



REPUBLIQUE DU
CAMEROUN
Paix – travail – patrie

UNIVERSITE DE
DOUALA

ECOLE NATIONALE
SUPERIEURE
POLYTECHNIQUE DE
DOUALA

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



PROJET DE PROGRAMMATION ORIENTE OBJET

THEME : Système de Gestion d'une Bibliothèque.

Filière : Génie informatique.

Option : Génie Logiciel.

MEMBRES DU GROUPE :

1. KAMGANG NGAMIGNI MARC AUREL 22G00178
2. KOUTA ERIC DONALD 24G01097
3. MINKO NYANGONO ULRICH 22G00257
4. MBOUMA ANNIE ORNELLA 22G00247
5. MBOUMELA TIFFA JAURES WILSON 22G00249
6. MBOUOMBOUO HAMED MOUDALIF 24G01112
7. MOMO FOUMTHIM GEORGE ANGELA NATACHA 22G00264
8. NGLITANG RUBEN 22G00500
9. SIGNING TSANGO MARYLINE 22G00368
10. TAMBOU NOUMSI CELIANE 22G00378

**SOUS L'ENCADREMENT DE :Dr.NOULAPEU
ANNEE SCOLAIRE :2024-2025**

SOMMAIRE

INTRODUCTION.....	7
SECTION 1 : CONTEXTE ET OBJECTIFS DE LA MISSION.....	7
SECTION 2 : L'EXECUTION DE LA MISSION.....	10
SECTION 2 : REPARTITION DES TACHES ET DIFFICULTER RENCONTEES	10
CONCLUSION.....	41
REFERENCE BIBLIOGRAPHIQUE.....	42
TABLE DES MATIERE.....	43

ABSTRACT

In a world where efficient information management is crucial, libraries must adapt to technological advancements to ensure fast and structured access to documentary resources. This project aims to design and develop a high-performance library management system, emphasizing the power and versatility of the Java programming language, renowned for its reliability and flexibility. This project presents an opportunity to combine theory and practice in computer science by implementing essential concepts such as database manipulation, graphical interface management, and system performance optimization. By integrating modern features, we aim to significantly enhance the experience of library users and administrators. In the following sections of our document, we will address two main parts: the context and objectives of our project, and its implementation.

INTRODUCTION

Dans un monde où la gestion efficace des informations est primordiale, les bibliothèques doivent s'adapter aux avancées technologiques pour assurer un accès rapide et structuré aux ressources documentaires. Ce projet vise donc à concevoir et développer un système de gestion de bibliothèque performant en mettant l'accent sur la puissance et la polyvalence du langage de programmation Java, reconnu pour sa fiabilité et sa flexibilité. Ce projet constitue une opportunité d'allier théorie et pratique en informatique, en mettant en œuvre des concepts essentiels tels que la manipulation de bases de données, la gestion des interfaces graphiques et l'optimisation des performances du système. En intégrant des fonctionnalités modernes, nous aspirons à améliorer significativement l'expérience des usagers et des gestionnaires de bibliothèques. Dans la suite de notre document nous aborderons donc deux grandes sections qui sont le contexte et les objectifs de notre projet et la mise en exécution de ce dernier.

SECTION 1 : CONTEXTE ET OBJECTIFS DE LA MISSION

1.1. CONTEXTE DE LA MISSION

Dans le cadre de l'apprentissage scolaire, il nous a été demandé de développer une application java permettant de gérer les Operations courantes d'une petite bibliothèque.

1.2. SPECIFICATIONS FONCTIONNELLES

L'application devra implémenter les fonctionnalités listées ci-dessous :

i. Gestion des Livres

- o Ajout d'un livre : Saisir les informations de base (titre, auteur, ISBN, année de Publication).
- o Suppression et modification : Permettre la mise à jour ou la suppression d'un Livre existant.
- o Consultation de l'inventaire : Afficher la liste complète des livres disponibles.

ii. Gestion des Emprunts

- o Emprunt d'un livre : Enregistrer l'emprunt par un utilisateur (nom, date D'emprunt).
 - o Retour de livre : Enregistrer le retour et mettre à jour le statut du livre.
 - o Historique : Consulter l'historique des emprunts pour chaque livre ou
-

utilisateur.

iii. Interface Utilisateur Console

o Un menu principal permettant de naviguer entre les fonctionnalités (gestion des livres, gestion des emprunts, consultation).

o Gestion des entrées utilisateur avec des validations de saisie.

iv. Persistance des Données (Optionnel pour les débutants).

o Utiliser des fichiers texte ou un format simple (par exemple, CSV) pour sauvegarder et restaurer les données entre plusieurs exécutions de l'application.

1.3. SPECIFICATIONS NON-FONCTIONNELLES

Les spécifications non-fonctionnelles décrivent toutes les contraintes techniques, ergonomiques et esthétiques auxquelles est soumis le système pour sa réalisation et pour son bon fonctionnement. En ce qui concerne notre application, nous avons dégagé les besoins suivants :

- La fiabilité : les données fournies par l'application doivent être fiable.
- La convivialité de l'interface graphique : l'application doit fournir une interface conviviale et simple pour tout type d'utilisateur car elle présente le premier contact entre l'utilisateur et l'application et par le biais de celle-ci on découvrira ses fonctionnalités.
- L'efficacité : l'application doit permettre l'accomplissement des tâches avec le minimum de manipulations.
- La sécurité ; l'application doit être sécurisé au niveau des données : authentification et contrôle.
- La portabilité : l'application doit être portable c'est-à-dire fonctionnelle sur n'importe quelle plateforme.

- La rapidité : le délai de réponse doit être relativement court.

1.4. EVALUATION DES MESURES DE SUCCES

Après la réalisation de l'application nous devons effectuer les tests de vérification et de validation afin de confirmer ou Infirmer la réussite de l'application.

SECTION 2 : L'EXECUTION DE LA MISSION

2.1. LOGICIELS ET LANGAGES UTILISES.

a. Eclipse

eclipse



Dans le monde informatique, Eclipse est un programme de développement intégré de diverses compétences développant applications informatiques utilisant notamment le langage Java, C/C, Python, PERL, Ruby et bien d'autres. Gratuit et open source, l'IDE Eclipse est l'un des IDE Java les plus populaires du marché informatique. Son vaste écosystème de plugins le rend particulièrement des développeurs, car il prend en charge des plugins et des fonctions personnalisables pour le développement de n'importe l'application.

b. Tomcat version 10



Tomcat Logiciel de serveur d'applications web open source conçu pour la programmation en Java et développé et maintenu par Jakarta, le groupe de projets open source Java de la fondation Apache.

c. SQLitestudio



SQLiteStudio est un éditeur de code SQL avancé multiplateforme gratuit et ouverte. Il est publié sous licence GPL et peut être utilisé librement dans n'importe quel but.

d. StarUML



StarUML est un modélisateur de logiciel sophistiqué destiné à prendre en charge la modélisation *souple* et *concise*. Il nous permettra de réaliser nos différents diagrammes.

e. Visual Studio Code



C'est un éditeur de code source et un environnement de développement intégré (IDE) de Microsoft.

f. CSS



Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium. Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

g. HTML

HTML (pour HyperText Markup Language, qu'on peut traduire en « langage de balisage hypertexte ») est **le langage utilisé pour structurer une page web et son contenu**

h. Java JDK

- Java est un langage de programmation de haut niveau orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien Java de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai

1995 au SunWorld.

Le JDK (Java™ Development Kit) est un logiciel destiné aux développeurs Java. Il inclut l'interpréteur Java, les classes Java et les outils de développement Java : compilateur, débogueur, désassembleur, appletviewer, générateur de fichiers de raccord et générateur de documentation.

Le JDK vous permet d'écrire des applications qui sont développées une seule fois et s'exécutent n'importe où sur n'importe quelle machine virtuelle Java. Les applications Java développées avec le JDK sur un système peuvent être utilisées sur un autre système sans modifier ou recompiler le code. Les fichiers de classe Java sont portables sur n'importe quelle machine virtuelle Java standard.

2.2. ETUDE DE L'EXISTANT

L'étude de l'existant est une phase importante pour bien comprendre le système actuel et définir des objectifs qui apporteront des solutions pour un système futur. Nous présenterons l'étude technique et la mise en œuvre de la solution. En commençant par décrire le système existant, Les règles de gestion du système futur et par la suite la méthodologie d'analyse et les outils utilisés pour le développement, enfin illustrer certaines fonctionnalités de l'application à travers quelques interfaces.

2.2.1. Présentation du système actuel

a. Description de l'existant

Le fonctionnement actuel de la bibliothèque de l'ENSPD repose principalement sur un système manuel de gestion des ressources :

- *Accueil des utilisateurs* : Les étudiants, enseignants et chercheurs sont libres de circuler dans la bibliothèque et de lire des documents sans enregistrements préalables.
 - *Organisation des ouvrages* : Les documents sont classés par catégories par exemple mathématique, physique, français etc.
-

- *Prêts et retours* : Le système de prêt est manuel. Lorsqu'un étudiant emprunte un document, le bibliothécaire enregistre la transaction dans un registre. Les retours sont également enregistrés manuellement.
- *Espaces de consultation* : La bibliothèque met à disposition des espaces de lecture et de travail, adaptés aux besoins des étudiants pour leurs recherches et révisions.

Cette gestion traditionnelle, présente des limites évidentes en termes de réactivité et de suivi des documents.

b. Critique de l'existant

Elle permet que ce soit sur le plan fonctionnel ou sur le plan organisationnel d'identifier les lacunes et les problèmes rencontrés avec le système actuellement utilisé. Cela permet également de déterminer les fonctionnalités et les améliorations nécessaires pour développer un système plus efficace. Nous retrouvons donc :

- L'incapacité à gérer facilement les documents ;
- L'inaptitude à gérer le suivi des utilisateurs ;
- La perte des données pouvant induire aux problèmes insécurités des données.
- La communication entre les différents acteurs impliqués dans la gestion de la bibliothèque peut être difficile et peu efficace.

2.3. PRESENTATION DE L'OUTIL DE MODELISATION UTILISE

2.3.1. Choix de la méthode de modélisation

Si l'on parle de choix de méthode d'analyse, cela souligne l'existence de différences entre celles-ci. Il convient donc de procéder par comparaison afin d'en choisir une. Il en existe une multitude de méthode d'analyse : BOOCH, MERISE, OOSE, SADT, SART, MACAO, APTE et le langage UML. Pour notre cas, la comparaison se fera entre les deux méthodes les plus en vue :

MERISE et UML.

Méthodes	Avantages	Inconvénients
MERISE	<ul style="list-style-type: none"> -La facilite de compréhension et de la représentation des données d'un système ; -MERISE permet de modéliser correctement une application des données et des traitements et permet aussi de passer du MCD au MLD puis MPD. 	<ul style="list-style-type: none"> -MERISE n'est pas en mesure de modéliser les données à Caractères sémantiques ; - La méthode MERISE est bien adaptée à l'automatisation de tâches séquentielles de gestion pure. En revanche, elle est mal adaptée aux environnements distribués où de multiples applications externes à un domaine viennent interagir avec l'application à modéliser.
U.M.L	<ul style="list-style-type: none"> -UML est un langage formel et normalisé : il permet un gain de précision et de Stabilité ; -UML de par son origine (la programmation orientée objet) s'affirme comme un ensemble de formalismes pour la conception de logiciel de base. 	<ul style="list-style-type: none"> Temps : inconvénients que certains développeurs pourraient trouver lorsqu'ils utilisent UML est le temps qu'il faut gérer et entretenir les diagrammes ; -Les diagrammes peuvent devenir accablants.

