



**CY Books**

# **Projet « CY Books » : Application de gestion pour bibliothécaire**

**UE – PROJET GÉNIE LOGICIEL**

BAH Sékou  
CLOVIS Betsaleel  
JULI Rithan  
LAZIZI Yassine  
MELAIMI Anis  
Groupe 10

Professeure encadrante : HAWARI Sirine

2023 / 2024

ING1 GI Groupe 10

I. Introduction.....	3
II. Analyse du projet.....	4
1) Contexte.....	4
2) Sujet.....	4
III. Partie technique.....	5
1) Conception et structure globale du projet.....	5
2) Manuel Technique du Projet CY BOOKS.....	7
a) Technologies et outils.....	7
b) Fonctionnement.....	9
c) Aspect technique.....	14
d) Limitations fonctionnelles de la plateforme.....	17
IV. Partie travail et gestion de projet.....	17
1) Organisation du travail dans le groupe.....	17
V. Conclusion.....	21
VI. Annexe.....	22
1) Script SQL.....	22

# I. Introduction

Le présent rapport met en évidence un projet de développement d'une application de gestion de bibliothèque pour les bibliothécaires, elle a été réalisée dans le cadre de l'UE Projet, dans le module *Projet Génie Logiciel*. L'objectif principal de ce projet est de créer une application novatrice et intuitive permettant à différents bibliothécaires de pouvoir rechercher, ajouter et supprimer des adhérents, tout en ayant la possibilité de gérer les emprunts. De plus, la recherche de livre via la base de données de la BNF y est intégrée, ainsi qu'un volet donnant des statistiques sur les livres les plus empruntés.

Le développement de cette plateforme repose sur une combinaison de langage et de conception d'interface graphique avec JavaFX, SQL et SceneBuilder. Ces technologies permettent de créer une interface ergonomique offrant une large gamme de fonctionnalités adaptées aux différents besoins.

Au cours de ce rapport, nous aborderons les différentes étapes du processus de développement technique, notamment la structure du projet, la conception de l'interface utilisateur et les fonctionnalités de gestion des usagers, des livres et des emprunts. Pour finir, nous accorderons également une attention particulière au déroulé complet du projet ainsi qu'à l'organisation du travail effectué.

## II. Analyse du projet

### 1) Contexte

Ce projet a été réalisé par notre équipe dans le cadre de l'UE Projet pour l'année scolaire 2023/2024. Nous avons collaboré sur ce projet pendant la fin du deuxième semestre, en utilisant les compétences acquises en JAVA, Scene Builder, SQL, et CSS pour développer une application de gestion de bibliothèque nommée "CY Books", répondant aux besoins des bibliothécaires.

Le projet s'est principalement déroulé dans un environnement d'apprentissage, avec pour objectif principal d'appliquer les concepts et compétences acquis en cours à un projet concret, tout en acquérant de nouvelles compétences comme l'utilisation des API. Il a également servi d'introduction au travail en équipe dans un contexte professionnel. En effet, nous avons respecté des schémas d'organisation d'entreprise, telles que les méthodes agiles (utilisation de Jira), participé à des réunions régulières (daily et weekly), et fait des retour de nos progrès à notre responsable, tout en respectant des deadlines et un cahier des charges prédéfini.

### 2) Sujet

Le projet vise à la conception d'une application de gestion d'une bibliothèque à travers la mise en place d'une application intuitive. Cette application doit offrir la possibilité de gérer les adhérents, gérer les emprunts, chercher des livres et avoir des statistiques liés à la bibliothèque.

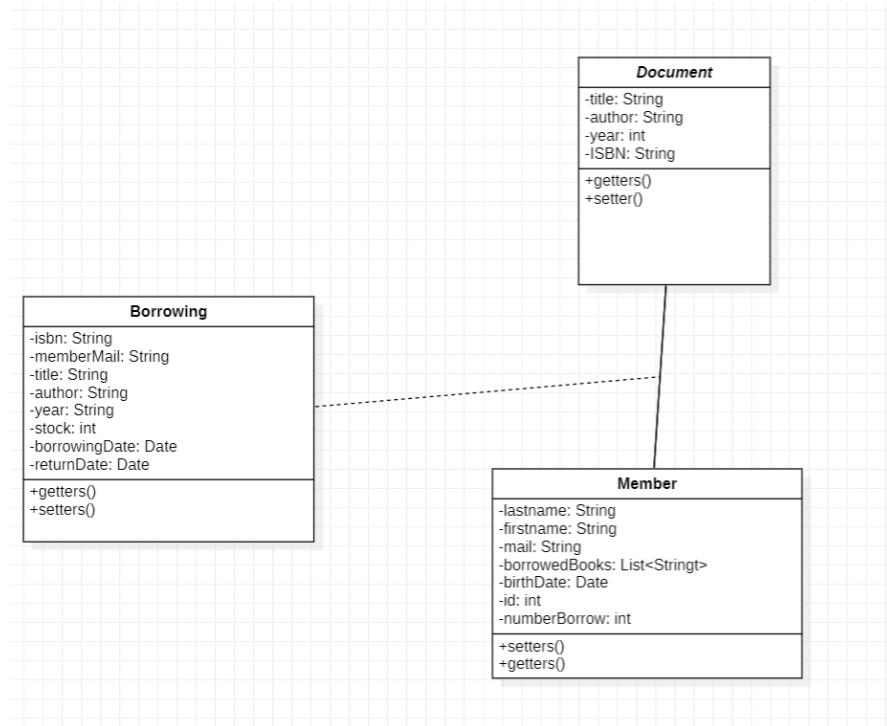
Le projet CYBooks devra respecter le cahier des charges fourni, avec les principales fonctionnalités souhaitées :

