## ligne horizontale



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 librarie 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**](#_a3p2trjsi9dq) **2**

[**MyLibrary**](#_aqv5ctcsn2g7) **2**

[Situation de départ](#_1l1dob79cp5j) 2

[Mise en oeuvre](#_l9jvy69e1na2) 2

[Résultat](#_gpxsafc7lke) 2

[**Table des matières**](#_bds9z63cwztr) **3**

[**1.0 Introduction**](#_92op49rhe0hn) **7**

[**2.0 Rappel du cahier des charges**](#_tvtijdq9p8dj) **7**

[2.1 Organisation](#_9ws1pmupemk) 7

[2.2 Restrictions](#_vm8y7p2bjn1x) 7

[2.3 Livrables](#_lfjurkfpzyr9) 7

[2.5 Tâches](#_c7wrjsvvohty) 8

[2.5.1 API](#_7xxdce33d0hi) 8

[2.6.1 Application C#](#_zepmj2a36dn9) 9

[**3.0 Analyse fonctionnelle**](#_rmxoomjvfi6f) **11**

[3.1 Fonctionnalitées](#_vagi80opbu7h) 11

[3.1.1 API](#_mthssdkzdqjf) 11

[Points d’entrées de gestion de session](#_ecfseibvyg5q) 11

[Connexion](#_hukhjscowru3) 11

[Déconnexion](#_rbtxf5q391fk) 11

[Points d’entrées de type POST](#_3xc0cr436b5d) 11

[Tous les livres de l’utilisateur](#_847u4gnpizm5) 11

[recherche personnalisée](#_9y45ma9y9w9i) 12

[Par IdLivre](#_21a5m6k2i1e8) 12

[dernier livre](#_8bhl5ymx774r) 12

[Références par idLivre](#_jurckwyix3bb) 13

[Ajouter un livre](#_pfv4zeko7vv5) 13

[Ajouter une référence de type Livre](#_witzbigck1s0) 14

[Ajouter une référence de type musique](#_5r03u8889bn9) 14

[Ajouter une référence de type lieu](#_kitpyi99b4sg) 15

[Commandes de type PUT](#_xexodfws4xtb) 15

[Modifier un livre](#_p5gqsmbamstd) 15

[Modifier une référence de type Livre](#_htymbi9dlkss) 16

[Modifier une référence de type Musique](#_ohd1g8iz0vv8) 16

[Modifier une référence de type Lieu](#_cj4cab2mbwf4) 17

[Points d’entrées de type DELETE](#_i12gtqhaat98) 17

[3.1.2 Application C#](#_awqgywi1zuf9) 17

[3.1.2.1 Connexion](#_nlhq8ci202mv) 18

[3.1.2.2 CollectionLivres](#_nq1kzz6xm1vt) 19

[3.1.2.2 CollectionReferences](#_wtnaflqpt1kx) 21

[3.2 Méthodologie](#_tpfhovvwe1sd) 24

[3.2 Base de données](#_jtbyvrpgp9tg) 25

[3.2.1 Contraintes](#_s8uk2pg5vcq1) 25

[3.2.2 Diagramme](#_ygzymqnef9hj) 25

[3.2.2 Tables](#_ys7enu1116qb) 26

[3.2.2.1 Type](#_4r7jmik6tkwe) 26

[3.2.2.2 Référence](#_u3wleg3bnotz) 26

[3.2.2.3 Livres](#_8pu6u1ijp9cd) 26

[3.2.2.4 Utilisateurs](#_sdinbh782qc9) 27

[3.2.3 Sql](#_ey9fklj5klk) 27

[3.2.3 Utilisateur de la base](#_tambstoshnbu) 32

[**4.0 Analyse organique**](#_z95ejbauviez) **33**

[4.1 Architecture du projets](#_apkpsppadtu) 33

[4.2 Système de sécurité de l’API](#_9y8472baivu1) 34

[4.3 Connexion](#_j7l3144drbf6) 35

[4.X Héritage des Card](#_r3q21n7u5npk) 36

[4.X Héritage des références](#_qceu5hil4jvu) 37

[4.3 Technologies utilisées](#_aj2zvmwl0oi1) 38

[**5.0 Interfaces**](#_odekltg4qllh) **39**

[5.1 Maquettes de l’application C#](#_40u3z1phvvqf) 39

[5.1.1 Connexion](#_jwwy1p1xxdru) 39

[5.1.2 Collection de livres](#_pgtup18vdqkd) 39

[5.1.3 Collection de références](#_66jo4erukytk) 40

[**6.0 Architecture du projet**](#_lzsfmhefiaa8) **41**

[Architecture générale](#_ba44zq1pw7md) 41

[API PHP](#_7q2vcih2g4rn) 41

[Passage des données](#_g5fn4j9i774b) 42

[Api Rest](#_ekh0arf7s489) 42

[MyLibrary](#_317qxw33z15k) 43

[index.php](#_uwhxe144buf9) 43

[Class encapsulées Livre, Reference, Type et Utilisateur](#_ga9nzwwt8pvx) 43

[Application C#](#_gk1f7748jx1y) 44

[Class ClientRest](#_p4w3kjf3krt2) 45

[Class Card](#_1pb11f6dv6p2) 45

[Class CardLivre](#_93gai19av1px) 46

[CardReference](#_qgmzg7mhllyf) 47

[CardReferenceLieu](#_wh5a2xe3smiy) 47

[CardReferenceLivre](#_rbfkb4cd7h01) 48

[CardReferenceMusique](#_2jec8gab1mv7) 48

[Class ImageInFiles.cs](#_h594gxwwis6q) 49

[Class ReferenceLieu](#_76jlk9r1gcdg) 50

[Class ReferenceLivre](#_u1uxwrt2gv8p) 50

[Class ReferenceMusique](#_bbx3zo3ihug1) 51

[Class Livre](#_eajd4erkqxxs) 51

[Class Reference](#_anoosqi1wxbx) 52

[Class Type](#_6497xby11osh) 54

[Class Utilisateur](#_yg7902zblu2j) 54

[frmConnexion](#_psyzox5zpkpz) 55

[FrmCollectionLivres](#_gl9xi3co3fb5) 57

[FrmCollectionReferences](#_gxw4n9t7mt3a) 59

[**7.0 Tests**](#_gry19vwdhgps) **61**

[7.1 Scénario de tests](#_cvr84bnvjf5x) 61

[2.6.1 API](#_l77iteu7eezu) 61

[2.6.1 Application C#](#_v0360xvwcn0t) 62

[7.2 Evolution des tests](#_vgbinp98amih) 65

[**8.0 Conclusion**](#_88dtbr4c4da) **65**

[8.1 Difficultés rencontrées](#_e3zb6ga6q1jr) 65

[8.2 Améliorations possibles](#_ktukdfasblqf) 66

[8.3 Bilan personnel](#_hkw47x9bzeg) 66

[**9.0 Annexes**](#_vlula2u8107i) **66**

[9.1 Sources](#_4ydxuy1tk96) 66

[**10.0 Code source**](#_imabw8ks2twb) **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** | 1. Requête de type GET 2. En cas de succès, la données de l’utilisateur *connecte* dans la base de données change de 0 à 1 |

