CY-BOOKS

Projet réalisé par :

Marius BROSSET
Alexy ANDRE
Estelle ROUDET
Zakaria LESHOB
Liam MARTIN

Projet encadré par :

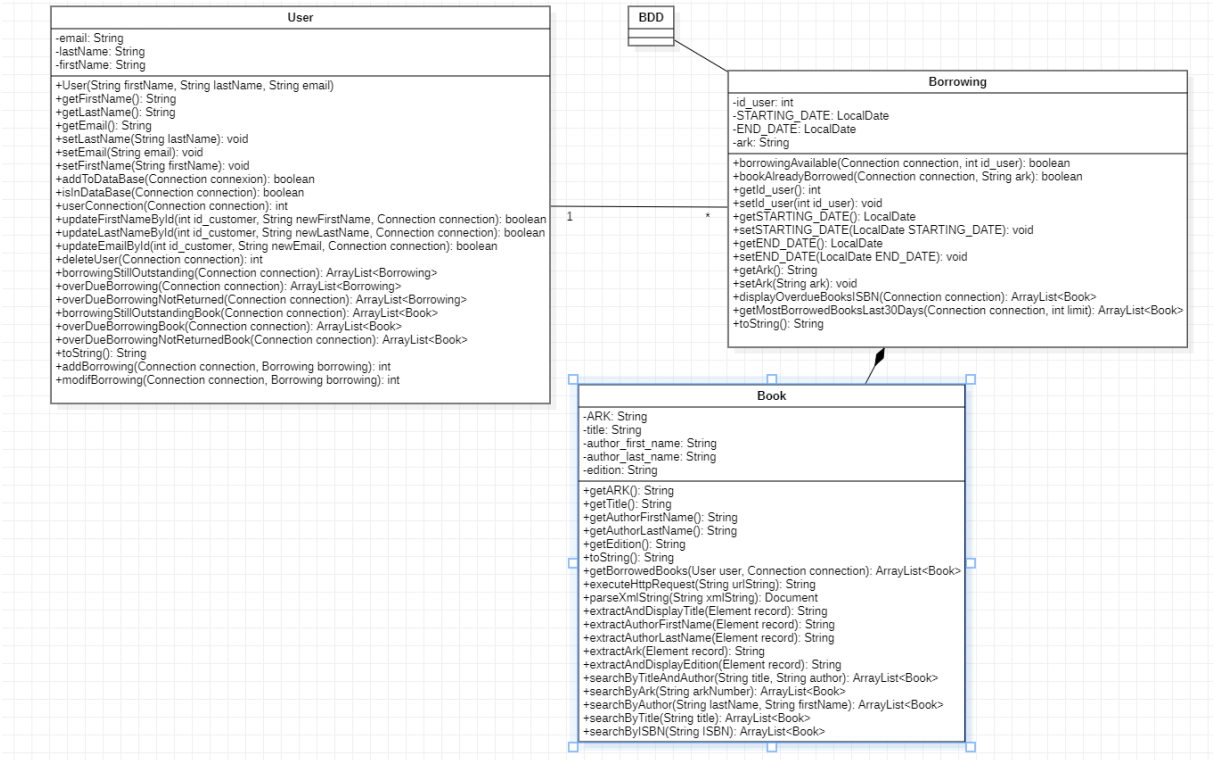
Rédouane BOUHAMOUM

Sommaire

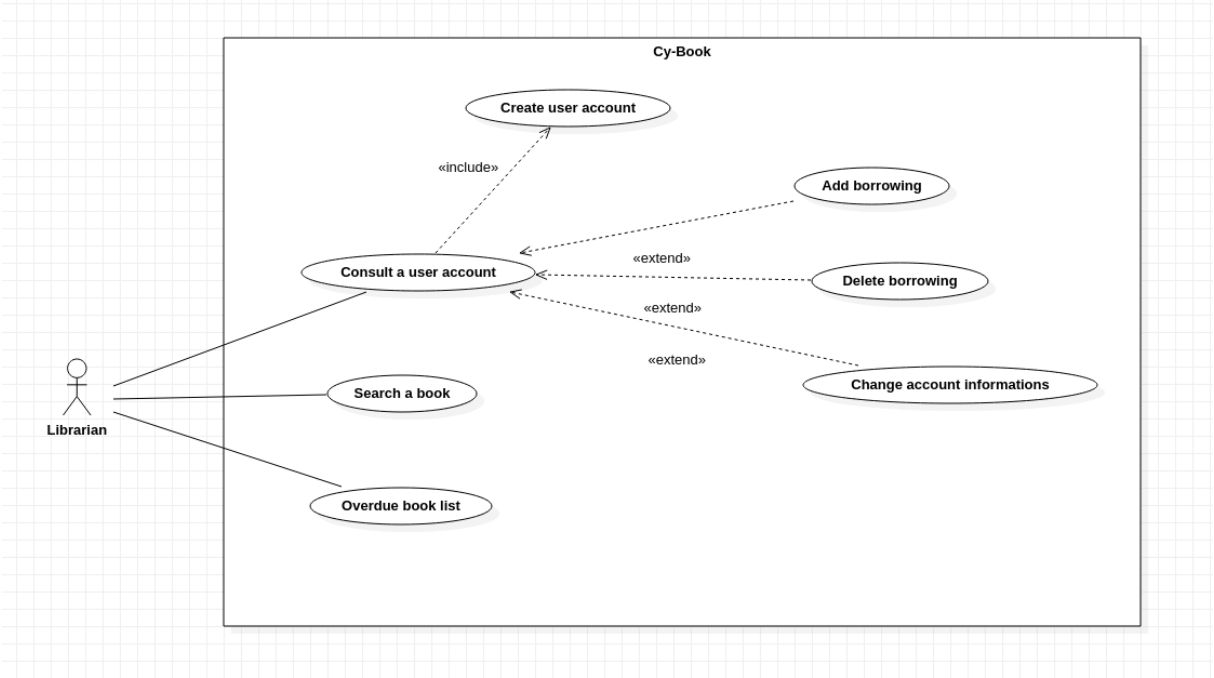
1. PRÉPARATION

1.1. Diagrammes

1.1.1. Diagramme de classes



1.1.2. Cas d'utilisation



1.2. Travail de groupe

1.2.1. GitHub

Les fichiers ont été régulièrement enregistrés sur un dépôt GitHub afin de faciliter le suivi de l'avancement du projet et de minimiser les erreurs lors des mises à jour.

1.2.2. Discord

Les réunions de groupe ont été tenues sur un serveur Discord spécialement créé pour le projet. Ce serveur a permis de centraliser les communications, de faciliter la coordination entre les membres de l'équipe et d'assurer une collaboration fluide et efficace tout au long du projet.

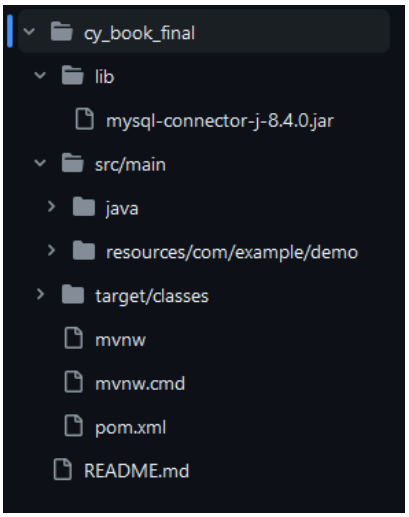
1.3. Répartition des tâches

Java FX : Marius BROSSET / Zakaria LESHOB / Alexy ANDRE
Java / SQL : Alexy ANDRE / Liam MARTIN
API : Estelle ROUDET

2. DÉVELOPPEMENT TECHNIQUE

2.1. Structure du code

Les fichiers .java sont contenus dans le dossier java et les .fxml dans le dossier ressources.



2.2. Java et SQL

Voici les différents types de recherches disponibles :

```
Choose the option of your choice by typing the associated number
(1) --> Book search
(2) --> Create a customer account
(3) --> Display customer information
(4) --> Stats

(0) --> Quit
```

2.2.1. Constructeurs

```
public User(String firstName, String lastName, String email) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
}
```

```java
public Borrowing(int id_user, LocalDate STARTING_DATE, LocalDate END_DATE, String ark) {
    this.id_user = id_user;
    this.STARTING_DATE = STARTING_DATE;
    this.END_DATE = END_DATE;
    this.ark = ark;
}
```

```java
public Book(String title, String author_first_name, String author_last_name, String ARK, String edition) {
    this.title = title;
    this.author_first_name = author_first_name;
    this.author_last_name = author_last_name;
    this.ARK = ARK;
    this.edition = edition;
}
```

