```
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   * Decs.    : 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   * Decs.    : 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",   ↵
              "Victor Hugo", "frmConnexion.png", 0),  null)
38          {
39
40          }
41          public CardLivre(Livre livre, frmCollectionLivres lvres) : base ↵
              ()
42          {
43              _livre = livre;
44              _lvres = lvres;
45
46              _image = new PictureBox();
47              _image.Size = new Size(150, 70);
48              _image.BackColor = Color.LightBlue;
49              try
50              {
51                  _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 methodes
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
```

```cs
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : CardReference.cs Card
 * Decs.    : Sert de modèles pour les card des références
 */

using MyLibrary.classes;
using WindowsFormsApp1;

namespace MyLibrary
{
    public abstract class CardReference : Card
    {
        #region varaibles d'instance
        Reference _reference;
        frmCollectionReferences _frm;
        #endregion

        #region propriétés
        public Reference ObjReference
        {
            get { return _reference; }
            set { _reference = value; }
        }

        protected frmCollectionReferences Frm
        {
            get { return _frm; }
            set { _frm = value; }
        }
        #endregion

        #region constructeurs
        public CardReference(Reference reference,
          frmCollectionReferences frm) : base()
        {
            _reference = reference;
            _frm = frm;
            Click += ClickCard;
        }
        #endregion
    }
}

```

```
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : CardReferenceLieu.cs cardReference
 * Decs.    : Permet de créer une card de référence de type lieu
 */

using MyLibrary.classes;
using System;
using System.Drawing;
using System.Windows.Forms;
using WindowsFormsApp1;

namespace MyLibrary
{
    public class CardReferenceLieu : CardReference
    {
        /// <summary>
        /// Permet de créer card de référence lieu
        /// </summary>
        /// <param name="reference"></param>
        /// <param name="frm"></param>
        public CardReferenceLieu(Reference reference,
          frmCollectionReferences frm) : base(reference, frm)
        {

            Label lbltxtTitre = new Label();
            lbltxtTitre.Text = "Titre : ";
            lbltxtTitre.Location = new Point(Location.X + 2, Location.Y
              + 20);
            Label lblTitre = new Label();
            lblTitre.Text = ObjReference.NomReference;
            lblTitre.Location = new Point(Location.X + 2, Location.Y +
              35);

            Label lblTxtDescription = new Label();
            lblTxtDescription.Text = "Description : ";
            lblTxtDescription.Location = new Point(190, 5);
            lblTxtDescription.Location = new Point(Location.X + 2,
              Location.Y + 80);
            Label lblDescription = new Label();
            lblDescription.Text = ObjReference.DescriptionLieu;
            lblDescription.Location = new Point(Location.X + 2,
              Location.Y + 95);
            lblDescription.Size = new Size(Size.Width - 2, Size.Height -
               80);

            //Ajout des événements de click
            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
                les données d'un livre
44          /// </summary>
45          /// <param name="livre">données du livre</param>
46          /// <param name="reference">données de référence du livre</
                param>
47          /// <param name="frm">form où la card est affichée</param>
48          public CardReferenceLivre(Livre livre, Reference reference,
                frmCollectionReferences frm) : base(reference, frm)
49          {
50
```

```
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,
                      Location.Y + 80);
73                  Label lblAuteur = new Label();
74                  lblAuteur.Text = _livre.Auteur;
75                  lblAuteur.Location = new Point(Location.X + 2,
                      Location.Y + 95);
76
77                  Label lbltxtTitre = new Label();
78                  lbltxtTitre.Text = "Titre : ";
79                  lbltxtTitre.Location = new Point(190, 5);
80                  lbltxtTitre.Location = new Point(Location.X + 2,
                      Location.Y + 120);
81                  Label lblTitre = new Label();
82                  lblTitre.Text = _livre.Titre;
83              lblTitre.Location = new Point(Location.X + 2, Location.Y +
                  135);
84
85              _image.Click += ClickCard;
86              lbltxtAuteur.Click += ClickCard;
87              lblAuteur.Click += ClickCard;
88              lbltxtTitre.Click += ClickCard;
89              lblTitre.Click += ClickCard;
90
91
92
93                  Controls.Add(_image);
94                  Controls.Add(lblAuteur);
95                  Controls.Add(lbltxtAuteur);
96                  Controls.Add(lblTitre);
97                  Controls.Add(lbltxtTitre);
98
99          }
```

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

```
 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,
             frmCollectionReferences frm) : base(reference, frm)
34         {
35             _image = new PictureBox();
36             _image.Size = new Size(150, 70);
37             _image.BackColor = Color.Orange;
38             try
39             {
40                 _image.Image = Image.FromFile(reference.NomImage);
41             }
42             catch (Exception ex)
43             {
44                 Console.WriteLine(ex.Message);
45             }
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,
                  Location.Y + 80);
54              Label lblAuteur = new Label();
55              lblAuteur.Text = ObjReference.Auteur;
56              lblAuteur.Location = new Point(Location.X + 2, Location.Y +
                   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
                   + 120);
62              Label lblTitre = new Label();
63              lblTitre.Text = ObjReference.NomReference;
64              lblTitre.Location = new Point(Location.X + 2, Location.Y +
                  135);
65
66              //Ajout des événements de click pour séléctionner 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
              frmSelectionReferences avec les données de l'objet
99          /// </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   * Decs.    : 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/
                  API_MyLibrary/";
31          }
32          #endregion
33
34          #region propriétés
35          //https://www.c-sharpcorner.com/UploadFile/8911c4/singleton-
              design-pattern-in-C-Sharp/
36          public static ClientRest Instance
37          {
38              get
39              {
40                  if (instance == null)
41                  {
42                      instance = new ClientRest();
43                  }
44                  return instance;
45              }
46          }
47          #endregion
48
49
50          #region Methodes
51          /// <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 =
                        MetadataPropertyHandling.Ignore;
75                      sr.Close();
76                  }
77              }
78              catch (Exception ex)
79              {
80                  return ex.ToString();
81              }
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                 requestObject.Method = method.ToUpper();
101                 HttpWebResponse responseObject = null;
102                 responseObject = (HttpWebResponse)
                      requestObject.GetResponse();
103             }
104             catch (Exception ex)
105             {
106                 Console.WriteLine(ex.ToString());
107                 return false;
108             }
109             return true;
110         }
111
112
113         /// <summary>
114         /// Permet de désérialiser le JSON
115         /// https://www.youtube.com/watch?v=CjoAYslTKX0
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>
                      (strJson);
123
124                 return jPerson;
125             }
126             catch (Exception ex)
127             {
128                 Console.WriteLine("error : " + ex);
129                 return null;
130             }
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
            utilisateur)
139         {
140             List<Livre> livres = new List<Livre>();
141             try
142             {
143                 dynamic livresDynamic = DeserialiseJSON(ApiRequest("?
                      email=" + utilisateur.Email + "&password=" +
                      utilisateur.Password + "&table=livres", "GET"));
144                 foreach (var element in livresDynamic)
145                 {
146                     livres.Add(new Livre(Convert.ToInt32(element
```

```
                    ["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
              recherche personnalisée
159         /// </summary>
160         /// <param name="utilisateur">utilisateur</param>
161         /// <param name="recherche">chaine personnalisée</param>
162         /// <returns></returns>
163         public List<Livre> LivresParUtilisateur(Utilisateur
              utilisateur, string recherche)
164         {
165             List<Livre> livres = new List<Livre>();
166             dynamic livresDynamic = DeserialiseJSON(ApiRequest("?
                email="+utilisateur.Email
                +"&password="+utilisateur.Password
                +"&table=livres&recherche="+recherche+"", "GET"));
167             foreach (var element in livresDynamic)
168             {
169                 livres.Add(new Livre(Convert.ToInt32(element
                    ["idLivre"]), Convert.ToString(element["titre"]),
                    Convert.ToString(element["auteur"]), Convert.ToString
                    (element["nomImage"]), Convert.ToInt32(element
                    ["idUtilisateur"])));
170             }
171             return livres;
172         }
173
174         /// <summary>
175         /// Récupère tous les types
176         /// </summary>
177         /// <param name="utilisateur"></param>
178         /// <returns></returns>
179         public List<Type> TousTypes(Utilisateur utilisateur)
180         {
181             List<Type> types = new List<Type>();
182             dynamic typesDynamic = DeserialiseJSON(ApiRequest("?email="
                + utilisateur.Email + "&password=" + utilisateur.Password
                + "&table=types", "GET"));
183             foreach(var element in typesDynamic)
184             {
```

```
185                 types.Add(new Type(Convert.ToInt32(element["idType"]),
                      Convert.ToString(element["nomType"])));
186             }
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
              utilisateur, Livre livre)
198         {
199             List<Reference> referencesParLivre = new List<Reference>();
200             dynamic referencesDynamic = DeserialiseJSON(ApiRequest("?
                email=" + utilisateur.Email + "&password=" +
                utilisateur.Password + "&table=references&idLivre="
                +livre.IdLivre.ToString() +"", "GET"));
201             foreach (var element in referencesDynamic)
202             {
203                 switch (Convert.ToInt32(element["idType"]))
204                 {
205                     case 1:
206                         referencesParLivre.Add(new ReferenceLivre
                      (Convert.ToInt32(element["idReference"]),
                      Convert.ToString(element["nomImage"]),
                      Convert.ToInt32(element["livreReference"]),
                      Convert.ToInt32(element["idLivre"])));
207                         break;
208                     case 2:
209                         referencesParLivre.Add(new ReferenceMusique
                      (Convert.ToInt32(element["idReference"]),
                      Convert.ToString(element["nomImage"]),
                      Convert.ToString(element["nomReference"]),
                      Convert.ToString(element["auteur"]), Convert.ToInt32
                      (element["idLivre"])));
210                         break;
211                     case 3:
212                         if(element["descriptionLieu"] == null)
213                         {
214                             referencesParLivre.Add(new ReferenceLieu
                      (Convert.ToInt32(element["idReference"]),
                      Convert.ToString(element["nomReference"]),
                      Convert.ToInt32(element["idLivre"])));
215                         }
216                         else
217                         {
218                             referencesParLivre.Add(new ReferenceLieu
                      (Convert.ToInt32(element["idReference"]),
```

```
                    Convert.ToString(element["nomReference"]),
                    Convert.ToString(element["descriptionLieu"]),
                    Convert.ToInt32(element["idLivre"])));
                        }

                        break;
                    }
                }

                return referencesParLivre;
            }

            /// <summary>
            /// Recupère un livre par son id
            /// </summary>
            /// <param name="utilisateur">utilisateur propriétaire du
              livre</param>
            /// <param name="idLivre">id du livre à rechercher</param>
            /// <returns></returns>
            public Livre LivreParIdLivre(Utilisateur utilisateur, int
              idLivre)
            {
                dynamic LivreDynamic = DeserialiseJSON(ApiRequest("?email="
                   + utilisateur.Email + "&password=" + utilisateur.Password
                   + "&table=livres&idLivre="+idLivre.ToString()+"",
                  "GET"));
                return new Livre(Convert.ToInt32(LivreDynamic["idLivre"]),
                  Convert.ToString(LivreDynamic["titre"]), Convert.ToString
                  (LivreDynamic["auteur"]), Convert.ToString(LivreDynamic
                  ["nomImage"]), Convert.ToInt32(LivreDynamic
                  ["idUtilisateur"]));

            }

            /// <summary>
            /// Récupère le dernier livre de l'utlisateur
            /// </summary>
            /// <param name="utilisateur"></param>
            /// <returns></returns>
            public int DernierLivreUtilisateur(Utilisateur utilisateur)
            {
                return Convert.ToInt32(DeserialiseJSON(ApiRequest("?email="
                   + utilisateur.Email + "&password=" + utilisateur.Password
                   + "&table=livres&tri=dernier", "GET")));
            }

            #endregion
        }
    }

