



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMO de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique



# MINI PROJET

*Spécialité : GSI*

---

un système de gestion de la bibliothèque

---

Réalisé par

— AMOURI YACINE

— AZIB OUASSIM

2024/2025

# Chapitre 1

## Practical work

### 1.1 introduction

Les design patterns sont des solutions réutilisables et éprouvées pour résoudre des problèmes courants lors de la conception de logiciels et dans cet TP on va les appliquer sur python

#### 1.1.1 Environnement logiciel

les outils utilisés sont :

-java

-StarUML

-Visual Studio Code 1.94

### 1.2 objectif

L'université de Bouira envisage de mettre en place un système de gestion bibliothécaire en collaboration avec le département d'informatique de l'université. L'intention est de créer un système facilitant la gestion de la bibliothèque possédant les caractéristiques suivantes :

- **Administration des ouvrages** : Ajout, suppression et recherche de livres.
- **Administration des utilisateurs** : Enregistrement, connexion et location de livres.
- **Administration des prêts** : Un utilisateur a la possibilité d'emprunter un ou plusieurs ouvrages.

## 1.3 Diagramme de Class

### — Singleton

**Classe :** `bibliotheque`

**Raison :** Le pattern Singleton est utilisé pour s'assurer qu'il existe une unique instance de la classe `bibliotheque`, accessible à tous les utilisateurs. Cela évite la duplication et centralise la gestion des livres et des utilisateurs.

### — Factory Method

**Classe :** `utilisateur`

**Raison :** Le pattern Factory Method permet de créer des objets `lecteur` ou `admin` sans exposer directement la logique de création. Cela rend l'extension du système plus simple si d'autres types d'utilisateurs sont ajoutés à l'avenir.

### — Builder

**Classe :** `LivreBuilder`

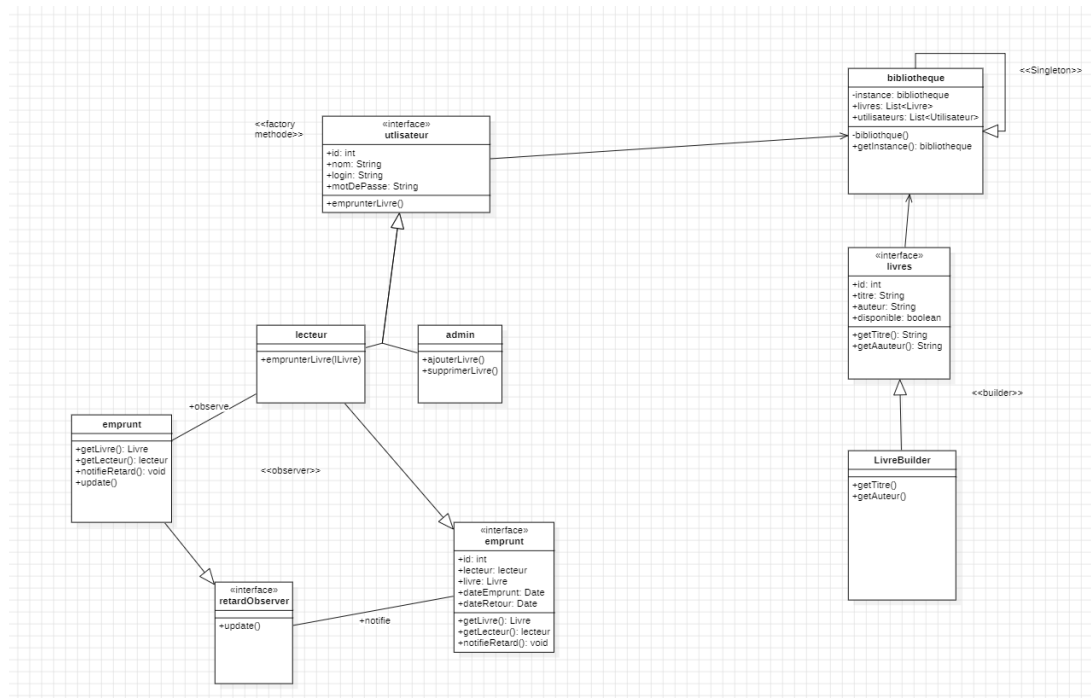
**Raison :** Le pattern Builder est utilisé pour construire des objets complexes de type `livres`, avec une flexibilité dans l'assignation des attributs (`titre`, `auteur`, `disponible`), ce qui facilite l'extension du modèle sans perturber la construction de l'objet.

