REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie
UNIVERSITE DE DOUALA
ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE DE DOUALA
B.P. 2701 Douala
Tél. (237) 697 542 240
Site web : www.enspd-udo.cm

REPUBLIC OF CAMEROON
Peace – Work – Fatherland
THE UNIVERSITY OF DOUALA
NATIONAL HIGHER POLYTECHNIC SCHOOL OF DOUALA
P.O.Box:2701 Douala
Phone :(237) 697 542 240
Email: contact@enspd-udo.cm

EXPOSE DE PROGRAMMATION ORIENTE OBJET

THÈME : SYSTÈME DE GESTION D'UNE BIBLIOTHEQUE

N°	NOMS	MATRICULES
1	LEUKAM TCHEUMALEU MAXIME	24G01101
2	DJOGU KAMENI DARELLE	22G00078
3	FADIMATOU ABDOU	24G01090
4	NGACHILI NJANKOUO FRED	24G01119
5	KAMGA FOYANG DARYL	24G01095
6	NDOUNGUE ESSOH ABEL THOMAS	22G00295
7	TEZEM YEMETIO TEDDY MIGUEL	24G01133
8	NGANSO NOUKEU ALEX	22G00306
9	MBAIAMMADJI SYLVESTRE BANYO	24G01339
10	FANHE MEIDO STEVEN	22G00129

SOUS SUPERVISION DE :

Dr NOULAPEU

FILIERE : GIT/ GLO

NIVEAU :

ANNEE SCOLAIRE: 2024-2025

TABLE DES MATIERES

LISTES DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

La bibliothèque est un pilier fondamental de la vie quotidienne où convergent des penseurs et chercheurs passionnés. Pour que tout fonctionne bien dans cette dernière, il faut savoir quels livres sont disponibles, qui les a empruntés et quand ils doivent être rendus. Si tout cela est géré à main avec des papiers et des cahiers, cela peut vite devenir compliqué et prendre beaucoup de temps. Des erreurs peuvent se produire et il est difficile d'avoir une vue d'ensemble claire.

Face à ces défis, l'avènement de l'informatique offre des solutions prometteuses. L'automatisation des processus bibliothécaires permet non seulement de rationaliser les opérations quotidiennes, mais également d'améliorer la précision des données, de libérer le personnel pour des tâches à plus forte valeur ajoutée et ultimement d'optimiser l'expérience tant pour la bibliothécaire que pour l'usager. C'est dans ce contexte que s'inscrit un projet de développement d'un système de gestion d'une bibliothèque.

Nous présenterons la conception et la réalisation d'une application logicielle spécifiquement pensée pour répondre aux besoins d'une bibliothèque. L'objectif principal est de créer un outil intuitif et efficace capable de gérer l'enregistrement des ouvrages, le cycle de vie des prêts (de l'emprunt au retour) et la consultation de l'état de fond du documentaire. En adoptant une approche orientée objet et en utilisant le langage de programmation JAVA, ce projet vise à fournir une solution pérenne et acceptable aux contraintes spécifiques.

A travers tous les chapitres qui suivront, nous explorerons en détail l'analyse des besoins qui a guidé à la conception du système, l'architecture du logiciel retenue, les étapes de développement et d'implémentation, ainsi que les résultats obtenus et les perspectives d'évolution du projet. L'ambition est de montrer comment une application simple mais bien conçue peut apporter une contribution significative à l'amélioration de la gestion et au rayonnement d'une bibliothèque.

CHAPITRE I : PRESENTATION DU PROJET

I. DESCRIPTION DU PROJET

Le projet consiste à développer une application logicielle en utilisant le langage JAVA, pour la gestion des opérations courantes d'une bibliothèque. L'application permettra d'enregistrer de nouveaux livres, de gérer les prêts et les retours, de consulter l'inventaire des ouvrages et de visualiser l'historique des emprunts, les données seront stockées et récupérées à partir des fichiers.

II. CONTEXTE DU PROJET

Le projet s'inscrit dans le besoin d'automatiser et d'améliorer l'efficacité de la gestion d'une bibliothèque. Les méthodes manuelles ou l'utilisation d'outils non spécifiques peuvent entrainer des pertes de temps, des erreurs et des difficultés de suivi. Un système informatisé dédié permettra de centraliser les informations, de simplifier les processus et d'offrir une meilleure vue d'ensemble des activités de la bibliothèque.

III. OBJECTIFS DU PROJET

L'objectif principal est de développer une application conviviale et fonctionnelle pour la gestion d'une bibliothèque. Les objectifs spécifiques incluent :

- Permettre l'enregistrement, la modification et la suppression des informations des livres.
- Faciliter l'enregistrement des prêts et des retours ouvrages.
- Offrir la possibilité de consulter l'inventaire des livres disponibles.
- Fournir un historique des prêts par livre et par utilisateurs.
- Utiliser une interface utilisateur simple et navigable.
- Assurer la persistance des données à l'aide des fichiers.
- Produire une documentation complète du projet (architecture, manuel utilisateur).

IV. CIBLES DU PROJET

- **Bibliothécaire** : leur fournir un outil simple pour gérer les livres, les emprunts et les retours sans complexité technique.
- **Etudiants en génie logiciel** : les initier aux bonnes pratiques de développement d'applications JAVA, à la modélisation UML, à la gestion de projet, et à la documentation technique.
- **Enseignants ou encadreurs pédagogiques** : utiliser ce projet comme support de cours ou évaluation pour les modules de programmation orientée objet, de génie logiciel ou d'analyse/conception.
- **Bibliothèques scolaires ou privées** : leur offrir une solution logicielle personnalisable et légère pour automatiser la gestion de leurs ressources.

