Rapport

Projet : Gestion intelligente d'une bibliothèque en Scala

Rédigé par : Fidèle Ledoux KENFACK (IAS-M2-DS2)

Introduction

Dans un monde de plus en plus axé sur la gestion automatisée des informations, la capacité à développer des systèmes efficaces et évolutifs est une compétence essentielle. C'est dans cette dynamique que s'inscrit ce projet réalisé dans le cadre du Master 2 Informatique, pour le cours de Programmation en Scala.

Ce travail a pour ambition de mettre en œuvre les principes de la programmation fonctionnelle pour résoudre un problème courant : la gestion d'une bibliothèque. À travers ce projet, nous avons conçu une application interactive permettant de gérer efficacement les livres, les utilisateurs, les emprunts, les retours et les réservations.

En respectant des principes tels que l'immuabilité, l'usage de collections fonctionnelles et l'architecture modulaire, ce système offre à la fois flexibilité, lisibilité du code et robustesse dans son fonctionnement.

Ce rapport présente l'ensemble de la démarche suivie, les choix techniques effectués, ainsi que les fonctionnalités implémentées avec une approche pédagogique et orientée vers l'avenir. l'ensemble de la démarche suivie pour la réalisation du projet, depuis l'analyse des besoins jusqu'à l'élaboration d'un système fonctionnel et évolutif.

1. Objectifs du projet

Le projet a plusieurs objectifs pédagogiques et techniques :

Objectifs pédagogiques :

- Appliquer les concepts de programmation fonctionnelle en Scala : cela signifie utiliser des structures de données immuables, des fonctions sans effets de bord, et favoriser la composition fonctionnelle. Notre code repose sur des case class immuables et des fonctions de transformation, à l'image de l'usage de copy(...) pour créer un nouvel état de la bibliothèque sans modifier l'état précédent.
- Utiliser des collections immuables (List, Map, Set) : tous les éléments de données (livres, utilisateurs, emprunts, réservations) sont stockés dans des List Scala, et non dans des structures mutables. Par exemple, lorsqu'on emprunte un livre, on crée une nouvelle liste d'emprunts mise à jour, sans jamais altérer directement la liste existante.

- Structurer un projet en entités métiers et fonctions pures : les actions (emprunt, retour, réservation, etc.) sont toutes définies comme des méthodes dans la classe Bibliothèque, qui retourne une nouvelle instance à chaque modification. Cela suit le paradigme fonctionnel.
- Travailler avec des cas concrets : la gestion d'une bibliothèque est un domaine suffisamment riche pour expérimenter plusieurs logiques de programmation (vérifications, boucles, conditions, gestion d'erreurs, etc.).

Objectifs techniques :

- Développer un système de gestion complète de bibliothèque.
- Gérer l'ajout, l'emprunt, le retour, la réservation et le tri de livres.
- Implémenter un système de pénalités en cas de retard.
- Fournir une interface utilisateur textuelle interactive.

2. Conception et architecture :

✓ 2.1 Modélisation des entités

Le système repose sur les entités suivantes :

- Livre : contient un ID, un titre, un auteur, un genre, et une date d'ajout.
- Utilisateur : contient un ID et un nom.
- Emprunt : relie un livre à un utilisateur, avec une date de début et une date de fin optionnelle.
- Réservation : lie un livre à un utilisateur avec une date de réservation.

✓ 2.2 Architecture fonctionnelle

L'architecture du système est entièrement fonctionnelle, respectant les principes fondamentaux du langage Scala :

- Immuabilité : toutes les entités sont des case class, donc immuables par défaut. Lorsqu'une modification est nécessaire (ex. : ajout d'un livre), le système crée une nouvelle instance de Bibliothèque avec une nouvelle liste, via copy(...).
- Pas d'effets de bord : aucune méthode n'altère directement l'état de l'application ; elles retournent un nouvel état. Cela garantit la traçabilité des opérations et la prédictibilité du système.
- Encapsulation des comportements : toutes les opérations sont regroupées dans la classe Bibliothèque, ce qui permet de centraliser la logique métier. Par exemple, emprunter Livre, retourner Livre, réserver Livre sont des méthodes internes qui agissent uniquement par transformation de données.
- Utilisation des méthodes fonctionnelles Scala : map, filter, find, foreach sont largement utilisés pour effectuer des traitements sur les listes sans utiliser de boucles impératives.