### — Observer

**Classes :** `emprunt`, `retardObserver`, `lecteur`

**Raison :** Le pattern Observer permet à un `emprunt` de notifier automatiquement les observateurs en cas de retard. Ce mécanisme réduit le couplage entre les classes et rend la notification de retard flexible et extensible.

### — Interface/Abstraction



**Interfaces :** utilisateur, livres, emprunt, retardObserver

**Raison :** L'utilisation d'interfaces permet de définir des contrats clairs pour les classes concrètes. Cela rend le système plus extensible, facilite le remplacement ou la modification des implémentations, et respecte le principe de ségrégation des interfaces.

## 1.4 Capture d'écran sur projets

### 1.4.1 etape 1

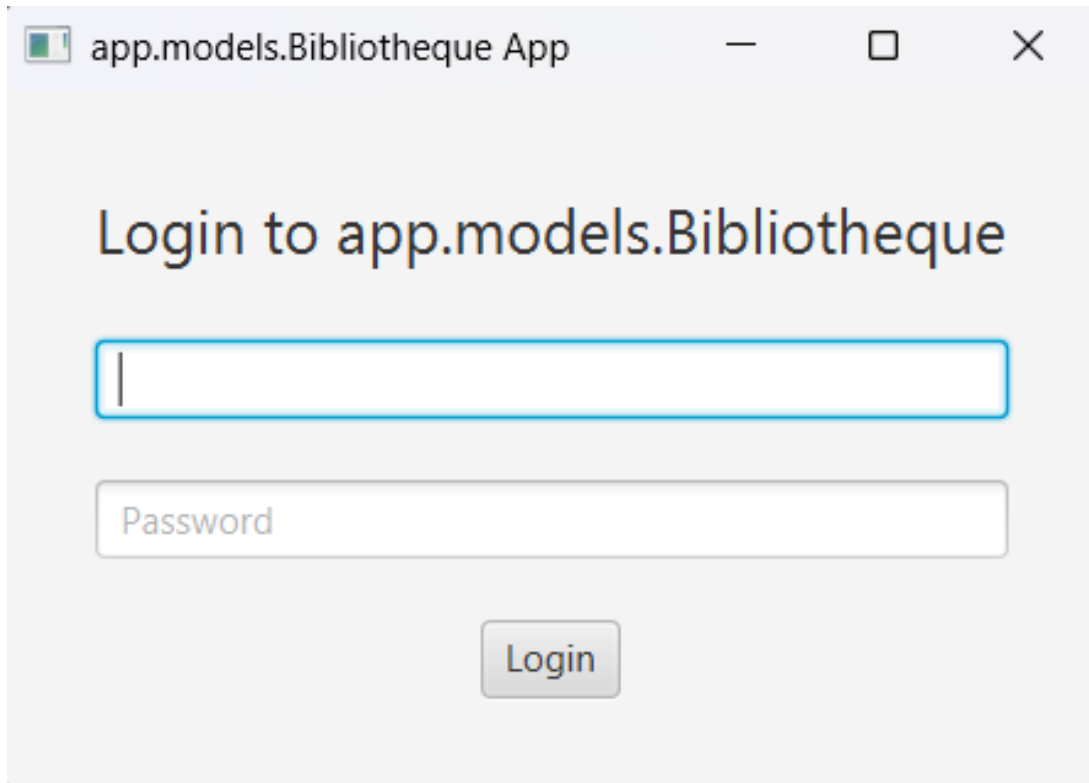


FIGURE 1.1 – Capture de l'écran de connexion

Cette capture d'écran montre la page d'authentification de l'application 'app.models.Bibliotheque', conçue pour deux types d'utilisateurs :

- **Lecteurs** : accès aux recherches de livres, emprunts, et compte personnel.
- **Administrateurs** : gestion complète des stocks, utilisateurs, et statistiques.