```

```
 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  * Decs.    : 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
             d'objets.
34         /// Permet de faire un CRUD sur les livres via un formulaire
35         /// </summary>
36         /// <param name="utilisateur">Données de l'tutilisateur</param>
37         /// <param name="frmPrecedente">Form de connexion</param>
38         public frmCollectionLivres(Utilisateur utilisateur,
             frmConnexion frmPrecedente)
39         {
40             //Initialisation des composants
41             InitializeComponent();
42             //Stockage des données dans les variables d'instances
43             _utilisateur = utilisateur;
44             _clientRest = ClientRest.Instance;
45             //Instanciations des lists
46             _listLivresUtilisateur = new List<Livre>();
47             _listCard = new List<Card>();
48             //Fermeture de la form de connexion
49             frmPrecedente.Close();
50             //Affichage des données
51             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/howplustoplussaveplusp
              icturebox
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 methode 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                          //Afficahge 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                      //Afficahge du message d'erreur
84                      MessageBox.Show("Une erreur s'est produite lors de
                      l'ajout du livre", "Erreur interne",
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
85                      Console.WriteLine(ex.Message);
86                  }
87              }
88              else
89              {
90                  //Afficahge 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 des images
98          /// https://stackoverflow.com/questions/6122984/load-a-bitmap-image-into-windows-forms-using-open-file-dialog
99          /// </summary>
100         /// <param name="sender"></param>
101         /// <param name="e"></param>
102         private void btnImporterImage_Click(object sender, EventArgs e)
103         {
104             //Ouverture du dialogue d'importation
105             using (OpenFileDialog dlg = new OpenFileDialog())
106             {
107                 dlg.Title = "Open Image";
108                 dlg.Filter = "Image Files (*.bmp;*.jpg;*.jpeg,*.png)|*.BMP;*.JPG;*.JPEG;*.PNG";
109 
110                 if (dlg.ShowDialog() == DialogResult.OK)
111                 {
112                     // Create a new Bitmap object from the picture file on disk,
113                     // and assign that to the PictureBox.Image property
114                     pbxImageAjouter.Image = new Bitmap(dlg.FileName);
115                 }
116             }
117         }
118 
119         /// <summary>
120         /// S'active lors de la fermeture de la form
121         /// </summary>
122         /// <param name="sender"></param>
123         /// <param name="e"></param>
124         private void frmCollectionLivres_FormClosed(object sender, FormClosedEventArgs e)
125         {
126             //déconnecte l'utilisateur
127             _utilisateur.Deconnexion();
128         }
129 
130         /// <summary>
131         /// Permet de mettre à jour les données de l'application sans filtre
132         /// </summary>
133         public  void RefreshView()
134         {
135             //remise à zéro des listes
136             flpListCard.Controls.Clear();
137             _listLivresUtilisateur.Clear();
138             _listLivresUtilisateur = _clientRest.LivresParUtilisateur(_utilisateur);
139             _listCard.Clear();
140 
```

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

```
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 séléctionnée
193             _cardSelectionne = null;
194             //Activation du bouton d'ajout et désactivation des boutons
                   de suppression et modifications
195             btnAjouter.Enabled = true;
196             btnModifier.Enabled = false;
197             btnSupprimer.Enabled = false;
198             //Mise à zéro des inputs
199             tbxAuteur.Text = null;
200             tbxTitre.Text = null;
201             pbxImageAjouter.Image = null;
202         }
203
204         /// <summary>
205         /// Permet de séléctionner une card et de stocker ses données
206         /// </summary>
207         /// <param name="card"></param>
208         public void SelectionnerCard(CardLivre card)
209         {
210             //stockage de la card dans une variable d'instance
211             _cardSelectionne = card;
212             //Activation des bouton de modification et de suppression
                  et désactivation du bouton d'ajout
213             btnModifier.Enabled = true;
214             btnSupprimer.Enabled = true;
215             btnAjouter.Enabled = false;
216             //Mise à zéro du style des card
217             foreach(CardLivre uneCard in _listCard)
218             {
219                 uneCard.BackColor = Color.White;
220                 uneCard.ForeColor = Color.Black;
221             }
222             //Mise en évidence de la card séléctionnée
223             card.BackColor = Color.Gray;
224             card.ForeColor = Color.White;
225             //Affichage des données dans les inputs
226             tbxAuteur.Text = card.ObjLivre.Auteur;
227             tbxTitre.Text = card.ObjLivre.Titre;
228             //Recherche de l'image par son nom
229             pbxImageAjouter.Image = Image.FromFile
                  (card.ObjLivre.NomImage);
230         }
231
232         /// <summary>
233         /// Modification d'image suite à la pression du bouton
234         /// </summary>
235         /// <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
                    ((Bitmap)pbxImageAjouter.Image);
245                     //Essai de modification du livre
246                     if (_cardSelectionne.ObjLivre.PutLivre
                    (_utilisateur, new Livre(0, tbxTitre.Text,
                    tbxAuteur.Text, imageEnregistrer.Nom +
                    imageEnregistrer.Extension, 0)))
247                     {
248                         //Suppression de l'image précédente
249                         File.Delete
                    (_cardSelectionne.ObjLivre.NomImage);
250                         //Sauvegarde de la nouvelle image
251                         imageEnregistrer.SaveBmp();
252                         //Afficahge du message en cas de succès
253                         MessageBox.Show("Le livre a été modifié",
                    "Livre modifié", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
254                         //Mise à jour de la vue
255                         RefreshView();
256                     }
257                     else
258                     {
259                         //Afficahge du message d'erreur
260                         MessageBox.Show("Une erreur s'est produite lors
                     de la modification du livre", "Erreur interne",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
261                     }
262                 }
263                 catch (Exception ex)
264                 {
265                     //Afficahge du message d'erreur
266                     MessageBox.Show("Une erreur s'est produite lors de
                    la modification du livre", "Erreur interne",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
267                     Console.WriteLine(ex.Message);
268                 }
269             }
270             else
271             {
272                 //Afficahge du message d'erreur
273                 MessageBox.Show("Veuillez ajouter remplir tous les
                  champs", "Inputs non valides", MessageBoxButtons.OK,
                  MessageBoxIcon.Error);
274             }
275         }
```