CHAPITRE II : ÉTUDE DE L'EXISTANT

I - PRÉSENTATION DE L'EXISTANT

Actuellement, la gestion des bibliothèques repose sur des méthodes traditionnelles ou des outils logiciels. Présentons une analyse détaillée des systèmes couramment utilisés :

1- Systèmes manuels basés sur papier

- Certaines bibliothèques conservent des registres physiques où les livres et les emprunts sont notés à la main.
- Cette méthode est archaïque, surjette aux erreurs et rend difficile la recherche d'informations.

2- Fichiers Excel ou tableurs simples

- Certains établissements utilisent des feuilles de calcul pour suivre les livres et les emprunteurs.
- Bien que plus organisé que le papier, ce système manque d'automatisation et de fonctionnalités avancées.

3- Logiciels de gestion basique

- Certaines bibliothèques emploient des logiciels simplistes permettant d'enregistrer les livres et les emprunts.
- Ces outils sont souvent rigides, avec des interfaces peu intuitives et aucune sauvegarde fiable des données.

4- Absence de centralisation des données

- Les informations sont souvent dispersées entre différents supports (papier, fichiers numériques), ce qui complique leur gestion et leur mise à jour.

Cette situation montre un besoin évident d'une solution plus structurée, automatisée et conviviale.

II - ANALYSE DES FORCES ET FAIBLESSES DE L'EXISTANT

A. Forces des systèmes actuels

- **Simplicité d'utilisation** : les méthodes manuelles ne nécessitent pas de compétences techniques particulières.
- **Coût initial faible** : les registres papier ou les fichiers Excel ne demandent pas d'investissement logiciel important.
- **Indépendance technologique :** ces systèmes fonctionnent sans dépendre d'une infrastructure informatique complexe.
- **Adaptabilité minimale** : les utilisateurs peuvent modifier facilement les entrées sans contraintes techniques.

B. Faiblesses majeures

- **Gestion chronophage et inefficace** : la saisie manuelle ralentit les opérations et augmente le risque d'erreurs (doublons, oublis).
- **Pas de sauvegarde sécurisée** : les données peuvent être perdues en cas de détérioration du papier ou de corruption des fichiers numériques.
- **Interface peu intuitive** : les outils existants ne guident pas l'utilisateur, rendant leur prise en main difficile.
- **Suivi des emprunts imprécis** : il est compliqué de vérifier les retards, les disponibilités ou l'historique des prêts.

III - IDENTIFICATION DES MENACES ET DES OPPORTUNITÉS D'AMÉLIORATION

A. Problématiques clés à résoudre

- **Saisie manuelle fastidieuse :** nécessité d'automatiser l'ajout, la modification et la suppression des livres.
- **Absence de persistance des données** : besoin d'un système qui sauvegarde les informations entre deux utilisations du logiciel.

- **Manque d'ergonomie** : une interface plus claire et interactive améliorerait l'expérience utilisateur.
- **Gestion approximative des emprunts** : difficulté à suivre les retours, les retards et les statistiques d'emprunt.

B. Opportunités d'évolution

- **Développement d'une application dédiée** : créer un outil sur mesure en Java pour répondre aux besoins spécifiques des bibliothèques.
- **Sauvegarde automatisée dans des fichiers (CSV, etc.)** : permettre une conservation sécurisée et une restauration facile des données.
- **Menu interactif et convivial** : proposer une navigation intuitive avec des messages d'aide et des validations de saisie.
- **Fonctionnalités avancées** : historique des emprunts, recherche multicritère, alertes pour les retards, etc.

CHAPITRE III : CAHIER DE CHARGES ET CONCEPTION

SECTION I : CAHIER DE CHARGES

I. EXIGENCES FONCTIONNELLES

1- Gestion des livres

1.1- Ajouter un livre

- L'application demande et enregistre les informations suivantes : Titre (obligatoire), Auteur (obligatoire), ISBN (format à valider), Année de publication, Edition, Genre (liste prédéfinie ou saisie libre), Nombre d'exemplaires.
- L'application généré un identifiant unique pour chaque livre ajouté.
- L'application vérifie si un livre avec le même ISBN existe déjà et propose une action appropriée (modifier l'exemplaire existant ou crée un nouvel exemplaire).

1.2- Modifier un livre

- L'application permet de rechercher un livre par son identifiant unique ou ISBN.
- Une fois le livre trouvé, l'application affiche les informations actuelles et permet à l'utilisateur de modifier n'importe quel champ.

1.3- Supprimer un livre

- L'application demande une confirmation avant de procéder à la suppression (logique ou physique).
- Pour la suppression logique, le livre est marqué comme supprimé mais reste dans la base de données avec une indication de la raison de la suppression.
- Pour la suppression physique, le livre est retiré définitivement de la base de données (avec avertissement).

1.5- Consulter l'inventaire

- L'application doit afficher la liste de tous les livres enregistrés avec au moins le titre, l'auteur de la disponibilité (nombre d'exemplaires disponibles).

- L’application offre des options de filtrage : par titre (partiel ou exact), par auteur (partiel ou exact), par genre par date d’acquisition (ascendance, descendance).

2- Gestion des emprunts

1.1- Enregistrer un emprunt

- L’application demande l’identifiant unique de l’utilisateur et l’identifiant unique du livre à emprunter.
- L’application vérifie si le demandeur de l’emprunt est un adhérent de la bibliothèque et si le livre est disponible (au moins un exemplaire non emprunté).
- L’application enregistre la date de l’emprunt.
- L’application calcule et affiche la date retour preuve (par exemple, 3 Jours après l’emprunt).
- L'application met à jour le statut du livre (nombre d'exemplaires disponibles diminué).
- L'application enregistre l'emprunt dans un fichier dédié.

1.2- Enregistrer un Retour

- L'application demande l'identifiant unique de l'utilisateur et/ou l'identifiant unique du livre retourné.
- L'application recherche l'enregistrement de l'emprunt correspondant.
- L'application enregistre la date de retour effective.
- L'application met à jour le statut du livre (nombre d'exemplaires disponibles augmenté) .
- L'application calcule le nombre de jours de retard (si la date de retour effective est postérieure à la date de retour prévue) et afficher un message informatif.