2.2.2. Ajouter un client

Cette fonction permet d'ajouter un client dans la base de données :

```java
private static void createCustomerAccount(Scanner sc, Connection connection) {
    String name_customer;
    String last_name_customer;
    String email_customer;
    name_customer = getUserInput(sc, prompt: "Enter name: ");
    if (name_customer.isEmpty()) {...}
    last_name_customer = getUserInput(sc, prompt: "Enter last name: ");
    if (last_name_customer.isEmpty()) {
        System.out.println("Last name cannot be empty.");
        return;
    }
    email_customer = getUserInput(sc, prompt: "Enter email: ");
    if (email_customer.isEmpty() || !isValidEmail(email_customer)) {
        System.out.println("Invalid email format.");
        return;
    }
    User u1 = new User(name_customer, last_name_customer, email_customer);
    if (!(u1.isInDataBase( connection))) {
        if(u1.addToDataBase(connection)){
            System.out.println("The customer has been added");
        }
        else{
            System.out.println("The user can't be add in the database");
        }
        System.out.println("The customer informations are : \n-->" + u1.getFirstName() + "\n-->" + u1.getLastName() + "\n-->" + u1.getEmail());
    }
    else {
        System.out.println("Customer already in the data base");
    }
}
```

2.2.3. Modification des données d'un client

On voit ici un exemple d'une fonction pour modifier les données d'un client (le prénom), on a les mêmes fonctions pour la modification du nom et du courriel.

```java
public static boolean updateFirstNameById(int id_customer ,String newFirstName, Connection connection) {

    boolean updated = false;
    try {
        String query = "UPDATE customer SET name_customer = ? WHERE id_customer = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            preparedStatement.setString( parameterIndex: 1, newFirstName);
            preparedStatement.setInt( parameterIndex: 2, id_customer);
            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                updated = true;
                System.out.println("First name updated successfully");
            } else {
                System.out.println("Failed to update first name");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return updated;
}
```

2.2.4. Gestion des emprunts

On a la classe « Borrowing » pour stocker les informations d'un emprunt, avec l'attribut booléen « borrowed » qui nous précise si l'emprunt est en cours ou non.

```java
public int addBorrowing(Connection connection, Borrowing borrowing){
    int id_user = this.userConnection(connection);
    if(id_user == -1 ) {
        return -1;
    }
    if( !(Borrowing.bookAlreadyBorrowed(connection,borrowing.getArk())))
        return -4;
    if(Borrowing.borrowingAvailable(connection,id_user)){
        String query = "INSERT INTO borrowing (id_customer, isbn, start_date, end_date,borrowed) VALUES (?, ?, ?, ?,?)";
        try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            preparedStatement.setInt( parameterIndex: 1, id_user);
            preparedStatement.setString( parameterIndex: 2, borrowing.getArk());
            preparedStatement.setDate( parameterIndex: 3, Date.valueOf(borrowing.getSTARTING_DATE()));
            preparedStatement.setDate( parameterIndex: 4, Date.valueOf(borrowing.getEND_DATE()));
            preparedStatement.setBoolean( parameterIndex: 5, x: true);
            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                return 0;
            } else {
                return -3;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return -3;
        }
    } else {
        return -2;
    }
}
```

2.3. Java FX

2.3.1. Changement entre les pages

Fonction loadPage :

```
3 usages
private void loadPage(String fxmlFile) {
    try {
        FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource(fxmlFile));
        Scene scene = new Scene(fxmlLoader.load());
        Stage stage = (Stage) buttonOverdueBook.getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Pour envoyer un objet à la page où on se rend, on utilise loadPage2 :

```
1 usage
private void loadPage2(String fxmlFile, User user) {
    try {
        FXMLLoader loader = new FXMLLoader(getClass().getResource(fxmlFile));
        Parent root = loader.load();

        CustomerInformationsController customerInformationsController = loader.getController();
        customerInformationsController.setUser(user);

        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

2.3.2. Application d'un style ccs

Liaison du style dans le container principal du fichier .fxml :

```
<VBox fx:id="rootVBox" stylesheets="@styles.css"
```

