Rapport de projet

Par Ophélie Cluzel

I. Description (cahier des charges)

Le but de ce projet est de proposer une interface graphique afin de représenter les différents équipements d'un joueur (Skin, Pioche et Planneur) et le profil de celui-ci. Pour cela on utilise Visual Studio 2019 qui nous permet de construire des interfaces à l'aide de composants/contrôles prédéfinis.

Ainsi cette interface va proposer différentes fonctionnalités telles que : la visualisation, l'ajout, la suppression et la modification d'équipements ainsi que la visualisation du profil du joueur avec la possibilité de choisir un avatar.

II. Analyse et diagramme de classes

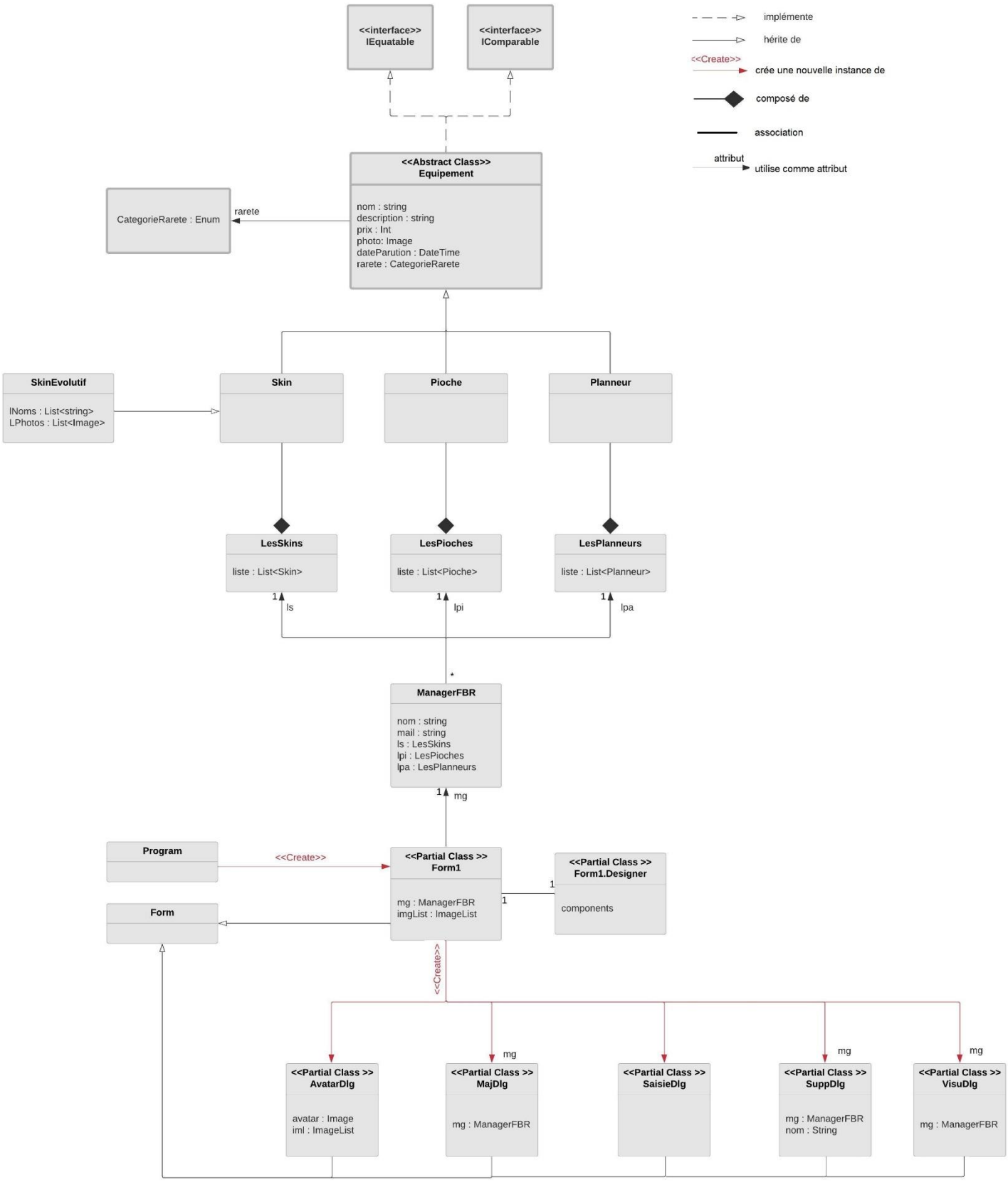
Analyse

La première classe est la classe **Equipement**, celle-ci va implémenter les interfaces **IEquatable** et **IComparable**. Les classes **Skin**, **Pioche** et **Planneur** héritent toutes trois de la classe **Equipement** qui est une classe abstraite. Puis les classes **LesSkins**, **LesPioches** et **LesPlanneurs** sont respectivement des compositions de **Skins**, **Pioches** et **Planneurs**. J'aurai dû faire une classe générique **LesEquipements<T>** car les classes **LesSkins**, **LesPioches** et les **Planneurs** sont en fait identiques. Pour finir la classe **ManagerFBR** utilise les classes **LesSkins**, **LesPioches** et **LesPlanneurs** comme attribut.