1.3- Prolonger un Emprunt (Optionnel)

- L'application permet de rechercher un emprunt par l'identifiant de l'utilisateur et/ou du livre.
- L'application affiche la date de retour prévue actuelle et permettre de saisir une nouvelle date de retour (dans certaines limites, par exemple, une seule prolongation possible).
- L'application met à jour la date de retour prévue dans l'enregistrement de l'emprunt.

1.4- Consulter l'Historique des Prêts

- Par Livre : L'application permet de rechercher un livre par son identifiant unique ou son ISBN et afficher la liste de tous les emprunts passés et présents pour ce livre (utilisateur, date d'emprunt, date de retour prévu, date de retour effective).
- Par Utilisateur : L'application permet de rechercher un utilisateur par son identifiant unique et afficher la liste de tous les livres qu'il a emprunté (date d'emprunt, date de retour prévu, date de retour effective).

2- Interface Utilisateur

- Menu principal clair avec des options numérotées ou textuelles pour accéder aux différentes fonctionnalités.
- Messages d'information et d'erreur explicites et faciles à comprendre.
- Gestion des entrées utilisateur : lecture des saisies, suppression des espaces superflus, conversion des types de données si nécessaire.
- Validation des entrées : vérification du format (ISBN, date), vérification de l'existence des identifiants (livre, utilisateur).

3- Persistance des Données

- Structure détaillée des fichiers texte pour le stockage des livres (champs et ordre), des utilisateurs (champs et ordre) et des prêts (champs et ordre).
- Description des opérations de lecture des fichiers au démarrage de l'application pour charger les données en mémoire.
- Description des opérations d'écriture dans les fichiers pour sauvegarder les modifications (à la fermeture ou sur commande).
- Gestion des erreurs potentielles lors de la lecture et de l'écriture des fichiers (fichier inexistant, format incorrect).

4- Rapports Simples

- Livres Actuellement Empruntés : Afficher la liste des livres qui ne sont pas encore retournés, avec le nom de l'emprunteur et la date de retour prévue.
- Livres en Retard : Afficher la liste des livres dont la date de retour prévue est dépassée, avec le nom de l'emprunteur et le nombre de jours de retard.

- Livres Disponibles : Afficher la liste des livres ayant au moins un exemplaire disponible.

II. EXIGENCES NON FONCTIONNELLES

1- Performance

L'application est réactive pour une base de données (quelques centaines de livres et d'utilisateurs). Les opérations courantes (recherche, ajout, emprunt, retour) s'effectuent en un temps raisonnable (quelques secondes au maximum).

2- Utilisabilité

- L'interface en mode console est claire, intuitive et facile à naviguer, même pour un utilisateur non expert en informatique.
- Les messages d'aide et les instructions sont concis et pertinents.

3- Fiabilité

- L'application gère les erreurs de saisie de manière robuste et évite les plantages.
- Les données sont sauvegardées de manière fiable.

4- Portabilité

Étant développé en Java, l'application doit être portable sur différents systèmes d'exploitation (Windows, MacOs, Linux) à condition qu'une machine virtuelle Java (JVM) soit installée.

5- Maintenabilité

Le code source est bien structuré, commenté et respecte les conventions de codage Java pour faciliter la maintenance et les évolutions futures.

6- Convivialité

L'application est conviviale et facile à utiliser avec une interface utilisateur intuitive.

7- Confidentialité de données

L'application garantie la confidentialité et la sécurité des données de la bibliothèque et des usagers.

8- Sauvegarde et récupération des données

L'application offre des fonctionnalités de sauvegarde régulière des données et la possibilité de les restaurer en cas de perte.

9- Evolutivité

L'application est conçue pour être évolutive afin de pouvoir prendre en charge un nombre croissant d'utilisateurs et de transactions avec des performances optimales.

III. EXIGENCES TECHNIQUES

- Langage : Java (version 8 ou supérieure)
- Méthodologie : Approche orientée objet
- Outils : IDE (NetBeans, Eclipse ou IntelliJ IDEA)
- Versioning (Optionnel mais recommandé) : Git

IV. SECURITE

Pour ce projet, la sécurité des données au sens strict (cryptage, gestion des accès avancées) n'est pas une priorité majeure, mais il est important d'éviter la perte accidentelle de données.

V. ANALYSE DES RISQUES

- Le retard dans les délais.
- Le manque de compétences techniques.
- La faible adoption de l'application.
- Les problèmes de sécurité.
- L'intégration complexe.

VI. LIVRABLES ATTENDUS

1. Code Source

- Code Java complet, bien organisé en classes et packages.
- Commentaires clairs et pertinents expliquant la logique du code.
- Respect des conventions de nommage Java.

2. Documentation Technique

- Rapport d'architecture décrivant les classes, leurs responsabilités et leurs interactions (diagramme de classes UML).
- Explication des choix techniques (pourquoi Java, pourquoi les fichiers texte/CSV, structure des fichiers).
- Diagrammes de cas d'utilisation illustrant les principales interactions utilisateur-système.

3. Manuel Utilisateur

- Guide d'installation (si nécessaire).
- Instructions détaillées pour l'utilisation de chaque fonctionnalité de l'application (avec des exemples).
- Description du menu principal et de la navigation.
- Conseils pour la gestion des erreurs courantes.

4- Rapport de Projet

- Description détaillée des étapes du développement.
- Répartition des tâches (si travail en équipe).
- Bilan des difficultés rencontrées et des solutions apportées.
- Auto-évaluation du projet par rapport aux objectifs initiaux.

SECTION II : CONCEPTION

I- LANGAGE DE MODELISATION : UML

Le langage de Modélisation Unifié traduit de l'anglais Unified Modeling Language est un langage de modélisation standardisé utilisé pour spécifier, visualiser, construire et documenter les systèmes logiciels. Contrairement à MERISE, il s'appuie sur les diagrammes pour permettre de représenter graphiquement et de communiquer les divers aspects du système d'information. Il se compose de quatorze diagrammes.