A l'issue de la comparaison entre ces deux méthodes à travers ce tableau, notre choix s'est porté sur la méthode UML pour la modélisation de notre projet.

2.3.1.1. Cycle de vie

Le cycle de vie d'une application représente l'ensemble des étapes du développement du logiciel, de sa conception à sa disparition. Elle permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation. Le choix du modèle de cycle de vie est important dans la construction d'un logiciel.

i.Les éléments du cycle de vie

Le cycle de vie comprend les éléments indispensables suivants :

- ❖ Spécification des besoins : elle consiste à définir la finalité du projet et son intégration dans une stratégie globale.
- ❖ Conception générale : dans cette activité, il s'agit de la préparation de l'architecture générale du logiciel.
- ❖ Conception détaillée : elle consiste à définir précisément chaque sous-ensemble du logiciel.
- ❖ Codage : il s'agit d'une traduction des fonctionnalités définies dans la phase de conception en langage de programmation.
- ❖ Tests unitaires : ils permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux normes définies dans la conception.
- ❖ Intégration : dite aussi tests systèmes, elle consiste à vérifier que le logiciel correspond exactement au cahier des charges du projet en obtenant enfin un manuel d'utilisation bien détaillé aux utilisateurs.

ü. Choix du modèle : MODELE DE CYCLE DE VIE EN V
MÉTHODE DU CYCLE EN V

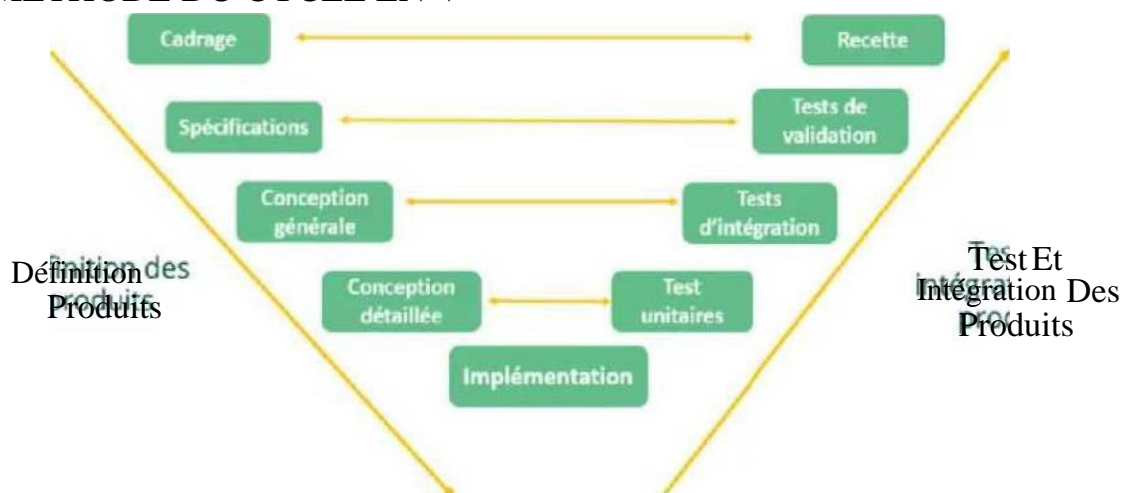


Figure 3 : modèle de cycle de vie en v

Le modèle du cycle de vie en V est un modèle conceptuel de gestion de projet, imaginé suite au problème de réactivité du modèle en cascade. Il permet,

en cas d'anomalie, d'éliminer le retour aux étapes précédentes tardivement. Les avantages du modèle du cycle de vie en V sont les suivants :

- La qualité de la mise en œuvre des tests.
- Modèle éprouvé dans l'industrie.
- Normalisé (ISO-12207, MILSTD-498).
- Deux types de tâches sont réalisées en parallèle : Verticalement on prépare l'étape suivante et Horizontalement : on prépare la vérification de la tâche en cours.

Il présente aussi des Inconvénients à savoir:

- La validation finale par le client très tardive augmente les risques de dépassement de délai et donc l'augmentation du coût.
- Phases séquentielles.
- Rigidité face à une évolution du besoin.

Nous avons opté pour le cycle de vie en V pour la conception et le développement de cette application. Ceci parce que ce cycle ouvre plus de porte à la vérification après la fin d'un module ce qui permet d'éviter les erreurs lors de la progression.

2.3.1.2. Présentation du langage de modélisation UML

UML (Unified Modeling Language, que l'on peut traduire par "langage de modélisation unifié ") est une notation permettant de modéliser un problème de façon standard. Ce langage est né en 1997 de la fusion de plusieurs méthodes existantes auparavant, et est devenu désormais la référence en termes de modélisation objet, à un tel point que sa connaissance est souvent nécessaire pour obtenir un poste de développeur objet.

Le langage UML fournit une panoplie d'outils permettant de représenter l'ensemble des éléments du monde objet (classes, objets) ainsi que les liens qui les relie. Toutefois, étant donné qu'une seule représentation est trop subjective,

UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues. Une vue est constituée d'un ou plusieurs diagrammes. On distingue deux types de vues :

- Les vues statiques, c'est-à-dire représentant le système physiquement (diagramme d'objets, diagramme de Classes, diagramme de composants, diagramme de déploiement).
- Les vues dynamiques, montrant le fonctionnement du système (diagramme de séquence, diagramme de communication, diagramme d'états-transitions, diagramme d'activités, diagramme de cas d'utilisation).

2.3.1.3. Les principaux diagrammes d'UML

- ❖ Diagramme de séquences : Il représente séquentiellement le déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs. Il peut servir à illustrer un cas d'utilisation.
 - ❖ Diagramme de classes : Le but d'un diagramme de classes est d'exprimer de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes. Une classe a des attributs, des opérations et des relations avec d'autres classes.
 - ❖ Diagramme d'objets : Ce diagramme permet la représentation d'instances des classes dans un état particulier et des liens entre instances.
 - ❖ Diagramme de composant : Ce diagramme permet de représenter les composants du système d'un point de vue physique d'une application en termes de modules : fichiers sources, bibliothèques, exécutables, etc.
 - ❖ Diagramme de déploiement : Ce type de diagramme UML montre la disposition physique des matériels qui composent le système (ordinateurs, périphériques, réseaux...) et la répartition des composants sur ces matériels. Les ressources matérielles sont représentées sous forme de nœuds, connectés par un support de communication.
 - ❖ Diagramme de collaboration : C'est une représentation spatiale des objets, et
-

Démonstration des interactions entre objets (instances de classes et acteurs).

- ❖ Diagramme d'états transitions : Ce diagramme Permet de décrire le comportement dynamique d'une entité (logiciel, composant, objet...), décrit par des états qui relient par des transitions.
- ❖ Diagramme d'activité : Ce diagramme est une variante des diagrammes d'états transitions. Il permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.
- ❖ Diagramme de navigation : Il représente de manière formelle l'ensemble des chemins possibles entre les principaux écrans proposés à l'utilisateur.
- ❖ Diagramme de cas d'utilisation : Ce diagramme permet de structurer les besoins des utilisateurs et d'identifier les possibilités d'interaction entre le système et les acteurs.

2.3.2. Modélisation du système

● DIAGRAMME DE CAS D'UTILISATION

a. Concepts et objectif du diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est un outil de modélisation qui permet de représenter les interactions entre les acteurs (utilisateurs) et un système. Son objectif principal est de décrire les différentes façons dont les utilisateurs interagissent avec le système pour accomplir des tâches spécifiques. Il aide à visualiser les fonctionnalités offertes par le système du point de vue des utilisateurs, en mettant l'accent sur ce que le système fait plutôt que sur comment il le fait.

Les concepts clés associés à un diagramme de cas d'utilisation comprennent :

- ❖ Acteurs : Avant de rechercher les besoins, la première tâche consiste à définir les limites du système (c'est à dire ce qui est inclus ou pas dans le système), puis à identifier les différentes entités intervenantes sur le système. Ces entités sont appelées acteurs. Les acteurs se représentent sous
-

la forme d'un petit personnage (stick man) ou sous la forme d'une case rectangulaire (appelé classeur) avec le mot clé « actor ».

Chaque acteur porte un nom.

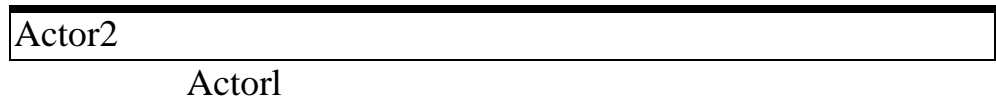


Figure 4 : représentation d'acteurs du diagramme de cas d'utilisation

- ❖ CAS D'UTILISATION : Le cas d'utilisation représente une fonctionnalité du système (visible de l'extérieur du système). Un cas d'utilisation se représente par une ellipse contenant le nom du cas d'utilisation (phrase commençant par un verbe à l'infinitif) et optionnellement un stéréotype au-dessus du nom. Les différents cas d'utilisation peuvent être représentés à l'intérieur d'un même rectangle représentant les limites du système.



Figure 5 : représentation d'un cas d'utilisation

❖ Relation

- Relation entre acteurs et cas d'utilisation : A chaque acteur est associé un ou plusieurs cas d'utilisations, la relation d'association peut aussi être appelée relation de communication. Elle est représentée par un trait continu reliant l'acteur et le cas d'utilisation.
- Relation entre acteurs : La seule relation possible entre 2 acteurs est la généralisation. Un acteur A est une généralisation d'un acteur B si l'acteur A peut-être substituée à l'acteur B. Le symbole utilisé pour la généralisation est une flèche avec un trait plein dont la pointe est un triangle ferme désignant le cas le plus général.



Figure 6 : représentation de la relation entre acteurs

- Relation entre cas d'utilisation o L'inclusion

Un cas d'utilisation A inclut un cas d'utilisation B si le comportement décrit par le cas d'utilisation A inclut le comportement du cas d'utilisation B : le cas d'utilisation A dépend du cas d'utilisation B. Lorsque A est sollicité, B l'est obligatoirement comme une partie de A.

Cette relation est représentée par le stéréotype « include ».



Figure 7 : représentation de la relation d'inclusion

- o L'extension

On dit qu'un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation

A peut être appelé au cours de l'exécution du cas d'utilisation B. Exécuter B peut éventuellement entraîner l'exécution de A. Cette relation est représentée par le stéréotype

« extend ».

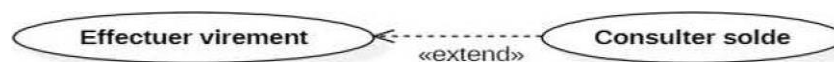


Figure 8 : représentation de la relation d'extension

o La généralisation

Un cas d'utilisation A est une généralisation d'un cas d'utilisation B si B est un cas particulier de A.