Intéressons-nous maintenant à la partie Windows Forms. La classe **Program.cs** va créer une instance de la classe **Form1** qui est en réalité une classe partielle divisée en deux parties : la partie **Form1.Designer.cs** qui va contenir les différents composants contenus dans le formulaire et la partie **Form1.cs** qui s'occupera de la partie évènement sur les différents contrôles. La classe **Form1.cs** dérive de la classe **Form** qui contient différentes méthodes propres aux formulaires qui vont nous être utiles comme : *Show()* et *Close()* pour ouvrir et fermer la fenêtre. La classe **Form1** va également utiliser la classe **ManagerFBR** comme attribut. C'est la classe **Form1** qui va créer les instances de toutes nos boites de dialogue : **AvatarDlg**, **MajDlg**, **SaisieDlg**, **SuppDlg** et **VisuDlg** car c'est un clic sur différents contrôles de **Form1** qui vont entrainer leur création. C'est donc la classe **Form1** qui va passer les paramètres utiles aux boites de dialogue notamment le paramètre de type **ManagerFBR**. Toutes ces boites de dialogue sont également des classes partielles divisées en une partie *.Designer.cs* et une partie *.cs* et dérivent de **Form** pour les mêmes raisons que **Form1**.

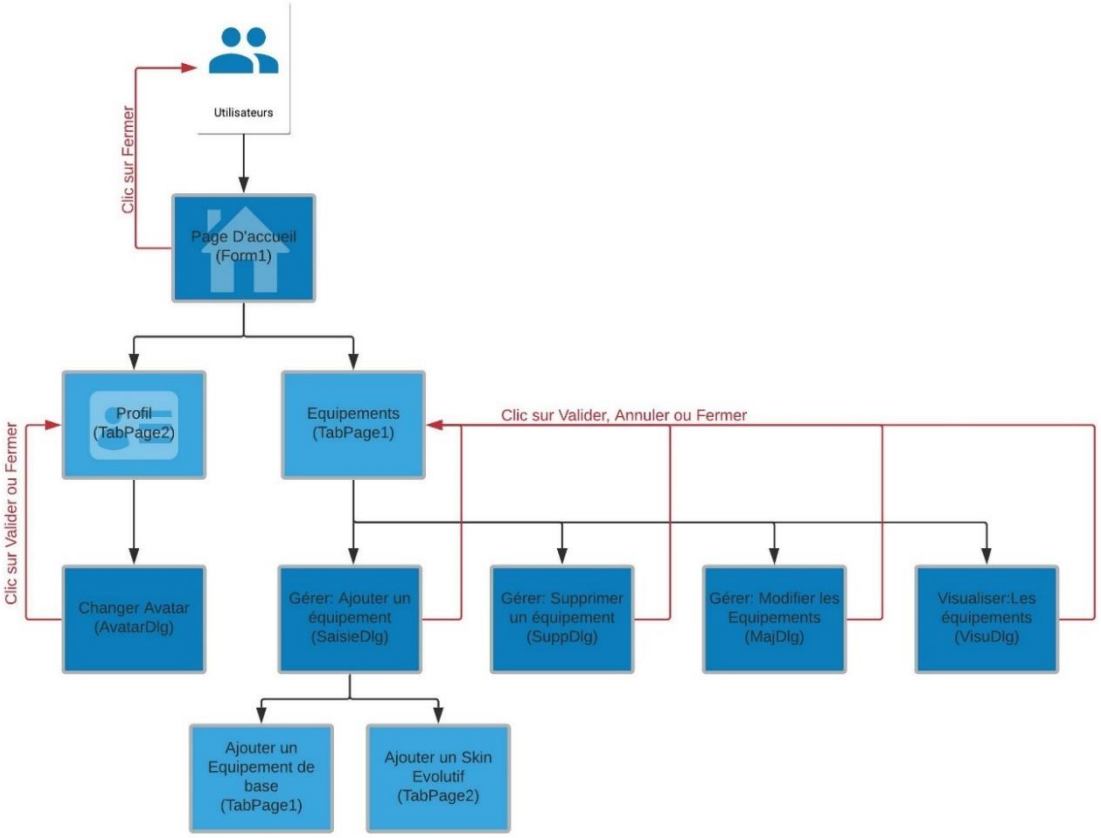
Je n'ai pas représenté les parties .Designer des classes partielles **AvatarDlg**, **MajDlg**,**SaisieDlg**,**SuppDlg** et **VisuDlg** pour ne pas surcharger le schéma. Je n'ai pas représenté non plus la classe **OpenFileDialog** dont une instance est créée par la classe **SaisieDlg** lors du clic sur le bouton Parcourir.

