



Ing-1 GI-1

Lucas BÉDUÉ

Audrey TRUONG

Maxime DUBIN-MASSÉ

Baptiste MOISSERON

Mathias GALISSON

Rapport – CY-BOOKS

26 mai 2024

Projet Java

Mohammed Haddache

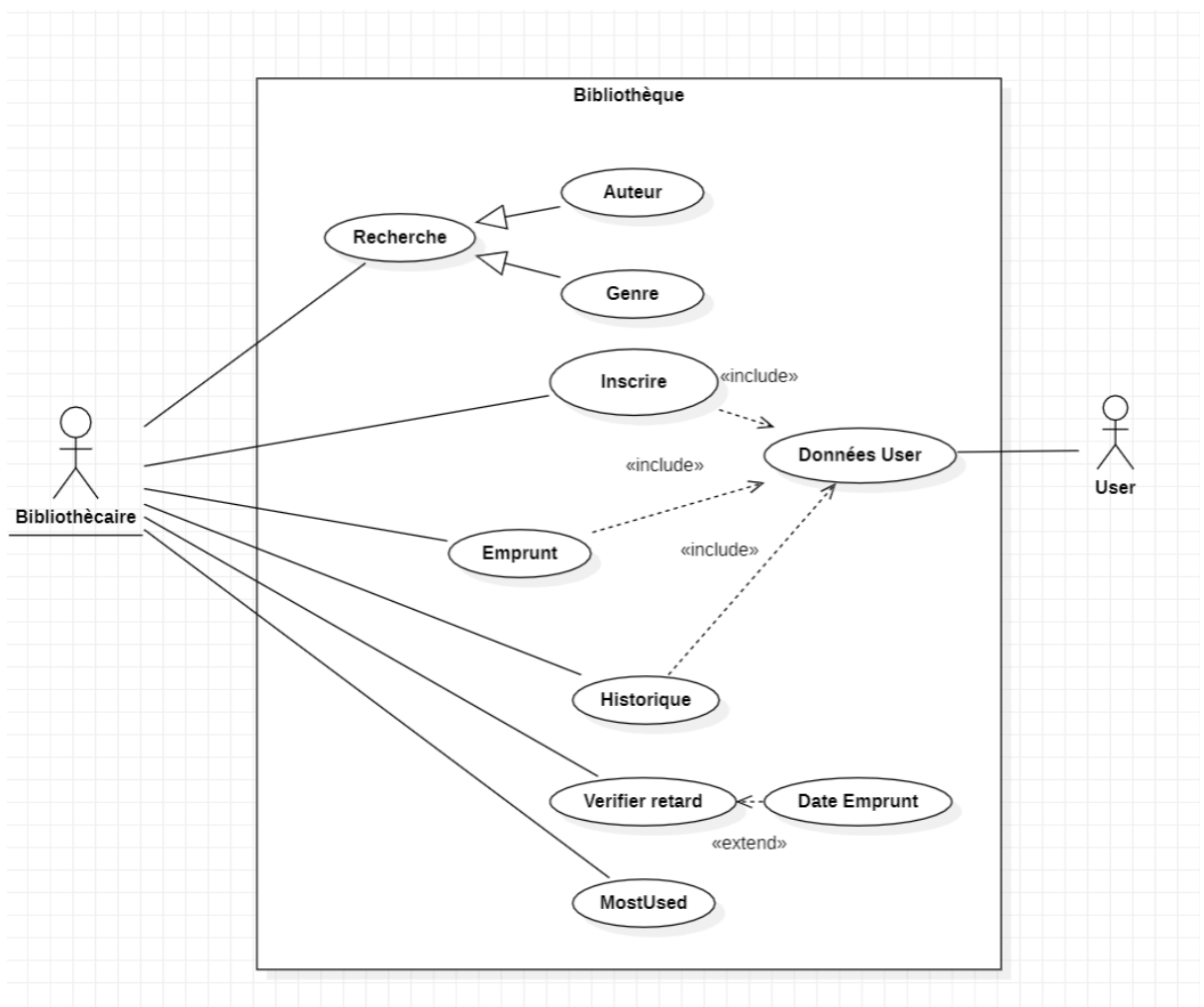
2023-2024

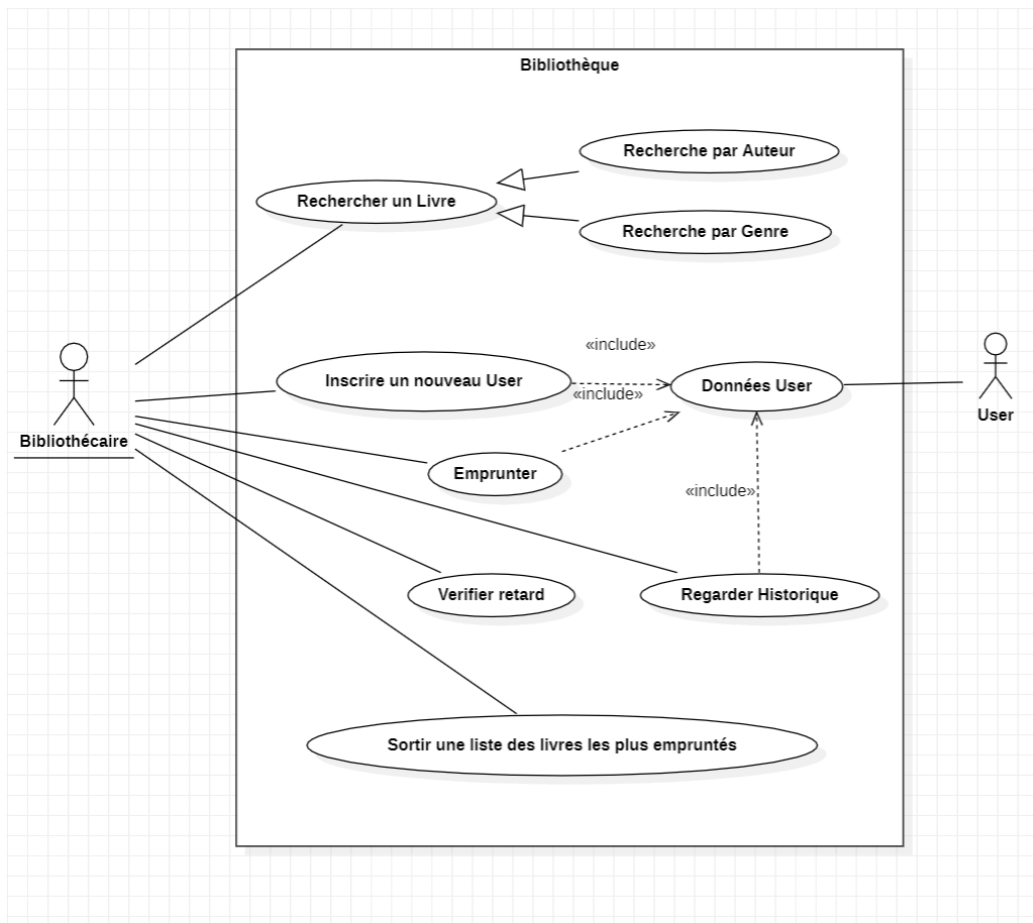
1. Introduction.....	3
2. Difficultés rencontrées.....	7
3. Fonctionnement de l'application.....	9
-Register an user:.....	9
-User list:.....	10
-Modify user:.....	10
-Check user history:.....	11
-Search user:.....	12
-Book search:.....	13
-Borrowed book:.....	14

