Dossier de Conception des Classes de l'Application E-Library

1. Introduction

Ce document décrit les classes principales de l'application E-Library, détaillant leurs attributs et leurs méthodes. Il inclut également un diagramme de classes pour visualiser les relations et les interactions entre ces composants.

2. Structures des Classes

L'application E-Library est structurée autour de trois classes de données fondamentales (Book, User) et deux classes de logique/interface (LibraryManager, MainWindow).

2.1 Classe Book

La classe Book représente une **copie physique unique** d'un livre dans la bibliothèque. Chaque copie possède un identifiant unique pour permettre un suivi précis.

Attributs :

Attribut	**Type**	**Description**
bookId	QString	Identifiant unique de la copie physique du livre (UUID).
title	QString	Titre du livre.
author	QString	Auteur du livre.
isbn	QString	ISBN du livre (peut être le même pour plusieurs copies).
isBorrowed	bool	Indique si cette copie est actuellement empruntée.
borrowedByUserId	QString	ID de l'utilisateur qui a emprunté le livre, si emprunté.
borrowDate	QDate	Date à laquelle le livre a été emprunté, si emprunté.
returnDueDate	QDate	Date de retour prévue, si emprunté.
isReserved	bool	Indique si cette copie est actuellement réservée.
reservedByUserId	QString	ID de l'utilisateur qui a réservé le livre, si réservé.

Méthodes :

| Méthode | Signature ```

2.2. Classe User

La classe User représente un usager de la bibliothèque, qu'il s'agisse d'un étudiant ou d'un enseignant.

Attributs :

Attribut	Type	Description
id	QString	Identifiant unique de l'utilisateur (UUID).
name	QString	Nom complet de l'utilisateur.
phoneNumber	QString	Numéro de téléphone de l'utilisateur.
gmailAddress	QString	Adresse Gmail de l'utilisateur.

Méthodes :

Méthode	Signature	Description
User	User(const QString& name, const QString& phoneNumber, const QString& gmailAddress)	Constructeur. Génère un ID unique.
toString	QString toString() const	Convertit les données de l'utilisateur en une chaîne formatée pour la sauvegarde.
fromString	static User fromString(const QString& data)	Crée un objet User à partir d'une chaîne de données lue d'un fichier.

2.3. Classe LibraryManager

La classe LibraryManager est le cœur de la logique métier de l'application. Elle gère la collection de livres et d'utilisateurs, ainsi que la persistance des données.

Attributs :

Attribut	Type	Description

m_booksFilePath	QString	Chemin du fichier où les données des livres sont stockées.
m_usersFilePath	QString	Chemin du fichier où les données des utilisateurs sont stockées.
m_books	QVector<Book>	Collection de toutes les copies physiques des livres.
m_users	QVector<User>	Collection de tous les utilisateurs enregistrés.
MAX_BORROW_DAYS	const int	Durée maximale d'emprunt en jours (14 jours).
PENALTY_PER_DAY	const double	Montant de la pénalité par jour de retard (100 FCFA).

Méthodes Publiques :

Méthode	**Signature**	**Description**
LibraryManager	explicit LibraryManager(const QString& booksFile, const QString& usersFile, QObject *parent = nullptr)	Constructeur. Initialise les chemins des fichiers et charge les données existantes.
~LibraryManager	~LibraryManager()	Destructeur. Sauvegarde toutes les données avant de quitter.
addBook	bool addBook(const Book& bookTemplate, int numberOfCopies)	Ajoute une ou plusieurs copies d'un livre à la bibliothèque.
removeBook	bool removeBook(const QString& bookId)	Supprime une copie physique spécifique d'un livre.
borrowBook	bool borrowBook(const QString& bookId, const QString& userId, const QDate& borrowDate)	Permet à un utilisateur d'emprunter un livre.

