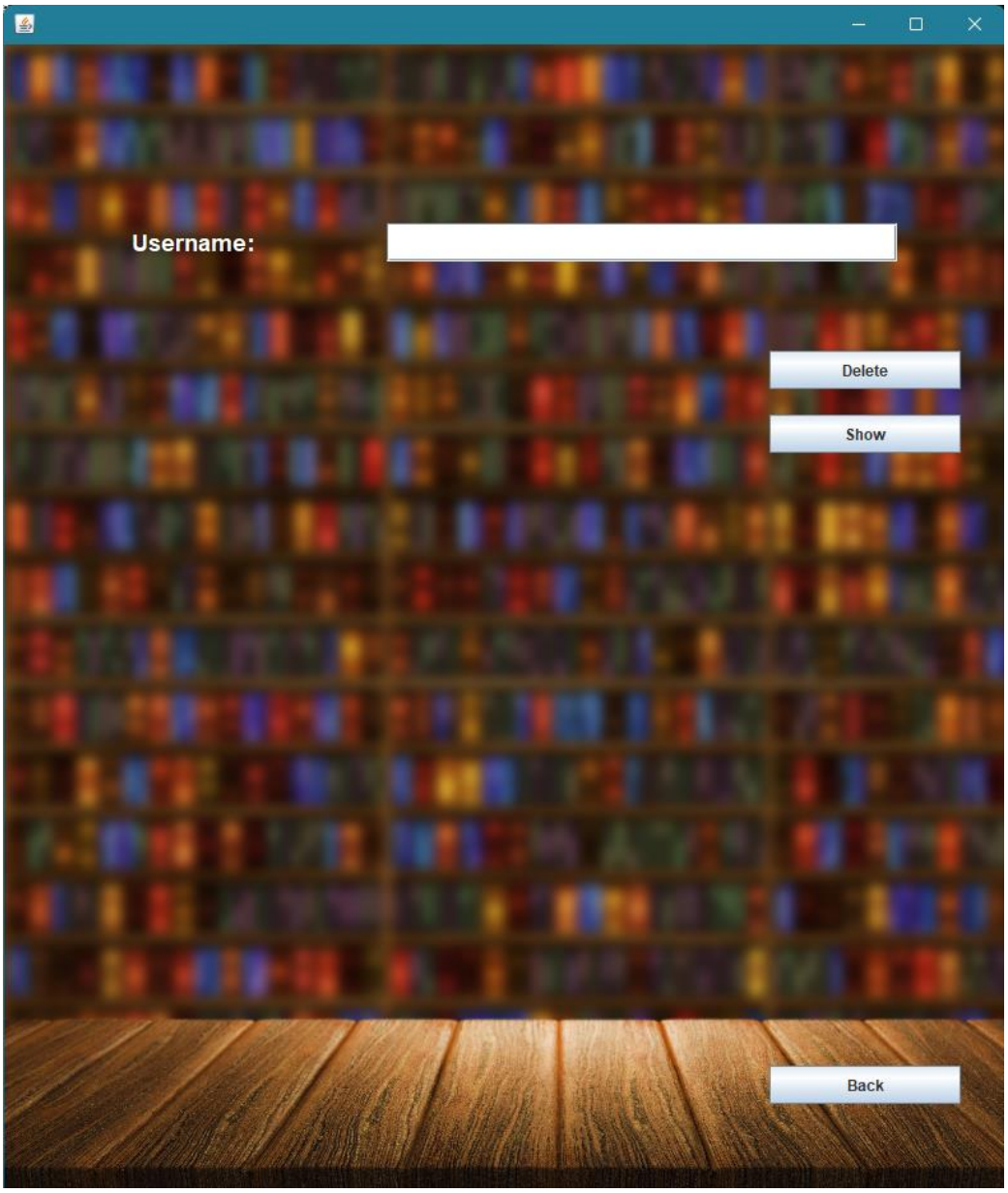
Supprimer un livre:



