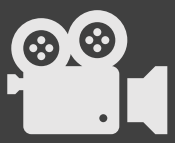
RAPPORT CDAA

Windows Form



BAH Mamadou REPECAUD Benjamin

TABLE DES MATIÈRES

I CAHIER DES CHARGES DE L'APPLICATION

L'objectif de l'application est, pour un cinéma choisi, d'accéder aux films qu'il projette.

Le cinéma choisi se fait soit par sélection d'une ville, soit par la saisie du nom du cinéma recherché.

L'utilisateur peut aussi filtrer le résultat, en n'ayant accès qu'aux films à l'affiche / prochainement, ou encore à toutes les comédies projetées dans le cinéma. Un film peut être sélectionné pour avoir accès à toutes ses informations et aux acteurs qui le composent.

Hormis les fonctionnalités de recherche et d'affichage, l'application permet la gestion des films projetés. On peut ainsi ajouter un film au cinéma courant ou modifier ceux déjà projetés.

L'intérêt d'une telle application est d'avoir tous les éléments globaux placés sur une seule et même fenêtre (hormis l'ajout, la modification et la description détaillée du film).

II ANALYSE

A- Informations à manipuler

L'application montre les films d'un cinéma sélectionné. Pour ce faire, il faut manipuler les différents attributs des classes qui ont été créées.

La majorité des informations à traiter sont des listes d'objets (cinémas à choisir, plusieurs films, nationalité à choisir…), par conséquent, l'utilisateur doit pouvoir le faire de manière intuitive et simple.

La diversité visuelle est également quelque chose d'important pour l'utilisateur, il doit pouvoir naviguer dans l'application sans être confronté à une certaine lassitude.

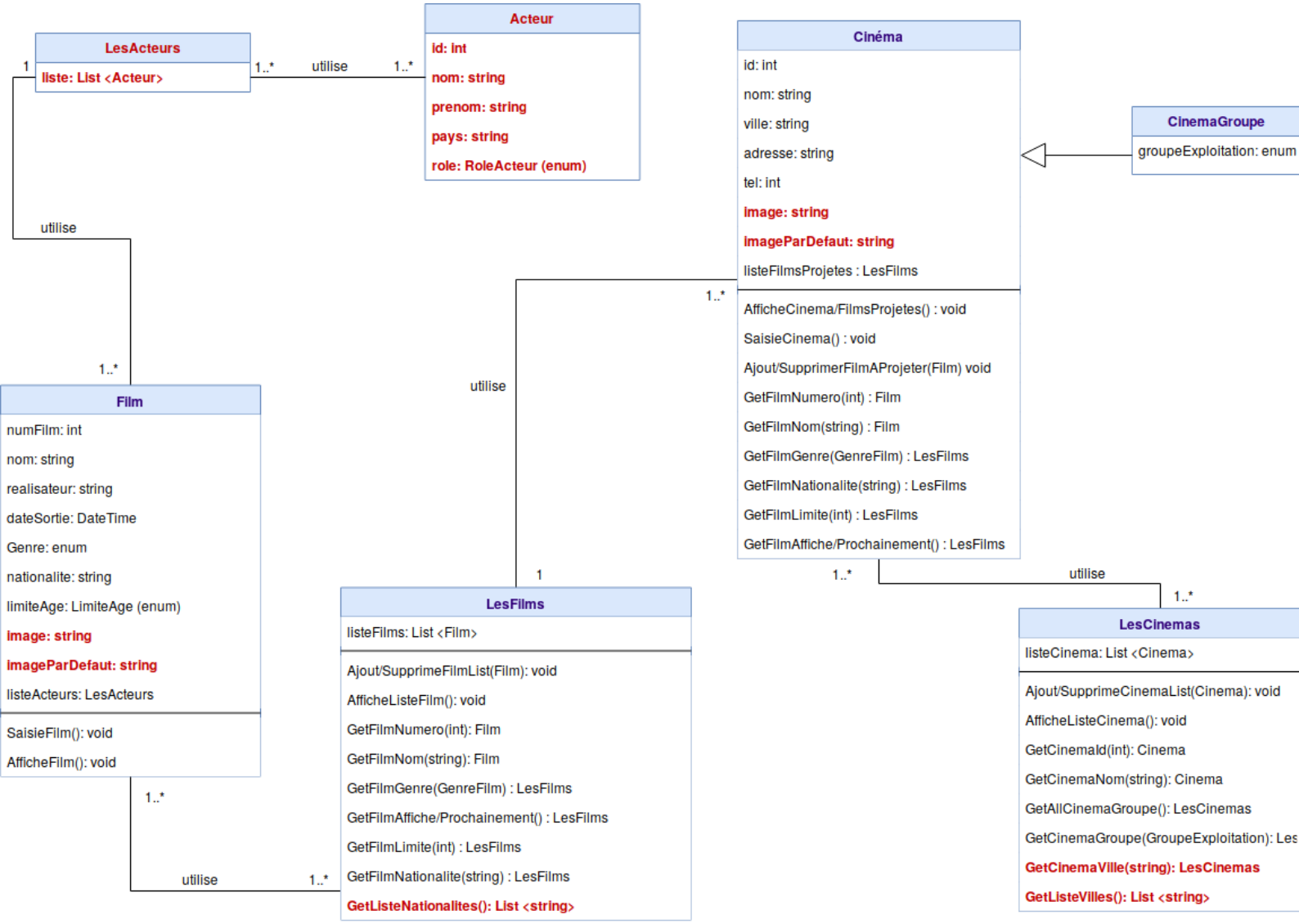
L'application est ainsi formée de différents composants disponibles dans Visual Studio qui répondent à la fois au besoin technique (manipulation des données) et au besoin visuel.

De nouveaux besoins sont également apparus.

Un film a dorénavant une affiche (attribut image et imageParDefaut) et une liste d'acteurs. Une classe Acteur et LesActeurs (contenant la liste des acteurs) ont ainsi été créées, avec l'initialisation habituelle des jeux de données ajoutés à chaque film. Des méthodes de recherche pour récupérer une liste de villes de tous les cinémas ou des nationalités des films ont également été ajoutées dans leurs classes respectives.

