



Rapport de projet

MyLibrary

Travail pratique individuel 2022

Karel Vilém Svoboda
I.DA-P4A
Version 1.0
18.05.2022

Résumé du rapport de TPI

MyLibrary

Situation de départ

Pour valider la formation d'Informaticien-ne CFC, les apprentis doivent effectuer un « Travail pratique individuel » (TPI) d'une durée de 88 heures (11 jours), sur la base d'un énoncé imposé. Le projet qui m'a été attribué est MyLibrary.

Son objectif est de noter les références culturelles que l'on retrouve lorsque l'on lit un livre.

Le projet est composé en deux parties, la partie backend sera faite en API PHP et communiquera avec la base de données indépendamment avec un CRUD. La partie frontend sera une application C# sous Windows Form et communiquera avec l'API en REST (Json).

L'application ne contiendra aucun système d'administration ou de modération. Toutes les données seront strictement liées à l'utilisateur et ne pourront être ajoutées, supprimées et modifiées que par ce dernier. La sécurité de la connexion sera gérée par l'API.

Deux utilisateurs sont pré-créés dans la base de données. L'application ne contiendra pas de systèmes de création de comptes.

Mise en oeuvre

L'API PHP sera développée sur un serveur Ubuntu en local et l'application sera, quant à elle, créée dans Visual Studio 2022. Afin de communiquer avec l'API, l'application C# utilisera la librairie System.Net et afin d'interpréter le JSON qu'elle reçoit, elle utilisera le package NuGet Newtonsoft.Json.

Résultat

Les utilisateurs finaux ont à leur disposition une application c# fluide et simple d'utilisation ainsi qu'une API utilisable de façon indépendante de l'application. Un manuel d'utilisateur est mis à disposition. Le code c# correspond aux normes du CFPT. Un rapport de projet contenant une documentation technique complète a également été rédigée.

Table des matières

Résumé du rapport de TPI	2
MyLibrary	2
Situation de départ	2
Mise en oeuvre	2
Résultat	2
Table des matières	3
1.0 Introduction	7
2.0 Rappel du cahier des charges	7
2.1 Organisation	7
2.2 Restrictions	7
2.3 Livrables	7
2.5 Tâches	8
2.5.1 API	8
2.6.1 Application C#	9
3.0 Analyse fonctionnelle	11
3.1 Fonctionnalités	11
3.1.1 API	11
Points d'entrées de gestion de session	11
Connexion	11
Déconnexion	11
Points d'entrées de type POST	11
Tous les livres de l'utilisateur	11
recherche personnalisée	12
Par IdLivre	12
dernier livre	12
Références par idLivre	13
Ajouter un livre	13
Ajouter une référence de type Livre	14
Ajouter une référence de type musique	14
Ajouter une référence de type lieu	15
Commandes de type PUT	15
Modifier un livre	15

Modifier une référence de type Livre	16
Modifier une référence de type Musique	16
Modifier une référence de type Lieu	17
Points d'entrées de type DELETE	17
3.1.2 Application C#	17
3.1.2.1 Connexion	18
3.1.2.2 CollectionLivres	19
3.1.2.2 CollectionReferences	21
3.2 Méthodologie	24
3.2 Base de données	25
3.2.1 Contraintes	25
3.2.2 Diagramme	25
3.2.2 Tables	26
3.2.2.1 Type	26
3.2.2.2 Référence	26
3.2.2.3 Livres	26
3.2.2.4 Utilisateurs	27
3.2.3 Sql	27
3.2.3 Utilisateur de la base	32
4.0 Analyse organique	33
4.1 Architecture du projets	33
4.2 Système de sécurité de l'API	34
4.3 Connexion	35
4.X Héritage des Card	36
4.X Héritage des références	37
4.3 Technologies utilisées	38
5.0 Interfaces	39
5.1 Maquettes de l'application C#	39
5.1.1 Connexion	39
5.1.2 Collection de livres	39
5.1.3 Collection de références	40
6.0 Architecture du projet	41
Architecture générale	41
API PHP	41
Passage des données	42
Api Rest	42

MyLibrary	43
index.php	43
Class encapsulées Livre, Reference, Type et Utilisateur	43
Application C#	44
Class ClientRest	45
Class Card	45
Class CardLivre	46
CardReference	47
CardReferenceLieu	47
CardReferenceLivre	48
CardReferenceMusique	48
Class ImageInFiles.cs	49
Class ReferenceLieu	50
Class ReferenceLivre	50
Class ReferenceMusique	51
Class Livre	51
Class Reference	52
Class Type	54
Class Utilisateur	54
frmConnexion	55
FrmCollectionLivres	57
FrmCollectionReferences	59
7.0 Tests	61
7.1 Scénario de tests	61
2.6.1 API	61
2.6.1 Application C#	62
7.2 Evolution des tests	65
8.0 Conclusion	65
8.1 Difficultés rencontrées	65
8.2 Améliorations possibles	66
8.3 Bilan personnel	66
9.0 Annexes	66
9.1 Sources	66
10.0 Code source	66

1.0 Introduction

L'objectif du projet MyLibrary est de créer une bibliothèque personnelle afin que l'utilisateur puisse noter les références culturelles qu'il retrouve dans les livres qu'il lit. Par exemple, dans le roman Notre-Dame de Paris de Victor Hugo, l'utilisateur pourra noter dans l'application la cathédrale Notre-Dame de Paris.

2.0 Rappel du cahier des charges

2.1 Organisation

Fonction	Nom	Email
Élève	Karel Vilém Svoboda	karel.svbd@eduge.ch
Maître d'apprentissage	Jasmina Travnjak	edu-travnjakj@eduge.ch
Expert	Borys Folomietow	borys@folomietow.ch
Expert	Jacques Ortola	jacques.ortola@gmail.com

2.2 Restrictions

1. 10 Jours de réalisation
2. Rendre le planning prévisionnel la première journée (02.05.2022) à 17h.

2.3 Livrables

Livrable	Date de rendu
Planning initial	02.05.2022
Planning initial et effectif	18.05.2022
Résumé du TPI	18.05.2022
Rapport de projet incluant le code source du projet	18.05.2022
Manuel utilisateur	18.05.2022
Journal de travail	18.05.2022

2.5 Tâches

2.5.1 API

Titre	Connexion à l'API
Description	<ol style="list-style-type: none">1. Requête de type GET2. En cas de succès, la donnée de l'utilisateur <i>connecte</i> dans la base de données change de 0 à 1
Titre	Connexion unique
Description	Lors de la connexion, l'API vérifie la donnée <i>connecte</i> dans la base de donnée et autorise une nouvelle connexion ou pas en conséquence
Titre	CRUD sur la table Livres
Description	<ol style="list-style-type: none">1. Pouvoir afficher les livres2. Pouvoir ajouter un livre3. Pouvoir modifier un livre4. Pouvoir supprimer un livre
Titre	Récupération dynamique de livres
Description	Pouvoir récupérer un livre par <ol style="list-style-type: none">1. Son auteur2. Sa description3. Par l'utilisateur qu'il l'a ajouté
Titre	CRUD sur la table References
Description	<ol style="list-style-type: none">1. Pouvoir afficher les références2. Pouvoir ajouter une référence3. Pouvoir modifier une référence4. Pouvoir supprimer une référence

Titre	Récupération dynamique de références
Description	Pouvoir récupérer une référence par <ol style="list-style-type: none"> 1. Son type 2. Du Livre 3. Son texte

2.6.1 Application C#

Titre	Connexion
Description	1. Si l'utilisateur renseigne les bonnes données, il est redirigé sur la form collection de livres 2. Si les données sont erronées un message d'erreur est renvoyé

Titre	Récupération des données lors de la connexion
Description	Lors d'une connexion réussie, l'utilisateur est stocké dans une instance objet <i>Utilisateur</i> avec ses informations <ol style="list-style-type: none"> 1. idUtilisateur 2. Email 3. Password (sha1)

Titre	Création dynamique des <i>Card</i> de livres
Description	1. L'application fait un appel à l'API qui lui retourne l'intégralité des livres ajoutés par l'utilisateur 2. Affiche les données dans des <i>Card</i> (Objet Windows forms hérité de Panel) créés dynamiquement. 3. Chaque <i>Card</i> est ensuite affichés dans une List

Titre	Recherche de <i>Card Livres</i>
Description	Permet de faire une recherche dans la

	liste des <i>Card</i> qui contiennent un : - Auteur donné - Titre donné
--	---

Titre	CRUD sur les Livres via les <i>Card</i>
Description	Grâce un formulaire, l'application doit 1. Pouvoir ajouter un livre 2. Pouvoir modifier un livre 3. Pouvoir supprimer un livre

Titre	Ouverture d'une form collection de références à partir d'une <i>Card</i> Livre
Description	Suite à l'appui d'un bouton <i>Référence</i> d'une <i>Card</i> Livre une form s'ouvre avec le nom du livre en tant que titre

Titre	Récupération des référence d'un livre et affichage des résultats dans des card référence de différents types
Description	Récupération des référence d'un livre et création de <i>Card</i> de différentes types 1. Musique 2. Livre 3. Lieu

Titre	Filtrage des <i>Card</i> référence
Description	Affichage des card demandés par 1. Type 2. Texte

Titre	CRUD sur les références via les <i>Card</i>
Description	Grâce un formulaire, l'application doit 1. Pouvoir ajouter une références 2. Pouvoir modifier une références 3. Pouvoir supprimer une références

- a. Message de confirmation lors de la suppression

3.0 Analyse fonctionnelle

3.1 Fonctionnalités

3.1.1 API

L'API peut-être être utilisée indépendamment du projet c#. À chaque utilisation, veuillez vous connecter pour utiliser les points d'entrées et vous déconnecter à la fin.

Points d'entrées de gestion de session

Points d'entrées de type GET

Connexion

Headers	Valeur attendue
session	connexion
email	email de l'utilisateur
password	mot de passe de l'utilisateur

Déconnexion

Headers	Valeur attendue
session	deconnection
email	email de l'utilisateur
password	mot de passe de l'utilisateur

Points d'entrées de type POST

Table Livres

Tous les livres de l'utilisateur

Headers	Valeur attendue

table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur

recherche personnalisée

Headers	Valeur attendue
table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur
recherche	chaîne de caractères à rechercher

Par IdLivre

Headers	Valeur attendue
table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idLivre	id du livre à rechercher

dernier livre

Headers	Valeur attendue
table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur
tri	dernier

Table références

Références par idLivre

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idLivre	idLivre

Table Types

Headers	Valeur attendue
table	types
email	email de l'utilisateur
password	mot de passe de l'utilisateur

Points d'entrées de type POST

Livres

Ajouter un livre

Headers	Valeur attendue
table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur
titre	nouveau titre
auteur	nouvel auteur
nomImage	nom de l'image

Références

Ajouter une référence de type Livre

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idType	1
livreReference	id du livre de référence (données du livres de référence)
idLivre	id du livre auquel appartient la référence

Ajouter une référence de type musique

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
nomImage	nom de l'image de couverture
nomReference	titre de la musique
auteur	nom de l'auteur
idLivre	id du livre auquel appartient la référence
idType	2

Ajouter une référence de type lieu

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
nomReference	nom de l'endroit
descriptionLieu	description du lieu
idLivre	id du livre auquel appartient la référence
idType	3

Commandes de type PUT

Table Livres

Modifier un livre

Headers	Valeur attendue
table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idLivre	id du livre à modifier
titre	nouveau titre
auteur	nouvel auteur
nomImage	nouveau nom d'image

Table References

Modifier une référence de type Livre

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
livreReference	id du livre à modifier dans la table livre
idType	1
titre	nouveau titre
auteur	nouvel auteur
nomImage	nouveau nom d'image

Modifier une référence de type Musique

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idReference	id de la référence
idType	2
titre	nouveau titre
auteur	nouvel auteur
nomImage	nouveau nom d'image

Modifier une référence de type Lieu

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idReference	id de la référence
idType	2
titre	nouveau titre
descriptionLieu	nouvelle description du lieu

Points d'entrées de type DELETE

Supprimer un livre

Headers	Valeur attendue
table	livres
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idLivre	id du livre à supprimer

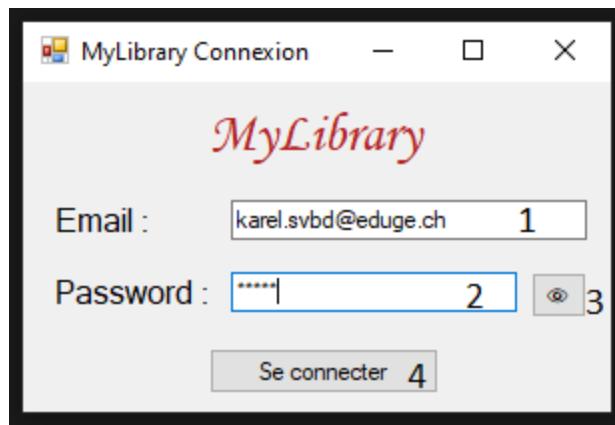
Supprimer une référence

Headers	Valeur attendue
table	references
email	email de l'utilisateur
password	mot de passe de l'utilisateur
idReference	id de la référence à supprimer

3.1.2 Application C#

3.1.2.1 Connexion

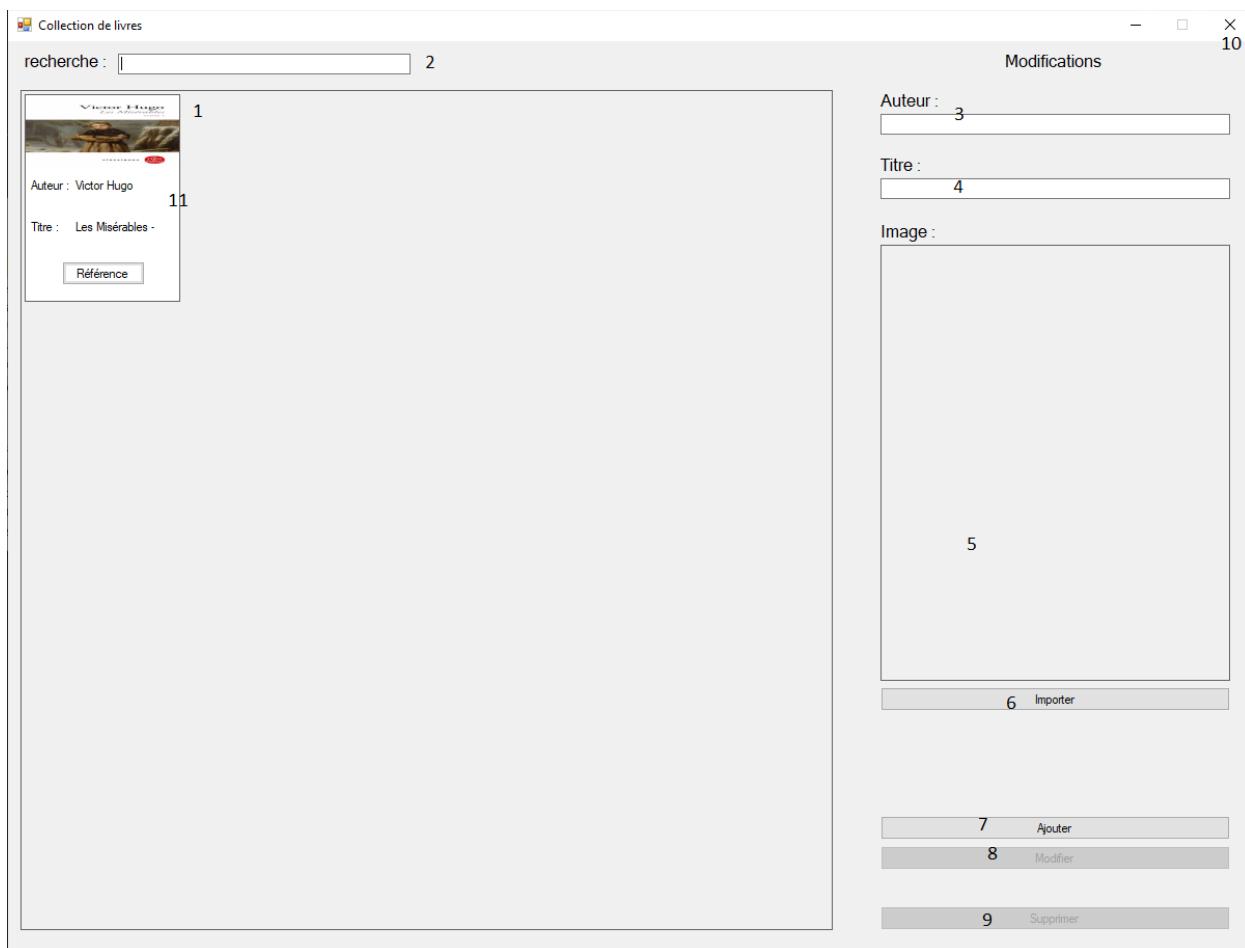
frmConnexion



1. Champs email
2. Champs mot de passe
3. Bouton d'affichage du mot de passe en clair (s'active uniquement si on maintient le bouton enfoncé)
4. Bouton de connexion

Réussite	Echec
Passage à la form collection de livres avec les données de l'utilisateur	Affichage d'un message d'erreur

3.1.2.2 CollectionLivres



1. Composant flowlayoutpanel qui contient la liste des Objets CardLivre qui affiche les livres de l'utilisateur
2. Recherche personnalisée. Recherche une chaîne de caractères dans l'auteur et le titre des livres et retourne le résultat dans la liste. L'application se met à jour à chaque changement de cet input.
3. Textbox sert de crud des livres. Si un livre est sélectionné, la donnée s'affiche. Sinon elle sert d'input pour ajouter la donnée à un nouveau livre.
4. Textbox sert de crud des livres. Si un livre est sélectionné, la donnée s'affiche. Sinon elle sert d'input pour ajouter la donnée à un nouveau livre.
5. Si l'utilisateur a sélectionné un livre dans la liste, cette picturebox affiche l'image de ce dernier en cherchant le fichier en local. Si l'utilisateur souhaite ajouter un livre, elle sert à indiquer à l'utilisateur l'image qu'il a sélectionnée dans l'open file dialog qui s'ouvre après la pression du bouton *Inspecter* (n.6).

6. Sert à ouvrir un openfiledialog qui enregistre l'image en locale en local si on ajoute ou modifie l'image d'un livre.
7. Permet de faire un appel au point d'entrée d'ajout de livre avec les données du formulaire. S'active uniquement si aucun livre n'est sélectionné.
8. Permet de faire un appel au point d'entrée de modification de livre avec les données du formulaire. S'active uniquement si un livre est sélectionné.
9. Permet de faire un appel au point d'entrée de suppression depuis le livre sélectionné. S'active uniquement si un livre est sélectionné. Affiche un message afin que l'utilisateur puisse confirmer la suppression.
10. Permet de fermer l'application. Dans l'événement formClosed, on appelle la méthode déconnexion() de l'utilisateur afin de remplacer la donnée dans la base (connecté de 1 à 0) afin que l'utilisateur puisse se reconnecter.
11. Objet CardLivre, affiche un livre dans la liste, composé d'une picturebox qui affiche l'image du livre, quatres label qui affichent l'auteur et le titre et un bouton qui permet d'accéder aux références du livre (ouvre frmCollectionReference(lui-même)).

3.1.2.2 CollectionReferences

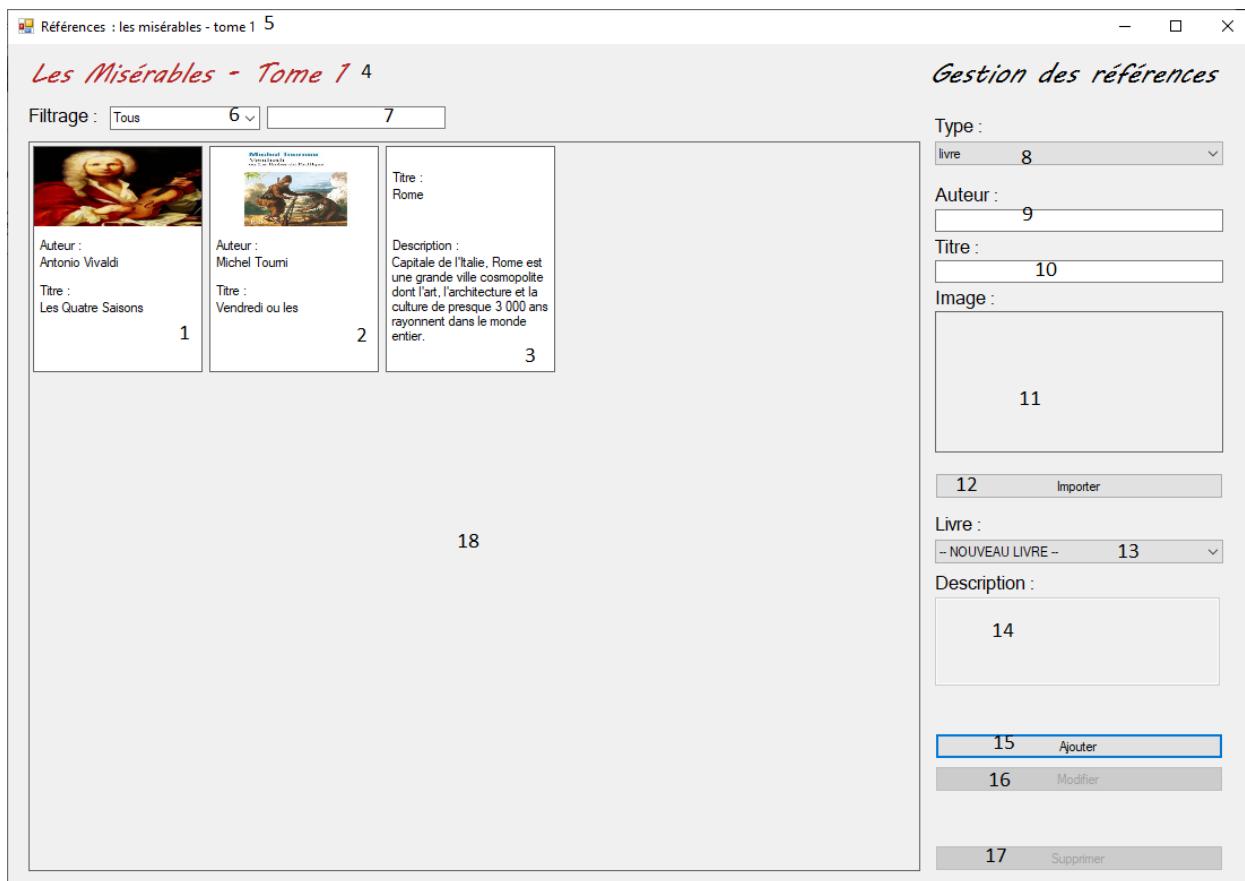
Permet à l'utilisateur d'interagir avec les références d'un livre. S'ouvre après une pression du bouton référence d'une CardLivre dans la frmCollectionLivres.

Pour afficher une référence, l'application contient une liste d'objet référence. À l'intérieur de cette dernière se trouve trois classes héritées qui sont générées automatiquement en fonction du type.

```
foreach (Reference reference in _references)
{
    Card nouvelleCard;
    switch (reference.IdType)
    {
        //Livre
        case 1:
            nouvelleCard = new CardReferenceLivre(_utilisateur, reference, this);
            flpReferences.Controls.Add(nouvelleCard);
            _cardsReferences.Add(nouvelleCard);
            break;
        //Musique
        case 2:
            nouvelleCard = new CardReferenceMusique(reference, this);
            flpReferences.Controls.Add(nouvelleCard);
            _cardsReferences.Add(nouvelleCard);
            break;
        //Lieu
        case 3:
            nouvelleCard = new CardReferenceLieu(reference, this);
            flpReferences.Controls.Add(nouvelleCard);
            _cardsReferences.Add(nouvelleCard);
            break;
    }
}
```

L'utilisation d'objet en fonction du type est utilisée à d'autres endroits dans le code. L'exemple le plus notable est le crud sur la table référence. Chaque type de référence appelle un point d'entrée différent de l'API, qui s'occupe de faire le tri et l'insérer dans la base de données.

Le système de tri se fait directement dans l'application C# au lieu d'appeler un point d'entrée de l'API comme dans la frmCollectionLivres.



1. CardReferenceMusique, hérite de cardReference. Affiche l'image, l'auteur et le titre de l'objet de l'objet de type Reference.
2. CardReferenceLivre, hérite de cardReference. Affiche l'image, l'auteur et le titre de l'objet de l'objet de type Reference qui est créé dynamiquement après récupération des données du livre.
3. CardReferenceLieu, hérite de CardReference. Affiche le titre et la description de l'endroit du champ de type Reference qu'il contient.
4. Label qui affiche le titre du livre.
5. Le texte de la page est généré dynamiquement avec le titre du livre.
6. Combobox qui permet un filtrage en fonction du type de référence.
 - a. Tous (affiche tous les types)
 - b. Livres (affiche uniquement les références de type Livre)
 - c. Musique (affiche uniquement les références de type Musique)
 - d. Lieu (affiche uniquement les références de type Lieu)
7. Permet de faire une recherche personnalisée. En cas de changement de text dans la textbox, l'application va rechercher dans les propriétés des références.

8. Permet de sélectionner le type de références. L'application s'adapte en fonction de ce dernier. L'application ne permet pas de modifier un type d'une card sélectionnée lors de la modification, en revanche, elle affiche le type actuel.
 - a. Type Livre : Auteur, Titre, Image, Livre (n13)
 - b. Type Musique : Auteur, Titre, Image
 - c. Type Lieu : Titre, Description
9. Si l'utilisateur veut ajouter une référence, la combobox sert d'input pour l'auteur, s'il sélectionne une référence pour la modifier ou la supprimer, elle affiche l'auteur actuel.
10. Si l'utilisateur veut ajouter une référence, la combobox sert d'input pour le titre, s'il sélectionne une référence pour la modifier ou la supprimer, elle affiche le titre actuel.
11. Si l'utilisateur veut ajouter ou modifier une référence, la picturebox affiche l'image sélectionnée après la pression du bouton *Importer*, s'il sélectionne une référence pour la modifier ou la supprimer, elle affiche l'image actuelle.
12. Ouvre un openfiledialog pour que l'utilisateur puisse sélectionner l'image.
13. Combobox qui contient la liste des livres de l'utilisateur et une donnée qui s'appelle *nouveau livre*. Si l'utilisateur sélectionne cette dernière alors l'application comprend qu'il veut ajouter un livre et donne accès aux input correspondant. Dans le cas contraire, l'application sélectionne le livre demandé.
14. Permet d'ajouter une description au lieu si le type Lieu est sélectionné.
15. Permet d'ajouter la référence. Un message s'affiche en cas d'erreur ou de succès.
16. Permet de modifier la référence. Un message s'affiche en cas d'erreur ou de succès.
17. Permet de supprimer la référence. Un message de confirmation s'affiche. Si l'utilisateur accepte, alors un message s'affiche en cas d'erreur ou de succès.

3.2 Méthodologie

Dans le cadre de ce projet, j'ai utilisé la méthodologie en 6 étapes. Dans le cahier des charges le projet était réparti en quatres étapes (analyse, implémentation, tests et documentation) j'ai donc réparti ces dernières dans la méthode en 6 étapes avec un code couleur.