Un diagramme est un élément graphique qui décrit une vue. On distingue trois grands groupes de diagrammes :

- **Diagrammes structurels ou diagrammes statiques :** ils illustrent la structure statique d'un système et comprennent les diagrammes de classes, d'objets, de composants, de déploiement, de paquetages, de structures composites et de profil.
- **Diagrammes comportementaux ou diagrammes dynamiques :** ils représentent les interactions et le comportement dynamique d'un système et comprennent les diagrammes de cas d'utilisation, d'activités et d'états-transitions.
- **Diagrammes d'interaction :** ils se concentrent sur les interactions entre les objets et comprennent les diagrammes de séquence, de communication, de temps et le diagramme global d'interaction.

Avantages : le langage UML est formel et normalisé, il est donc facilement utilisable, de plus il est axé sur l'utilisation d'outils pour plus de précision. Le fait d'être graphique rend la compréhension de système abstraits et complexes mieux cernables. Il s'adapte facilement à la programmation.

Inconvénients : l'apprentissage nécessaire pour avoir une bonne maitrise du langage et des outils est assez important. Pour fonctionner correctement, diagrammes UML doivent être synchronisés avec le code du logiciel, qui nécessite du temps pour mettre en place et à entretenir, et ajoute du travail à un projet de développement logiciel.

II- PROCESSUS DE DEVELOPPEMENT

La méthodologie adoptée dans l'analyse et la conception des systèmes représente de nos jours un choix stratégique pour l'équipe projet, leur permettant de mener à bien et à terme les projets tout en respectant les délais annoncés au client, la qualité du logiciel attendu et les coûts. Vu l'évolution des besoins et des exigences des utilisateurs finaux, les applications d'entreprises deviennent de plus en plus complexes, difficiles à concevoir et à développer. Pour la conception, le développement et la réalisation de notre application, nous avons opté pour le processus de développement en « V » qui demeure actuellement le cycle de vie le plus connu et certainement le plus convenable aux projets complexes.

Ce processus nous a accompagné du début du projet jusqu'à son implémentation. Ci-dessous, sont illustrées les différentes phases du modèle en V :

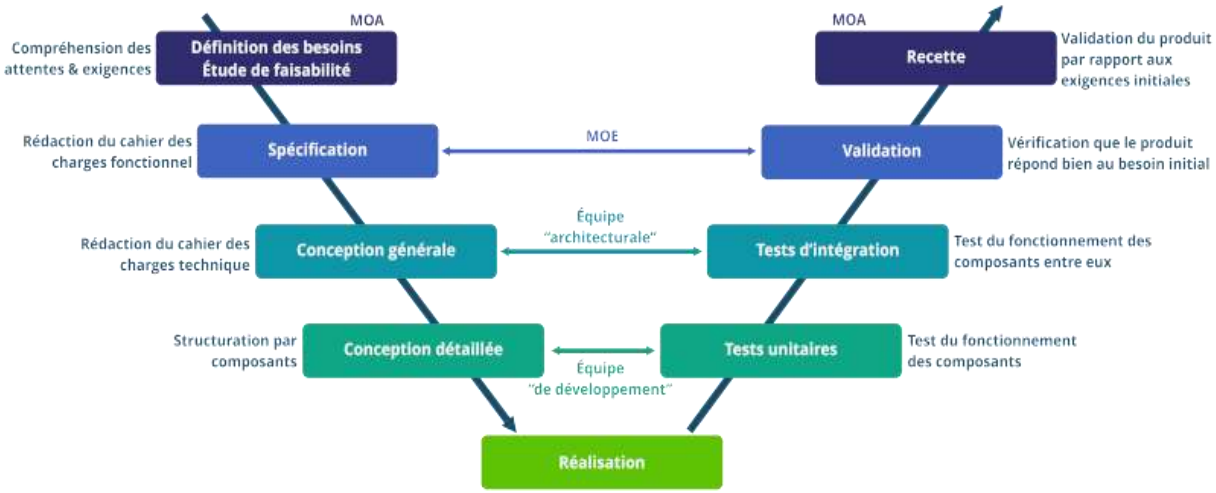

Figure 1: Cycle de vie en V

III- MODELISATIONS

A- DESCRIPTION FONCTIONNELLE

1- Identification des acteurs

Par définition, un acteur est un élément externe qui interagit directement avec le système. Cet élément peut être un utilisateur ou un système (autre ordinateur, programme, base de données). Dans notre cas, nous aurons comme acteurs:

- **Les adhérents de la bibliothèque :** ils ont la possibilité de consulter les livres présents dans la bibliothèque et d'emprunter des livres de leur choix en fonction de leur disponibilité.
- **L'administrateur de la bibliothèque :** il s'occupe de la gestion des comptes utilisateurs, de la base de données, de la gestion des livres, de la gestion des emprunts et des retours.

2- Cas d'utilisation

Un cas d'utilisation est un artefact qui définit une séquence d'actions qui produisent un résultat concret pour la valeur. En fonction des acteurs, nous avons pu ressortir les cas d'utilisations suivants :

Tableau 1:Cas d'utilisation

Acteurs	Cas d'utilisations
Les adhérents de la bibliothèque	• Consulter les livres • Emprunter un livre • Retourner un livre
L'administrateur de la bibliothèque(bibliothécaire)	• Ajouter un livre • Supprimer et modifier un livre • Gérer les pénalités • Consulter l'inventaire