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

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

```
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : frmCollectionReferences.cs Form
 * Decs.    : Vue de la collection des références d'un livre
 */

using MyLibrary;
using MyLibrary.classes;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class frmCollectionReferences : Form
    {
        //Variables d'instances
        private Livre _livre;
        private Utilisateur _utilisateur;
        private ClientRest _clientRest;
        private List<Reference> _references;
        private List<Card> _cardsReferences;
        private List<Livre> _livres;
        private CardReference _cardSelectionne;
        private frmCollectionLivres _frmCollectionLivres;

        //Référence ambigu
        private List<MyLibrary.classes.Type> _types;

        //Propriétés
        public Livre ObjLivre
        {
            get { return _livre; }
            set { _livre = value; }
        }
        /// <summary>
        /// Vue
        /// </summary>
        /// <param name="livre"></param>
        /// <param name="utilisateur"></param>
        public frmCollectionReferences(Livre livre, Utilisateur
          utilisateur, frmCollectionLivres frmCollectionLivres)
        {
            _livre = livre;
            _utilisateur = utilisateur;
            _references = new List<Reference>();
            _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,
                    tbxTitre.Text, tbxAuteur.Text, imageEnregistrer.Nom +
                     imageEnregistrer.Extension,
                    _utilisateur.IdUtilisateur);
89                          if (nouveauLivre.PostLivre(_utilisateur))
90                          {
91
92                              imageEnregistrer.SaveBmp();
93                              new ReferenceLivre(0,
                    nouveauLivre.NomImage,
                    _clientRest.DernierLivreUtilisateur(_utilisateur),
                    _livre.IdLivre).PostReference(_utilisateur);
94                              _frmCollectionLivres.RefreshView();
95                              UpdateFormView();
96                          }
97                      }
98                      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,
                  _livre);
153
154             cbxLivre.Items.Add("-- NOUVEAU LIVRE --");
155             foreach (Livre lvr in _livres)
156             {
157                 cbxLivre.Items.Add(lvr.Titre);
158             }
159             cbxLivre.SelectedIndex = 0;
160
161
162             if(cbxFiltreType.SelectedIndex == 0 && tbxRecherche.Text ==
                  "")
163             {
164                 foreach (Reference reference in _references)
165                 {
166                     Card nouvelleCard;
167                     switch (reference.IdType)
168                     {
169                         //Livre
170                         case 1:
171                             nouvelleCard = new CardReferenceLivre
                        (_utilisateur, reference, this);
172                             flpReferences.Controls.Add(nouvelleCard);
173                             _cardsReferences.Add(nouvelleCard);
174                             break;
175                         //Musique
176                         case 2:
177                             nouvelleCard = new CardReferenceMusique
                        (reference, this);
178                             flpReferences.Controls.Add(nouvelleCard);
179                             _cardsReferences.Add(nouvelleCard);
```

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

```
226             tbxTitre.Enabled = etat;
227             tbxAuteur.Enabled = etat;
228             tbxDescription.Enabled = etat;
229             cbxLivre.Enabled = etat;
230         }
231
232
233         /// <summary>
234         /// Essaie de supprimer une référence affiche des popups en
              conséquence
235         /// </summary>
236         /// <param name="sender"></param>
237         /// <param name="e"></param>
238         private void btnSupprimer_Click(object sender, EventArgs e)
239         {
240             DialogResult dr = MessageBox.Show("Voulez-vous vraiment
                supprimer cette référence", "Suppression de Référence",
                MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
241
242             if (dr == DialogResult.Yes)
243             {
244                 //Essai de suppression de la référence
245                 if (_cardSelectionne.ObjReference.DeleteReference
                    (_utilisateur))
246                 {
247                     MessageBox.Show("La référence à été supprimée",
                    "Référence Supprimée", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
248                     UpdateFormView();
249                 }
250                 else
251                 {
252                     MessageBox.Show("Un problème s'est produit lors de
                    l'envoie de la donnée", "Erreur",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
253                 }
254             }
255
256
257
258         }
259
260         /// <summary>
261         /// Mets à jour les types
262         /// </summary>
263         /// <param name="_types">List des types</param>
264         private void majCbxTypes(List<MyLibrary.classes.Type> _types)
265         {
266             cbxType.Items.Clear();
267             foreach(var type in _types)
268             {
269                 cbxType.Items.Add(type.NomType);
270             }
```

```
271             //Séléction par défault
272             cbxType.SelectedIndex = 0;
273         }
274 
275         private void cbxType_SelectedIndexChanged(object sender,
              EventArgs e)
276         {
277             //Désactivation de tous les inputs
278             EtatTousElements(false);
279 
280             //Activation des inputs en fonction du type de référence
281             switch (cbxType.SelectedIndex)
282             {
283                 //Livre
284                 case 0:
285                     tbxTitre.Enabled = true;
286                     btnImporterImage.Enabled = true;
287                     tbxAuteur.Enabled = true;
288                     cbxLivre.Enabled = true;
289                     break;
290                 //Musique
291                 case 1:
292                     tbxTitre.Enabled = true;
293                     btnImporterImage.Enabled = true;
294                     tbxAuteur.Enabled = true;
295                     break;
296                 //Lieu
297                 case 2:
298                     tbxTitre.Enabled= true;
299                     tbxDescription.Enabled = true;
300                     break;
301             }
302         }
303 
304         /// <summary>
305         /// Se déclache lorsqu'une carte est séléctionnée
306         /// Sert à récupérer les données de la carte
307         /// </summary>
308         /// <param name="card">Card séléctionnée</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
                la card séléctionnée
317             foreach (var uneCard in _cardsReferences)
318             {
319                 if (uneCard == card)
320                 {
321                     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 =                                    ↵
                      card.ObjReference.DescriptionLieu;
340                     pbxImage.Image = Image.FromFile                          ↵
                      (card.ObjReference.NomImage);
341                     cbxType.SelectedIndex = 0;
342                     break;
343                 //Musique
344                 case 2:
345                     tbxTitre.Text = card.ObjReference.NomReference;
346                     tbxAuteur.Text = card.ObjReference.Auteur;
347                     tbxDescription.Text =                                    ↵
                      card.ObjReference.DescriptionLieu;
348                     pbxImage.Image = Image.FromFile                          ↵
                      (card.ObjReference.NomImage);
349                     cbxType.SelectedIndex = 1;
350                     break;
351                 //Lieu
352                 case 3:
353                     tbxTitre.Text = card.ObjReference.NomReference;
354                     tbxAuteur.Text = card.ObjReference.Auteur;
355                     tbxDescription.Text =                                    ↵
                      card.ObjReference.DescriptionLieu;
356                     cbxType.SelectedIndex = 2;
357                     break;
358             }
359         }
360 
361         /// <summary>
362         /// S'active quand une card est séléctionnée
363         /// </summary>
364         public void SelectionCard()
365         {
366             foreach (var uneCard in _cardsReferences)
367             {
368                uneCard.BackColor = Color.White;
369                uneCard.ForeColor = Color.Black;
```

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

```
                    "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",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
409                 }
410                 break;
411             }
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,
              EventArgs e)
442         {
443             InputParDefault();
444         }
445
446         private void cbxLivre_SelectedIndexChanged(object sender,
              EventArgs e)
447         {
448             if (cbxLivre.SelectedIndex != 0)
449             {
```

```
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-
                  bitmap-image-into-windows-forms-using-open-file-dialog
465             using (OpenFileDialog dlg = new OpenFileDialog())
466             {
467                 dlg.Title = "Open Image";
468                 dlg.Filter = "Image Files (*.bmp;*.jpg;*.jpeg,*.png)|
                      *.BMP;*.JPG;*.JPEG;*.PNG";
469 
470                 if (dlg.ShowDialog() == DialogResult.OK)
471                 {
472                     // Create a new Bitmap object from the picture file
                        on disk,
473                     // and assign that to the PictureBox.Image property
474                     pbxImage.Image = new Bitmap(dlg.FileName);
475                 }
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
                    (recherche.ToLower()) || reference.Auteur.ToLower
                    ().Contains(recherche.ToLower()) ||
                    reference.NomReference.ToLower().Contains
                    (recherche.ToLower()))
501                 {
502                     switch (reference.IdType)
503                     {
504                         //Livre
505                         case 1:
506                                 nouvelleCard = new CardReferenceLivre
                    (_utilisateur, reference, this);
507                                 listCard.Add(nouvelleCard);
508                             break;
509                         //Musique
510                         case 2:
511                                 nouvelleCard = new CardReferenceMusique
                    (reference, this);
512                                 listCard.Add(nouvelleCard);
513 
514                             break;
515                         //Lieu
516                         case 3:
517                                 nouvelleCard = new CardReferenceLieu
                    (reference, this);
518                                 listCard.Add(nouvelleCard);
519                             break;
520                     }
521                 }
522             }
523             return listCard;
524         }
525 
526         private List<CardReference> RechercheReferenceParFiltre(int
            idType)
527         {
528             List<CardReference> listCard = new List<CardReference>();
529             foreach (Reference reference in _references)
530             {
531                 CardReference nouvelleCard;
532                 switch (reference.IdType)
533                 {
534                     //Livre
535                     case 1:
536                         if (idType == 1)
537                         {
538                             nouvelleCard = new CardReferenceLivre
                    (_utilisateur, reference, this);
539                             listCard.Add(nouvelleCard);
540                         }
```