Tâches à réaliser											
Informer											
Lecture énoncé											
Synthétisation de l'énoncé											
Planifier											
Ajout des tâches dans l'analyse fonctionnelle											
Création du diagramme de Gantt											
Analysé Fonctionnelle											
Mise en place de l'environnement de versions (Github)											
Mise en place du système de backups (Google drive)											
Décider											
Décider des points d'entrées de l'API											
Création du diagramme de la base de données											
Réaliser											
Base de données											
Création de l'utilisateur de base de données											
Ajout des deux utilisateurs dans la table											
API Rest en PHP											
Création des objets PHP répliqués de la base											
Relier la base et l'API en PDO (singleton)											
Création du CRUD dans l'API PHP											
Création des fonctions SELECT											
Création de la connexion											
Système d'hachage du mot de passe											
Retour d'une liste de livres en fonction de l'auteur et du titre											
Retour de collection d'un livre et de l'utilisateur											
Création des fonctions INSERT											
Création des fonctions DELETE											
Création des fonctions UPDATE											
Sécurité (contrôle des GET)											
Sécurité (contrôle du format de données)											
Création des points d'entrée											
Envoie des données en JSON											
Application C# Windows Forms											
Importer les librairies NuGet											
Création des visuels des forms											
Créer les fonctions qui permettent de relier les forms											
Création des objets C# répliqués de la base											
Connexion											
Appel du point d'entrée de connexion											
Passage du résultat json en objet											
En fonction de la réponse http, l'utilisateur a un message d'erreur ou il passe à la form collection de livres											
Collection de livres											
Création de l'objet card											
Affichage des card											
Création dynamique des card avec le retour JSON											
Recherche de card par auteur et titre											
Ajouter un livre											
Modifier un livre											
Supprimer un livre											
Refuser une suppression si le livre est dans une collection de références											
Bouton qui redirige vers les références du livre											
Collection des références											
Affichage du titre du livre											
Affichage des références											
filtrer en fonction du type											
recherche par texte											
ajouter une référence											
modifier une référence											
supprimer une référence avec message de confirmation											
Contrôler											
Tests											
Évaluer											
Finitions											
Générale											
Documentation											
<table border="1" style="margin-left: auto; margin-right: 0;"> <tr> <td>Analyse</td><td>12:00</td></tr> <tr> <td>Implémentation</td><td>50:00</td></tr> <tr> <td>Tests</td><td>04:00</td></tr> <tr> <td>Documentation</td><td>22:00</td></tr> <tr> <td>Total</td><td>88:00</td></tr> </table>		Analyse	12:00	Implémentation	50:00	Tests	04:00	Documentation	22:00	Total	88:00
Analyse	12:00										
Implémentation	50:00										
Tests	04:00										
Documentation	22:00										
Total	88:00										

Planning prévisionnel avec le code couleur

3.2 Base de données

3.2.1 Contraintes

1. Les contraintes d'intégrité ont l'option *on cascade*.
2. InnoDB
3. encodage UTF-8

3.2.2 Diagramme

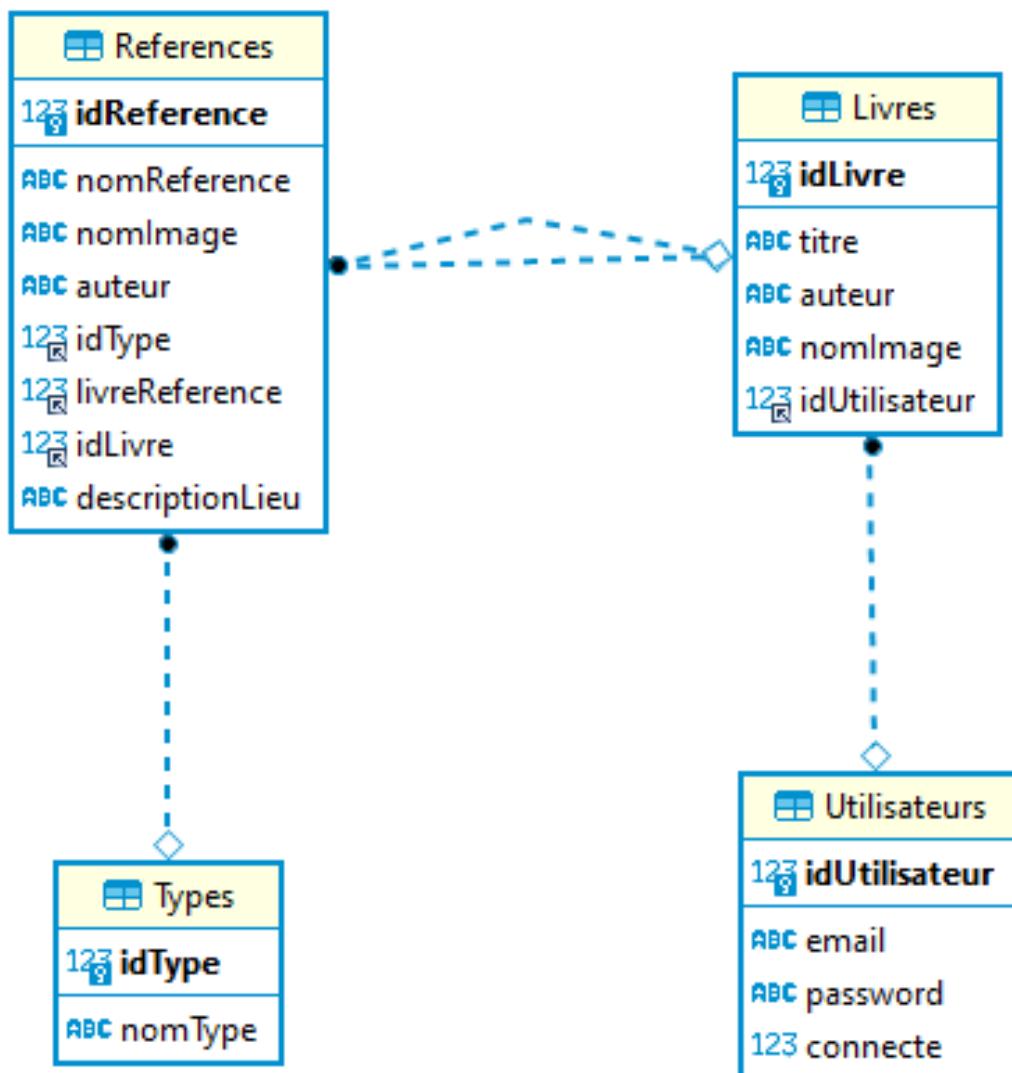


Diagramme de la base de données v1.0

3.2.2 Tables

3.2.2.1 Type

idType	int(11) clé primaire, auto-incrémentation, Non Null
nomType	varchar(255), Non Null chanson, lieu ou livre

3.2.2.2 Référence

idReference	int(11) clé primaire, auto-incrémentation
nomReference	varchar(100), Non Null
nomImage	varchar(255), Null
auteur	varchar(100), Null
idType	int(11), Non Null, clé étrangère de la table Types Référence le type
livreReference	int(11), Null, clé étrangère de la table Livres Sert si la référence est un livre
idLivre	int(11), Non Null, clé étrangère de la table Livres (idLivre) Afin de savoir à quel livre appartient la référence
desctiptionLieu	mediumtext, Null permet d'ajouter une description à un lieu

3.2.2.3 Livres

idLivre	int(11) clé primaire, auto-incrémentation
titre	varchar(100), Non Null
auteur	varchar(255), Non Null
nomImage	varchar(255), Non Null
idUtilisateur	int(11) clé primaire Sert à savoir quel utilisateur ajouté le livre

3.2.2.4 Utilisateurs

idUtilisateur	int(11) clé primaire, auto-incrémentation
email	varchar(255), Non Null
password	varchar(255), Non Null (encrypté en sha1)
connecte	bool, Non Null permet d'indiquer si l'utilisateur est connecté

3.2.3 Sql

```
-- MySQL dump 10.13 Distrib 5.5.62, for Win64 (AMD64)
--
-- Host: localhost Database: MyLibrary
-----
-- Server version 5.5.5-10.3.34-MariaDB-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `Livres`
--
```

```
DROP TABLE IF EXISTS `Livres`;  
/*!40101 SET @saved_cs_client    = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
CREATE TABLE `Livres` (  
  `idLivre` int(11) NOT NULL AUTO_INCREMENT,  
  `titre` varchar(100) NOT NULL,  
  `auteur` varchar(255) NOT NULL,  
  `nomImage` varchar(255) NOT NULL,  
  `idUtilisateur` int(11) NOT NULL,  
  PRIMARY KEY (`idLivre`),  
  KEY `Livres_FK` (`idUtilisateur`),  
  CONSTRAINT `Livres_FK` FOREIGN KEY (`idUtilisateur`) REFERENCES `Utilisateurs`  
(`idUtilisateur`) ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=100 DEFAULT CHARSET=utf8;  
/*!40101 SET character_set_client = @saved_cs_client */;  
  
--  
-- Dumping data for table `Livres`  
--  
  
LOCK TABLES `Livres` WRITE;  
/*!40000 ALTER TABLE `Livres` DISABLE KEYS */;  
INSERT INTO `Livres` VALUES (99,'Les Misérables - Tome 1','Victor  
Hugo','99JOLYBU.png',1);  
/*!40000 ALTER TABLE `Livres` ENABLE KEYS */;  
UNLOCK TABLES;  
  
--  
-- Table structure for table `References`  
--  
  
DROP TABLE IF EXISTS `References`;  
/*!40101 SET @saved_cs_client    = @@character_set_client */;
```

```
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `References` (
  `idReference` int(11) NOT NULL AUTO_INCREMENT,
  `nomReference` varchar(100) DEFAULT NULL,
  `nomImage` varchar(255) DEFAULT NULL,
  `auteur` varchar(100) DEFAULT NULL,
  `idType` int(11) NOT NULL,
  `livreReference` int(11) DEFAULT NULL,
  `idLivre` int(11) DEFAULT NULL,
  `descriptionLieu` mediumtext DEFAULT NULL,
  PRIMARY KEY (`idReference`),
  KEY `References_FK` (`idType`),
  KEY `References_FK_2` (`livreReference`),
  KEY `References_FK_3` (`idLivre`),
  CONSTRAINT `References_FK` FOREIGN KEY (`idType`) REFERENCES `Types`(`idType`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `References_FK_2` FOREIGN KEY (`livreReference`) REFERENCES `Livres`(`idLivre`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `References_FK_3` FOREIGN KEY (`idLivre`) REFERENCES `Livres`(`idLivre`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=96 DEFAULT CHARSET=utf8;

/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Dumping data for table `References`
-- 

LOCK TABLES `References` WRITE;

/*!40000 ALTER TABLE `References` DISABLE KEYS */;
INSERT INTO `References` VALUES (95,'Les Quatre Saisons','7PLPH5GJ.png','Antonio Vivaldi',2,NULL,99,NULL);
/*!40000 ALTER TABLE `References` ENABLE KEYS */;

UNLOCK TABLES;
```

```
--  
-- Table structure for table `Types`  
--  
  
DROP TABLE IF EXISTS `Types`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
CREATE TABLE `Types` (  
  `idType` int(11) NOT NULL AUTO_INCREMENT,  
  `nomType` varchar(100) NOT NULL,  
  PRIMARY KEY (`idType`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;  
/*!40101 SET character_set_client = @saved_cs_client */;  
  
--  
-- Dumping data for table `Types`  
--  
  
LOCK TABLES `Types` WRITE;  
/*!40000 ALTER TABLE `Types` DISABLE KEYS */;  
INSERT INTO `Types` VALUES (1,'livre'),(2,'musique'),(3,'lieu');  
/*!40000 ALTER TABLE `Types` ENABLE KEYS */;  
UNLOCK TABLES;  
  
--  
-- Table structure for table `Utilisateurs`  
--  
  
DROP TABLE IF EXISTS `Utilisateurs`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
CREATE TABLE `Utilisateurs` (
```

```
`idUtilisateur` int(11) NOT NULL AUTO_INCREMENT,  
 `email` varchar(255) NOT NULL,  
 `password` varchar(255) NOT NULL,  
 `connecte` tinyint(1) NOT NULL DEFAULT 0,  
 PRIMARY KEY (`idUtilisateur`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;  
/*!40101 SET character_set_client = @saved_cs_client */;  
  
--  
-- Dumping data for table `Utilisateurs`  
--  
  
LOCK TABLES `Utilisateurs` WRITE;  
/*!40000 ALTER TABLE `Utilisateurs` DISABLE KEYS */;  
INSERT INTO `Utilisateurs` VALUES  
(1,'karel.svbd@eduge.ch','f6889fc97e14b42dec11a8c183ea791c5465b658',0),(2,'ami  
r.yns@eduge.ch','9189e50bd7408fbf02cf59bd58a4776351bdbb72',0);  
/*!40000 ALTER TABLE `Utilisateurs` ENABLE KEYS */;  
UNLOCK TABLES;  
  
--  
-- Dumping routines for database 'MyLibrary'  
--  
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;  
  
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;  
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;  
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;  
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;  
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

3.2.3 Utilisateur de la base

Nom d'utilisateur	Mot de passe
MyLibraryAPI	r6.4>NRM`tC~y*gN

Cet utilisateur possède uniquement les droits sur la base de données *MyLibrary*

```
CREATE USER 'MyLibraryAPI'@'%' IDENTIFIED BY 'r6.4>NRM`tC~y*gN';
GRANT Alter ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Create ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Create view ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Delete ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Delete history ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Drop ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Grant option ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Index ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Insert ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT References ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Select ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Show view ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Trigger ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Update ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Alter routine ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Create routine ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Create temporary tables ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Execute ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Lock tables ON MyLibrary.* TO 'MyLibraryAPI'@'%';
GRANT Grant option ON MyLibrary.* TO 'MyLibraryAPI'@'%';
```

Script SQL de l'utilisateur

4.0 Analyse organique

4.1 Architecture du projets

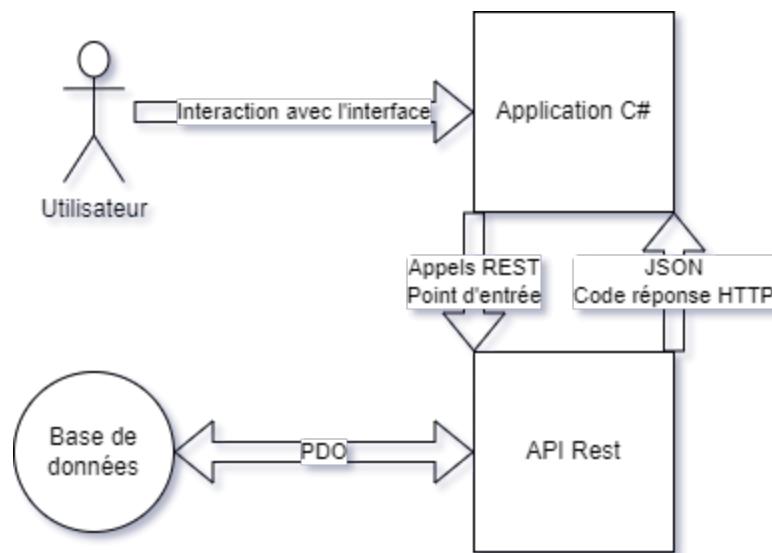
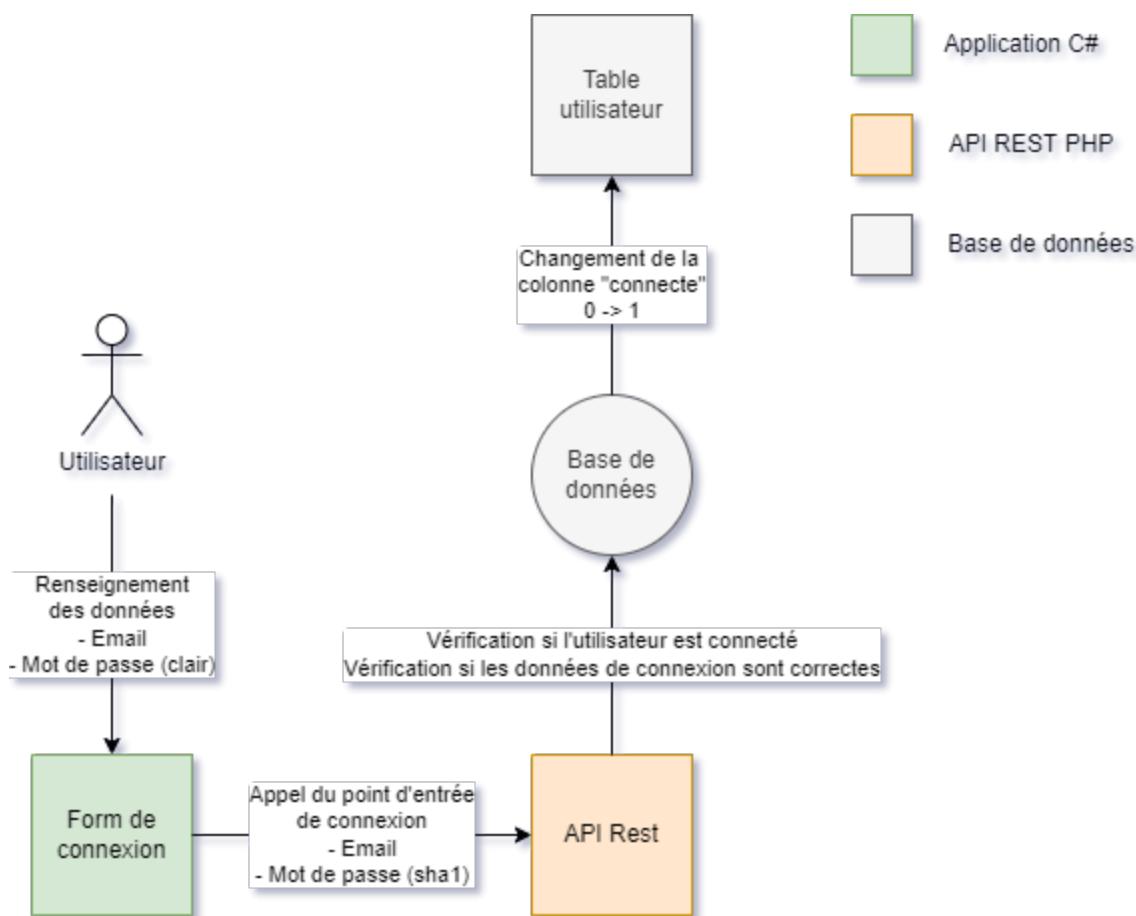


Figure 1 : Diagramme simplifié du projet

4.2 Système de sécurité de l'API

Pour utiliser les points d'entrées de l'API il faut commencer par se connecter. Puis, il faut transmettre à chaque appel l'email et le mot de passe (chiffré en sha1). Une seule connexion simultanée est possible. Voici un diagramme qui explique le procédé.



4.3 Connexion

Pour se connecter l'utilisateur utilise la frmConnexion puis crée une instance de l'objet utilisateur. Ce dernier appelle sa méthode TestConnexion qui fait une demande à l'API via la méthode ApiRequest de la class ClientRest. En fonction de la réponse, l'utilisateur sera redirigé vers la formCollectionLivres ou aura un message d'erreur qui s'affiche.

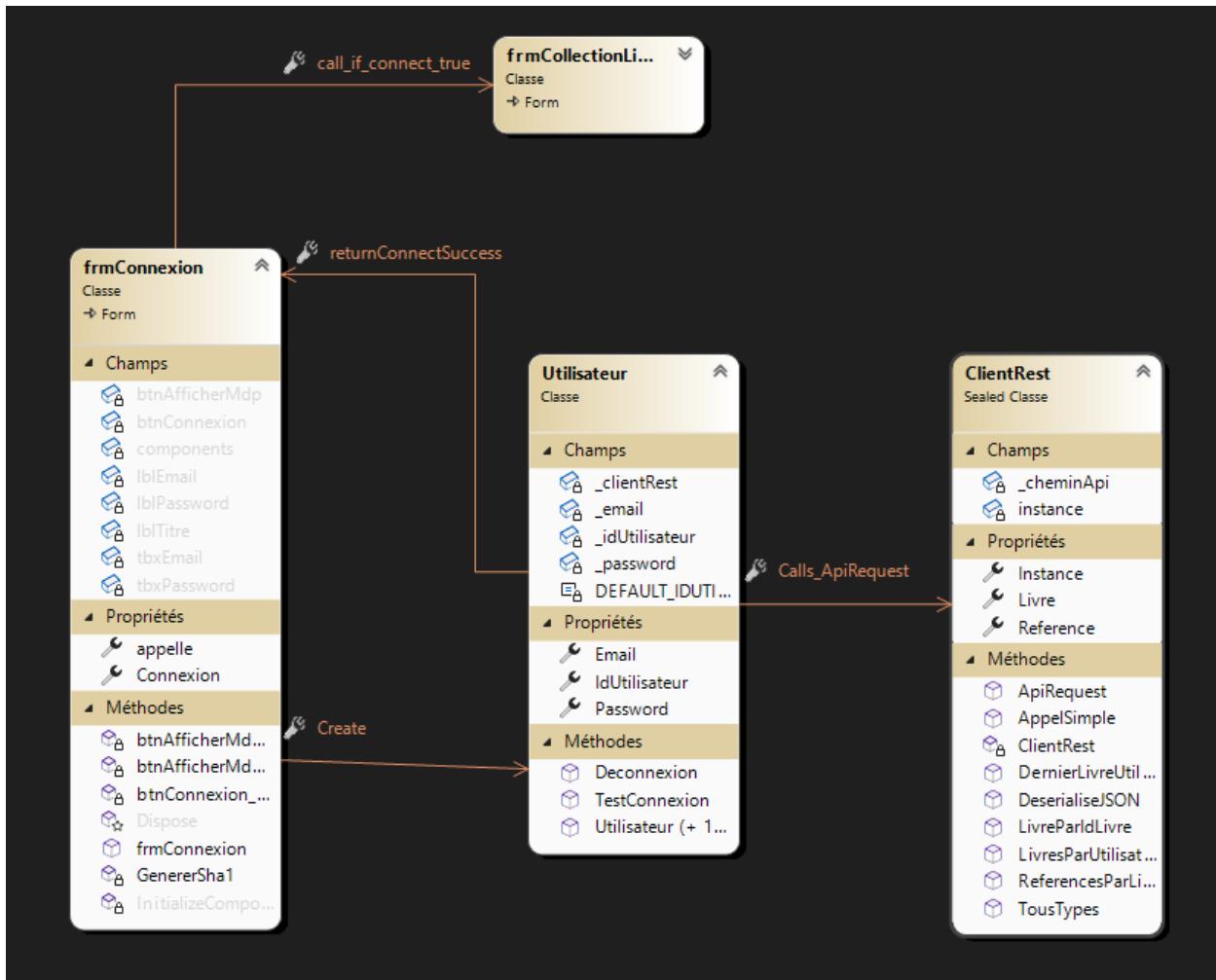


Diagramme de connexion

4.X Héritage des Card

Les Card sont les composants qui affichent les données. Pour une meilleure uniformisation, ces composants sont tous hérités de la classe abstraite card qui hérite elle-même du composant Windows Forms Panel.

Pour afficher les livres dans la frmCollectionLivres, on utilise des instances de la class CardLivre. Avec cela on ajoute une instance de la class Livre afin de stocker les éléments.

Pour afficher les référence j'ai créé une class abstraite CardReference et trois classes hérités qui sont générés en fonction du type de référence. Les données sont stockées dans une instance de la class référence. Le système d'héritage des class références est expliqué dans le chapitre suivant.

Pour la CardReferenceLieu nous sommes dans l'obligation de créer une instance de la class Livre pour stocker les données.

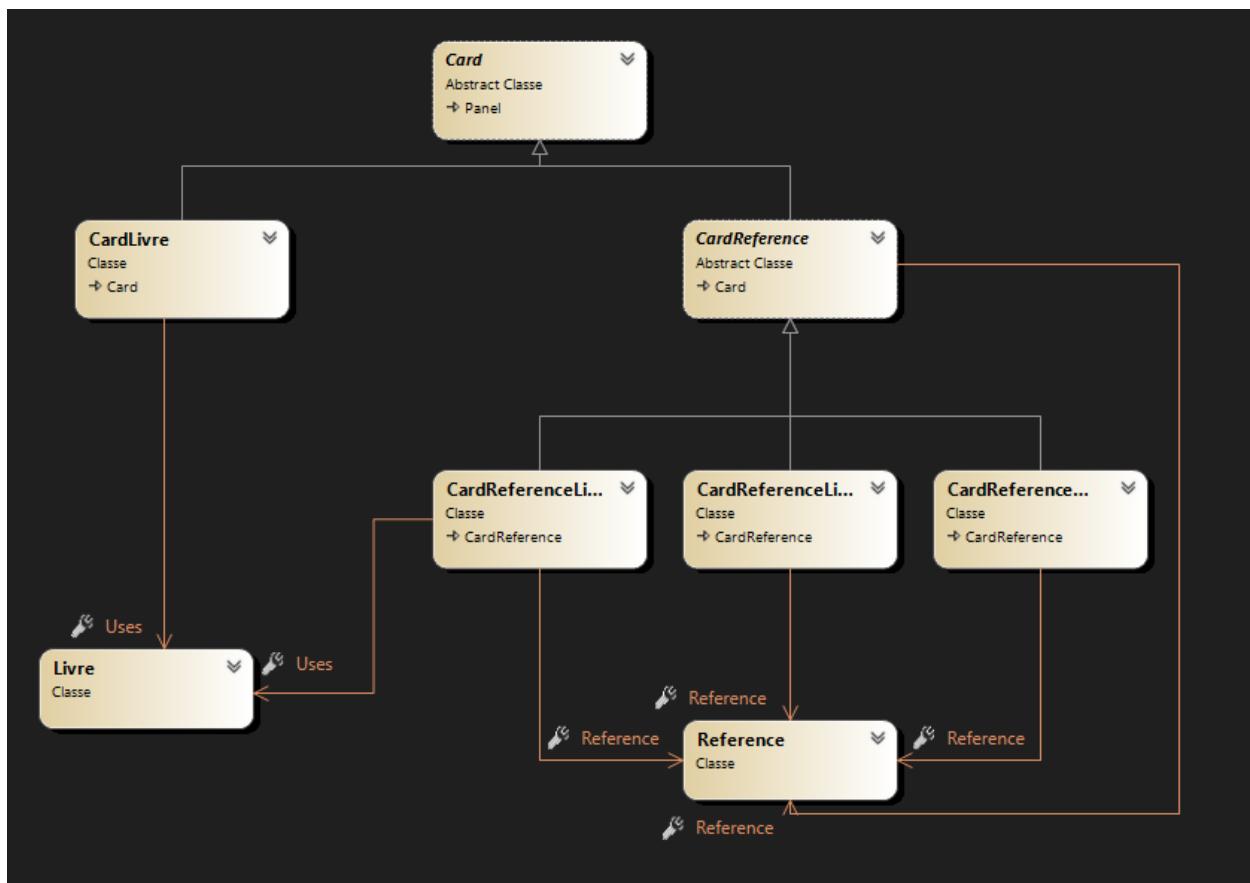


Diagramme d'héritage des card

4.X Héritage des références

Le système a été créé pour mieux gérer les requêtes à l'API car chaque class appelle la classe ClientRest.ApiRequest ou ClientRest.AppelSimple avec ses propres données.

Methode ClientRest.ReferenceParLivre() : Lors de la récupération des références, on crée dynamiquement des objets en fonction du type puis on retourne une liste avec toutes les références.

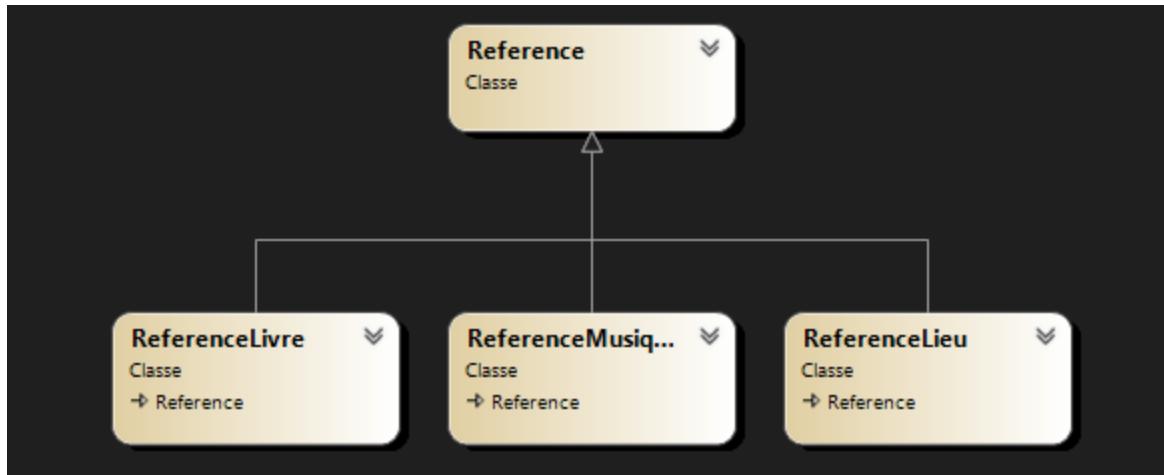


Diagramme de classe des références