Tout comme le film, le cinéma a également un attribut image et imageParDefaut.

B- Diagramme UML des classes



En rouge : les ajouts depuis la partie 1

III DIAGRAMME DE NAVIGATION

L'application se résume à quatre fenêtres différentes :

- Page principale (Form1)
- Page d'ajout d'un film (Ajouter)
- Page de modification des films (Modifier)
- Page descriptive du film sélectionné (DescFilm)

Elle s'articule de cette façon :

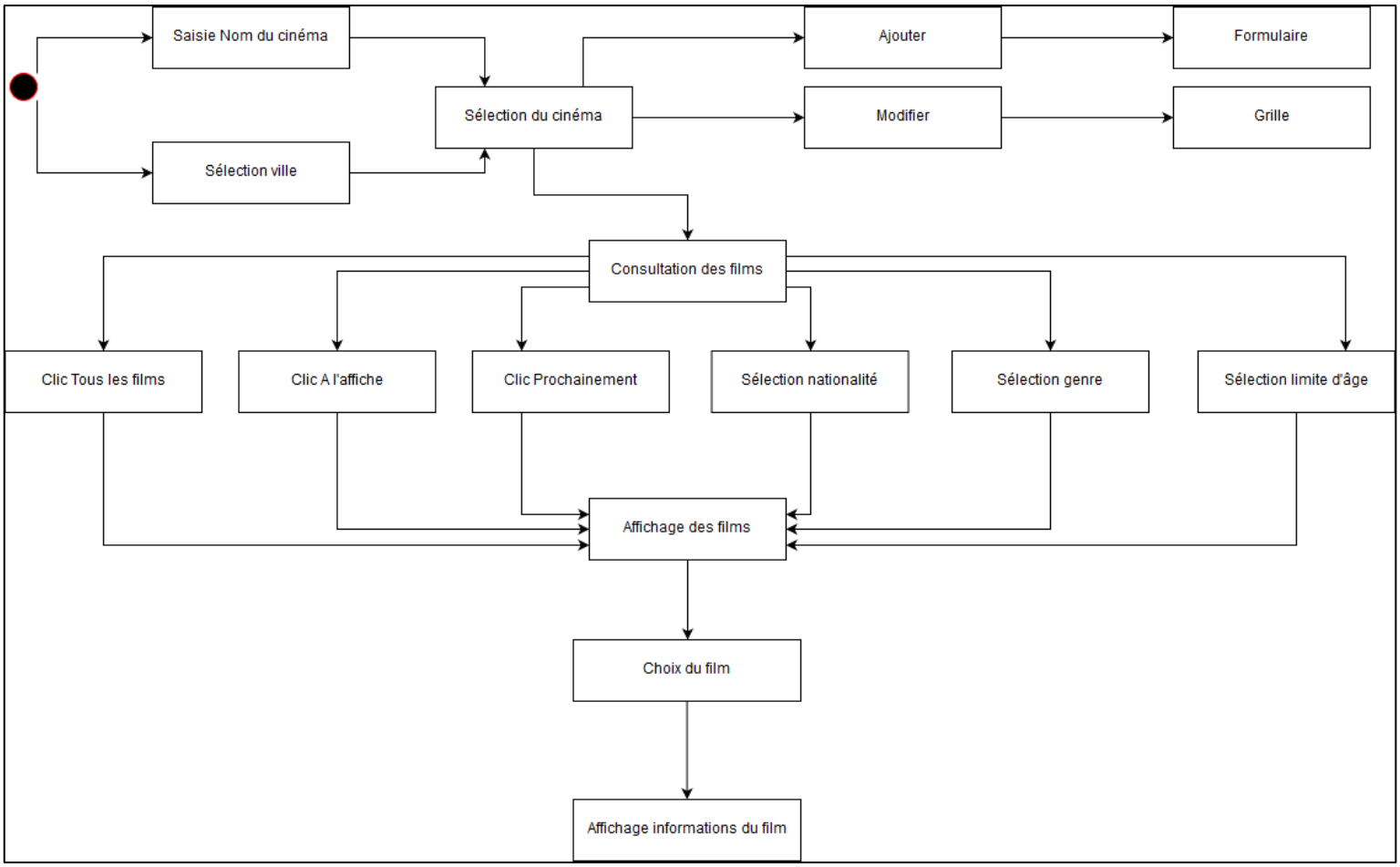
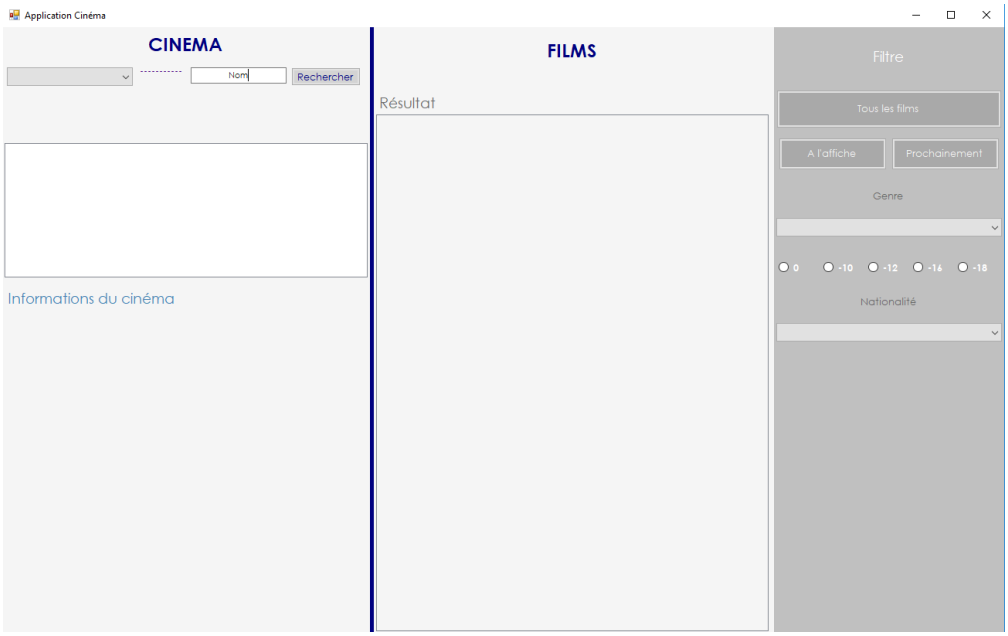