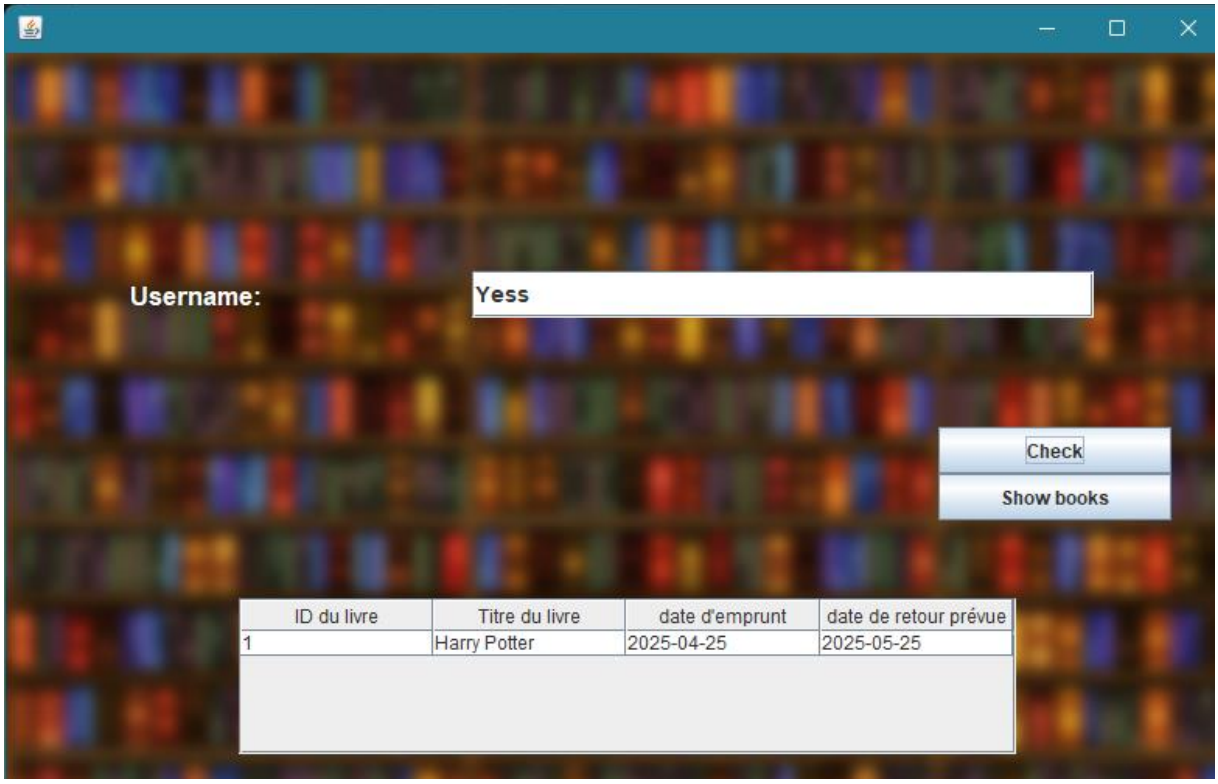
Modifier Livre:


Donner l'id du livre a modifier
100
Donner nouveau titre:
hhhhhhhhhhhhhhhhhhkkkk
Titre doit avoir au maximum 20 caractères
Donner nouveau auteur:
Auteur ne peut pas etre vide
Donner nouveau genre:
ffgggg
Modifier