Diagramme de classes



III. Diagramme de navigation et description des interfaces

Diagramme de navigation

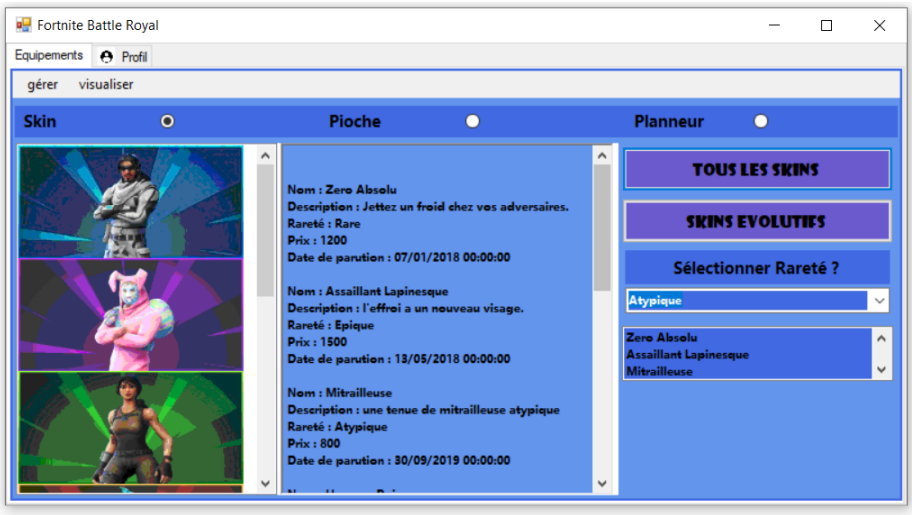


Description des interfaces

1. **Form1**

La page d'accueil est tout d'abord constituée d'un *TabControl* reliés à deux *TabPage* que je vais détailler ci-dessous.

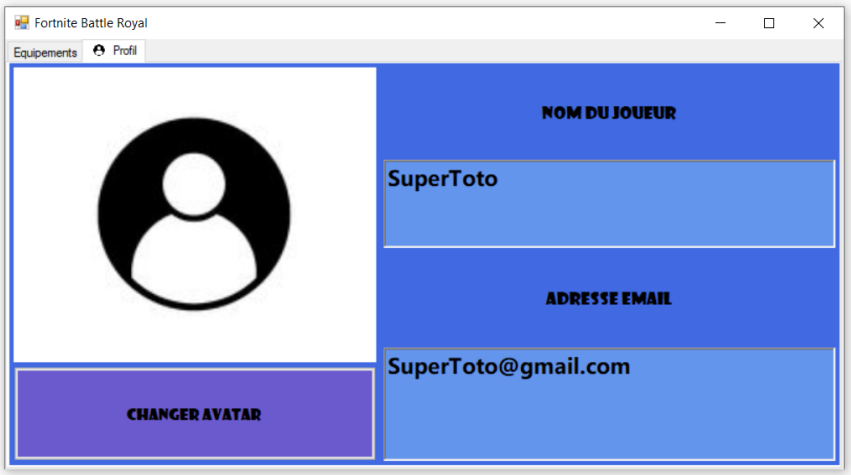
- TabPage1 : Equipements



Cette première page va contenir les informations relatives aux équipements.
Elle est constituée d'un *MenuStrip* composé de plusieurs *Items* sur lesquels on peut cliquer (ajouter, supprimer, modifier et visualiser les équipements) et qui seront en fait reliés à nos différentes boîtes de dialogue.