1. Introduction

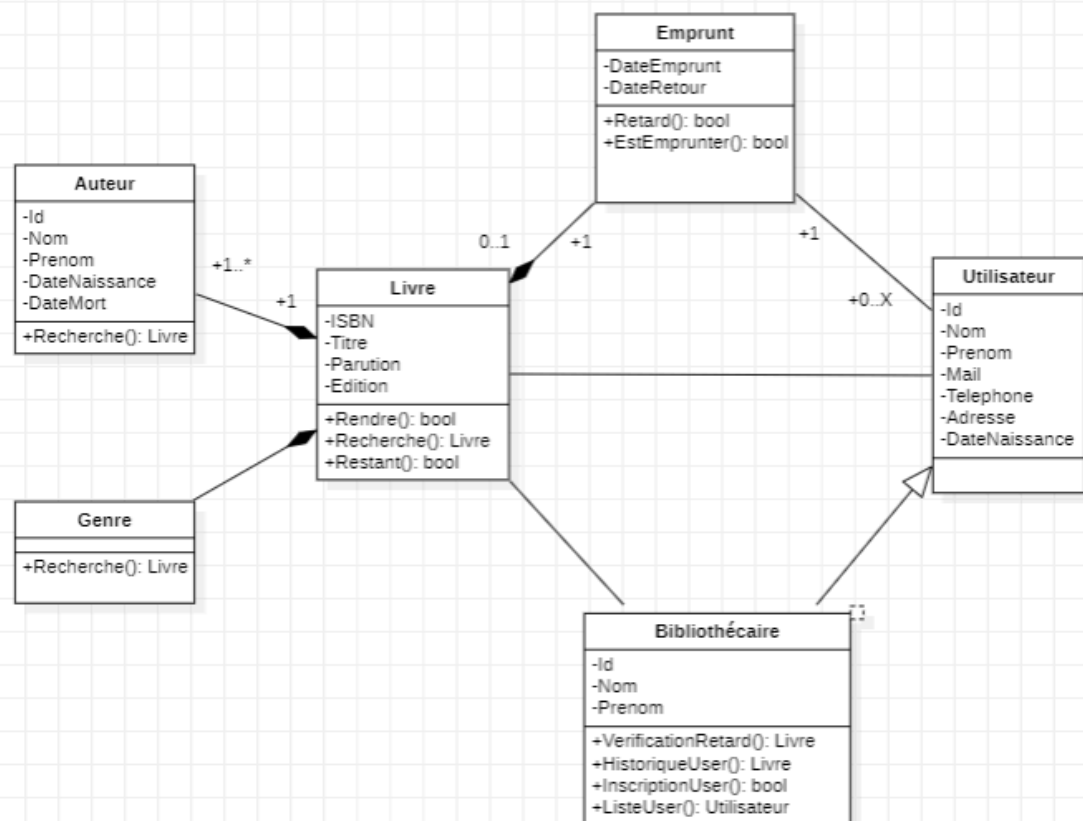
Le lundi 7 mai nous avons eu une réunion nous expliquant le déroulement des projets et une présentation de ceux-ci. Nous avons choisi d'un accord commun le projet CY-BOOKS car il était pour nous le projet le plus réalisable dans le temps donné et avec nos capacités actuelles. Suite à cela nous nous sommes réunis pour parler du projet et commencer à identifier les points à travailler.