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

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

```
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : frmConnexion.cs Form
 * Decs.    : Vue de la connexion
 *
 * IMPORTANT : SI VOUS RENCONTREZ DU PROBLEME DU LOGIN, CHANGEZ LA VELEUR DANS LA BASE MyLibrary.Utilisateur.Connecte DE 1 à 0
 *             Si vous avez des problème à générer les solution, veuillez lancer l'application une fois afin que les design s'affichent
 */

using MyLibrary.classes;
using System;
using System.Security.Cryptography;
using System.Text;
using System.Windows.Forms;

namespace MyLibrary
{
    public partial class frmConnexion : Form
    {
        public frmConnexion()
        {
            //initialisation des composants
            InitializeComponent();
        }

        private void btnConnexion_Click(object sender, EventArgs e)
        {
            //Vérification si les champs sont remplis
            if(tbxEmail.Text != "" && tbxPassword.Text != "")
            {
                try
                {
                    //Création d'un nouvel utilisateur avec les données des champs
                    var user = new Utilisateur(tbxEmail.Text, GenererSha1(tbxPassword.Text).ToLower());
                    //Tentative de connexion à l'API
                    if (user.TestConnexion())
                    {
                        //Affichage de la nouvelle form
                        frmCollectionLivres collectionLivres = new frmCollectionLivres(user, this);
                        collectionLivres.Show();
                    }
                    //Si les données de connexion ne correspondent pas
```

```
48                      else
49                      {
50                          //Message d'erreur
51                          MessageBox.Show("Erreur lors de la connexion",
                        "Attention Requise", MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
52                      }
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,
                        MessageBoxIcon.Error);
58                  }
59
60              }
61              else
62              {
63                  //Message d'erreur
64                  MessageBox.Show("Veuillez remplir tous les champs",
                        "Attention Requise", MessageBoxButtons.OK,
                        MessageBoxIcon.Warning);
65              }
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          //Afficahge 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
              passe
94          private void btnAfficherMdp_MouseUp(object sender,
              MouseEventArgs e)
95          {
96              //Cache le mot de passe
97              tbxPassword.PasswordChar = '*';
98          }
99      }
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  * Decs.    : 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 propriétes
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
              (nomAleatoire(), extension, data) { }
71
72          /// <summary>
73          /// Image avec nom aléatoire et sans type (type par défault
              png)
74          /// </summary>
75          /// <param name="data">Bitmap de l'image</param>
76          public ImageInFile(Bitmap data) : this(DEFAULT_EXTENSION, data)
               { }
77          #endregion
78
79
80
81
82          #region methodes
83          /// <summary>
84          /// Image avec nom aléatoire et sans type (type par défault
              png)
85          /// Source du code :
86          /// https://stackoverflow.com/questions/1344221/how-can-i-
              generate-random-alphanumeric-strings
87          /// </summary>
88          private static string nomAleatoire()
89          {
90              Random random = new Random();
91              const string chars =
                  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
92              return new string(Enumerable.Repeat(chars, 8).Select(s => s
                  [random.Next(s.Length)]).ToArray());
93          }
94
95          /// <summary>
96          /// Permet d'enregistrer l'image en local
97          /// </summary>
98          public void SaveBmp()
99          {
```

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

```
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : Livre.cs class
 * Decs.    : Permet d'encapsuler la table Livres de la base de données
 */

namespace MyLibrary.classes
{
    public class Livre
    {
        #region Variables d'instances
        private int _idLivre, _idUtilisateur;
        private string _titre, _auteur, _nomImage;
        private ClientRest _clientRest;
        #endregion

        #region Propriétés
        public int IdLivre
        {
            get { return _idLivre; }
            set { _idLivre = value; }
        }

        public int IdUtilisateur
        {
            get { return _idUtilisateur; }
            set { _idUtilisateur = value; }
        }

        public string Titre
        {
            get { return _titre; }
            set { _titre = value; }
        }

        public string Auteur
        {
            get { return _auteur; }
            set { _auteur = value; }
        }

        public string NomImage
        {
            get { return _nomImage; }
            set { _nomImage = value; }
        }
        #endregion

```

```
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
              (Prop : Email, Password) </param>
79          /// <returns>True = code 201, False = erreur</returns>
80          public bool PostLivre(Utilisateur utilisateur)
81          {
82              return _clientRest.AppelSimple("?
                  table=livres&email="+utilisateur.Email
                  +"&password="+utilisateur.Password+"&titre="+_titre
                  +"&auteur="+_auteur+"&nomImage="+_nomImage+"", "POST");
83          }
84
85          /// <summary>
86          /// Permet à l'utilisateur de modifier le livre de la class
87          /// </summary>
88          /// <param name="utilisateur">utilisateur qui modifie le livre
              (Prop : Email, Password)</param>
89          /// <param name="nouvellesDonnees">Objet livre avec les
              nouvelles informations</param>
90          /// <returns>True = code 201, False = erreur</returns>
91          public bool PutLivre(Utilisateur utilisateur, Livre
              nouvellesDonnees)
92          {
93              //Mise à jour des varaibles d'instances
94              _auteur = nouvellesDonnees.Auteur;
95              _titre = nouvellesDonnees.Titre;
96              _nomImage = nouvellesDonnees.NomImage;
97
98
```

```
 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   * Decs.    : 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,
            _descriptionLieu;
20         #endregion
21
22         #region Propriétés
23         public int IdReference
24         {
25             get { return _idReference; }
26             set { _idReference = value; }
27         }
28
29         public string NomReference
30         {
31             get { return _nomReference; }
32             set { _nomReference = value; }
33         }
34
35         public string NomImage
36         {
37             get { return _nomImage; }
38             set { _nomImage = value; }
39         }
40
41         public string Auteur
42         {
43             get { return _auteur; }
44             set { _auteur = value; }
45         }
46
47         public int IdType
48         {
49             get { return _idType; }
50             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
72          #endregion
73
74          #region Constructeurs
75          /// <summary>
76          /// Permet d'encapsuler la table References de la base de
              données
77          /// Sert également de class mère aux références livre, musique
              et lieu
78          /// </summary>
79          /// <param name="idReference">int(11)</param>
80          /// <param name="nomReference">varchar(255)</param>
81          /// <param name="nomImage">varchar(255)</param>
82          /// <param name="auteur">varchar(100)</param>
83          /// <param name="idType">int(11)</param>
84          /// <param name="livreReference">int(11) livres, idLivres</
              param>
85          /// <param name="idLivre">int(11) livres, IdLivres </param>
86          /// <param name="descriptionLieu">mediumtext</param>
87          public Reference(int idReference, string nomReference, string
              nomImage, string auteur, int idType, int livreReference, int
              idLivre, string descriptionLieu)
88          {
89              _idReference = idReference;
90              _nomReference = nomReference;
91              _nomImage = nomImage;
92              _auteur = auteur;
93              _idType = idType;
94              _livreReference = livreReference;
95              _idLivre = idLivre;
96              _descriptionLieu = descriptionLieu;
97          }
98
```

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

```
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   * Decs.    : 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 description
25          /// </summary>
26          /// <param name="idReference">int(11)</param>
27          /// <param name="titre">varchar(255)</param>
28          /// <param name="descriptionLieu">madiumtext</param>
29          /// <param name="idLivre">int(11)</param>
30          public ReferenceLieu(int idReference, string titre, string descriptionLieu, int idLivre) : base(idReference, titre, "", "", 3, 0, idLivre, descriptionLieu) { }
31
32          /// <summary>
33          /// Permet de créer une référence de type Lieu sans description
34          /// </summary>
35          /// <param name="idReference">int(11)</param>
36          /// <param name="titre">varchar(255)</param>
37          /// <param name="idLivre">int(11)</param>
38          public ReferenceLieu(int idReference, string titre, int idLivre) : this(idReference, titre, DEFAULT_DESCRIPTIONLIEU, idLivre) { }
39
40          /// <summary>
41          /// Permet d'ajouter la référence dans la base par l'API
42          /// </summary>
43          /// <param name="utilisateur">Utilisateur qui ajoute la référence</param>
44          /// <returns>
45          /// true = 201
46          /// false = erreur
47          /// </returns>
```