3- Diagramme de cas d'utilisation

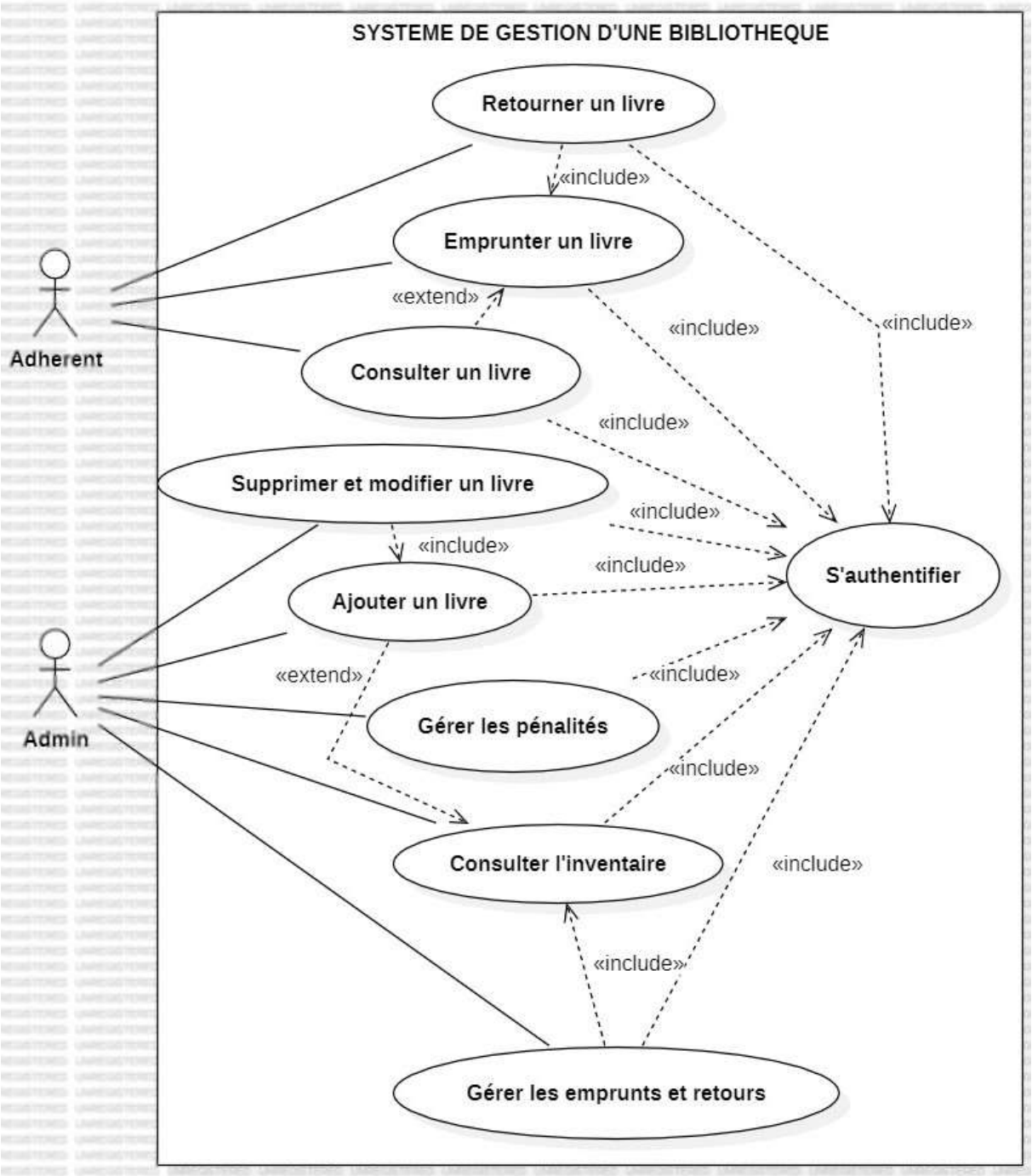

Figure 2: Diagramme de cas d'utilisation

B- MODÉLISATION STATIQUE : DIAGRAMME DE CLASSES

Le diagramme de classes représente clairement la structure d'un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets. Très utilisés par les ingénieurs logiciels pour documenter l'architecture des logiciels, les diagrammes de classe sont un type de diagramme de structure qui décrit ce qui doit être présent dans le système modélisé.