Elle contient ensuite 3 *RadioButton* qui nous permettrons de choisir pour quel type d'équipement nous voulons afficher les informations dans la fenêtre : Skin, Pioche ou Planneur. Ces *RadioButton* sont groupés c'est-à-dire que si l'un d'entre eux est coché, un autre ne peut être coché au même moment. Par défaut on définit le *RadioButton* Skin coché (Checked = true), ainsi quand on démarre l'application c'est les informations à propos des skins que l'on visualisera par défaut.
Puis elle contient une *ListBox* d'image qui va contenir la liste des images du type d'équipement sélectionné avec le *RadioButton* du joueur.
Pour finir elle contient une *RichTextBox* qui va contenir les informations des différents équipements, une *ComboBox* contenant les différentes raretés, une *ListBox* contenant la liste des noms des équipement et une *PictureBox* contenant la photo de l'équipement sélectionné dans la liste.
Elle contient également 2 *Button* lorsque le *RadioButton* Skin est coché car on peut choisir entre afficher tous les skins ou seulement les skins évolutifs et lorsque les *RadioButton* Pioche ou Planneur sont cochés, le 2^{ième} *Button* disparait (setVisible=false).

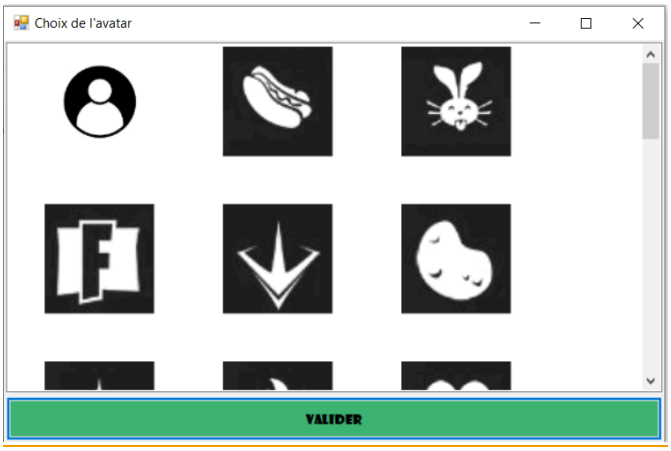
- TabPage2 : Profil



La deuxième Page va contenir les informations relatives au joueur : son nom et son email seront affichés dans deux *RichTextBox* et son avatar dans une *PictureBox*.
Il y a également un *Button* qui permet de changer d'avatar qui fera appel à la boite de dialogue AvatarDlg. L'avatar choisit sera affiché dans la *PictureBox* mais sera aussi affiché comme Icone de la *TabPage*

.

2. Changer d'avatar : AvatarDlg

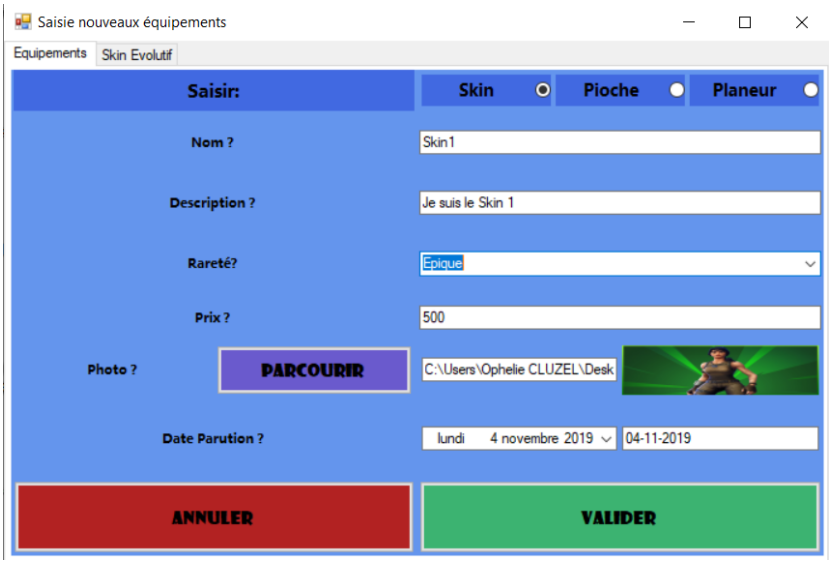


Cette page va permettre de sélectionner un avatar pour le joueur. Pour cela elle dispose d'une *ListView* multi-colonnes d'images.
Pour quitter la page et revenir au Form1 on peut soit cliquer sur Fermer (la croix) soit cliquer sur le *Button* Valider (dans ce cas l'image choisie sera définie en avatar du joueur).

