Projet C#

Conception d'une bibliothèque de films

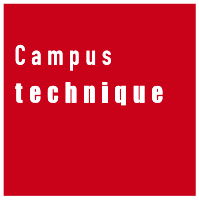
1er Bachelier en Informatique

Techniques Informatiques II

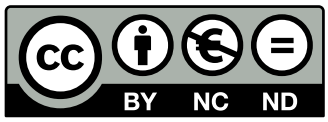
(Cours)

Auteur :
Terencio AGOZZINO

Enseignant :
Antoine MALAISE



Année académique 2015 - 2016



Je remercie en particulier
Monsieur Antoine MALAISE,
enseignant du cours de C#
pour ses conseils dans le but
de l'amélioration de ce projet.

Ainsi que Monsieur Fabrice SCOPEL,
enseignant concernant les laboratoires
de bases de programmation pour ses aides
face à divers problèmes algorithmiques.

Table des matières

1 Introduction

Dans le cadre de mon cours "Techniques Informatiques", j'ai dû réaliser un projet en C#. Le but de ce projet était de réaliser une bibliothèque de films en faisant abstraction de technologies, frameworks ou autres aides lors de la conception de celle-ci. Cela permettant d'apprendre et de comprendre ces outils au sein de la programmation.

De ce fait, il était important de veiller à la complexité notamment lors de la conception de l'interface graphique, de la sérialisation, d'algorithmes, ... tout cela en utilisant uniquement ce que les *Windows Forms* peuvent fournir.

De plus, l'utilisation de base de données SQL était interdite pour l'enregistrement des films, ce qui m'a amené à utiliser des *Lists* et un *DataGridView*.

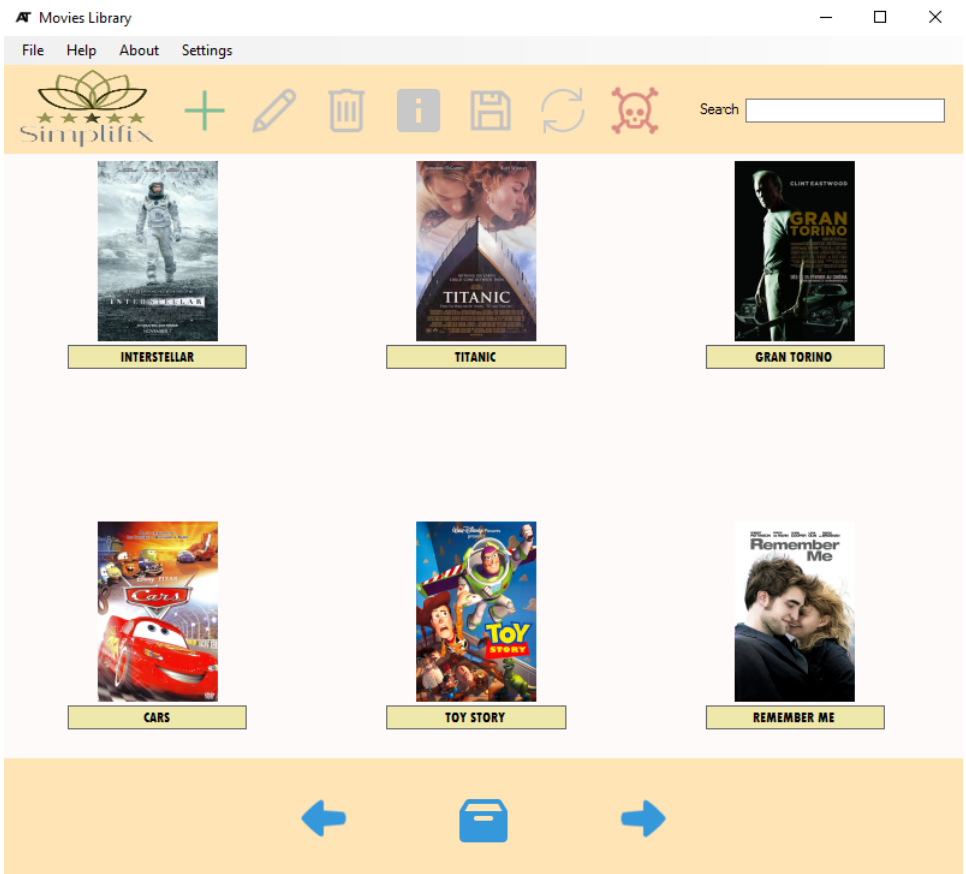


FIGURE 1 – Preview

2 Argumentation des choix personnels effectués

2.1 Rapport

J'ai décidé de faire le rapport en LaTeX pour les nombreux avantages qu'offrent LaTeX. En effet, LaTeX offre une mise en page claire et nette permettant de se focaliser sur le contenu et non sur la forme. De plus, celui-ci dispose d'un panel assez large de fonctionnalités qui facilitent la rédaction d'un rapport.