| **Titre** | Connexion unique |
| --- | --- |
| **Description** | Lors de la connexion, l’API vérifie la donnée *connecte* dans la base de donnée et autorise un nouvelle connexion ou pas en conséquence |

| **Titre** | CRUD sur la table Livres |
| --- | --- |
| **Description** | 1. Pouvoir afficher les livres 2. Pouvoir ajouter un livre 3. Pouvoir modifier un livre 4. Pouvoir supprimer un livre |

| **Titre** | Récupération dynamique de livres |
| --- | --- |
| **Description** | Pouvoir récupérer un livre par   1. Son auteur 2. Sa description 3. Par l’utilisateur qu’il l’a ajouté |

| **Titre** | CRUD sur la table References |
| --- | --- |
| **Description** | 1. Pouvoir afficher les références 2. Pouvoir ajouter un référence 3. Pouvoir modifier un référence 4. Pouvoir supprimer un référence |

| **Titre** | Récupération dynamique de références |
| --- | --- |
| **Description** | Pouvoir récupérer une référence par   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 *Utilisateur* avec ses informations   1. idUtilisateur 2. Email 3. Password (sha1) |

| **Titre** | Création dynamique des *Card* 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 *Card* (Objet Windows forms hérité de Panel) créés dynamiquement. 3. Chaque *Card* est ensuite affichés dans une List |

| **Titre** | Recherche de *Card Livres* |
| --- | --- |
| **Description** | Permet de faire une recherche dans la liste des *Card* qui contiennent un :   * Auteur donné * Titre donné |

| **Titre** | CRUD sur les Livres via les *Card* |
| --- | --- |
| **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 *Card* Livre |
| --- | --- |
| **Description** | Suite à l’appuie d’un bouton *Référence* d’une *Card* 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 *Card* de différentes types   1. Musique 2. Livre 3. Lieu |

| **Titre** | Filtrage des *Card* référence |
| --- | --- |
| **Description** | Affichage des card demandés par   1. Type 2. Texte |

| **Titre** | CRUD sur les références via les *Card* |
| --- | --- |
| **Description** | Grâce un formulaire, l’application doit   1. Pouvoir ajouter une références 2. Pouvoir modifier une références 3. Pouvoir supprimer une références 4. Message de confirmation lors de la suppression |

# 3.0 Analyse fonctionnelle

## 3.1 Fonctionnalitées

### 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 | deconnexion |
| 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 |