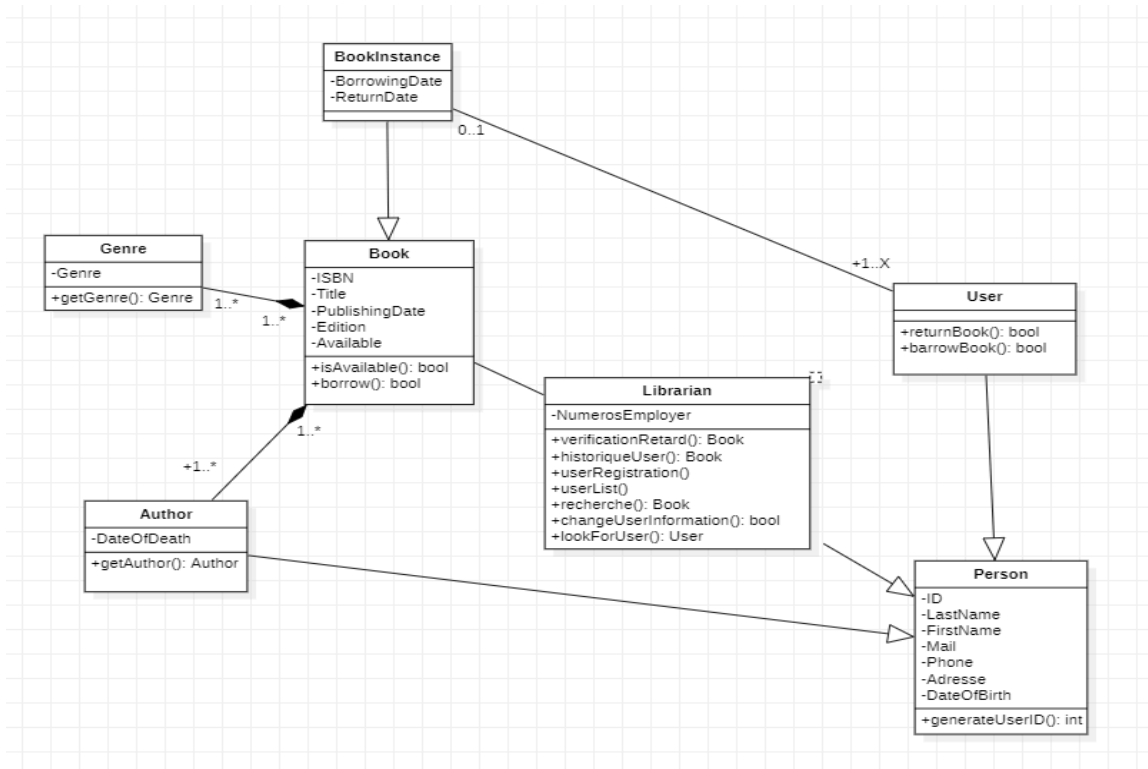
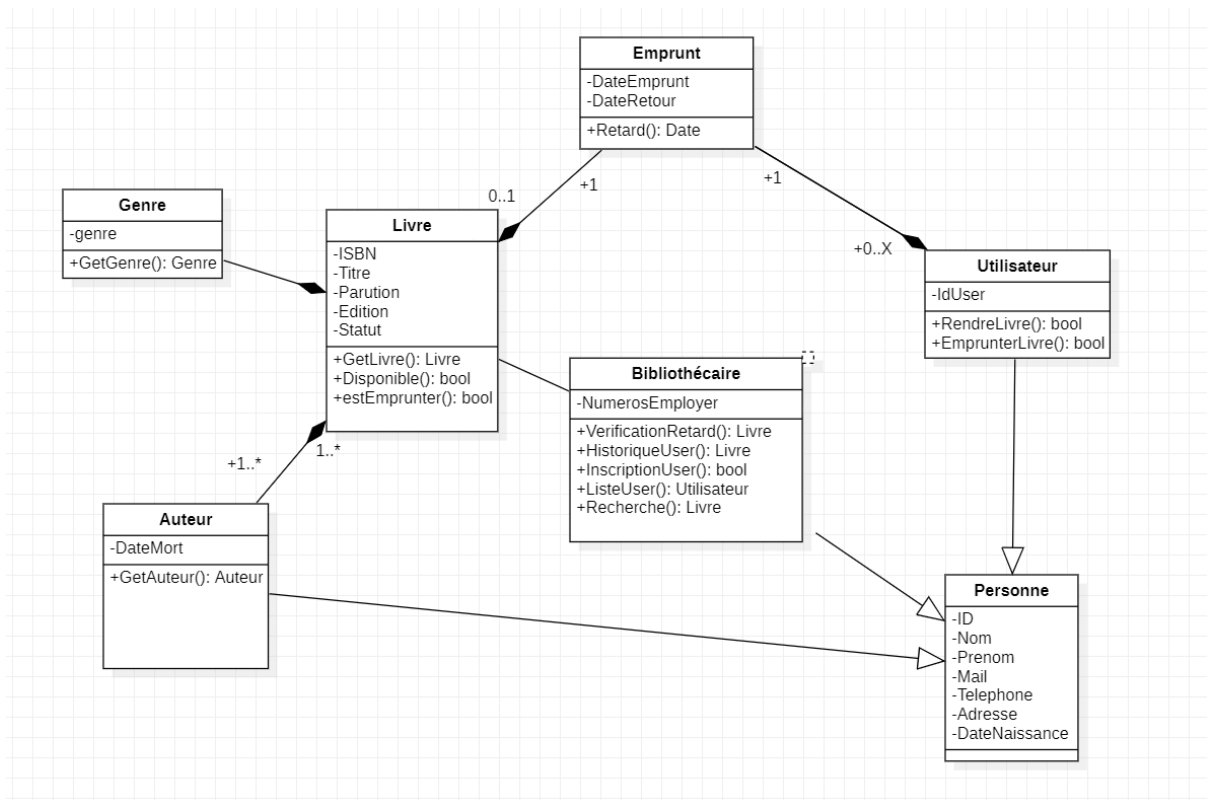
Nous nous sommes donc penchés sur la lecture du sujet pour extraire deux diagrammes : un diagramme de cas d'utilisation et un diagramme de classes ce qui nous facilitera le début de conception du projet. Pour le diagramme de cas d'utilisation nous avons identifié les acteurs comme étant le bibliothécaire et le user. Pour les actions nous avons utilisé celles demandées dans le sujet ce qui nous donne comme premier jet :





Vient ensuite le diagramme de classes qui s'inspire du diagramme de cas d'utilisation. Pour ce diagramme nous avons apporté plus de précision et décrit la structure globale qu'aura le projet. Ce diagramme nous a posé pas mal de soucis car plus nous avançons dans le projet plus nous nous rendions compte qu'il fallait apporter des modifications pour garder la cohérence de celui-ci. On a d'abord modifier les relations user, librarian et author en créant une généralisation pour simplifier et alléger les classes. Ensuite nous avons réfléchi à la meilleure façon de représenter la classe d'emprunt et comment la relier et implémenter avec une base de données. On a donc plusieurs étapes dans le développement du diagramme de classe :





2. Difficultés rencontrées

Au cours du projet des difficultés sont apparues et ont ralenti la progression du développement. La première a été comme dit plus haut la conception des diagrammes que nous avons dû adapter et modifier le long du projet. Ensuite chacun a rencontré des problèmes de son côté en fonction de sur quoi il travaillait.