3. Gérer, Ajouter : SaisieDlg

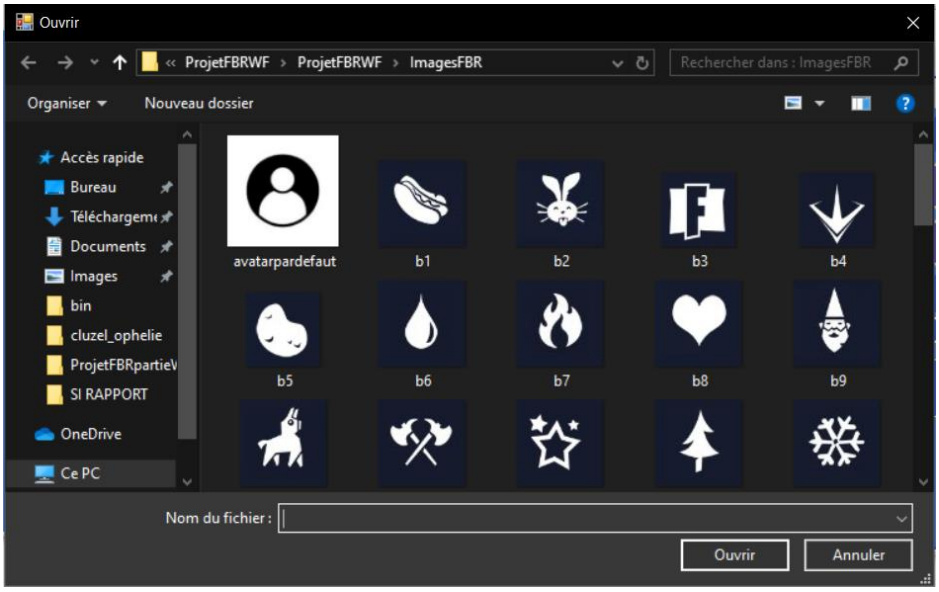
Cette page est elle aussi constituée d'un TabControl relié à deux TabPage

➢ TabPage1 : Equipements de base

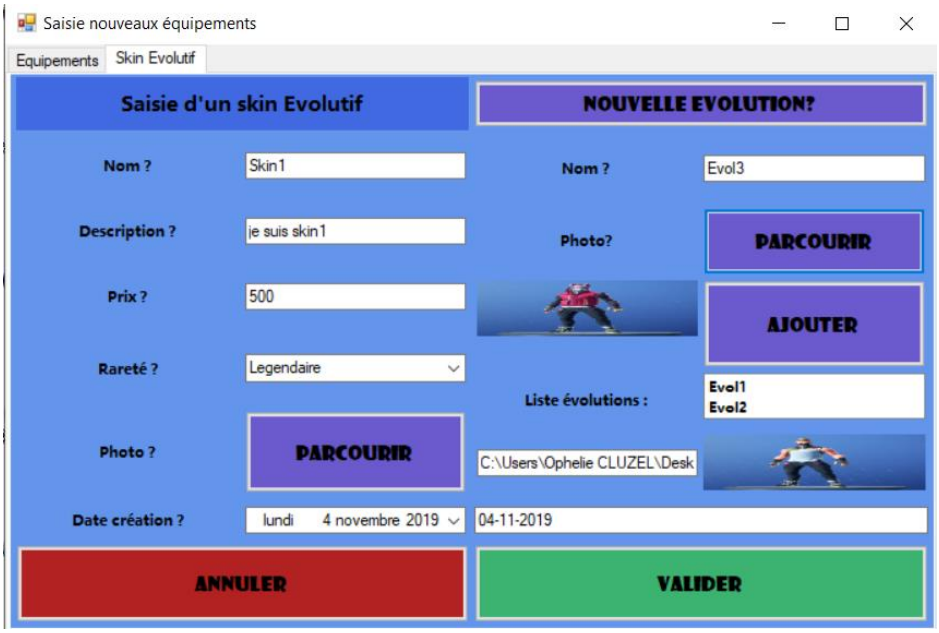


Cette première page va contenir un formulaire d'ajout d'un équipement à la collection du Joueur.
On retrouve ici encore nos 3 *RadioButton* qui vont nous permettre de spécifier quel type d'équipement nous souhaitons ajouter. Puis on a 3 *TextBox* dans lesquelles l'utilisateur pourra saisir le nom, la description et le prix de l'équipement. Il y a également une *ComboBox* permettant de sélectionner la rareté, un *Button* Parcourir qui permettra de choisir une photo et une fois celle-ci choisie elle sera affichée dans une *PictureBox* et le nom du fichier dans une *TextBox* et enfin pour finir un *DateTimePicker* qui nous permettra de sélectionner la date de parution dans un calendrier et une fois celle-ci choisie elle sera affichée dans une *TextBox*.