**Éléments visibles sur l'interface :**

- **Champ "Login"** : Identifiant unique (ex : email, matricule).
- **Champ "Password"** : Mot de passe sécurisé.
- **Bouton "Login"** : Valide la connexion et redirige vers le tableau de bord adapté (lecteur ou administrateur).

## 1.4.2 Etape 2

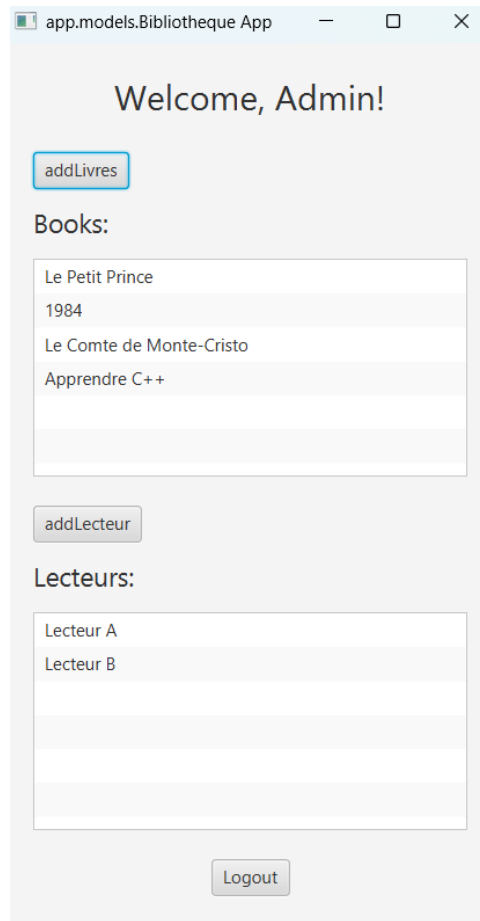


FIGURE 1.2 – Interface après connexion admin

pour connecte tant que `-ADMIN-USERNAME= " admin "` et `PASSWORD = "admin"`

L'application permet d'ajouter de nouveaux livres dans la base de données via une interface simple et intuitive.

- **Ajout de nouveaux livres** : accessible via un bouton `addLivres` ou un menu déroulant.
- **Saisie des métadonnées** : titre, auteur

## Gestion des Lecteurs

Les administrateurs peuvent consulter et gérer la liste des lecteurs inscrits dans le système.

- **Consultation de la liste des lecteurs** : affichage des comptes actifs.
- **Exemples de comptes** : Lecteur A, Lecteur B.

## Actions Globales

- **Déconnexion sécurisée (*Logout*)** : fermeture de session assurant la protection des accès administrateur.

### 1.4.3 Etape 3

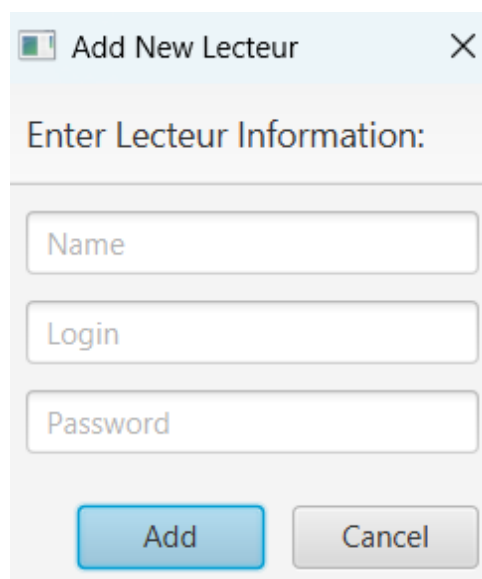


FIGURE 1.3 – ajouter un lecteur

L'interface d'ajout permet à un administrateur d'enregistrer un nouveau lecteur dans le système. Elle se compose des champs obligatoires et de boutons d'action.

#### 1. Champs requis

- **Name** : nom complet du lecteur).
- **Login** : identifiant unique (email ou matricule).
- **Password** : mot de passe temporaire

#### 2. Boutons d'action

- **Add** :
  - Valide l'inscription. .
- **Cancel** :

- Annule l'opération sans sauvegarder.