Diagramme de navigation de l'application, du point de départ jusqu'à la fin. Chaque case découle de la case qui lui est reliée. (Clic prochainement entraîne l'affichage des films)

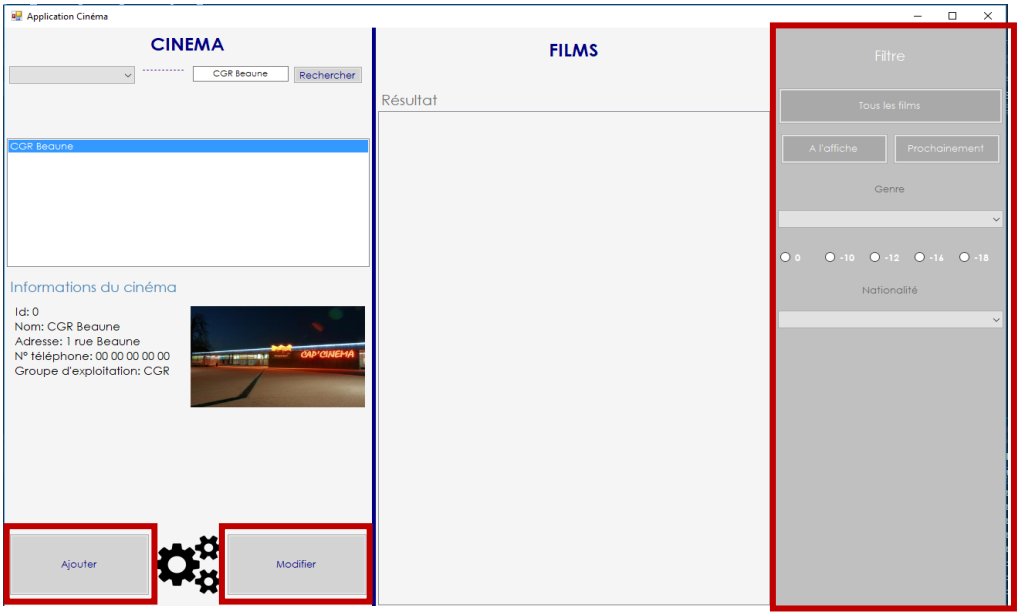
A- Choisir un cinéma



Page principale de l'application

Pour accéder aux trois dernières, l'utilisateur doit sélectionner un cinéma, soit en choisissant une ville et sélectionner le cinéma qu'il souhaite dans cette ville, soit en saisissant le nom du cinéma.

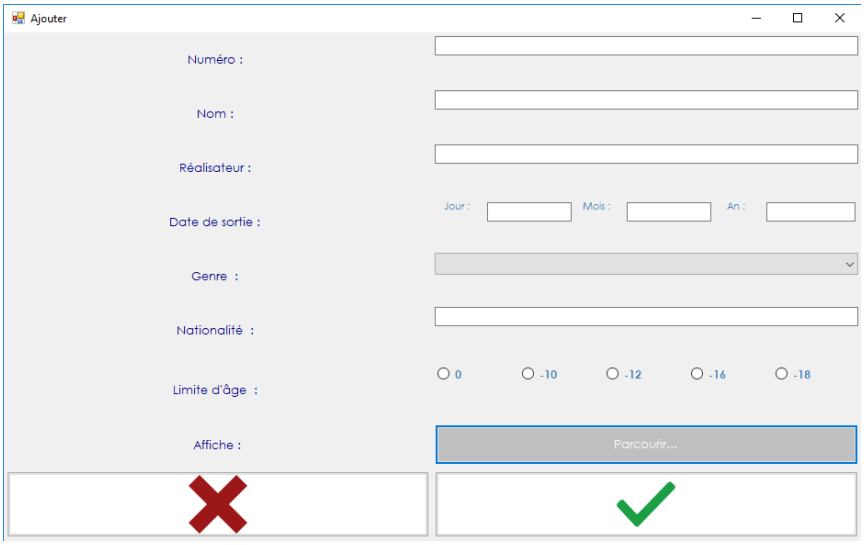
Trois choix s'offrent ensuite à lui : ajouter un film au cinéma sélectionné, modifier les films qu'ils projettent, ou tout simplement consulter ses films dans la partie droite de la page principale.



Affichage cinéma choisi

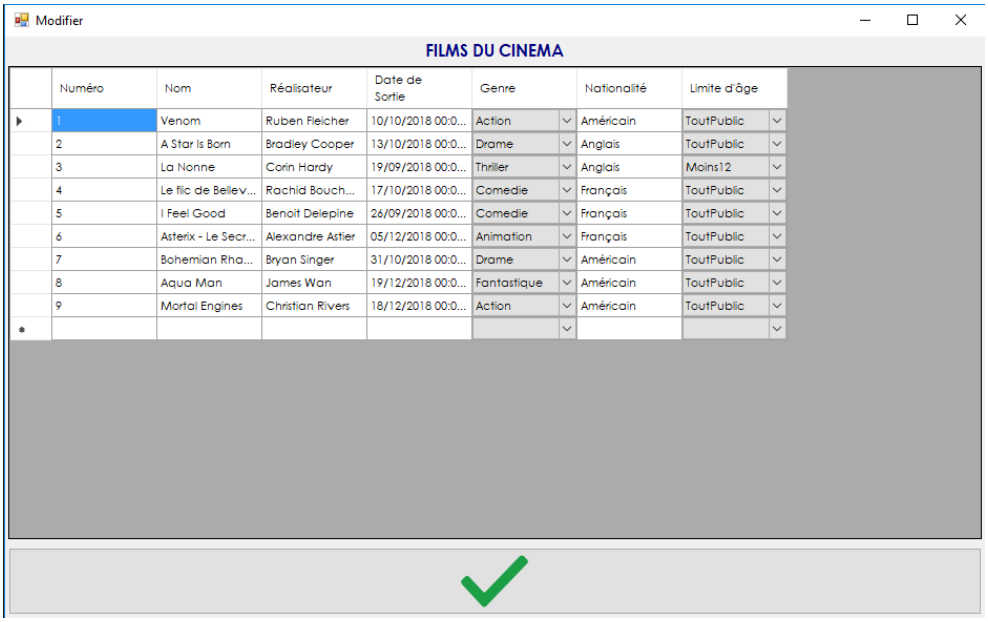
B- Ajouter un film

Ajouter un film provoque l'ouverture de la page d'ajout, avec un formulaire avec tous les attributs à renseigner hormis les acteurs.