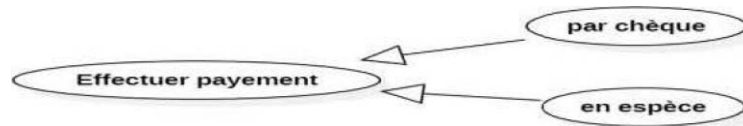


Figure 9 : représentation de la relation de généralisation

b. Modélisation du diagramme des cas d'utilisation

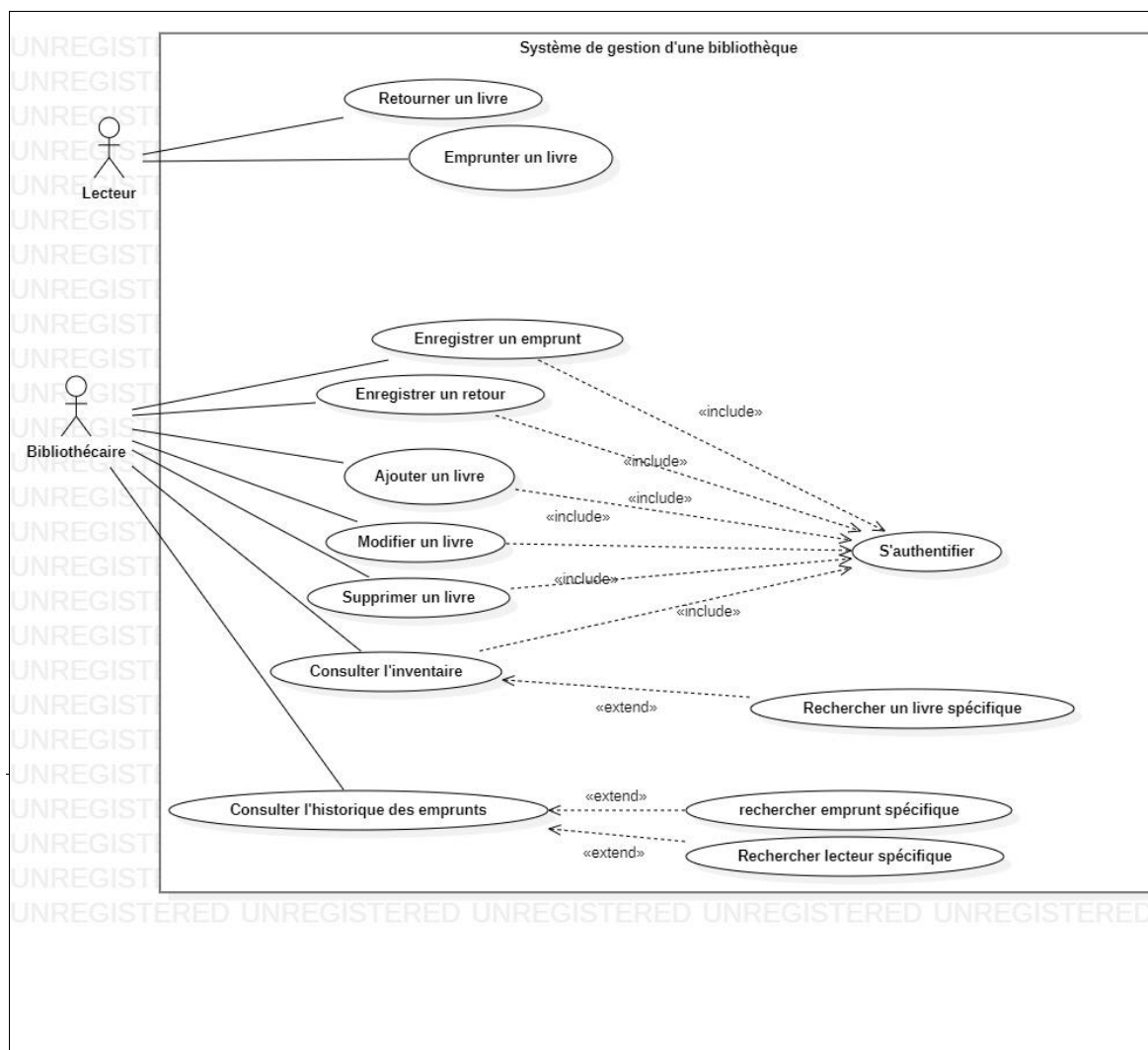


Figure 10 : diagramme de cas d'utilisation du projet

● . DIAGRAMME DE CLASSES

a. Concepts et objectifs du diagramme de classes

Le diagramme de classe est un outil de modélisation qui permet de représenter les classes et les relations entre ces classes dans un système informatique. Son objectif principal est de visualiser la structure statique d'un système, en mettant en évidence les classes, les attributs, les méthodes et les relations entre les différentes entités.

Les concepts clés associés à un diagramme de classes sont :

- ❖ **Classe** : Une classe est une représentation abstraite d'un d'ensemble d'objets, elle contient les informations nécessaires à la construction de l'objet (c'est-à-dire la définition des attributs et des méthodes). La classe peut donc être considérée comme le modèle, le moule ou la notice qui va permette la construction d'un objet.

Nous pouvons encore parler de type (comme pour une donnée). On dit également qu'un objet est l'instance d'une classe (la concrétisation d'une classe).

- Une classe est représentée par un rectangle (appelé aussi classeur) divisé en 3 parties :
- La première partie contient le nom de la classe
- La seconde contient les attributs de la classe. La syntaxe d'un attribut est la suivante.

Visibilité nomAttribut [multiplicité] : typeAttribut = Initialisation,

- La dernière contient les méthodes de la classe. La syntaxe d'une méthode est la suivante :

NomClasse
+NomAttribute1: type +NomAttribute2 : type *NomAttribute3 : type
- NomMethode1() : voici

```
-
NomMethode2()
: voici
```

Figure 11 : représentation d'une classe

- Relation : elle indique comment les classes sont associées les unes aux autres, telles que l'héritage, l'agrégation, la composition et l'association.
- Héritage : C'est un principe de division par généralisation et spécialisation, représenté par un trait reliant les deux classes et dont l'extrémité du côté de la classe mère comporte un triangle.

La classe fille hérite de tous les attributs et méthodes, qu'ils soient publics, protégés ou privés. Cependant, elle ne peut pas utiliser directement les attributs et méthodes privés (que ce soit en lecture ou en écriture), sauf par l'intermédiaire d'une méthode héritée (publique ou protégée).

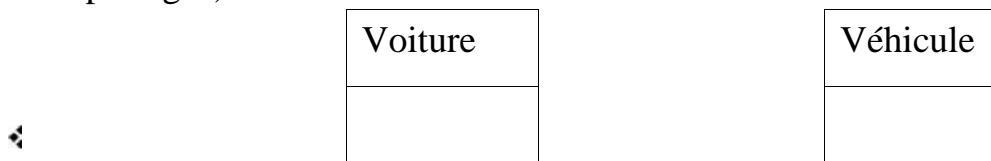


Figure 12 : représentation de la relation d'héritage entre classe

- Association : Une association est une relation entre deux classes ou plus, qui décrit les connexions structurelles entre leur instance. On distingue : l'association binaire, l'association n-aire, l'association réflexive, la classe association. Une association se caractérise par:
- Le nom de l'association : L'association peut être ornée d'un texte, avec un éventuel sens de lecture, qui permet de nous informer de l'intérêt de cette relation.
- Le rôle : Chaque extrémité d'une association peut être nommée. Ce nom est appelé rôle et indique la manière dont l'objet est vu de l'autre côté de l'association.



- Émetteur • récepteur

Figure 13 : représentation de la relation d'association entre

- La cardinalité ou multiplicité : comparable aux cardinalités du système Merise, elle sert à compter le nombre minimum et maximum d'instances de chaque classe dans la relation liant 2 ou plusieurs classes.

Tableau 8 : cardinalité dans un diagramme de classe

Cardinalité	Signification
0..1	Zéro ou une fois
1..1 (ou 1)	Une et une seule fois
0.. * (ou *)	De zéro à plusieurs fois
	D'une à plusieurs fois
m... n	Entre m et n fois
n. n (ou n)	n fois

- La navigabilité : les associations sont bidirectionnelles et peuvent être parcourues dans les 2 sens. Lorsque l'association est contrainte pour devenir unidirectionnelle, le sens de navigation qui reste possible est spécifié par une flèche. Il est préférable de laisser les associations bidirectionnelles. UML autorise d'alerter sur le sens de la navigation interdit avec une croix en plus de la flèche.

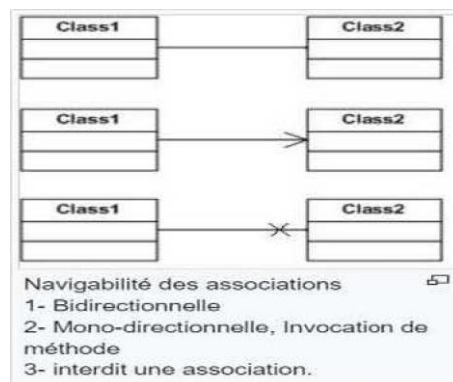


Figure 14 : navigabilité dans un diagramme de classe

On distingue :

- Association binaire : Une association binaire est représentée par un simple trait continu, reliant les deux classes.

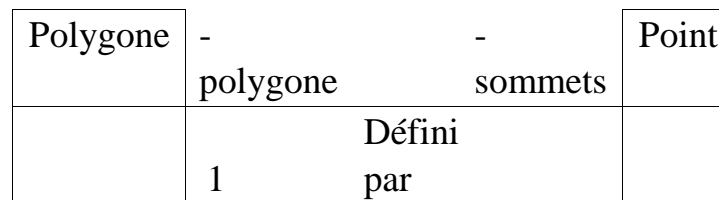


Figure 15 : association binaire

- Association n-aire : Une association qui lie plus de 2 classes entre elles, est une association n-aire. L'association n-aire se représente par un losange d'où part un trait allant à chaque classe. L'association n-aire est difficile à interpréter et souvent source d'erreur, elle est donc le plus souvent remplacée par un ensemble d'associations binaires afin de lever toute ambiguïté.



Figure 16 : association n-aire

- Association réflexive : Une association qui lie une classe avec elle-même est une association réflexive.

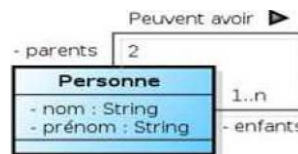
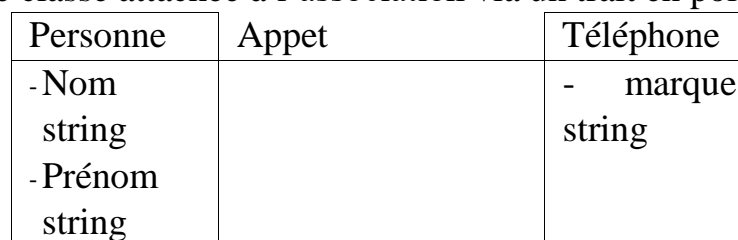


Figure 17 : association réflexive

- Classe association : Une association peut apporter de nouvelles informations (attributs et méthodes) qui n'appartiennent à aucune des deux classes qu'elle relie et qui sont spécifiques à l'association. Ces nouvelles informations peuvent être représentées par une nouvelle classe attachée à l'association via un trait en pointillés.