FIGURE 2 – Logo LaTeX

2.2 Héritage

L'héritage est presque inexistant pour la simple et bonne raison que concernant les fonctionnalités de base de ce projet, faire de l'héritage n'était pas spécialement nécessaire. S'il était nécessaire de développer en profondeur ce projet, comme notamment en offrant la possibilité à l'utilisateur d'ajouter une série, il aurait été plus judicieux d'avoir une approche orientée objet.

2.3 Interface graphique

L'entiéreté de la GUI (*Graphical User Interface*) a été conçue de manière exclusivement personnelle, ce qui laisse sous-entendre que cette interface graphique n'est inspirée d'aucune autre.

L'ajout, l'édition et l'affichage d'informations de films ont été réalisés à l'aide d'un seul *Panel* en manipulant la visibilité des *Labels*, *Textbox* et des *Buttons* afin d'éviter de devoir allouer de la mémoire pour la création d'autres composants et pour optimiser la vitesse en n'initilisant les composants qu'une seule fois.

Pour les *Layouts*, la *Windows Forms* est composé d'un *Layout* situé en haut, définissant les boutons liés aux commandes de manipulation de films, un autre, situé en bas, délimitant les boutons de navigation de films et un troisième situé au centre pour l'affichage de films. Concernant le *Layout* central, celui-ci laisse sa place à un autre *Layout* qui contient un *DataGridView* lors de l'affichage de tous les films, cela permettant également une optimisation de la mémoire et de la vitesse.

3 Implémentations des différentes fonctions

3.1 Sérialisation

```
1         /// <summary>
2         /// Serializes the list of data to the designated file path.
3         /// </summary>
4         /// <typeparam name="T">type of Object to serialize</typeparam>
5         /// <param name="list">object list to serialize</param>
6         /// <param name="path">path for the text file</param>
7         /// <returns>A string representing serialized data</returns>
8         public static string Serialize<T>(List<T> list, string path) {
9             string json = JsonConvert.SerializeObject(list, Formatting.Indented);
10            File.WriteAllText(path, json);
11            return json;
12        }
```

Les films de la bibliothèque sont sérialisés avec JSON en vue d'un poids et d'une vitesse de récupération plus moindre qu'offre la sérialisation binaire. Davantage, JSON a l'avantage de permettre de sérialiser les films dans un fichier texte propre et lisible. De plus, cette méthode est générique permettant de fonctionner avec diverses *Lists*.

3.2 Désérialisation

```
1         /// <summary>
2         /// Deserializes the path designated  to a list of data.
3         /// </summary>
4         /// <typeparam name="T">type of Object to deserialize</typeparam>
5         /// <param name="path">path for the text file</param>
6         /// <returns>A deserialized list</returns>
7         public static List<T> Deserialize<T>(string path)  {
8             List<T> dblDeserialize = new List<T>();
9             if (File.Exists(path)) {
10                string json = File.ReadAllText(path);
11                dblDeserialize = JsonConvert.DeserializeObject<List<T>>(json);
12            }
13            return dblDeserialize;
14        }
```

Les films de la bibliothèque sont de même désérialisés avec JSON pour les mêmes avantages que cité ci-dessus. Cette méthode est de même générique, afin d'optimiser l'évolutivité.

4 Les points forts du projet

— L'interface graphique est ergonomique et simple d'utilisation regroupant en grande partie, les fonctionnalités de bases que peut fournir une bibliothèque de films.

— L'implémentation est aisée à lire, avec une documentation écrite pour chaque méthode et en respectant au maximum les conventions du langage.

Exemple de documentation :

```
    /// <summary>
    /// A random documentation.
    /// </summary>
    /// <param name="first parameter"></param>
    /// <param name="second parameter"></param>
    /// <param name="third parameter"></param>
    /// <returns>something to return></returns>
```

— Le programme a été conçu pour être le plus optimisé possible selon les restrictions. De même, l'interface graphique est créée en outre dynamiquement, ce qui permet une optimisation du code.

Exemple d'implémentation dynamique pour la création de boutons liés au controlleur :

```
  /// <summary>
  /// Adds Buttons to the Form.
  /// </summary>
  private void AddButtonsForm() {
    for (int i = 0; i < 10; i++) {
      cmdForm[i] = new Button();
      if (i < 7) {
        cmdForm[i].Location = new Point(140 + (i * 60), 11);
        tlpTop.Controls.Add(cmdForm[i], i + 1, 0);
      } else {
        cmdForm[i].Location = new Point(240 + ((i % 6) * 130), 25);
        tlpBottom.Controls.Add(cmdForm[i], (i + 1) % 7, 0);
      }
      cmdForm[i].Anchor = AnchorStyles.None;
      cmdForm[i].Click += new EventHandler(ButtonClick);
      cmdForm[i].FlatAppearance.BorderSize = 0;
      cmdForm[i].FlatAppearance.MouseOverBackColor = Color.Transparent;
      cmdForm[i].FlatStyle = FlatStyle.Flat;
      cmdForm[i].Image = dblDefaultImages[i];
      cmdForm[i].Name = "cmdDynamic" + i;
      cmdForm[i].MouseEnter += new EventHandler(ButtonEnter);
      cmdForm[i].MouseLeave += new EventHandler(ButtonLeave);
      cmdForm[i].Size = new Size(48, 48);
      cmdForm[i].Tag = i + 1;
      cmdForm[i].TabIndex = i + 1;
    }
  }
```