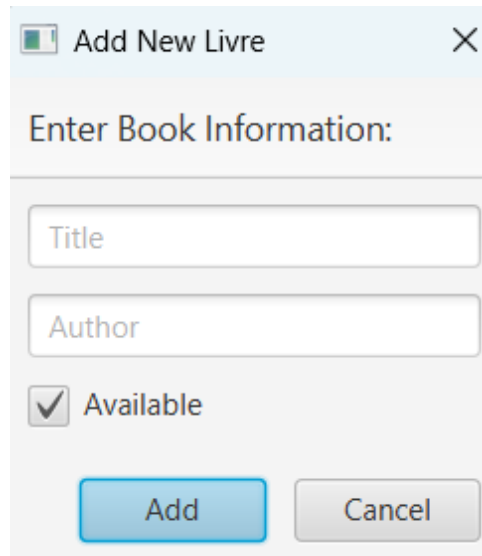


FIGURE 1.4 – ajouter un livre

### Champs requis

- **title** : titre de livre).
- **author** : author de livre (le nom complet de l'auteur).
- **Boutons d'action**
  - **Add** :
    - Valide l'inscription. .
  - **Cancel** :
    - Annule l'opération sans sauvegarder.



#### 1.4.4 Etape 4

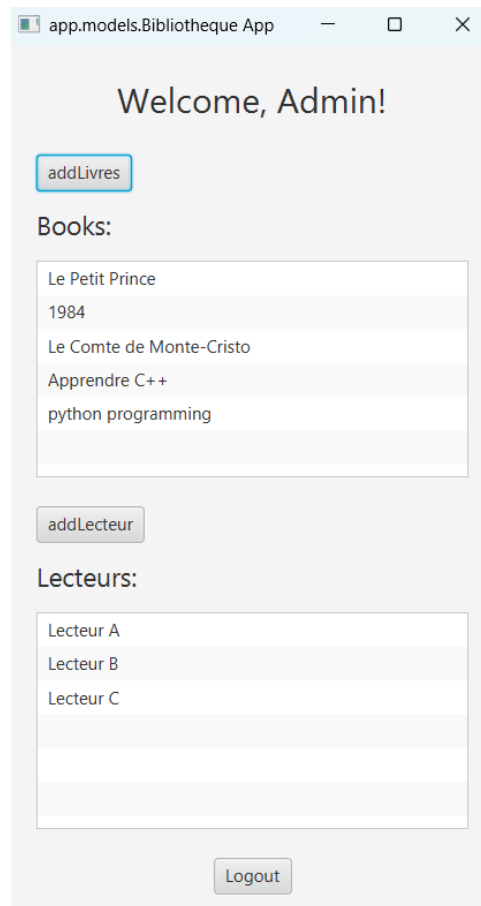


FIGURE 1.5 – dashbord apres ajouter livre et lecteur

#### 1.4.5 Etape 5

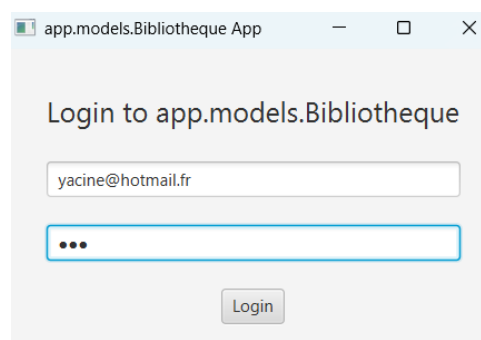


FIGURE 1.6 – connecte tantque lecteur

L'interface de connexion comprend les éléments suivants :

- **Email** : exemple utilisé — `yacine@hotmail.fr`.

- **password** : exemple utilisé — 123.
- **Bouton "Login"** :
  - Déclenche l'authentification.
  - Redirige vers l'interface lecteur après validation des identifiants.