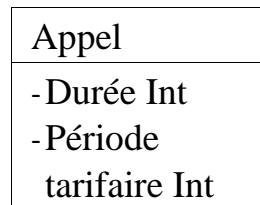


Figure 18 : classe association

- ❖ Visibilité : elle spécifie la portée des attributs et des méthodes par rapport à d'autres classes. Elle ne peut prendre que 4 valeurs:

Tableau 9 : niveau de visibilité dans un diagramme de classe

Caractère			Description
	Accès public	Public	Toutes les autres classes ont accès à cet attribut
	Accès protégé	Protected	Seules la classe elle-même et les classes filles (héritage) ont accès à cet attribut.
	Accès package	Package	Classe visible uniquement dans le package.
		Private	Seule la classe elle-même a accès à cet attribut

- ❖ L'agrégation : c'est une association qui représente une relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble. Elle est représentée par un trait reliant les deux classes et dont l'origine se distingue d'extrémité (la classe subordonnée) par un losange vide.
- ❖ La composition : dite également agrégation forte, décrit une contenance structurelle entre instances. Ainsi la destruction de l'objet composite implique la destruction de ses composants. L'origine de cette association est représentée par un losange plein.

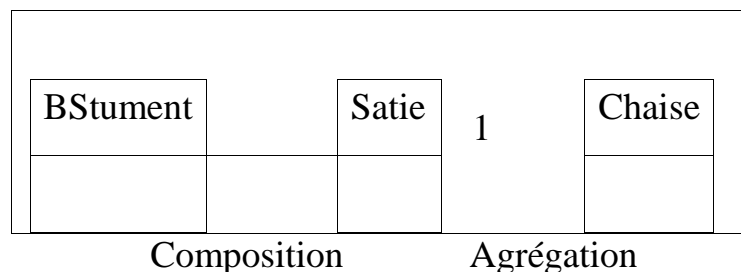


Figure 19 : agrégation et composition

b. Modélisation du diagramme de classes.

CONCEPTION ET REALISATION D'UN SYSTÈME DE GESTION DE BIBLIOTHEQUE

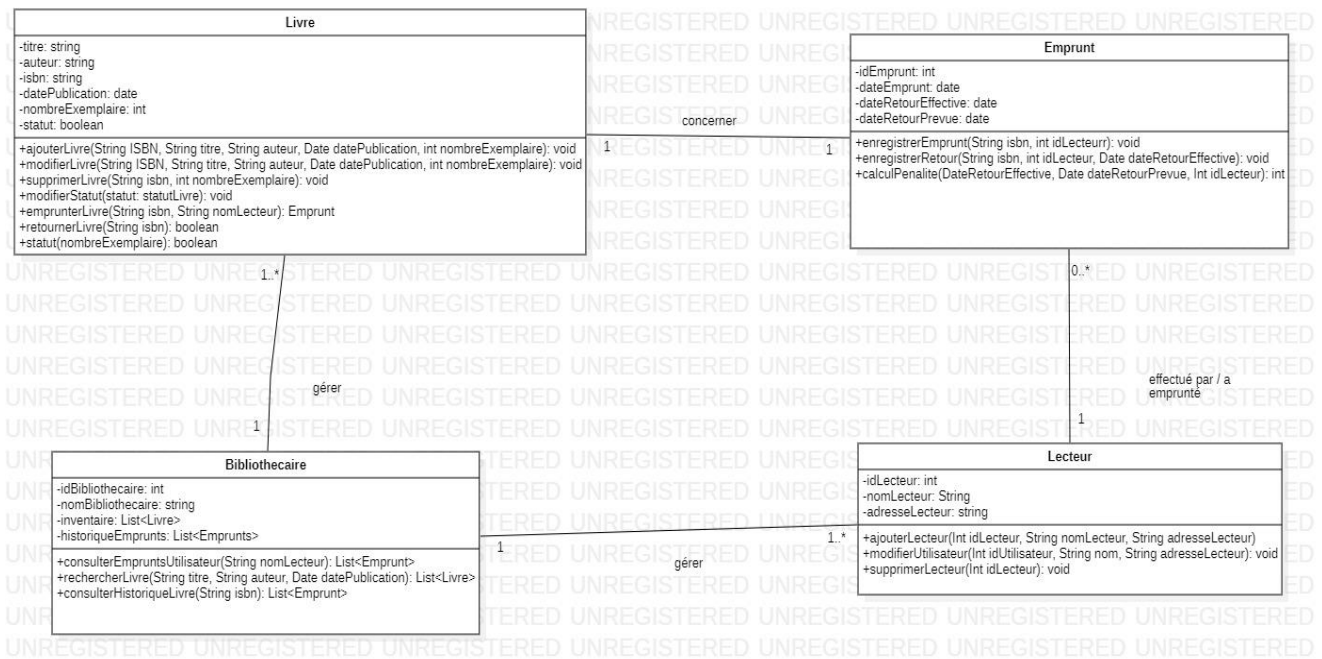


Figure 20 : diagramme de classes du projet

● DIAGRAMME DE SEQUENCE

a. Concepts et objectifs du diagramme de séquence

Le diagramme de séquence est un outil de modélisation utilisé en génie logiciel pour représenter la manière dont les objets interagissent dans une séquence temporelle pour accomplir un comportement spécifique. Son objectif principal est de visualiser et de décrire la séquence d'interactions entre les différents objets d'un système dans le cadre d'un scénario particulier.

Les concepts clés associés à un diagramme de séquence sont :

- ❖ **Objet** : Dans un diagramme de séquence, l'objet à la même représentation que dans le diagramme des objets. C'est-à-dire un rectangle dans lequel figure le nom de l'objet. Le nom de l'objet est généralement souligné
- ❖ **Ligne de vie** : Comme il représente la dynamique du système, le diagramme de séquence fait entrer en action les instances de classes intervenant dans la réalisation d'un cas d'utilisation particulier. A chaque objet est associé une ligne de vie (en trait pointillés à la verticale de l'objet) qui peut être considéré comme un axe temporel (le temps s'écoule du haut vers le bas). Dans ce genre de diagramme, la quantification du temps n'a pas d'importance. La ligne de vie indique les périodes d'activité de l'objet

(généralement, les moments où l'objet exécute une de ces méthodes). Lorsque l'objet est détruit, la ligne de vie s'achève par une croix.

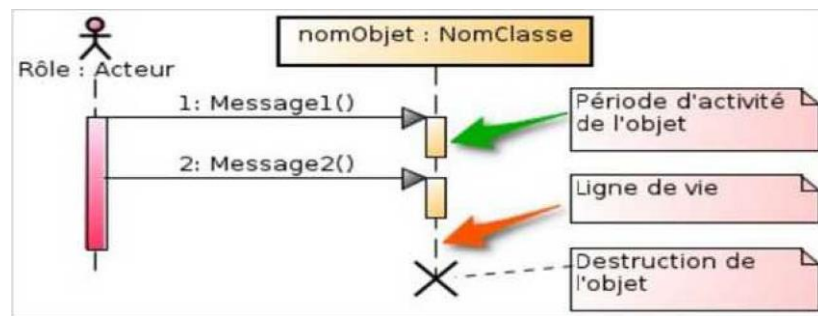


Figure 22 : représentation de la ligne de vie

❖ Message : Il définit une communication particulière entre des lignes de vie. Ainsi, un message est une communication d'un objet vers un autre objet. La réception d'un message est considérée par l'objet récepteur comme un événement qu'il faut traiter (ou pas). Plusieurs types de messages existent, les plus communs sont :

- L'invocation d'une opération ; message synchrone (appel d'une méthode de l'objet cible). La réception d'un message synchrone doit provoquer chez le destinataire le lancement d'une de ses méthodes (qui souvent porte le même nom que le message). L'expéditeur du message reste bloqué pendant toute l'exécution de la méthode et attend donc la fin de celle-ci avant de pouvoir lancer un nouveau message. C'est le message le plus fréquemment utilisé. Il est représenté flèche avec un triangle plein à son extrémité.
- L'envoi d'un signal : message asynchrone (typiquement utilisé pour la gestion événementielle). Dans le cas d'un message asynchrone, l'expéditeur n'attend pas la fin de l'activation de la méthode invoquée chez le destinataire. Il est représenté flèche simple.

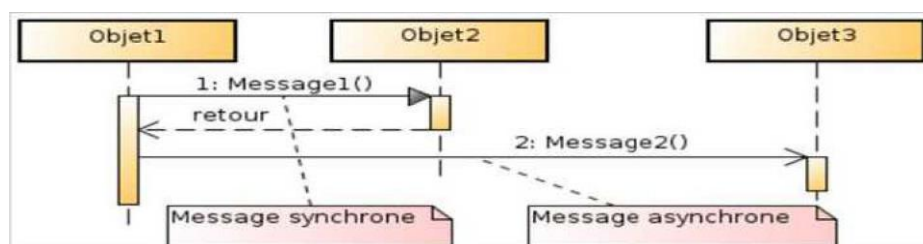


Figure 23 : représentation des messages synchrones et asynchrones

- La création ou la destruction d'une instance de classe au cours du cycle principal. La création d'un objet est matérialisée par un message spécifique, appel d'un constructeur, généralement accompagné du stéréotype « create » qui pointe sur le début (le sommet) de la ligne de vie de l'objet créé (Le rectangle de l'instance de la classe est alors surbaissé). La destruction d'un objet est représentée par une croix à la fin de sa ligne de vie. Souvent l'objet est détruit suite à la réception d'un message mais ce n'est pas obligatoire. Dans ce cas-là, il porte le stéréotype « destroy ».

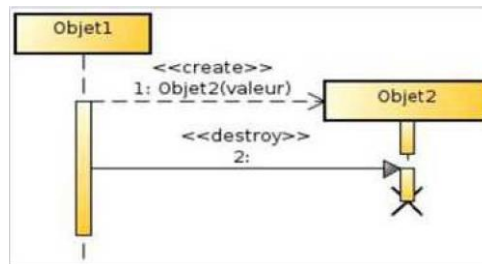
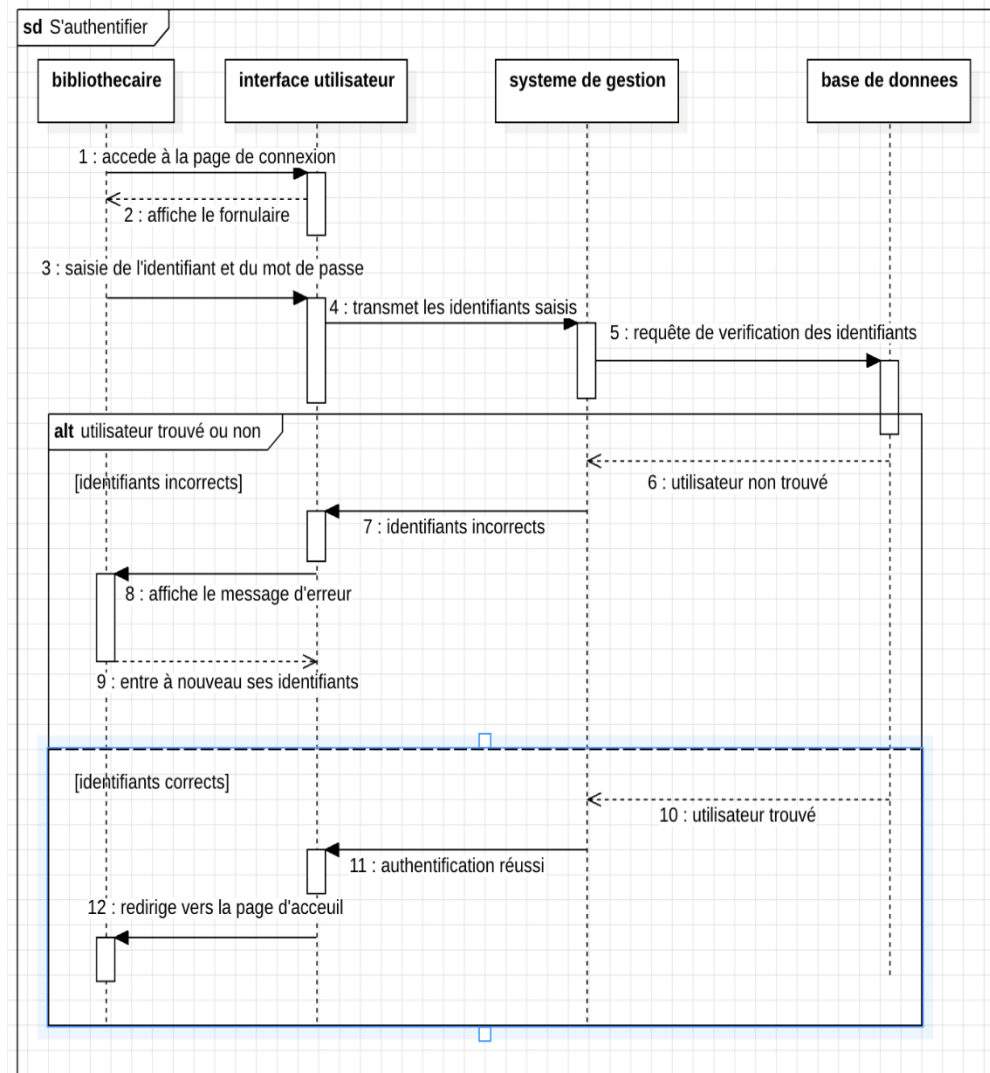