Le clic sur le bouton parcourir ouvre une fenêtre *OpenFileDialog* :



- TabPage2 : SkinEvolutif



Cette première page va contenir un formulaire d'ajout d'un skin évolutif à la collection du Joueur. En plus des informations basiques à remplir pour un équipement, celui-ci va nous permettre d'ajouter la liste des évolutions du Skin : pour chaque évolution on peut saisir son nom dans une *TextBox*, sa photo grâce au *Button* Parcourir qui sera ensuite affichée dans une *PictureBox* et enfin une *ListBox* qui contiendra la liste des évolutions déjà ajoutées. Le *Button* Ajouter permet d'ajouter l'évolution à la liste et le *Button* Nouvelle Evolution permet de saisir une nouvelle évolution.

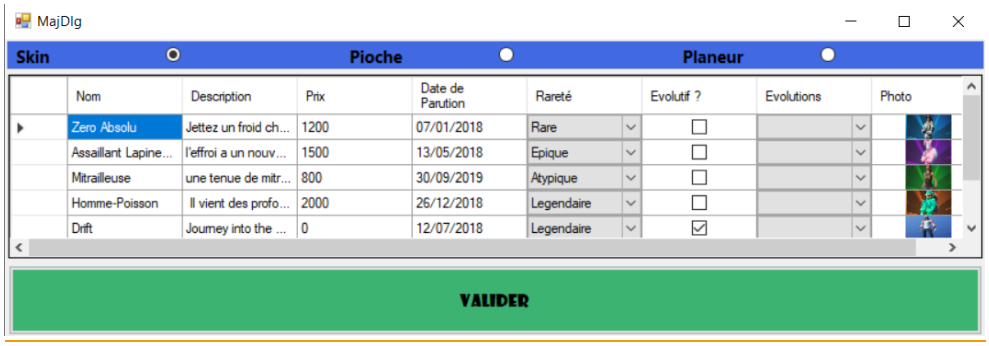
Pour quitter la page et revenir au Form1 on peut soit cliquer sur Fermer (la croix) soit cliquer sur le *Button* Annuler (dans ce cas aucun équipement ne sera ajouté) soit cliquer sur le *Button* Valider (dans ce cas l'équipement saisi sera ajouté).

4. Gérer, Supprimer : SuppDlg

Cette page va permettre de supprimer un équipement en fonction de son Nom.
Elle contient donc encore une fois 3 *RadioButton* Skin, Pioche et Planneur permettant de sélectionner le type d'équipement à supprimer. Elle contient ensuite une *RichTextBox* pour saisir le nom de l'équipement à supprimer et un Button Rechercher qui va afficher l'équipement au nom correspondant : il va afficher sa photo dans une *PictureBox* et ses informations dans une *RichTextBox*. Si l'équipement n'est pas trouvé il affiche un message dans la *RichTextBox*.

Pour quitter la page et revenir au Form1 on peut soit cliquer sur Fermer (la croix) soit cliquer sur le *Button* Annuler (dans ce cas aucun équipement ne sera supprimé) soit cliquer sur le *Button* Valider (dans ce cas l'équipement choisi sera supprimé).

5. Gérer, Modifier : MajDlg



Cette page va permettre de modifier certaines informations à propos des équipements.

Elle contient donc encore une fois 3 *RadioButton* Skin, Pioche et Planneur permettant de sélectionner le type d'équipement à modifier. Elle contient également un *DataGridView* qui sera une sorte de tableau avec dans chaque colonne les différentes caractéristiques liées aux équipements.

Pour quitter la page et revenir au Form1 on peut soit cliquer sur Fermer (la croix) soit cliquer sur le *Button* Valider (dans ce cas les équipements seront modifiés).

6. Visualiser les Equipements : VisuDlg

Cette page va permettre de visualiser les équipements sous forme d'une arborescence.

Elle contient donc encore une fois 3 *RadioButton* Skin, Pioche et Planneur permettant de sélectionner le type d'équipement à visualiser. Les équipements seront triés par rareté dans un *TreeView*. Si on sélectionne un équipement dans le *TreeView*, sa photo sera affichée dans une *PictureBox* et ses informations dans une *RichTextBox*.

Pour quitter la page et revenir au Form1 on clique sur Fermer (la croix).

IV. Analyse des Fonctionnalités

La première fonctionnalité disponible sur la page d'accueil de l'application est de pouvoir basculer entre les deux *TabPage* intitulées Equipements et Profil. Il suffit à l'utilisateur de cliquer sur la page qu'il veut afficher. Dans le code ce changement se fait automatiquement avec l'utilisation du composant *TabControl* nous n'avons pas besoin de coder l'évènement.

Dans la *TabPage* intitulée Equipements, l'utilisateur peut :

- Choisir le type d'équipement à afficher en cochant un des 3 RadioButton Skin, Pioche ou Planneur, cela va déclencher un évènement :

```
private void RBSkin_CheckedChanged(object sender, EventArgs e)
private void RBPioche_CheckedChanged(object sender, EventArgs e)
private void RBPlanneur_CheckedChanged(object sender, EventArgs e)
```

  Cela va remplir la liste d'image avec les images du type d'équipement sélectionné et la liste des équipements avec les noms des équipements du type sélectionné.
  Lorsque le bouton planneur ou pioche est coché, le bouton SkinEvolutif disparait (*SetVisible=false*).

