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

PROJET DE POO : DOCUMENT TECHNIQUE

THEME : APPLICATION DE GESTION D'UNE BIBLIOTHEQUE

Rédigé par :

NOMS & PRENOMS	MATRICULE
FOHOM GAINGNE MERVEILLES DIANE	22G00136
FOUEJIO NGUEFACK NILIMA JODIE	24G01092
GUEMBU SIME GUY VALAIRE	22G00460
GUEMOGNE GERALDINE ESTHER	22G00153
MEKOUNDE ELAME GUSTAVE CYRIAC	22G00253
MOHAMMED EL BACHIR ISMAÏLA	22G00262
NKAMOLOUM TENE DAMIEN ANTHONY	22G00328
NONGNING LELE STEVE JORDAN	24G01122
NTAKEU BAPIAH LEANE YVANNA	22G00339
TIEMOUO DJIOLIO DORCAS	24G01134

Sous la supervision de :

Dr NOULAPEU

Année académique 2024/2025

SOMMAIRE

INTRODUCTION

La gestion efficace d'une bibliothèque nécessite une solution logicielle capable de centraliser les opérations liées aux livres, aux lecteurs, aux emprunts et à l'administration du système. Dans ce contexte, la présente étude vise à développer une application de gestion de bibliothèque destinée à être utilisée en local par un responsable. Cette solution devra répondre à un ensemble d'exigences fonctionnelles et non fonctionnelles précises, tout en s'adaptant aux contraintes techniques du projet.

Ce document technique présente dans un premier temps une analyse détaillée des besoins à satisfaire, avant d'aborder la modélisation UML et la conception du système, éléments indispensables à une mise en œuvre rigoureuse, modulaire et maintenable.

CHAPITRE I : ANALYSE DES EXIGENCES DU PROJET

Cette section a pour objectif de définir les fonctionnalités attendues ainsi que les contraintes techniques à respecter. L'application de gestion de bibliothèque visée devra permettre au responsable de la bibliothèque d'assurer une gestion efficace des livres, des emprunts et des retours, tout en garantissant une interaction fluide.

I.1 EXIGENCES FONCTIONNELLES

Avant de concevoir l'architecture du système, il est essentiel de définir clairement les fonctionnalités attendues. L'application devra offrir les services suivants :

❖ **Gestion des livres**

- Permettre au responsable de la bibliothèque d'ajouter de nouveaux livres en renseignant leur titre, auteur, ISBN et année de publication.
- Autoriser le responsable de la bibliothèque à modifier les informations des livres existants.
- Permettre au responsable de la bibliothèque de supprimer un livre de la bibliothèque si nécessaire.
- Afficher l'inventaire des livres disponibles afin de faciliter leur consultation.

❖ **Gestion des emprunts et des retours**

- Enregistrer l'emprunt d'un livre en associant l'utilisateur, la date d'emprunt et le livre concerné. Le responsable de la bibliothèque devra valider cet emprunt.
- Assurer l'enregistrement du retour d'un livre avec mise à jour de son statut, sous la supervision du responsable de la bibliothèque.
- Offrir une fonctionnalité de consultation de l'historique des emprunts et des retours, par livre ou par utilisateur, accessible au responsable de la bibliothèque.

❖ **Gestion des lecteurs**

- Consultation de la liste des lecteurs : Affiche tous les lecteurs inscrits et Permet de filtrer par nom via une barre de recherche.

- Modification des informations d'un lecteur : Ouvre un formulaire de modification des coordonnées, Permet de mettre à jour le nom et les informations personnelles et Enregistre les modifications.
- Ajout d'un nouveau lecteur : Ouvre un formulaire pour saisir les informations d'un nouveau lecteur et Enregistre le nouveau lecteur dans le système.
- Suppression d'un lecteur : Permet d'accéder au profil d'un lecteur pour suppression, affiche une fenêtre de confirmation avant de retirer le lecteur et confirme la suppression et met à jour la liste.