- Gestion du temps et des dates : nous utilisons LocalDate pour gérer les dates d'emprunt et de retour, ainsi que la fonction Period.between(...) pour calculer les retards.

L'ensemble de ces choix rend le système robuste, évolutif et simple à maintenir.

3. Fonctionnalités principales

✓ 3.1 Ajouter ou supprimer un livre

Cette fonctionnalité permet à l'administrateur de gérer le catalogue de livres. Elle repose sur deux méthodes de la classe Bibliothèque : ajouter Livre et retirer Livre.

Exemple :

biblio = biblio.ajouter Livre(Livre(1, "1984", "George Orwell", "Roman"))

biblio = biblio.retirer Livre(1)

✓ 3.2 Inscrire un utilisateur

Permet d'enregistrer un nouvel utilisateur dans la bibliothèque.

biblio = biblio.inscrire Utilisateur(Utilisateur(1, "Fidèle Ledoux"))

✓ 3.3 Emprunter un livre

Un utilisateur peut emprunter un livre s'il est disponible.

biblio = biblio.emprunter Livre(1, 1, LocalDate.now)

✓ 3.4 Retourner un livre

Cette méthode met à jour l'état de l'emprunt et déclenche une réservation si elle existe.

biblio = biblio.retourner Livre(1, LocalDate.now.plusDays(16))

✓ 3.5 Réserver un livre

Un utilisateur peut réserver un livre s'il est déjà emprunté.

biblio = biblio.reserver Livre(1, 2, LocalDate.now)

✓ 3.6 Afficher les livres disponibles / empruntés

Permet de visualiser les livres selon leur statut d'emprunt.

biblio.livres Disponibles.foreach(println)

biblio.livres Empruntes.foreach(println)

✓ 3.7 Afficher les emprunts d'un utilisateur

Permet de suivre l'historique de lecture d'un utilisateur.

biblio.emprunts Par Utilisateur(1).foreach(println)

- ✓ **3.8 Vérification des retards**

Calcule et affiche les pénalités selon la durée maximale autorisée.

biblio.verifierRetards()

- ✓ **3.9 Tri des livres**

Tri dynamique par auteur, genre ou date d'ajout.

biblio.trier Livres Par Auteur()

biblio.trier Livres Par Genre()

biblio.trier Livres Par Date()

4. Interface utilisateur : menu interactif

L'interface se compose d'un menu console textuel qui s'affiche en boucle tant que l'utilisateur ne choisit pas de quitter.

Exemple d'options :

1. Ajouter un livre

2. Inscrire un utilisateur

3. Emprunter un livre

4. Retourner un livre

...

0. Quitter

Chaque option déclenche l'appel à une fonction spécifique dans l'objet Bibliothèque. Des messages clairs guident l'utilisateur dans ses choix.

5. Gestion des cas d'usage et traitement des erreurs

Notre application a été conçue de manière à simuler des cas réalistes de gestion d'une bibliothèque. Voici quelques scénarios d'usage traités par notre système, avec des solutions adaptées à chaque cas :

- Contrôle de disponibilité à l'emprunt : la méthode emprunter Livre vérifie automatiquement si un livre est déjà emprunté (sans date de retour) avant de permettre un nouvel emprunt. Si le livre n'est pas disponible, une réservation est suggérée.
- Réservation automatique après retour : lorsqu'un livre est retourné avec retourner Livre, le système vérifie s'il existe des réservations associées. Si oui, le livre est automatiquement attribué au premier utilisateur en file d'attente.
- Réservation en double interdite : grâce à un simple filtre dans réserver Livre, un utilisateur ne peut réserver un livre qu'une seule fois, évitant ainsi les abus.