1- Illustration du diagramme de classes

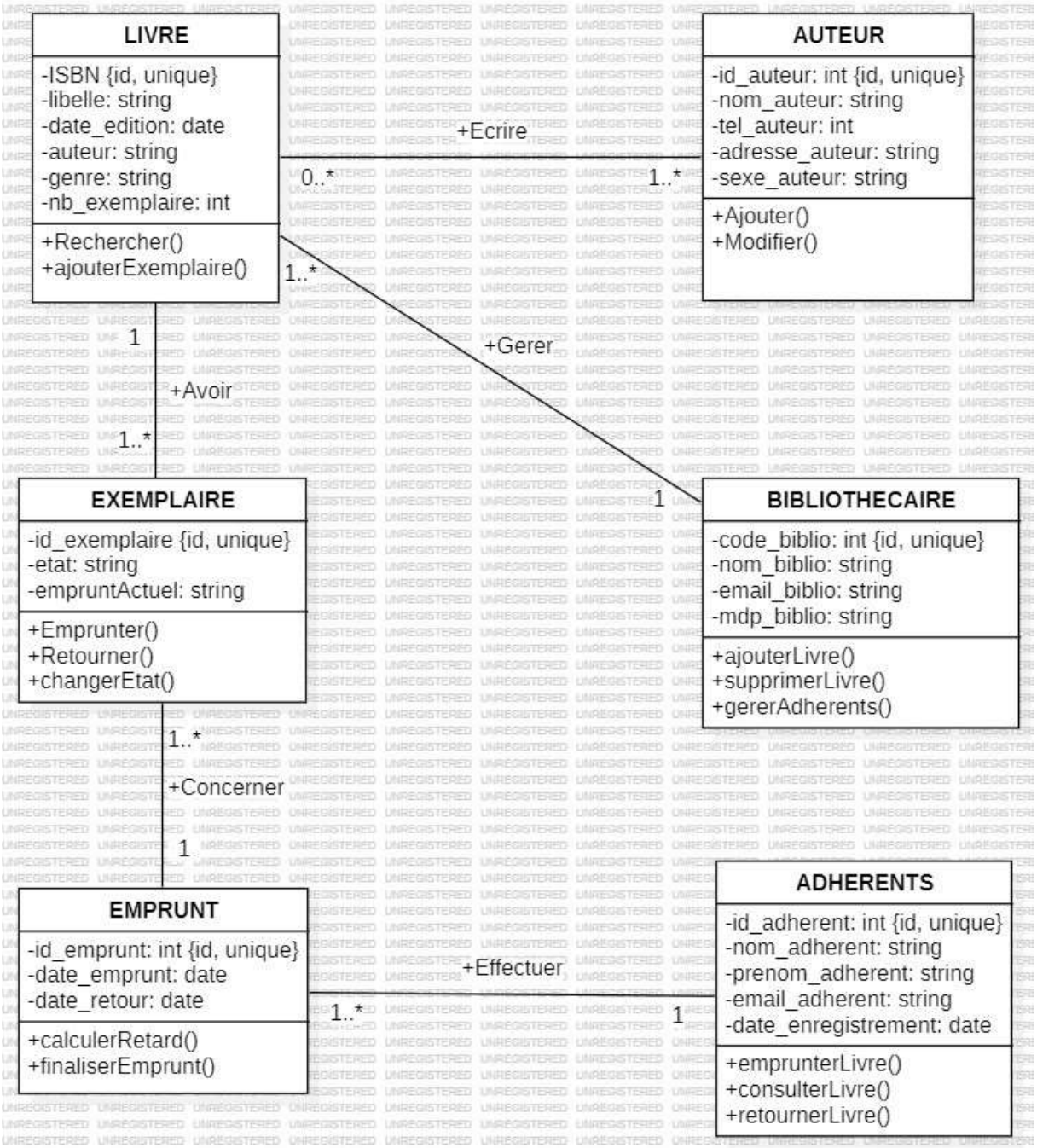

Figure 3: Diagrammes de classe

C- MODÉLISATIONS DYNAMIQUES : DIAGRAMMES DE SÉQUENCES

Le diagramme de séquence est un type de diagramme utilisé en génie logiciel pour modéliser la séquence d'interactions entre les objets d'un système au fil du temps. Il met l'accent sur la chronologie des messages échangés entre les objets pour accomplir une certaine fonctionnalité.