❖ **Gestion des exemplaires d'un livre**

- Consultation des exemplaires disponibles : affiche la liste des exemplaires associés à un livre dans sa fiche.
- Ajout d'un nouvel exemplaire : Permet de saisir l'état et la disponibilité d'un nouvel exemplaire et enregistre l'exemplaire dans le système.
- Modification d'un exemplaire : Ouvre un formulaire pour modifier les détails d'un exemplaire existant.
- Suppression d'un exemplaire : Permet de supprimer un exemplaire après confirmation de l'action.

❖ **Gestion du compte responsable**

- Accès à la section « Gestion du compte » : permet au responsable d'accéder à ses paramètres de compte.
- Modification des informations du responsable : Ouvre un formulaire pour mettre à jour les informations personnelles (nom, contact), permet de changer le mot de passe et enregistre les modifications après validation.

I.2 EXIGENCES NON FONCTIONNELLES

❖ **Accessibilité**

- Interface utilisateur intuitive et accessible.

- Compatible avec différents appareils (PC, tablettes, smartphones).

❖ **Performance**

- Temps de réponse rapide lors de la recherche dans le catalogue.
- Capacité à gérer un grand nombre d'utilisateurs simultanément

❖ **Sécurité**

- Protection des données personnelles des utilisateurs.
- Authentification sécurisée pour l'accès aux fonctionnalités sensibles.

❖ **Scalabilité**

- Possibilité d'ajouter de nouvelles fonctionnalités à l'avenir (ex. : réservations en ligne, suggestions de livres).

I.3 EXIGENCES TECHNIQUES

Le système devra respecter les contraintes techniques suivantes afin d'assurer son bon fonctionnement et sa pérennité :

- **Langage de programmation** : L'application sera développée en Java (version 8 ou supérieure).
- **Approche de développement** : Utilisation du paradigme de la programmation orientée objet pour une conception modulaire et réutilisable.
- **Interface utilisateur** : Fonctionnement en mode console ou graphique interactif pour une utilisation simplifiée.
- **Persistance des données (optionnel)** : Les informations pourront être sauvegardées dans des fichiers texte **(CSV, JSON)** ou dans une base de données pour assurer leur conservation entre plusieurs exécutions.
- **Environnement de développement** : Utilisation d'un IDE tel que **NetBeans**, **Eclipse** ou **IntelliJ IDEA**.
- **Versioning (optionnel)** : Adoption de Git pour suivre l'évolution du projet et faciliter la collaboration.

Cette analyse des exigences fonctionnelles et techniques constitue une base solide pour la conception du système. Dans la section suivante, nous nous intéresserons à la modélisation du système à l'aide de diagrammes UML afin de mieux structurer son architecture et ses interactions.

CHAPITRE II : MODÉLISATION ET CONCEPTION DU SYSTÈME

II.1 MODELISATION UML

La modélisation **UML (Unified Modeling Language)** permet de représenter de manière visuelle l'organisation et le fonctionnement de l'application avant son implémentation. Dans cette section, nous allons utiliser trois types de diagrammes UML essentiels :

- **Le diagramme de cas d'utilisation** : Il permet d'illustrer les principales fonctionnalités du système ainsi que l'interaction entre les utilisateurs et l'application.
- **Le diagramme de classes** : Il représente la structure interne du système en détaillant les objets, leurs attributs, leurs méthodes ainsi que les relations entre eux.
- **Le diagramme de séquence :** qui illustre les interactions entre objets ou acteurs d'un système au fil du temps, montrant l'ordre des messages échangés pour réaliser une fonctionnalité spécifique.

II.1.1 Diagramme De Cas D'utilisation

Le diagramme de cas d'utilisation permet d'identifier les principales actions réalisables au sein du système. Il représente les interactions entre l'utilisateur et les fonctionnalités offertes par l'application.

L'unique **acteur** de l'application est **le « Responsable De La Bibliothèque »**. Ce choix s'explique par la nature du système, qui est une application desktop installée localement sur les ordinateurs des responsables de la bibliothèque. Contrairement à un système en ligne avec plusieurs niveaux d'accès, cette solution garantit que seules les personnes autorisées puissent gérer les livres et les emprunts.