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

```
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : ReferenceLivre.cs class
 * Decs.    : Permet de créer une référence de type Livre
 */

using MyLibrary.classes;

namespace MyLibrary
{
    public class ReferenceLivre : Reference
    {
        /// <summary>
        /// Permet de créer une référence de type Livre
        /// </summary>
        /// <param name="idReference">int(11)</param>
        /// <param name="livreReference">int(11)</param>
        /// <param name="idLivre">int(11)</param>
        public ReferenceLivre(int idReference, string NomImage, int
          livreReference, int idLivre) : base(idReference, "", NomImage,
           "", 1, livreReference, idLivre, "") { }

        /// <summary>
        /// Permet d'envoyer une référence de type livre dans la table
          référence (clé étangères)
        /// </summary>
        /// <param name="utilisateur">Utilisateur qui envoie la
          référence</param>
        /// <returns>
        /// true = 201
        /// false = erreur
        /// </returns>
        public override bool PostReference(Utilisateur utilisateur)
        {
            return ClientRest.Instance.AppelSimple("?
              table=references&email=" + utilisateur.Email + "&password="
               + utilisateur.Password + "&idLivre=" + IdLivre.ToString()
              + "&idType=" + IdType.ToString() + "&livreReference=" +
              LivreReference.ToString() + "", "POST");
        }

        /// <summary>
        /// Permet de modifier la référence
        /// </summary>
        /// <param name="utilisateur">utilisateur qui modifie la
          référence</param>
        /// <param name="reference">novuelles données</param>
        /// <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 varaibles 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
```

```
/* Projet   : MyLibrary - TPI 2022
 * Version  : 1.0
 * Date     : 18.05.2022
 *
 * Auteur   : Karel V. Svoboda
 * Classe   : I.DA-P4A
 *
 * Class    : ReferenceMusique.cs class
 * Decs.    : Permet de créer une référence de type musique
 */

using MyLibrary.classes;

namespace MyLibrary
{
    public class ReferenceMusique : Reference
    {
        /// <summary>
        /// Permet de créer une référence de type musique
        /// </summary>
        /// <param name="idReference">int(11)</param>
        /// <param name="nomImage">varchar(255)</param>
        /// <param name="titre">varchar(255)</param>
        /// <param name="auteur">varchar(100)</param>
        /// <param name="idLivre">int(11)</param>
        public ReferenceMusique(int idReference, string nomImage, string titre, string auteur, int idLivre) : base(idReference, titre, nomImage, auteur, 2, 0, idLivre, ""){ }

        /// <summary>
        /// Permet d'envoyer la référence à l'API
        /// </summary>
        /// <param name="utilisateur">Utilisateur qui envoie la référence</param>
        /// <returns>
        /// true = code 201
        /// false = erreur
        /// </returns>
        public override bool PostReference(Utilisateur utilisateur)
        {
            return ClientRest.Instance.AppelSimple("?table=references&email=" + utilisateur.Email + "&password=" + utilisateur.Password + "&nomReference=" + NomReference + "&auteur=" + Auteur + "&nomImage=" + NomImage + "&idLivre=" + IdLivre + "&idType=" +IdType+ "", "POST");
        }

        /// <summary>
        /// Permet de modifier une référence par l'API
        /// </summary>
        /// <param name="utilisateur">Utilisateur qui modifie la référence</param>
        /// <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,
              Reference reference)
51          {
52              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");
53          }
54      }
55  }
56
```

```
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   * Decs.    : 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   * Decs.    : 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
              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
              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
              (DEFAULT_IDUTILISATEUR, email, password){ }
68          #endregion
69
70          #region Methodes
71          /// <summary>
72          /// Permet de faire une tentative de connexion avec les données
              de la classe
73          /// </summary>
74          /// <returns>
75          /// true = code 200 (connexion ok et l'utilisateur est connecté
              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=" +
                   _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
              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="
                    + _email + "&password=" + _password + "", "get");
94          }
95
96          #endregion
```

```
97      }
98  }
99
```

```php
<?php
/*  Projet  : API_MyLibrary
    Auteur  : Svoboda Karel Vilém
    Desc.   : API qui permet de gérer la base de données MyLibrary
    Date    : 18.05.2022
    Version : 1
*/

//autorisation des sources externes
header('Access-Control-Allow-Origin: *');
//définition du type d'application
header('Content-Type: application/json');

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

//require("models/sqlFunctions.php");

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

$response = null;

//vérification si l'utilisateur a transmit ses données
if (isset($_GET["email"]) && $_GET["password"]) {

    //Switch qui permet déterminer le type de requête de l'utilisateur
    switch ($_SERVER['REQUEST_METHOD']) {
        case 'GET':
            //maipulation de la session de l'utilisateur
            if (isset($_GET["session"])) {
                switch ($_GET["session"]) {
                        //connexion
                    case "connexion":
                        if ($mydatabase->statusConnexionUtilisateur($_GET["email"]) == 0) {

                            $user = new Utilisateur(0, $_GET["email"], $_GET["password"], 0);

                            //Si les données sont trouvées dans la base
                            if ($mydatabase->testConnexion($user)) {
                                http_response_code(200);
                                $response = array(
                                    "200" => "La session est activée",
                                );

                                //récupération des données de l'utilisateur
```

```
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, $mydatabase->recevoirUtilisateurParEmail($user->getEmail())->returnArrayForJSON());
56                          } else {
57                              http_response_code(400);
58                              $response = array(
59                                  "400" => "Fausses données de connexion",
60                              );
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"], $_GET
                      ["password"], 0);
101                 if ($mydatabase->statusConnexionUtilisateur($_GET
                      ["email"]) == 1) {
102                     switch ($_GET["table"]) {
103                         case "livres":
104                             if (isset($_GET["recherche"])) {
105                                 $user->setIdUtilisateur($mydatabase-
                      >recevoirUtilisateurParEmail($_GET["email"])-
                      >getIdUtilisateur());
106                                 http_response_code(200);
107                                 $response = array();
108                                 foreach ($mydatabase-
                      >rechercheLivreParAuteurOuTitre($user, $_GET
                      ["recherche"]) as $unLivre) {
109                                     array_push($response, $unLivre-
                      >returnArrayForJSON());
110                                 }
111                             }else if(isset($_GET["idLivre"])){
112                                 http_response_code(200);
113                                 $response = array();
114                                 $response = $mydatabase-
                      >rechercheLivreParIdLivre($_GET["idLivre"])-
                      >returnArrayForJSON();
115                             }
116                             else if(isset($_GET["tri"])){
117                                 http_response_code(200);
118                                 switch($_GET["tri"]){
119                                     case "dernier":
120                                         http_response_code(201);
121                                         $response = array();
122                                         $response = $mydatabase-
                      >dernierLivreUtilisateur($mydatabase-
                      >recevoirUtilisateurParEmail($_GET["email"]))-
                      >getIdLivre();
123                                         break;
124                                 }
125                             }
126                             else {
127                                 $user->setIdUtilisateur($mydatabase-
                      >recevoirUtilisateurParEmail($_GET["email"])-
                      >getIdUtilisateur());
128                                 //récupération des livres en fonction
                      de l'idUtilisateur
129                                 http_response_code(200);
```

```
130                                 $response = array();
131                                 foreach ($mydatabase-
                    >livresParUtilisateur($user) as $unLivre) {
132                                     array_push($response, $unLivre-
                    >returnArrayForJSON());
133                                 }
134                             }
135 
136                             break;
137                         case "references":
138                             if ($mydatabase->statusConnexionUtilisateur
                    ($_GET["email"]) == 1) {
139                                 if (isset($_GET["idLivre"])) {
140                                     http_response_code(200);
141                                     $response = array();
142                                     foreach($mydatabase-
                    >referencesParLivre(new Livre($_GET["idLivre"], "",
                    "", "", 0)) as $reference){
143                                         array_push($response,
                    $reference->returnArrayForJSON());
144                                     }
145                                 }
146                             }
147                             break;
148                         case "types":
149                             if ($mydatabase->statusConnexionUtilisateur
                    ($_GET["email"]) == 1) {
150                                 http_response_code(200);
151                                 $response = array();
152                                 foreach ($mydatabase->tousTypes() as
                    $unType) {
153                                     array_push($response, $unType-
                    >returnArrayForJSON());
154                                 }
155                             }
156                             break;
157                         default:
158                             break;
159                     }
160                 } else {
161                     http_response_code(403);
162                     $response = array(
163                         "Veuillez vous connecter pour poursuivre"
164                     );
165                 }
166             }
167 
168             break;
169         case "POST":
170 
171             //if($mydatabase->testConnexion(new Utilisateur()))
172             if ($mydatabase->statusConnexionUtilisateur($_GET["email"])
                == 1) {
```