FONCTIONNALITÉS DU PROJET	
	<ul style="list-style-type: none"><li>Cette application doit être utilisée uniquement par le bibliothécaire, lui permettant de gérer les usagers (notamment les inscriptions, mais aussi les modifications d'informations, ou la recherche d'un usager bien précis), gérer les livres (recherche multi-critères), et les emprunts.</li><li>Cette application doit inclure un temps maximal d'emprunt par livre, ainsi qu'un nombre maximal de livres empruntés par le même usager à la fois. Des alertes, ou des affichages contrastés doivent permettre aux bibliothécaires de voir les problèmes d'emprunts actuels. Une liste des retards doit également être affichée à la demande.</li><li>Un historique des emprunts, dates, durées et retards, doit être possible lors de l'affichage du profil d'un usager spécifique.</li><li>La recherche de livres doit se faire avec différents critères de filtrage : ces critères doivent ensuite être envoyés dans l'API de la BNF afin de récupérer la liste des livres possibles.</li><li>Les résultats des recherches doivent être affichés page par page si le nombre est trop élevé.</li><li>Le stockage des emprunts doit se faire en local, soit par l'intermédiaire d'une base de données, soit via des fichiers : ce choix technique est laissé libre.</li><li>Il est possible aussi d'afficher également une liste des livres les plus empruntés pendant les 30 derniers jours.</li><li>Votre application doit être logique et ne pas autoriser des actions telles qu'assigner un emprunt à une date passée, permettre l'emprunt d'un livre si il est déjà emprunté (on part du principe qu'il n'y a que X exemplaires de chaque livre, X étant une constante à définir), ... .</li><li>Si une base de données locale est utilisée, c'est à votre programme de lancer le serveur si il ne l'est pas déjà en faisant un appel système.</li></ul>

### III. Partie technique

#### 1) Conception et structure globale du projet

- **Diagramme de classes**



Initialement, nous avions conçu un premier diagramme de classes qui comprenait des classes filles de Document : Roman, Bande dessinée, Nouvelle, Conte, etc. De plus, dans la classe Document, nous avions inclus un attribut éditeur et une classe associée Topics, de sorte que la classe Document possédait un attribut “topics : list<Topics>”.

Cependant, au cours de l'avancement du projet, nous avons constaté plusieurs limitations imposées par l'API et les données de la Bibliothèque Nationale de France (BNF) :

Recherche de l'éditeur: L'API ne permettait pas de rechercher l'éditeur des documents. Par conséquent, l'attribut éditeur dans la classe Document ne pouvait pas être correctement implémenté.

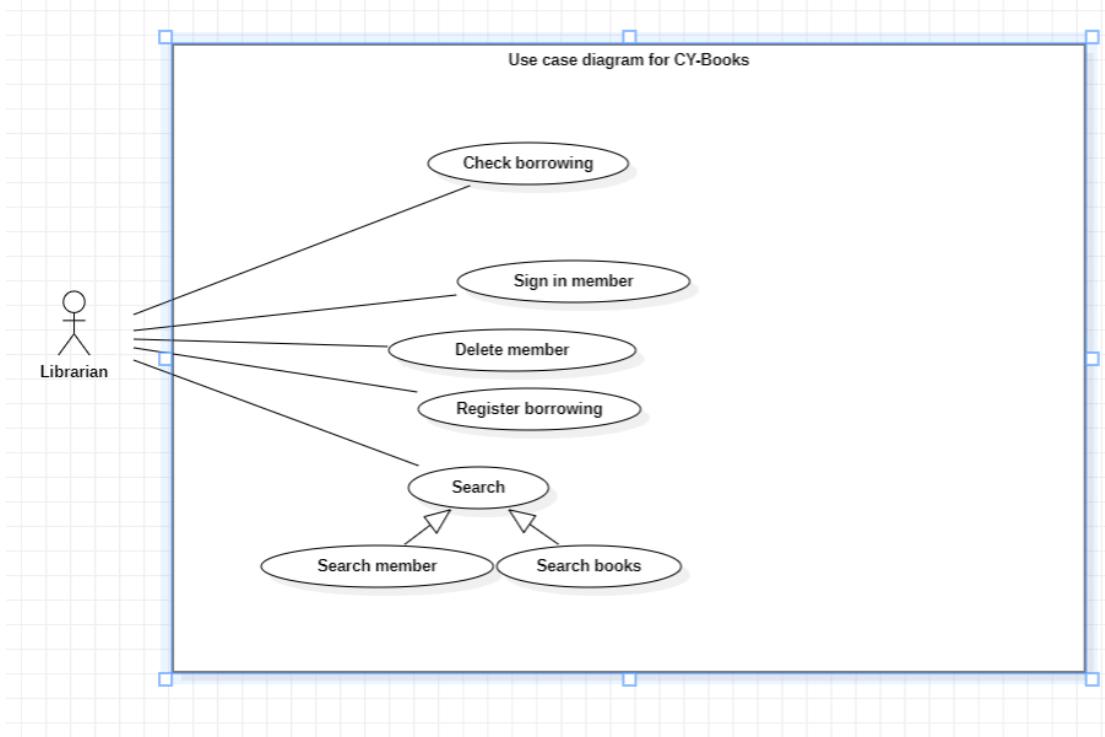
Classification des documents: Dans la BNF, les documents ne sont pas classés en catégories telles que Roman, BD, Conte, etc. Cela a rendu notre classification initiale obsolète et non applicable aux données réelles que nous manipulons.

Par ailleurs l'API nous fournit un id pour chacun des livres que nous allons récupérer en tant que ISBN.