Le responsable de la bibliothèque a un accès exclusif à toutes les fonctionnalités du système, notamment :

- L'ajout, la modification et la suppression des livres.
- L'enregistrement et le suivi des emprunts et des retours.

- La consultation de l'inventaire des livres et de l'historique des emprunts.

La figure ci-dessous illustre le diagramme de cas d'utilisation du système de gestion d'une bibliothèque.

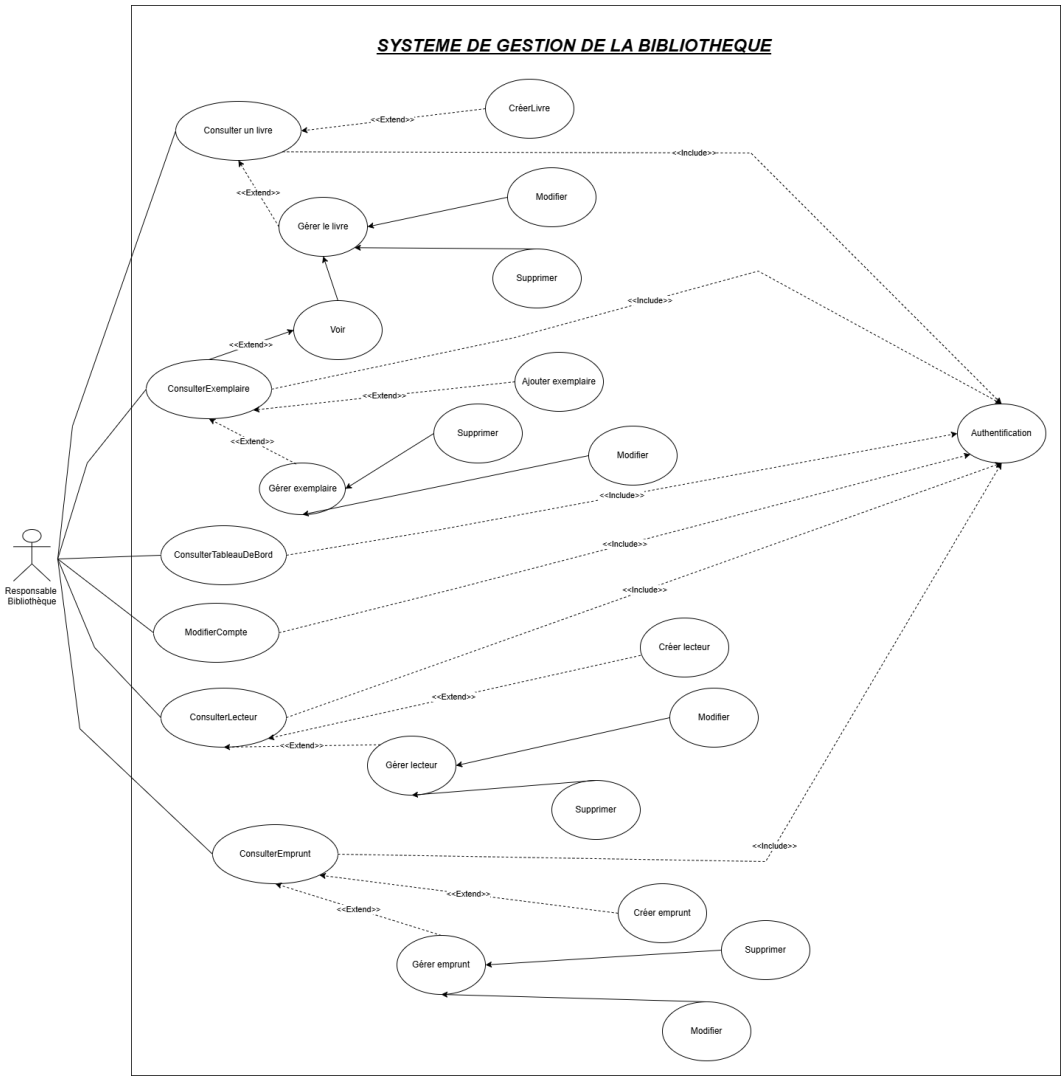


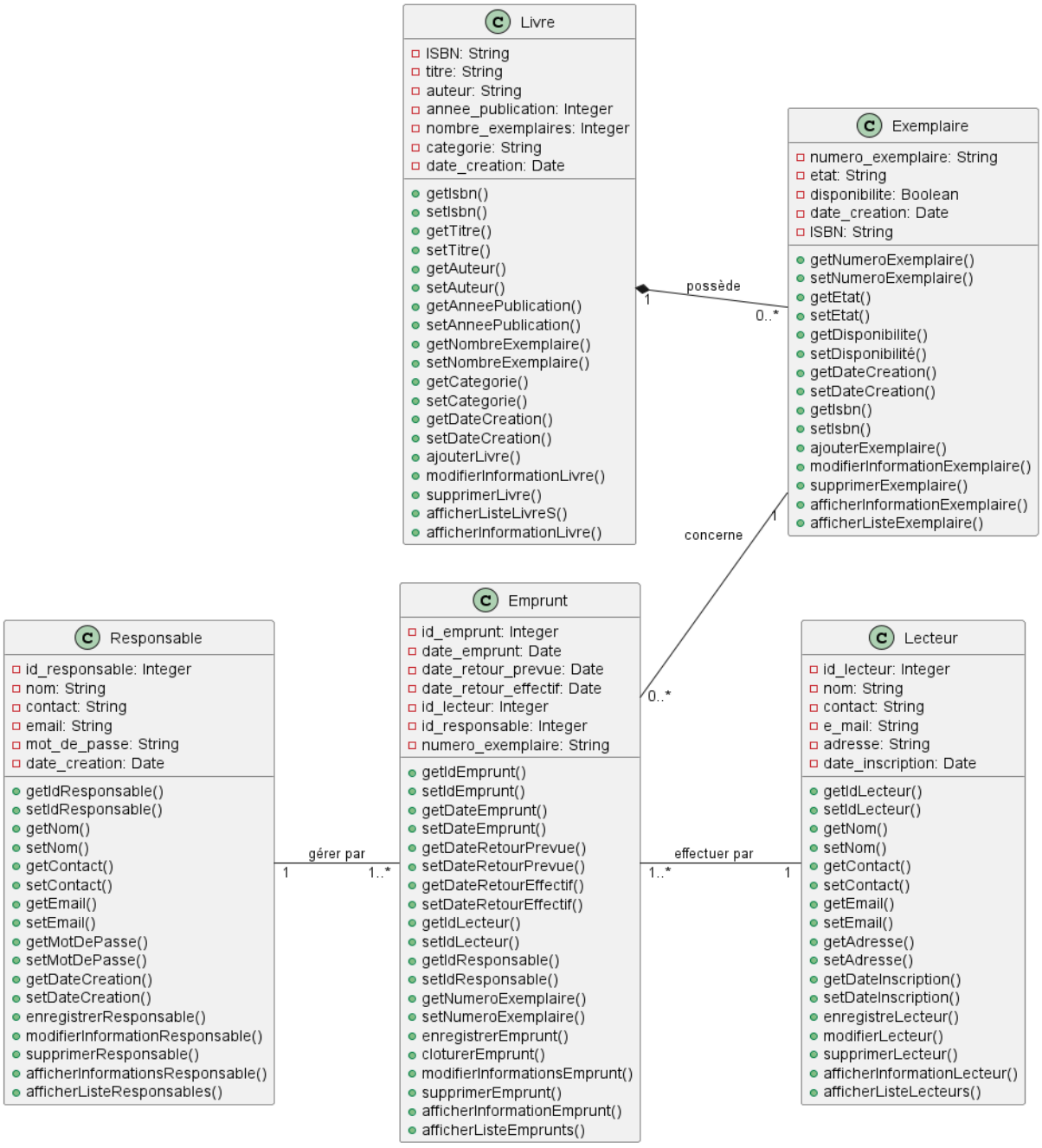
Figure 1: Diagramme de cas d'utilisation du système