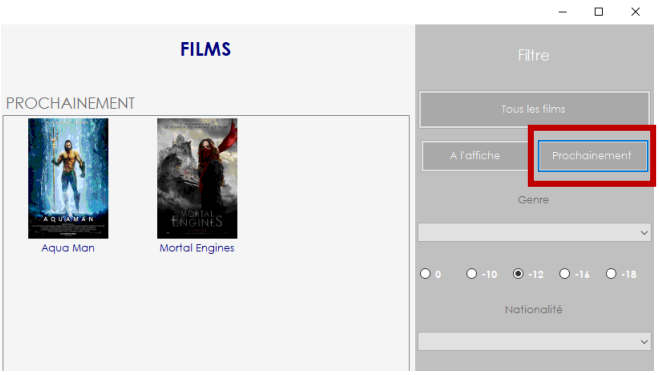
C- Modifier les films projetés dans le cinéma

Une grille de modification est ouverte quand l'on clique sur le bouton Modifier.



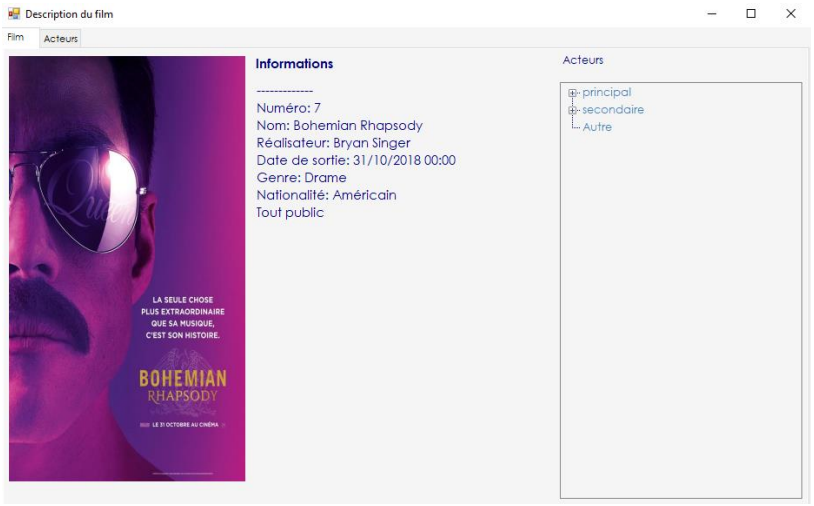
D- Consulter les films du cinéma

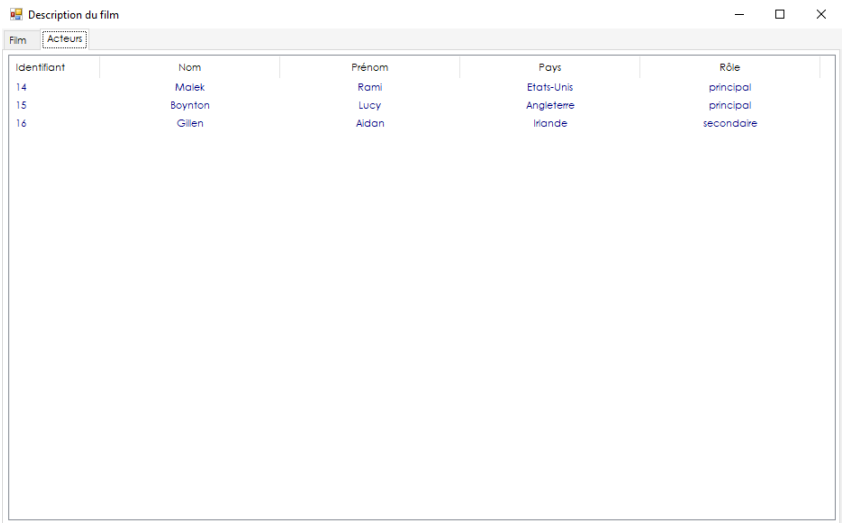
Plusieurs filtres s'offrent à l'utilisateur. Des films à l'affiche, prochainement en passant par les films par genre, tout est possible. Selon ces critères les films sont affichés en fonction.



E- Accéder aux informations d'un film

L'utilisateur peut ensuite double-cliquer sur l'un des films pour accéder aux informations le concernant. Une nouvelle fenêtre s'ouvre, avec un onglet sur le film en général et ses acteurs principaux et secondaires, puis un autre affichant un tableau des acteurs du film.





IV DESCRIPTION DES INTERFACES

A- ComboBox

La plupart des choix multiples ont été réalisés avec un ComboBox. Les types énumérés étaient récupérés simplement avec un Enum.GetNames, à l'inverse des villes des cinémas et de la nationalité des films. En effet, les récupérer implique la gestion des doublons. Deux méthodes ont été rajoutées, GetCinemaVille pour LesCinemas et GetListeNationalite pour LesFilms. Ces deux méthodes parcourent la liste d'élément et l'ajoute à la liste si ce dernier n'est pas déjà dedans. Cette liste sera ensuite stockée dans le ComboBox en question.

```
public List<string> GetListeVilles()
{
    List<string> listeville = new List<string>();

    foreach(Cinema c in this.ListeCinema)
    {
        if(listeville.Contains(c.Ville) == false)
        {
            listeville.Add(c.Ville);
        }
    }
    return listeville;
}
```

```
public List<string> GetListeNationalites()
{
    List<string> listenationalite = new List<string>();

    foreach(Film f in this.ListeFilm)
    {
        if(listenationalite.Contains(f.Nationalite) == false)
        {
            listenationalite.Add(f.Nationalite);
        }
    }
    return listenationalite;
}
```

```
private void InitListeVilles()
{
    for(int i=0; i< lesVilles.Count;i++)
    {
        ListeVilles.Items.Add(lesVilles[i]);
    }
}
```

```
public void InitListeNationalites()
{
    for(int i=0; i<lesNationalites.Count; i++)
    {
        ListeNationalite.Items.Add(lesNationalites[i]);
    }
}
```

Ces objets ont également été utilisés dans les fenêtres d'ajout et de modification des films.