```

1 référence
public List<Reference> ReferencesParLivre(Utilisateur utilisateur, Livre livre)
{
    List<Reference> referencesParLivre = new List<Reference>();
    dynamic referencesDynamic = DeserialiseJSON(ApiRequest("?email=" + utilisateur.Email + "&password="));
    foreach (var element in referencesDynamic)
    {
        switch (Convert.ToInt32(element["idType"]))
        {
            case 1:
                referencesParLivre.Add(new ReferenceLivre(Convert.ToInt32(element["idReference"])), Convert.ToInt32(element["idLivre"]));
                break;
            case 2:
                referencesParLivre.Add(new ReferenceMusique(Convert.ToInt32(element["idReference"])), Convert.ToInt32(element["idLivre"]));
                break;
            case 3:
                if(element["descriptionLieu"] == null)
                {
                    referencesParLivre.Add(new ReferenceLieu(Convert.ToInt32(element["idReference"])), Convert.ToInt32(element["idLivre"]));
                }
                else
                {
                    referencesParLivre.Add(new ReferenceLieu(Convert.ToInt32(element["idReference"])), Convert.ToInt32(element["idLivre"]));
                }
                break;
        }
    }
    return referencesParLivre;
}
  
```

Génération dynamique des références en fonction du type

4.3 Technologies utilisées

Technologie	Information
Système d'exploitation	Windows 10 Éducation
Serveur de développement	sous système Ubuntu 20.04 en wsl2
Serveur web	Apache 2.4.41
Serveur base de données	mysql Ver 15.1 Distrib 10.3.34-MariaDB, for debian-linux-gnu (x86_64) using readline 5.2
Logiciel de versions	Github Desktop v2.9.11
Service de backups	Google Drive
Version PHP	PHP 7.4.3
Logiciel de développement de l'application locale	Microsoft Visual Studio Community 2022 (64 bits) - Current
Version C#	4.1.0-5.22165.10+e555772db77ca828b0 2b4bd547c318387f11d01f
Framework	Microsoft .NET Framework Version 4.8.04084
Logiciel d'édition de fichier du serveur	Visual Studio Code v1.66.2
Logiciel de débogage de l'API	Postman v9.16.0
Logiciel de base de données	DBeaver v22.0.0.202203060510
Logiciel de création de diagramme	draw.io v17.4.6
Planification	Microsoft Excel
Documentation Technique	Google docs
Manuel utilisateur	Google docs
Journal de bord	Markdown
Documentation C#	Extension visual studio VSdocman

5.0 Interfaces

5.1 Maquettes de l'application C#

5.1.1 Connexion

My Library

Email :
karel.svbd@eduge.ch

Password :

Connect

201 → Form collection de livres

401 → Les données de connexion sont fausses

Maquette connexion

5.1.2 Collection de livres

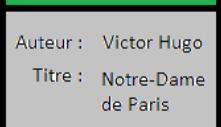
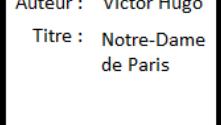
collection de livres

recherche : auteur

Auteur : Victor Hugo Titre : Notre-Dame de Paris Collection	Auteur : Victor Hugo Titre : Notre-Dame de Paris Collection	Auteur : Victor Hugo Titre : Notre-Dame de Paris Collection
Auteur : Victor Hugo Titre : Notre-Dame de Paris Collection	Auteur : Victor Hugo Titre : Notre-Dame de Paris Collection	Auteur : Victor Hugo Titre : Notre-Dame de Paris Collection

Auteur :
Titre :
Image :
Ajouter Modifier
Supprimer

Collection de livres sans sélection

collection de livres	
recherche :	<input type="text" value="auteur"/> 
	Auteur : Victor Hugo Titre : Notre-Dame de Paris <input type="button" value="Collection"/>
	Auteur : Victor Hugo Titre : Notre-Dame de Paris <input type="button" value="Collection"/>
	Auteur : Victor Hugo Titre : Notre-Dame de Paris <input type="button" value="Collection"/>
	Auteur : Victor Hugo Titre : Notre-Dame de Paris <input type="button" value="Collection"/>
	Auteur : Victor Hugo Titre : Notre-Dame de Paris <input type="button" value="Collection"/>
Auteur :	<input type="text" value="Victor Hugo"/>
Titre :	<input type="text" value="Notre-Dame de Paris"/>
Image :	<input type="text" value="img.png"/>
	<input type="button" value="Ajouter"/> <input type="button" value="Modifier"/>
	<input type="button" value="Supprimer"/>

Collection de livres avec sélection

5.1.3 Collection de références

Collection de références X

Livre : Vendredi ou les Limbes du Pacifique

Liste des références

Type : Mot clé :



Auteur : Victor Hugo
Titre : Notre-Dame de Paris



Auteur : Jul
Titre : My World

Titre : Montreux
Desc. : Super lieu où il fait super beau

nom référence :

nom image :

auteur :

livre reference :

Livre

Collection de références

6.0 Architecture du projet

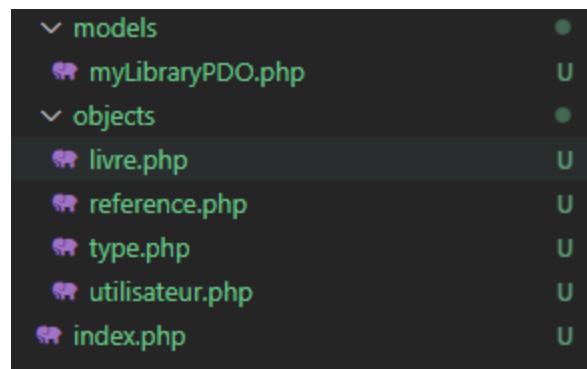
Architecture générale

L'objectif a été d'encapsuler les données aux maximum pour une meilleure organisation. C'est pour cette raison que les deux projets ont une liste d'objets donc les propriétés sont identiques aux tables dans la base.

API PHP

Les classes d'encapsulation des tables de la base sont sous le dossier objects. Le dossier modèle contient le fichier myLibraryPDO.php qui fait la liaison avec la base de données.

Le fichier index.php est celui que l'utilisateur appelle quand il veut utiliser l'API.



Les dossiers sont mis en commun dans le fichier index.php grâce à des fonctions require.

```
//Connexion PDO
require("models/myLibraryPDO.php");
$mydatabase = new MyLibrary();

//importation des objets
require("objects/livre.php");
require("objects/reference.php");
require("objects/type.php");
require("objects/utilisateur.php");
```

Passage des données

Ce diagramme explique comment les données sont traitées dans l'application.

1. L'utilisateur appelle un point d'entrée
2. Un objet est créé à partir des headers passés en get
3. On appelle la fonction avec les données encapsulés en paramètres
4. On passe les données dans les prepared statements grâce aux bindParam.
5. On exécute la requête grâce au PDO
6. Le résultat passe par la variable d'instance
7. Le résultat passe par la variable d'instance
8. Le résultat passe par le return de la fonction
9. Le résultat est renvoyé grâce au JSON et aux réponses Http

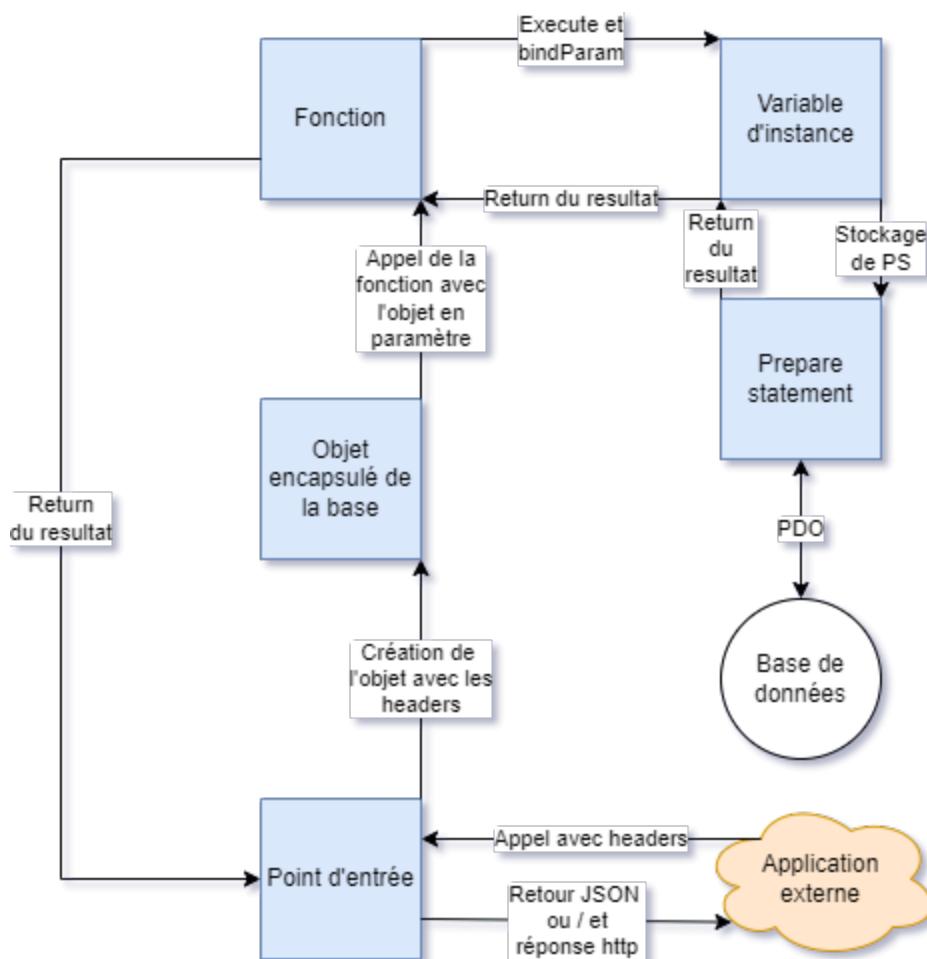


Diagramme de passage de données

Api Rest

MyLibrary

Cette class sert à communiquer avec la base de données. Elle utilise la technologie PDO et des prepare statements pour l'optimisation. Le résumé du fonctionnement de cette classe est résumé dans le chapitre passage de données.

index.php

C'est par cette page que l'utilisateur passe ses requêtes. Les points d'entrées sont séparés par méthodes de requête.

- GET pour recevoir
- POST pour envoyer
- PUT pour modifier
- DELETE pour supprimer

Class encapsulées Livre, Reference, Type et Utilisateur

Toutes ces classes ont en commun qu'elles ont comme objectif d'encapsuler les données de la base de données. Elles ont également toutes la fonction return ArrayForJSON() qui sert à renvoyer un array formaté avec les données de l'objet. La class Livre possède la fonction ReferenceDeLivre en plus qui permet de créer une référence à partir d'un livre.

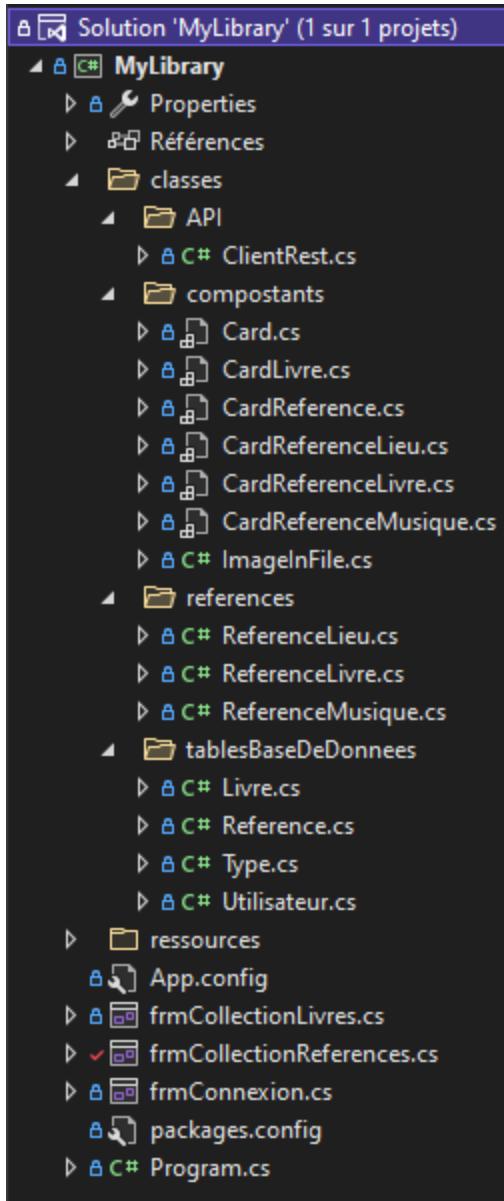
```
//Fonctions

/** Permet de retourner un array avec les informations de l'objet
 *
 * @return array array formé pour JSON
 */
public function returnArrayForJSON(){
    return array(
        "idLivre" => $this->_idLivre,
        "titre" => $this->_titre,
        "auteur" => $this->_auteur,
        "nomImage" => $this->_nomImage,
        "idUtilisateur" => $this->_idUtilisateur
    );
}

public function ReferenceDeLivre(Livre $livre){
    return new Reference(0, "", "", "", 1, $this->_idLivre, $livre->getIdLivre(), "");
}
```

Class Livre avec les fonction returnArrayForJSON et ReferenceDeLivre

Application C#



L'application possède trois forms

- frmCollectionLivres.cs
- frmCollectionReferences.cs
- frmConnexion.cs

Le dossier API contient le fichier qui communique avec l'API Rest.

- ClientRest.cs

Le dossier références répertorie les class qui héritent de références.

- ReferenceLieu.cs
- ReferenceLivres.cs
- ReferenceMusique.cs

Le dossier tablesBaseDeDonnées continent les class qui encapsulent les tables de la base de données.

- Livre.cs -> MyLibrary.Livres
- Reference.cs -> MyLibrary.References
- Type.cs -> MyLibrary.Types
- Utilisateurs.cs -> MyLibrary.Utilisateurs

L'appel des des points d'entrées

Les commandes de type post, put et delete sont directement intégrées à l'objet afin qu'un simple appel de la méthode permette de récupérer les données et les envoyer à l'API. Les commandes de type get sont gérés par la class ClientRest.cs

Class ClientRest

Classe en singleton gérant les appels à l'API

Dans l'application il existe deux class pour les appels. La première, ApiRequest(), sert à appeler l'API avec un retour JSON. Contrairement à AppelSimple qui contrôle simplement la réponse Http. Si il est différent de 2XX, elle retourne false.

Nom	Fonction
Champs	
_cheminAPI	Chemin de l'API sans headers
instance	Stocke l'instance
Propriétés	
Instance	Sert de point d'entrée à la classe
Méthodes	
ApiRequest	Permet de faire une requête à une API et retourne un string
AppelSimple	Permet de faire une requête à une API et retourne un booléen
ClientRest	Constructeur fermé
DernierLivreUtilisateur	retourne le dernier livre de l'utilisateur
DeserialiseJSON	Déserialise le Json en le passant en dynamic
LivresParUtilisateur	Classe polymorphe <ul style="list-style-type: none"> - permet de retourner tous les livres d'un utilisateur - permet de retourner les livres d'un utilisateur avec la recherche personnalisée
ReferencesParLivre	Permet de retourner toutes les références d'un livre et de les trier en fonction de leur type
TousTypes	Retourne tous les types de la table Types de la base

Class Card

Cette class abstraite est un composant qui hérite de System.Windows.Form.Panel elle sert de modèle.

Nom	Fonction
Champs	
taille	Permet de définir la taille de la card
Fonctions	
Card	Constructeur
ClickCard	S'enclenche après un click

Class CardLivre

Class qui hérite de Card, sert à afficher un livre

Nom	Fonction
Champs	
_image	élément picturebox qui stocke l'image
_livre	instance de la class Livre, stocke les données
_livres	instance de la class frmCollectionLivres
Propriétés	
ObjLivre	Sert à accéder au champs _livre
Méthodes	
CardLivre	Constructeur polymorphe - Sans données il utilise des données par défaut - Livre et frmCollectionLivres Affiche le titre et les auteurs dans des labels Recherche l'image dans le fichier et l'affiche dans une picturebox Affiche une bouton afin de pouvoir accéder au références du livre

ClickCard	Appelle la méthode SelectionnerCard dans la frmCollectionLivres avec les données de la card
ClickReference	S'active après la pression du bouton reference

CardReference

Class abstraite qui hérite de Card, sert de modèles aux class des cardReference

Nom	Fonction
Champs	
_frm	stocke la frmCollectionLivre
_reference	stocke les données de la référence
Propriétés	
Frm	Donne accès à _frm
ObjReference	Donne accès à _reference
Méthodes	
CardReference	Ajoute l'événement ClickCard et stocke les valeurs dans les champs

CardReferenceLieu

Class qui hérite de cardReference, sert à afficher les référence de type lieu.

Nom	Fonction
Methodes	
CardReferenceLieu	Stocke les valeurs quand les champs, ajoute les événem click au éléments
ClickCard	Appelle la méthode SelectionnerCard

CardReferenceLivre

Class abstraite qui hérite de CardReference, sert à afficher les card de type ReferenceLivre

Nom	Fonction
Champs	
_image	Picturebox de l'élément
_livre	Stocke les information du livre
Propriétés	
ObjLivre	Permet d'interagir avec les informations du livre
Méthodes	
CardReferenceLieu	Constructeur polymorphe <ul style="list-style-type: none"> - Si on appelle le constructeur avec un objet livre, la card est créée avec les données de ce dernier. - Si on appelle le constructeur avec une referenceLivre, on va rechercher un livre en fonction de son id (livreReference) et on appelle le constructeur précédent.
ClickCard	Appelle la méthode SeleccionnerCard

CardReferenceMusique

Class abstraite qui hérite de CardReference, sert à afficher les card de type ReferenceMusique

Nom	Fonction
Champs	
_image	Picturebox de l'élément
Propriétés	
ReferenceMusique	Permet d'interagir avec les information

	de la card
Méthodes	
CardReferenceMusique	Crée une cardMusique avec les données de la d'une instance de classe Reference
ClickCard	Appelle la methode SeleccionnerCard

Class ImageInFiles.cs

Permet de stocker une image en Bitmap avec des informations supplémentaires.

Nom	Fonctions
Champs	
_data	Bitmap de l'image
_extension	nom de l'extension (.png ou .jpg par exemple)
_nom	nom de l'image
DEFAULT_EXTENSION	extension par défaut
Propriétés	
Data	permet d'accéder à _data
Extension	permet d'accéder à _extension
Nom	permet d'accéder à _nom
Méthodes	
ImageInFiles	Constructeur polymorphe <ul style="list-style-type: none"> - Si on lui transmet un nom, une extension et une bitmap. On stocke simplement les éléments - Si on ne lui transmet pas de nom, l'application crée un nom aléatoire - Si on lui transmet uniquement une Bitmap alors l'application donne l'extension par défaut

	.png et crée un nom aléatoire
NomAleatoire	Crée une chaîne de 8 caractères aléatoires.
SaveBmp	permet d'enregistrer l'image dans le fichier de la solution

Class ReferenceLieu

Permet de stocker une référence de type lieu

Nom	Fonction
Champs	
DEFAULT_DESCRIPTION	Description si aucune n'est passée en paramètre.
Méthodes	
PostReference	Permet d'ajouter la référence dans la base de données
PutReference	Permet de modifier la référence dans la base de données
ReferenceLieu	Constructeur polymorphe - Si on transmet aucune description, on crée l'instance avec une description par défaut

Class ReferenceLivre

Permet de stocker une référence de type livre

Nom	Fonction
Méthodes	
PostReference	Permet d'ajouter la référence dans la base de données
PutLivre	Permet d'envoyer un livre à la base
PutReference	Permet de modifier la référence dans la

	base de données
ReferenceLivre	Permet de créer une instance de type Reference Livre

Class ReferenceMusique

Permet de stocker une référence de type musique

Nom	Fonction
Méthodes	
PostReference	Permet d'ajouter la référence dans la base de données
PutReference	Permet de modifier la référence dans la base de données
ReferenceMusique	Permet de créer une instance d'un objet de type referenceMusique

Class Livre

Permet d'encapsuler la table Livres dans la base de données

Nom	
Champs	
_auteur	Permet de stocker l'auteur - varchar(100)
_clientRest	Permet de stocker une instance de la classe ClientRest
_idLivre	Permet de stocker un id de livre - int(11)
_idUtilisateur	Permet de stocker un id d'utilisateur - int(11)
_nomImage	Permet de stocker un nom d'image - varchar(255)
_titre	Permet de stocker un titre

	- varchar(255)
Propriétés	
Auteur	Permet d'accéder à _auteur
IdLivre	Permet d'accéder à _idLivre
IdUtilisateur	permet d'accéder à _idUtilisateur
NomImage	permet d'accéder à _nomImage
Titre	permet d'accéder à _titre
Méthodes	
DeleteLivre	Permet de supprimer le livre de la base de données
Livre	Permet de créer un instance d'objet de type Livre
PostLivre	Permet d'envoyer le livre à la base de données
PutLivre	Permet de modifier le livre dans la base. Mettre les nouvelles données dans le paramètre livre.

Class Reference

Permet d'encapsuler la table References dans la base de données

Nom	
Champs	
_auteur	Permet de stocker l'auteur - int(100)
_descriptionLieu	Permet de stocker la description du lieu - À transmettre uniquement si le type est Lieu - mediumtext
_idLivre	Permet de stocker l'id du livre dont la référence est attachée

	- int(11)
_idReference	Permet de stocker l'id de la référence - int(11)
_idType	Permet de stocker le type de la référence 1. Livre 2. Musique 3. Lieu - int(11)
_livreReference	Si le type est 1 (livre) alors il faut rattacher le livre de référence dans la table Livres - int(11)
_nomImage	Stocke le nom de l'image - int(11)
_nomReference	Stocke le nom de la référence - varchar(255)
Propriétés	
Auteur	permet d'accéder à _auteur
DescriptionLieu	permet d'accéder à _descriptionLieu
IdLivre	permet d'accéder à _idLivre
IdReference	permet d'accéder à _idReference
IdType	permet d'accéder à _idType
LivreReference	permet d'accéder à _livreReference
IdType	permet d'accéder à _idType
NomImage	permet d'accéder à _nomImage
NomReference	permet d'accéder à _nomReference
Méthodes	
DeleteReference	Permet de supprimer la référence dans la base par l'API
PostReference	(Non implémentée, sert de modèle)

PutReference	(Non implémentée, sert de modèle)
Reference	Permet de créer un instance d'objet de type Reference

Class Type

Permet d'encapsuler la table Types dans la base de données

Nom	
Champs	
_idType	Permet de stocker l'id du type - int(11)
_nomType	Permet de stocker le nom du type - varchar(255)
Propriétés	
IdType	Permet d'accéder à _idType
NomType	Permet d'accéder à _nomType
Méthodes	
Type	Permet de créer une instance de type Type

Class Utilisateur

Permet d'encapsuler la table Utilisateur dans la base de données

Nom	
Champs	
_clientRest	Instance de la class ClientRest
_email	Permet de stocker l'email de l'utilisateur - varchar(255)
_idUtilisateur	Permet de stocker l'id de l'utilisateur - int(11)
_password	Permet de stocker le mot de passe de

	l'utilisateur - varchar(255)
DEFAULT_IDUTILISATEUR	Permet de donner un id d'utilisateur par défaut
Propriétés	
Email	Permet d'accéder à _email
IdUtilisateur	Permet d'accéder à _idUtilisateur
Password	Permet d'accéder à _password
Méthodes	
Deconnexion	Permet à de faire un appel de déconnexion à l'API
TestConnexion	Permet de faire une tentative de connexion à l'API.
Utilisateur	Permet de créer une instance de l'objet utilisateur

frmConnexion

Sert de login à l'utilisateur, deux textbox sont présentes pour renseigner les données.

frmConnexion Class

Namespace: [MyLibrary](#)

Assembly: WindowsFormsApp1 (in WindowsFormsApp1.exe)

VB	C#	C++	JScript	Copy
<pre>public class frmConnexion : Form</pre>				

Constructors

Name	Description
 frmConnexion()	
Top	

Methods

Name	Description
Dispose(bool)	Nettoyage des ressources utilisées.
GenererSha1(string)	Permet de créer un sha1 à partir d'un string
InitializeComponent()	Méthode requise pour la prise en charge du concepteur - ne modifiez pas le contenu de cette méthode avec l'éditeur de code.
btnAfficherMdp_MouseDown(object, MouseEventArgs)	
btnAfficherMdp_MouseUp(object, MouseEventArgs)	
btnConnexion_Click(object, EventArgs)	

Fields

Name	Description
btnAfficherMdp	
btnConnexion	
components	Variable nécessaire au concepteur.
IblEmail	
IblPassword	
IblTitre	
tbxEmail	
tbxPassword	

Top

FrmCollectionLivres

Permet d'afficher les livres des utilisateurs dans une liste. Un crud peut être effectué sur ces dernières grâce à un formulaire. Il est possible de les filtrer en fonction d'une recherche personnalisée.

frmCollectionLivres Class

Namespace: MyLibrary

Assembly: WindowsFormsApp1 (in WindowsFormsApp1.exe)