- Visualiser la liste des images associées aux équipements et utiliser la barre de défilement.
- Cliquer sur le bouton « Tous les Skins », cela déclenche un évènement :

```
private void BTSkin_Click(object sender, EventArgs e)
```

  Cela va remplir la liste d'image avec les images de tous les skins et la zone de texte avec les informations sur tous les skins.
- Cliquer sur le bouton « Skins évolutifs », cela déclenche un évènement :

```
private void BESkin_Click(object sender, EventArgs e)
```

  Cela va remplir la liste d'image avec les images des skins évolutifs et de leurs évolutions et la zone de texte avec les informations sur les skins évolutifs
- Choisir une rareté dans la liste déroulante de rareté, cela déclenche un évènement :

```
private void ListeRarete_SelectedIndexChanged(object sender, EventArgs e)
```

  Cela va remplir la zone de texte avec les équipements de la rareté sélectionnée.
- Choisir un Equipement dans la liste d'équipements, cela déclenche un évènement :

```
private void ListeEquipements_SelectedIndexChanged(object sender, EventArgs e)
```

  Cela va afficher les informations de l'équipement dans la zone de texte et sa photo dans la *PictureBox*.
- choisir différentes options dans un menu déroulant (*MenuStrip*), le clic sur les différents éléments de ce menu va générer un évènement. Nous allons donc coder ce qu'il se passe lors du clic sur les différents éléments :

  ➢

```
private void ajouterToolStripMenuItem_Click(object sender, EventArgs e)
```

  Voyons ce qu'il se passe lors du clic sur l'option « Ajouter » du menu déroulant « gérer ». Tout d'abord une nouvelle fenêtre pour saisir un équipement va s'ouvrir (*SaisieDlg*). L'utilisateur pourra alors osciller entre deux *TabPage* intitulées Equipement et SkinEvolutif.
  Dans la *TabPage* Equipement il va pouvoir cocher un des 3 *RadioButton* pour choisir le type d'équipement à saisir. Puis il va pouvoir remplir les zones de textes pour spécifier le nom, la description et le prix de l'équipement. Il va pouvoir également sélectionner une rareté dans la liste déroulante, une date dans le calendrier et une photo en cliquant sur le bouton parcourir : il va ouvrir une nouvelle fenêtre (*OpenFileDlg*) qui va ouvrir l'explorateur de fichier qui va nous permettre d'importer une image stockée sur notre ordinateur, l'image choisie et le nom du fichier son affichés dans la fenêtre *SaisieDlg*.
  Une fois toutes les informations saisies on peut soit cliquer sur le bouton Annuler, dans ce cas l'équipement ne sera pas ajouté soit cliquer sur valider et l'équipement sera ajouté à la collection du joueur. Dans les deux cas la fenêtre de saisie se ferme et on retourne à la *tabPage* Equipement du *Form1*.
  Dans la TabPage Skin Evolutif c'est le même principe sauf que l'utilisateur va pouvoir saisir plusieurs évolutions pour un skin. Pour cela il va pouvoir remplir la zone de texte pour le nom de l'évolution et cliquer sur le bouton parcourir pour ajouter une image (cela ouvre une nouvelle fenêtre *OpenFileDlg*). Une fois cela fini il peut cliquer sur ajouter et son évolution sera ajoutée dans la liste des évolutions. Pour saisir une nouvelle évolution il peut cliquer sur le bouton Nouvelle Evolution qui va remettre tous les champs à zéro (sauf la liste des évolutions qu'il garde en mémoire).

➢ `private void supprimerToolStripMenuItem_Click(object sender, EventArgs e)`

Voyons ce qu'il se passe lors du clic sur l'option « Supprimer » du menu déroulant « gérer ». Tout d'abord une nouvelle fenêtre pour saisir le nom de l'équipement à supprimer va s'ouvrir (SuppDlg). L'utilisateur va pouvoir cocher un des 3 *RadioButton* pour choisir le type d'équipement à supprimer. Ensuite il va pouvoir écrire le nom de l'équipement qu'il veut supprimer dans une zone de texte, une fois cela fait il peut cliquer sur le bouton Rechercher qui va afficher l'image et les informations de l'équipement correspondant et un message d'erreur si aucun équipement à ce nom n'est trouvé. Enfin il peut soit cliquer sur le bouton Annuler, dans ce cas aucun équipement ne sera supprimé, soit cliquer sur Valider dans ce cas l'équipement sera supprimé de la collection du joueur. Dans les deux cas la fenêtre se ferme.