En conséquence, nous avons dû revoir notre modèle conceptuel pour aligner notre conception avec les capacités et les limitations de l'API ainsi que les données disponibles dans la BNF. Cela impliquait la simplification de notre diagramme de classes (en 3 classes) et l'abandon de la catégorisation des documents. Notre modèle révisé se concentre désormais sur des structures de données plus génériques, compatibles avec les informations réellement accessibles et manipulables via l'API et les ressources de la BNF.

Par ailleurs, la récupération des informations depuis la base de données se fait directement dans les classes controller afin de simplifier le code, réduire le nombre de fichiers et de réduire le temps de développement.

- **Diagramme des cas d'utilisation**



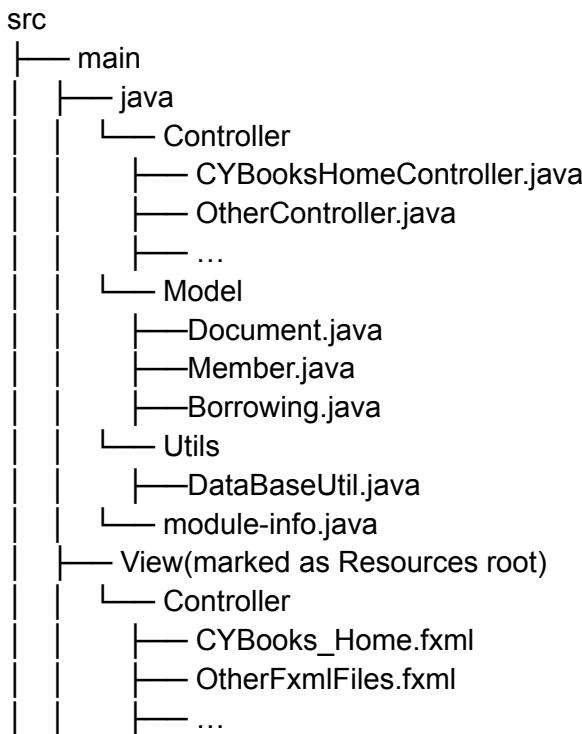
- **Modèle MVC (“Model-View-Controller”)**

Nous avons décidé de structurer notre projet en suivant les normes du modèle MVC, en faisant 3 packages : Model, View et Controller.

- ❖ Le package “Model” contient les entités Document, Member et Borrowing
- ❖ Le package “View” contient les interfaces utilisateur pour la gestion de la bibliothèque du point de vue du bibliothécaire. On y retrouve les vues fxml de

- chacune de nos pages de l'application (page de gestion des emprunts, gestion des membres, recherche d'un livre, et page statistiques)
- ❖ Le package “Controller” gère les actions de l'utilisateur comme la recherche de livres, l'emprunt ou le retour d'emprunt, l'ajout et la suppression de membres. Il interagit avec le modèle pour effectuer ces actions et met à jour les vues en fonction des résultats des actions du modèle.

Voici l'architecture de nos fichiers



## 2) Manuel Technique du Projet CY BOOKS

### a) Technologies et outils.

Coder le JAVA sur IntelliJ : Nous avons choisi cela car c'est l'outil avec lequel nous avons découvert JAVA et car cet IDE (Environnement de développement intégré) est très facile d'utilisation, possédant beaucoup de raccourcis facilitant énormément le développement.

Projet en JavaFX : Sur IntelliJ, nous avons la possibilité d'indiquer que le projet que nous faisons est un projet JavaFX. C'est donc ce que nous avons fait.

Faire un dépôt Git lié à IntelliJ : Le rendu devra se faire via un lien git, il est donc obligatoire de travailler dessus. Il faut de plus, se familiariser avec les outils de gestion de versions, qui servent à coder à plusieurs. L'utilisation de Git facilite également le suivi des modifications, la collaboration et la gestion des versions du code source.

Pour finir, Git se configure très bien avec la plupart des IDE, comme avec IntelliJ, et permet à chacun d'avoir sa branche où travailler puis de centraliser son travail dans une branche commune.

Base de données sur MySQL WorkBench: Nous avons déjà eu une expérience il y a peu de temps (projet développement Web) sur cet outil de gestion de base de données. MySQL Workbench est ergonomique, intuitif, et déjà installé sur nos ordinateurs, ce qui facilite son utilisation. L'application fournit un localhost auquel nous nous connectons via notre classe DataBaseUtil.java pour interagir avec la base de données. (cf annexes pour le script de notre base de données)

Le front de l'application avec SceneBuilder: Conformément aux directives reçues de nos professeurs, JavaFX doit être utilisé pour développer l'interface utilisateur (UI). SceneBuilder est un outil qui permet de concevoir des interfaces JavaFX de manière visuelle, ce qui simplifie le développement des interfaces graphiques. Nous avons déjà une expérience avec SceneBuilder et son intégration avec JavaFX rend son utilisation logique et naturelle pour ce projet.

Jira pour la gestion du projet: Jira permet de créer, assigner et suivre l'avancement des tâches. Jira fonctionne en créant des tickets pour chaque tâche ou fonctionnalité à réaliser, que nous pouvons ensuite organiser en sprints (périodes de travail définies). Chaque membre de l'équipe peut mettre à jour l'état de ses tickets, ce qui offre une visibilité claire sur le progrès de chaque tâche et facilite la collaboration.