```
VB C# C++ JScript Copy
public class frmCollectionLivres : Form
```

Constructors

	Name	Description
⌚	frmCollectionLivres(Utilisateur, frmConnexion)	Form de collection de livres Affiche les livres de l'utilisateur dans une liste d'objets. Permet de faire un CRUD sur les livres via un formulaire

Properties

	Name	Description
🔧	Appelle	
🔧	CardLivre	
🔧	ClientRest	
Top		

◀ Methods

Name	Description
⌚ AfficherReference(Livre)	
⌚ * Dispose(bool)	Clean up any resources being used.
⌚ InitializeComponent()	Required method for Designer support - do not modify the contents of this method with the code editor.
⌚ RefreshView()	Permet de mettre à jour les données de l'application sans filtre
⌚ RefreshView(string)	Permet de mettre à jour les données de l'application avec un filtre
⌚ SelectionnerCard(CardLivre)	Permet de sélectionner une card et de stocker ses données
⌚ VerificationInputs()	Vérification si les inputs sont remplis
⌚ VueParDefault()	Permet de vider la carte sélectionnée et de remettre à zéro la vue de l'application
⌚ btnAjouter_Click(object, EventArgs)	Ajout d'un livre Source du code de l'importation de l'image : https://www.codeproject.com/Questions/546631/howplustoplussavepluspictureboxplusimageplusinplus
⌚ btnImporterImage_Click(object, EventArgs)	Pression du bouton d'importation d'image Source du code d'importation et transformation en Bitmap des images https://stackoverflow.com/questions/6122984/load-a-bitmap-image-into-windows-forms-using-open-file-dialog
⌚ btnModifier_Click(object, EventArgs)	Modification d'image suite à la pression du bouton
⌚ btnSupprimer_Click(object, EventArgs)	Pression du bouton de suppression
⌚ flpListCard_Click(object, EventArgs)	Mise à zéro de la vue si l'utilisateur clique sur flpListCard
⌚ frmCollectionLivres_Click(object, EventArgs)	S'enclanche si l'utilisateur click sur la form Mise à zéro des données
⌚ frmCollectionLivres_FormClosed(object, FormClosedEventArgs)	S'active lors de la fermeture de la form
⌚ tbxRecherche_TextChanged(object, EventArgs)	S'enclanche lors d'un changement de texte de la tbxRecherche

FrmCollectionReferences

Permet d'afficher les références d'un livre dans une liste. Un crud peut être effectué sur ces dernières grâce à un formulaire. Il est possible de les trier par type ou par une recherche personnalisée.

frmCollectionReferences Class

Namespace: WindowsFormsApp1

Assembly: WindowsFormsApp1 (in WindowsFormsApp1.exe)

VB C# C++ JScript

Copy

```
public class frmCollectionReferences : Form
```

◀ Constructors

Name	Description
⌚ frmCollectionReferences(Livre, Utilisateur, frmCollectionLivres)	Vue

◀ Properties

Name	Description
⌚ ObjLivre	
Top	

◀ Methods

	Name	Description
④ *	Dispose(bool)	Clean up any resources being used.
④	EtatTousElements(bool)	Mise à jour de l'états des inputs
④	InitializeComponent()	Required method for Designer support - do not modify the contents of this method with the code editor.
④	InputParDefault()	
④	RechercheReferenceParFiltre(int)	
④	RechercheReferenceParFiltre(string)	
④	RechercheReferenceParFiltre(string, int)	
④	SelectionCard()	
④	SelectionCard(CardReference)	Se déclache lorsqu'une carte est sélectionnée Sert à récupérer les données de la carte
④	UpdateFormView()	Mise à jour des éléments de la vue
④	btnAjouter_Click(object, EventArgs)	
④	btnImporterImage_Click(object, EventArgs)	
④	btnModifier_Click(object, EventArgs)	
④	btnSupprimer_Click(object, EventArgs)	Essaie de supprimer une référence affiche des popups en conséquence
④	cbxFiltreType_SelectedIndexChanged(object, EventArgs)	
④	cbxLivre_SelectedIndexChanged(object, EventArgs)	
④	cbxType_SelectedIndexChanged(object, EventArgs)	
④	frmCollectionReferences_Click(object, EventArgs)	
④	majCbxTypes(List<Type>)	
④	tbxRecherche_TextChanged(object, EventArgs)	
④	textBox1_TextChanged(object, EventArgs)	

7.0 Tests

7.1 Scénario de tests

2.6.1 API

Titre	Connexion à l'API
Résultat attendu	<ul style="list-style-type: none"> 3. Requête de type GET 4. En cas de succès, la données de l'utilisateur <i>connecte</i> dans la base de données change de 0 à 1
Résultat obtenu	Fonctionnel

Titre	Connexion unique
Résultat attendu	Lors de la connexion, l'API vérifie la donnée <i>connecte</i> dans la base de donnée et autorise un nouvelle connexion ou pas en conséquence
Résultat obtenu	Fonctionnel

Titre	CRUD sur la table Livres
Résultat attendu	<ul style="list-style-type: none"> 5. Pouvoir afficher les livres 6. Pouvoir ajouter un livre 7. Pouvoir modifier un livre 8. Pouvoir supprimer un livre
Résultat obtenu	Fonctionnel

Titre	Récupération dynamique de livres
Résultat attendu	Pouvoir récupérer un livre par <ul style="list-style-type: none"> 4. Son auteur 5. Sa description 6. Par l'utilisateur qu'il l'a ajouté
Résultat obtenu	Fonctionnel

Titre	CRUD sur la table References
Résultat attendu	<ul style="list-style-type: none"> 5. Pouvoir afficher les références 6. Pouvoir ajouter un référence 7. Pouvoir modifier un référence 8. Pouvoir supprimer un référence
Résultat obtenu	Fonctionnel

Titre	Récupération dynamique de références
Résultat attendu	<p>Pouvoir récupérer une référence par</p> <ul style="list-style-type: none"> 4. Son type 5. Du Livre 6. Son texte
Résultat obtenu	Fonctionnel

2.6.1 Application C#

Titre	Connexion
Résultat attendu	<ul style="list-style-type: none"> 3. Si l'utilisateur renseigne les bonnes données, il est redirigé sur la form collection de livres 4. Si les données sont erronées un message d'erreur est renvoyé
Résultat obtenu	Fonctionnel

Titre	Récupération des données lors de la connexion
Résultat attendu	<p>Lors d'une connexion réussie, l'utilisateur est stocké dans une instance objet <i>Utilisateur</i> avec ses informations</p> <ul style="list-style-type: none"> 4. idUtilisateur 5. Email 6. Password (sha1)
Résultat obtenu	Fonctionnel

Titre	Création dynamique des <i>Card</i> de livres
Résultat attendu	<p>4. L'application fait un appel à l'API qui lui retourne l'intégralité des livres ajoutés par l'utilisateur</p> <p>5. Affiche les données dans des <i>Card</i> (Objet Windows forms hérité de Panel) créés dynamiquement.</p> <p>6. Chaque <i>Card</i> est ensuite affichés dans une List</p>
Résultat obtenu	Fonctionnel

Titre	Recherche de <i>Card Livres</i>
Résultat attendu	Permet de faire une recherche dans la liste des <i>Card</i> qui contiennent un :
	<ul style="list-style-type: none"> - Auteur donné - Titre donné
Résultat obtenu	Fonctionnel

Titre	CRUD sur les Livres via les <i>Card</i>
Résultat attendu	Grâce un formulaire, l'application doit
	<p>4. Pouvoir ajouter un livre</p> <p>5. Pouvoir modifier un livre</p> <p>6. Pouvoir supprimer un livre</p>
Résultat obtenu	Fonctionnel

Titre	Ouverture d'une form collection de références à partir d'une <i>Card Livre</i>
Résultat attendu	Suite à l'appui d'un bouton <i>Référence</i> d'une <i>Card Livre</i> une form s'ouvre avec le nom du livre en tant que titre
Résultat obtenu	Fonctionnel

Titre	Récupération des référence d'un livre et affichage des résultats dans des card référence de différents types
Résultat attendu	Récupération des référence d'un livre et création de <i>Card</i> de différentes types 4. Musique 5. Livre 6. Lieu
Résultat obtenu	Fonctionnel

Titre	Filtrage des <i>Card</i> référence
Résultat attendu	Affichage des card demandés par 3. Type 4. Texte
Résultat obtenu	Fonctionnel

Titre	CRUD sur les références via les <i>Card</i>
Résultat attendu	Grâce un formulaire, l'application doit 4. Pouvoir ajouter une références 5. Pouvoir modifier une références 6. Pouvoir supprimer une références b. Message de confirmation lors de la suppression
Résultat obtenu	Fonctionnel

7.2 Evolution des tests

N. Test	J.1	J.2	J.3	J.4	J.5	J.6	J.7	J.8	J.9	J.10	J.11	
1	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	X Fonctionnel
2	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	O Pas complètement fonctionnel
3	X	X	O	✓	✓	✓	✓	✓	✓	✓	✓	✓ Fonctionnel
4	X	X	X	O	O	✓	✓	✓	✓	✓	✓	
5	X	X	X	X	X	O	✓	✓	✓	✓	✓	
6	X	X	X	X	X	✓	✓	✓	✓	✓	✓	
7	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	
8	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	
9	X	X	O	O	O	✓	✓	✓	✓	✓	✓	
10	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	
11	X	X	O	✓	✓	✓	✓	✓	✓	✓	✓	
12	X	X	X	X	✓	O	O	✓	✓	✓	✓	
13	X	X	X	X	X	O	O	O	✓	✓	✓	
14	X	X	X	X	X	O	O	O	✓	✓	✓	
15	X	X	X	X	X	X	O	O	X	X	✓	

8.0 Conclusion

8.1 Difficultés rencontrées

Difficulté	Solution
Impossible de faire des sessions en PHP avec l'application C#	Remplacement du système de session par l'ajout de la colonne <i>connecte</i> dans la table Utilisateurs
Manque de temps pour la documentation	Ajout de l'extension VSdocman à Visual Studio

8.2 Améliorations possibles

Par ordre de priorités, l'application n'est pas entièrement sécurisée. Par exemple, il serait intéressant de filtrer les points d'entrée afin de vérifier la taille ainsi que le type de données.

Pour ajouter de la sécurité, il serait également intéressant de changer le système de connexion. Remplacer le changement de booléen par une génération d'une clé aléatoire qui serait transmise lors d'une connexion réussie et qui serait détruite à la fin de cette dernière.

8.3 Bilan personnel

Lors de ce projet j'ai utilisé des technologies avec lesquelles j'avais l'habitude de travailler. Par conséquent, j'ai eu très peu de mauvaises surprises lors de mon développement.

Cependant, si je devais changer une chose lors de mon développement ça serait l'organisation de mon travail quand j'utilise deux technologies qui se complètent dans un temps limité. Pour être plus concret, quand je développe un projet qui utilise une API, je vais dorénavant d'abord développer le point d'entrée puis tout de suite la fonctionnalité afin d'éviter un surplus de code ou des bugs dû au manque de tests.

9.0 Annexes

9.1 Sources

Image de couverture :

https://cdn.radiofrance.fr/s3/cruiser-production/2020/08/1a66ff4f-16c9-4bb5-ac58-579755aa15c4/838_livre.jpg

10.0 Code source

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : CardLivre.cs CardLivre
9   * Descs.    : Sert de modèles aux card
10  */
11 using System;
12 using System.Windows.Forms;
13 using System.Drawing;
14
15 namespace MyLibrary.classes
16 {
17     abstract public class Card : Panel
18     {
19         public Size taille = new Size(150, 200);
20
21         public Card() : base()
22         {
23             Size = taille;
24             BackColor = Color.White;
25             BorderStyle = BorderStyle.FixedSingle;
26         }
27
28         protected abstract void ClickCard(object o, EventArgs e);
29     }
30 }
31
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : CardLivre.cs CardLivre
9   * Descs.    : Permet de créer une card de type livre
10  */
11
12 using MyLibrary.classes;
13 using System;
14 using System.Drawing;
15 using System.Windows.Forms;
16
17 namespace MyLibrary
18 {
19     public class CardLivre : Card
20     {
21         #region variables d'instances
22         PictureBox _image;
23         private Livre _livre;
24         private frmCollectionLivres _lvres;
25         #endregion
26
27         #region propriétés
28         public Livre ObjLivre
29         {
30             get { return _livre; }
31             set { _livre = value; }
32         }
33         #endregion
34
35         #region constructeurs
36
37         public CardLivre() : this(new Livre(0, "Notre Dame de paris",    ↴
38             "Victor Hugo", "frmConnexion.png", 0), null)
39         {
40         }
41         public CardLivre(Livre livre, frmCollectionLivres lvres) : base ↴
42             ()
43         {
44             _livre = livre;
45             _lvres = lvres;
46
47             _image = new PictureBox();
48             _image.Size = new Size(150, 70);
49             _image.BackColor = Color.LightBlue;
50             try
51             {
52                 _image.Image = Image.FromFile(livre.NomImage);
```

```
52         }
53         catch (Exception ex)
54     {
55         Console.WriteLine(ex.Message);
56     }
57
58     _image.SizeMode = PictureBoxSizeMode.StretchImage;
59
60     Label lbltxtAuteur = new Label();
61     lbltxtAuteur.Text = "Auteur : ";
62     lbltxtAuteur.Location = new Point(Location.X + 2, Location.Y + 80);
63     Label lblAuteur = new Label();
64     lblAuteur.Text = livre.Auteur;
65     lblAuteur.Location = new Point(Location.X + 45, Location.Y + 80);
66
67     Label lbltxtTitre = new Label();
68     lbltxtTitre.Text = "Titre : ";
69     lbltxtTitre.Location = new Point(190, 5);
70     lbltxtTitre.Location = new Point(Location.X + 2, Location.Y + 120);
71     Label lblTitre = new Label();
72     lblTitre.Text = livre.Titre;
73     lblTitre.Location = new Point(Location.X + 45, Location.Y + 120);
74
75     Button btnReference = new Button();
76     btnReference.Text = "Référence";
77     btnReference.Location = new Point(Location.X + 35, Location.Y + 160);
78     btnReference.Width = 80;
79     Click += ClickCard;
80     lbltxtAuteur.Click += ClickCard;
81     lblAuteur.Click += ClickCard;
82     lbltxtTitre.Click += ClickCard;
83     lblTitre.Click += ClickCard;
84     _image.Click += ClickCard;
85     btnReference.Click += ClickReference;
86
87
88     Controls.Add(_image);
89     Controls.Add(lblAuteur);
90     Controls.Add(lbltxtAuteur);
91     Controls.Add(lblTitre);
92     Controls.Add(lbltxtTitre);
93     Controls.Add(btnReference);
94 }
95 #endregion
96
97 #region méthodes
98 protected override void ClickCard(object o, EventArgs e)
99 {
```

```
100         _lvres.SelectionnerCard(this);
101     }
102
103     private void ClickReference(object o, EventArgs e)
104     {
105         _lvres.AfficherReference(_livre);
106
107     }
108     #endregion
109 }
110 }
111
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : CardReference.cs Card
9   * Descs.    : Sert de modèles pour les card des références
10  */
11
12 using MyLibrary.classes;
13 using WindowsFormsApp1;
14
15 namespace MyLibrary
16 {
17     public abstract class CardReference : Card
18     {
19         #region varables d'instance
20         Reference _reference;
21         frmCollectionReferences _frm;
22         #endregion
23
24         #region propriétés
25         public Reference ObjReference
26         {
27             get { return _reference; }
28             set { _reference = value; }
29         }
30
31         protected frmCollectionReferences Frm
32         {
33             get { return _frm; }
34             set { _frm = value; }
35         }
36         #endregion
37
38         #region constructeurs
39         public CardReference(Reference reference,
40                             frmCollectionReferences frm) : base()
41         {
42             _reference = reference;
43             _frm = frm;
44             Click += ClickCard;
45         }
46     }
47 }
48
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : CardReferenceLieu.cs cardReference
9   * Descs.    : Permet de créer une card de référence de type lieu
10  */
11
12 using MyLibrary.classes;
13 using System;
14 using System.Drawing;
15 using System.Windows.Forms;
16 using WindowsFormsApp1;
17
18 namespace MyLibrary
19 {
20     public class CardReferenceLieu : CardReference
21     {
22         /// <summary>
23         /// Permet de créer card de référence lieu
24         /// </summary>
25         /// <param name="reference"></param>
26         /// <param name="frm"></param>
27         public CardReferenceLieu(Reference reference,
28             frmCollectionReferences frm) : base(reference, frm)
29         {
30
31             Label lbltxtTitre = new Label();
32             lbltxtTitre.Text = "Titre : ";
33             lbltxtTitre.Location = new Point(Location.X + 2, Location.Y +
34                 + 20);
35             Label lblTitre = new Label();
36             lblTitre.Text = ObjReference.NomReference;
37             lblTitre.Location = new Point(Location.X + 2, Location.Y +
38                 35);
39
40             Label lblTxtDescription = new Label();
41             lblTxtDescription.Text = "Description : ";
42             lblTxtDescription.Location = new Point(190, 5);
43             lblTxtDescription.Location = new Point(Location.X + 2,
44                 Location.Y + 80);
45             Label lblDescription = new Label();
46             lblDescription.Text = ObjReference.DescriptionLieu;
47             lblDescription.Location = new Point(Location.X + 2,
48                 Location.Y + 95);
49             lblDescription.Size = new Size(Size.Width - 2, Size.Height -
50                 80);
51
52             //Ajout des événements de click
53             lbltxtTitre.Click += ClickCard;
```

```
48         lblTitre.Click += ClickCard;
49         lblTxtDescription.Click += ClickCard;
50         lblDescription.Click += ClickCard;
51
52         //ajout des éléments dans les controls de la card
53         Controls.Add(lblTitre);
54         Controls.Add(lbltxtTitre);
55         Controls.Add(lblDescription);
56         Controls.Add(lblTxtDescription);
57     }
58
59     public Reference Reference
60     {
61         get => default;
62         set
63         {
64         }
65     }
66
67     public Livre Livre
68     {
69         get => default;
70         set
71         {
72         }
73     }
74
75     protected override void ClickCard(object o, EventArgs e)
76     {
77         Frm.SelectionCard(this);
78     }
79 }
80 }
81 }
```

```
1 /* Projet    : MyLibrary - TPI 2022
2  * Version   : 1.0
3  * Date     : 18.05.2022
4 *
5  * Auteur    : Karel V. Svoboda
6  * Classe   : I.DA-P4A
7 *
8  * Class    : CardReferenceLivre.cs cardReference
9  * Decs.    : Permet de créer une card de référence de type livre
10 */
11
12 using MyLibrary.classes;
13 using System;
14 using System.Drawing;
15 using System.Windows.Forms;
16 using WindowsFormsApp1;
17
18 namespace MyLibrary
19 {
20     public class CardReferenceLivre : CardReference
21     {
22         #region variables d'instances
23         private Livre _livre;
24         private PictureBox _image;
25         #endregion
26
27         public Livre ObjLivre
28         {
29             get { return _livre; }
30             set { _livre = value; }
31         }
32
33         public Reference Reference
34         {
35             get => default;
36             set
37             {
38             }
39         }
40
41         #region constructeurs
42         /// <summary>
43         /// Permet de créer une card de référence de type livre avec
44         /// les données d'un livre
45         /// </summary>
46         /// <param name="livre">données du livre</param>
47         /// <param name="reference">données de référence du livre</
48         /// param>
49         /// <param name="frm">form où la card est affichée</param>
50         public CardReferenceLivre(Livre livre, Reference reference,
51             frmCollectionReferences frm) : base(reference, frm)
52         {
53
54     }
```

```
51         _livre = livre;
52         ObjReference.NomReference = _livre.Titre;
53         ObjReference.Auteur = _livre.Auteur;
54         ObjReference.NomImage = _livre.NomImage;
55         _image = new PictureBox();
56         _image.Size = new Size(150, 70);
57         _image.BackColor = Color.LightBlue;
58
59     try
60     {
61         _image.Image = Image.FromFile(_livre.NomImage);
62         _image.SizeMode = PictureBoxSizeMode.StretchImage;
63
64     }
65     catch (Exception ex)
66     {
67         Console.WriteLine(ex.Message);
68     }
69
70     Label lbltxtAuteur = new Label();
71     lbltxtAuteur.Text = "Auteur : ";
72     lbltxtAuteur.Location = new Point(Location.X + 2,           ↴
73                                         Location.Y + 80);
74     Label lblAuteur = new Label();
75     lblAuteur.Text = _livre.Auteur;
76     lblAuteur.Location = new Point(Location.X + 2,           ↴
77                                         Location.Y + 95);
78
79     Label lbltxtTitre = new Label();
80     lbltxtTitre.Text = "Titre : ";
81     lbltxtTitre.Location = new Point(190, 5);
82     lbltxtTitre.Location = new Point(Location.X + 2,           ↴
83                                         Location.Y + 120);
84     Label lblTitre = new Label();
85     lblTitre.Text = _livre.Titre;
86     lblTitre.Location = new Point(Location.X + 2, Location.Y +  ↴
87                                         135);
88
89     _image.Click += ClickCard;
90     lbltxtAuteur.Click += ClickCard;
91     lblAuteur.Click += ClickCard;
92     lbltxtTitre.Click += ClickCard;
93     lblTitre.Click += ClickCard;
94
95         Controls.Add(_image);
96         Controls.Add(lblAuteur);
97         Controls.Add(lbltxtAuteur);
98         Controls.Add(lblTitre);
99         Controls.Add(lbltxtTitre);
}
```

```
100
101     public CardReferenceLivre(Utilisateur utilisateur, Reference
102         referenceLivre, frmCollectionReferences frm) : this
103             (ClientRest.Instance.LivreParIdLivre(utilisateur,
104                 referenceLivre.LivreReference), referenceLivre, frm) { }
105
106     #endregion
107
108     #region méthodes
109     /// <summary>
110     /// Appelle la fonction SelectionCard de la form collection
111     /// références
112     /// </summary>
113     /// <param name="o"></param>
114     /// <param name="e"></param>
115     protected override void ClickCard(object o, EventArgs e)
116     {
117         Frm.SelectionCard(this);
118     }
119
120 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : CardReferenceMusique.cs cardReference
9   * Decs.     : Permet de créer une card de référence de type musique
10  */
11
12 using MyLibrary.classes;
13 using System;
14 using System.Drawing;
15 using System.Windows.Forms;
16 using WindowsFormsApp1;
17
18 namespace MyLibrary
19 {
20     public class CardReferenceMusique : CardReference
21     {
22         #region Variables d'instances
23         private PictureBox _image;
24         #endregion
25
26         #region Constructeurs
27
28         /// <summary>
29         /// Permet de créer une card de type référence de musique
30         /// </summary>
31         /// <param name="reference">Données de la card</param>
32         /// <param name="frm">form dans laquelle la card s'affiche</param>
33         public CardReferenceMusique(Reference reference,
34                                     frmCollectionReferences frm) : base(reference, frm)
35         {
36             _image = new PictureBox();
37             _image.Size = new Size(150, 70);
38             _image.BackColor = Color.Orange;
39             try
40             {
41                 _image.Image = Image.FromFile(reference.NomImage);
42             }
43             catch (Exception ex)
44             {
45                 Console.WriteLine(ex.Message);
46             }
47             //Définition de la mise en page de l'image
48             _image.SizeMode = PictureBoxSizeMode.StretchImage;
49
50             //Création des éléments visuels
51             Label lbltxtAuteur = new Label();
```

```
52         lbltxtAuteur.Text = "Auteur : ";
53         lbltxtAuteur.Location = new Point(Location.X + 2,
54                                         Location.Y + 80);
54         Label lblAuteur = new Label();
55         lblAuteur.Text = ObjReference.Auteur;
56         lblAuteur.Location = new Point(Location.X + 2, Location.Y + ↵
57                                         95);
57
58         Label lbltxtTitre = new Label();
59         lbltxtTitre.Text = "Titre : ";
60         lbltxtTitre.Location = new Point(190, 5);
61         lbltxtTitre.Location = new Point(Location.X + 2, Location.Y ↵
62                                         + 120);
62         Label lblTitre = new Label();
63         lblTitre.Text = ObjReference.NomReference;
64         lblTitre.Location = new Point(Location.X + 2, Location.Y + ↵
65                                         135);
65
66         //Ajout des événements de click pour sélectionner la card
67         lbltxtAuteur.Click += ClickCard;
68         lblAuteur.Click += ClickCard;
69         lbltxtTitre.Click += ClickCard;
70         lblTitre.Click += ClickCard;
71
72         //Ajout dans les controls de la card
73         Controls.Add(_image);
74         Controls.Add(lblAuteur);
75         Controls.Add(lbltxtAuteur);
76         Controls.Add(lblTitre);
77         Controls.Add(lbltxtTitre);
78     }
79
80     public ReferenceMusique ReferenceMusique
81     {
82         get => default;
83         set
84         {
85         }
86     }
87
88     public Reference Reference
89     {
90         get => default;
91         set
92         {
93         }
94     }
95     #endregion
96
97     /// <summary>
98     /// Appel la fonction Selection de la form
99     frmSelectionReferences avec les données de l'objet
    /// </summary>
```

```
100     /// <param name="o"></param>
101     /// <param name="e"></param>
102     protected override void ClickCard(object o, EventArgs e)
103     {
104         Frm.SelectionCard(this);
105     }
106 }
107 }
108 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 0.6
3   * Date      : 10.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : ClientRest.cs Class
9   * Descs.    : Sert à faire des appels à l'API
10  */
11
12 using Newtonsoft.Json;
13 using System;
14 using System.Collections.Generic;
15 using System.IO;
16 using System.Net;
17
18 namespace MyLibrary.classes
19 {
20     public sealed class ClientRest
21     {
22         #region variables d'instances
23         private string _cheminApi;
24         private static ClientRest instance = null;
25         #endregion
26
27         #region constructeurs
28         private ClientRest()
29         {
30             _cheminApi = "http://localhost/ProjetsWeb/MyLibrary/src/
31             API_MyLibrary/";
32         }
33         #endregion
34
35         #region propriétés
36         //https://www.c-sharpcorner.com/UploadFile/8911c4/singleton-
37         design-pattern-in-C-Sharp/
38         public static ClientRest Instance
39         {
40             get
41             {
42                 if (instance == null)
43                 {
44                     instance = new ClientRest();
45                 }
46             }
47         }
48
49         #region Méthodes
50         /// <summary>
```

```
52     /// Permet de faire une requête à une API
53     /// </summary>
54     /// <param name="url">url de la demande</param>
55     /// <returns>le résultat de la requête</returns>
56     public string ApiRequest(string url, string method)
57     {
58         string strresulttest = null;
59         try
60         {
61             url = _cheminApi + url;
62             string strurltest = String.Format(url);
63             WebRequest requestObject = WebRequest.Create(strurltest);
64             requestObject.Method = method.ToUpper();
65             HttpWebResponse responseObject = null;
66             responseObject = (HttpWebResponse)requestObject.GetResponse();
67
68
69             using (Stream steam = responseObject.GetResponseStream())
70             {
71                 StreamReader sr = new StreamReader(steam);
72                 strresulttest = sr.ReadToEnd();
73                 var settings = new JsonSerializerSettings();
74                 settings.MetadataPropertyHandling =
75                     MetadataPropertyHandling.Ignore;
76                 sr.Close();
77             }
78         }
79         catch (Exception ex)
80         {
81             return ex.ToString();
82         }
83
84         return strresulttest;
85     }
86
87     /// <summary>
88     /// Fait un appel avec une réponse en booléen en fonction du code de l'API
89     /// </summary>
90     /// <param name="url">url relatif</param>
91     /// <param name="method">GET, POST, DELETE, PUT</param>
92     /// <returns></returns>
93     public bool AppelSimple(string url, string method)
94     {
95         try
96         {
97             url = _cheminApi + url;
98             string strurltest = String.Format(url);
99             WebRequest requestObject = WebRequest.Create(strurltest);
```

```
100             (strurltest);
101             requestObject.Method = method.ToUpper();
102             HttpWebResponse responseObject = null;
103             responseObject = (HttpWebResponse)
104                 requestObject.GetResponse();
105             }
106             catch (Exception ex)
107             {
108                 Console.WriteLine(ex.ToString());
109                 return false;
110             }
111         }
112 
113     /// <summary>
114     /// Permet de déserialiser le JSON
115     /// https://www.youtube.com/watch?v=CjoAYs1TKX0
116     /// </summary>
117     /// <param name="strJson">string en Json</param>
118     public dynamic DeserialiseJSON(string strJson)
119     {
120         try
121         {
122             var jPerson = JsonConvert.DeserializeObject<dynamic>
123                 (strJson);
124 
125             return jPerson;
126         }
127         catch (Exception ex)
128         {
129             Console.WriteLine("error : " + ex);
130             return null;
131         }
132 
133     /// <summary>
134     /// Permet de récupérer les livres par leur utilisateur
135     /// </summary>
136     /// <param name="utilisateur"></param>
137     /// <returns></returns>
138     public List<Livre> LivresParUtilisateur(Utilisateur
139         utilisateur)
140     {
141         List<Livre> livres = new List<Livre>();
142         try
143         {
144             dynamic livresDynamic = DeserialiseJSON(ApiRequest("?
145                 email=" + utilisateur.Email + "&password=" +
146                 utilisateur.Password + "&table=livres", "GET"));
147             foreach (var element in livresDynamic)
148             {
149                 livres.Add(new Livre(Convert.ToInt32(element
```

```
...yLibrary\WindowsFormsApp1\classes\API\ClientRest.cs 4
        ["idLivre"]), Convert.ToString(element["titre"]),
        Convert.ToString(element["auteur"]),
        Convert.ToString(element["nomImage"]),
        Convert.ToInt32(element
        ["idUtilisateur"])));
147    }
148    }
149    catch(Exception ex)
150    {
151        Console.WriteLine(ex.ToString());
152    }
153
154    return livres;
155}
156
157 /// <summary>
158 /// Permet de récupérer les livres par leur utilisateur et une
159 /// recherche personnalisée
160 /// </summary>
161 /// <param name="utilisateur">utilisateur</param>
162 /// <param name="recherche">chaîne personnalisée</param>
163 /// <returns></returns>
164 public List<Livre> LivresParUtilisateur(Utilisateur
165     utilisateur, string recherche)
166 {
167     List<Livre> livres = new List<Livre>();
168     dynamic livresDynamic = DeserialiseJSON(ApiRequest("?
169         email=" + utilisateur.Email
170         +"&password=" + utilisateur.Password
171         +"&table=livres&recherche=" + recherche + "", "GET"));
172     foreach (var element in livresDynamic)
173     {
174         livres.Add(new Livre(Convert.ToInt32(element
175             ["idLivre"]),
176             Convert.ToString(element["titre"]),
177             Convert.ToString(element["auteur"]),
178             Convert.ToInt32(element
179             ["idUtilisateur"])));
180     }
181     return livres;
182 }
183
184 /// <summary>
185 /// Récupère tous les types
186 /// </summary>
187 /// <param name="utilisateur"></param>
188 /// <returns></returns>
189 public List<Type> TousTypes(Utilisateur utilisateur)
190 {
191     List<Type> types = new List<Type>();
192     dynamic typesDynamic = DeserialiseJSON(ApiRequest("?email="
193         + utilisateur.Email + "&password=" + utilisateur.Password
194         + "&table=types", "GET"));
195     foreach (var element in typesDynamic)
196     {
```