2.3.3. Connexion à la base de données

Spécification des identifiants et du port :

```
try {
    Connection connection = DriverManager.getConnection(
            url: "jdbc:mysql://127.0.0.1:3306/cy_book",
            user: "root",
            password: "book"
    );
```

2.4. API

2.4.1. ExecuteHttpRequest

executeHttpRequest sert à envoyer une requête httpGet à un URL donnée et récupère la réponse sous forme de chaîne de caractères. On utilise un BufferReader pour lire la réponse du serveur ligne par ligne. Chaque ligne est ensuite ajoutée au String Builder réponse puis renvoyé à la fin de la fonction.

```
public static String executeHttpRequest(String urlString) throws IOException {
    StringBuilder response = new StringBuilder();
    // A class used to create mutable string objects.
    // Unlike String, StringBuilder objects can be modified without creating new objects every time.
    HttpURLConnection conn = (HttpURLConnection) new URL(urlString).openConnection();
    // .openConnection(): Opens a connection to the specified URL.
    conn.setRequestMethod("GET");
    // This is a GET request, used to retrieve data from the server.

    try (BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()))) {
        // Opens an input stream from the HTTP connection, allowing to read the server's response.
        String line;
        while ((line = reader.readLine()) != null) {
            // Retrieves a line from the BufferedReader and adds it to the response
            response.append(line);
        }
    }
    // Converts the content of the StringBuilder to a string
    return response.toString();
}
```

2.4.2. ParseXmlString :

Prend en argument la réponse envoyée par executeHttpRequest qui est sous format XmlString, on utilise DocumentBuildFactory une classe utilisée pour obtenir des Document Builder spécialement utilisé pour analyser des documents XML. La méthode parse() permet d'analyser le contenu XML et renvoie l'objet Document qui contient le document XML analysé, c'est à dire décomposé en éléments et attributs.

```
5 usages  new *
public static Document parseXmlString(String xmlString) throws Exception {
    // newInstance creates a new instance of DocumentBuilder
    // DocumentBuilderFactory is a class used for parsing XML documents
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    // Parses the provided XML content
    return factory.newDocumentBuilder().parse(new InputSource(new StringReader(xmlString)));
}
```

2.4.3. ExtractAndDisplayTitle :

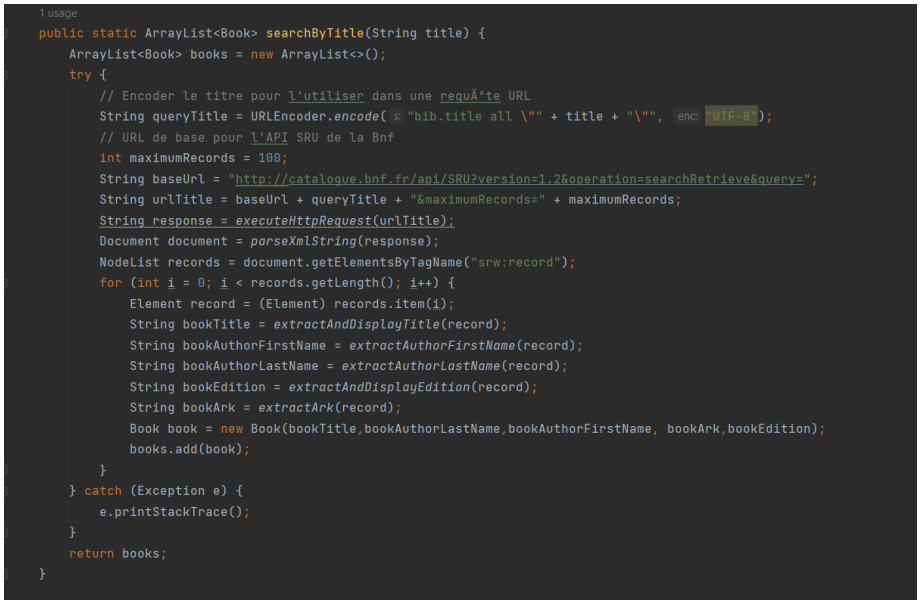
Prend un élément XML et extrait le titre en fonction du schéma UNIMARC. Cette fonction récupère tous les Datafield et cherche celui contenant le tag 200 qui est le champ principal du titre du livre puis il recherche tous les subfield de ce Datafield et recherche le sous champs "a" qui représente le titre principal.

```
5 usages
private static String extractAndDisplayTitle(Element record) {
    // Initialize an empty string to store the title
    String title = "";

    // Returns all child elements of record with the mxc tag...
    NodeList titleList = record.getElementsByTagName("mxc:datafield");
    for (int j = 0; j < titleList.getLength(); j++) {
        // Casts the j-th element to Element and stores it in datafield
        Element datafield = (Element) titleList.item(j);
        // Searches for the title field with the tag 200
        if ("200".equals(datafield.getAttribute( name: "tag"))) {
            // Checks the tag attributes with value 200
            // 200 in UNIMARC represents the title in the XML schema
            NodeList subfields = datafield.getElementsByTagName("mxc:subfield");
            // Iterates through the elements retrieved in subfields
            for (int k = 0; k < subfields.getLength(); k++) {
                // Casts the k-th element of subfields to Element
                Element subfield = (Element) subfields.item(k);
                // Searches for the subfield 'a' which represents the main title in UNIMARC
                if ("a".equals(subfield.getAttribute( name: "code"))) {
                    // If subfield has 'a', store the title
                    title = subfield.getTextContent();
                }
            }
        }
    }
```