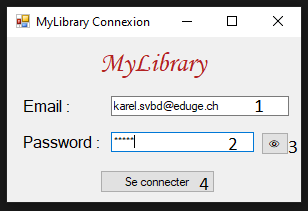
Supprimer une référence

| **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

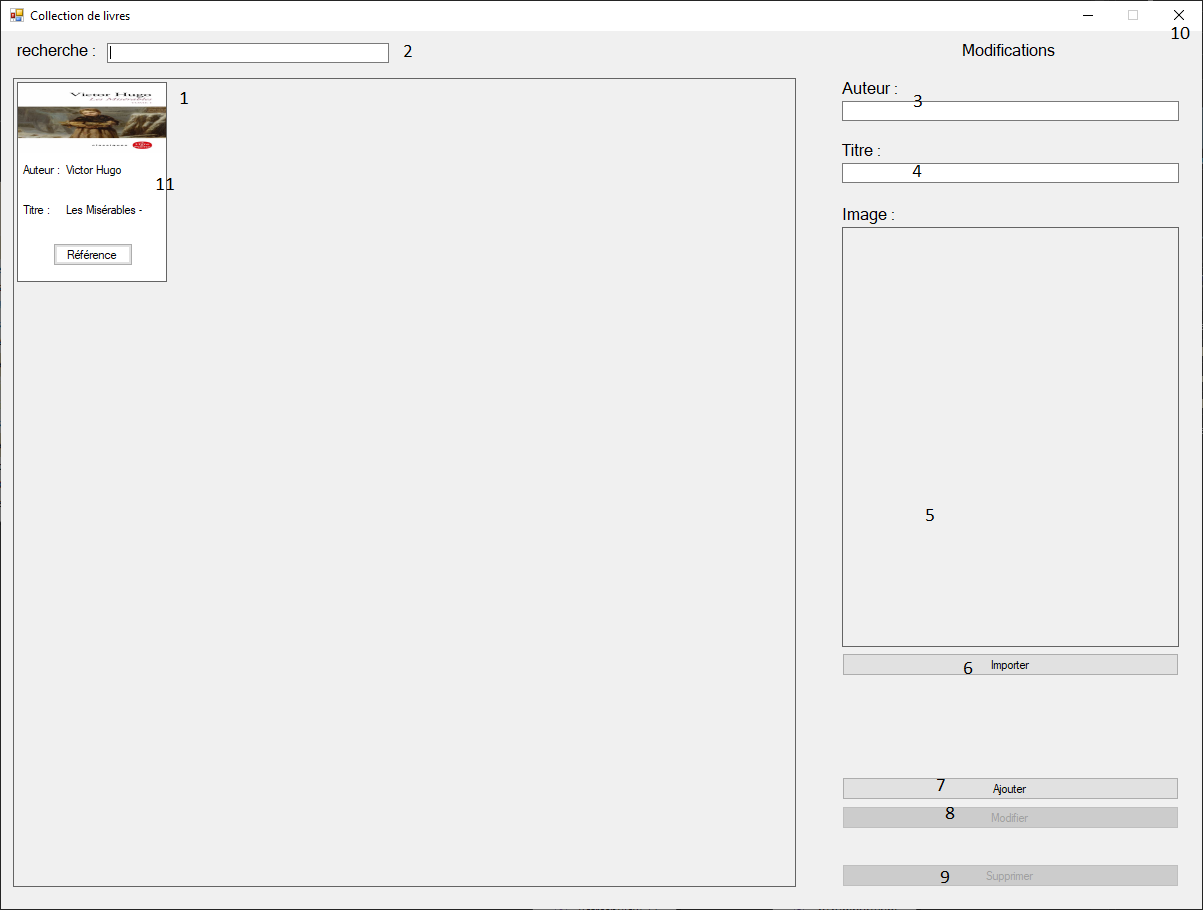
frmConnexion



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



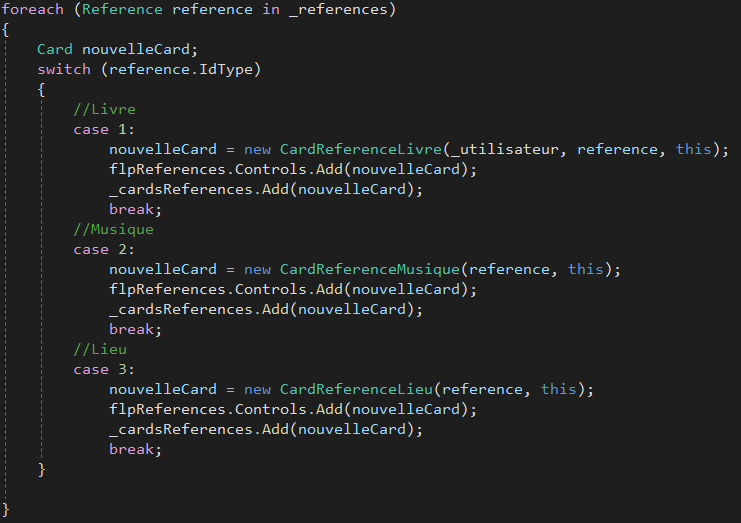
1. Composant flowlayoutpanel qui contient la liste des Objets CardLivre qui affiche les livres de l’utilisateur
2. Recherche personnalisée. Recherche une chaîne de caractères dans l’auteur et le titre des livres et retourne le résultat dans la liste. L’application se met à jour à chaque changement de 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 (connecte 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

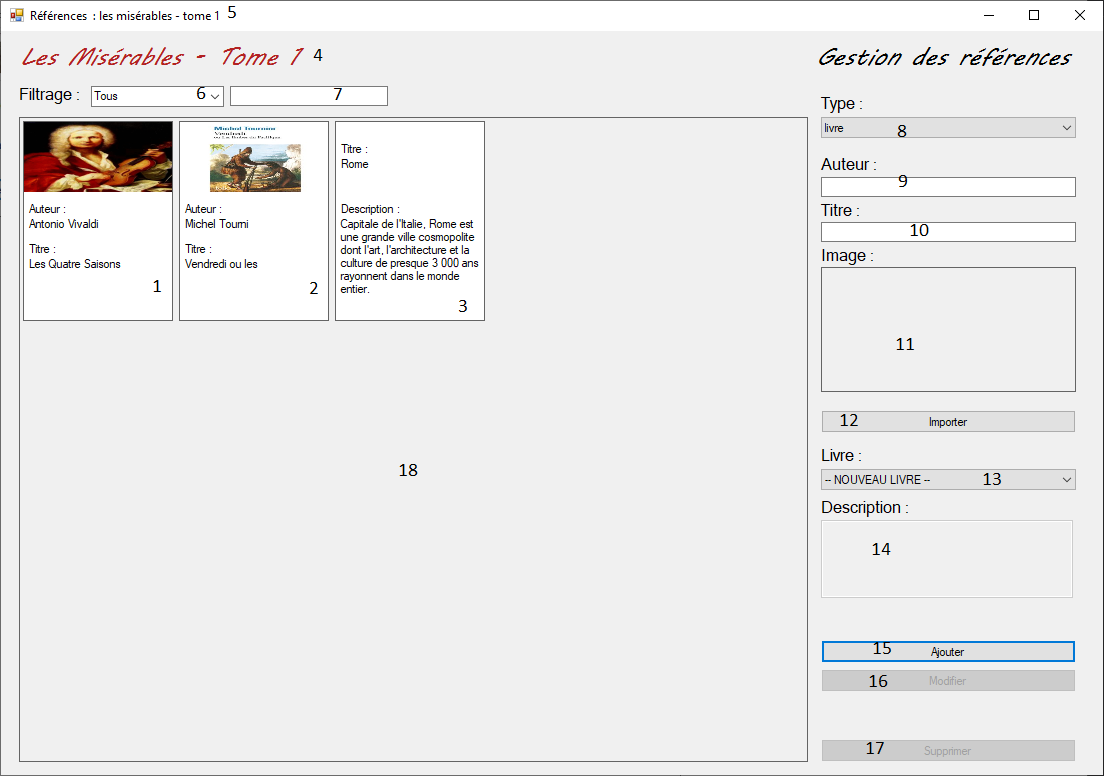
Permet à l'utilisateur d'interagir avec les références d’un livre. S’ouvre après une pression du bouton référence d’une CardLivre dans la frmCollectionLivres.

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.



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.

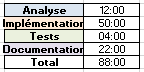
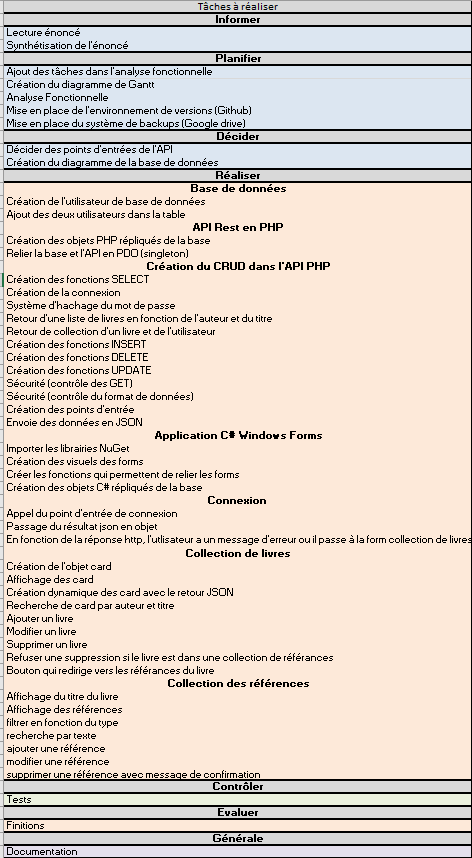


1. CardReferenceMusique, hérite de cardReference. Affiche l’image, l’auteur et le titre de l’objet de l’objet de type Reference.
2. CardReferenceLivre, hérite de cardReference. Affiche l’image, l’auteur et le titre de l’objet de l’objet de type Reference qui est créé dynamiquement après récupération 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.
   1. Tous (affiche tous les types)
   2. Livres (affiche uniquement les références de type Livre)
   3. Musique (affiche uniquement les références de type Musique)
   4. 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.
   1. Type Livre : Auteur, Titre, Image, Livre (n13)
   2. Type Musique : Auteur, Titre, Image
   3. 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.