II.1.2 Diagramme De Classe

Le diagramme de classes représente la structure de notre système de gestion de bibliothèque, en détaillant les principales entités et leurs interactions. Voici une explication concise de chaque classe :

- **Livre :** Cette classe représente les livres disponibles dans la bibliothèque. Elle contient des informations telles que le **titre, l'auteur**, **la catégorie**, **le nombre de pages**, **le nombre d'exemplaires disponibles** et **la date de création**. Les méthodes associées permettent d'ajouter, de supprimer, de modifier et d'afficher les informations des livres.
- **Exemplaire :** La classe Exemplaire représente les copies individuelles d'un livre. Elle inclut des détails **comme l'identifiant de l'exemplaire, son état, sa disponibilité et la date de création**. Les méthodes permettent de gérer les exemplaires, notamment en les ajoutant, les supprimant, modifiant leur état et affichant leurs informations.
- **Emprunt :** Cette classe gère les emprunts de livres par les lecteurs. Elle contient des informations sur **l'identifiant de l'emprunt, la date d'emprunt et la date de retour**. Les méthodes permettent **d'enregistrer, de clôturer, de supprimer** et **d'afficher les emprunts**.
- **Lecteur :** La classe Lecteur représente les utilisateurs de la bibliothèque. Elle inclut des informations telles que **l'identifiant du lecteur, son nom, ses coordonnées, son adresse e-mail et son adresse physique**. Les méthodes permettent **d'enregistrer, de modifier, de supprimer** et **d'afficher les informations des lecteurs**.
- **Responsable :** La classe Responsable représente les administrateurs de la bibliothèque. Elle inclut des informations comme **l'identifiant du responsable, son nom, ses coordonnées, son email** et **son mot de passe**. Les méthodes permettent de gérer les informations des responsables, notamment en les enregistrant, les modifiant et les supprimant.

Concernant les relations, un **livre** peut être lié de plusieurs **exemplaires**, et la suppression d'un livre entraîne celle de ses exemplaires, ce qui se traduit par une relation de dépendance. Chaque **emprunt** est lié à un exemplaire spécifique et un lecteur peut effectuer plusieurs emprunts. Un responsable gère plusieurs emprunts, facilitant le suivi des emprunts sous sa supervision.