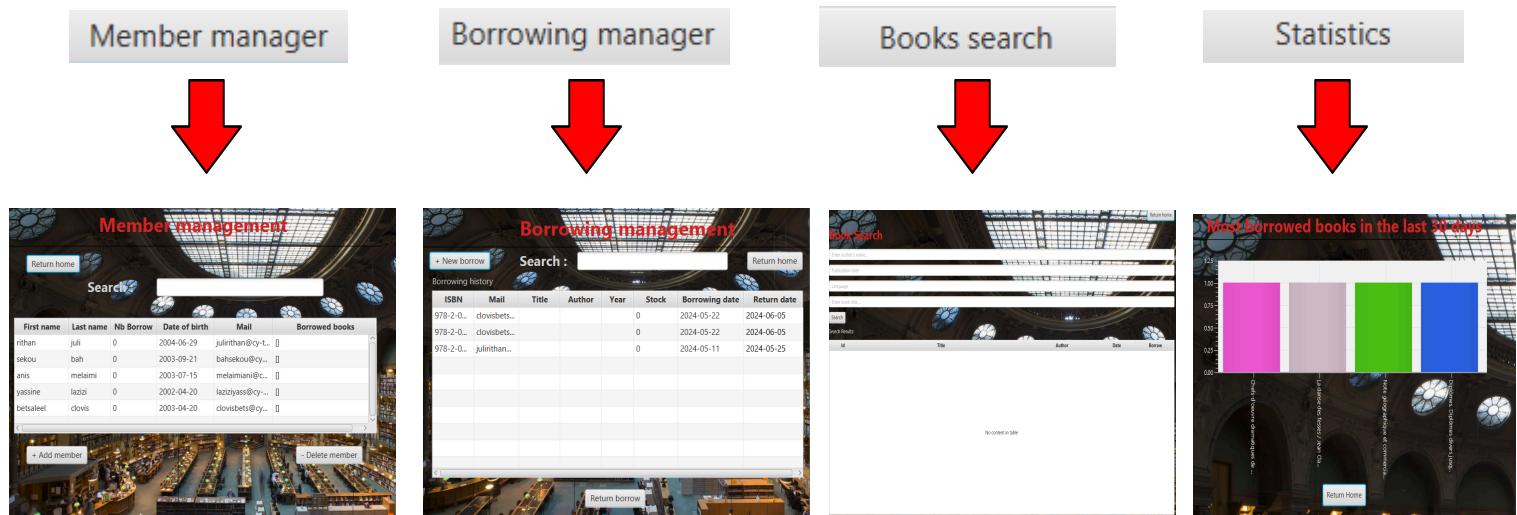
StarUML pour modéliser les diagrammes: C'est le logiciel qu'on a utilisé pour modéliser nos diagrammes pendant le semestre, il est très simple d'utilisation et répond parfaitement à nos besoins, de plus nous l'avons tous déjà sur nos ordinateurs.

## b) Fonctionnement

Voici la page d'accueil, elle s'ouvre dès qu'on lance l'application. Depuis celle-ci on peut accéder à quatre pages : la page de gestion des membres, la page de gestion des emprunts, la page de recherche de livres et la page des statistiques.



Figure 1 - Page d'accueil de l'application



Voici la page de gestion des membres, elle s'ouvre lorsqu'on clique sur « member manager » depuis la page d'accueil. Elle permet d'ajouter un membre en appuyant sur « Add member » et de supprimer un membre en appuyant sur « Delete member ». Grâce à la barre de recherche, on peut rechercher un membre plus facilement. On peut également retourner sur la page d'accueil grâce au bouton “Return Home”.

Member management					
First name		Last name	Nb Borrow	Date of birth	Mail
rithan	juli	0	2004-06-29	julirithan@cy-t...	<input type="button" value=""/>
sekou	bah	0	2003-09-21	bahsekou@cy...	<input type="button" value=""/>
anis	melaimi	0	2003-07-15	melaimani@c...	<input type="button" value=""/>
yassine	lazizi	0	2002-04-20	laziziyass@cy...	<input type="button" value=""/>
betsaleel	clovius	0	2003-04-20	cloviusbets@cy...	<input type="button" value=""/>

+ Add member      - Delete member

Figure 2 - Page d'accueil de gestion des membres

L'ajout d'un membre se fait grâce à son nom, son prénom, sa date de naissance et grâce à son mail avec une vérification sur le format du mail. La suppression d'un membre se fait en remplissant le mail et le nom de famille du membre que l'on souhaite supprimer avec vérification de l'existence du membre et de la correspondance entre le mail et le nom.

New member

Last name :

First name :

Birth date :

Mail :

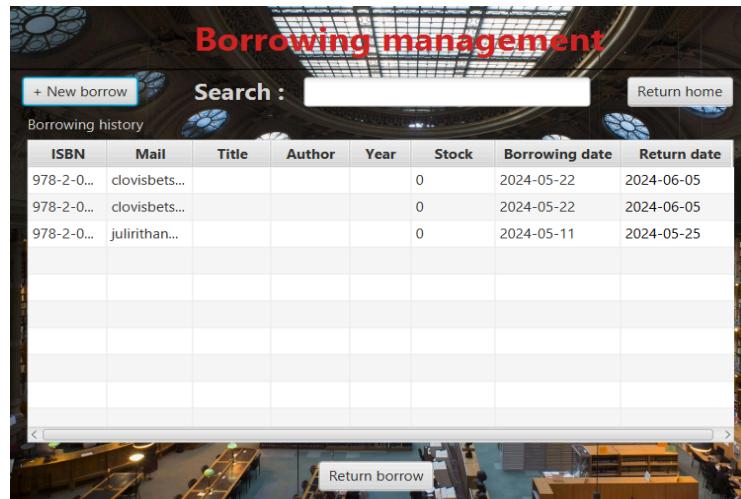
Delete member

Mail :

Last Name :

Figure 3 - Inscription & Suppression de membre

Voici la page de la gestion des emprunts, elle peut s'ouvrir depuis la page d'accueil. Elle permet de voir les différents livres qui sont empruntés actuellement avec la date d'emprunt et la date du retour. Il y a la possibilité d'ajouter un emprunt en appuyant sur "New Borrow" et la possibilité de rendre un livre en appuyant sur "Return Borrow". On peut également retourner sur la page d'accueil grâce au bouton "Return Home".