Pour la base de données la plus problématique a été la gestion du pilote pour lier JAVA à MySQL car nous avons eu du mal à trouver un pilote fonctionnel et opérationnel. Beaucoup de pilotes trouvés n'étaient pas à jour et certains étaient obsolètes. Il fallait aussi trouver un pilote qui fonctionne sur linux et windows.

Pour la partie API, nous avons commencé par lire la documentation relative à l'API et effectué nos premiers tests sans utiliser Java. Ensuite, nous avons implémenté des recherches uniques, comme une recherche en fonction de l'auteur.

La première difficulté rencontrée concernait la recherche multiple avec plusieurs facteurs, en raison de l'utilisation de 'and' dans la requête. Pour surmonter cela, nous avons codé une fonction de recherche en lien avec request pour unifier toutes les recherches possibles. Cette fonction permet de sauvegarder toutes les informations concernant la recherche, puis search de request fait le lien entre l'instance Search et la commande devant être lancée pour effectuer la recherche dans l'API.

Nous avons ensuite décortiqué le fichier et créé des fonctions permettant de le décomposer pour obtenir les différentes informations recherchées pour la sélection de livres, et ce, sans bibliothèques externes, ce qui fut un processus long.

Enfin, la dernière difficulté résidait dans la transformation de ces données en instances des books dans la base de données. Nous avons dû gérer la classe d'emprunt, en sachant comment dissocier les livres et les associer correctement.

La gestion des différentes façons d'entrer un numéro de téléphone valide doit être prise en compte. Il faut pouvoir entrer des numéros avec +33, des points, des tirets, avec ou sans espaces entre les chiffres.

Du même acabit, la gestion des nom et prénom qui peuvent être composés de caractères spéciaux, les accents. Il a donc fallu prendre en compte ces contraintes, car cela nous a posé problème lors de la gestion de l'api, pour le nom et prénom des auteurs ce qui nous a bloqué à plusieurs reprises.

Pour ce qui est du côté XML, la première difficulté a été de trouver les éléments qui nous intéressaient dans le fichier. Lorsque cela fut fait, on s'est rendu compte que tous les champs n'étaient pas présents pour chaque livre, ce qui nous a forcés à prendre des mesures pour les champs qui n'existaient pas. De plus, même si les champs en question existaient, ils ne suivaient pas tous la même norme, ce qui nous a posé problème à plusieurs reprises, nous obligeant à faire des vérifications pour avoir seulement les données souhaitées.

Du côté du FXML, il y a eu 3 grandes barrières, la première et plus simple était le fait que nous avons dû apprendre à coder au fur et à mesure des nécessités. En cours, nous avons eu des exemples mais tous les codes étaient déjà donnés, la première semaine a donc été dédiée à décortiquer les notions et fonctions pour pouvoir commencer à créer notre interface visuelle.