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

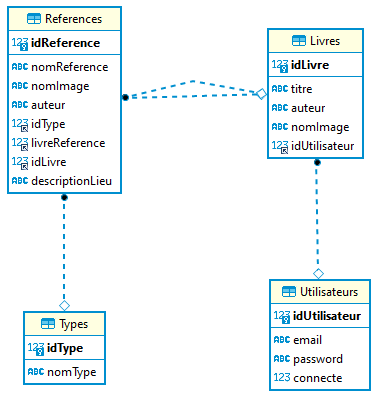


Diagramme de la base de données v1.0

### 

### 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 live 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,'amir.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 \*/;

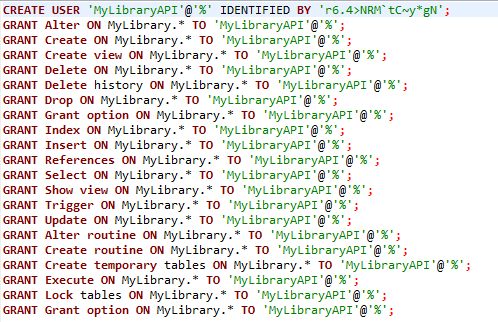
/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!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*



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*

## 

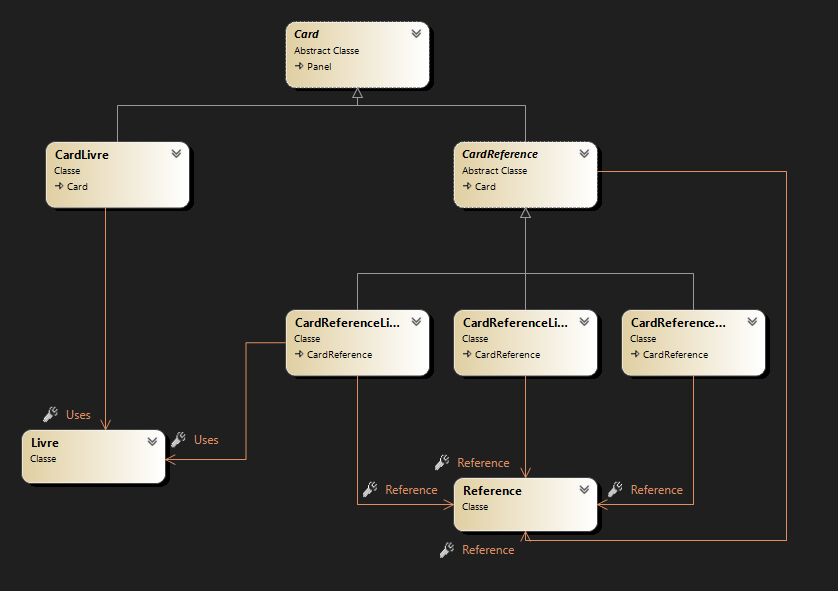
## 4.X Héritage des Card

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*

## 

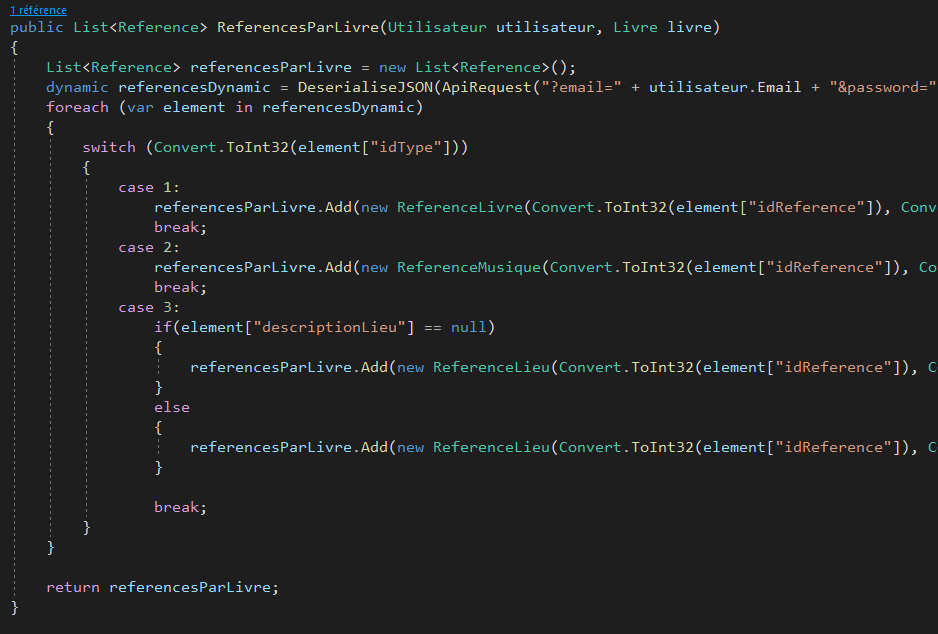
## 4.X Héritage des références

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*



*Génération dynamique des références en fonction du type*

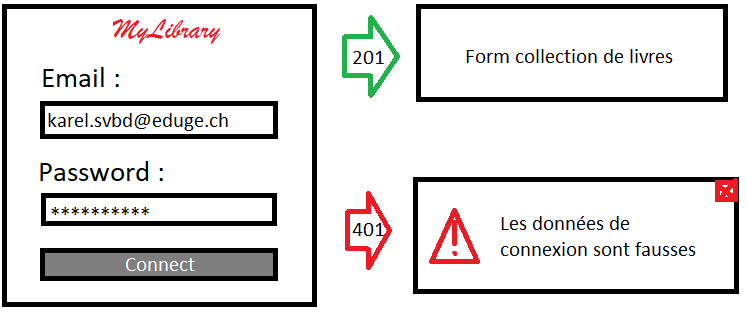
## 4.3 Technologies utilisées

| **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+e555772db77ca828b02b4bd547c318387f11d01f |
| **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