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

```
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 
210                                     //envoie de la référence à la base
211                                     $mydatabase->ajouterReferenceLivre($referenceLivre);
212                                     http_response_code(201);
213                                     $response = $referenceLivre->returnArrayForJSON();
214                                 }
215                                 else{
216                                     http_response_code(401);
217                                 }
218                                 break;
219                             //musique
220                             case "2":
221                                 if(isset($_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 
235                                     $reference = new Reference(0, $_GET["nomReference"], "", "", $_GET["idType"], 0, $_GET["idLivre"], $_GET["descriptionLieu"]);
236                                     $mydatabase->ajouterReferenceLieu($reference);
237                                     http_response_code(201);
238                                     $response = $reference->
```

```
                    >returnArrayForJSON();
239                                     }
240                                     else{
241                                         http_response_code(401);
242                                     }
243                                     break;
244                                 default:
245                             }
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($_GET["email"]) ↵
                   == 1) {
263                 switch($_GET["table"]){
264                     case 'livres':
265                         if (isset($_GET["idLivre"]) && isset($_GET      ↵
                      ["titre"]) && isset($_GET["auteur"]) && isset($_GET   ↵
                      ["nomImage"])) {
266                         //Création d'un objet Livre à partir des        ↵
                      données dans le get
267                         $livre = new Livre($_GET["idLivre"], $_GET      ↵
                      ["titre"], $_GET["auteur"], $_GET["nomImage"],        ↵
                      $mydatabase->recevoirUtilisateurParEmail($_GET        ↵
                      ["email"])->getIdUtilisateur());
268                         $mydatabase->modifierLivre($livre);
269                         http_response_code(201);
270                         $response = $livre->returnArrayForJSON();
271                       }
272                       break;
273                       case 'references':
274                         if(isset($_GET["idReference"])){
275                             if(isset($_GET["idType"])){
276                                 switch($_GET["idType"]){
277                                     //livre
278                                     case "1":
279                                         if (isset($_GET                 ↵
                      ["livreReference"]) && isset($_GET["titre"]) && isset ↵
                      ($_GET["auteur"]) && isset($_GET["nomImage"])) {
280                                             //Création d'un objet Livre ↵
                       à partir des données dans le get
```

```
281                                     $livre = new Livre($_GET
["livreReference"], $_GET["titre"], $_GET["auteur"],
$_GET["nomImage"], $mydatabase-
>recevoirUtilisateurParEmail($_GET["email"])-
>getIdUtilisateur());
282                                     $mydatabase->modifierLivre
($livre);
283                                     http_response_code(201);
284                                     $response = $livre-
>returnArrayForJSON();
285                                 }
286                                 break;
287                             //musique
288                             case "2":
289                                 if(isset($_GET["idReference"])
&& isset($_GET["nomImage"]) && isset($_GET
["nomReference"]) && isset($_GET["auteur"]) && isset
($_GET["idLivre"])){
290                                     $reference = new Reference
($_GET["idReference"], $_GET["nomReference"], $_GET
["nomImage"], $_GET["auteur"], $_GET["idType"], 0,
$_GET["idLivre"], "");
291                                     $mydatabase-
>modifierReferenceMusique($reference);
292                                     http_response_code(201);
293                                     $response = $reference-
>returnArrayForJSON();
294                                 }
295                                 else{
296                                     http_response_code(401);
297                                 }
298                                 break;
299                             //lieu
300
301                             case '3':
302                                 if (isset($_GET
["nomReference"]) && isset($_GET["descriptionLieu"])
&& isset($_GET["idLivre"])) {
303                                     $reference = new Reference
($_GET["idReference"], $_GET["nomReference"], "", "",
 $_GET["idType"], 0, $_GET["idLivre"], $_GET
["descriptionLieu"]);
304                                     $mydatabase-
>modifierReferenceLieu($reference);
305                                     http_response_code(201);
306                                     $response = $reference-
>returnArrayForJSON();
307                                 }
308                                 else{
309                                     http_response_code(404);
310                                 }
311                                 //modifierReferenceLieu
312                                 break;
```

```
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"], "", "",
                    "", 0);
334                         $mydatabase->supprimerLivre($livre);
335                         break;
336                     case "references":
337                         if(isset($_GET["idReference"])){
338                             $reference = new Reference($_GET
                    ["idReference"], null, null, null, null, null, null,
                    null, null);
339                             $mydatabase->supprimerReference
                    ($reference);
340                             http_response_code(201);
341 
342                         }
343                         break;
344                 }
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 
```

```php
<?php
    /*  Projet  : API_MyLibrary
        Auteur  : Svoboda Karel Vilém
        Class   : Livre
        Desc.   : Permet de répliquer les données dans la table Livres
    */

    class Livre
    {
        //Variables d'instances
        private $_idLivre, $_titre, $_auteur, $_nomImage,
          $_idUtilisateur;

        //Propriétés
        public function getIdLivre(){
            return $this->_idLivre;
        }
        public function setIdLivre(int $idLivre){
            $this->_idLivre = $idLivre;
        }

        public function getTitre(){
            return $this->_titre;
        }
        public function setTitre(string $titre){
            $this->_titre = $titre;
        }

        public function getAuteur(){
            return $this->_auteur;
        }
        public function setAuteur(string $auteur){
            $this->_auteur = $auteur;
        }

        public function getNomImage(){
            return $this->_nomImage;
        }
        public function setNomImage(string $nomImage){
            $this->_nomImage = $nomImage;
        }

        public function getIdUtilisateur(){
            return $this->_idUtilisateur;
        }
        public function setIdUtilisateur(int $idUtilisateur){
            $this->_idUtilisateur = $idUtilisateur;
        }

        //Constructeur

        /** Permet de créer un livre
         *
```

```
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
              $auteur, string $nomImage, int $idUtilisateur)
60         {
61             $this->_idLivre = $idLivre;
62             $this->_titre = $titre;
63             $this->_auteur = $auteur;
64             $this->_nomImage = $nomImage;
65             $this->_idUtilisateur = $idUtilisateur;
66         }
67
68         //Fonctions
69
70         /** Permet de retourner un array avec les informations de
              l'objet
71          *
72          * @return array array formé pour JSON
73          */
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,
                  $livre->getIdLivre(), "");
86         }
87     }
88 ?>
```

```php
<?php
/*  Projet  : API_MyLibrary
    Auteur  : Svoboda Karel Vilém
    Class   : Livre
    Desc.   : Permet de répliquer les données dans la table Livres
*/

class MyLibrary
{
    // <!> À modifier lors de la mise en production <!>
    //Données de connexion à la base
    private $_serveur = 'mysql:host=localhost';
    private $_bdd = 'dbname=MyLibrary';
    private $_user = 'MyLibraryAPI';
    private $_mdp = 'r6.4>NRM`tC~y*gN';

    private $_connexionPDO = null;

    //Utilisateurs
    private $_retourUtilisateurParEmailPassword = null;
    private $_retourUtilisateurParEmail = null;
    private $_connexionUtilisateur = null;
    private $_deconnexionUtilisateur = null;
    private $_statusConnexionUtilisateur = null;

    //Livres
    private $_afficherLivreParIdUtilisateur = null;
    private $_rechercheLivreParIdLivre = null;
    private $_ajouterLivre = null;
    private $_rechercheLivreParAuteurOuTitre = null;
    private $_dernierLivreUtilisateur = null;

    //Références
    private $_referencesParLivre = null;
    private $_ajouterReference = null;


    private $_ajouterReferenceMusique;
    private $_modifierReferenceMusique;

    private $_ajouterReferenceLivre;

    private $_ajouterReferenceLieu;
    private $_modifierReferenceLieu;

    //Types
    private $_tousTypes = null;

    //Constructeur
    public function __construct()
    {
        try{
            $this->_connexionPDO = new PDO($this->_serveur.';'.$this-
```

```
                >_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
                password=:password";
62            $this->_retourUtilisateurParEmailPassword = $this-
                >_connexionPDO->prepare($sql);
63            $this->_retourUtilisateurParEmailPassword->setFetchMode
                (PDO::FETCH_ASSOC, 'MyLibrary');
64
65            $sql = "SELECT * FROM Utilisateurs WHERE email=:email";
66            $this->_retourUtilisateurParEmail = $this->_connexionPDO-
                >prepare($sql);
67            $this->_retourUtilisateurParEmail->setFetchMode
                (PDO::FETCH_ASSOC, 'MyLibrary');
68
69            $sql = "UPDATE Utilisateurs SET connecte=1 WHERE
                idUtilisateur=:idUtilisateur;";
70            $this->_connexionUtilisateur = $this->_connexionPDO-
                >prepare($sql);
71            $this->_connexionUtilisateur->setFetchMode
                (PDO::FETCH_ASSOC, 'MyLibrary');
72
73            $sql = "UPDATE Utilisateurs SET connecte=0 WHERE
                idUtilisateur=:idUtilisateur;";
74            $this->_deconnexionUtilisateur = $this->_connexionPDO-
                >prepare($sql);
75            $this->_deconnexionUtilisateur->setFetchMode
                (PDO::FETCH_ASSOC, 'MyLibrary');
76
77            $sql = "SELECT connecte FROM Utilisateurs WHERE
                email=:email";
78            $this->_statusConnexionUtilisateur = $this->_connexionPDO-
                >prepare($sql);
79            $this->_statusConnexionUtilisateur->setFetchMode
                (PDO::FETCH_ASSOC, 'MyLibrary');
80
81            //Livres
82            $sql = "INSERT INTO Livres (titre, auteur, nomImage,
                idUtilisateur) VALUES
                (:titre, :auteur, :nomImage, :idUtilisateur);";