...yLibrary\WindowsFormsApp1\classes\API\ClientRest.cs 5

```
185         types.Add(new Type(Convert.ToInt32(element["idType"]),
186                     Convert.ToString(element["nomType"])));
187     }
188     return types;
189 }
190
191 /// <summary>
192 /// Récupère tous toutes les références d'un livre et les classe en fonction du type
193 /// </summary>
194 /// <param name="utilisateur"></param>
195 /// <param name="livre"></param>
196 /// <returns></returns>
197 public List<Reference> ReferencesParLivre(Utilisateur
198     utilisateur, Livre livre)
199 {
200     List<Reference> referencesParLivre = new List<Reference>();
201     dynamic referencesDynamic = DeserialiseJSON(ApiRequest("?
202         email=" + utilisateur.Email + "&password=" +
203         utilisateur.Password + "&table=references&idLivre=" +
204         livre.IdLivre.ToString() + "", "GET"));
205     foreach (var element in referencesDynamic)
206     {
207         switch (Convert.ToInt32(element["idType"]))
208         {
209             case 1:
210                 referencesParLivre.Add(new ReferenceLivre
211                     (Convert.ToInt32(element["idReference"]),
212                     Convert.ToString(element["nomImage"]),
213                     Convert.ToInt32(element["livreReference"]),
214                     Convert.ToInt32(element["idLivre"])));
215                     break;
216             case 2:
217                 referencesParLivre.Add(new ReferenceMusique
218                     (Convert.ToInt32(element["idReference"]),
219                     Convert.ToString(element["nomImage"]),
220                     Convert.ToString(element["nomReference"]),
221                     Convert.ToString(element["auteur"]),
222                     Convert.ToInt32(element["idLivre"])));
223                     break;
224             case 3:
225                 if(element["descriptionLieu"] == null)
226                 {
227                     referencesParLivre.Add(new ReferenceLieu
228                         (Convert.ToInt32(element["idReference"]),
229                         Convert.ToString(element["nomReference"]),
230                         Convert.ToInt32(element["idLivre"])));
231                 }
232                 else
233                 {
234                     referencesParLivre.Add(new ReferenceLieu
235                         (Convert.ToInt32(element["idReference"]),
236                         Convert.ToString(element["nomReference"]),
237                         Convert.ToInt32(element["idLivre"])));
238                 }
239             }
240         }
241     }
242 }
```

```
...yLibrary\WindowsFormsApp1\classes\API\ClientRest.cs 6
    Convert.ToString(element["nomReference"]),
    Convert.ToString(element["descriptionLieu"]),
    Convert.ToInt32(element["idLivre"])));
219        }
220
221        break;
222    }
223}
224
225    return referencesParLivre;
226}
227
228 /// <summary>
229 /// Recupère un livre par son id
230 /// </summary>
231 /// <param name="utilisateur">utilisateur propriétaire du      ↵
232     livre</param>
233 /// <param name="idLivre">id du livre à rechercher</param>
234 /// <returns></returns>
235 public Livre LivreParIdLivre(Utilisateur utilisateur, int      ↵
236     idLivre)
237 {
238     dynamic LivreDynamic = DeserialiseJSON(ApiRequest("?email=" ↵
239         + utilisateur.Email + "&password=" + utilisateur.Password ↵
240         + "&table=livres&idLivre=" + idLivre.ToString() + "",      ↵
241         "GET"));
242     return new Livre(Convert.ToInt32(LivreDynamic["idLivre"]),
243         Convert.ToString(LivreDynamic["titre"]),
244         Convert.ToString(LivreDynamic["auteur"]),
245         Convert.ToString(LivreDynamic["nomImage"]),
246         Convert.ToInt32(LivreDynamic["idUtilisateur"]));
247 }
248
249 /// <summary>
250 /// Récupère le dernier livre de l'utilisateur
251 /// </summary>
252 /// <param name="utilisateur"></param>
253 public int DernierLivreUtilisateur(Utilisateur utilisateur)
254 {
255     return Convert.ToInt32(DeserialiseJSON(ApiRequest("?email=" ↵
256         + utilisateur.Email + "&password=" + utilisateur.Password ↵
257         + "&table=livres&tri=dernier", "GET")));
258 }
259
260 #endregion
261 }
262 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : frmCollectionLivres.cs Form
9   * Descs.    : Vue de la collection de Livres d'un utilisateur
10  */
11
12 using System;
13 using System.Collections.Generic;
14 using System.Drawing;
15 using System.IO;
16 using System.Windows.Forms;
17 using MyLibrary.classes;
18 using WindowsFormsApp1;
19
20 namespace MyLibrary
21 {
22     public partial class frmCollectionLivres : Form
23     {
24         //Variables d'instances
25         Utilisateur _utilisateur;
26         ClientRest _clientRest;
27         List<Livre> _listLivresUtilisateur;
28         List<Card> _listCard;
29         CardLivre _cardSelectionne;
30
31         /// <summary>
32         /// Form de collection de livres
33         /// Affiche les livres de l'utilisateur dans une liste ↗
34         /// d'objets.
35         /// Permet de faire un CRUD sur les livres via un formulaire
36         /// </summary>
37         /// <param name="utilisateur">Données de l'utilisateur</param>
38         /// <param name="frmPrecedente">Form de connexion</param>
39         public frmCollectionLivres(Utilisateur utilisateur, ↗
40             frmConnexion frmPrecedente)
41         {
42             //Initialisation des composants
43             InitializeComponent();
44             //Stockage des données dans les variables d'instances
45             _utilisateur = utilisateur;
46             _clientRest = ClientRest.Instance;
47             //Instanciations des lists
48             _listLivresUtilisateur = new List<Livre>();
49             _listCard = new List<Card>();
50             //Fermeture de la form de connexion
51             frmPrecedente.Close();
52             //Affichage des données
53             RefreshView();
```

```
52     }
53
54     /// <summary>
55     /// Ajout d'un livre
56     /// Source du code de l'importation de l'image :
57     /// https://www.codeproject.com/Questions/546631/
      howplustoplussavepluspictureboxplusimageplusinplus
58     /// </summary>
59     /// <param name="sender"></param>
60     /// <param name="e"></param>
61     private void btnAjouter_Click(object sender, EventArgs e)
62     {
63         //Vérification des champs
64         if (VerificationInputs())
65         {
66             //tentative d'ajout de livre
67             try
68             {
69                 ImageInFile imageEnregistrer = new ImageInFile
                  ((Bitmap)pbxImageAjouter.Image);
70                 //Création du livre et appel de la méthode d'ajout
71                 if (new Livre(0, tbxTitre.Text, tbxAuteur.Text,
                  imageEnregistrer.Nom + imageEnregistrer.Extension,
                  0).PostLivre(_utilisateur))
72                 {
73                     //enregistrement de l'image
74                     imageEnregistrer.SaveBmp();
75                     //Affichage du message de confirmation
76                     MessageBox.Show("La donnée a été ajoutée",
                  "Livre ajouté", MessageBoxButtons.OK,
                  MessageBoxIcon.Information);
77                     //mise à jour de la vue
78                     RefreshView();
79                 }
80             }
81             catch (Exception ex)
82             {
83                 //Affichage du message d'erreur
84                 MessageBox.Show("Une erreur s'est produite lors de
                  l'ajout du livre", "Erreur interne",
                  MessageBoxButtons.OK, MessageBoxIcon.Error);
                  Console.WriteLine(ex.Message);
85             }
86         }
87     }
88     else
89     {
90         //Affichage du message d'erreur
91         MessageBox.Show("Veuillez ajouter remplir tous les
                  champs", "Inputs non valides", MessageBoxButtons.OK,
                  MessageBoxIcon.Error);
92     }
93 }
94 }
```

```
95     /// <summary>
96     /// Pression du bouton d'importation d'image
97     /// Source du code d'importation et transformation en Bitmap      ↵
98     /// des images
99     /// https://stackoverflow.com/questions/6122984/load-a-bitmap-    ↵
100    /// image-into-windows-forms-using-open-file-dialog
101    /// </summary>
102    /// <param name="sender"></param>
103    /// <param name="e"></param>
104    private void btnImporterImage_Click(object sender, EventArgs e)
105    {
106        //Ouverture du dialogue d'importation
107        using (OpenFileDialog dlg = new OpenFileDialog())
108        {
109            dlg.Title = "Open Image";
110            dlg.Filter = "Image Files (*.bmp;*.jpg;*.jpeg,*.png)|    ↵
111                *.BMP;*.JPG;*.JPEG;*.PNG";
112            if (dlg.ShowDialog() == DialogResult.OK)
113            {
114                // Create a new Bitmap object from the picture file ↵
115                // on disk,
116                // and assign that to the PictureBox.Image property
117                pbxImageAjouter.Image = new Bitmap(dlg.FileName);
118            }
119        }
120        /// <summary>
121        /// S'active lors de la fermeture de la form
122        /// </summary>
123        /// <param name="sender"></param>
124        private void frmCollectionLivres_FormClosed(object sender,      ↵
125            FormClosedEventArgs e)
126        {
127            //déconnecte l'utilisateur
128            _utilisateur.Deconnexion();
129        }
130        /// <summary>
131        /// Permet de mettre à jour les données de l'application sans    ↵
132        /// filtre
133        /// </summary>
134        public void RefreshView()
135        {
136            //remise à zéro des listes
137            flpListCard.Controls.Clear();
138            _listLivresUtilisateur.Clear();
139            _listLivresUtilisateur = _clientRest.LivresParUtilisateur    ↵
140            (_utilisateur);
141            _listCard.Clear();
```

```
141         //Création de card pour chaque livre de l'utilisateur
142         foreach (Livre livresPourCard in _listLivresUtilisateur)
143         {
144             Card nouvelleCard = new CardLivre(livresPourCard,      ↵
145                                         this);
146             _listCard.Add(nouvelleCard);
147             flpListCard.Controls.Add(nouvelleCard);
148         }
149     }
150 
151     /// <summary>
152     /// Permet de mettre à jour les données de l'application avec      ↵
153     /// un filtre
154     /// </summary>
155     /// <param name="recherche">Chaîne de caractères à rechercher      ↵
156     /// dans l'auteur et le titre</param>
157     public void RefreshView(string recherche)
158     {
159         //remise à zéro des listes
160         flpListCard.Controls.Clear();
161         _listLivresUtilisateur.Clear();
162         _listLivresUtilisateur = _clientRest.LivresParUtilisateur      ↵
163             (_utilisateur, recherche);
164         _listCard.Clear();
165 
166         //Création de card pour chaque livre de l'utilisateur après      ↵
167         // le filtrage
168         foreach (Livre livresPourCard in _listLivresUtilisateur)
169         {
170             CardLivre nouvelleCard = new CardLivre(livresPourCard,      ↵
171                                         this);
172             _listCard.Add(nouvelleCard);
173             flpListCard.Controls.Add(nouvelleCard);
174         }
175     }
176 
177     /// <summary>
178     /// Mise à zéro de la vue si l'utilisateur clique sur      ↵
179     /// flpListCard
180     /// </summary>
181     /// <param name="sender"></param>
182     /// <param name="e"></param>
183     private void flpListCard_Click(object sender, EventArgs e)
184     {
185         VueParDefault();
186     }
187 
188     /// <summary>
189     /// Permet de vider la carte sélectionnée et de remettre à zéro      ↵
190     /// la vue de l'application
191     /// </summary>
192     private void VueParDefault()
193     {
```

```
186         //Mise à zéro du style de toutes les cards
187         foreach (CardLivre uneCard in _listCard)
188         {
189             uneCard.BackColor = Color.White;
190             uneCard.ForeColor = Color.Black;
191         }
192         //Mise à zéro de la card sélectionnée
193         _cardSelectionne = null;
194         //Activation du bouton d'ajout et désactivation des boutons ↵
195         //de suppression et modifications
196         btnAjouter.Enabled = true;
197         btnModifier.Enabled = false;
198         btnSupprimer.Enabled = false;
199         //Mise à zéro des inputs
200         tbxAuteur.Text = null;
201         tbxTitre.Text = null;
202         pbxImageAjouter.Image = null;
203     }
204
205     /// <summary>
206     /// Permet de sélectionner une card et de stocker ses données
207     /// </summary>
208     /// <param name="card"></param>
209     public void SelectionnerCard(CardLivre card)
210     {
211         //stockage de la card dans une variable d'instance
212         _cardSelectionne = card;
213         //Activation des bouton de modification et de suppression ↵
214         //et désactivation du bouton d'ajout
215         btnModifier.Enabled = true;
216         btnSupprimer.Enabled = true;
217         btnAjouter.Enabled = false;
218         //Mise à zéro du style des card
219         foreach(CardLivre uneCard in _listCard)
220         {
221             uneCard.BackColor = Color.White;
222             uneCard.ForeColor = Color.Black;
223         }
224         //Mise en évidence de la card sélectionnée
225         card.BackColor = Color.Gray;
226         card.ForeColor = Color.White;
227         //Affichage des données dans les inputs
228         tbxAuteur.Text = card.ObjLivre.Auteur;
229         tbxTitre.Text = card.ObjLivre.Titre;
230         //Recherche de l'image par son nom
231         pbxImageAjouter.Image = Image.FromFile
232             (card.ObjLivre.NomImage);
233     }
234
235     /// <summary>
236     /// Modification d'image suite à la pression du bouton
237     /// </summary>
238     /// <param name="sender"></param>
```

```
236     /// <param name="e"></param>
237     private void btnModifier_Click(object sender, EventArgs e)
238     {
239         //Vérification des inputs
240         if (VerificationInputs())
241         {
242             try
243             {
244                 ImageInFile imageEnregistrer = new ImageInFile
245                 ((Bitmap)pbxImageAjouter.Image);
246                 //Essai de modification du livre
247                 if (_cardSelectionne.ObjLivre.PutLivre
248                     (_utilisateur, new Livre(0, tbxTitre.Text,
249                         tbxAuteur.Text, imageEnregistrer.Nom +
250                         imageEnregistrer.Extension, 0)))
251                 {
252                     //Suppression de l'image précédente
253                     File.Delete
254                     (_cardSelectionne.ObjLivre.NomImage);
255                     //Sauvegarde de la nouvelle image
256                     imageEnregistrer.SaveBmp();
257                     //Affichage du message en cas de succès
258                     MessageBox.Show("Le livre a été modifié",
259                         "Livre modifié", MessageBoxButtons.OK,
260                         MessageBoxIcon.Information);
261                     //Mise à jour de la vue
262                     RefreshView();
263                 }
264             }
265             else
266             {
267                 //Affichage du message d'erreur
268                 MessageBox.Show("Une erreur s'est produite lors de
269                     la modification du livre", "Erreur interne",
270                     MessageBoxButtons.OK, MessageBoxIcon.Error);
271             }
272             catch (Exception ex)
273             {
274                 //Affichage du message d'erreur
275                 MessageBox.Show("Veuillez ajouter remplir tous les
276                     champs", "Inputs non valides", MessageBoxButtons.OK,
277                     MessageBoxIcon.Error);
278             }
279         }
280     }
```

```
276
277     /// <summary>
278     /// Pression du bouton de suppression
279     /// </summary>
280     /// <param name="sender"></param>
281     /// <param name="e"></param>
282     private void btnSupprimer_Click(object sender, EventArgs e)
283     {
284         //affichage du message de confirmation
285         DialogResult dr = MessageBox.Show("Voulez-vous vraiment      ↵
286             supprimer ce livre", "Suppression de livre",      ↵
287             MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);      ↵
288
289         //Si l'utilisateur confirme la suppression
290         if (dr == DialogResult.Yes)
291         {
292             try
293             {
294                 //Tentative de suppression de livre
295                 if (_cardSelectionne.ObjLivre.DeleteLivre      ↵
296                     (_utilisateur))
297                 {
298                     //Mise à zéro de la vue
299                     RefreshView();
300                     VueParDefault();
301                     //Affichage du message de confirmation
302                     MessageBox.Show("Le livre a été supprimé",      ↵
303                         "Livre supprimé", MessageBoxButtons.OK,      ↵
304                         MessageBoxIcon.Information);
305                 }
306             }
307             catch(Exception ex)
308             {
309                 MessageBox.Show("Problème lors de la suppression du      ↵
310                     livre", "Erreur", MessageBoxButtons.OK,      ↵
311                     MessageBoxIcon.Error);
312                 Console.WriteLine(ex.Message);
313             }
314         }
315         /// <summary>
316         /// Vérification si les inputs sont remplis
317         /// </summary>
318         /// <returns>True = inputs remplis, False = un ou plusieurs      ↵
319             input(s) est ou sont vide(s)</returns>
320         private bool VerificationInputs()
321         {
322             //Vérification des inputs
323             if(tbxAuteur.Text == "" && tbxRecherche.Text == "")
```

```
321             return false;
322         }
323         return true;
324     }
325
326     /// <summary>
327     /// S'enclanche lors d'un changement de texte de la ↵
328     /// tbxRecherche
329     /// </summary>
330     /// <param name="sender"></param>
331     /// <param name="e"></param>
332     private void tbxRecherche_TextChanged(object sender, EventArgs ↵
333     e)
334     {
335         //Si le champs n'est pas vide, on appelle la fonction qui ↵
336         //affiche les livres en fonction du filtre
337         //Sinon affichage de tous les livres de l'utilisateur
338         if (tbxRecherche.Text != "")
339         {
340             RefreshView(tbxRecherche.Text);
341         }
342         else
343         {
344             RefreshView();
345         }
346
347         public void AfficherReference(Livre livre)
348         {
349             frmCollectionReferences collectionReferences = new ↵
350                 frmCollectionReferences(livre, _utilisateur, this);
351             collectionReferences.Show();
352
353             /// <summary>
354             /// S'enclanche si l'utilisateur click sur la form
355             /// Mise à zéro des données
356             /// </summary>
357             /// <param name="sender"></param>
358             /// <param name="e"></param>
359             private void frmCollectionLivres_Click(object sender, EventArgs ↵
360             e)
361             {
362                 VueParDefault();
363             }
364         }
```

```
1 /* Projet    : MyLibrary - TPI 2022
2  * Version   : 1.0
3  * Date      : 18.05.2022
4  *
5  * Auteur    : Karel V. Svoboda
6  * Classe    : I.DA-P4A
7  *
8  * Class     : frmCollectionReferences.cs Form
9  * Descs.    : Vue de la collection des références d'un livre
10 */
11
12 using MyLibrary;
13 using MyLibrary.classes;
14 using System;
15 using System.Collections.Generic;
16 using System.Drawing;
17 using System.Windows.Forms;
18
19 namespace WindowsFormsApp1
20 {
21     public partial class frmCollectionReferences : Form
22     {
23         //Variables d'instances
24         private Livre _livre;
25         private Utilisateur _utilisateur;
26         private ClientRest _clientRest;
27         private List<Reference> _references;
28         private List<Card> _cardsReferences;
29         private List<Livre> _livres;
30         private CardReference _cardSelectionne;
31         private frmCollectionLivres _frmCollectionLivres;
32
33         //Référence ambigu
34         private List<MyLibrary.classes.Type> _types;
35
36         //Propriétés
37         public Livre ObjLivre
38         {
39             get { return _livre; }
40             set { _livre = value; }
41         }
42         /// <summary>
43         /// Vue
44         /// </summary>
45         /// <param name="livre"></param>
46         /// <param name="utilisateur"></param>
47         public frmCollectionReferences(Livre livre, Utilisateur
48                                         utilisateur, frmCollectionLivres frmCollectionLivres)
49         {
50             _livre = livre;
51             _utilisateur = utilisateur;
52             _references = new List<Reference>();
53             _cardsReferences = new List<Card>();
```

```
53         _frmCollectionLivres = frmCollectionLivres;
54
55         InitializeComponent();
56         //Changement dynamique du nom de la form
57         Text += " : " + _livre.Titre.ToLower();
58         lblTitreLivre.Text = _livre.Titre;
59         _clientRest = ClientRest.Instance;
60         _types = _clientRest.TousTypes(_utilisateur);
61         _livres = _clientRest.LivresParUtilisateur(_utilisateur);
62
63         majCbxTypes(_types);
64
65         cbxFiltreType.Items.Add("Tous");
66         cbxFiltreType.Items.Add("Livres");
67         cbxFiltreType.Items.Add("Musiques");
68         cbxFiltreType.Items.Add("Lieux");
69         cbxFiltreType.SelectedIndex = 0;
70
71         UpdateFormView();
72         btnAjouter.Enabled = true;
73         btnModifier.Enabled = true;
74         btnSupprimer.Enabled = true;
75     }
76
77     private void btnAjouter_Click(object sender, EventArgs e)
78     {
79         ImageInFile imageEnregistrer = null;
80         //Ajout test de la combobox
81         switch (cbxType.SelectedIndex)
82         {
83             case 0:
84                 imageEnregistrer = new ImageInFile((Bitmap) pbxImage.Image);
85                 if (cbxLivre.SelectedIndex == 0)
86                 {
87                     //new ReferenceLivre(0)
88                     Livre nouveauLivre = new Livre(0,
89                         tbxTitre.Text, tbxAuteur.Text, imageEnregistrer.Nom +
90                         imageEnregistrer.Extension,
91                         _utilisateur.IdUtilisateur);
92                     if (nouveauLivre.PostLivre(_utilisateur))
93                     {
94                         imageEnregistrer.SaveBmp();
95                         new ReferenceLivre(0,
96                             nouveauLivre.NomImage,
97                             _clientRest.DernierLivreUtilisateur(_utilisateur),
98                             _livre.IdLivre).PostReference(_utilisateur);
99                         _frmCollectionLivres.RefreshView();
100                        UpdateFormView();
101                    }
102                }
103            else
```

```
99             {
100                 imageEnregistrer = new ImageInFile((Bitmap) pbxImage.Image);
101                 if (new ReferenceLivre(0, imageEnregistrer.Nom, _livres[cbxLivre.SelectedIndex - 1].IdLivre, _livre.IdLivre).PostReference(_utilisateur))
102                 {
103                     _frmCollectionLivres.RefreshView();
104                     MessageBox.Show("La référence et le livre ont été ajoutées", "Référence Ajoutée", MessageBoxButtons.OK, MessageBoxIcon.Information);
105                     UpdateFormView();
106                 }
107                 else
108                 {
109                     MessageBox.Show("Un problème s'est produit lors de l'envoie de la donnée", "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
110                 }
111             }
112             break;
113         case 1:
114             imageEnregistrer = new ImageInFile((Bitmap) pbxImage.Image);
115             if (new ReferenceMusique(0, imageEnregistrer.Nom + imageEnregistrer.Extension, tbxTitre.Text, tbxAuteur.Text, ObjLivre.IdLivre).PostReference(_utilisateur))
116             {
117                 imageEnregistrer.SaveBmp();
118                 MessageBox.Show("La référence à été ajoutée", "Référence Ajoutée", MessageBoxButtons.OK, MessageBoxIcon.Information);
119                 UpdateFormView();
120             } else{
121                 MessageBox.Show("Un problème s'est produit lors de l'envoie de la donnée", "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
122             }
123             break;
124         case 2:
125             if (new ReferenceLieu(0, tbxTitre.Text, tbxDescription.Text, ObjLivre.IdLivre).PostReference(_utilisateur))
126             {
127                 MessageBox.Show("La référence à été ajoutée", "Référence Ajoutée", MessageBoxButtons.OK, MessageBoxIcon.Information);
128                 UpdateFormView();
129             }
130             else
131             {
132                 MessageBox.Show("Un problème s'est produit lors
```

```
de l'envoie de la donnée", "Erreur",
MessageBoxButtons.OK, MessageBoxIcon.Error);
133                         }
134                     break;
135                 }
136             }
137         }
138
139     /// <summary>
140     /// Mise à jour des éléments de la vue
141     /// </summary>
142     private void UpdateFormView()
143     {
144         //Mise à 0 des List
145         _references.Clear();
146         _cardsReferences.Clear();
147         cbxLivre.Items.Clear();
148         flpReferences.Controls.Clear();
149         //cbxFiltreType.SelectedIndex = 0;
150
151         //récupération des éléments
152         _references = _clientRest.ReferencesParLivre(_utilisateur,
153
154             _livre);
155
156         cbxLivre.Items.Add("-- NOUVEAU LIVRE --");
157         foreach (Livre lvr in _livres)
158         {
159             cbxLivre.Items.Add(lvr.Titre);
160         }
161         cbxLivre.SelectedIndex = 0;
162
163
164         if(cbxFiltreType.SelectedIndex == 0 && tbxRecherche.Text == ""
165             "")
166         {
167             foreach (Reference reference in _references)
168             {
169                 Card nouvelleCard;
170                 switch (reference.IdType)
171                 {
172                     //Livre
173                     case 1:
174                         nouvelleCard = new CardReferenceLivre
175                         (_utilisateur, reference, this);
176                         flpReferences.Controls.Add(nouvelleCard);
177                         _cardsReferences.Add(nouvelleCard);
178                         break;
179                     //Musique
180                     case 2:
181                         nouvelleCard = new CardReferenceMusique
182                         (reference, this);
183                         flpReferences.Controls.Add(nouvelleCard);
184                         _cardsReferences.Add(nouvelleCard);
185                 }
186             }
187         }
188     }
189 }
```

```
180                     break;
181                     //Lieu
182                     case 3:
183                         nouvelleCard = new CardReferenceLieu      ↵
184                             (reference, this);
185                         flpReferences.Controls.Add(nouvelleCard);
186                         _cardsReferences.Add(nouvelleCard);
187                         break;
188                     }
189                 }
190             }
191             else
192             {
193                 List<CardReference> cards = new List<CardReference>();
194                 if(cbxFiltreType.SelectedIndex != 0 &&      ↵
195                     tbxRecherche.Text == "")
196                 {
197                     cards.AddRange(RechercheReferenceParFiltre      ↵
198                         (cbxFiltreType.SelectedIndex));
199                 }
200                 else if(cbxFiltreType.SelectedIndex == 0 &&      ↵
201                     tbxRecherche.Text != "")
202                 {
203                     cards.AddRange(RechercheReferenceParFiltre      ↵
204                         (tbxRecherche.Text));
205                 }
206                 else
207                 {
208                     cards.AddRange(RechercheReferenceParFiltre      ↵
209                         (tbxRecherche.Text, cbxFiltreType.SelectedIndex));
210                 }
211
212                 foreach(CardReference card in cards)
213                 {
214                     flpReferences.Controls.Add(card);
215                     _cardsReferences.Add(card);
216                 }
217             }
218
219             /// <summary>
220             /// Mise à jour de l'états des inputs
221             /// </summary>
222             /// <param name="etat">True = activation, false =
223                     désactivation</param>
224             private void EtatTousElements(bool etat)
225             {
226                 btnImporterImage.Enabled = etat;
```



```
271         //Sélection par défaut
272         cbxType.SelectedIndex = 0;
273     }
274
275     private void cbxType_SelectedIndexChanged(object sender,      ↴
276                                         EventArgs e)
277     {
278         //Désactivation de tous les inputs
279         EtatTousElements(false);
280
281         //Activation des inputs en fonction du type de référence
282         switch (cbxType.SelectedIndex)
283         {
284             //Livre
285             case 0:
286                 tbxTitre.Enabled = true;
287                 btnImporterImage.Enabled = true;
288                 tbxAuteur.Enabled = true;
289                 cbxLivre.Enabled = true;
290                 break;
291             //Musique
292             case 1:
293                 tbxTitre.Enabled = true;
294                 btnImporterImage.Enabled = true;
295                 tbxAuteur.Enabled = true;
296                 break;
297             //Lieu
298             case 2:
299                 tbxTitre.Enabled= true;
300                 tbxDescription.Enabled = true;
301                 break;
302         }
303
304     /// <summary>
305     /// Se déclache lorsqu'une carte est sélectionnée
306     /// Sert à récupérer les données de la carte
307     /// </summary>
308     /// <param name="card">Card sélectionnée</param>
309     public void SelectionCard(CardReference card)
310     {
311         cbxType.Enabled = false;
312         _cardSelectionne = card;
313         btnAjouter.Enabled = false;
314         btnModifier.Enabled = true;
315         btnSupprimer.Enabled = true;
316         //Changement de style pour les card et mise en évidence de ↴
317         // la card sélectionnée
318         foreach (var uneCard in _cardsReferences)
319         {
320             if (uneCard == card)
321             {
322                 card.BackColor = Color.Gray;
```

```
322                     card.ForeColor = Color.White;
323                 }
324             else
325             {
326                 uneCard.BackColor = Color.White;
327                 uneCard.ForeColor = Color.Black;
328             }
329         }
330     }
331
332     //Mise en place des données dans les inputs
333     switch (card.ObjReference.IdType)
334     {
335         //Livre
336         case 1:
337             tbxTitre.Text = card.ObjReference.NomReference;
338             tbxAuteur.Text = card.ObjReference.Auteur;
339             tbxDescription.Text =
340                 card.ObjReference.DescriptionLieu; ↗
341             pbxImage.Image = Image.FromFile ↗
342                 (card.ObjReference.NomImage);
343             cbxType.SelectedIndex = 0;
344             break;
345         //Musique
346         case 2:
347             tbxTitre.Text = card.ObjReference.NomReference;
348             tbxAuteur.Text = card.ObjReference.Auteur;
349             tbxDescription.Text =
350                 card.ObjReference.DescriptionLieu; ↗
351             pbxImage.Image = Image.FromFile ↗
352                 (card.ObjReference.NomImage);
353             cbxType.SelectedIndex = 1;
354             break;
355         //Lieu
356         case 3:
357             tbxTitre.Text = card.ObjReference.NomReference;
358             tbxAuteur.Text = card.ObjReference.Auteur;
359             tbxDescription.Text =
360                 card.ObjReference.DescriptionLieu; ↗
361             cbxType.SelectedIndex = 2;
362             break;
363     }
364
365     /// <summary>
366     /// S'active quand une card est sélectionnée
367     /// </summary>
368     public void SelectionCard()
369     {
370         foreach (var uneCard in _cardsReferences)
371         {
372             uneCard.BackColor = Color.White;
373             uneCard.ForeColor = Color.Black;
```

```
370         }
371     }
372
373     private void btnModifier_Click(object sender, EventArgs e)
374     {
375
376         switch (_cardSelectionne.ObjReference.IdType)
377         {
378             case 1:
379                 ImageInFile imageEnregistrer = new ImageInFile
380 ((Bitmap)pbxImage.Image);
381                 imageEnregistrer.SaveBmp();
382                 new ReferenceLivre
383 (_cardSelectionne.ObjReference.IdReference,
384 imageEnregistrer.Nom + imageEnregistrer.Extension,
385 _cardSelectionne.ObjReference.LivreReference,
386 _cardSelectionne.ObjReference.IdLivre).PutLivre
387 (_utilisateur, new Livre
388 (_cardSelectionne.ObjReference.LivreReference,
389 tbxTitre.Text, tbxAuteur.Text, imageEnregistrer.Nom +
390 imageEnregistrer.Extension, 0));
391                 UpdateFormView();
392                 break;
393             //Musique
394             case 2:
395                 imageEnregistrer = new ImageInFile((Bitmap)
396 pbxImage.Image);
397                 if (_cardSelectionne.ObjReference.PutReference
398 (_utilisateur, new ReferenceMusique(0,
399 imageEnregistrer.Nom + imageEnregistrer.Extension,
400 tbxTitre.Text, tbxAuteur.Text, ObjLivre.IdLivre)))
401                 {
402                     imageEnregistrer.SaveBmp();
403                     MessageBox.Show("La référence à été modifiée",
404 "Référence modifiée", MessageBoxButtons.OK,
405 MessageBoxIcon.Information);
406                     UpdateFormView();
407                 }
408                 else
409                 {
410                     MessageBox.Show("Un problème s'est produit lors
411 de l'envoie de la donnée", "Erreur",
412 MessageBoxButtons.OK, MessageBoxIcon.Error);
413                 }
414                 break;
415             //Lieu
416             case 3:
417                 if (_cardSelectionne.ObjReference.PutReference
418 (_utilisateur, new ReferenceLieu(0, tbxTitre.Text,
419 tbxDescription.Text, ObjLivre.IdLivre)))
420                 {
421                     MessageBox.Show("La référence à été modifiée",
422 "
```

```
    "Référence modifiée", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
404        UpdateFormView();
405    }
406    else
407    {
408        MessageBox.Show("Un problème s'est produit lors de l'envoie de la donnée", "Erreur",
409        MessageBoxButtons.OK, MessageBoxIcon.Error);
410    }
411    break;
412}
413}
414}
415
416    private void flpReferences_Paint(object sender, PaintEventArgs e)
417    {
418    }
419}
420
421    private void flpReferences_Click(object sender, EventArgs e)
422    {
423        InputParDefault();
424    }
425
426    private void InputParDefault()
427    {
428        cbxType.Enabled = true;
429        cbxType.SelectedIndex = 0;
430        _cardSelectionne = null;
431        SelectionCard();
432        btnModifier.Enabled = false;
433        btnSupprimer.Enabled = false;
434        btnAjouter.Enabled = true;
435        tbxAuteur.Text = "";
436        tbxTitre.Text = "";
437        tbxDescription.Text = "";
438        pbxImage.Image = null;
439    }
440
441    private void frmCollectionReferences_Click(object sender,
442        EventArgs e)
443    {
444        InputParDefault();
445    }
446
447    private void cbxLivre_SelectedIndexChanged(object sender,
448        EventArgs e)
449    {
450        if (cbxLivre.SelectedIndex != 0)
451        {
452            UpdateFormView();
453        }
454    }
455}
```

```
450             EtatTousElements(false);
451             cbxLivre.Enabled = true;
452         }
453     else
454     {
455         InputParDefault();
456         tbxAuteur.Enabled = true;
457         tbxTitre.Enabled = true;
458         btnImporterImage.Enabled = true;
459     }
460 }
461
462 private void btnImporterImage_Click(object sender, EventArgs e)
463 {
464     //https://stackoverflow.com/questions/6122984/load-a-
465     //bitmap-image-into-windows-forms-using-open-file-dialog
466     using (OpenFileDialog dlg = new OpenFileDialog())
467     {
468         dlg.Title = "Open Image";
469         dlg.Filter = "Image Files (*.bmp;*.jpg;*.jpeg,*.png) | *.BMP;*.JPG;*.JPEG;*.PNG";
470
471         if (dlg.ShowDialog() == DialogResult.OK)
472         {
473             // Create a new Bitmap object from the picture file on disk,
474             // and assign that to the PictureBox.Image property
475             pbxImage.Image = new Bitmap(dlg.FileName);
476         }
477     }
478
479 private void cbxFiltreType_SelectedIndexChanged(object sender, EventArgs e)
480 {
481     UpdateFormView();
482 }
483
484 private void textBox1_TextChanged(object sender, EventArgs e)
485 {
486
487 }
488
489 private void tbxRecherche_TextChanged(object sender, EventArgs e)
490 {
491     UpdateFormView();
492 }
493
494 private List<CardReference> RechercheReferenceParFiltre(string recherche)
495 {
496     List<CardReference> listCard = new List<CardReference>();
```

```
497         CardReference nouvelleCard;
498         foreach (Reference reference in _references)
499         {
500             if(reference.DescriptionLieu.ToLower().Contains
501                 (recherche.ToLower() || reference.Auteur.ToLower()
502                  .Contains(recherche.ToLower()) ||
503                  reference.NomReference.ToLower().Contains
504                  (recherche.ToLower())))
505             {
506                 switch (reference.IdType)
507                 {
508                     //Livre
509                     case 1:
510                         nouvelleCard = new CardReferenceLivre
511                         (_utilisateur, reference, this);
512                         listCard.Add(nouvelleCard);
513                         break;
514                     //Musique
515                     case 2:
516                         nouvelleCard = new CardReferenceMusique
517                         (reference, this);
518                         listCard.Add(nouvelleCard);
519                         break;
520                     }
521                 }
522             }
523         return listCard;
524     }
525
526     private List<CardReference> RechercheReferenceParFiltre(int
527         idType)
528     {
529         List<CardReference> listCard = new List<CardReference>();
530         foreach (Reference reference in _references)
531         {
532             CardReference nouvelleCard;
533             switch (reference.IdType)
534             {
535                 //Livre
536                 case 1:
537                     if (idType == 1)
538                     {
539                         nouvelleCard = new CardReferenceLivre
540                         (_utilisateur, reference, this);
541                         listCard.Add(nouvelleCard);
542                     }
543             }
544         }
545     }
```

```
541                     break;
542                     //Musique
543                     case 2:
544                         if (idType == 2)
545                         {
546                             nouvelleCard = new CardReferenceMusique      ↵
547                             (reference, this);
548                             listCard.Add(nouvelleCard);
549                         }
550                         break;
551                         //Lieu
552                         case 3:
553                             if (idType == 3)
554                             {
555                                 nouvelleCard = new CardReferenceLieu      ↵
556                                 (reference, this);
557                                 listCard.Add(nouvelleCard);
558                             }
559                             break;
560                         }
561                     return listCard;
562                 }
563
564             private List<CardReference> RechercheReferenceParFiltre(string      ↵
565                 recherche, int idType)
566             {
567                 List<CardReference> listCard = new List<CardReference>();
568                 CardReference nouvelleCard;
569                 foreach (Reference reference in _references)
570                 {
571                     if (reference.DescriptionLieu.ToLower().Contains
572                         (recherche.ToLower() || reference.Auteur.ToLower
573                         () .Contains(recherche.ToLower()) ||
574                         reference.NomReference.ToLower().Contains
575                         (recherche.ToLower())))
576                     {
577                         switch (reference.IdType)
578                         {
579                             //Livre
580                             case 1:
581                                 if (idType == 1)
582                                 {
583                                     nouvelleCard = new CardReferenceLivre      ↵
584                                     (_utilisateur, reference, this);
585                                     listCard.Add(nouvelleCard);
586                                 }
587                                 break;
588
589                             //Musique
590                             case 2:
591                                 if(idType == 2)
```

```
586                     {
587                         nouvelleCard = new CardReferenceMusique ↵
588                         (reference, this);
589                         listCard.Add(nouvelleCard);
590                     }
591
592                     break;
593 //Lieu
594 case 3:
595     if(idType == 3)
596     {
597         nouvelleCard = new CardReferenceLieu      ↵
598         (reference, this);
599         listCard.Add(nouvelleCard);
600     }
601     break;
602 }
603 }
604 return listCard;
605 }
606 }
607 }
608 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : frmConnexion.cs Form
9   * Descs.    : Vue de la connexion
10  *
11  * IMPORTANT : SI VOUS RENCONTREZ DU PROBLEME DU LOGIN, CHANGEZ LA ↵
12    VELEUR DANS LA BASE MyLibrary.Utilisateur.Connecte DE 1 à 0 ↵
13    Si vous avez des problème à générer les solution, ↵
14    veuillez lancer l'application une fois afin que les design ↵
15    s'affichent ↵
16  */
17
18  using MyLibrary.classes;
19  using System;
20  using System.Security.Cryptography;
21  using System.Text;
22  using System.Windows.Forms;
23
24  namespace MyLibrary
25  {
26      public partial class frmConnexion : Form
27      {
28          public frmConnexion()
29          {
30              //initialisation des composants
31              InitializeComponent();
32          }
33
34          private void btnConnexion_Click(object sender, EventArgs e)
35          {
36              //Vérification si les champs sont remplis
37              if(tbxEmail.Text != "" && tbxPassword.Text != "")
38              {
39                  try
40                  {
41                      //Création d'un nouvel utilisateur avec les données ↵
42                      //des champs
43                      var user = new Utilisateur(tbxEmail.Text,
44                      GenererSha1(tbxPassword.Text).ToLower());
45                      //Tentative de connexion à l'API
46                      if (user.TestConnexion())
47                      {
48                          //Affichage de la nouvelle form
49                          frmCollectionLivres collectionLivres = new ↵
50                          frmCollectionLivres(user, this);
51                          collectionLivres.Show();
52                      }
53                      //Si les données de connexion ne correspondent pas
54                  }
55              }
56          }
57      }
58  }
```

```
48             else
49             {
50                 //Message d'erreur
51                 MessageBox.Show("Erreur lors de la connexion", "Attention Reprise", MessageBoxButtons.OK,
52                                 MessageBoxIcon.Error);
53             }
54         catch(Exception ex)
55         {
56             //Message d'erreur
57             MessageBox.Show("Problème interne lors de la connexion : " + Environment.NewLine + ex, "Erreur interne", MessageBoxButtons.OK,
58                                 MessageBoxIcon.Error);
59         }
60     }
61     else
62     {
63         //Message d'erreur
64         MessageBox.Show("Veuillez remplir tous les champs", "Attention Reprise", MessageBoxButtons.OK,
65                                 MessageBoxIcon.Warning);
66     }
67
68     /// <summary>
69     /// Permet de créer un sha1 à partir d'un string
70     /// </summary>
71     /// <param name="text">Text à générer</param>
72     /// <returns>Sha1 du texte</returns>
73     private string GenererSha1(string text)
74     {
75         using (SHA1 sha1Hash = SHA1.Create())
76         {
77             //Conversion en Bytes
78             byte[] sourceBytes = Encoding.UTF8.GetBytes(text);
79             //Création du Sha1
80             byte[] hashBytes = sha1Hash.ComputeHash(sourceBytes);
81             //Convertis le tableau de byte en string
82             return BitConverter.ToString(hashBytes).Replace("-", String.Empty);
83         }
84     }
85
86     //Affichage au clair du mot de passe lors du maintien du bouton d'affichage
87     private void btnAfficherMdp_MouseDown(object sender, MouseEventArgs e)
88     {
89         //Affichage du mot de passe en clair
90         tbxPassword.PasswordChar = '\0';
```

```
91      }
92
93      //Changement de la vue des données afin de masquer le mot de ↵
94      //passe
95      private void btnAfficherMdp_MouseUp(object sender, ↵
96          MouseEventArgs e)
97      {
98          //Cache le mot de passe
99          tbxPassword.PasswordChar = '*';
100     }
101 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : ImageInFile.cs class
9   * Descs.    : Permet de stoquer une image avec ses informations
10  */
11
12 using System;
13 using System.Drawing;
14 using System.Linq;
15
16 namespace MyLibrary
17 {
18     public class ImageInFile
19     {
20         #region variables d'instances
21         private string _nom;
22         private string _extension;
23         private Bitmap _data;
24         #endregion
25
26
27         #region constantes
28         const string DEFAULT_EXTENSION = ".png";
29         #endregion
30
31         #region propriétés
32         public string Nom
33         {
34             get { return _nom; }
35             set { _nom = value; }
36         }
37
38         public string Extension
39         {
40             get { return _extension; }
41             set { _extension = value; }
42         }
43
44         public Bitmap Data
45         {
46             get { return _data; }
47             set { _data = value; }
48         }
49         #endregion
50
51         #region constructeurs
52         /// <summary>
53         /// Permet de stoquer une image avec ses informations
```

```
54     /// </summary>
55     /// <param name="nom">nom de l'image</param>
56     /// <param name="extension">type</param>
57     /// <param name="data">Bitmap de l'image</param>
58     private ImageInFile(string nom, string extension, Bitmap data)
59     {
60         _nom = nom;
61         _extension = extension;
62         _data = data;
63     }
64
65     /// <summary>
66     /// Image avec nom aléatoire
67     /// </summary>
68     /// <param name="extension">type</param>
69     /// <param name="data">Bitmap de l'image</param>
70     public ImageInFile(string extension, Bitmap data) : this
71         (nomAleatoire(), extension, data) { }
72
73     /// <summary>
74     /// Image avec nom aléatoire et sans type (type par défaut
75     /// png)
76     /// </summary>
77     /// <param name="data">Bitmap de l'image</param>
78     public ImageInFile(Bitmap data) : this(DEFAULT_EXTENSION, data) { }
79
80
81
82     #region méthodes
83     /// <summary>
84     /// Image avec nom aléatoire et sans type (type par défaut
85     /// png)
86     /// Source du code :
87     /// https://stackoverflow.com/questions/1344221/how-can-i-generate-random-alphanumeric-strings
88     private static string nomAleatoire()
89     {
90         Random random = new Random();
91         const string chars =
92             "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
93         return new string(Enumerable.Repeat(chars, 8).Select(s => s
94             [random.Next(s.Length)]).ToArray());
95     }
96
97     /// <summary>
98     /// Permet d'enregistrer l'image en local
99     /// </summary>
100    public void SaveBmp()
101    {
```

```
100         _data.Save(_nom + _extension);
101     }
102     #endregion
103 }
104 }
105
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : Livre.cs class
9   * Descs.    : Permet d'encapsuler la table Livres de la base de données
10  */
11
12 namespace MyLibrary.classes
13 {
14     public class Livre
15     {
16         #region Variables d'instances
17         private int _idLivre, _idUtilisateur;
18         private string _titre, _auteur, _nomImage;
19         private ClientRest _clientRest;
20         #endregion
21
22         #region Propriétés
23         public int IdLivre
24         {
25             get { return _idLivre; }
26             set { _idLivre = value; }
27         }
28
29         public int IdUtilisateur
30         {
31             get { return _idUtilisateur; }
32             set { _idUtilisateur = value; }
33         }
34
35         public string Titre
36         {
37             get { return _titre; }
38             set { _titre = value; }
39         }
40
41         public string Auteur
42         {
43             get { return _auteur; }
44             set { _auteur = value; }
45         }
46
47         public string NomImage
48         {
49             get { return _nomImage; }
50             set { _nomImage = value; }
51         }
52     #endregion
53 }
```

```
54     #region Constructeurs
55     /// <summary>
56     /// Permet d'encapsuler la table Livres de la base de données
57     /// </summary>
58     /// <param name="idLivre">int(11)</param>
59     /// <param name="titre">varchar(255)</param>
60     /// <param name="auteur">varchar(100)</param>
61     /// <param name="nomImage">varchar(255)</param>
62     /// <param name="idUtilisateur">int(11)</param>
63     public Livre(int idLivre, string titre, string auteur, string nomImage, int idUtilisateur) ↵
64     {
65         _clientRest = ClientRest.Instance;
66         _idLivre = idLivre;
67         _titre=titre;
68         _auteur=auteur;
69         _nomImage=nomImage;
70         _idUtilisateur=idUtilisateur;
71     }
72 #endregion
73
74     #region Methodes
75     /// <summary>
76     /// Permet à un utilisateur d'envoyer le livre de la class
77     /// </summary>
78     /// <param name="utilisateur">utilisateur qui envoie le livre ↵
79     (Prop : Email, Password) </param>
80     /// <returns>True = code 201, False = erreur</returns>
81     public bool PostLivre(Utilisateur utilisateur)
82     {
83         return _clientRest.AppelSimple("?
84             table=livres&email="+utilisateur.Email ↵
85             +"&password="+utilisateur.Password+"&titre="+_titre ↵
86             +"&auteur="+_auteur+"&nomImage="+_nomImage+"", "POST");
87     }
88
89     /// <summary>
90     /// Permet à l'utilisateur de modifier le livre de la class
91     /// </summary>
92     /// <param name="utilisateur">utilisateur qui modifie le livre ↵
93     (Prop : Email, Password)</param>
94     /// <param name="nouvellesDonnees">Objet livre avec les
95     nouvelles informations</param>
96     /// <returns>True = code 201, False = erreur</returns>
97     public bool PutLivre(Utilisateur utilisateur, Livre
98         nouvellesDonnees)
```

```
99         return _clientRest.AppelSimple("?email=" +  
    utilisateur.Email + "&password=" + utilisateur.Password +  
    "&titre=" + _titre + "&auteur=" + _auteur + "&nomImage=" +  
    _nomImage + "&table=livres&idLivre=" + _idLivre + "",  
    "PUT");  
100     }  
101  
102     /// <summary>  
103     /// Permet de supprimer le livre de la class  
104     /// </summary>  
105     /// <param name="utilisateur">utilisateur qui souhaite  
    supprimer le livre</param>  
106     /// <returns>True = code 201, False = erreur</returns>  
107     public bool DeleteLivre(Utilisateur utilisateur)  
108     {  
109         return _clientRest.AppelSimple("?email=" + utilisateur.Email +  
    "&password=" + utilisateur.Password +  
    "&table=livres&idLivre=" + _idLivre + "", "DELETE");  
110     }  
111     #endregion  
112     }  
113 }  
114
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6
7  namespace MyLibrary
8  {
9      internal static class Program
10     {
11         /// <summary>
12         /// Point d'entrée principal de l'application.
13         /// </summary>
14         [STAThread]
15         static void Main()
16         {
17             Application.EnableVisualStyles();
18             Application.SetCompatibleTextRenderingDefault(false);
19             new frmConnexion().Show();
20             Application.Run();
21         }
22     }
23 }
24 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : References.cs class
9   * Descs.    : Permet d'encapsuler la table References de la base de      ↵
   *               données
10  *           Sert également de class mère aux références livre,          ↵
   *               musique et lieu
11 */
12
13 namespace MyLibrary.classes
14 {
15     public class Reference
16     {
17         #region Variables d'instances
18         private int _idReference, _idType, _livreReference, _idLivre;
19         private string _nomReference, _nomImage, _auteur,
20             _descriptionLieu;
21         #endregion
22
23         #region Propriétés
24         public int IdReference
25         {
26             get { return _idReference; }
27             set { _idReference = value; }
28         }
29
30         public string NomReference
31         {
32             get { return _nomReference; }
33             set { _nomReference = value; }
34         }
35
36         public string NomImage
37         {
38             get { return _nomImage; }
39             set { _nomImage = value; }
40         }
41
42         public string Auteur
43         {
44             get { return _auteur; }
45             set { _auteur = value; }
46         }
47
48         public int IdType
49         {
50             get { return _idType; }
51             set { _idType = value; }
```

```
51     }
52
53     public int LivreReference
54     {
55         get { return _livreReference; }
56         set { _livreReference = value; }
57     }
58
59     public int IdLivre
60     {
61         get { return _idLivre; }
62         set { _idLivre = value; }
63     }
64     public string DescriptionLieu
65     {
66         get { return _descriptionLieu; }
67         set { _descriptionLieu = value; }
68     }
69
70
71 #endregion
72
73 #region Constructeurs
74 /// <summary>
75 /// Permet d'encapsuler la table References de la base de      ↵
76     données
77 /// Sert également de class mère aux références livre, musique    ↵
78     et lieu
79 /// </summary>
80 /// <param name="idReference">int(11)</param>
81 /// <param name="nomReference">varchar(255)</param>
82 /// <param name="nomImage">varchar(255)</param>
83 /// <param name="auteur">varchar(100)</param>
84 /// <param name="idType">int(11)</param>
85 /// <param name="livreReference">int(11) livres, idLivres</      ↵
86     param>
87 /// <param name="idLivre">int(11) livres, IdLivres </param>
88 /// <param name="descriptionLieu">mediumtext</param>
89     public Reference(int idReference, string nomReference, string    ↵
90         nomImage, string auteur, int idType, int livreReference, int    ↵
91         idLivre, string descriptionLieu)
92     {
93         _idReference = idReference;
94         _nomReference = nomReference;
95         _nomImage = nomImage;
96         _auteur = auteur;
97         _idType = idType;
98         _livreReference = livreReference;
99         _idLivre = idLivre;
100        _descriptionLieu = descriptionLieu;
101    }
```

```
99      /// <summary>
100     /// Sert de modèles aux class enfants
101     /// Permet d'envoyer la référence à l'API
102     /// </summary>
103     /// <param name="utilisateur">Utilisateur qui envoie la
104     /// référence</param>
105     /// <returns>Retourne forcément false si elle n'est pas
106     /// override</returns>
107     public virtual bool PostReference(Utilisateur utilisateur)
108     {
109         return false;
110     }
111
112     /// <summary>
113     /// Sert de modèles aux class enfants
114     /// Permet de modifier la référence à l'API
115     /// </summary>
116     /// <param name="utilisateur">Utilisateur qui envoie la
117     /// référence</param>
118     /// <param name="reference">nouvelles données de la référence</
119     /// param>
120     /// <returns>Retourne forcément false si elle n'est pas
121     /// override</returns>
122     public virtual bool PutReference(Utilisateur utilisateur,
123                                     Reference reference)
124     {
125         return false;
126     }
127
128     /// <summary>
129     /// Sert de modèles aux class enfants
130     /// Permet de supprimer la référence à l'API
131     /// </summary>
132     /// <param name="utilisateur">Utilisateur qui supprime la
133     /// référence</param>
134     /// <returns>Retourne forcément false si elle n'est pas
135     /// override</returns>
136     public virtual bool DeleteReference(Utilisateur utilisateur)
137     {
138         return ClientRest.Instance.AppelSimple("?
139             table=references&email=" + utilisateur.Email +
140             "&password=" + utilisateur.Password + "&idReference=" +
141             "_idReference + "", "DELETE");
142     }
143
144     //public Reference
145     #endregion
146 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date     : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe   : I.DA-P4A
7   *
8   * Class    : ReferenceLieu.cs class
9   * Descs.   : Permet de créer une référence de type Lieu
10  */
11
12
13 using MyLibrary.classes;
14
15 namespace MyLibrary
16 {
17     public class ReferenceLieu : Reference
18     {
19         #region constantes
20         private const string DEFAULT_DESCRIPTIONLIEU = "";
21         #endregion
22
23         /// <summary>
24         /// Permet de créer une référence de type Lieu avec la      ↴
25             description
26         /// </summary>
27         /// <param name="idReference">int(11)</param>
28         /// <param name="titre">varchar(255)</param>
29         /// <param name="descriptionLieu">mediumtext</param>
30         /// <param name="idLivre">int(11)</param>
31         public ReferenceLieu(int idReference, string titre, string      ↴
32             descriptionLieu, int idLivre) : base(idReference, titre, "",      ↴
33             "", 3, 0, idLivre, descriptionLieu) { }
34
35         /// <summary>
36         /// Permet de créer une référence de type Lieu sans description
37         /// </summary>
38         /// <param name="idReference">int(11)</param>
39         /// <param name="titre">varchar(255)</param>
40         /// <param name="idLivre">int(11)</param>
41         public ReferenceLieu(int idReference, string titre, int      ↴
42             idLivre) : this(idReference, titre, DEFAULT_DESCRIPTIONLIEU,      ↴
43             idLivre) { }
44
45         /// <summary>
46         /// Permet d'ajouter la référence dans la base par l'API
47         /// </summary>
48         /// <param name="utilisateur">Utilisateur qui ajoute la      ↴
49             référence</param>
50         /// <returns>
51         /// true = 201
52         /// false = erreur
53         /// </returns>
```

```
48     public override bool PostReference(Utilisateur utilisateur)
49     {
50         return ClientRest.Instance.AppelSimple("?
51             table=references&email=" + utilisateur.Email + "&password=" +
52             + utilisateur.Password + "&nomReference=" + NomReference +
53             "&descriptionLieu=" + DescriptionLieu + "&idLivre=" +
54             IdLivre.ToString() + "&idType=" + IdType.ToString() + "", "POST");
55     }
56 
57     /// <summary>
58     /// Permet de modifier une référence dans la base par l'API
59     /// </summary>
60     /// <param name="utilisateur"></param>
61     /// <param name="reference"></param>
62     /// <returns>
63     /// true = 201
64     /// false = erreur
65     /// </returns>
66     public override bool PutReference(Utilisateur utilisateur,
67                                     Reference reference)
68     {
69         return ClientRest.Instance.AppelSimple("?
70             table=references&idReference=" + IdReference.ToString() + " +
71             "&email=" + utilisateur.Email + "&password=" +
72             utilisateur.Password + "&nomReference=" +
73             reference.NomReference + "&descriptionLieu=" +
74             reference.DescriptionLieu + "&idLivre=" + reference.IdLivre +
75             "&idType=" + reference.IdType + "", "PUT");
76     }
77 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : ReferenceLivre.cs class
9   * Descs.    : Permet de créer une référence de type Livre
10  */
11
12 using MyLibrary.classes;
13
14 namespace MyLibrary
15 {
16     public class ReferenceLivre : Reference
17     {
18         /// <summary>
19         /// Permet de créer une référence de type Livre
20         /// </summary>
21         /// <param name="idReference">int(11)</param>
22         /// <param name="livreReference">int(11)</param>
23         /// <param name="idLivre">int(11)</param>
24         public ReferenceLivre(int idReference, string NomImage, int    ↪
25                           livreReference, int idLivre) : base(idReference, "", NomImage,    ↪
26                           "", 1, livreReference, idLivre, "") { }
27
28         /// <summary>
29         /// Permet d'envoyer une référence de type livre dans la table    ↪
30         /// référence (clé étagères)
31         /// </summary>
32         /// <param name="utilisateur">Utilisateur qui envoie la    ↪
33         /// référence</param>
34         /// <returns>
35         /// true = 201
36         /// false = erreur
37         /// </returns>
38         public override bool PostReference(Utilisateur utilisateur)
39         {
40             return ClientRest.Instance.AppelSimple("?
41                 table=references&email=" + utilisateur.Email + "&password="    ↪
42                 + utilisateur.Password + "&idLivre=" + IdLivre.ToString()    ↪
43                 + "&idType=" + IdType.ToString() + "&livreReference=" +    ↪
44                 LivreReference.ToString() + "", "POST");
45         }
46
47         /// <summary>
48         /// Permet de modifier la référence
49         /// </summary>
50         /// <param name="utilisateur">utilisateur qui modifie la    ↪
51         /// référence</param>
52         /// <param name="reference">nouvelles données</param>
53         /// <returns>
```

```
45      /// true = 201
46      /// false = erreur
47      /// </returns>
48      public override bool PutReference(Utilisateur utilisateur,      ↵
        Reference reference)
49      {
50
51          return ClientRest.Instance.AppelSimple("?
  table=references&idReference=" + IdReference.ToString() +      ↵
    "&email=" + utilisateur.Email + "&password=" +
  utilisateur.Password + "&nomReference=" +
  reference.NomReference + "&auteur=" + reference.Auteur +
  "&nomImage=" + reference.NomImage + "&idLivre=" +
  reference.IdLivre + "&idType=" + reference.IdType + "",      ↵
    "PUT");
52      }
53
54      /// <summary>
55      /// Permet à l'utilisateur de modifier le livre de la class
56      /// </summary>
57      /// <param name="utilisateur">utilisateur qui modifie le livre      ↵
        (Prop : Email, Password)</param>
58      /// <param name="nouvellesDonnees">Objet livre avec les
        nouvelles informations</param>
59      /// <returns>True = code 201, False = erreur</returns>
60      public bool PutLivre(Utilisateur utilisateur, Livre      ↵
        nouvellesDonnees)
61      {
62          //Mise à jour des variables d'instances
63          string _auteur = nouvellesDonnees.Auteur;
64          string _titre = nouvellesDonnees.Titre;
65          string _nomImage = nouvellesDonnees.NomImage;
66          int _idLivre = nouvellesDonnees.IdLivre;
67
68
69          return ClientRest.Instance.AppelSimple("?email=" +
  utilisateur.Email + "&password=" + utilisateur.Password +
  "&titre=" + _titre + "&auteur=" + _auteur + "&nomImage=" +
  _nomImage + "&table=livres&idLivre=" + _idLivre + "",      ↵
    "PUT");
70      }
71  }
72 }
73 }
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur     : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : ReferenceMusique.cs class
9   * Descs.    : Permet de créer une référence de type musique
10  */
11
12 using MyLibrary.classes;
13
14 namespace MyLibrary
15 {
16     public class ReferenceMusique : Reference
17     {
18         /// <summary>
19         /// Permet de créer une référence de type musique
20         /// </summary>
21         /// <param name="idReference">int(11)</param>
22         /// <param name="nomImage">varchar(255)</param>
23         /// <param name="titre">varchar(255)</param>
24         /// <param name="auteur">varchar(100)</param>
25         /// <param name="idLivre">int(11)</param>
26         public ReferenceMusique(int idReference, string nomImage, string ↪
27             titre, string auteur, int idLivre) : base(idReference, titre, ↪
28             nomImage, auteur, 2, 0, idLivre, ""){ }
29
30         /// <summary>
31         /// Permet d'envoyer la référence à l'API
32         /// </summary>
33         /// <param name="utilisateur">Utilisateur qui envoie la      ↪
34             référence</param>
35         /// <returns>
36         /// true = code 201
37         /// false = erreur
38         public override bool PostReference(Utilisateur utilisateur)
39         {
40             return ClientRest.Instance.AppelSimple(?           ↪
41                 table=references&email=" + utilisateur.Email + "&password=" ↪
42                     + utilisateur.Password + "&nomReference=" + NomReference + ↪
43                     "&auteur=" + Auteur + "&nomImage=" + NomImage +           ↪
44                     "&idLivre=" + IdLivre + "&idType=" + IdType+ "", "POST");
45         }
46
47         /// <summary>
48         /// Permet de modifier une référence par l'API
49         /// </summary>
50         /// <param name="utilisateur">Utilisateur qui modifie la      ↪
51             référence</param>
52         /// <param name="reference">Nouvelles données de la référence</ ↪
```

```
        param>
46    /// <returns>
47    /// true = code 201
48    /// false = erreur
49    /// </returns>
50    public override bool PutReference(Utilisateur utilisateur,
51                                Reference reference)
52    {
53        return ClientRest.Instance.AppelSimple("?
54        table=references&idReference=" + IdReference.ToString() +
55        "&email=" + utilisateur.Email + "&password=" +
56        utilisateur.Password + "&nomReference=" +
57        reference.NomReference + "&auteur=" + reference.Auteur +
58        "&nomImage=" + reference.NomImage + "&idLivre=" +
59        reference.IdLivre + "&idType=" + reference.IdType + "",
60        "PUT");
61    }
62}
63}
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : Type.cs class
9   * Descs.    : Permet d'encapsuler la table Types de la base de données
10  */
11
12 namespace MyLibrary.classes
13 {
14     public class Type
15     {
16         #region Variables d'instances
17         private int _idType;
18         private string _nomType;
19         #endregion
20
21         #region Propriétés
22         public int IdType
23         {
24             get { return _idType; }
25             set { _idType = value; }
26         }
27
28         public string NomType
29         {
30             get { return _nomType; }
31             set { _nomType = value; }
32         }
33         #endregion
34
35         #region Constructeurs
36         /// <summary>
37         /// Permet d'encapsuler la table Types de la base de données
38         /// </summary>
39         /// <param name="idType">int(11)</param>
40         /// <param name="nomType">varchar(255)</param>
41         public Type(int idType, string nomType)
42         {
43             _idType = idType;
44             _nomType = nomType;
45         }
46         #endregion
47     }
48 }
49
```

```
1  /* Projet    : MyLibrary - TPI 2022
2   * Version   : 1.0
3   * Date      : 18.05.2022
4   *
5   * Auteur    : Karel V. Svoboda
6   * Classe    : I.DA-P4A
7   *
8   * Class     : Type.cs class
9   * Descs.    : Permet d'encapsuler la table Utilisateurs de la base de      ↵
   données
10  */
11
12 namespace MyLibrary.classes
13 {
14     public class Utilisateur
15     {
16         #region Variables d'instances
17         private int _idUtilisateur;
18         private string _email, _password;
19         private ClientRest _clientRest;
20         #endregion
21
22         #region Constantes
23         private const int DEFAULT_IDUTILISATEUR = 0;
24         #endregion
25
26         #region Propriétées
27         public int IdUtilisateur
28         {
29             get { return _idUtilisateur; }
30             set { _idUtilisateur = value; }
31         }
32
33         public string Email
34         {
35             get { return _email; }
36             set { _email = value; }
37         }
38
39         public string Password
40         {
41             get { return _password; }
42             set { _password = value; }
43         }
44
45         #endregion
46
47         #region Constructeurs
48         /// <summary>
49         /// Permet d'encapsuler la table Utilisateurs de la base de      ↵
   données
50         /// </summary>
51         /// <param name="idUtilisateur">int(11)</param>
```

```
52     /// <param name="email">varchar(255)</param>
53     /// <param name="password">varchar(255)</param>
54     public Utilisateur(int idUtilisateur, string email, string      ↴
55         password)
55     {
56         _clientRest = ClientRest.Instance;
57         _idUtilisateur = idUtilisateur;
58         _email = email;
59         _password = password;
60     }
61
62     /// <summary>
63     /// Permet d'encapsuler la table Utilisateurs de la base de      ↴
63     /// données sans idUtilisateur
64     /// </summary>
65     /// <param name="email">varchar(255)</param>
66     /// <param name="password">varchar(255)</param>
67     public Utilisateur(string email, string password) : this      ↴
67         (DEFAULT_IDUTILISATEUR, email, password){ }
68 #endregion
69
70     #region Méthodes
71     /// <summary>
72     /// Permet de faire une tentative de connexion avec les données      ↴
72     /// de la classe
73     /// </summary>
74     /// <returns>
75     /// true = code 200 (connexion ok et l'utilisateur est connecté      ↴
75     /// dans la base)
76     /// false = erreur ou mauvaise données de connexion erronées
77     /// </returns>
78     public bool TestConnexion()
79     {
80         return _clientRest.AppelSimple("?session=connexion&email=" + ↪
80             _email + "&password=" + _password + "", "get");
81     }
82
83     /// <summary>
84     /// Permet à l'utilisateur de se déconnecter
85     /// L'UTILISATEUR DOIT SE RECONNECTER S'IL VEUT REFAIRE DES      ↴
85     /// REQUÊTES à L'API
86     /// </summary>
87     /// <returns>
88     /// true = code 200 (déconnexion ok)
89     /// false = erreur
90     /// </returns>
91     public bool Deconnexion()
92     {
93         return _clientRest.AppelSimple("?session=deconnexion&email=" ↪
93             + _email + "&password=" + _password + "", "get");
94     }
95
96 #endregion
```

```
97     }
98 }
99
```

```
1 <?php
2 /* Projet : API_MyLibrary
3    Auteur : Svoboda Karel Vilém
4    Desc. : API qui permet de gérer la base de données MyLibrary
5    Date : 18.05.2022
6    Version : 1
7 */
8
9 //autorisation des sources externes
10 header('Access-Control-Allow-Origin: *');
11 //définition du type d'application
12 header('Content-Type: application/json');
13
14 //Connexion PDO
15 require("models/myLibraryPDO.php");
16 $mydatabase = new MyLibrary();
17
18 //require("models/sqlFunctions.php");
19
20 //importation des objets
21 require("objects/livre.php");
22 require("objects/reference.php");
23 require("objects/type.php");
24 require("objects/utilisateur.php");
25
26 $response = null;
27
28 //vérification si l'utilisateur a transmit ses données
29 if (isset($_GET["email"]) && $_GET["password"]) {
30
31     //Switch qui permet déterminer le type de requête de l'utilisateur
32     switch ($_SERVER['REQUEST_METHOD']) {
33         case 'GET':
34             //maipulation de la session de l'utilisateur
35             if (isset($_GET["session"])) {
36                 switch ($_GET["session"]) {
37                     //connexion
38                     case "connexion":
39                         if ($mydatabase->statusConnexionUtilisateur
($_GET["email"]) == 0) {
40
41                             $user = new Utilisateur(0, $_GET["email"],  ↵
$_GET["password"], 0);
42
43                             //Si les données sont trouvées dans la base
44                             if ($mydatabase->testConnexion($user)) {
45                                 http_response_code(200);
46                                 $response = array(
47                                     "200" => "La session est activée",
48                                     );
49
50                             //récupération des données de  ↵
l'utilisateur
```