B- ListBox

Le composant ListBox permet l'affichage et la sélection d'un cinéma. C'est dans cette liste que sont affichés les cinémas appartenant à une ville sélectionnée ou correspondant à un nom saisi. La méthode InitListeCinema permet de stocker les différents films récupérer selon les critères cités précédemment.

```
private void InitListCinemaVille()
{
    ListeCinema.Items.Clear();
    foreach(Cinema c in lesCinemasDeLaVille.ListeCinema)
    {
        ListeCinema.Items.Add(c.Nom);
    }
}
```

C- ListView

x *Avec image*

L'affichage des affiches de cinéma dans la page d'accueil se fait en associant une ListView et une ImageList.

Pour ce faire, il faut tout d'abord remplir l'ImageList des images des films devant être affichés. Ce travail est fait dans la méthode InitAffichesFilms.

```
private void InitAffichesFilms()
{
    ImageListFilm.ImageSize = new Size(100, 150);
    ImageListFilm.Images.Clear();

    for (int i=0; i<filmsProjetes.ListeFilm.Count; i++)
    {
        ImageListFilm.Images.Add(Image.FromFile(filmsProjetes.ListeFilm[i].Image));
    }
}
```

C'est ensuite dans la méthode InitListeFilms que l'on ajoute un à un les films. La méthode Add de ListView permet de renseigner deux paramètres : le texte et l'indice de l'image devant être affichée avec le texte. Chaque film a son affiche correspondante.

```
private void InitListeFilms()
{
    LVFilm.Clear();
    for (int i = 0; i < filmsProjetes.ListeFilm.Count; i++)
    {
        LVFilm.Items.Add(filmsProjetes.ListeFilm[i].Nom, i);

    }
    InitAffichesFilms();
}
```

x A colonnes

C'est dans la page Description du film que l'on a utilisé une ListView à colonnes. L'onglet 'Acteurs' permet d'observer en détails les informations des acteurs du film.

Il suffit d'initialiser les colonnes souhaitées et de définir le contenu de celles-ci. Une ligne se définit avec l'objet ListViewItem, comprenant un 'item' et des sous-items. Tous ces éléments forment une ligne. La ListViewItem est ensuite ajoutée à la ListView finale.

```
private void InitLVActeurs()
{
    LVActeurs.View = View.Details;

    LVActeurs.Columns.Add("Identifiant", 100, HorizontalAlignment.Center);
    LVActeurs.Columns.Add("Nom", 200, HorizontalAlignment.Center);
    LVActeurs.Columns.Add("Prénom", 200, HorizontalAlignment.Center);
    LVActeurs.Columns.Add("Pays", 200, HorizontalAlignment.Center);
    LVActeurs.Columns.Add("Rôle", 200, HorizontalAlignment.Center);

    foreach(Acteur a in this.film.ListeActeurs.ListeActeurs)
    {
        ListViewItem lvActeurs = new ListViewItem(a.Id.ToString());
        lvActeurs.SubItems.Add(a.Nom);
        lvActeurs.SubItems.Add(a.Prenom);
        lvActeurs.SubItems.Add(a.Pays);
        lvActeurs.SubItems.Add(a.RoleS);
        LVActeurs.Items.AddRange(new ListViewItem[] { lvActeurs });
    }
}
```

D- PictureBox

L'affiche de film dans la page Description de film a été faite avec un PictureBox. L'attribut StretchImage du composant permet d'adapter l'image à la taille de la case qui lui est réservée.

```
private void InitDescriptionFilm()
{
    ImgFilm.SizeMode = PictureBoxSizeMode.StretchImage;
    ImgFilm.Image = Image.FromFile(film.Image);
    LabelDescFilm.Text = film.ToString();

}
```

E- TreeView

Pour utiliser ce composant, une classe Acteur et LesActeurs (contenant une liste d'Acteur) ont été créées. Un attribut listeActeur a été ajouté à un Film. Le TreeView permet d'avoir trois nœuds principaux : acteur au rôle principal, acteur au rôle secondaire et autre. Sont ensuite stockés dedans les acteurs dont le rôle correspond.

La méthode GetActeurRole permet la récupération d'une liste d'acteur pour un rôle donné. C'est elle qui nous a permis de différencier le rôle de chaque acteur.

```
private void InitTreeView()
{
    foreach(RoleActeur r in Enum.GetValues(typeof(RoleActeur)))
    {
        TreeNode noeud = new TreeNode(r.ToString());
        LesActeurs listeA = film.ListeActeurs.GetActeurRole(r);

        foreach(Acteur a in listeA.ListeActeurs)
        {
            TreeNode na = new TreeNode(a.Nom + " " + a.Prenom);
            na.Name = a.Id.ToString();

            noeud.Nodes.Add(na);
        }

        TVActeurs.Nodes.Add(noeud);
    }
}
```