83            $this->_ajouterLivre = $this->_connexionPDO->prepare($sql);
84            $this->_ajouterLivre->setFetchMode(PDO::FETCH_ASSOC,
                'MyLibrary');
85
86            $sql = "SELECT * FROM Livres WHERE idLivre=:idLivre";
87            $this->_rechercheLivreParIdLivre = $this->_connexionPDO-
                >prepare($sql);
```

```
88              $this->_rechercheLivreParIdLivre->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
89
90              $sql = "UPDATE Livres SET titre=:titre, auteur=:auteur,
                  nomImage=:nomImage WHERE idLivre=:idLivre;";
91              $this->_modifierLivre = $this->_connexionPDO->prepare
                  ($sql);
92              $this->_modifierLivre->setFetchMode(PDO::FETCH_ASSOC,
                  'MyLibrary');
93
94              $sql = "DELETE FROM Livres WHERE idLivre=:idLivre;";
95              $this->_supprimerLivre = $this->_connexionPDO->prepare
                  ($sql);
96              $this->_supprimerLivre->setFetchMode(PDO::FETCH_ASSOC,
                  'MyLibrary');
97
98              $sql = "SELECT * FROM Livres WHERE
                  idUtilisateur=:idUtilisateur";
99              $this->_afficherLivreParIdUtilisateur = $this-
                  >_connexionPDO->prepare($sql);
100             $this->_afficherLivreParIdUtilisateur->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
101
102             $sql = "SELECT * FROM Livres WHERE auteur LIKE :auteur OR
                  titre LIKE :titre AND idUtilisateur=:idUtilisateur";
103             $this->_rechercheLivreParAuteurOuTitre = $this-
                  >_connexionPDO->prepare($sql);
104             $this->_rechercheLivreParAuteurOuTitre->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
105             //dernierIdLivre
106             $sql = "SELECT * FROM Livres WHERE
                  idUtilisateur=:idUtilisateur ORDER BY idLivre DESC LIMIT
                  1";
107             $this->_dernierLivreUtilisateur = $this->_connexionPDO-
                  >prepare($sql);
108             $this->_dernierLivreUtilisateur->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
109
110             //Références
111             $sql = "SELECT * FROM MyLibrary.References WHERE
                  idLivre=:idLivre";
112             $this->_referencesParLivre = $this->_connexionPDO->prepare
                  ($sql);
113             $this->_referencesParLivre->setFetchMode(PDO::FETCH_ASSOC,
                  'MyLibrary');
114
115             $sql = "DELETE FROM MyLibrary.References WHERE
                  idReference=:idReference;";
116             $this->_supprimerReference = $this->_connexionPDO->prepare
                  ($sql);
117             $this->_supprimerReference->setFetchMode(PDO::FETCH_ASSOC,
                  'MyLibrary');
118
```

```
119                 // Référence type musique
120             $sql = "INSERT INTO MyLibrary.References (nomReference,
                  nomImage, idType, auteur, idLivre) VALUES
                  (:nomReference, :nomImage, :idType, :auteur, :idLivre);";
121             $this->_ajouterReferenceMusique = $this->_connexionPDO-
                  >prepare($sql);
122             $this->_ajouterReferenceMusique->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
123
124             $sql = "UPDATE MyLibrary.References SET
                  nomReference=:nomReference, nomImage=:nomImage,
                  idType=:idType, auteur=:auteur, idLivre=:idLivre WHERE
                  idReference=:idReference;";
125             $this->_modifierReferenceMusique = $this->_connexionPDO-
                  >prepare($sql);
126             $this->_modifierReferenceMusique->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
127
128                 // Référence type Livre
129             $sql = "INSERT INTO MyLibrary.References (livreReference,
                  idType, idLivre) VALUES
                  (:livreReference, :idType, :idLivre);";
130             $this->_ajouterReferenceLivre = $this->_connexionPDO-
                  >prepare($sql);
131             $this->_ajouterReferenceLivre->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
132
133                 // Référence type Lieu
134             $sql = "INSERT INTO MyLibrary.References (nomReference,
                  idType, idLivre, descriptionLieu) VALUES
                  (:nomReference, :idType, :idLivre, :descriptionLieu);";
135                 $this->_ajouterReferenceLieu = $this->_connexionPDO-
                      >prepare($sql);
136                 $this->_ajouterReferenceLieu->setFetchMode
                      (PDO::FETCH_ASSOC, 'MyLibrary');
137
138                 $sql = "UPDATE MyLibrary.References SET
                      nomReference=:nomReference, idType=:idType,
                      idLivre=:idLivre, descriptionLieu=:descriptionLieu
                      WHERE idReference=:idReference;";
139             $this->_modifierReferenceLieu = $this->_connexionPDO-
                  >prepare($sql);
140             $this->_modifierReferenceLieu->setFetchMode
                  (PDO::FETCH_ASSOC, 'MyLibrary');
141
142             //Types
143             $sql = "SELECT * FROM Types";
144             $this->_tousTypes = $this->_connexionPDO->prepare($sql);
145             $this->_tousTypes->setFetchMode(PDO::FETCH_ASSOC,
                  'MyLibrary');
146
147         }
148         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
            (':email' => $utilisateur->getEmail(), ':password' =>
            $utilisateur->getPassword()));
184         return isset($this->_retourUtilisateurParEmailPassword-
            >fetchAll()[0][0]);
185     }
186 
187     /** Retourne un utilisateur par email
188      *
189      * @string email de l'utilisateur à rechercher
190      *
191      * @return Utilisateur utilisateur avec par email
192      */
193     public function recevoirUtilisateurParEmail(string $email){
194         $this->_retourUtilisateurParEmail->execute(array(':email' =>
            $email));
```

```
195         $resultat = $this->_retourUtilisateurParEmail->fetchAll();
196         return new Utilisateur($resultat[0][0], $resultat[0][1],
              $resultat[0][2], $resultat[0][3]);
197     }
198
199     /** retourne si un utilisateur est connecté en fonction de son
          email
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' =>
              $email));
207         $resultat = $this->_statusConnexionUtilisateur->fetchAll();
208         return $resultat[0][0];
209     }
210
211     /** Permet d'ajouter un livre
212      *
213      * @Livre livre à ajouter
214      * @return null
215      */
216     public function ajouterLivre(Livre $livre){
217         $this->_ajouterLivre->execute(array(':titre' => $livre-
              >getTitre(), ':auteur' => $livre->getAuteur(), ':nomImage' =>
               $livre->getNomImage(), ':idUtilisateur' => $livre-
              >getIdUtilisateur()));
218     }
219
220     /** Permet de modifier un livre
221      *
222      * @Livre livre AVEC les données modifiées sauf les IDs
223      * @return null
224      */
225     public function modifierLivre(Livre $livre){
226         $this->_modifierLivre->execute(array(':idLivre'=>$livre-
              >getIdLivre(),':titre' => $livre->getTitre(), ':auteur' =>
              $livre->getAuteur(), ':nomImage' => $livre->getNomImage()));
227     }
228
229
230     /** Permet de supprimer un livre
231      *
232      * @Livre livre à supprimer
233      * @return null
234      */
235     public function supprimerLivre(Livre $livre){
236         $this->_supprimerLivre->execute(array(':idLivre'=>$livre-
              >getIdLivre()));
237     }
238
```

```php
    /** Permet de retourener tous les livres d'utilisateur
     *
     * @Utilisateur utilisateur en question
     * @return array(Livres) retourne la liste des Livres
     */
    public function livresParUtilisateur(Utilisateur $utilisateur){
        $this->_afficherLivreParIdUtilisateur->execute(array(':idUtilisateur' => $utilisateur->getIdUtilisateur()));
        $resultat = $this->_afficherLivreParIdUtilisateur->fetchAll();
        $arrayLivre = array();
        foreach($resultat as $unLivre){
            array_push($arrayLivre, new Livre($unLivre[0], $unLivre[1], $unLivre[2], $unLivre[3], $unLivre[4]));
        }
        return $arrayLivre;
    }

    /** Permet de récupérer le dernier livre d'un utilisateur
     *
     * @Utilisateur utilisateur
     */
    public function dernierLivreUtilisateur(Utilisateur $utilisateur){
        $this->_dernierLivreUtilisateur->execute(array(':idUtilisateur' => $utilisateur->getIdUtilisateur()));
        $resultat = $this->_dernierLivreUtilisateur->fetchAll();
        $arrayLivre = array();
        foreach($resultat as $unLivre){
            array_push($arrayLivre, new Livre($unLivre[0], $unLivre[1], $unLivre[2], $unLivre[3], $unLivre[4]));
        }
        return $arrayLivre[0];
    }

    /** Permet de faire une recherche grâce à un filtre personnalisé.
      Le programme va chercher dans la base les éléments de
      l'utilisateur avec les filtres en quesiton sous les tables Auteur
      et Titre
     *
     * @Utilisateur $utilisateur utilisateur qui fait la requête
     * @string chaîne de caractères à rechercher
     */
    public function rechercheLivreParAuteurOuTitre(Utilisateur $utilisateur, string $recherche){
        $this->_rechercheLivreParAuteurOuTitre->execute(array(':idUtilisateur' => $utilisateur->getIdUtilisateur(), ':auteur' => "%".$recherche."%", ':titre' => "%".$recherche."%"));
        $resultat = $this->_rechercheLivreParAuteurOuTitre->fetchAll();
        //Création d'une instance de Livre pour chaque donnée trouvée et retour du résultat dans un array
        $arrayLivre = array();
        foreach($resultat as $unLivre){
            array_push($arrayLivre, new Livre($unLivre[0], $unLivre[1],
```

```
                $unLivre[2], $unLivre[3], $unLivre[4]));
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-
              >getIdLivre()));
297         $resultat = $this->_referencesParLivre->fetchAll();
298         $arrayReference = array();
299         foreach($resultat as $uneReference){
300             array_push($arrayReference, new Reference($uneReference[0],
                  $uneReference[1], $uneReference[2], $uneReference[3],
                 $uneReference[4], $uneReference[5], $uneReference[6],
                 $uneReference[7]));
301         }
302         return $arrayReference;
303     }
304 
305     public function rechercheLivreParIdLivre(int $idLivre){
306         $this->_rechercheLivreParIdLivre->execute(array(':idLivre' =>
              $idLivre));
307         $resultat = $this->_rechercheLivreParIdLivre->fetchAll();
308         return new Livre($resultat[0][0], $resultat[0][1], $resultat[0]
              [2], $resultat[0][3], $resultat[0][4]);
309     }
310 
311     public function ajouterReference(Reference $reference){
312         error_log($this->_ajouterReference->execute(array
              (':nomReference' => $reference->getNomReference(),
              ':nomImage' => $reference->getNomImage(), ':auteur' =>
              $reference->getAuteur(), ':idType' => $reference->getIdType
              (), ':livreReference' => $reference->getLivreReference(),
              ':resume' => $reference->getDescriptionLieu())));
313     }
314 
315     public function ajouterReferenceMusique(Reference $reference){
316         $this->_ajouterReferenceMusique->execute(array(':nomReference'
              => $reference->getNomReference(), ':nomImage' => $reference-
              >getNomImage(), ':auteur' => $reference->getAuteur(),
              ':idLivre' => $reference->getIdLivre(),  ':idType' =>
              $reference->getIdType()));
```