returnBook	QPair<bool, double> returnBook(const QString& bookId)	Permet de retourner un livre, en calculant une pénalité si nécessaire.
reserveBook	bool reserveBook(const QString& bookId, const QString& userId)	Permet à un utilisateur de réserver un livre.
cancelReservation	bool cancelReservation(const QString& bookId)	Annule une réservation de livre.
getAllBooks	QVector<Book> getAllBooks() const	Récupère toutes les copies de livres.
addUser	bool addUser(const User& user)	Ajoute un nouvel utilisateur au système.
findUserByDetails	User* findUserByDetails(const QString& name, const QString& phone, const QString& gmail)	Recherche un utilisateur par ses détails.
getAllUsers	QVector<User> getAllUsers() const	Récupère tous les utilisateurs enregistrés.
sendUpdateEmails	void sendUpdateEmails(const QString& subject, const QString& body)	Simule l'envoi d'e-mails de mise à jour à tous les utilisateurs (fonctionnalité simulée)

Méthodes Privées (pour la persistance des données) :

Méthode	**Signature**	**Description**
loadBooks	void loadBooks()	Charge les données des livres depuis le fichier.
saveBooks	void saveBooks()	Sauvegarde les données des livres dans le fichier.
loadUsers	void loadUsers()	Charge les données des utilisateurs depuis le fichier.

saveUsers	void saveUsers()	Sauvegarde les données des utilisateurs dans le fichier.
sendEmail	void sendEmail(const QString& recipientEmail, const QString& subject, const QString& body)	Fonction de simulation d'envoi d'e-mail.

2.4. Classe MainWindow

La classe MainWindow est responsable de l'interface utilisateur graphique (GUI) de l'application. Elle interagit avec l'utilisateur et communique avec LibraryManager pour exécuter les opérations de la bibliothèque.

Attributs :

Attribut	**Type**	**Description**
ui	Ui::MainWindow *	Pointeur vers l'interface utilisateur générée par Qt Designer.
m_libraryManager	LibraryManager	Instance de la classe LibraryManager pour la logique métier.
m_currentUserId	QString	ID de l'utilisateur actuellement connecté.
m_currentUserName	QString	Nom de l'utilisateur actuellement connecté.
LIBRARIAN_PASSWORD	const QString	Mot de passe codé en dur pour le bibliothécaire.

Slots Privés (réactions aux interactions utilisateur) :

Slot	**Signature**	**Description**
on_librarianRadioButton_toggled	void on_librarianRadioButton_toggled(bool checked)	Gère le basculement du bouton radio "Bibliothécaire".

on_studentTeacherRadioButton_toggled	void on_studentTeacherRadioButton_toggled(bool checked)	Gère le basculement du bouton radio "Étudiant/Enseignant".
on_loginButton_clicked	void on_loginButton_clicked()	Gère le clic sur le bouton de connexion.
on_addBookButton_clicked	void on_addBookButton_clicked()	Gère le clic sur le bouton "Ajouter Livre".
on_removeBookButton_clicked	void on_removeBookButton_clicked()	Gère le clic sur le bouton "Supprimer Livre".
on_sendUpdatesButton_clicked	void on_sendUpdatesButton_clicked()	Gère le clic sur le bouton "Envoyer Mises à Jour".
on_borrowBookButton_clicked	void on_borrowBookButton_clicked()	Gère le clic sur le bouton "Emprunter Livre".
on_returnBookButton_clicked	void on_returnBookButton_clicked()	Gère le clic sur le bouton "Retourner Livre".
on_reserveBookButton_clicked	void on_reserveBookButton_clicked()	Gère le clic sur le bouton "Réserver Livre".
on_cancelReservationButton_clicked	void on_cancelReservationButton_clicked()	

Méthodes Privées (utilitaires GUI) :

Méthode	Signature	Description
updateBooksTableLibrarian	void updateBooksTableLibrarian()	Met à jour le tableau des livres affiché pour le bibliothécaire.
updateBooksTableStudentTeacher	void updateBooksTableStudentTeacher()	Met à jour le tableau des livres affiché pour l'étudiant/enseignant.
showMessage	void showMessage(const QString& title, const QString& message)	Affiche une boîte de dialogue avec un message à l'utilisateur.