- Messages d'erreur clairs : si un identifiant de livre ou d'utilisateur n'existe pas, ou si une action est impossible (ex. emprunter un livre déjà pris), des messages explicites s'affichent pour guider l'utilisateur.

Ces mécanismes de gestion rendent notre application robuste, fiable et intuitive à utiliser, même dans un environnement réel. Les emprunts ne peuvent se faire que si le livre est disponible.

- Les retours sont automatiquement suivis de l'emprunt par la première personne ayant réservé le livre.
- Un même utilisateur ne peut pas réserver plusieurs fois le même livre.
- Des messages sont affichés en cas d'erreur (choix invalide, livre inexistant, etc.).

6. Recommandations et améliorations futures

Au fil du développement, nous avons identifié plusieurs pistes d'enrichissement du projet, qui pourraient faire l'objet de futures itérations ou d'une extension en projet de fin d'étude.

- Interface graphique conviviale : le passage à une interface graphique (JavaFX ou Scala.js) offrirait une meilleure expérience utilisateur. Le menu texte actuel fonctionne bien pour les tests, mais une interface visuelle permettrait une navigation plus rapide et intuitive.
- Persistance des données : actuellement, les données sont en mémoire. Intégrer une base de données relationnelle (PostgreSQL, SQLite) ou NoSQL (MongoDB) permettrait de conserver l'état entre différentes sessions et d'étendre les fonctionnalités à un usage en réseau.
- Rapports automatiques : ajout de la génération de rapports PDF/CSV avec la liste des livres empruntés, les utilisateurs pénalisés, ou l'état du stock.
- Système d'authentification : ajout d'un identifiant et mot de passe pour chaque utilisateur, avec rôles (administrateur / lecteur).
- Recherche multicritère : permettre aux utilisateurs de chercher un livre selon le titre, l'auteur, le genre, ou la disponibilité, avec des filtres dynamiques.

Ces idées sont parfaitement réalisables en s'appuyant sur l'architecture modulaire déjà en place dans notre code, et pourraient enrichir le projet pour un usage professionnel ou public. d'amélioration peuvent être envisagées :

- Ajouter une interface graphique (JavaFX, Scala.js).
- Intégrer une base de données pour persister les données.
- Générer des rapports PDF ou fichiers CSV des emprunts et retards.
- Ajouter un système d'authentification par mot de passe.
- Permettre des recherches multicritères dans la liste des livres.

7. Présentation graphique et captures d'écran

Pour appuyer la démonstration de notre application et illustrer ses fonctionnalités clés, nous avons intégré des captures d'écran représentatives des différentes interfaces du programme. Ces visuels offrent un aperçu concret de l'interaction entre l'utilisateur et le système via le terminal, tout en mettant en valeur l'intuitivité de la navigation dans notre menu textuel.

Captures prévues :