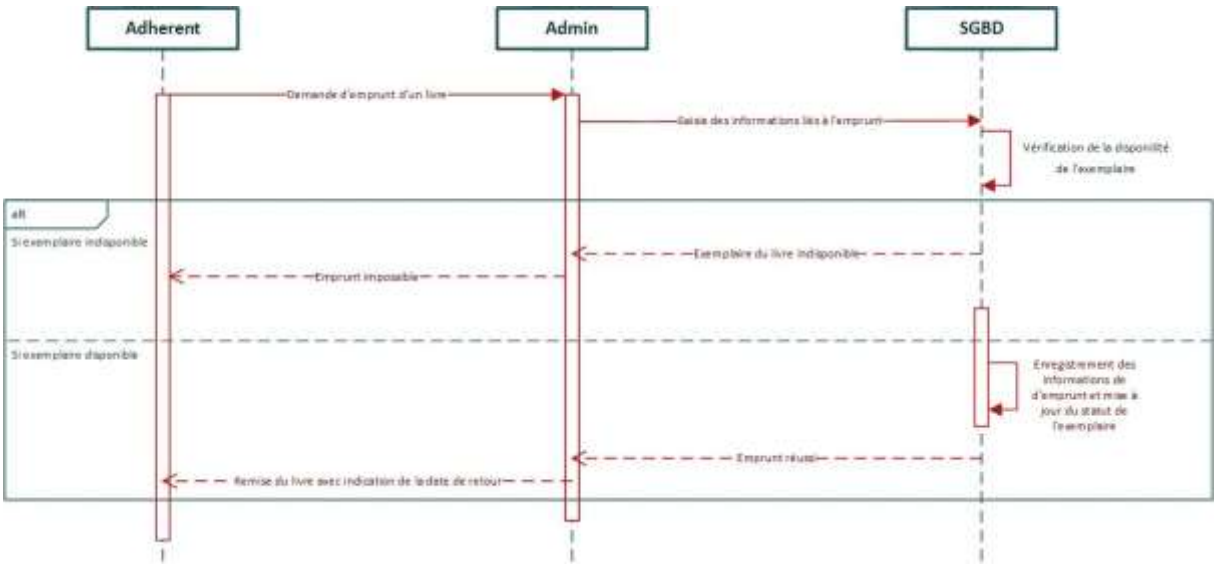

Figure 4: Diagramme de séquence d'emprunt d'un livre

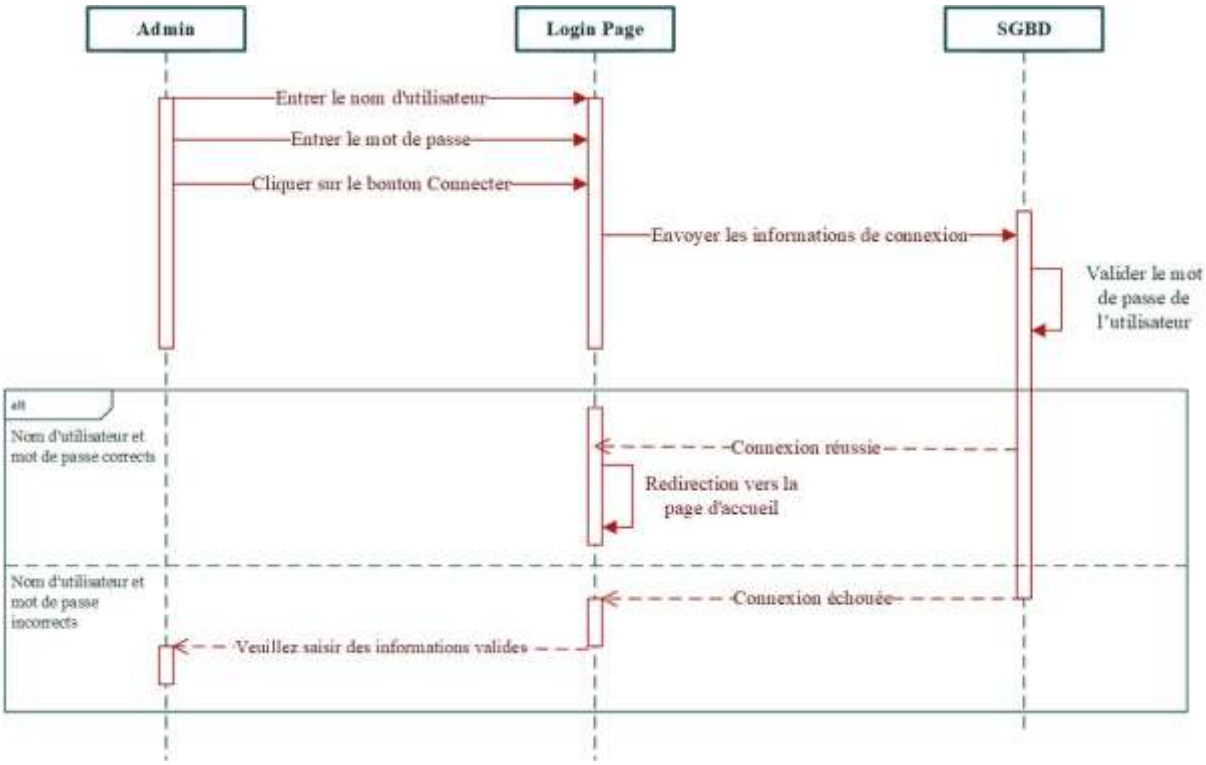

Figure 5: Diagramme de séquences de l'authentification

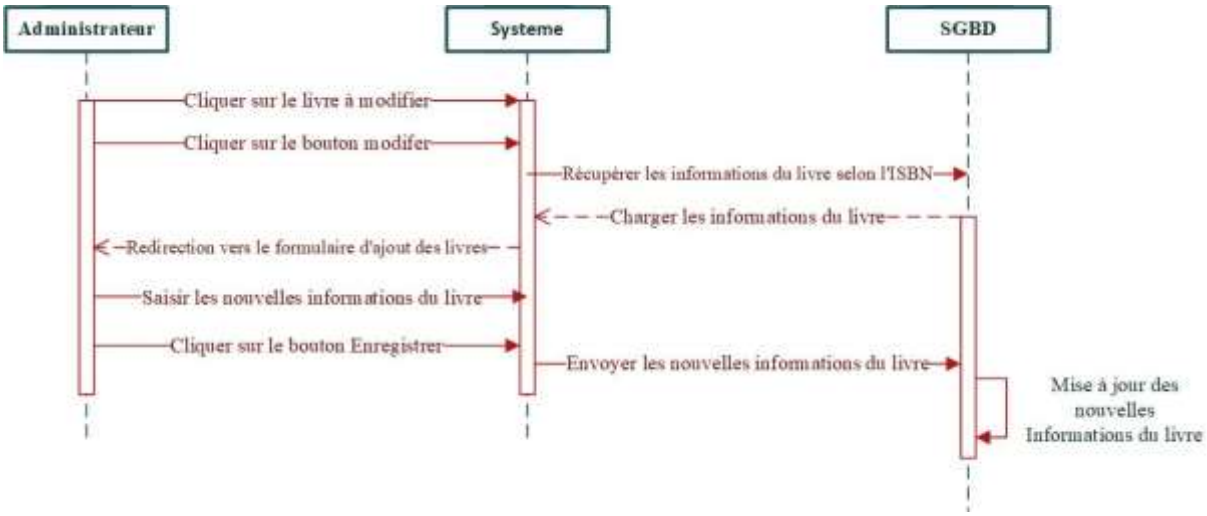

Figure 6: Diagramme de séquences de modification d'un livre

IV- ARCHITECTURE DE L'APPLICATION

L'architecture 1 tier ou architecture monolithique est un modèle où tous les composants d'une application (interface utilisateur, logique métier et gestion de base de données) sont intégrés dans une seule couche. Cela signifie que tous les éléments fonctionnent sur un même niveau, souvent sur une seule machine ou serveur. Comme caractéristiques principales :

- **Couche unique :** tous les éléments de l'application coexistent dans une seule couche.
- **Déploiement simple :** facilité de déploiement car tout est regroupé.
- **Accès rapide :** pas de latence réseau entre les différentes couches.

Les avantages de cette architecture sont entre autres :

- **La simplicité :** développement et maintenance simplifiées.
- **Le coût :** moins cher en termes d'infrastructures et de maintenance.
- **L'accès direct :** interaction directe avec la base de données.

CHAPITRE IV : IMPLEMENTATION, INTERPRETATION ET DISCUSSIONS DES RESULTATS

SECTION I : ENVIRONNEMENT DE TRAVAIL

I. REALISATION DU TRAVAIL

L'implémentation du système de gestion de bibliothèque a été réalisée dans un environnement' de développement structuré et méthodique. Le processus de développement s'est déroulé en plusieurs phases distinctes, chacune correspondant aux modules fonctionnels identifiés lors de la phase d'analyse.

- Méthodologie de développement :
  - Approche itérative avec cycles de développement courts.
  - Tests unitaires intégrés à chaque étape.
  - Validation continue avec les parties prenantes.
  - Documentation parallèle au développement.
- Organisation de travail : Le développement s'est organisé autour de sprint permettant une livraison progressive des fonctionnalités. Cette approche agile a facilité l'adaptation aux retours utilisateurs et l'ajustement des priorités en cours de projet.