3. Représentation Synoptique des Interactions entre les Classes (Diagramme de Classes UML)

Voici un diagramme de classes simplifié pour visualiser les relations entre les principales classes de l'application.

```
classDiagram
    class Book {
        + QString bookId
        + QString title
        + QString author
        + QString isbn
        + bool isBorrowed
        + QString borrowedByUserId
        + QDate borrowDate
        + QDate returnDueDate
```

```
    + bool isReserved
    + QString reservedByUserId
    + Book(title, author, isbn)
    + toString() QString
    + static fromString(data) Book
}

class User {
    + QString id
    + QString name
    + QString phoneNumber
    + QString gmailAddress
    + User(name, phoneNumber, gmailAddress)
    + toString() QString
    + static fromString(data) User
}

class LibraryManager {
    - QString m_booksFilePath
    - QString m_usersFilePath
    - QVector~Book~ m_books
    - QVector~User~ m_users
    + const int MAX_BORROW_DAYS
    + const double PENALTY_PER_DAY
    + LibraryManager(booksFile, usersFile)
    + ~LibraryManager()
    + addBook(bookTemplate, numberOfCopies) bool
    + removeBook(bookId) bool
```

+ borrowBook(bookId, userId, borrowDate) bool

+ returnBook(bookId) QPair~bool, double~

+ reserveBook(bookId, userId) bool

+ cancelReservation(bookId) bool

+ getAllBooks() QVector~Book~

+ addUser(user) bool

+ findUserByDetails(name, phone, gmail) User*

+ getAllUsers() QVector~User~

+ sendUpdateEmails(subject, body) void

- loadBooks() void

- saveBooks() void

- loadUsers() void

- saveUsers() void

- sendEmail(recipientEmail, subject, body) void

}

class MainWindow {

- Ui::MainWindow *ui

- LibraryManager m_libraryManager

- QString m_currentUserId

- QString m_currentUserName

- const QString LIBRARIAN_PASSWORD

+ MainWindow(parent)

+ ~MainWindow()

-- Slots --

+ on_librarianRadioButton_toggled(checked) void

+ on_studentTeacherRadioButton_toggled(checked) void

+ on_loginButton_clicked() void

```
    + on_addBookButton_clicked() void
    + on_removeBookButton_clicked() void
    + on_sendUpdatesButton_clicked() void
    + on_borrowBookButton_clicked() void
    + on_returnBookButton_clicked() void
    + on_reserveBookButton_clicked() void
    + on_cancelReservationButton_clicked() void
    -- Private Methods --
    - updateBooksTableLibrarian() void
    - updateBooksTableStudentTeacher() void
    - showMessage(title, message) void
}

MainWindow --> LibraryManager : utilise (agrégation)
LibraryManager o-- Book : contient (composition)
LibraryManager o-- User : contient (composition)
Book --|> QString : utilise (héritage d'interface implicite via types Qt)
User --|> QString : utilise (héritage d'interface implicite via types Qt)
```

Explication des Interactions :

- **MainWindow utilise LibraryManager (Agrégation)**:
    - MainWindow détient une instance de LibraryManager (m_libraryManager).
    - Toutes les actions de l'interface utilisateur qui nécessitent une logique métier (ajouter un livre, emprunter, etc.) sont déléguées à l'objet m_libraryManager.
    - MainWindow ne gère pas directement les données des livres ou des utilisateurs, mais fait appel à LibraryManager pour cela.
- **LibraryManager contient Book (Composition)**:
    - LibraryManager gère une collection (QVector) d'objets Book.

    - La durée de vie des objets Book est gérée par LibraryManager (chargement depuis le fichier, ajout, suppression).
- **LibraryManager contient User (Composition)**:
    - Similaire à Book, LibraryManager gère une collection (QVector) d'objets User.
    - La durée de vie des objets User est gérée par LibraryManager.
- **Book et User utilisent des types Qt (QString, QDate, QUuid)**:
    - Bien qu'il n'y ait pas d'héritage direct dans le sens C++, ces classes dépendent fortement des types de données de Qt pour leurs attributs et méthodes (par exemple, pour la gestion des chaînes, des dates et des UUID). C'est une dépendance forte sans être un héritage formel d'interface.

Ce diagramme met en évidence la séparation des préoccupations : MainWindow gère la présentation, tandis que LibraryManager gère la logique métier et la persistance, en s'appuyant sur les structures de données Book et User.