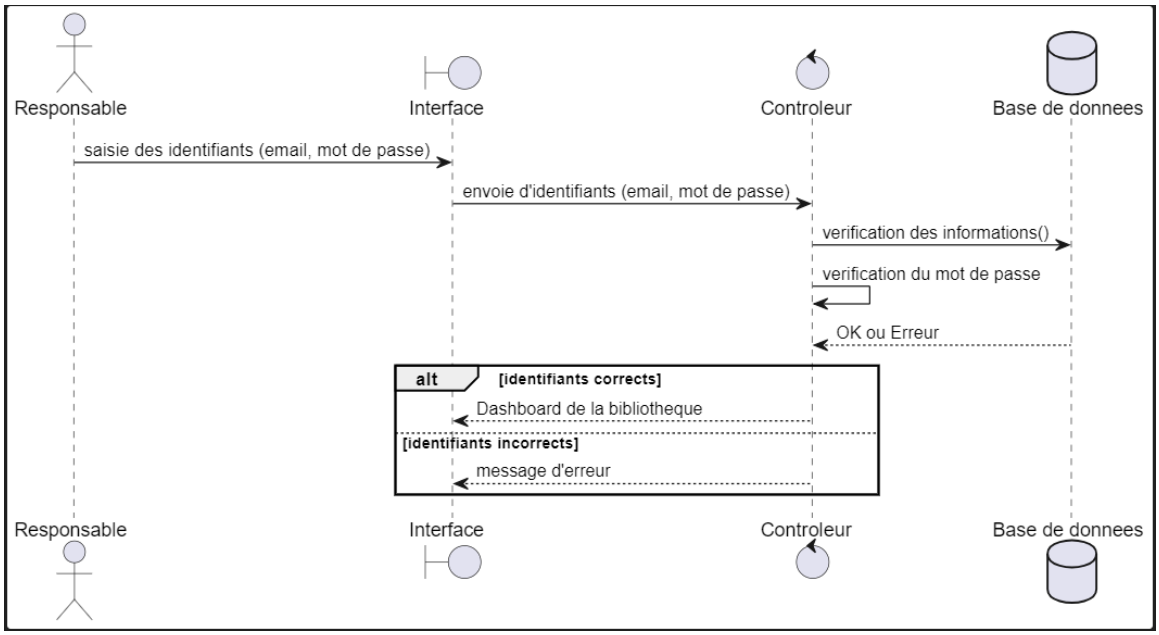
II.1.3 Diagramme De Séquence

Les deux diagrammes de séquence ci-dessous illustrent clairement le flux d'interactions et les processus impliqués dans la l'emprunt d'un livre et la connexion du responsable.

- **Diagramme de séquence pour la connexion du responsable**

✓ **Acteurs et objets :**
- Responsable : initie la connexion.
- Interface : gère l'affichage et la saisie des informations.
- Contrôleur : traite la logique d'authentification.
- Base de données : stocke les informations d'identification.

✓ **Flux d'interactions :**
- Le responsable saisit ses identifiants (email, mot de passe) via l'interface.
- Ces informations sont envoyées au contrôleur pour vérification.
- Le contrôleur consulte la base de données pour valider les identifiants.
- Si les identifiants sont corrects, le responsable accède au tableau de bord ; sinon, un message d'erreur est affiché.

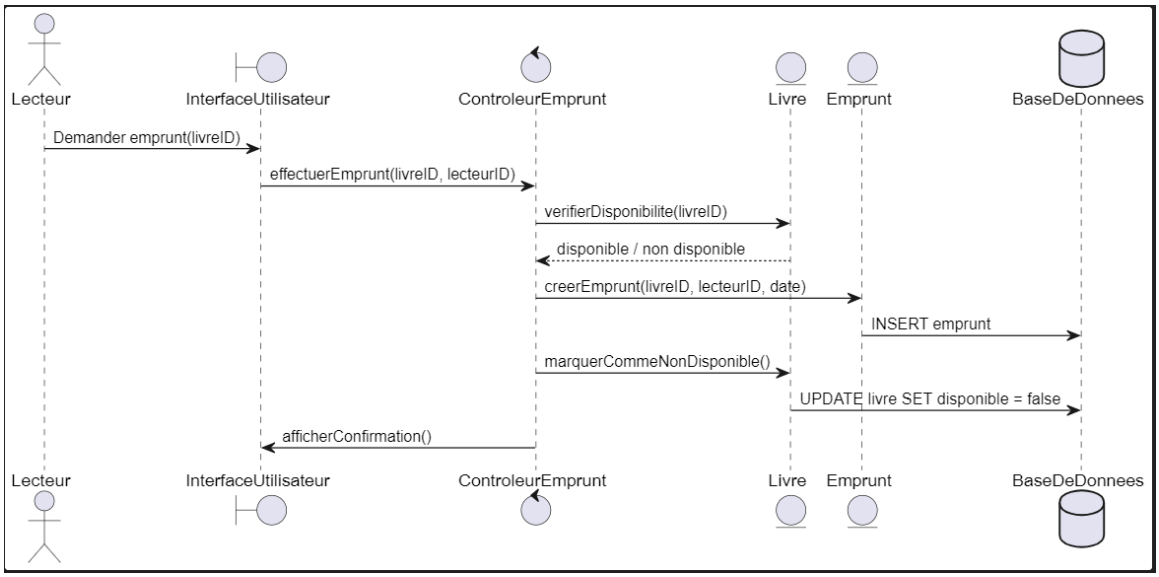


❖ **Diagramme de séquence pour l'emprunt d'un livre**

✓ **Acteurs et objets :**
- Lecteur : initie l'emprunt.
- Interface Utilisateur : gère les interactions avec le lecteur.
- Contrôleur Emprunt : traite la logique d'emprunt.
- Livre : représente l'élément emprunté.
- Base de Données : stocke les informations.