1. Outils de développement

- Environnement de développement intégré (IDE) :
  - NetBeans
  - Intégration Git pour le versioning
  - Debugger intégré pour le débogage en temps réel.
- Système de gestion de versions :
  - Git pour le contrôle de version
  - GitHub pour l'hébergement
  - Branches de développement pour chaque fonctionnalité
- Base de données :
  - MySQL pour la modélisation et l'administration.

- Backup automatisé quotidien.
- Index optimisés pour les requêtes fréquentes.

2. Choix technologiques

➢ Langage de programmation : JAVA pour plusieurs raisons :

- Le langage de notre cours en salle.
- Syntaxe claire favorisant la maintenance.
- Excellent support pour les applications web.
- Communauté active et documentation extensive.

SECTION II : TESTS, INTERPRETATION ET DISCUSSIONS DES RESULTATS

Dans cette partie, nous parlerons d'une série de tests que nous avons effectués sur l'application afin de vérifier son bon fonctionnement et la fiabilité de ses fonctionnalités. De ce fait, nous avons documenté chacun de ces tests de manière détaillée, en fournissant des explications claires.

I- TEST FONCTIONNELS

Le test logiciel est une activité essentielle du processus de développement de logiciels qui vise à évaluer la qualité et le bon fonctionnement d'un logiciel. Les avantages du test comprennent la prévention des bogues, la rédaction des coûts de développement et l'amélioration des performances. Dans le cadre de notre phase de test, nous évaluerons les fonctionnalités ci-après :

- L'authentification
- L'ajout d'un livre
- La modification d'un livre
- L'enregistrement des emprunts
- La consultation de l'inventaire

1- Authentification

Prérequis : posséder un compte en tant qu'administrateur

Environnement de test : ordinateur portable

Testeur : les membres de ce groupe

Tableau 2: Test d'authentification

N°	Action	Attendu	Résultat
1	Exécutez l'application	Affichage de l'interface de connexion / inscription	OK
2	Cliquer sur le bouton Sign In	Affichage du formulaire de connexion	OK
3	Remplir des informations erronées	Message d'erreur	OK

4	Remplir des informations correctes	Accès à la page d'accueil	OK

2- Ajout d'un livre

Prérequis : posséder un compte en tant qu'administrateur

Environnement de test : ordinateur portable

Testeur : les membres de ce groupe

Tableau 3: Test d'ajout d'un livre

N°	Action	Attendu	Résultat
1	Cliquer sur l'onglet ManageBook	Affichage de la liste des livres	**OK**
2	Cliquer sur « Add Books »	Affichage du formulaire d'ajout des livres	OK
3	Remplir des valeurs similaires ou existantes	Message d'erreur	OK
4	Remplir des valeurs nouvelles	Enregistrement dans la base de données	OK

3- Modification d'un livre

Prérequis : posséder un compte en tant qu'administrateur

Environnement de test : ordinateur portable

Testeur : les membres de ce groupe

Tableau 4: Test de modification d'un livre

N°	Action	Attendu	Résultat
1	Cliquer sur l'onglet ManageBook	Affichage du da la liste des livres	**OK**
2	Cliquer sur le bouton «Modifier » dans la colonne « Action » du livre à modifier	Affichage du formulaire de modification avec les informations actuelles du livre	OK
3	Modifier la valeur des champs et cliquer sur « Update »	Mise à jour dans la base de données	OK

4- Enregistrement des emprunts

Prérequis : posséder un compte en tant qu'administrateur pour le bibliothécaire et être un adhérent à la bibliothèque pour le demandeur de l'emprunt

Environnement de test : ordinateur portable

Testeur : les membres de ce groupe

Tableau 5: Test d'enregistrement des emprunts

N°	Action	Attendu	Résultat
1	Cliquer sur l'onglet ManageBorrow	Affichage de la page de gestion des emprunts	**OK**
2	Entrer le titre du livre ou un mot-clé du titre dans le premier champ de l'entête	Affichage des informations du livre en fonction de sa disponibilité	OK
3	Entrer l'ID de l'adhérent dans le	Affichage des informations de l'adhérent et du	OK

	deuxième champ de l'entête	nombre d'emprunts qu'il a effectué	
4	Insérer l'ISBN du livre dans le champ de saisie inférieur	Affichage des informations correspondantes	OK
5	Ajouter la date d'emprunt ainsi que celle de retour	Affichage des deux dates	
6	Cliquer sur « Emprunter »	Enregistrement dans la base de données	OK

5- Consultation de l'inventaire

Prérequis : posséder un compte en tant qu'administrateur

Environnement de test : ordinateur portable

Testeur : les membres de ce groupe

Tableau 6: Test de consultation d'inventaire

N°	Action	Attendu	Résultat
1	Cliquer sur l'onglet « ManageBook »	Affichage de la liste des livres et de leur quantité respective en stock avant et après emprunt	**OK**
2	Cliquer sur « EtatEmprunt »	Affichage des informations sur les livres empruntés	OK

CONCLUSION

Ce projet de système de gestion de bibliothèque en Java a permis de développer une application complète répondant aux besoins essentiels d'une petite bibliothèque. L'application finale automatise efficacement la gestion des livres (ajout, modification, suppression, consultation) et le suivi des emprunts avec historique, le tout accessible via une interface console intuitive et une persistance des données par fichiers.

Au-delà des fonctionnalités implémentées, ce projet a constitué un excellent exercice pratique pour maîtriser les concepts de programmation orientée objet, la gestion des structures de données, la validation des entrées utilisateur et la documentation technique. La méthodologie employée, alliant analyse fonctionnelle, conception UML et implémentation, illustre parfaitement l'approche professionnelle du développement logiciel.

Les perspectives d'évolution sont nombreuses : interface graphique, base de données relationnelle, système de notifications, gestion multi-utilisateur ou module statistiques. Ce projet constitue ainsi une base solide pour comprendre les enjeux du développement logiciel et offre une expérience complète applicable à des projets de plus grande envergure.