...ml\ProjetsWeb\MyLibrary\src\API_MyLibrary\index.php 2

```
51             $user = $mydatabase->recevoirUtilisateurParEmail($user->getEmail());
52
53             $mydatabase->connexionUtilisateur($user);
54             //Retour réponse positive et ajout des données de l'utilisateur
55             $response = array_merge($response,
56             $mydatabase->recevoirUtilisateurParEmail($user->getEmail())->returnArrayForJSON());
57             } else {
58                 http_response_code(400);
59                 $response = array(
60                     "400" => "Fausses données de connexion",
61                 );
62             }
63             //Si l'utilisateur essaie de se connecter malgré le fait qu'une session est déjà active
64             else {
65                 http_response_code(401);
66                 $response = array(
67                     "401" => "La session de cet utilisateur est déjà active",
68                 );
69             }
70
71             break;
72             //déconnexion de l'utilisateur
73             case "deconnexion":
74                 if ($mydatabase->statusConnexionUtilisateur($_GET["email"]) == 1) {
75                     $user = $mydatabase->recevoirUtilisateurParEmail($_GET["email"]);
76                     $mydatabase->deconnexionUtilisateur($user);
77
78                     http_response_code(201);
79                     $response = array(
80                         "201" => "l'utilisateur est déconnectée",
81                     );
82                 }
83                 else{
84                     http_response_code(401);
85                     $response = array(
86                         "401" => "L'utilisateur n'est pas connecté",
87                     );
88                 }
89                 break;
90             default:
91                 $response = array(
```

```
92                     "400" => "Point d'entrée inconnu",
93                 );
94             break;
95         }
96     }
97
98     //manipulation de la table livres
99     if (isset($_GET["table"])) {
100         $user = new Utilisateur(0, $_GET["email"], $_GET
101                               ["password"], 0);
102         if ($mydatabase->statusConnexionUtilisateur($_GET
103                               ["email"]) == 1) {
104             switch ($_GET["table"]) {
105                 case "livres":
106                     if (isset($_GET["recherche"])) {
107                         $user->setIdUtilisateur($mydatabase-
108 >recevoirUtilisateurParEmail($_GET["email"])->getIdUtilisateur());
109                         http_response_code(200);
110                         $response = array();
111                         foreach ($mydatabase-
112 >rechercheLivreParAuteurOuTitre($user, $_GET
113                               ["recherche"]) as $unLivre) {
114                             array_push($response, $unLivre-
115 >returnArrayForJSON());
116                         }
117                         }else if(isset($_GET["idLivre"])){
118                             http_response_code(200);
119                             $response = array();
120                             $response = $mydatabase-
121 >rechercheLivreParIdLivre($_GET["idLivre"])->returnArrayForJSON();
122                         }
123                         else if(isset($_GET["tri"])){
124                             http_response_code(200);
125                             switch($_GET["tri"]){
126                                 case "dernier":
127                                     http_response_code(201);
128                                     $response = array();
129                                     $response = $mydatabase-
130 >dernierLivreUtilisateur($mydatabase-
131 >recevoirUtilisateurParEmail($_GET["email"]))->getIdLivre();
132                                     break;
133                                 }
134                             }
135                         else {
136                             $user->setIdUtilisateur($mydatabase-
137 >recevoirUtilisateurParEmail($_GET["email"])->getIdUtilisateur());
138                             //récupération des livres en fonction
139                             //de l'idUtilisateur
140                             http_response_code(200);
```