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

```php
<?php
    /*  Projet  : API_MyLibrary
        Auteur  : Svoboda Karel Vilém
        Class   : Reference
        Desc.   : Permet de répliquer les données dans la table References
    */

    class Reference
    {
        //Variables d'instances
        private $_idReference, $_nomReference, $_nomImage, $_auteur, $_idType, $_livreReference, $_idLivre, $_descriptionLieu;

        //Propriétés
        public function getIdReference(){
            return $this->_idReference;
        }
        public function setIdReference($idReference){
            $this->_idReference = $idReference;
        }

        public function getNomReference(){
            return $this->_nomReference;
        }
        public function setNomReference($nomReference){
            $this->_nomReference = $nomReference;
        }

        public function getNomImage(){
            return $this->_nomImage;
        }
        public function setNomImage($nomImage){
            $this->_nomImage = $nomImage;
        }

        public function getAuteur(){
            return $this->_auteur;
        }
        public function setAuteur($auteur){
            $this->_auteur = $auteur;
        }

        public function getIdType(){
            return $this->_idType;
        }
        public function setIdType($idType){
            $this->_idType = $idType;
        }

        public function getLivreReference(){
            return $this->_livreReference;
        }
```

```php
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,
              $auteur, $idType, $livreReference, $idLivre,
              $descriptionLieu)
83          {
84              $this->_idReference = $idReference;
85              $this->_nomReference = $nomReference;
86              $this->_nomImage = $nomImage;
87              $this->_auteur = $auteur;
88              $this->_idType = $idType;
89              $this->_livreReference = $livreReference;
90              $this->_idLivre = $idLivre;
91              $this->_descriptionLieu = $descriptionLieu;
92          }
93
94          //Fonctions
95
96          /** Permet de retourner un array avec les informations de
              l'objet
97           *
98           * @return array array formé pour JSON
99           */
100         public function returnArrayForJSON() : array
101         {
```

```
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 ?>
```

```php
<?php
    /*  Projet  : API_MyLibrary
        Auteur  : Svoboda Karel Vilém
        Class   : Type
        Desc.   : Permet de répliquer les données dans la table Types
    */

    class Type
    {
        //Variables d'instances
        private $_idType, $_nomType;

        //Propriétés
        public function getIdType(){
            return $this->_idType;
        }
        public function setIdType($idType){
            $this->_idType = $idType;
        }

        public function getNomType(){
            return $this->_nomType;
        }
        public function setNomType($nomType){
            $this->_nomType = $nomType;
        }

        //Constructeur

        /** Permet de créer un Type de reference
         *
         *  @param integer int(11) $idType
         *  @param string varchar(255) $nomType
         */
        function __construct(int $idType, string $nomType)
        {
            $this->_idType = $idType;
            $this->_nomType = $nomType;
        }

        //Fonctions

        /** Permet de retourner un array avec les informations de
          l'objet
         *
         * @return array array formé pour JSON
         */
        public function returnArrayForJSON(){
            return array(
                "idType" => $this->_idType,
                "nomType" => $this->_nomType,
            );
        }
```

```
53      }
54  ?>
```

```php
<?php
    /*  Projet  : API_MyLibrary
        Auteur  : Svoboda Karel Vilém
        Class   : Type
        Desc.   : Permet de répliquer les données dans la table Utilisateurs
    */

    class Utilisateur
    {
        //Variables d'instances
        private $_idUtilisateur, $_email, $_password, $_connecte;

        //Propriétés
        public function getIdUtilisateur(){
            return $this->_idUtilisateur;
        }
        public function setIdUtilisateur(int $idUtilisateur){
            $this->_idUtilisateur = $idUtilisateur;
        }

        public function getEmail(){
            return $this->_email;
        }
        public function setEmail(string $email){
            $this->_email = $email;
        }

        public function getPassword(){
            return $this->_password;
        }
        public function setPassword(string $password){
            $this->_password = $password;
        }

        public function getConnecte(){
            return $this->_connecte;
        }
        public function setConnecte(bool $connecte){
            $this->_connecte = $connecte;
        }

        //Constructeur

        /** Permet de créer un Utilisateur
         *
         *  @param integer int(11) $idUtilisateur
         *  @param string varchar(255) $email
         *  @param string varchar(255) $password
         */
        function __construct(int $idUtilisateur, string $email, string $password, bool $connecte)
        {
```

```php
            $this->_idUtilisateur = $idUtilisateur;
            $this->_email = $email;
            $this->_password = $password;
            $this->_connecte = $connecte;
        }

        //Fonctions

        /** Permet de retourner un array avec les informations de l'objet
         *
         * @return array array formé pour JSON
         */
        public function returnArrayForJSON(){
            return array(
                "idUtilisateur" => $this->_idUtilisateur,
                "email" => $this->_email,
                "password" => $this->_password,
                "connecte" => $this->_connecte
            );
        }
    }
?>
```