Figure 24 : création et destruction d'un objet

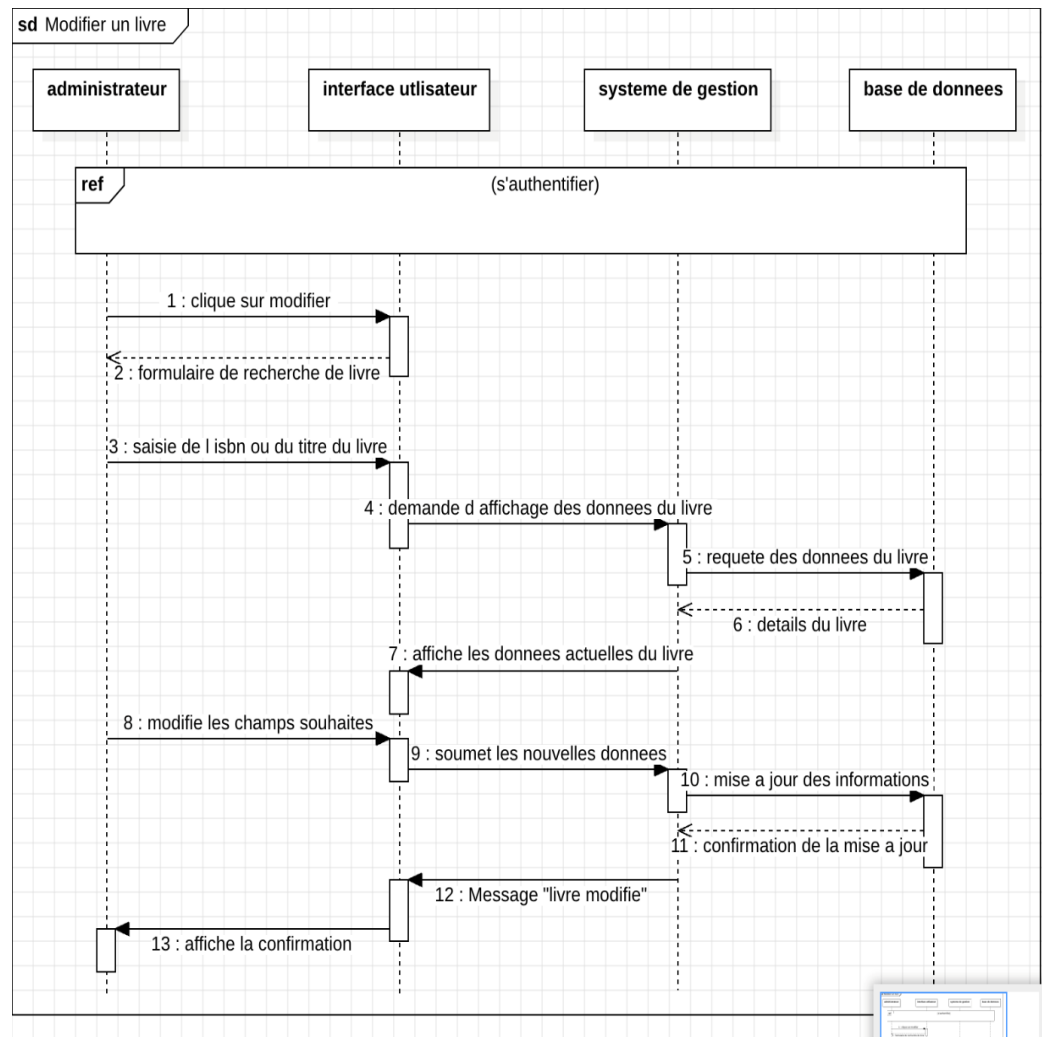
b. Modélisation du diagramme de séquence