...ml\ProjetsWeb\MyLibrary\src\API_MyLibrary\index.php

```
130                     $response = array();
131                     foreach ($mydatabase-
132                         >livresParUtilisateur($user) as $unLivre) {
133                             array_push($response, $unLivre-
134                         >returnArrayForJSON());
135                     }
136                     break;
137                 case "references":
138                     if ($mydatabase->statusConnexionUtilisateur =>
139                         ($_GET["email"]) == 1) {
140                         if (isset($_GET["idLivre"])) {
141                             http_response_code(200);
142                             $response = array();
143                             foreach($mydatabase-
144                                 >referencesParLivre(new Livre($_GET["idLivre"], "", "",
145                                     "", "", 0)) as $reference){
146                                     array_push($response,
147                                         $reference->returnArrayForJSON());
148                                 }
149                             }
150                         break;
151                     }
152                     case "types":
153                     if ($mydatabase->statusConnexionUtilisateur =>
154                         ($_GET["email"]) == 1) {
155                         http_response_code(200);
156                         $response = array();
157                         foreach ($mydatabase->tousTypes() as
158                             $unType) {
159                             array_push($response, $unType-
160                         >returnArrayForJSON());
161                         }
162                         break;
163                     default:
164                         break;
165                     }
166                 }
167             break;
168         case "POST":
169             //if($mydatabase->testConnexion(new Utilisateur()))
170             if ($mydatabase->statusConnexionUtilisateur($_GET["email"]) =>
171                 == 1) {
```

```
173     switch($_GET["table"]){
174         case 'livres':
175             if (isset($_GET["titre"]) && isset($_GET
176 ["auteur"]) && isset($_GET["nomImage"])){
177                 //Création d'un objet Livre à partir des
178                 données dans le get
179                 $livre = new Livre(0, $_GET["titre"], $_GET
180 ["auteur"], $_GET["nomImage"], $mydatabase-
181 >recevoirUtilisateurParEmail($_GET["email"])->getIdUtilisateur());
182                     $mydatabase->ajouterLivre($livre);
183                     http_response_code(201);
184                     $response = $livre->returnArrayForJSON();
185                 }
186             else{
187                 $response = array(
188                     "401" => "Veuillez renseigner tous les
189 points d'entrées",
190                 );
191             }
192             break;
193         case 'references':
194             if(isset($_GET["idType"])){
195                 switch($_GET["idType"]){
196                     //livre
197                     case "1":
198                         if (isset($_GET["nomReference"]) &&
199                             isset($_GET["auteur"]) && isset($_GET["nomImage"])
200 && isset($_GET["idLivre"])){
201                             //Création d'un livre à partir
202                             des données de la référence
203                             $livre = new Livre(0, $_GET
204 ["nomReference"], $_GET["auteur"], $_GET["nomImage"],
205                             $mydatabase->recevoirUtilisateurParEmail($_GET
206 ["email"])->getIdUtilisateur());
207                             $mydatabase->ajouterLivre
208                             ($livre);
209                             //récupération du dernier
210                             idLivre et passage dans l'objet grâce au SET
211                             $livre->setIdLivre($mydatabase-
212 >dernierLivreUtilisateur($mydatabase-
213 >recevoirUtilisateurParEmail($_GET["email"]))->getIdLivre());
214
215                             $referenceLivre = $livre-
216 >ReferenceDeLivre(new Livre($_GET["idLivre"], "", "", "",
217 "", 0));
218
219                             //envoie de la référence à la
220                             base
221                             $mydatabase-
222 >ajouterReferenceLivre($referenceLivre);
223                             http_response_code(201);
224             }
```

...ml\ProjetsWeb\MyLibrary\src\API_MyLibrary\index.php 6

```
205             $response = $referenceLivre->returnArrayForJSON();
206         }
207         else if(isset($_GET["livreReference"]) && isset($_GET["idLivre"])){
208             $referenceLivre = new Reference(0, "", "", "", $_GET["idType"], $_GET["livreReference"], $_GET["idLivre"], "");
209             //envoie de la référence à la base
210             $mydatabase->ajouterReferenceLivre($referenceLivre);
211             http_response_code(201);
212             $response = $referenceLivre->returnArrayForJSON();
213         }
214     }
215     else{
216         http_response_code(401);
217     }
218     break;
219     //musique
220     case "2":
221         if(isset($_GET["nomImage"]) && isset($_GET["nomReference"]) && isset($_GET["auteur"]) && isset($_GET["idLivre"])){
222             $reference = new Reference(0, $_GET["nomReference"], $_GET["nomImage"], $_GET["auteur"], $_GET["idType"], 0, $_GET["idLivre"], "");
223             $mydatabase->ajouterReferenceMusique($reference);
224             http_response_code(201);
225             $response = $reference->returnArrayForJSON();
226         }
227     else{
228         http_response_code(401);
229     }
230     break;
231     //lieu
232     case '3':
233         if (isset($_GET["nomReference"]) && isset($_GET["descriptionLieu"]) && isset($_GET["idLivre"])) {
234             $reference = new Reference(0, $_GET["nomReference"], "", "", $_GET["idType"], 0, $_GET["idLivre"], $_GET["descriptionLieu"]);
235             $mydatabase->ajouterReferenceLieu($reference);
236             http_response_code(201);
237             $response = $reference->returnArrayForJSON();
238         }
239     }
240 }
```

```
239             >returnArrayForJSON();
240         }
241         else{
242             http_response_code(401);
243         }
244         break;
245     default:
246     }
247 }
248 // < ! > ajouter sécurité de isset
249
250         break;
251 }
252
253
254 } else {
255     http_response_code(403);
256     $response = array(
257         "Veuillez vous connecter pour poursuivre"
258     );
259 }
260 break;
261 case "PUT":
262 if ($mydatabase->statusConnexionUtilisateur($_GET["email"]) =>
263 == 1) {
264     switch($_GET["table"]){
265         case 'livres':
266             if (isset($_GET["idLivre"]) && isset($_GET
267 ["titre"]) && isset($_GET["auteur"]) && isset($_GET
268 ["nomImage"])) {
269                 //Création d'un objet Livre à partir des
270                 données dans le get
271                 $livre = new Livre($_GET["idLivre"], $_GET
272 ["titre"], $_GET["auteur"], $_GET["nomImage"],
273 $mydatabase->recevoirUtilisateurParEmail($_GET
274 ["email"])->getIdUtilisateur());
275                 $mydatabase->modifierLivre($livre);
276                 http_response_code(201);
277                 $response = $livre->returnArrayForJSON();
278             }
279             break;
280         case 'references':
281             if(isset($_GET["idReference"])){
282                 if(isset($_GET["idType"])){
283                     switch($_GET["idType"]){
284                         case "1":
285                             if (isset($_GET
286 ["livreReference"]) && isset($_GET["titre"]) && isset
287 ($_GET["auteur"]) && isset($_GET["nomImage"])) {
288                                 //Création d'un objet Livre
289                                 à partir des données dans le get
290             }
```

```
...ml\ProjetsWeb\MyLibrary\src\API_MyLibrary\index.php 8
281                                     $livre = new Livre($_GET ↵
282                                         ["livreReference"], $_GET["titre"], $_GET["auteur"], ↵
283                                         $_GET["nomImage"], $mydatabase- ↵
284                                         >recevoirUtilisateurParEmail($_GET["email"])- ↵
285                                         >getIdUtilisateur()); ↵
286                                     $mydatabase->modifierLivre ↵
287                                         ($livre); ↵
288                                     http_response_code(201); ↵
289                                     $response = $livre- ↵
290                                         >returnArrayForJSON(); ↵
291                                         } ↵
292                                         break; ↵
293                                         //musique ↵
294                                         case "2": ↵
295                                         if(isset($_GET["idReference"]) ↵
296                                         && isset($_GET["nomImage"]) && isset($_GET ↵
297                                         ["nomReference"]) && isset($_GET["auteur"]) && isset ↵
298                                         ($_GET["idLivre"])){ ↵
299                                         $reference = new Reference ↵
300                                         ($_GET["idReference"], $_GET["nomReference"], $_GET ↵
301                                         ["nomImage"], $_GET["auteur"], $_GET["idType"], 0, ↵
302                                         $_GET["idLivre"], ""); ↵
303                                         $mydatabase- ↵
304                                         >modifierReferenceMusique($reference); ↵
305                                         http_response_code(201); ↵
306                                         $response = $reference- ↵
307                                         >returnArrayForJSON(); ↵
308                                         } ↵
309                                         else{ ↵
310                                         http_response_code(401); ↵
311                                         } ↵
312                                         break; ↵
313                                         //lieu ↵
314                                         case '3': ↵
315                                         if (isset($_GET ↵
316                                         ["nomReference"]) && isset($_GET["descriptionLieu"]) ↵
317                                         && isset($_GET["idLivre"])) { ↵
318                                         $reference = new Reference ↵
319                                         ($_GET["idReference"], $_GET["nomReference"], "", "", ↵
320                                         $_GET["idType"], 0, $_GET["idLivre"], $_GET ↵
321                                         ["descriptionLieu"]); ↵
322                                         $mydatabase- ↵
323                                         >modifierReferenceLieu($reference); ↵
324                                         http_response_code(201); ↵
325                                         $response = $reference- ↵
326                                         >returnArrayForJSON(); ↵
327                                         } ↵
328                                         else{ ↵
329                                         http_response_code(404); ↵
330                                         } ↵
331                                         //modifierReferenceLieu ↵
332                                         break; ↵
333                                         }
```

```
313                     default:
314                         }
315
316                         }
317                         }
318                         break;
319                     }
320                 }else {
321                     http_response_code(403);
322                     $response = array(
323                         "Veuillez vous connecter pour poursuivre"
324                     );
325                 }
326
327
328                     break;
329             case "DELETE":
330                 if (isset($_GET["table"])){
331                     switch($_GET["table"]){
332                         case "livres":
333                             $livre = new Livre($_GET["idLivre"], "", "", "",
334                                         "", 0);
335                             $mydatabase->supprimerLivre($livre);
336                             break;
337                         case "references":
338                             if(isset($_GET["idReference"])){
339                                 $reference = new Reference($_GET
340                                     ["idReference"], null, null, null, null, null, null,
341                                     null, null);
342                                 $mydatabase->supprimerReference
343                                     ($reference);
344                                 http_response_code(201);
345
346                             }
347                             break;
348                         }
349                     } else {
350                         http_response_code(400);
351                         $response = array(
352                             "400" => "Données de connexion manquantes",
353                             "Headers à ajouter" => "email / password"
354                         );
355                     }
356
357 echo json_encode($response);
358
```

```
1 <?php
2     /* Projet : API_MyLibrary
3      Auteur : Svoboda Karel Vilém
4      Class : Livre
5      Desc. : Permet de répliquer les données dans la table Livres
6 */
7
8     class Livre
9     {
10         //Variables d'instances
11         private $_idLivre, $_titre, $_auteur, $_nomImage,
12             $_idUtilisateur;
13
14         //Propriétés
15         public function getIdLivre(){
16             return $this->_idLivre;
17         }
18         public function setIdLivre(int $idLivre){
19             $this->_idLivre = $idLivre;
20         }
21
22         public function getTitre(){
23             return $this->_titre;
24         }
25         public function setTitre(string $titre){
26             $this->_titre = $titre;
27         }
28
29         public function getAuteur(){
30             return $this->_auteur;
31         }
32         public function setAuteur(string $auteur){
33             $this->_auteur = $auteur;
34         }
35
36         public function getNomImage(){
37             return $this->_nomImage;
38         }
39         public function setNomImage(string $nomImage){
40             $this->_nomImage = $nomImage;
41         }
42
43         public function getIdUtilisateur(){
44             return $this->_idUtilisateur;
45         }
46         public function setIdUtilisateur(int $idUtilisateur){
47             $this->_idUtilisateur = $idUtilisateur;
48         }
49
50         //Constructeur
51
52         /** Permet de créer un livre
53         *
```

```
53     * @param integer int(11) $idLivre
54     * @param string varchar(100) $titre
55     * @param string varchar(255) $auteur
56     * @param string varchar(255) $nomImage
57     * @param integer int(11) $idUtilisateur
58     */
59     function __construct(int $idLivre, string $titre, string
60                           $auteur, string $nomImage, int $idUtilisateur)
61     {
62         $this->_idLivre = $idLivre;
63         $this->_titre = $titre;
64         $this->_auteur = $auteur;
65         $this->_nomImage = $nomImage;
66         $this->_idUtilisateur = $idUtilisateur;
67     }
68     //Fonctions
69
70     /** Permet de retourner un array avec les informations de
71      l'objet
72      *
73      * @return array array formé pour JSON
74      */
74     public function returnArrayForJSON(){
75         return array(
76             "idLivre" => $this->_idLivre,
77             "titre" => $this->_titre,
78             "auteur" => $this->_auteur,
79             "nomImage" => $this->_nomImage,
80             "idUtilisateur" => $this->_idUtilisateur
81         );
82     }
83
84     public function ReferenceDeLivre(Livre $livre){
85         return new Reference(0, "", "", "", 1, $this->_idLivre,
86                             $livre->getIdLivre(), "");
87     }
88 ?>
```

```
1 <?php
2 /* Projet : API_MyLibrary
3    Auteur : Svoboda Karel Vilém
4    Class : Livre
5    Desc. : Permet de répliquer les données dans la table Livres
6 */
7
8 class MyLibrary
9 {
10    // <!> À modifier lors de la mise en production <!>
11    //Données de connexion à la base
12    private $_serveur = 'mysql:host=localhost';
13    private $_bdd = 'dbname=MyLibrary';
14    private $_user = 'MyLibraryAPI';
15    private $_mdp = 'r6.4>NRM`tC~y*gN';
16
17    private $_connexionPDO = null;
18
19    //Utilisateurs
20    private $_retourUtilisateurParEmailPassword = null;
21    private $_retourUtilisateurParEmail = null;
22    private $_connexionUtilisateur = null;
23    private $_deconnexionUtilisateur = null;
24    private $_statusConnexionUtilisateur = null;
25
26    //Livres
27    private $_afficherLivreParIdUtilisateur = null;
28    private $_rechercheLivreParIdLivre = null;
29    private $_ajouterLivre = null;
30    private $_rechercheLivreParAuteurOuTitre = null;
31    private $_dernierLivreUtilisateur = null;
32
33    //Références
34    private $_referencesParLivre = null;
35    private $_ajouterReference = null;
36
37
38    private $_ajouterReferenceMusique;
39    private $_modifierReferenceMusique;
40
41    private $_ajouterReferenceLivre;
42
43    private $_ajouterReferenceLieu;
44    private $_modifierReferenceLieu;
45
46    //Types
47    private $_tousTypes = null;
48
49    //Constructeur
50    public function __construct()
51    {
52        try{
53            $this->_connexionPDO = new PDO($this->_serveur.';'.$this-
```