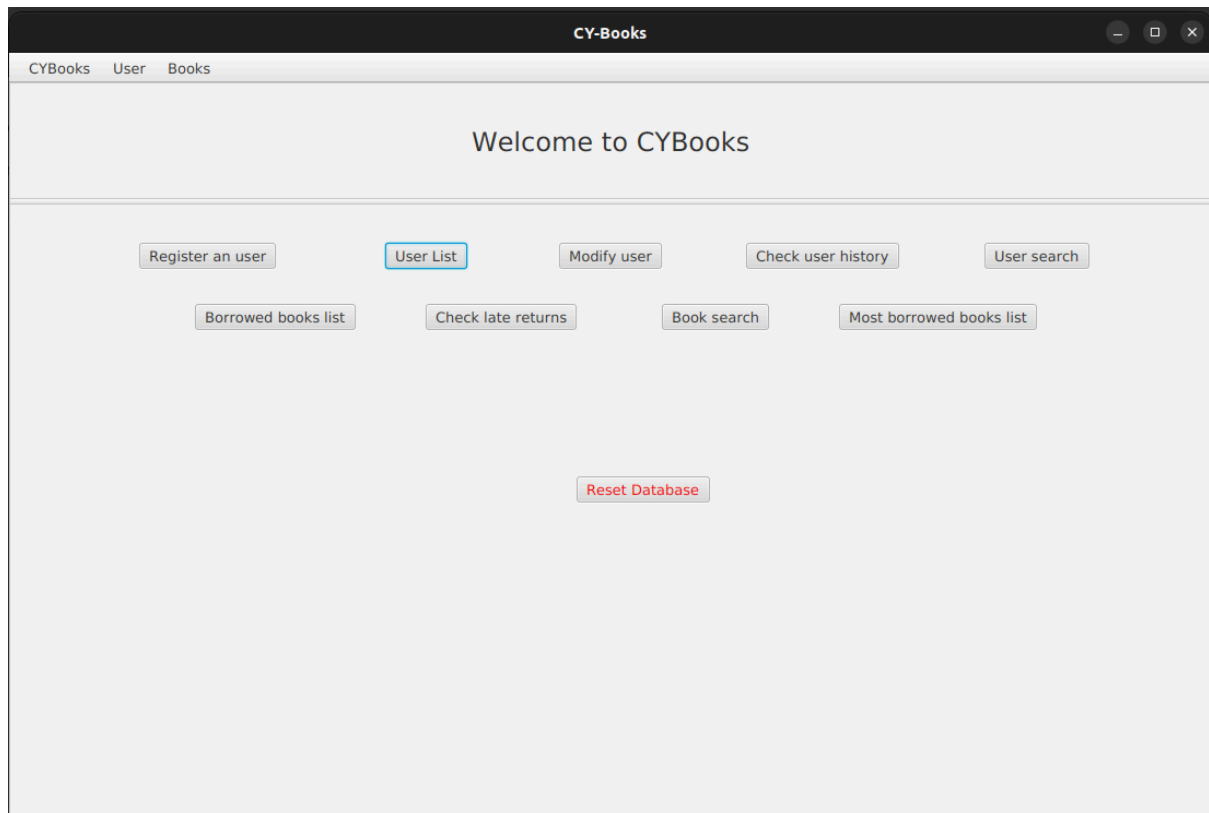
Après cela, il a fallu pouvoir amener les informations à l'écran et de nouveau des difficultés ont été rencontrées. Les informations ne s'affichaient pas ou de nombreuses exceptions telles que des `NullPointerException` ou des `InvocationTargetException` ont été rencontrées, notre première réaction fut d'essayer de mettre des blocs `try catch` mais les erreurs ont continué à apparaître. Il a fallu beaucoup de tentatives mais nous avons fini par trouver que le problème venait du fait que nous n'avions pas set le `fx:id` de la `TableView` au tableau d'où viendrait les informations que nous souhaitions afficher.

Enfin le dernier problème majeur fut de devoir pouvoir changer entre les pages. Chacune des pages devait avoir son propre controller pour les actions qui seraient effectuable dessus et notre interface avait pour base un `BorderPane` avec un `MenuBar` en haut qui possède les boutons pour changer entre toute les pages depuis n'importe qu'elle d'entre elle et au centre du `BorderPane` ete affiche le contenu de chaque page. Nous avons commencé par essayer de créer des objets `Scene` pour chaque page mais malheureusement les `BorderPane` prennent des objets `Node` dans leur différentes parties. Il a donc fallu créer toutes les pages en `Node` pour ensuite changer le centre du `BorderPane` principal mais cela n'a pas directement fonctionné. Le problème venait encore du fait qu'il fallait set le `fx:id` du `BorderPane` principal à celui que nous changions dans notre programme (nous changions de contenu en utilisant `rootLayout.setCenter(notrePage)` et `rootLayout` était notre `BorderPane` principal).

Ce projet nous a ainsi permit d'apprendre des bases du FXML même si le manque d'explications claires en cours nous a certainement forcé à apprendre de nous-mêmes à travers nos erreurs ce qui n'est pas une mauvaise chose en soi.