*Figure 4 - Gestion des emprunts*

Pour ajouter un livre, il faut le mail de l'adhérent et l'ISBN du livre avec des vérifications d'existences pour ceux-ci. La date d'emprunt est déjà pré-remplie à la date du jour sans qu'on puisse la modifier. Pour rendre un livre, il faut l'ISBN du livre et le mail de celui qui a emprunté le livre avec les vérifications nécessaires.

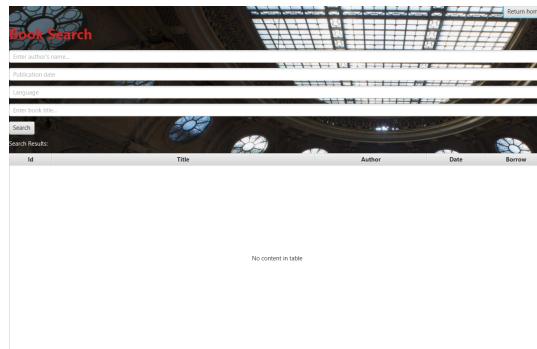
The image contains two side-by-side screenshots of a library application interface. Both screenshots show a large, ornate library hall with bookshelves in the background.

**Left Screenshot (New borrow):** A red arrow points down to this screen. It features a title "New borrow" at the top. Below it are three input fields: "Mail :" with a placeholder "[REDACTED]", "ISBN book :" with a placeholder "[REDACTED]", and "Borrowing date :" with a placeholder "2024-05-26". At the bottom are "Cancel" and "Save" buttons.

**Right Screenshot (Return borrowing):** A red arrow points down to this screen. It features a title "Return borrowing" at the top. Below it are two input fields: "ISBN :" with a placeholder "[REDACTED]" and "Mail :" with a placeholder "[REDACTED]". At the bottom are "Cancel" and "Return Borrow" buttons.

*Figure 5 - Nouvel emprunt & Retour d'un emprunt*

Voici la page de recherche des livres accessible depuis la page d'accueil. Celle-ci permet de rechercher le livre que l'on souhaite en fonction du titre, de l'auteur, de la langue et de la date de parution de celui-ci. On peut également retourner sur la page d'accueil grâce au bouton "Return Home".



*Figure 6 - Recherche d'un livre*

Le résultat de la recherche affiche l'ISBN de chaque livre, son titre, son auteur et sa date. Pour chaque livre, on a la possibilité d'appuyer sur le bouton "Borrow" afin de l'emprunter.

A screenshot of the same book search application. A large red arrow points downwards from the search bar to the results table. The search bar contains the name 'Voltaire'. The results table lists several books by Voltaire, including 'Cinquième partie' (1730), 'La Pucelle poème par Voltaire' (1730), 'Chapitre XXI. Addition à Candide. Ce qui arrive en France à Candide et à Martin' (1731), 'Commentaire sur l'Esprit des lois de Montesquieu. Par M. de Voltaire' (1738), 'Fragments sur l'Inde, sur le général Lally et sur le contre de Mysore' (1773), 'Relation de la mort du chevalier de La Barre, par Monsieur Casse\*\*\*, avocat au Conseil du Roi, à M...' (1798), 'Ravenhercule, secrétaire à un homme charmé' (1798), 'Lettres secrètes de Mr. de Voltaire. Publiées par Mr. L. B.' (1798), 'Épître à Henri Quatre, sur l'avènement de Louis XVI. Par M. De V.' (1798), 'Lettres Archées de Londres sur les Anglais et autres sujets, par M. de Voltaire' (1799), 'Théâtre choisi de Voltaire (Edition critique, précédée d'une notice littéraire)' (1870), and 'La Henriade, poème sur Voltaire, suivi de l'École sur la croise-Antoine. Nouvelle édition revue et corrigée' (1815). Each row in the table includes an 'Author' column (all listed as 'Voltaire (1694-1778)') and a 'Borrow' button.

*Figure 7 - Résultats de la recherche*

Lorsque l'on veut emprunter un livre depuis la page de recherche en cliquant sur "Borrow", l'ISBN se pré-remplit dans la page "New borrow" afin de faciliter l'emprunt du livre.

A screenshot of a web application interface titled "New borrow". The background features a photograph of a grand hall with arched windows and stained glass. The form contains three input fields: "Mail :" with an empty input field, "ISBN book :" with the value "bpt6k54612100", and "Borrowing date :" with the value "2024-05-25". Below the form are two buttons: "Cancel" and "Save". A large red arrow points downwards from the "Borrow" button at the top of the page to the "New borrow" form.

Figure 8 - Nouvel emprunt avec l'ISBN du livre prérempli

---

Voici la page "Statistiques" accessible depuis la page d'accueil. Cette page montre les dix livres les plus empruntés pendant les trente derniers jours avec le nombre d'emprunts et le titre de chacun des livres. On peut également retourner sur la page d'accueil grâce au bouton "Return Home".