Cas 1 : S'authentifier

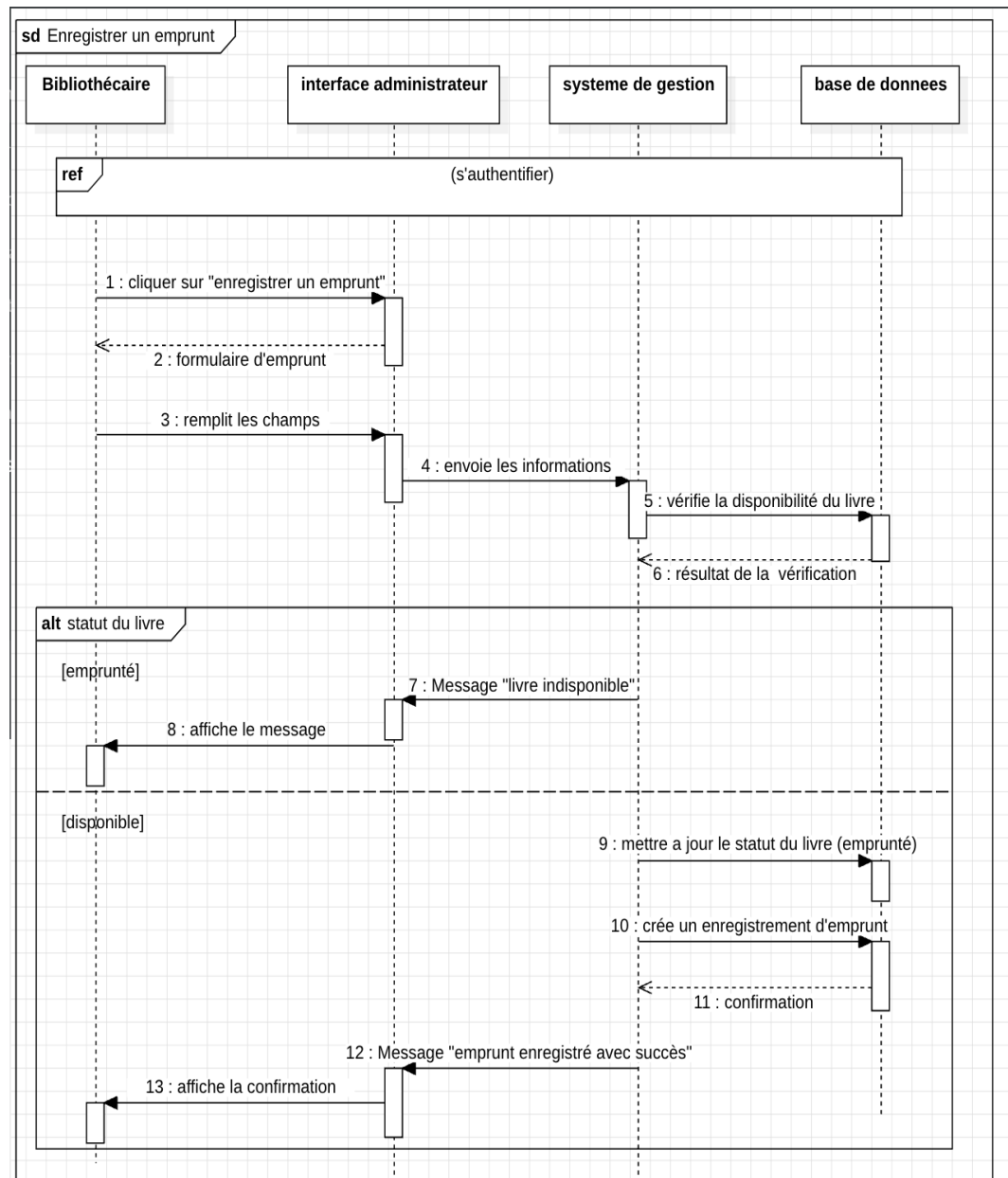
CONCEPTION ET REALISATION D'UN SYSTÈME DE GESTION DE BIBLIOTHEQUE



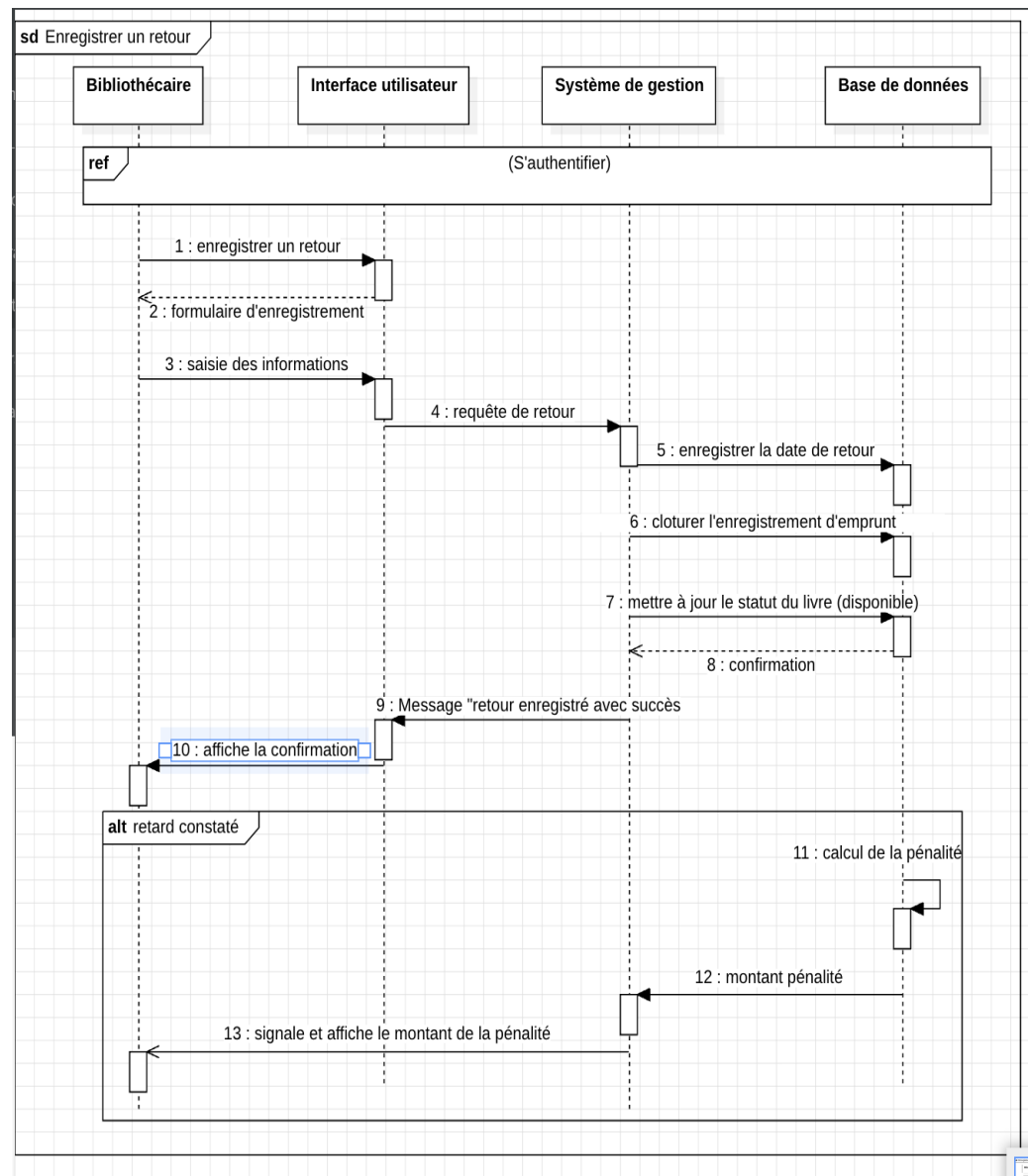
Cas 2 : Modifier un livre



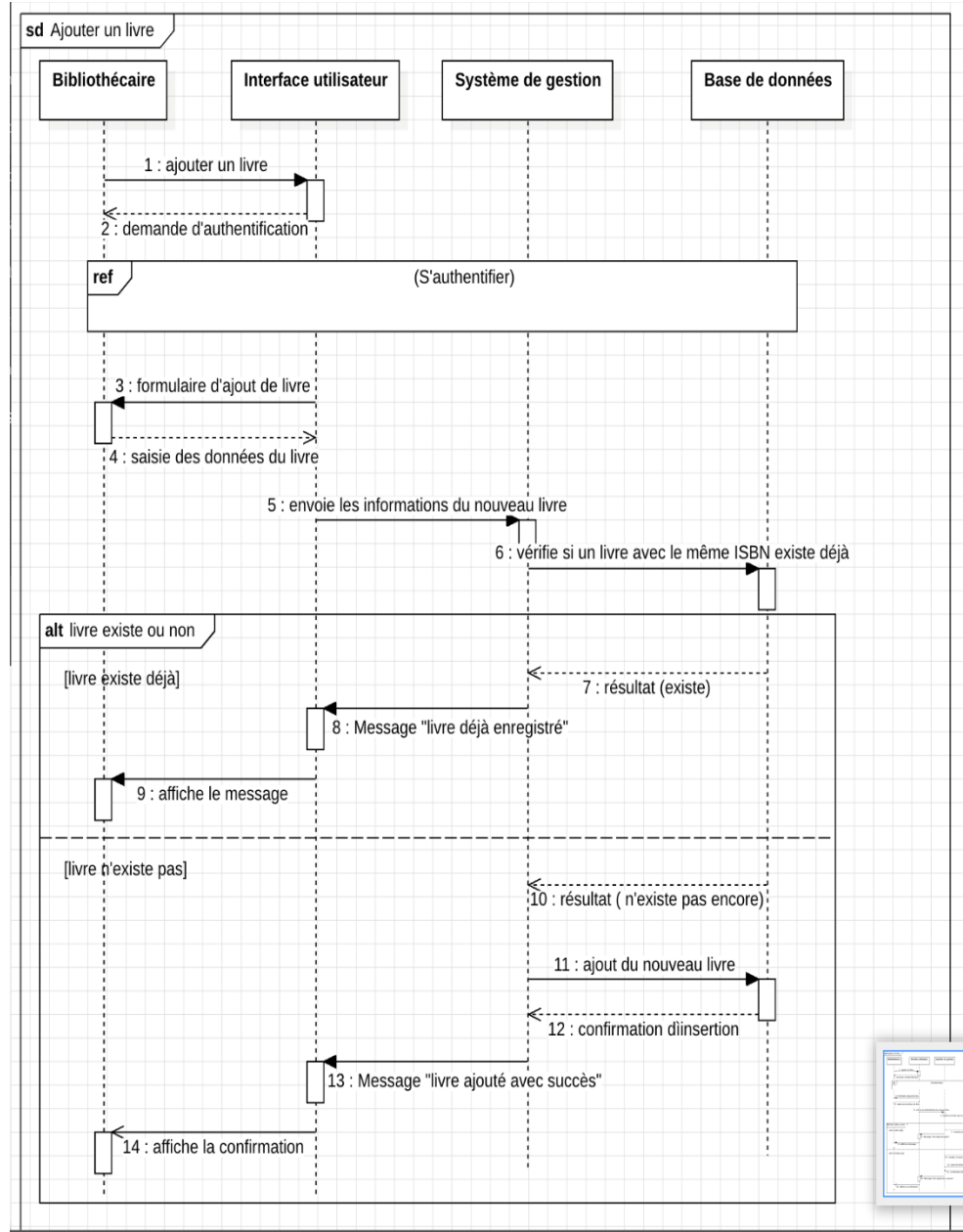
Cas 3 : Enregistrer un emprunt



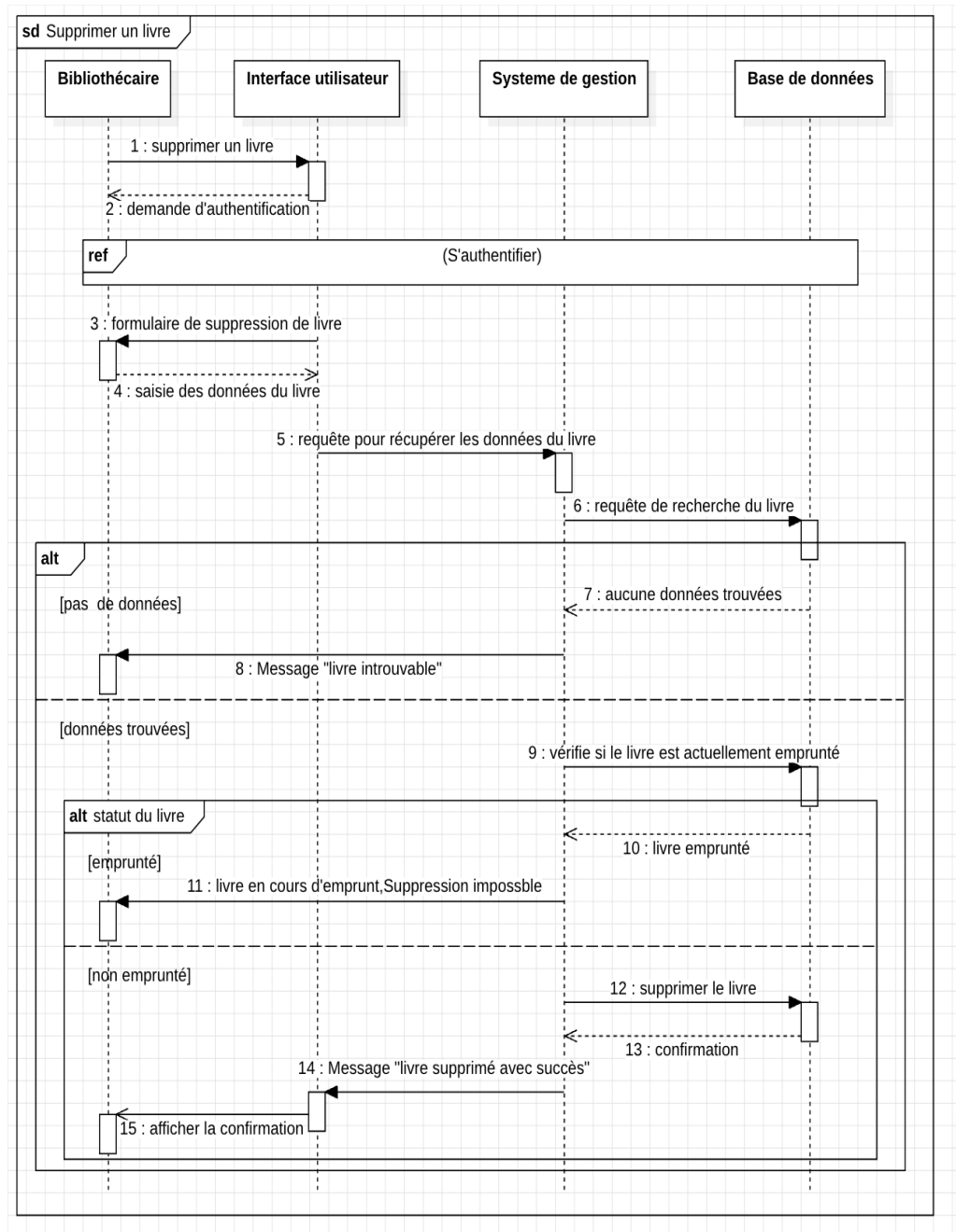
Cas 4 : Enregistrer un retour



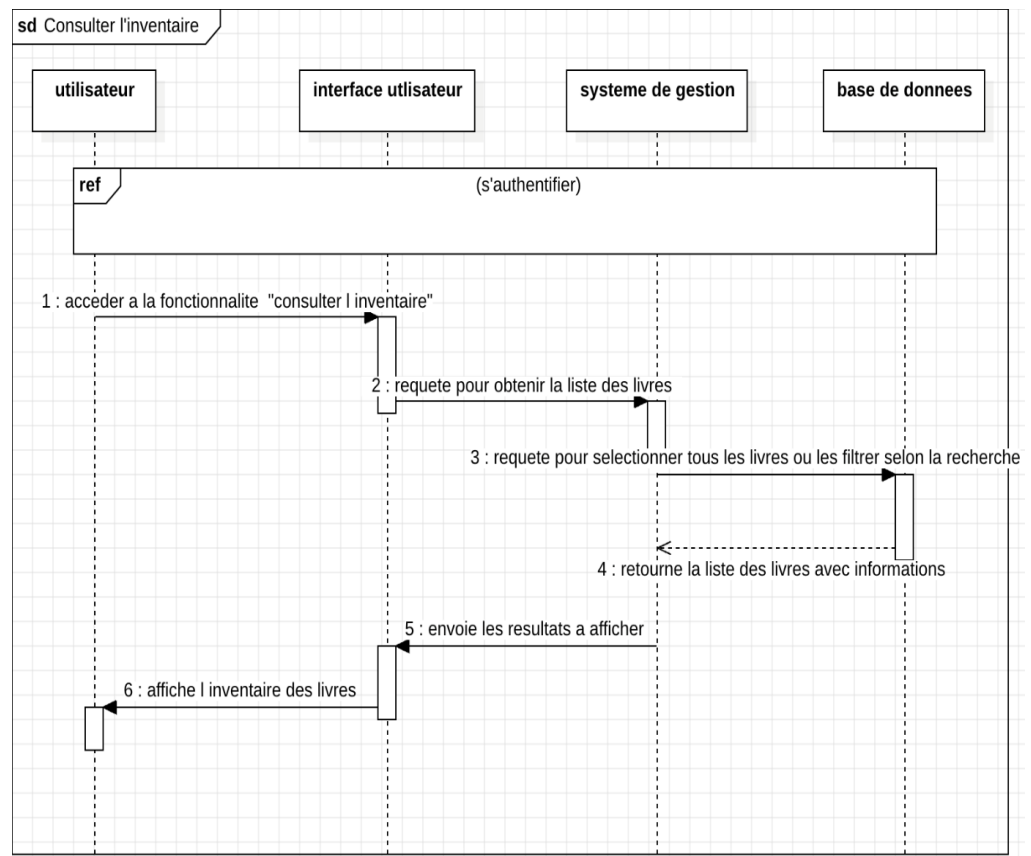
Cas 5 : Ajouter un livre



Cas 6 : Supprimer un livre



Cas 7 : Consulter l'inventaire



2.4. PRESENTATION DE L'APPLICATION

Quelques fonctionnalités de l'application

a. Authentification

La figure suivante présente l'interface d'authentification où doit passer tous les administrateurs afin de se connecter à l'application:

The screenshot shows a web application interface for the ENSPD library. At the top, there is a dark blue header with a logo on the left and a small circular icon on the right. The main content area is light gray and features a white login box in the center. The box is titled "BIENVENUE A LA BIBLIOTHEQUE DE l'ENSPD !!". Below the title, there is a section labeled "Connexion". It contains two input fields: "Nom de Bibliothécaire :" with a dropdown menu showing "MBOUMELA JAURÈS", and "Mot de passe :" with a masked password field. A blue "Se connecter" button is positioned below the password field. At the bottom of the page, there is a dark blue footer with three columns of text. The first column is titled "Bibliothèque ENSPD" and describes the center as a resource and documentation center for students and teachers. The second column is titled "Liens Rapides" and lists links for Accueil, Catalogue des livres, Horaires d'ouverture, and Règlement intérieur. The third column is titled "Contact" and provides the university name, address (BP 2701, Douala, Cameroun), phone number (+237) 233 40 29 00, and email address bibliotheque@enspd-udo.cm.

Figure 25 : interface de connexion

Pour avoir accès à l'application l'administrateur doit rentrer ses coordonnées ; puis le système vérifie dans sa base de données les identifiants entrés ; s'ils correspondent l'accès est octroyé.

b. Interface d'accueil

Une fois connecté une page de bienvenue s'affiche :

The screenshot shows the welcome page of the ENSPD library system. The layout is similar to the login page, with a dark blue header and footer. The main content area is light gray and features a white box in the center. The box is titled "BIENVENUE A LA BIBLIOTHEQUE DE l'ENSPD !!". Below the title, there is a green box with the text "Bienvenue très chrèr(e) MBOUMELA JAURÈS". Below this, it says "Dans cette bibliothèque, votre identifiant sera : 8". A blue "CONTINUER" button is positioned below the text. At the bottom of the page, there is a dark blue footer with three columns of text, identical to the one in Figure 25. The first column is titled "Bibliothèque ENSPD", the second "Liens Rapides", and the third "Contact".

Figure 26 : Page de bienvenue

En cliquant sur « Continuer » l'administrateur a alors accès à la page d'accueil :

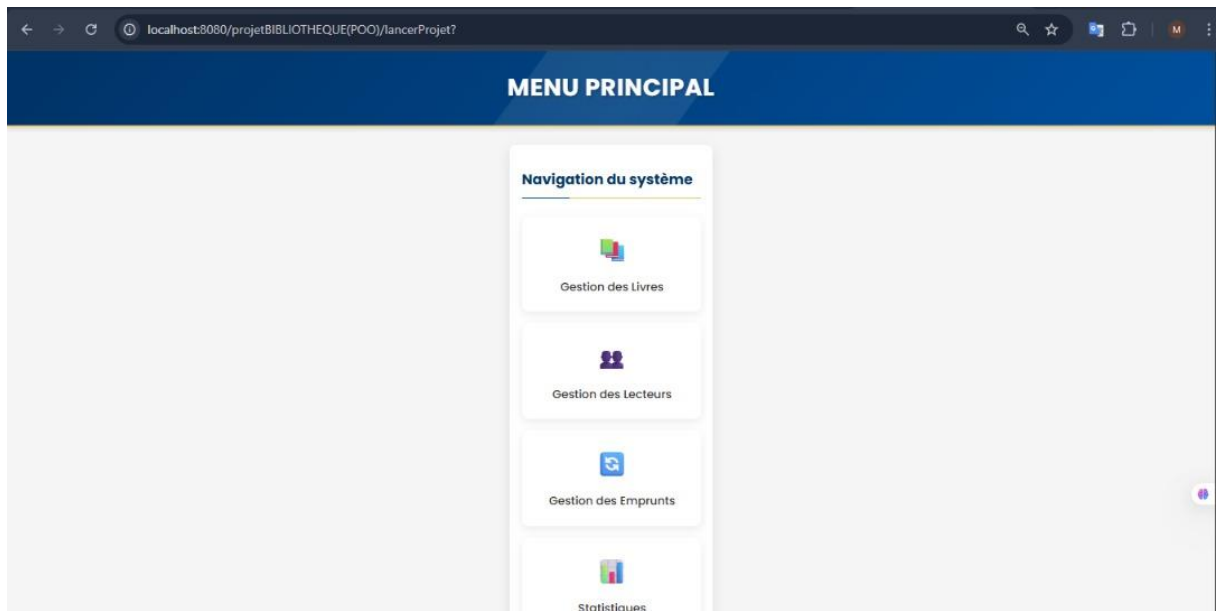
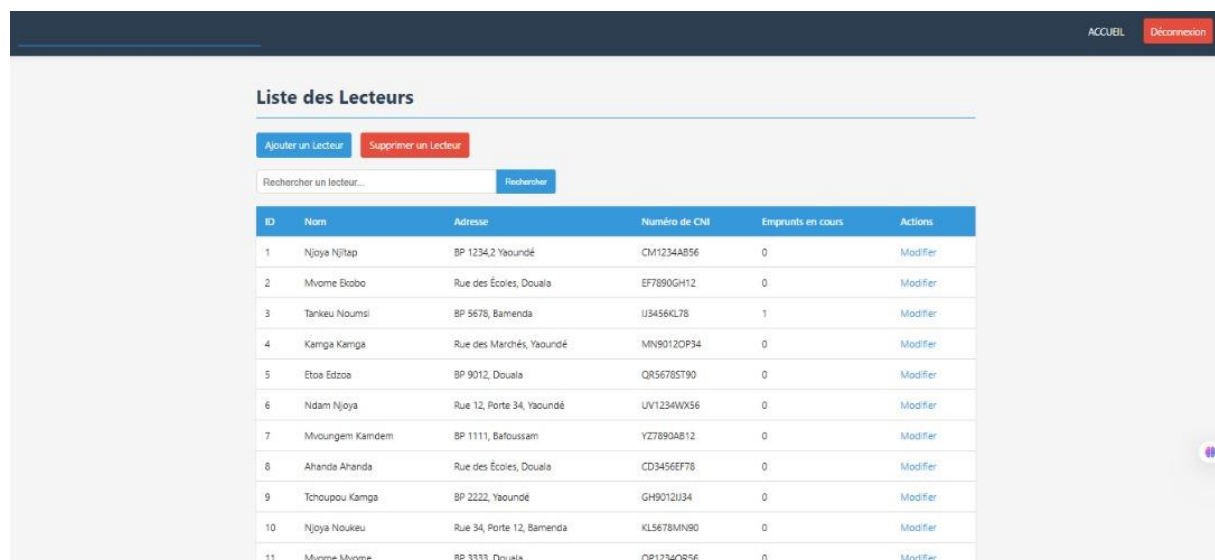


Figure 27 : Page d'accueil

Sur cette page nous avons les différents gestionnaires de notre bibliothèque.

c. Historique des lecteurs

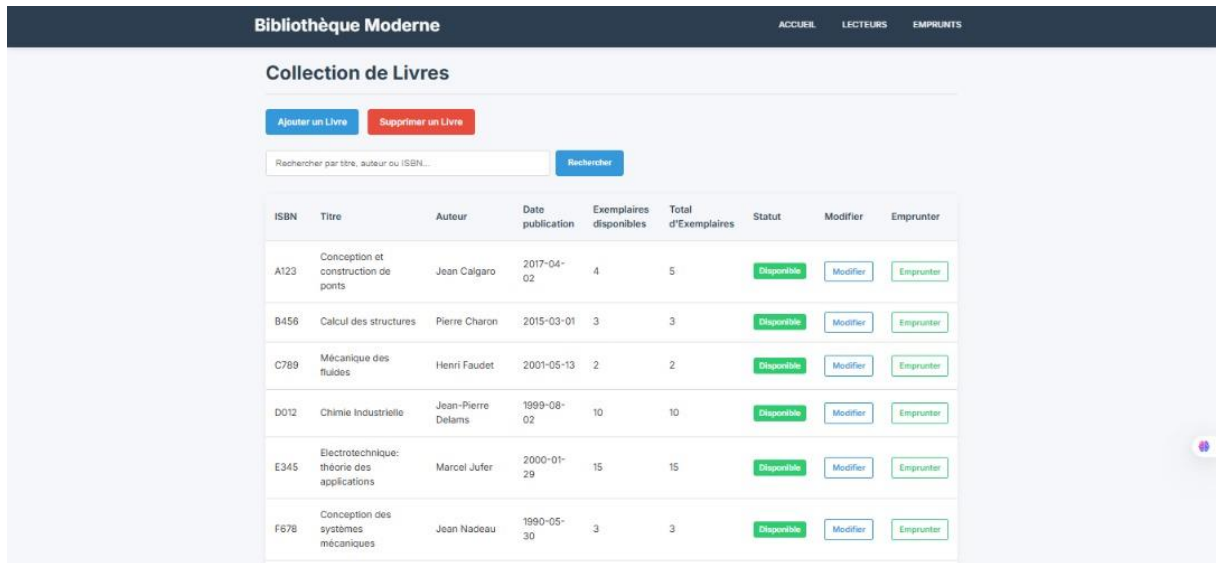


ID	Nom	Adresse	Numéro de CNI	Emprunts en cours	Actions
1	Njoya Njitap	BP 12342 Yaoundé	CM1234AB56	0	Modifier
2	Mvome Ekobo	Rue des Écoles, Douala	EF7890GH12	0	Modifier
3	Tankeu Nouns	BP 5678, Bamenda	IJ3456KL78	1	Modifier
4	Kanga Kanga	Rue des Marchés, Yaoundé	MN9012OP34	0	Modifier
5	Etoa Edzoa	BP 9012, Douala	QR5678ST90	0	Modifier
6	Ndam Njoya	Rue 12, Porte 34, Yaoundé	UV1234WX56	0	Modifier
7	Mvougem Kamdem	BP 1111, Batoussam	YZ7890AB12	0	Modifier
8	Ahanda Ahanda	Rue des Écoles, Douala	CD3456EF78	0	Modifier
9	Tchoupou Kanga	BP 2222, Yaoundé	GH9012IJ34	0	Modifier
10	Njoya Njoukeu	Rue 34, Porte 12, Bamenda	KL5678MN90	0	Modifier
11	Mvome Mvome	BP 3333, Douala	OP1234QR56	0	Modifier

Figure 28 : Historique des lecteurs

Ici nous retrouvons tous les lecteurs qui se sont fait enregistrés.