Figure 1 : Interface du menu principal affichée au lancement de l'application

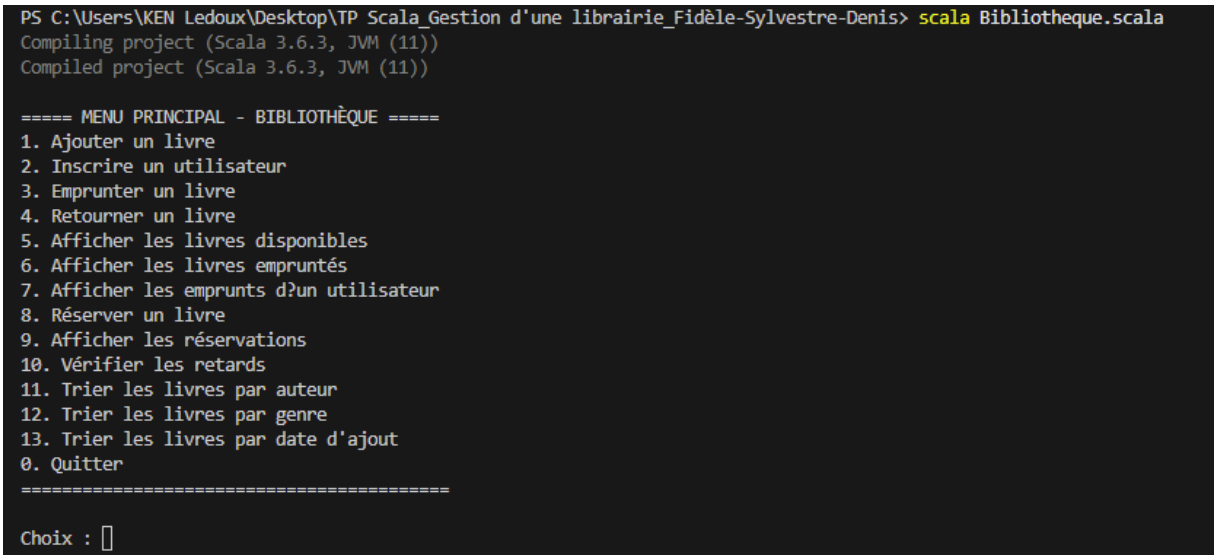


Figure 2 : Ajout d'un livre par l'utilisateur Sylvestre

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 1
ID : 1
Titre : LE POUVOIR DE VAINCRE
Auteur : Sylvestre KACOU
Genre : ACTION
Le livre 'LE POUVOIR DE VAINCRE' de Sylvestre KACOU (Genre : ACTION) a été ajouté avec succès.
```

Figure 3 : Inscrire un utilisateur

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 2
ID utilisateur : 2
Nom : Denis
Utilisateur Denis (ID: 2) inscrit avec succès.
```

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 2
ID utilisateur : 1
Nom : Fidèle Ledoux
Utilisateur Fid?le Ledoux (ID: 1) inscrit avec succès.
```

Figure 4 : Emprunt d'un livre par Fidèle Ledoux avec confirmation à l'écran

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 3
ID livre : 1
ID utilisateur : 1
Livre 1 emprunté avec succès par l'utilisateur 1.
```

Figure 5 : Message retour du livre emprunté

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 4
ID livre à retourner : 1
Livre 1 retourné avec succès.
```

Figure 6 : Affichage des livres disponibles

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 5
 Livres disponibles :
1 - Le Petit Prince
2 - 1984
3 - Les Misérables
```