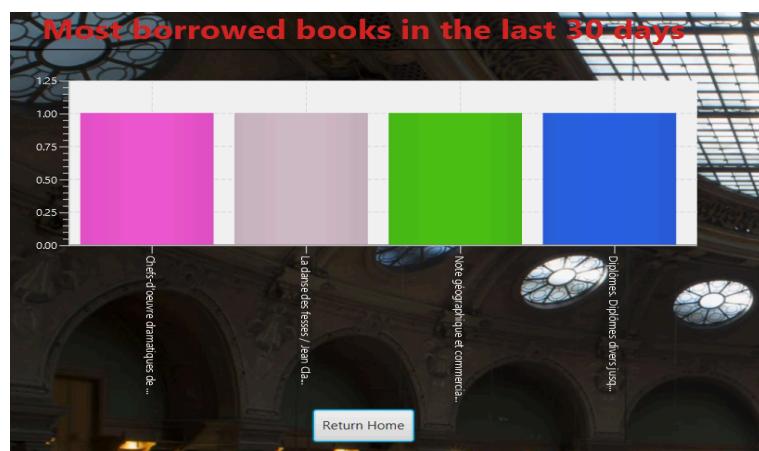


Figure 9 - Statistiques

## c) Aspect technique

- **Page d'accueil :**
  - On utilise le contrôleur CYBooksHomeController pour définir les actions des boutons présents sur cette page à travers des méthodes utilisant une méthode « loadView » qui permet d'afficher une page
  - On accède à cette page en exécutant la fonction main de la classe CYBooksHome
- **Page de gestion des membres :**
  - On utilise le contrôleur CYBooksMemberController pour :
    - Définir les actions des boutons présents sur cette page à travers des méthodes utilisant une méthode « loadView » qui permet d'afficher une page
    - Initialiser le tableau dans lequel il y a les membres avec leurs informations grâce à la méthode « initialize »
    - Remplir le tableau des membres grâce à la table “users” dans la base de donnée Library en utilisant la méthode « loadMemberData »
  - On utilise la classe model Member afin d'instancier les membres présents dans la base de données.
- **Page nouveau membre :**
  - On utilise le contrôleur CYBooksNewMemberController pour :
    - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
    - Vérifier qu'on ait aucun champ vide avec la méthode “isEmpty”
    - Conditionner l'absence de caractères spéciaux sur le nom et le prénom grâce à la méthode “addNameValidation”
    - Conditionner l'unicité et le format de l'adresse mail avec respectivement les méthodes “isMailUnique” et “isValidMail”
    - Conditionner la date de naissance pour qu'il faille avoir au moins dix ans pour être membre
    - Initialiser les champs avec les conditions grâce à la méthode « initialize »
    - Enregistrer les membres dans la table “users” de la BDD avec la méthode “SaveNewMember”, s'occupant de vérifier si le nombre d'emprunts est bien inférieur à la limite.

- **Page supprimer membre :**
  - On utilise le contrôleur CYBooksDeleteMember pour :
    - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
    - Vérifier que le mail et le nom de famille saisis sont en accord avec ce qu'on a dans la BDD, pour pouvoir le supprimer. Cela se fait avec la méthode “DeleteMember”.
- **Page de gestion des emprunts :**
  - On utilise le contrôleur CYBooksBorrowingController pour :
    - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
    - Initialiser le tableau dans lequel il y a les emprunts avec leurs informations grâce à la méthode “initialize”
    - Remplir le tableau des emprunts grâce à la table books dans la base de données Library ainsi qu'à l'API pour récupérer le titre, l'auteur et la date du livre en fonction de l'ISBN. Pour cela on utilise la méthode “loadBorrowingData”
    - Rediriger vers les pages permettant de faire un nouvel emprunt ou de retourner un emprunt.
  - On utilise la classe model Borrowing afin d'instancier les emprunts présents dans la base de données.
- **Page nouvel emprunt :**
  - On utilise le contrôleur CYBooksNewBorrowingController pour :
    - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
    - Vérifier l'existence et l'association entre l'ISBN et le mail avec respectivement les méthodes “checkIdExists” et “checkMemberExists”
    - Voir si le livre n'est pas déjà emprunté avec la méthode “checkBookNotBorrowed”
    - Conditionner le nombre d'emprunt du membre pour pouvoir emprunter avec la méthode “canUserBorrowMoreBooks”
    - Pré-remplir l'ISBN si on accède à cette page depuis la page de recherche de livres en ayant appuyer sur “Borrow” grâce à la méthode “loadNewBorrowingView” présente dans CYBooksSearchController
    - Emprunter un livre avec La méthode “saveNewBorrowing”
- **Page retour emprunt :**

- On utilise le contrôleur CYBooksReturnBorrowingController pour :
  - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
  - Vérifier l'existence de L'ISBN et du mail saisi en vérifiant qu'il y ait bien un résultat avec la requête SQL
  - Vérifier de l'existence de l'association entre l'ISBN et le mail en comparant l'ISBN saisi et l'ISBN associé au livre emprunté
  - La méthode “deleteBorrowing” permet de rendre un livre

- **Page de recherche de livre:**

- On utilise le contrôleur CYBooksSearchController pour :
  - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
  - Initialiser le tableau dans lequel il y a les livres avec leurs informations grâce à la méthode « initialize »
  - Lancer une recherche en apparaissant à la page une et sans les données des précédentes recherches grâce à la méthode “handleSearch”
  - Afficher plus de résultat grâce à la méthode “loadMoreBooks”
  - Rechercher des livres par API grâce à une URL en fonction des informations mises dans les champs auteur, titre, langue et date par la méthode “searchBooks”
- On utilise la classe model Document afin d'instancier les livres recherchés par l'API Gallica de la BNF.