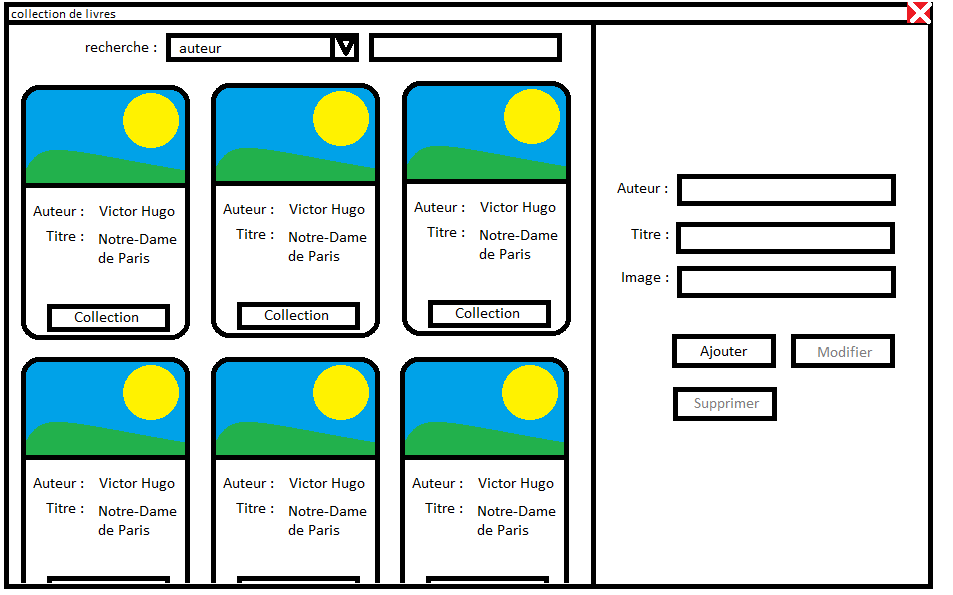
## 5.1 Maquettes de l’application C#

### 5.1.1 Connexion

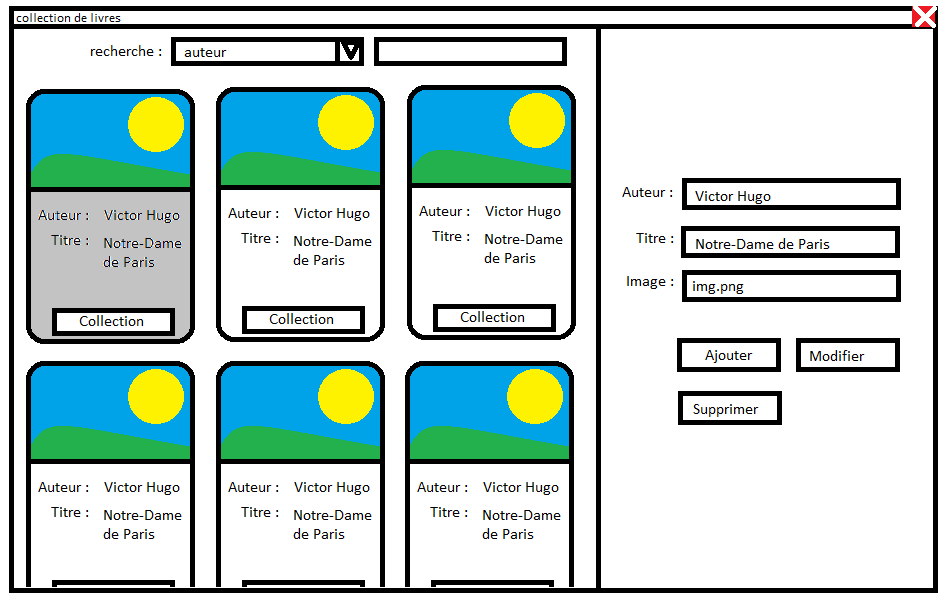


Maquette connexion

### 5.1.2 Collection de livres

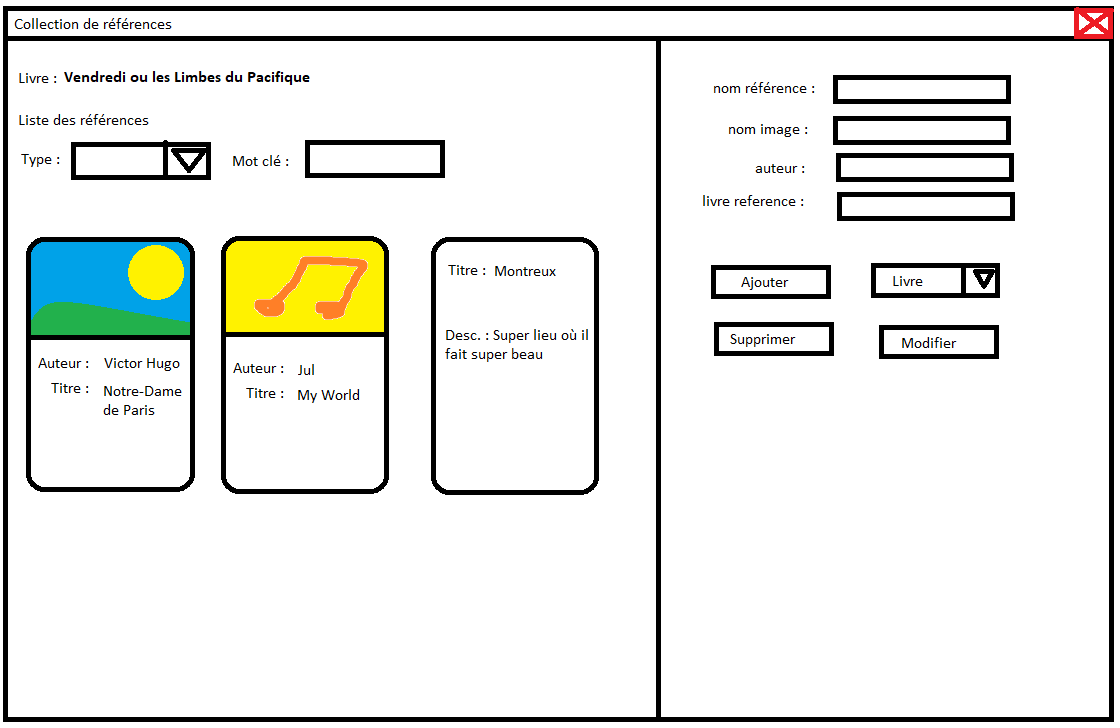


Collection de livres sans sélection



Collection de livres avec sélection

### 5.1.3 Collection de références



Collection de références

# 6.0 Architecture du projet

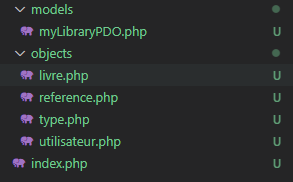
## Architecture générale

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.