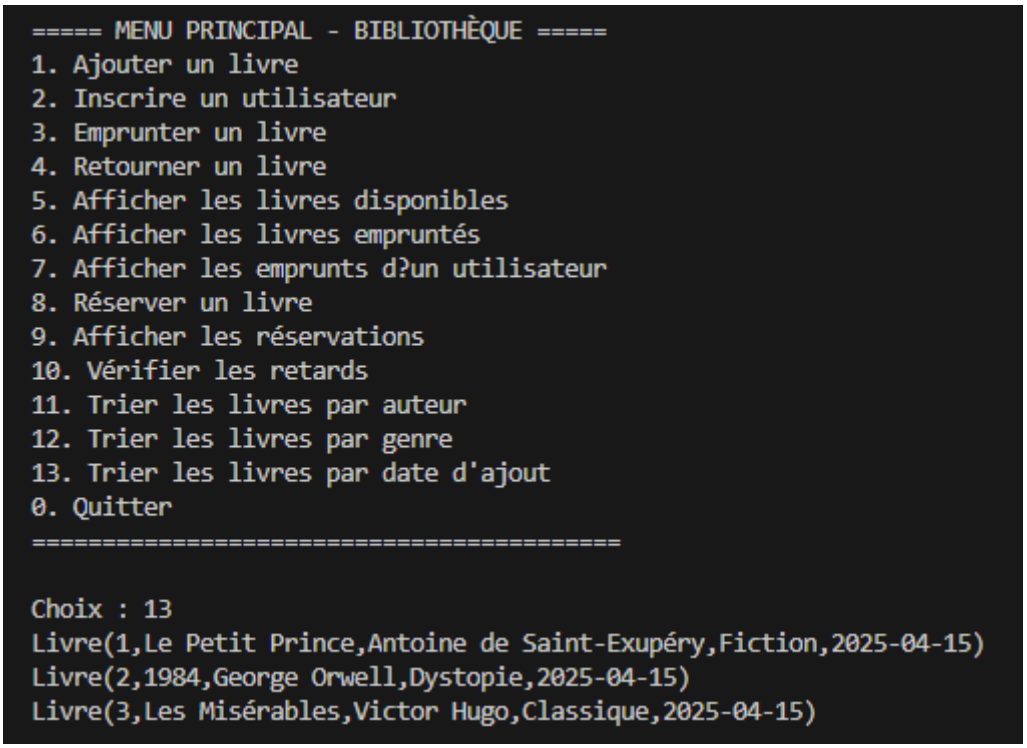
Figure 7 : Trier les livres par auteurs

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 11
Livre(1,Le Petit Prince,Antoine de Saint-Exupéry,Fiction,2025-04-15)
Livre(2,1984,George Orwell,Dystopie,2025-04-15)
Livre(3,Les Misérables,Victor Hugo,Classique,2025-04-15)
```

Figure 8 : Trie des livres par genres

```
===== MENU PRINCIPAL - BIBLIOTHÈQUE =====
1. Ajouter un livre
2. Inscrire un utilisateur
3. Emprunter un livre
4. Retourner un livre
5. Afficher les livres disponibles
6. Afficher les livres empruntés
7. Afficher les emprunts d?un utilisateur
8. Réserver un livre
9. Afficher les réservations
10. Vérifier les retards
11. Trier les livres par auteur
12. Trier les livres par genre
13. Trier les livres par date d'ajout
0. Quitter
========================================

Choix : 12
Livre(1,LE POUVOIR DE VAINCRE,Sylvestre KACOU,ACTION,2025-04-15)
Livre(3,Les Misérables,Victor Hugo,Classique,2025-04-15)
Livre(2,1984,George Orwell,Dystopie,2025-04-15)
Livre(1,Le Petit Prince,Antoine de Saint-Exupéry,Fiction,2025-04-15)
```

Figure 9 : Trier les livres par date d'ajout



Chaque capture met en avant un cas d'usage réel implémenté dans notre code et démontre la réactivité de notre logique métier. Nous avons volontairement opté pour une approche simple mais fonctionnelle qui permet une prise en main immédiate par l'utilisateur

Conclusion

La réalisation de ce projet fut une opportunité enrichissante, tant sur le plan technique que pédagogique. Nous avons pu explorer les atouts du langage Scala dans un cadre concret, en exploitant ses fonctionnalités puissantes comme les case class, la gestion de collections immuables, et la modélisation fonctionnelle.

Notre application va au-delà d'un simple système de prêt de livres : elle intègre des éléments de logique métier réelle (réservations automatiques, pénalités de retard, tri des données) tout en respectant les contraintes de modularité et de maintenabilité.

Ce projet a également renforcé notre esprit de collaboration et notre rigueur en développement logiciel. En tant qu'étudiant du Master 2, nous sommes fiers de présenter cette solution, qui illustre concrètement l'utilité de la programmation fonctionnelle pour concevoir des applications fiables et élégantes.

En perspective, nous souhaitons approfondir l'intégration d'une interface graphique, la persistance des données et la mise en réseau du système pour une application déployable à plus grande échelle.