F- DataGridView

Comme en TP, le DataGridView a été implémenté comme outil de modification. Les différents films du cinéma ont été ajoutés à la grille, certains de ses attributs étant modifiables. Les colonnes correspondent à tous les attributs d'un film (hormis l'image et la liste de ses acteurs). Certains sont de type DataGridViewTextBoxColumn pour le texte, d'autres DataGridViewComboBoxColumn pour les listes déroulantes. Chaque ligne ('Rows') de la grille contient un film.

```
public void InitDGV()
{
    DataGridViewTextBoxColumn num = new DataGridViewTextBoxColumn();
    DataGridViewTextBoxColumn nom = new DataGridViewTextBoxColumn();
    DataGridViewTextBoxColumn realisateur = new DataGridViewTextBoxColumn();
    DataGridViewTextBoxColumn datesortie = new DataGridViewTextBoxColumn();
    DataGridViewTextBoxColumn limite = new DataGridViewTextBoxColumn();
    DataGridViewTextBoxColumn nationalite = new DataGridViewTextBoxColumn();

    num.HeaderText = "Numéro";
    num.Name = "Numero";
    num.ReadOnly = true;

    nom.HeaderText = "Nom";
    nom.Name = "Nom";
    nom.ReadOnly = false;
```

```
datesortie.HeaderText = "Date de Sortie";
datesortie.Name = "DateSortie";
datesortie.ReadOnly = true;

DataGridViewComboBoxColumn DGVCBCGenre = new DataGridViewComboBoxColumn();
DGVCBCGenre.HeaderText = "Genre";
DGVCBCGenre.Name = "Genre";
DGVCBCGenre.DataSource = Enum.GetNames(typeof(GenreFilm));

nationalite.HeaderText = "Nationalité";
nationalite.Name = "Nationalite";
nationalite.ReadOnly = true;

DataGridViewComboBoxColumn DGVCBCLimite = new DataGridViewComboBoxColumn();
DGVCBCLimite.HeaderText = "Limite d'âge";
DGVCBCLimite.Name = "Limite";
DGVCBCLimite.DataSource = Enum.GetNames(typeof(LimiteAge));

limite.HeaderText = "Limite d'âge";
limite.Name = "LimiteAge";
limite.ReadOnly = false;
```

```
        DGVFilms.Columns.AddRange(new DataGridViewColumn[] { num, nom, realisateur, datesortie, DGVCBCGenre, nationalite, DGVCBCLimite });

        foreach (Film f in this.cinema.ListeFilmsProjetes.ListeFilm)
        {
            DataGridViewRow dgvrFilms = new DataGridViewRow();
            DGVFilms.Rows.Add(new String[] { f.Numero.ToString(), f.Nom, f.Realisateur, f.DateSortie.ToString(), f.GenreS, f.Nationalite, f.LimiteS});
        }
    }
```

Chaque modification est prise en compte via l'événement CellEndEdit. L'attribut du film en question est modifié grâce à ses accesseurs.

```
private void DGVFilms_CellEndEdit(object sender, DataGridViewCellEventArgs e)
{
    int ligne = e.RowIndex;
    int col = e.ColumnIndex;

    DataGridViewCell cell = ((DataGridView) sender).Rows[ligne].Cells[col] ;

    Film f = this.cinema.ListeFilmsProjetes.ListeFilm[ligne];

    switch(col)
    {
        case 1:
            f.Nom = cell.Value.ToString();
        break;

        case 2:
            f.Realisateur = cell.Value.ToString();
        break;

        case 4:
            f.GenreS = cell.Value.ToString();
        break;

        case 6:
            f.LimiteS = cell.Value.ToString();
        break;
    }
}
```

G- OpenFileDialog

Ajouter un film nécessite la sélection d'une image dans nos répertoires. L'outil OpenFileDialog assure cette fonctionnalité. Le répertoire de départ est fixé sur le dossier 'Affiches' qui a été créé dans l'application, seuls des fichiers au format jpg sont admis. Si l'utilisateur ne sélectionne aucune affiche, une image par défaut est automatiquement ajoutée.

```
//-------------------------------------------------Image
if (this.cheminImage != null)
{
    ftemp.Image = this.cheminImage;
}
else
{
    ftemp.Image = ftemp.ImageParDefaut;
}
```

V ANALYSE DES FONCTIONNALITES

Toutes les fonctionnalités de la page principale sont des événements au clic ou à la sélection. A partir de ces événements, les composants qui y sont liés sont remplis. Des attributs dans l'application tel que CinemaCourant ou FilmProjetes stockent dynamiquement les choix de l'utilisateur.

Les méthodes de remplissage récupèrent ensuite ces variables et les manipulent en conséquence.

```
private void BParcourir_Click(object sender, EventArgs e)
{

    FenetreParcourir.Title = "Sélectionner une image";
    FenetreParcourir.InitialDirectory = System.Windows.Forms.Application.StartupPath + @"\Affiches\";
    FenetreParcourir.Filter = "Fichiers jpg (*.jpg)|*.jpg";


    if(FenetreParcourir.ShowDialog() == DialogResult.OK)
    {
        cheminImage = FenetreParcourir.FileName;
    }
    else
    {
        cheminImage = FenetreParcourir.InitialDirectory + "defaut.jpg";
    }

    FenetreParcourir.Dispose();
}
```

Par exemple, au clic sur le bouton à l'affiche, la variable FilmProjetes qui contient tous les films sera filtrée par les films qui sont déjà sortis grâce à la méthode GetFilmAffiche.

```
private void BAffiche_Click(object sender, EventArgs e)
{
    if (ListeCinema.SelectedItem != null)
    {
        filmsProjetes = cinemaCourant.ListeFilmsProjetes;
        filmsProjetes = filmsProjetes.GetFilmAffiche();
        InitListeFilms();
        LabelResultat.Text = "A l'affiche".ToUpper();
    }
}
```

```
private void InitListeFilms()
{
    LVFilm.Clear();
    for (int i = 0; i < filmsProjetes.ListeFilm.Count; i++)
    {
        LVFilm.Items.Add(filmsProjetes.ListeFilm[i].Nom, i);

    }
    InitAffichesFilms();
}
```