✓ **Flux d'interactions :**

- Le lecteur demande un emprunt en fournissant l'ID du livre.
- L'interface utilisateur transmet cette demande au contrôleur.
- Le contrôleur vérifie la disponibilité du livre dans la base de données.
- Si le livre est disponible, il crée un enregistrement d'emprunt et met à jour la base de données.
- Le contrôleur informe l'interface de la confirmation de l'emprunt.



II.2 CONCEPTION DU SYSTEME

La conception du système repose sur une architecture logicielle robuste, évolutive et adaptée aux besoins spécifiques d'une bibliothèque universitaire. L'objectif principal est de fournir une solution intuitive et fiable, destinée uniquement aux responsables de la bibliothèque. Ainsi, la simplicité d'usage, la clarté de l'interface, la sécurité des données et la maintenabilité du code ont guidé nos choix de conception. Le système est bâti selon les principes de **la programmation orientée objet** et respecte une architecture modulaire en couches, facilitant à la fois la collaboration en équipe, les tests unitaires, la réutilisabilité des composants, et l'intégration continue.

II.2.1 Choix De L'architecture

Nous avons adopté une architecture **MVC (Modèle-Vue-Contrôleur)**, bien connue pour sa clarté dans la séparation des responsabilités, afin d'assurer une meilleure organisation du code et

une évolutivité à long terme. Cette architecture est subdivisée en trois principales couches fonctionnelles :

❖ **Couche Présentation (Vue)**

Elle correspond à l'interface utilisateur, et permet l'interaction entre les responsables de la bibliothèque et le système.

- **Technologies utilisées** : JSP, CSS, JSTL (pour les boucles et conditions dynamiques).
- **Fonctionnalités typiques** :
  - Saisie des identifiants (login).
  - Navigation via un tableau de bord.
  - Gestion des entités : lecteurs, livres, emprunts, exemplaires.
- **Exemples d'interfaces** :
  - **index.jsp** : page d'authentification avec gestion des erreurs.
  - **dashboard.jsp** : statistiques clés sur les activités de la bibliothèque.
  - **form_emprunt.jsp, livres.jsp, compte.jsp**, etc.

❖ **Couche Métier (Contrôleur + Logique)**

Cette couche constitue le cœur de l'application. Elle implémente les règles de gestion et assure la coordination entre les vues et les données.

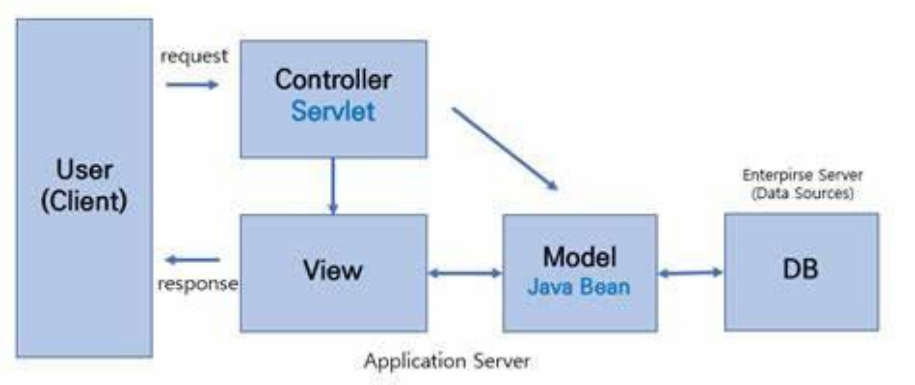
- **Responsabilités** :
  - Authentifier les utilisateurs (via **AuthServlet**).
  - Appliquer les règles métiers (ex. : nombre maximum d'emprunts par lecteur).
  - Gérer les opérations sur les entités (**CRUD** : création, lecture, mise à jour, suppression).
- **Fonctions typiques** :
  - **authentifierResponsable()**
  - **effectuerEmprunt(), clôturerRetour()**
  - **ajouterLivre(), modifierLecteur(),** etc.