d. Inventaire de livres



ISBN	Titre	Auteur	Date publication	Exemplaires disponibles	Total d'Exemplaires	Statut	Modifier	Emprunter
A123	Conception et construction de ponts	Jean Calgato	2017-04-02	4	5	Disponible	Modifier	Emprunter
B456	Calcul des structures	Pierre Charon	2015-03-01	3	3	Disponible	Modifier	Emprunter
C789	Mécanique des fluides	Henri Faudet	2001-05-13	2	2	Disponible	Modifier	Emprunter
D012	Chimie Industrielle	Jean-Pierre Delams	1999-08-02	10	10	Disponible	Modifier	Emprunter
E345	Electrotechnique: théorie des applications	Marcel Jufer	2000-01-29	15	15	Disponible	Modifier	Emprunter
F678	Conception des systèmes mécaniques	Jean Nadeau	1990-05-30	3	3	Disponible	Modifier	Emprunter

Figure 29 : Inventaire des livres de la bibliothèque

Nous retrouvons tous les livres que possède la bibliothèque ; il y a la possibilité d'ajouter un livre quand il y a arrivage ou de supprimer quand il y a rupture.

e. Ajouter un lecteur

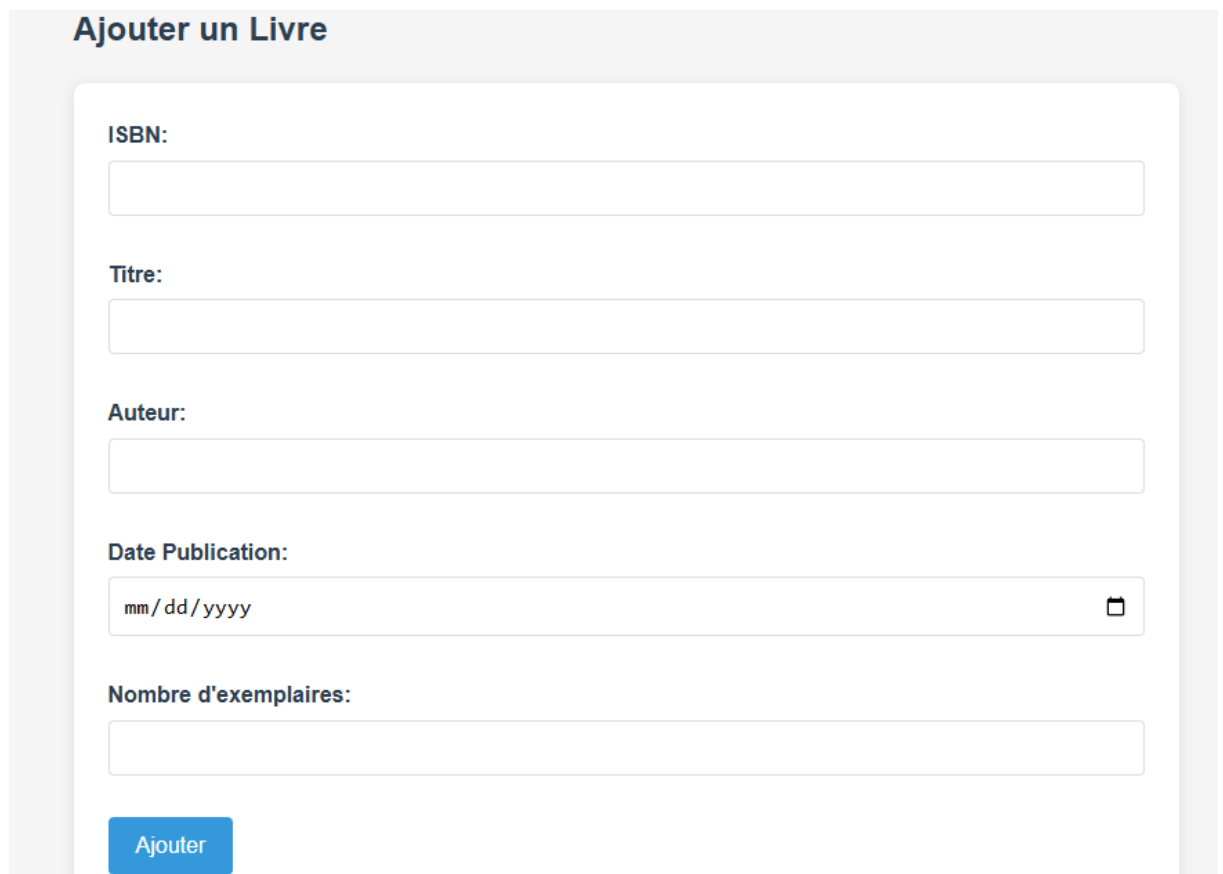


The screenshot shows a web interface with a light blue header. On the left, the word 'ACCEUIL' is in large blue letters. To its right, 'Ajouter un Lecteur' is in black text and underlined. Below this, there is a white form box with a blue border on the left. Inside the form, there are three input fields, each with a label and a red asterisk indicating it is required: 'Nom *', 'Adresse *', and 'Numéro de CNI *'. Each input field has a placeholder text: 'Entrez le nom du lecteur', 'Entrez l'adresse du lecteur', and 'Entrez le numéro de CNI' respectively. At the bottom left of the form is a blue button with the white text 'Ajouter'.

Figure 30 : Interface d'ajout d'un lecteur

Pour ajouter un nouveau lecteur l'administrateur doit rentrer le nom, adresse et numéro de cni de ce dernier. Une fois ces informations ajoutées il s'ajoute dans notre inventaire vu plus haut.

f. Ajouter un livre



The image shows a web form titled "Ajouter un Livre" (Add a Book). The form is contained within a light gray box with a white background. It features five input fields and a submit button. The fields are labeled "ISBN:", "Titre:", "Auteur:", "Date Publication:", and "Nombre d'exemplaires:". The "Date Publication:" field includes a date format hint "mm/dd/yyyy" and a calendar icon. The submit button is blue and labeled "Ajouter".


Ajouter un Livre

ISBN:

Titre:

Auteur:

Date Publication:



Nombre d'exemplaires:

Ajouter

Figure 31 : Interface d'ajout d'un livre

Pour ajouter un livre on entre son ISBN, son titre, son auteur, sa date de publication et le nombre d'exemplaire. Une fois ces informations entrées vous cliquez sur ajouter et l'inventaire des livres se mettra à jour automatiquement.

g. Historique des emprunts

Historique des Emprunts							
ID	ISBN	ID Lecteur	ID Bibliothécaire	Date Emprunt	Date Retour Prévue	Date Retour Effective	Actions
1	A123	3	8	2025-05-18	2026-02-18	2025-05-18	
2	A123	-3	8	2025-05-18	2025-05-29	2025-05-18	
3	C789	23	8	2025-05-19	2025-05-30	2025-05-19	
4	C789	23	8	2025-05-19	2025-05-30	2025-05-19	
5	C789	12	8	2025-05-19	2025-05-31	2025-05-19	

Figure 32 : Historique des emprunts

Ici nous retrouvons toutes les informations relatives aux emprunts notamment les dates d’emprunt et de retour ; les identifiants des bibliothécaires ayant validé l’emprunt et ceux des lecteurs ayant effectués les emprunts.

SECTION 3 : REPARTITION DES TACHES ET DIFFICULTEES RENCONTREES

3.1. REPARTITION DES TACHES.

Pour la réalisation optimale de ce projet nous avons réparti les taches de la façon suivante :

- **Equipe A (MBOUMELA)**

Codage en java des fonctionnalités du système et mode d'emploi.

- **Equipe B (MBOUOMBOUO)**

Codage front-end de l'application (Html, Css, javascript)

- **Equipe C (SIGNING)**

Confection base de donnée SQLite.

- **Equipe D (TAMBOU, KOUTA, MOMO)**

Réalisation des diagrammes UML

- **Equipe E (MBOUMA, MINKO, MOMO)**

Les différentes documentations et rapport du projet

- **Equipe F (KAMGANG, NGLITANG)**

Personnel de soutien.

3 .2. DIFFICULTEES RENCONTREES.

- La représentation graphique avec JAVA FX était plus difficile à comprendre que le JEE du fait qu'on connaît mieux le HTML
- La gestion des dépendances avec le fichier web.wml est à prendre avec précision
- Difficultés à connecter avec une base de données MySQL raison pour laquelle on a choisi SQLite
- Difficulté de mise en place d'une sécurité notoire car aucune authentification n'est prise en compte après la connexion

CONCLUSION

La réalisation de ce système de gestion de bibliothèque marque l'aboutissement d'un processus itératif et rigoureux. Nous avons pu concrétiser ce projet grâce à l'emploi de divers langages et méthodes. Le développement de l'interface utilisateur a bénéficié de la flexibilité et la richesse du HTML, CSS, JAVASCRIPT ; l'interaction avec la base de données a été réalisé avec JAVA ; la gestion et l'organisation des bases de données ont reposé sur un système de base de données relationnelle réalisé avec SQLTESTUDIO. Ce projet, bien que parvenant à ses objectifs initiaux reste limité dans le temps ; l'évolution technologique offre des perspectives d'amélioration continue. Des évolutions futures pourraient inclure l'intégration de technologies comme l'intelligence artificielle pour recommander des ouvrages aux lecteurs ou encore le développement d'une application mobile pour un accès facilité aux services de la bibliothèque. Ce projet constitue donc une base solide pour une gestion optimisée des bibliothèques, tout en ouvrant la voie à de nouvelles innovations pour enrichir l'accès au savoir et à la culture.

REFERENCES ET BIBLIOGRAPHIE

- ❖ <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=http://projet.eu.org/pedago/sin/term/3->
- ❖ <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://perso.liris.cnrs.fr/christine.solnon/coursUML>.

TABLE DES MATIERES

INTRODUCTION.....	6
SECTION 1 : CONTEXTE ET OBJECTIFS DE LA MISSION.....	7
1.1. CONTEXTE DE LA MISSION	5
1.2. SPECIFICATIONS FONCTIONNELLES	5
1.3. SPECIFICATIONS NON-FONCTIONNELLES	6
1.4. EVALUATION DES MESURES DE SUCCES	7
SECTION 2 : L'EXECUTION DE LA MISSION	8
2.1. LOGICIELS ET LANGAGES UTILISES.....	8
a. Eclipse	
b.Tomcat version 10.....	8
c.SQLitestudio.....	8
d.StarUML.....	9
e. Visual Studio Code.....	9
f. CSS	9
g. HTML.....	9
h. Java JDK	9
2.2. ETUDE DE L'EXISTANT	10
a. Description de l'existant	10
b. Critique de l'existant.....	11
2.3. PRESENTATION DE L'OUTIL DE MODELISATION UTILISE.....	14
2.3.1. Choix de la méthode d'utilisation.....	14
2.3.1.1. Cycle de vie.....	15
2.3.1.2. Présentation du langage de modélisation UML.....	17
2.3.1.3. Les principaux diagrammes UML.....	18
2.3.2. Modélisation du système.....	19
2.4. PRESENTATION DE L'APPLICATION.....	38
SECTION 3 : REPARTITION DES TACHES ET DIFFICULTEES RENCONTREES.....	40
3.1. REPARTITION DES TACHES.....	40
3.2DIFFICULTEES RENCONTREES	40
CONCLUSION.....	41
REFERENCES ET BIBLIOGRAPHIE.....	42

CONCEPTION ET REALISATION D'UN SYSTÈME DE GESTION DE :
BIBLIOTHEQUE