VI BILAN

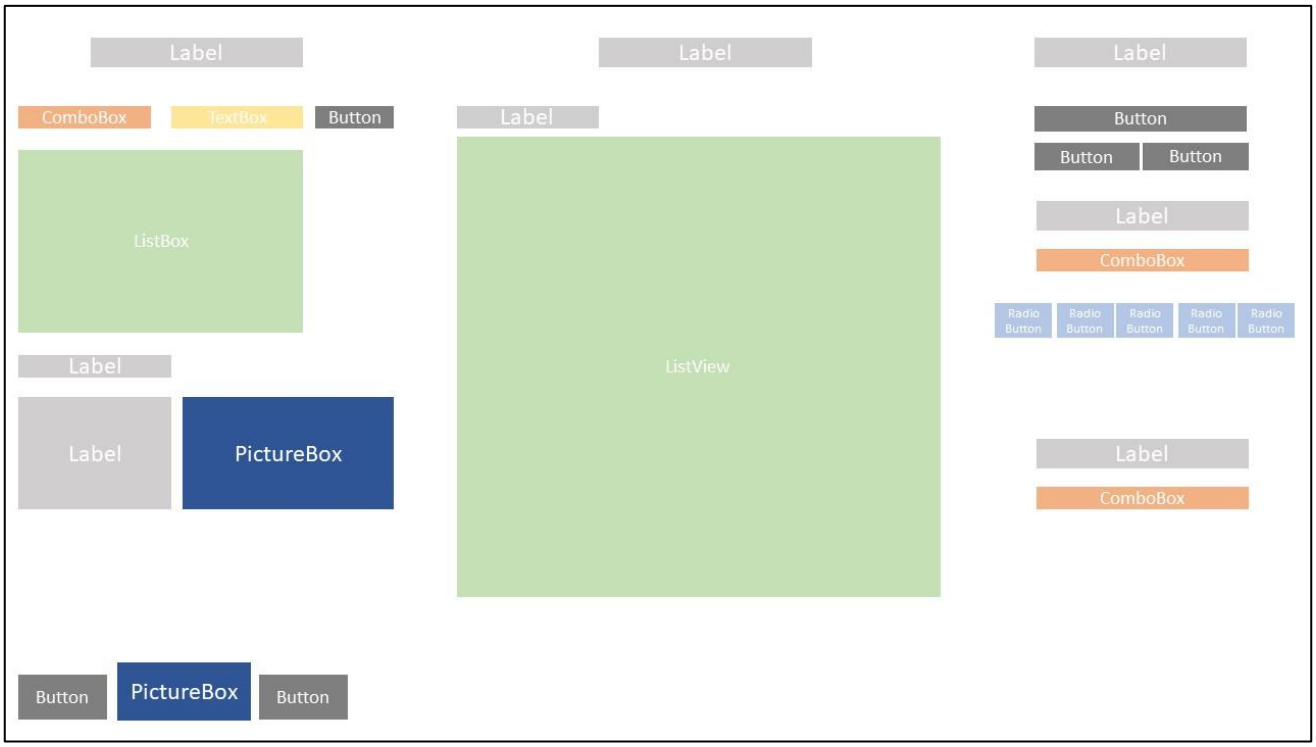
Le projet final présente une application fonctionnelle, facilement utilisable et qui répond à la demande.

Tout au long de ce projet, nous avons pu mettre en pratique les compétences et connaissances acquises pendant les cours de CDAA et en développer de nouvelles par nos propres recherches.

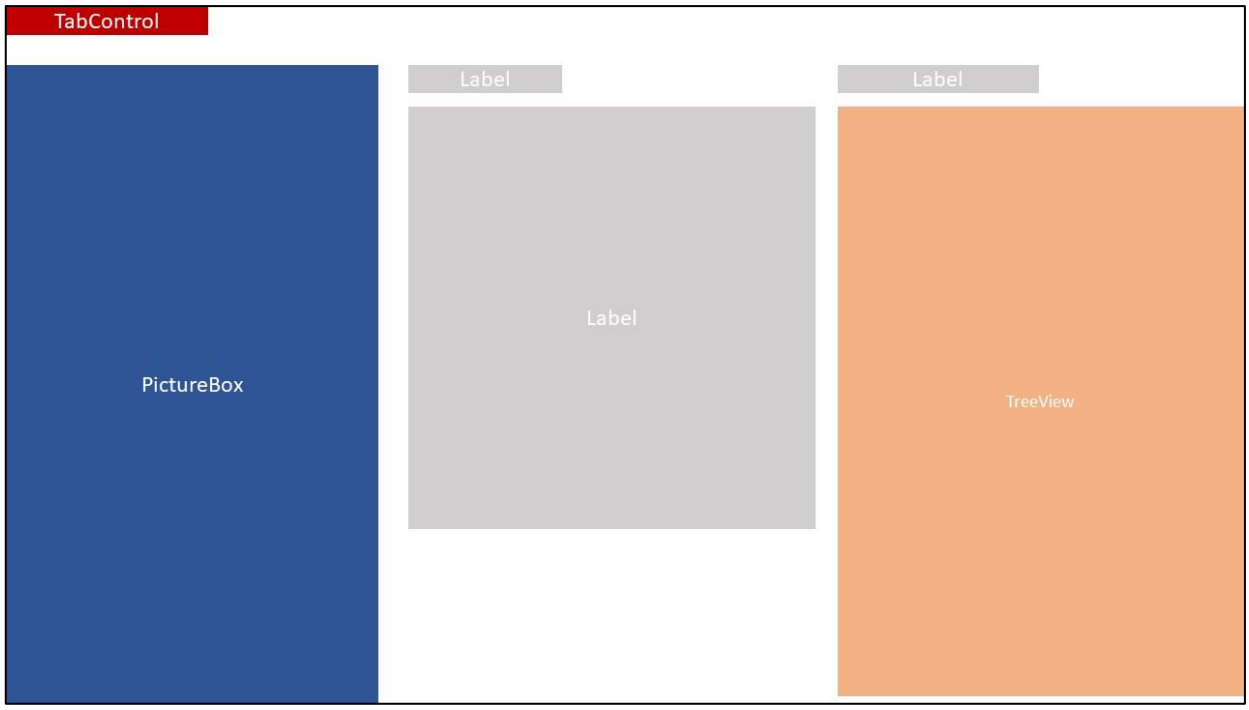
Nous avons pu nous rendre compte de l'intérêt des différentes composantes et fonctionnalités de C# et Windows Form, et ce qui peut nous être demandé en entreprise. Cela ajoute du concret au projet.

Avoir étudié et manipulé toutes ces composantes en un seul projet nous a apporté de l'expérience. Nous savons désormais ce qui doit être mis en œuvre pour produire une application professionnelle respectant la demande.