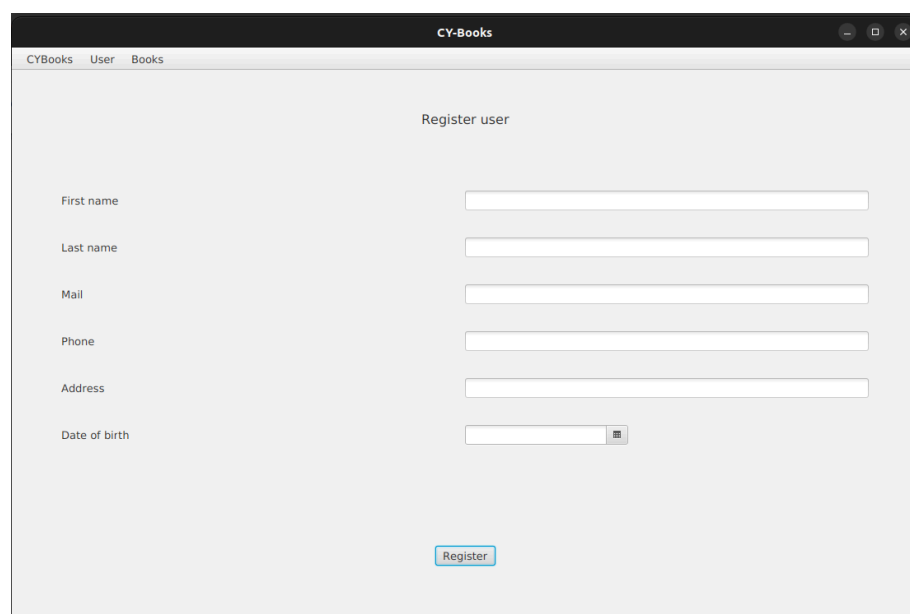
3. Fonctionnement de l'application

Lors du démarrage de notre application la fenêtre suivante s'affiche,



Sur celle-ci toutes les fonctions de notre application s'affichent et le libraire peut choisir celle qu'il souhaite.

-Register an user:

The screenshot shows the "Register user" form within the CY-Books application. The form is titled "Register user" and contains several input fields for user registration: "First name", "Last name", "Mail", "Phone", "Address", and "Date of birth". Each field is represented by a white text input box. The "Date of birth" field includes a small calendar icon. At the bottom of the form, there is a blue "Register" button.

Cependant si on cherche à ajouter quelqu'un d'autre mais que leur prénom, nom de famille, adresse et date de naissance sont les mêmes, l'application retournera que l'utilisateur existe déjà.

[illegible]

-Modify user:

CYBooks User Books

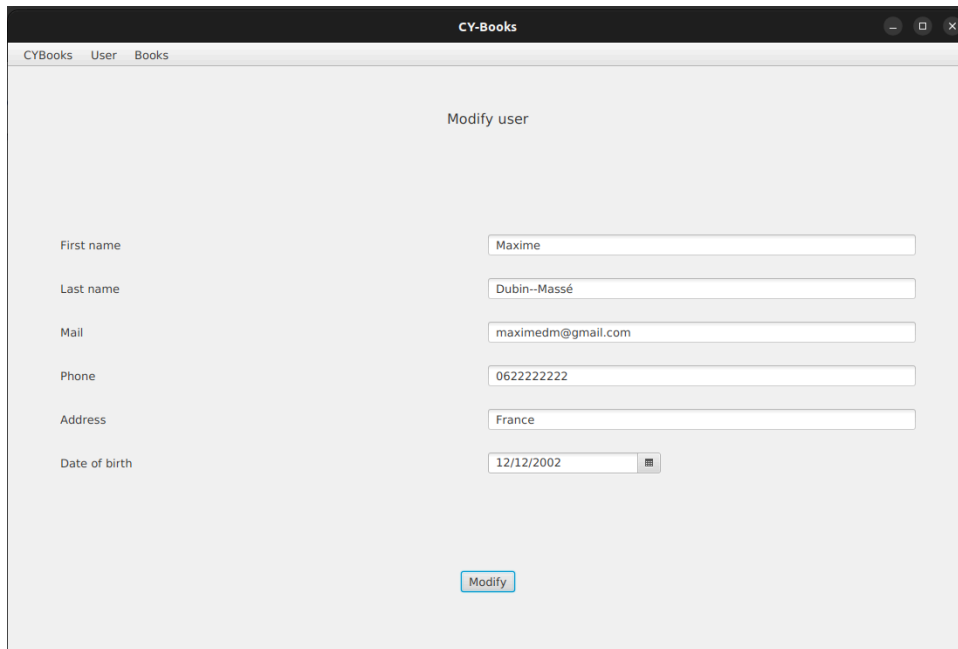
Search user to modify

ID

Search

Sur cette page, le libraire commence par saisir l'ID de l'utilisateur à modifier.
Si ce dernier n'existe pas, un message d'erreur s'affiche.