2.4.4. SearchByTitle :

Cette fonction encode le titre pour le rendre sûr à utiliser, maximum records spécifie le nombre maximum de résultats puis exécute la requête en utilisant httpRequest et la parse grâce à parseXml. Puis Document.getElementByTagName récupère tous les records trouvés et pour chaque record rentre dans les fonctions extractAndDisplay selon ce que la fonction doit retourner (titre, auteur, éditions, ark ...) et retourne un Book contenant.



```
1 usage
public static ArrayList<Book> searchByTitle(String title) {
    ArrayList<Book> books = new ArrayList<>();
    try {
        // Encoder le titre pour l'utiliser dans une requÃªte URL
        String queryTitle = URLEncoder.encode( s: "bib.title all \"" + title + "\"",  enc: "UTF-8");
        // URL de base pour l'API SRU de la Bnf
        int maximumRecords = 100;
        String baseUrl = "http://catalogue.bnf.fr/api/SRU?version=1.2&operation=searchRetrieve&query=";
        String urlTitle = baseUrl + queryTitle + "&maximumRecords=" + maximumRecords;
        String response = executeHttpRequest(urlTitle);
        Document document = parseXmlString(response);
        NodeList records = document.getElementsByTagName("srw:record");
        for (int i = 0; i < records.getLength(); i++) {
            Element record = (Element) records.item(i);
            String bookTitle = extractAndDisplayTitle(record);
            String bookAuthorFirstName = extractAuthorFirstName(record);
            String bookAuthorLastName = extractAuthorLastName(record);
            String bookEdition = extractAndDisplayEdition(record);
            String bookArk = extractArk(record);
            Book book = new Book(bookTitle,bookAuthorLastName,bookAuthorFirstName, bookArk,bookEdition);
            books.add(book);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return books;
}
```

3. PROBLEMES RENCONTRES

3.1. Utilisation de GitHub

Lors de l'avancement de notre projet, nous avons rencontré plusieurs difficultés liées à l'utilisation de GitHub, ce qui a impacté notre capacité à collaborer efficacement.

3.2. Problème de liaison de nos projets respectifs

Chacun des membres de l'équipe travaillait sur des parties distinctes du projet. Nous avons eu du mal à lier nos projets respectifs à un dépôt GitHub commun. Nous avons eu des problèmes de synchronisation des branches et de fusion des différentes parties du code.

3.3. Problèmes avec l'API

3.3.1. Limites de SPARQL

Au début de notre projet nous avons choisi l'utilisation de la base de données SPARQL pour faire de recherche dans l'API. La base de données SPARQL a bien fonctionné pour la plupart de nos requêtes et nous offrait de nombreuses possibilités, mais nous avons rencontré des difficultés pour la récupération du numéro unique d'un livre, l'ISBN et le ARK. La base de données SPARQL ne nous permettait pas de récupérer le numéro ISBN à partir d'un livre précis, ou du moins nous n'avons pas réussi. Nous avons donc dû recommencer à zéro et passer au requêtes SRU et XML, nous avons donc finalement réussi à récupérer le numéro ARK et rechercher par numéro ARK et par ISBN. Mais cela nous a fait perdre beaucoup de temps.