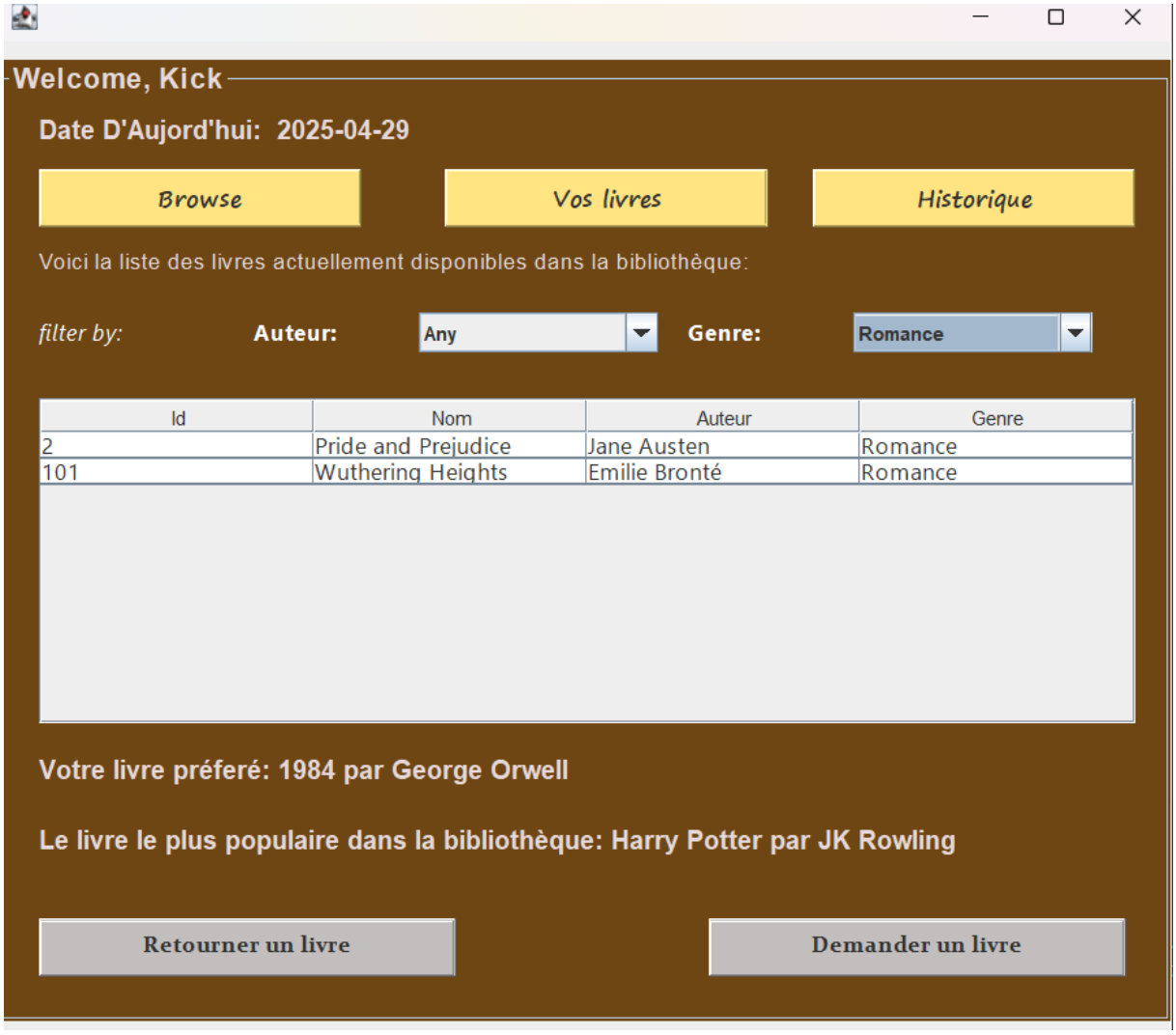
Supprimer user:



Vérifier l’historique d’un user:


Username:
Yess
Check
Show books
ID du livre
Titre du livre
date d'emprunt
date de retour prévue
1
Harry Potter
2025-04-25
2025-05-25

Page d’utilisateur:



Interface de retour d’un livre:

Interface de demande d’un livre:

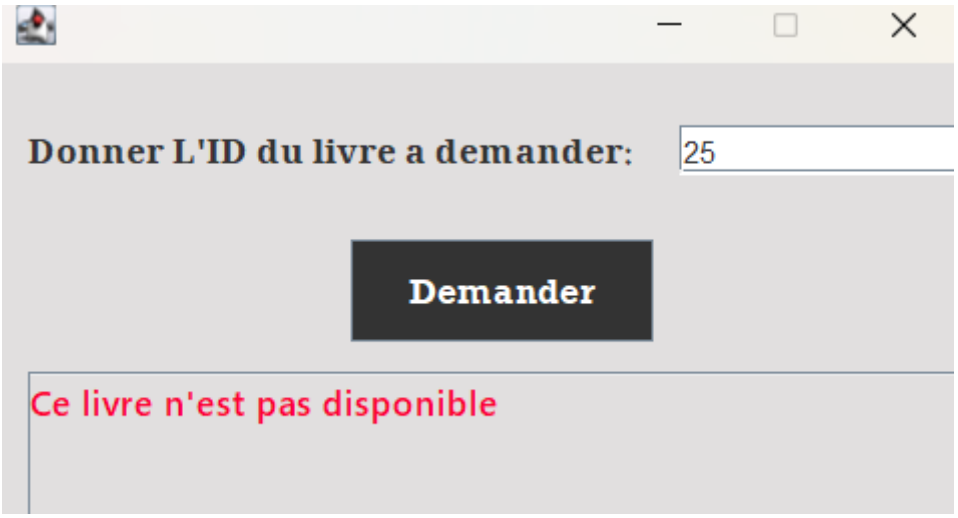


Diagramme de cas d’utilisation :