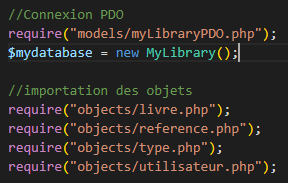
## API PHP

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.



## 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.

## 

*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 |
| Methodes |  |
| 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 | Désérialise le Json en le passant en dynamic |
| LivresParUtilisateur | Classe polymorphe   * 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 |
| \_lvres | 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 SelecionnerCard 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 |
| Methodes |  |
| CardReferenceLieu | Constructeur polymorphe   * Si on appelle le constructeur avec un objet livre, la card est crée 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 SelecionnerCard |

### 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 SelecionnerCard |

### 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 défault |
| Propriétés |  |
| Data | permet d’accéder à \_data |
| Extension | permet d’accéder à \_extension |
| Nom | permet d’accéder à \_nom |
| Méthodes |  |
| ImageInFiles | Constructeur polymorphe   * 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. |
| Methodes |  |
| 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 |
| --- | --- |
| Methodes |  |
| 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 |

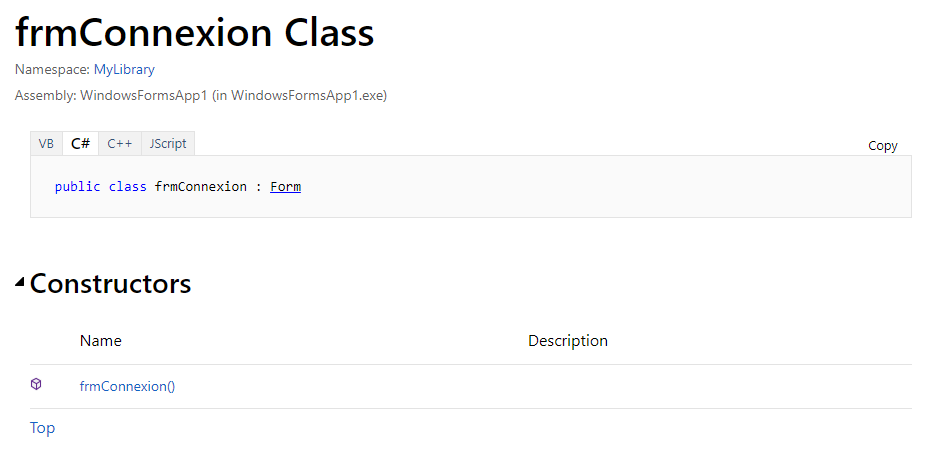
### Class Utilisateur

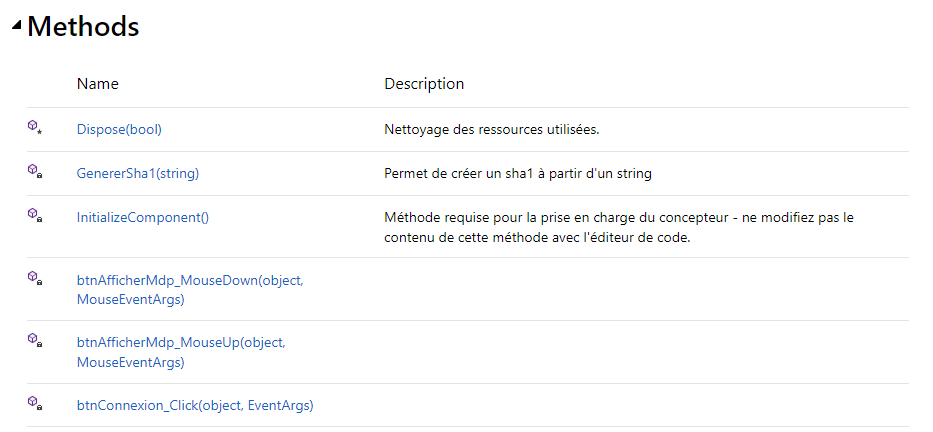
Permet d’encapsuler la table Utilisateur dans la base de données

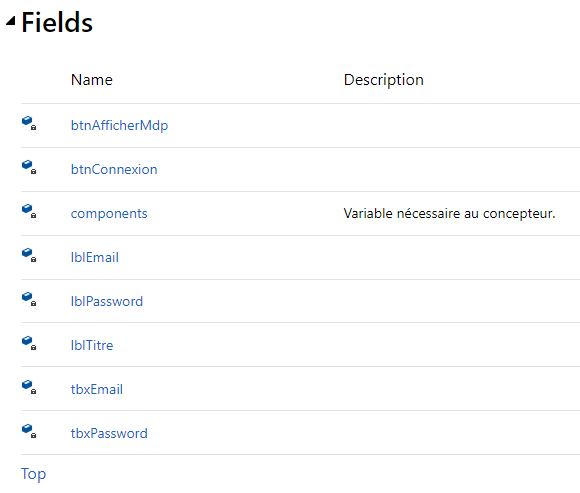
| Nom |  |
| --- | --- |
| Champs |  |
| \_clientRest | Instance de la class ClientRest |
| \_email | Permet de stocker l’email de l’utilisateur   * varchar(255) |
| \_idUtilisateur | Permet de stocker l’id de l’utilisateur   * int(11) |
| \_password | Permet de stocker le mot de passe de l’utilisateur   * varchar(255) |
| DEFAULT\_IDUTILISATEUR | Permet de donner un id d’utilisateur par défault |
| 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.