- **Page des statistiques:**

- On utilise le contrôleur CYBooksStatisticsController pour :
  - Définir les actions des boutons présents sur cette page à travers des méthodes appelant « loadView » pour afficher une page
  - Initialiser le tableau dans lequel il y a un diagramme en bâton contenant les livres les plus empruntés des 30 derniers jours grâce à la méthode « initialize ». En effet, cette méthode exécute une requête SQL pour obtenir les dix livres les plus empruntés au cours des 30 derniers jours. Puis pour chaque livre, la méthode récupère le nombre de fois où lisbn apparaît dans la table historic et effectue une requête API externe pour obtenir le titre du livre via l'API de Gallica.
  - Tronquer les titres pour plus de visibilité à l'aide de la méthode « truncateTitle».
  - Appliquer des couleurs aléatoires et des étiquettes personnalisées aux barres du diagramme à l'aide des méthodes «applyBarColorsAndLabels» et «generateRandomColor».

#### d) Limitations fonctionnelles de la plateforme.

- On suppose qu'il y a un exemplaire pour chaque livre.
- Le diagramme de classe initial a été modifié en cours de projet pour mieux répondre aux besoins, notamment pour la récupération des données de l'API.

## IV. Partie travail et gestion de projet

### 1) Organisation du travail dans le groupe

Le projet a débuté par une réunion avec notre encadrante, ce qui a lancé l'organisation initiale du projet. Nous avons tout d'abord commencé par configurer notre environnement de développement en mettant en place un dépôt Git lié à IntelliJ et en nous familiarisant avec Jira pour le planning.

Notre encadrante nous a initié à l'utilisation de Jira, un outil de gestion de projet qui permet de créer, assigner et suivre l'avancement des tâches. Jira fonctionne en créant des tickets pour chaque tâche ou fonctionnalité à réaliser, que nous pouvons ensuite organiser en sprints (périodes de travail définies, un sprint pour 1 semaine durant ce projet). Chaque membre de l'équipe peut mettre à jour l'état de ses tickets, ce qui offre une visibilité claire sur le progrès de chaque tâche et facilite la collaboration.

Ensuite, nous avons établi un planning en listant toutes les tâches que nous aurons à réaliser au cours du projet, afin de les reporter sur Jira. Nous avons également réalisé un premier diagramme de classes et un diagramme de cas d'utilisation pour mieux comprendre et structurer notre application. Puis nous avons modélisé l'interface graphique de notre application (organisation des pages, emplacements des boutons)

Pour assurer une communication continue et efficace, nous avions quotidiennement des réunions internes appelées "daily" où les membres de l'équipe se retrouvaient pour discuter de leurs progrès, des obstacles rencontrés et des prochaines étapes. De plus, nous avions des réunions hebdomadaires ("weekly") avec notre encadrante. Pendant ces sessions, nous présentions nos avancées, et elle nous fournissait des conseils ainsi que des points d'amélioration pour optimiser notre travail et corriger d'éventuels problèmes.

Ces pratiques nous ont permis de maintenir une bonne organisation, de respecter notre planning, et de progresser efficacement tout au long du projet.

- Rôles et répartition en groupes:

Scrum Master : Sekou

Workers : Anis, Betsaleel, Rithan, Yassine et Sékou

Nous nous sommes répartis en 2 groupes : API and SQL team & UI SceneBuilder team

**API and SQL team :** Sékou, Anis et Betsaleel (nous nous sommes regroupés à trois car le concept d'API est nouveau pour nous, ce qui explique pourquoi cette équipe compte plus de membres).

**UI SceneBuilder team :** Rithan et Yassine

**API and SQL team :**

- Page de recherche de livres :
  - Utilisation de l'API Gallica pour rechercher des livres dans la page de recherche de livres en fonction de l'auteur, de la langue, du titre et de la date grâce à une requête faite par URL
  - Affichage des informations (auteur, titre, date, isbn) dans un tableview et possibilité de l'emprunter en cliquant sur le bouton borrow associé qui ramène à la page nouvel emprunt
  - Redirection vers page d'accueil via bouton
- Page nouvel emprunt :
  - Préremplissage de l'ISBN si l'on a cliqué sur le bouton borrow depuis la page de recherche
  - Vérification de l'existence de l'ISBN dans l'API
- Page gestion des emprunts:
  - Obtention du titre, de l'auteur et de la date grâce à lisbn obtenu par la base de donnée en faisant une requête URL grâce à l'API
- Création de la classe DataBaseUtil pour se connecter à la base de données via LocalHost.
- Création de la base de données "Library"
- Statistiques sur les 10 livres les plus empruntés au cours des 30 derniers jours. Pour cela, nous comptons le nombre de fois où chaque ISBN apparaît sur la table **historic** sur cette période. Nous utilisons la commande DATE\_SUB(CURDATE(), INTERVAL 30 DAY) pour filtrer les enregistrements dont la date d'emprunt est dans les 30 derniers jours. Enfin, nous limitons les résultats à 10 avec LIMIT 10.
- Obtention et affichage du titre, grâce à lisbn obtenu par la base de donnée en faisant une requête URL grâce à l'API sur la page des statistiques