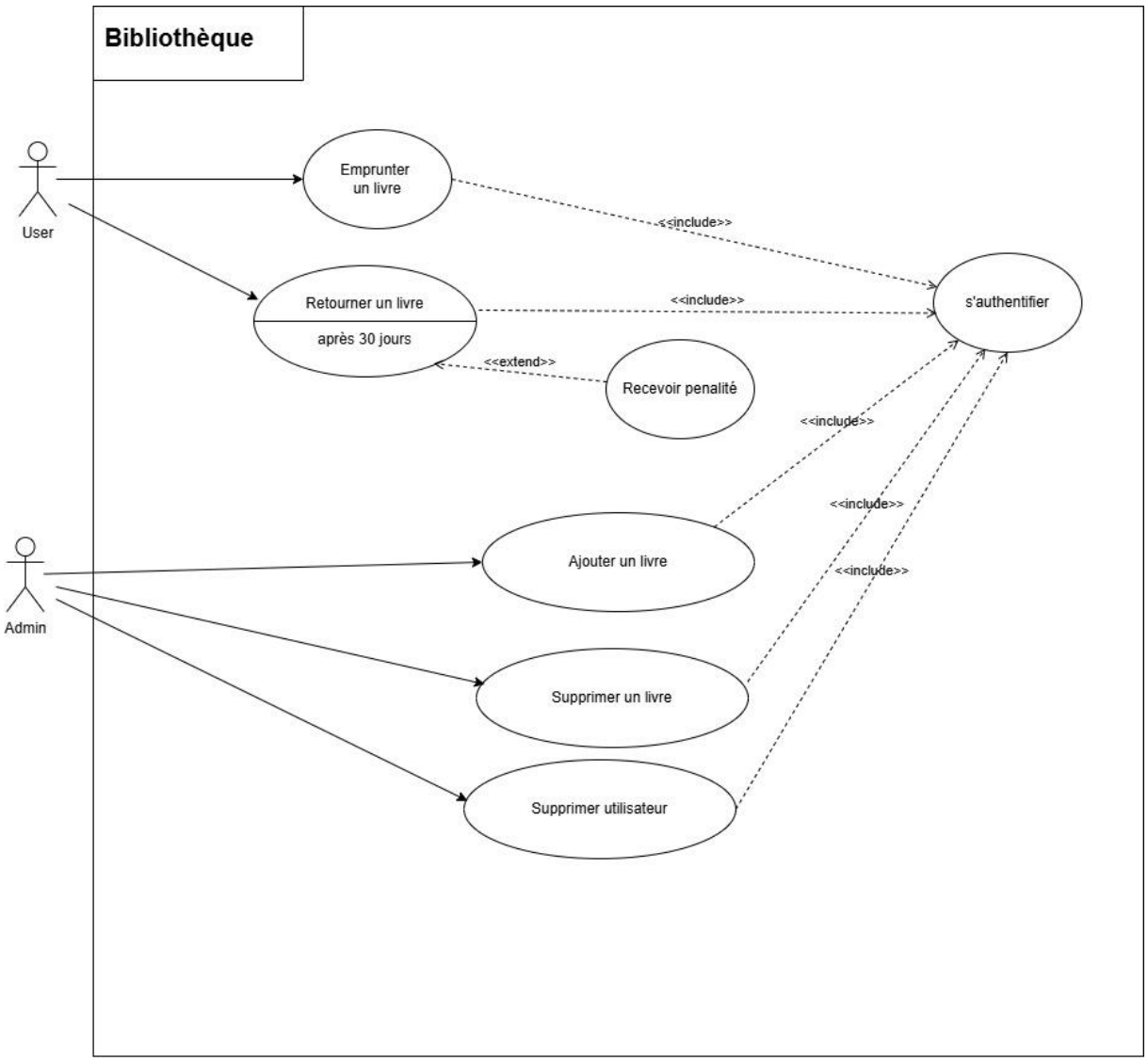


Diagramme Entité-Association :