...MyLibrary\src\API_MyLibrary\models\myLibraryPDO.php 2

```
        >_bdd, $this->_user, $this->_mdp, array(
54            PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
55            PDO::ATTR_PERSISTENT => true
56        ));
57
58        //Prepare statements
59
60        //Utilisateurs
61        $sql = "SELECT * FROM Utilisateurs WHERE email=:email AND
62            password=:password";
63        $this->_retourUtilisateurParEmailPassword = $this-
64            >_connexionPDO->prepare($sql);
65        $this->_retourUtilisateurParEmailPassword->setFetchMode
66            (PDO::FETCH_ASSOC, 'MyLibrary');

67        $sql = "SELECT * FROM Utilisateurs WHERE email=:email";
68        $this->_retourUtilisateurParEmail = $this->_connexionPDO-
69            >prepare($sql);
70        $this->_retourUtilisateurParEmail->setFetchMode
71            (PDO::FETCH_ASSOC, 'MyLibrary');

72        $sql = "UPDATE Utilisateurs SET connecte=1 WHERE
73            idUtilisateur=:idUtilisateur";
74        $this->_connexionUtilisateur = $this->_connexionPDO-
75            >prepare($sql);
76        $this->_connexionUtilisateur->setFetchMode
77            (PDO::FETCH_ASSOC, 'MyLibrary');

78        $sql = "UPDATE Utilisateurs SET connecte=0 WHERE
79            idUtilisateur=:idUtilisateur";
80        $this->_deconnexionUtilisateur = $this->_connexionPDO-
81            >prepare($sql);
82        $this->_deconnexionUtilisateur->setFetchMode
83            (PDO::FETCH_ASSOC, 'MyLibrary');

84        //Livres
85        $sql = "INSERT INTO Livres (titre, auteur, nomImage,
86            idUtilisateur) VALUES
87            (:titre, :auteur, :nomImage, :idUtilisateur);";
88        $this->_ajouterLivre = $this->_connexionPDO->prepare($sql);
89        $this->_ajouterLivre->setFetchMode(PDO::FETCH_ASSOC,
90            'MyLibrary');

91        $sql = "SELECT * FROM Livres WHERE idLivre=:idLivre";
92        $this->_rechercheLivreParIdLivre = $this->_connexionPDO-
93            >prepare($sql);
```

...MyLibrary\src\API_MyLibrary\models\myLibraryPDO.php 3

```
88     $this->_rechercheLivreParIdLivre->setFetchMode      ↵
89         (PDO::FETCH_ASSOC, 'MyLibrary');
```

```
90     $sql = "UPDATE Livres SET titre=:titre, auteur=:auteur,      ↵
91         nomImage=:nomImage WHERE idLivre=:idLivre;"      ↵
92     $this->_modifierLivre = $this->_connexionPDO->prepare      ↵
93         ($sql);      ↵
94     $this->_modifierLivre->setFetchMode(PDO::FETCH_ASSOC,      ↵
95         'MyLibrary');
```

```
96     $sql = "DELETE FROM Livres WHERE idLivre=:idLivre;"      ↵
97     $this->_supprimerLivre = $this->_connexionPDO->prepare      ↵
98         ($sql);      ↵
99     $this->_supprimerLivre->setFetchMode(PDO::FETCH_ASSOC,      ↵
100        'MyLibrary');
```

```
101    $sql = "SELECT * FROM Livres WHERE      ↵
102        idUtilisateur=:idUtilisateur";      ↵
103    $this->_afficherLivreParIdUtilisateur = $this-      ↵
104        >_connexionPDO->prepare($sql);      ↵
105    $this->_afficherLivreParIdUtilisateur->setFetchMode      ↵
106        (PDO::FETCH_ASSOC, 'MyLibrary');
```

```
107    $sql = "SELECT * FROM Livres WHERE auteur LIKE :auteur OR      ↵
108        titre LIKE :titre AND idUtilisateur=:idUtilisateur";      ↵
109    $this->_rechercheLivreParAuteurOuTitre = $this-      ↵
110        >_connexionPDO->prepare($sql);      ↵
111    $this->_rechercheLivreParAuteurOuTitre->setFetchMode      ↵
112        (PDO::FETCH_ASSOC, 'MyLibrary');
```

```
113 //dernierIdLivre
114 $sql = "SELECT * FROM Livres WHERE      ↵
115     idUtilisateur=:idUtilisateur ORDER BY idLivre DESC LIMIT      ↵
116     1";      ↵
117 $this->_dernierLivreUtilisateur = $this->_connexionPDO-      ↵
118     >prepare($sql);      ↵
119 $this->_dernierLivreUtilisateur->setFetchMode      ↵
120     (PDO::FETCH_ASSOC, 'MyLibrary');
```

```
121 //Références
122 $sql = "SELECT * FROM MyLibrary.References WHERE      ↵
123     idLivre=:idLivre";      ↵
124 $this->_referencesParLivre = $this->_connexionPDO->prepare      ↵
125         ($sql);      ↵
126 $this->_referencesParLivre->setFetchMode(PDO::FETCH_ASSOC,      ↵
127         'MyLibrary');
```

```
128 $sql = "DELETE FROM MyLibrary.References WHERE      ↵
129     idReference=:idReference;"      ↵
130 $this->_supprimerReference = $this->_connexionPDO->prepare      ↵
131         ($sql);      ↵
132 $this->_supprimerReference->setFetchMode(PDO::FETCH_ASSOC,      ↵
133         'MyLibrary');
```

```
119          // Référence type musique
120          $sql = "INSERT INTO MyLibrary.References (nomReference,
121                                         nomImage, idType, auteur, idLivre) VALUES
122                                         (:nomReference, :nomImage, :idType, :auteur, :idLivre);";
123          $this->_ajouterReferenceMusique = $this->_connexionPDO-
124              >prepare($sql);
125          $this->_ajouterReferenceMusique->setFetchMode
126              (PDO::FETCH_ASSOC, 'MyLibrary');
127
128          // Référence type Livre
129          $sql = "INSERT INTO MyLibrary.References (livreReference,
130                                         idType, idLivre) VALUES
131                                         (:livreReference, :idType, :idLivre);";
132          $this->_ajouterReferenceLivre = $this->_connexionPDO-
133              >prepare($sql);
134          $this->_ajouterReferenceLivre->setFetchMode
135              (PDO::FETCH_ASSOC, 'MyLibrary');
136
137          $sql = "UPDATE MyLibrary.References SET
138              nomReference=:nomReference, idType=:idType,
139              idLivre=:idLivre, descriptionLieu=:descriptionLieu
140              WHERE idReference=:idReference;";
141
142          //Types
143          $sql = "SELECT * FROM Types";
144          $this->_tousTypes = $this->_connexionPDO->prepare($sql);
145          $this->_tousTypes->setFetchMode(PDO::FETCH_ASSOC,
146                                         'MyLibrary');
147
148      }
149      catch (PDOException $e) {
```

```
149         //Affichage de l'erreur dans les logis
150         error_log("Erreur PDO ! : " . $e->getMessage());
151         die();
152     }
153 }
154
155 /** Permet de connecter un utilisateur
156 *  <!> Methode vulnérable à utiliser strictement après une connexion correcte de l'utilisateur (testConnexion) <!>
157 *
158 * @Utilisateur $utilisateur utilisateur à connecter
159 *
160 * @return null
161 */
162 public function connexionUtilisateur(Utilisateur $utilisateur){
163     $this->_connexionUtilisateur->execute(array(':idUtilisateur' => $utilisateur->getIdUtilisateur()));
164 }
165
166 /** Permet de déconnecter un utilisateur
167 *
168 * @Utilisateur $utilisateur utilisateur à déconnecter
169 *
170 * @return null
171 */
172 public function deconnexionUtilisateur(Utilisateur $utilisateur){
173     $this->_deconnexionUtilisateur->execute(array('idUtilisateur' => $utilisateur->getIdUtilisateur()));
174 }
175
176 /** Permet de faire une tentative de connexion
177 *
178 * @Utilisateur $utilisateur utilisateur à connecter
179 *
180 * @return bool connexion correcte = true, connexion fausse = false
181 */
182 public function testConnexion(Utilisateur $utilisateur){
183     $this->_retourUtilisateurParEmailPassword->execute(array(
184         ':email' => $utilisateur->getEmail(), ':password' => $utilisateur->getPassword()));
185     return isset($this->_retourUtilisateurParEmailPassword->fetchAll()[0][0]);
186 }
187
188 /** Retourne un utilisateur par email
189 *
190 * @string email de l'utilisateur à rechercher
191 *
192 * @return Utilisateur utilisateur avec par email
193 */
194 public function recevoirUtilisateurParEmail(string $email){
195     $this->_retourUtilisateurParEmail->execute(array(':email' => $email));
```

```
195     $resultat = $this->_retourUtilisateurParEmail->fetchAll();
196     return new Utilisateur($resultat[0][0], $resultat[0][1],
197                             $resultat[0][2], $resultat[0][3]);
198 }
199
200 /**
201 * @string email utilisateur
202 *
203 * @return bool true = connecté, flase = pas connecté
204 */
205 public function statusConnexionUtilisateur(string $email){
206     $this->_statusConnexionUtilisateur->execute(array(':email' =>
207             $email));
208     $resultat = $this->_statusConnexionUtilisateur->fetchAll();
209     return $resultat[0][0];
210 }
211
212 /**
213 * @Livre livre à ajouter
214 * @return null
215 */
216 public function ajouterLivre(Livre $livre){
217     $this->_ajouterLivre->execute(array(':titre' => $livre-
218             >getTitre(), ':auteur' => $livre->getAuteur(), ':nomImage' =>
219             $livre->getNomImage(), ':idUtilisateur' => $livre-
220             >getIdUtilisateur()));
221 }
222
223 /**
224 * @Livre livre AVEC les données modifiées sauf les IDs
225 * @return null
226 */
227 public function modifierLivre(Livre $livre){
228     $this->_modifierLivre->execute(array(':idLivre'=>$livre-
229             >getIdLivre(),':titre' => $livre->getTitre(), ':auteur' =>
230             $livre->getAuteur(), ':nomImage' => $livre->getNomImage()));
231 }
232
233 /**
234 * @Livre livre à supprimer
235 * @return null
236 */
237 public function supprimerLivre(Livre $livre){
238     $this->_supprimerLivre->execute(array(':idLivre'=>$livre-
239             >getIdLivre()));
240 }
```

```
239     /** Permet de retourner tous les livres d'utilisateur
240      *
241      * @Utilisateur utilisateur en question
242      * @return array(Livres) retourne la liste des Livres
243     */
244     public function livresParUtilisateur(Utilisateur $utilisateur){
245         $this->_afficherLivreParIdUtilisateur->execute(array
246             (:idUtilisateur' => $utilisateur->getIdUtilisateur()));
247         $resultat = $this->_afficherLivreParIdUtilisateur->fetchAll();
248         $arrayLivre = array();
249         foreach($resultat as $unLivre){
250             array_push($arrayLivre, new Livre($unLivre[0], $unLivre[1],
251                 $unLivre[2], $unLivre[3], $unLivre[4]));
252         }
253         return $arrayLivre;
254     }
255
256     /** Permet de récupérer le dernier livre d'un utilisateur
257      *
258      * @Utilisateur utilisateur
259     */
260     public function dernierLivreUtilisateur(Utilisateur $utilisateur){
261         $this->_dernierLivreUtilisateur->execute(array(':idUtilisateur' =>
262             $utilisateur->getIdUtilisateur()));
263         $resultat = $this->_dernierLivreUtilisateur->fetchAll();
264         $arrayLivre = array();
265         foreach($resultat as $unLivre){
266             array_push($arrayLivre, new Livre($unLivre[0], $unLivre[1],
267                 $unLivre[2], $unLivre[3], $unLivre[4]));
268         }
269         return $arrayLivre[0];
270     }
271
272     /** Permet de faire une recherche grâce à un filtre personnalisé.
273      * Le programme va chercher dans la base les éléments de
274      * l'utilisateur avec les filtres en question sous les tables Auteur
275      * et Titre
276      *
277      * @Utilisateur $utilisateur utilisateur qui fait la requête
278      * @string chaîne de caractères à rechercher
279     */
280     public function rechercheLivreParAuteurOuTitre(Utilisateur
281         $utilisateur, string $recherche){
282         $this->_rechercheLivreParAuteurOuTitre->execute(array
283             (:idUtilisateur' => $utilisateur->getIdUtilisateur(),
284             ':auteur' => "%".$recherche."%", ':titre' => "%".
285                 $recherche."%"));
286         $resultat = $this->_rechercheLivreParAuteurOuTitre->fetchAll();
287         //Création d'une instance de Livre pour chaque donnée trouvée
288         //et retour du résultat dans un array
289         $arrayLivre = array();
290         foreach($resultat as $unLivre){
291             array_push($arrayLivre, new Livre($unLivre[0], $unLivre[1],
```

```
280         }
281     return $arrayLivre;
282 }
283
284 public function tousTypes(){
285     $this->_tousTypes->execute();
286     $resultat = $this->_tousTypes->fetchAll();
287
288     $arrayTitre = array();
289     foreach($resultat as $unType){
290         array_push($arrayTitre, new Type($unType[0], $unType[1]));
291     }
292     return $arrayTitre;
293 }
294
295 public function referencesParLivre(Livre $livre){
296     $this->_referencesParLivre->execute(array(':idLivre' => $livre->
297         getIdLivre()));
298     $resultat = $this->_referencesParLivre->fetchAll();
299     $arrayReference = array();
300     foreach($resultat as $uneReference){
301         array_push($arrayReference, new Reference($uneReference[0], -->
302             $uneReference[1], $uneReference[2], $uneReference[3], -->
303             $uneReference[4], $uneReference[5], $uneReference[6], -->
304             $uneReference[7]));
305     }
306     return $arrayReference;
307 }
308
309 public function rechercheLivreParIdLivre(int $idLivre){
310     $this->_rechercheLivreParIdLivre->execute(array(':idLivre' => -->
311         $idLivre));
312     $resultat = $this->_rechercheLivreParIdLivre->fetchAll();
313     return new Livre($resultat[0][0], $resultat[0][1], $resultat[0] -->
314         [2], $resultat[0][3], $resultat[0][4]);
315
316     public function ajouterReference(Reference $reference){
317         error_log($this->_ajouterReference->execute(array(
318             ':nomReference' => $reference->getNomReference(),
319             ':nomImage' => $reference->getNomImage(), ':auteur' =>
320             $reference->getAuteur(), ':idType' => $reference->getIdType()
321             (), ':livreReference' => $reference->getLivreReference(),
322             ':resume' => $reference->getDescriptionLieu())));
323     }
324
325     public function ajouterReferenceMusique(Reference $reference){
326         $this->_ajouterReferenceMusique->execute(array(':nomReference' =>
327             $reference->getNomReference(), ':nomImage' => $reference-
328             >getNomImage(), ':auteur' => $reference->getAuteur(),
329             ':idLivre' => $reference->getIdLivre(), ':idType' =>
330             $reference->getIdType()));
```

```
317     }
318
319     public function modifierReferenceMusique(Reference $reference){
320         $this->_modifierReferenceMusique->execute(array(':idReference' =>
321             => $reference->getIdReference(), ':nomReference' =>
322             $reference->getNomReference(), ':nomImage' => $reference-
323             >getNomImage(), ':auteur' => $reference->getAuteur(),
324             ':idLivre' => $reference->getIdLivre(), ':idType' =>
325             $reference->getIdType()));
326     }
327
328     public function supprimerReference(Reference $reference){
329         $this->_supprimerReference->execute(array(':idReference' =>
330             $reference->getIdReference()));
331     }
332
333     public function ajouterReferenceLivre(Reference $reference){
334         $this->_ajouterReferenceLivre->execute(array(':livreReference' =>
335             => $reference->getLivreReference(), ':idLivre' => $reference-
336             >getIdLivre(), ':idType' => $reference->getIdType()));
337     }
338 }
339 ?>
```

```
1 <?php
2     /* Projet : API_MyLibrary
3      Auteur : Svoboda Karel Vilém
4      Class : Reference
5      Desc. : Permet de répliquer les données dans la table
6          References
7 */
8
9     class Reference
10    {
11        //Variables d'instances
12        private $_idReference, $_nomReference, $_nomImage, $_auteur,
13            $_idType, $_livreReference, $_idLivre, $_descriptionLieu;
14
15        //Propriétés
16        public function getIdReference(){
17            return $this->_idReference;
18        }
19        public function setIdReference($idReference){
20            $this->_idReference = $idReference;
21        }
22
23        public function getNomReference(){
24            return $this->_nomReference;
25        }
26        public function setNomReference($nomReference){
27            $this->_nomReference = $nomReference;
28        }
29
30        public function getNomImage(){
31            return $this->_nomImage;
32        }
33        public function setNomImage($nomImage){
34            $this->_nomImage = $nomImage;
35        }
36
37        public function getAuteur(){
38            return $this->_auteur;
39        }
40        public function setAuteur($auteur){
41            $this->_auteur = $auteur;
42        }
43
44        public function getIdType(){
45            return $this->_idType;
46        }
47        public function setIdType($idType){
48            $this->_idType = $idType;
49        }
50
51        public function getLivreReference(){
52            return $this->_livreReference;
53        }
```

```
52     public function setLivreReference($LivreReference){  
53         $this->_livreReference = $LivreReference;  
54     }  
55  
56     public function getIdLivre(){  
57         return $this->_idLivre;  
58     }  
59     public function setIdLivre($idLivre){  
60         $this->_idLivre = $idLivre;  
61     }  
62  
63     public function getDescriptionLieu(){  
64         return $this->_descriptionLieu;  
65     }  
66     public function setDescriptionLieu($descriptionLieu){  
67         $this->_descriptionLieu = $descriptionLieu;  
68     }  
69  
70     //Constructeur  
71  
72     /** Permet de créer une Reference  
73      *  
74      * @param integer int(11) $idReference  
75      * @param string varchar(100) $nomReference  
76      * @param string varchar(255) $nomImage  
77      * @param string varchar(100) $auteur  
78      * @param int(11) $idType  
79      * @param integer int(11) $livreReference  
80      * @param integer int(11) $idUtilisateur  
81      */  
82     function __construct($idReference, $nomReference, $nomImage,      ↗  
83                         $auteur, $idType, $livreReference, $idLivre,      ↗  
84                         $descriptionLieu)  
85     {  
86         $this->_idReference = $idReference;  
87         $this->_nomReference = $nomReference;  
88         $this->_nomImage = $nomImage;  
89         $this->_auteur = $auteur;  
90         $this->_idType = $idType;  
91         $this->_livreReference = $livreReference;  
92         $this->_idLivre = $idLivre;  
93         $this->_descriptionLieu = $descriptionLieu;  
94     }  
95  
96     //Fonctions  
97  
98     /** Permet de retourner un array avec les informations de      ↗  
99       l'objet  
100      *  
101      * @return array array formé pour JSON  
102      */  
103     public function returnArrayForJSON() : array  
104     {
```

```
102         return array(
103             "idReference" => $this->_idReference,
104             "nomReference" => $this->_nomReference,
105             "nomImage" => $this->_nomImage,
106             "auteur" => $this->_auteur,
107             "idType" => $this->_idType,
108             "livreReference" => $this->_livreReference,
109             "idLivre" => $this->_idLivre,
110             "descriptionLieu" => $this->_descriptionLieu
111         );
112     }
113 }
114 ?>
```

```
1 <?php
2     /* Projet : API_MyLibrary
3      Auteur : Svoboda Karel Vilém
4      Class : Type
5      Desc. : Permet de répliquer les données dans la table Types
6 */
7
8     class Type
9     {
10         //Variables d'instances
11         private $_idType, $_nomType;
12
13         //Propriétés
14         public function getIdType(){
15             return $this->_idType;
16         }
17         public function setIdType($idType){
18             $this->_idType = $idType;
19         }
20
21         public function getNomType(){
22             return $this->_nomType;
23         }
24         public function setNomType($nomType){
25             $this->_nomType = $nomType;
26         }
27
28         //Constructeur
29
30         /** Permet de créer un Type de référence
31          *
32          * @param integer int(11) $idType
33          * @param string varchar(255) $nomType
34          */
35         function __construct(int $idType, string $nomType)
36         {
37             $this->_idType = $idType;
38             $this->_nomType = $nomType;
39         }
40
41         //Fonctions
42
43         /** Permet de retourner un array avec les informations de l'objet
44          *
45          * @return array array formé pour JSON
46          */
47         public function returnArrayForJSON(){
48             return array(
49                 "idType" => $this->_idType,
50                 "nomType" => $this->_nomType,
51             );
52         }

```

53 }

54 ?>

```
1 <?php
2     /* Projet : API_MyLibrary
3      Auteur : Svoboda Karel Vilém
4      Class : Type
5      Desc. : Permet de répliquer les données dans la table Utilisateurs
6 */
7
8     class Utilisateur
9     {
10         //Variables d'instances
11         private $_idUtilisateur, $_email, $_password, $_connecte;
12
13         //Propriétés
14         public function getIdUtilisateur(){
15             return $this->_idUtilisateur;
16         }
17         public function setIdUtilisateur(int $idUtilisateur){
18             $this->_idUtilisateur = $idUtilisateur;
19         }
20
21         public function getEmail(){
22             return $this->_email;
23         }
24         public function setEmail(string $email){
25             $this->_email = $email;
26         }
27
28         public function getPassword(){
29             return $this->_password;
30         }
31         public function setPassword(string $password){
32             $this->_password = $password;
33         }
34
35         public function getConnecte(){
36             return $this->_connecte;
37         }
38         public function setConnecte(bool $connecte){
39             $this->_connecte = $connecte;
40         }
41
42         //Constructeur
43
44         /** Permet de créer un Utilisateur
45          *
46          * @param integer int(11) $idUtilisateur
47          * @param string varchar(255) $email
48          * @param string varchar(255) $password
49          */
50         function __construct(int $idUtilisateur, string $email, string $password, bool $connecte)
51     {
```

```
52         $this->_idUtilisateur = $idUtilisateur;
53         $this->_email = $email;
54         $this->_password = $password;
55         $this->_connecte = $connecte;
56     }
57
58     //Fonctions
59
60     /** Permet de retourner un array avec les informations de      ↵
       l'objet
61     *
62     * @return array array formé pour JSON
63     */
64     public function returnArrayForJSON(){
65         return array(
66             "idUtilisateur" => $this->_idUtilisateur,
67             "email" => $this->_email,
68             "password" => $this->_password,
69             "connecte" => $this->_connecte
70         );
71     }
72 }
73 ?>
```