5 Les points faibles du projet

— Structure du projet très pauvre, ce qui rend le projet peu évolutif.
Effectivement, le projet possède la structure suivante :

  — Movies.Models
    — Movie

  — Movies.Run
    — Program

  — Movies.View
    — MovieForm.Designer.cs

    — MovieForm.cs

  Il aurait fallu davantage structurer le projet en plusieurs classes et packages, notamment en créant un package *Movies.Mouse* comportant une nouvelle *Mouse.cs* s'occupant de toutes les interactions liées à la souris.

— Projet également peu évolutif suite à l'utilisation de *Windows Forms* suite aux restrictions du projet. Dans le cadre d'un projet évolutif, il aurait été nécessaire d'utiliser un WPF (*Windows Presentation Foundation*) avec une structure en MVVM (*Model-View-ViewModel*).

— Application non fluide suite à l'usage de *Windows Forms*.

6 Quelques erreurs du programme

— Positionnement du *DataGridView* non adapté pour l'affichage des derniers films de la liste dû au remplissage du *Layout* par celui-ci. Néanmoins, il est possible d'afficher les derniers films de la liste à l'aide des en-têtes.

— Risque d'une levée d'exception *IndexOutOfRangeException* lors d'une séquence de suppression en rafale à l'aide du mnémonique ($CTRL + W$) dans le *DataGridView*.

— Non-affichage de l'image lors de l'édition de l'image d'un film. Cela est dû au fait que la *PictureBox* contenant ce film, ne trouve pas le *Path* de la nouvelle image vu que celle-ci est ajouté uniquement si l'utilisateur décide de sauvegarder l'image afin de ne pas avoir de fichiers contenant des images non utilisable ultérieurement.

Remarque : il est possible de contrer ce problème, notamment en créant un dossier temporaire '*tmp*' stockant les images ayant été modifiés et supprimer celui-ci lors de la sauvegarde.

— Les énoncés de ce projet laissaientt sous-entendre qu'il fallait donner la possibilité à l'utilisateur de pouvoir choisir une bibliothèque lors du démarrage de l'application. Cette recommandation n'a pas été respecté afin de ne pas nuire à l'ergonomie. Effectivement, pour respectée cette recommandation, il aurait fallu créer un *MessageBox* au début de l'application, ce qui aurait été esthétiquement peu plaisant (vu que l'interface graphique s'afficherait uniquement après la disparition du *MessageBox*.

Remarque : il est toutefois possible de remplacer sa bibliothèque en remplaçant le fichier '*json.txt*' → la raison d'avoir laissé le fichier texte modifidable par l'utilisateur.

— Risque d'une levée d'exception *OutOfMemoryException* lors d'une recherche de plusieurs films très rapidement. Ce problème est notamment dû à l'utilisateur de *PictureBox* et donc de *Windows Forms*.

— Les choix de visualisation de l'interface ne sont pas sauvegardés.

Remarque : cela était faisable en attribuant une valeur binaire '*1*' ou une variable bool '*true*' pour la couleur des bordures et de même pour la couleur de fond.

7 Apports positifs/négatifs de ce projet

N'ayant pas de connaissance particulière dans le C# avant la réalisation de ce projet, celui-ci m'a permis d'approfondir, ainsi qu'adapter mes notions provenant d'autres langages précédemment utilisés.

8 Manuel d'utilisateur

8.1 Liste des mnémoniques

1. File
   - CTRL + N : création d'un nouveau film.
   - CTRL + E : édition du film.
   - CTRL + W : supression du film.
   - CTRL + I : informations du film.
   - CTRL + S : sauvegarde des modifications.
   - CTRL + L : chargement de la précédente sauvegarde.
   - CTRL + R : réinitialisation de la bibliothèque de films.
   - CTRL + LEFT : page précédente.
   - CTRL + B : affichage de tous les films.
   - CTRL + RIGHT : page suivante.
   - CTRL + Q : quitter l'application.
2. Help
   - CTRL + H : aide.
3. About
   - ALT + A : à propos de l'auteur.
4. Settings
   - Color's Border : application d'une couleur aux bordures.
   - Color's Center : application d'une couleur centrale.