### 1.4.6 Etape 6

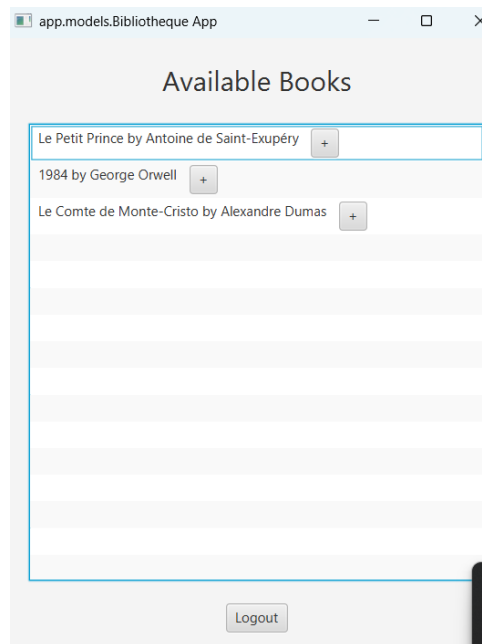


FIGURE 1.7 – Catalogue des livres disponibles

Cette section décrit les principales fonctionnalités accessibles à un lecteur connecté.

#### 1. Liste des livres

- **Affichage simplifié** : chaque livre est présenté avec son titre et son auteur (exemple : *Le Petit Prince* par *Antoine de Saint-Exupéry*).
- **Bouton "+"** : permet d'ajouter le livre à un panier d'emprunt, déclenchant une réservation immédiate.

#### 2. Fonctionnalités clés

- **Réservation en un clic** : rapide et intuitive, sans confirmation supplémentaire.
- **Bouton "Logout"** : déconnexion sécurisée de l'interface.

```

Livres empruntés: Le Petit Prince
Livres empruntés: 1984
Livres empruntés: Le Comte de Monte-Cristo

```

FIGURE 1.8 – livres empruntés

Cette section détaille les actions automatiques ou implicites liées à la gestion des emprunts.

- **Ajout instantané** : un simple clic sur le bouton "+" (dans l'interface des livres) ajoute automatiquement le livre à la liste des emprunts en cours.
- **Gestion des emprunts** :
  - Suivi des ouvrages empruntés par l'utilisateur.
  - **Durée de prêt** : affichage de la date de retour estimée pour chaque livre emprunté.

#### 1.4.7 Etape 7

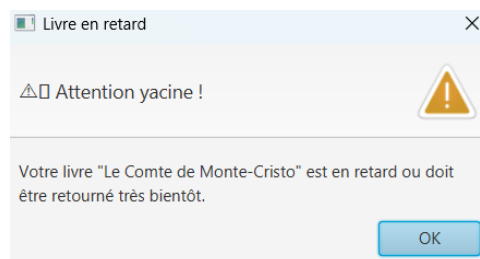


FIGURE 1.9 – notification

L'application notifie automatiquement les utilisateurs lorsqu'un livre approche de sa date limite de retour ou est déjà en retard.

- **Titre du message** : *"Livre en retard"* (correction automatique d'une faute fréquente : *"Live"* au lieu de *"Livre"*).
- **Destinataire** : message personnalisé — *"Attention Yacine!"*.
- **Détails du message** :
  - **Livre concerné** : *Le Comte de Monte-Cristo*.
  - **Statut** : *"En retard ou doit être retourné très bientôt"* (la formulation reste volontairement vague en l'absence de date précise).
- **Bouton "OK" Action** : ferme simplement la notification :

## 1.5 Conclusion :

Ce mini-projet de gestion de bibliothèque nous a permis de concevoir une application simple, mais fonctionnelle, répondant aux besoins essentiels d'un environnement de prêt de livres. Grâce à une interface intuitive et une séparation claire entre les rôles de lecteur et d'administrateur, l'outil facilite à la fois la consultation, la réservation et la gestion des ouvrages.

Ce travail a été l'occasion de mettre en pratique plusieurs compétences clés, notamment la structuration d'une base de données, la gestion des interfaces utilisateur, la logique d'authentification et le développement orienté utilisateur. Il pose ainsi les bases d'un système évolutif qui pourrait, à terme, être enrichi de nouvelles fonctionnalités telles que des rappels par email, la gestion des pénalités de retard ou encore une recherche avancée.

En résumé, ce projet constitue une excellente introduction au développement d'applications de gestion, tout en répondant à un besoin concret, compréhensible et utile.