Chaque servlet est dédiée à une entité ou une fonctionnalité précise (par exemple : **LivreServlet**, **LecteurServlet**, **DashboardServlet**, etc.), garantissant ainsi une bonne lisibilité et testabilité du code.

❖ **Couche Persistance (Modèle/DAO)**

La couche de persistance est chargée de l'interaction avec la base de données (MySQL). Elle repose sur le modèle **DAO (Data Access Object)**, qui permet de centraliser et structurer les requêtes SQL.

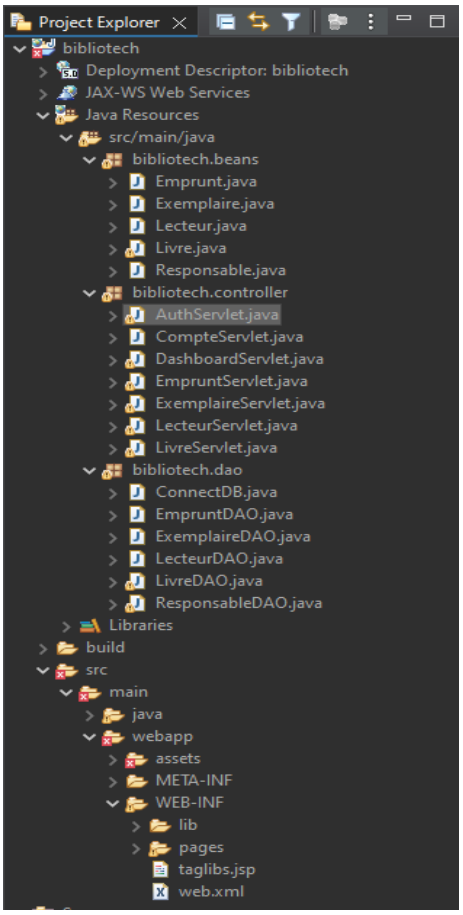
- **Responsabilités** :
  - Exécuter les requêtes SQL (SELECT, INSERT, UPDATE, DELETE).
  - Mapper les résultats en objets métiers.
  - Gérer les connexions à la base de données de manière sécurisée et efficace.
- **Bonnes pratiques appliquées** :
  - **DAO** par entité **: LivreDAO, LecteurDAO, EmpruntDAO**, etc.
  - Requêtes paramétrées pour éviter les injections SQL.
  - Indexation des colonnes critiques pour optimiser les performances de recherche.
  - Gestion fine des exceptions et des transactions.



II.2.2 Bénéfice de cette conception

L'architecture choisie apporte de nombreux avantages qui ont été constatés tout au long du développement du projet :

- **Modularité** : chaque couche peut évoluer indépendamment. Par exemple, l'interface peut être remplacée par une application mobile sans impact sur les règles métier.
- **Maintenance facilitée** : l'isolement des responsabilités rend le débogage plus simple et la modification plus sûre.
- **Sécurité renforcée** : les accès aux données sont exclusivement réalisés via la couche métier et persistance, empêchant toute manipulation non contrôlée.
- **Optimisation des performances** : la séparation logique des composants permet d'affiner l'analyse des goulets d'étranglement, notamment au niveau SQL, en utilisant des outils comme `EXPLAIN` pour améliorer les requêtes.
- **Facilité de test** : chaque servlet, DAO ou vue peut être testée indépendamment, favorisant ainsi une meilleure fiabilité globale du système.



CONCLUSION

Ce travail a permis d'établir une base solide pour le développement d'une application de gestion de bibliothèque fiable et performante. L'analyse des exigences a clairement identifié les fonctionnalités essentielles ainsi que les contraintes techniques à respecter. La modélisation UML a ensuite permis de visualiser de manière structurée les composants du système, les interactions entre les utilisateurs et les entités, ainsi que les séquences d'utilisation. La conception qui en découle garantit une architecture logicielle claire, facilitant l'implémentation future, les évolutions possibles et la maintenance. L'ensemble de cette démarche contribue à la création d'un outil adapté aux besoins réels d'une bibliothèque moderne, tout en assurant sécurité, performance et évolutivité.