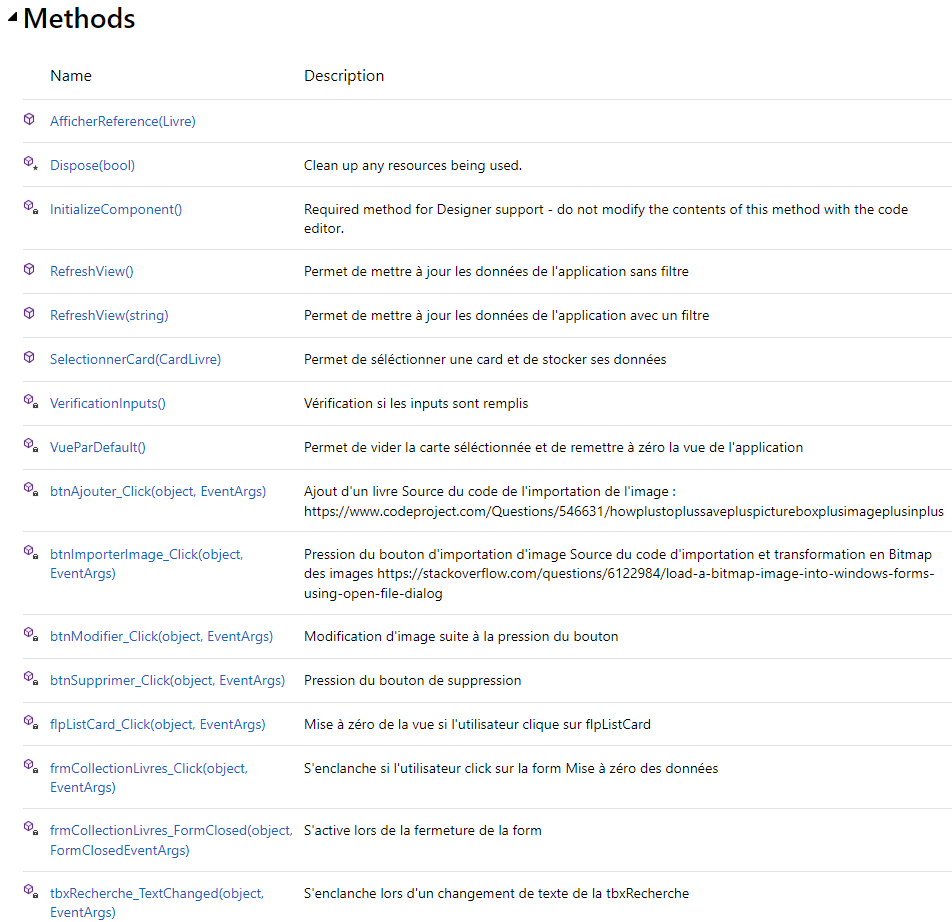


### FrmCollectionLivres

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.



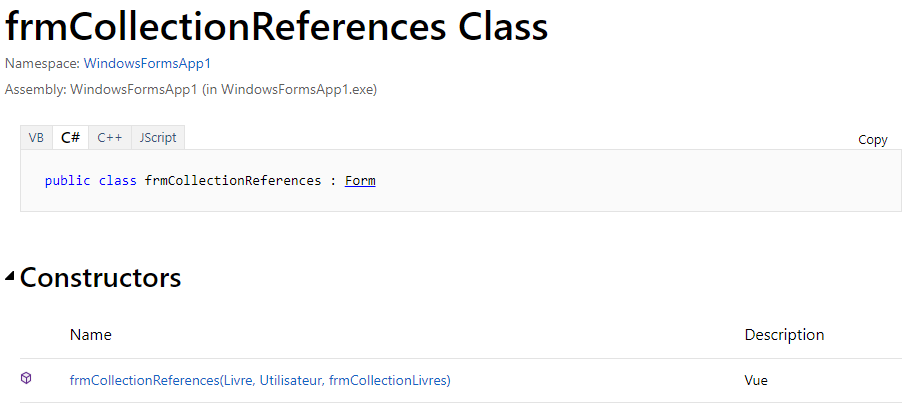


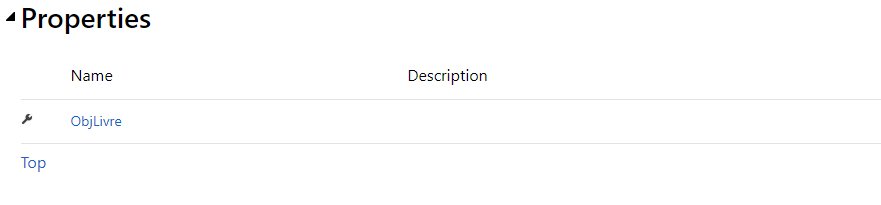


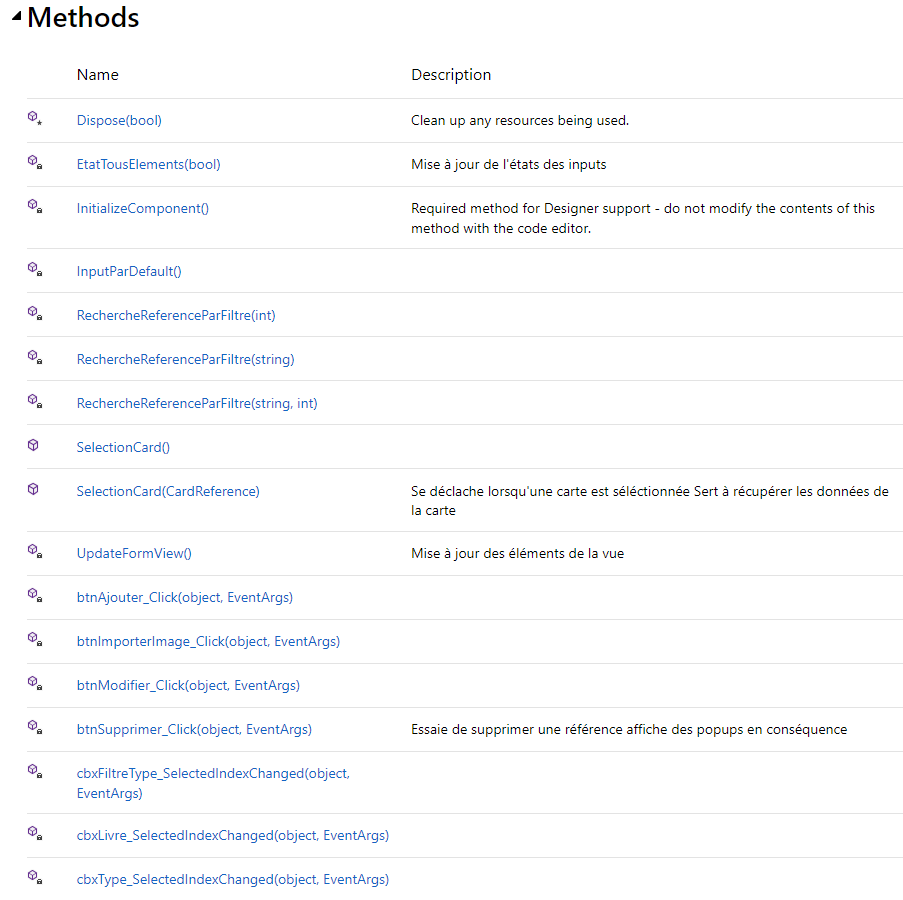
### 

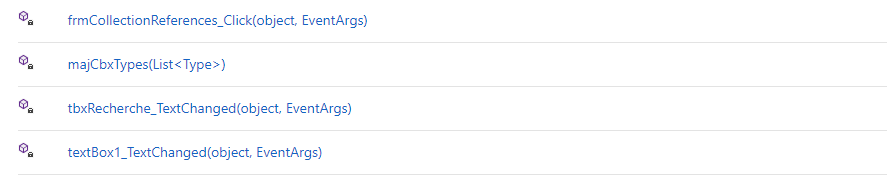
### FrmCollectionReferences

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.