## UI SceneBuilder team :

- Création des pages FXML, boutons pour la navigation entre les pages et fonctionnalités et d'un controller pour chaque page
- Page d'accueil : Boutons pour accéder aux pages suivantes : Gestion des emprunts, Gestion des membres, Chercher un livre et Accéder aux statistiques
- Page gestion des emprunts :
  - Affichage des emprunts par récupération des informations via une jointure sur l'id (cf SQL en annexe) dans un tableView avec les colonnes suivantes : ISBN, mail, titre, auteur, année, stock, date d'emprunt, date de retour
  - Redirections vers page d'accueil, page pour nouvel emprunt et pour récupération d'emprunt
  - Affichage de la date de retour en rouge si cette date est dépassée
- Page nouvel emprunt :
  - Méthode pour enregistrer un nouvel emprunt avec la date du jour préremplie et non modifiable dans le champ date (bouton save) : Ajout dans la BDD
  - Vérification du format du mail de l'utilisateur et de l'existence de l'utilisateur,
  - Bouton pour annuler et revenir à la page précédente (page gestion des emprunts)
- Page retour emprunt :
  - Méthode pour enregistrer la récupération d'un livre emprunté avec vérification de l'ISBN et du mail (bouton return borrowing) : Suppression de la BDD
  - Bouton pour annuler et revenir à la page précédente (page gestion des emprunts)
  - Vérification de la correspondance entre isbn et mail de l'utilisateur (et existence dans la table books)
  - Bouton pour annuler et revenir à la page précédente (page gestion des emprunts)
- Page gestion des membres:
  - Affichage des membres par récupération des informations de la table users dans un tableView avec les colonnes suivantes : nom, prénom, nombre d'emprunts, date de naissance, mail, liste des isbn des livres empruntés
  - Redirections vers page d'accueil, page pour nouveau membre et pour supprimer un membre
- Page nouveau membre:
  - Méthode pour enregistrer un nouveau membre (bouton save) : Ajout dans la BDD
  - Vérification du nom et prénom (première lettre en majuscule le reste en minuscule et pas de chiffres ni caractères spéciaux)
  - Vérification du mail
  - Date picker pour le choix de la date de naissance + le membre doit avoir au minimum dix ans (le calendrier est initialisé à 10 ans plus tôt)
  - Bouton pour annuler et revenir à la page précédente (page gestion des emprunts)
- Page supprimer membre :
  - Méthode pour supprimer un membre (bouton delete) : suppression de la BDD

- Vérification de la correspondance entre nom et mail de l'utilisateur (et existence dans la table users)
- Filtrage des information sur la page gestion des emprunts et gestion des membres en affichant seulement les lignes contenant le mot clé tapé dans le textfield

## V. Conclusion

Pour conclure, nous avons réussi à développer une application de gestion de bibliothèque, CYBooks, qui est à la fois fonctionnelle et optimisée. Cette application répond aux principaux besoins énoncés dans le cahier des charges grâce à ses différentes fonctionnalités, comme la gestion des emprunts, des retours, des membres et la consultation des livres de la bibliothèque.

Ce projet a impliqué le développement de plusieurs modules en Java, ainsi que la création et la gestion de la base de données avec MySQL. Nous avons également utilisé JavaFX pour concevoir une interface utilisateur interactive et intuitive, facilitant ainsi l'utilisation de l'application.

Au cours de ce projet, nous avons acquis une compréhension approfondie de l'interaction entre la programmation orientée objet en Java, la gestion de base de données en SQL et l'utilisation des API. Le projet met l'accent sur l'importance de modéliser les interfaces d'une application pour structurer notre code et sur l'utilisation de JavaFX pour créer une interface utilisateur fluide et ergonomique.

Grâce à ce projet, nous avons non seulement mis en pratique nos compétences en programmation Java et en conception de base de données, mais nous avons aussi appris à travailler efficacement en équipe en utilisant des outils de gestion de projet comme Jira et Git. Cette expérience nous a permis de mieux comprendre les processus de développement logiciel et de nous préparer pour les défis futurs dans notre cursus.

## IV. Annexe

### 1) Script SQL

```
-- Creation of a database
```

```
CREATE DATABASE IF NOT EXISTS Library;
```

```
USE Library;
```

```
-- Creation of a table `users`
```

```
CREATE TABLE IF NOT EXISTS users (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    firstname VARCHAR(255) NOT NULL,
```

```
    lastname VARCHAR(255) NOT NULL,
```

```
    member_in_good_standing BOOLEAN DEFAULT TRUE,
```

```
    email VARCHAR(255) UNIQUE NOT NULL,
```

```
    number_borrowing INT DEFAULT 0,
```

```
        CHECK (number_borrowing BETWEEN 0 AND 3),
```

```
    birth_date DATE NOT NULL
```

```
);
```

```
-- Création de la table `livres_empruntes`
```

```
CREATE TABLE IF NOT EXISTS books (
```

```
    isbn VARCHAR(255) NOT NULL PRIMARY KEY,
```

```
    user_id INT NOT NULL,
```

```

loan_date DATE NOT NULL,
return_date DATE NOT NULL,
quantity_available INT,
total_quantity INT DEFAULT 1,
FOREIGN KEY (user_id) REFERENCES users(id)
);

```

-- Insertion of members in table users

```

INSERT INTO users (id,firstname, lastname, member_in_good_standing, email, birth_date,
number_borrowing) VALUES
(1,'rithan', 'juli', TRUE,'julirithan@cy-tech.fr','2004-06-29',0),
(2,'sekou', 'bah', TRUE,'bahsekou@cy-tech.fr','2003-09-21',0),
(3,'anis', 'melaimi', TRUE,'melaimiani@cy-tech.fr','2003-07-15',0),
(4,'yassine', 'lazizi', TRUE,'laziziyass@cy-tech.fr','2002-04-20',0),
(5,'betsaleel', 'clovis', TRUE,'clovisbets@cy-tech.fr','2003-04-20',0);

```

CREATE TABLE IF NOT EXISTS historic (

```

id INT AUTO_INCREMENT PRIMARY KEY,
isbn VARCHAR(255) NOT NULL,
loan_date DATE NOT NULL
);

```