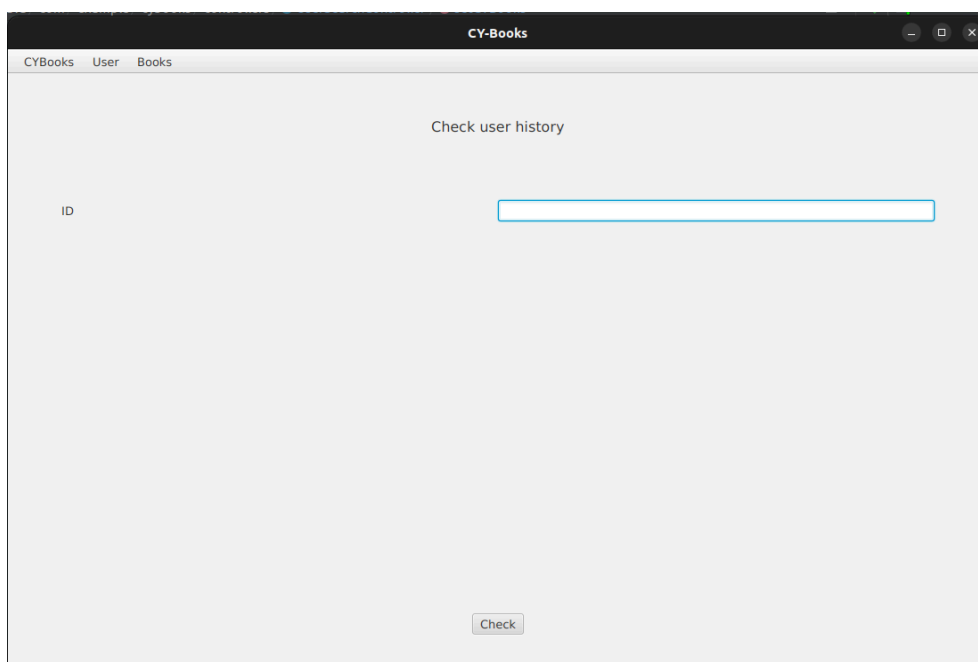
Sinon les informations actuelles de cet utilisateur seront affichées.



The screenshot shows a web browser window titled "CY-Books" with a navigation bar containing "CYBooks", "User", and "Books". The main content area is titled "Modify user". It contains a form with the following fields: "First name" (Maxime), "Last name" (Dubin--Massé), "Mail" (maximedm@gmail.com), "Phone" (0622222222), "Address" (France), and "Date of birth" (12/12/2002). A "Modify" button is located at the bottom of the form.

Le libraire peut ensuite modifier les champs qu'il souhaite, et les données seront mises à jour dans la base de données.

-Check user history:

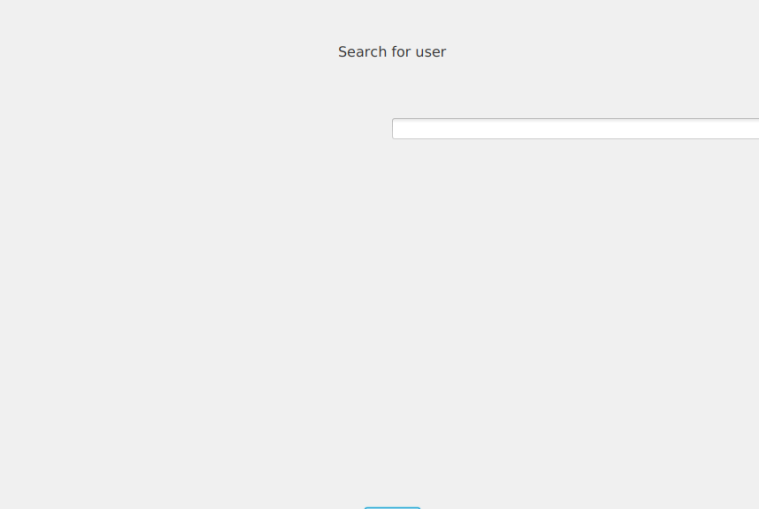


The screenshot shows a web browser window titled "CY-Books" with a navigation bar containing "CYBooks", "User", and "Books". The main content area is titled "Check user history". It contains a form with a single field labeled "ID". A "Check" button is located at the bottom of the form.

Cette page permet d'entrer l'ID d'un utilisateur pour afficher l'historique de ses emprunts.

[illegible]

-Search user:



The screenshot shows a web browser window titled "CY-Books". The browser's address bar contains the text "CYBooks User Books". The main content area of the browser is light gray and contains the text "Search for user" centered at the top. Below this text, on the left side, is the label "ID". To the right of the "ID" label is a long, empty white rectangular input field. At the bottom center of the page, there is a small blue button with the text "Search" in white.

Cette page permet d'entrer l'ID d'un utilisateur pour afficher plus d'informations sur cette personne ainsi que les livres qu'elle emprunte actuellement.