## 

## 

# 7.0 Tests

## 7.1 Scénario de tests

### 2.6.1 API

| **Titre** | Connexion à l’API |
| --- | --- |
| **Résultat attendu** | 1. Requête de type GET 2. En cas de succès, la données de l’utilisateur *connecte* 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 *connecte* 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** | 1. Pouvoir afficher les livres 2. Pouvoir ajouter un livre 3. Pouvoir modifier un livre 4. Pouvoir supprimer un livre |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | Récupération dynamique de livres |
| --- | --- |
| **Résultat attendu** | Pouvoir récupérer un livre par   1. Son auteur 2. Sa description 3. Par l’utilisateur qu’il l’a ajouté |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | CRUD sur la table References |
| --- | --- |
| **Résultat attendu** | 1. Pouvoir afficher les références 2. Pouvoir ajouter un référence 3. Pouvoir modifier un référence 4. Pouvoir supprimer un référence |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | Récupération dynamique de références |
| --- | --- |
| **Résultat attendu** | Pouvoir récupérer une référence par   1. Son type 2. Du Livre 3. Son texte |
| **Résultat obtenu** | Fonctionnel |

### 2.6.1 Application C#

| **Titre** | Connexion |
| --- | --- |
| **Résultat attendu** | 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 |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | Récupération des données lors de la connexion |
| --- | --- |
| **Résultat attendu** | Lors d’une connexion réussie, l’utilisateur est stocké dans une instance objet *Utilisateur* avec ses informations   1. idUtilisateur 2. Email 3. Password (sha1) |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | Création dynamique des *Card* de livres |
| --- | --- |
| **Résultat attendu** | 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 *Card* (Objet Windows forms hérité de Panel) créés dynamiquement. 3. Chaque *Card* est ensuite affichés dans une List |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | Recherche de *Card Livres* |
| --- | --- |
| **Résultat attendu** | Permet de faire une recherche dans la liste des *Card* qui contiennent un :   * Auteur donné * Titre donné |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | CRUD sur les Livres via les *Card* |
| --- | --- |
| **Résultat attendu** | Grâce un formulaire, l’application doit   1. Pouvoir ajouter un livre 2. Pouvoir modifier un livre 3. Pouvoir supprimer un livre |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | Ouverture d’une form collection de références à partir d’une *Card* Livre |
| --- | --- |
| **Résultat attendu** | Suite à l’appuie d’un bouton *Référence* d’une *Card* Livre 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 *Card* de différentes types   1. Musique 2. Livre 3. Lieu |
| **Résultat obtenu** | Fonctionnel |

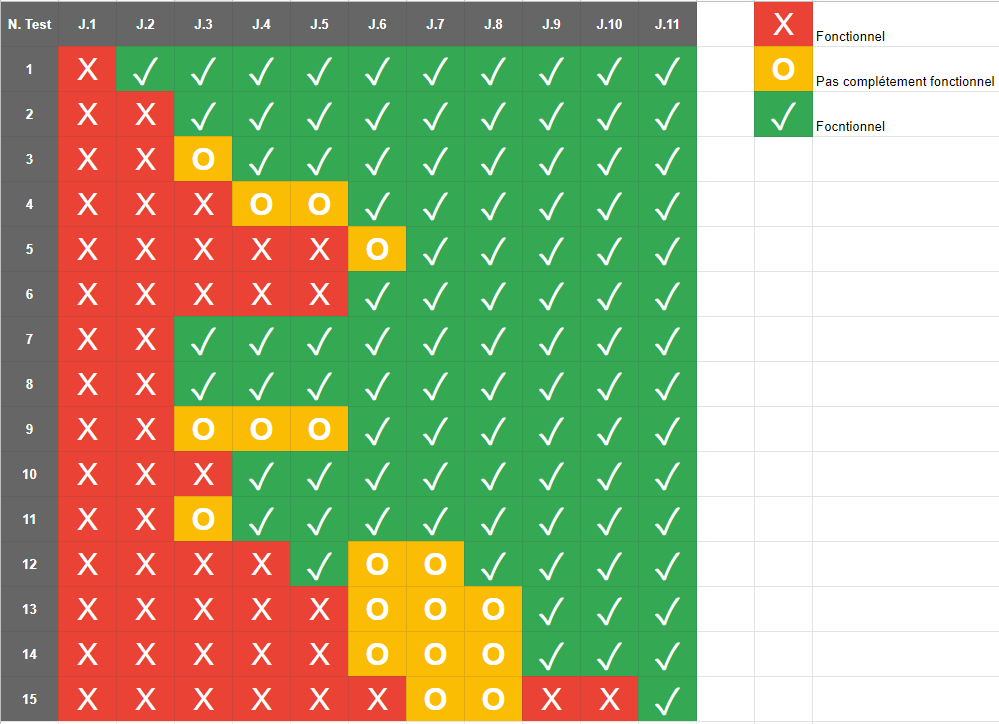
| **Titre** | Filtrage des *Card* référence |
| --- | --- |
| **Résultat attendu** | Affichage des card demandés par   1. Type 2. Texte |
| **Résultat obtenu** | Fonctionnel |

| **Titre** | CRUD sur les références via les *Card* |
| --- | --- |
| **Résultat attendu** | Grâce un formulaire, l’application doit   1. Pouvoir ajouter une références 2. Pouvoir modifier une références 3. Pouvoir supprimer une références 4. Message de confirmation lors de la suppression |
| **Résultat obtenu** | Fonctionnel |

## 

## 

## 7.2 Evolution des tests



# 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 *connecte* 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