VII ANNEXE

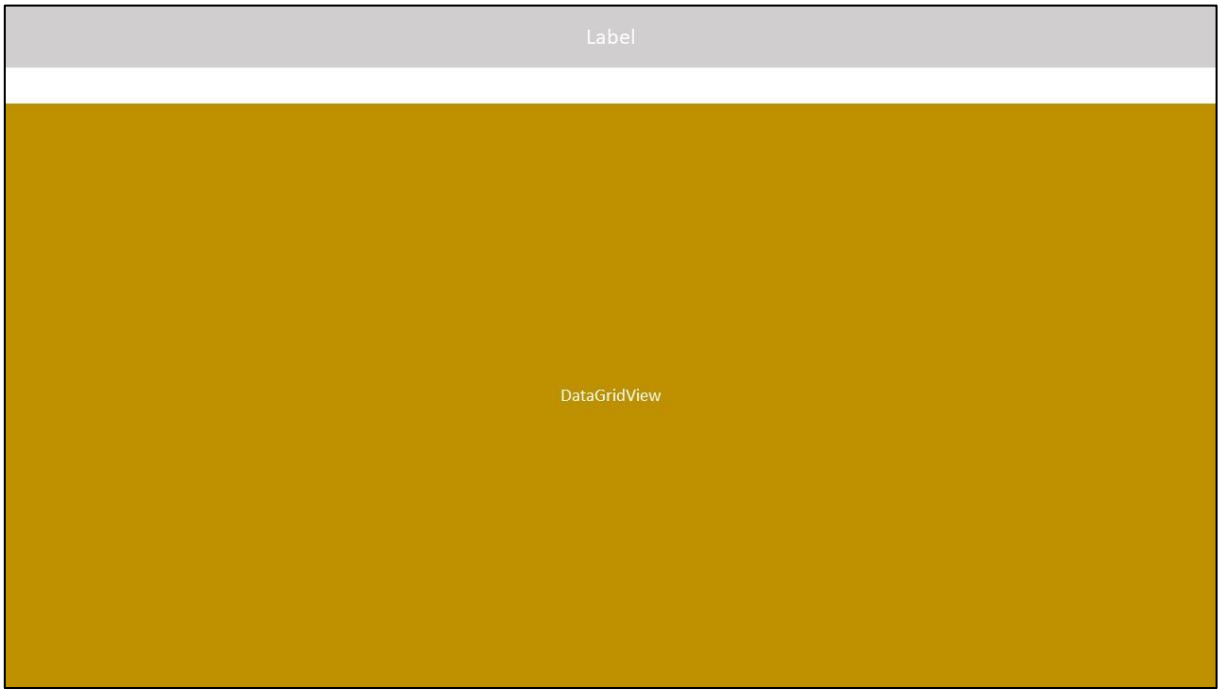


Structure page principale

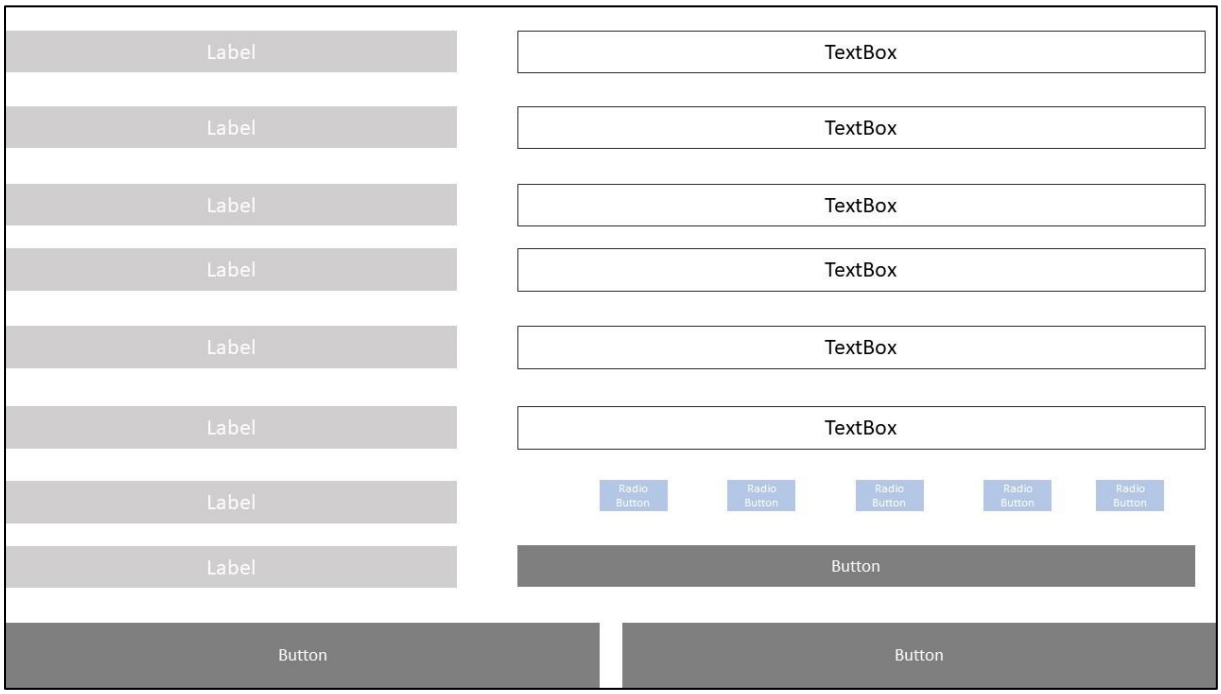


Structure page Description de film

Structure page de description des auteurs



Structure page de modification



Structure page d'ajout

VIII LEXICOGRAPHIE

https://stackoverflow.com/questions/15366294/listview-in-c-sharp-with-images

https://docs.microsoft.com/fr-fr/dotnet/api/system.windows.forms.listviewitem?view=netframework-4.7.2

https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.datagridviewcolumncollection.addrange?view=netframework-4.7.2

https://www.codeproject.com/Articles/32643/GMap-NET-Great-Maps-for-Windows-Forms-and-Presenta