➢ `private void modifierToolStripMenuItem_Click(object sender, EventArgs e)`

Voyons ce qu'il se passe lors du clic sur l'option « Modifier » du menu déroulant « gérer ». Tout d'abord une nouvelle fenêtre pour modifier les équipements va s'ouvrir (*MajDlg*). L'utilisateur va pouvoir cocher un des 3 *RadioButton* pour choisir le type d'équipement à modifier. Il va pouvoir modifier uniquement les attributs prix et rareté des équipements (*ReadOnly=false*). Pour cela il clique sur la cellule du tableau qu'il veut modifier et entre une nouvelle valeur de prix, pour la rareté il sélectionne une nouvelle valeur dans la liste déroulante.
Une fois cela fini il peut cliquer sur le bouton Valider pour que les modifications sur les équipements soient prises en compte et la fenêtre se ferme.

➢ `private void lesEquipementsToolStripMenuItem_Click(object sender, EventArgs e)`

Voyons ce qu'il se passe lors du clic sur l'option « LesEquipements » du menu déroulant « visualiser ». Tout d'abord une nouvelle fenêtre pour visualiser les équipements va s'ouvrir (*VisuDlg*). L'utilisateur va pouvoir cocher un des 3 *RadioButton* pour choisir le type d'équipement à visualiser. Cela va remplir l'arborescence avec le type d'élément choisit. L'utilisateur peut également sélectionner un élément dans l'arbre : lorsqu'il clique dessus cela va afficher ses informations dans une zone de texte et son image dans une *PictureBox*. Pour fermer la fenêtre et revenir au Form1 il peut cliquer sur la croix.

Dans la TabPage intitulée Profil l'utilisateur peut :

- Visualiser son nom, son email et son avatar.
- Cliquer sur le bouton Changer D'avatar, cela déclenche un événènement :
  `private void button1_Click(object sender, EventArgs e)`

  Cela va ouvrir une nouvelle fenêtre (*AvatarDlg*) dans lequel il va pouvoir sélectionner une image dans une liste (*ListView*) et cliquer sur un bouton Valider. L'image qu'il aura choisit va être récupérée et va être affichée dans la *PictureBox* de la *TabPage* Profil et également en icone du *TabControl*.

V. Dossier de Programmation

J'ai décidé d'expliquer plus en détail la partie d'ajout d'un équipement notamment la partie récupération du nouvel équipement dans Form1. Pour cela je vais expliquer les méthodes `ajouterToolStripMenuItem_Click de Form1 et BValider_Click et BAnnuler_Click de SaisieDlg.`

```
private void ajouterToolStripMenuItem_Click(object sender, EventArgs e)
{
    SaisieDlg ssd = new SaisieDlg();
    DialogResult res = ssd.ShowDialog();
    if (res == DialogResult.OK)
    {
        if (ssd.Coché == "Skin")
        {
            mg.Ls.Ajoute(ssd.LeSkin);
            InitListeEquipements();
            InitListePhotos();
        }
        else
        { if (ssd.Coché == "Pioche")
            {
                mg.Lpi.Ajoute(ssd.LaPioche);
                InitListeEquipements();
                InitListePhotos();
            }

            else
            {
                mg.Lpa.Ajoute(ssd.LePlanneur);
                InitListeEquipements();
                InitListePhotos();
            }
        }
    }
}
```

Cette méthode se situe dans *Form1*, elle réagit à l'évènement correspondant au fait que l'option « ajouter » a été sélectionnée dans le menu déroulant. On commence donc par créer une nouvelle instance de *SaisieDlg*. Puis on va afficher cette nouvelle fenêtre en récupérant au passage la valeur de sa Propriété *DialogResult* qui va nous indiquer comment l'utilisateur a quitté la fenêtre (clic sur quel bouton).

Si la valeur de ce *DialogResult* est égale à OK (clic sur le bouton Valider de la fenêtre *SaisieDlg*) on va pouvoir ajouter un nouvel équipement à la collection du joueur cependant pour cela nous avons besoin de connaître le type du nouvel équipement à ajouter. Pour cela on récupère la valeur de la propriété *Coché* de *SaisieDlg* qui nous indique le type d'équipement qui a été saisi. Si cette propriété vaut « Skin », on va récupérer la propriété *LeSkin* de l'instance de *SaisieDlg* qui contient le Skin qui a été saisi et on va l'ajouter à la collection de Skin en faisant appel à la fonction *Ajoute*() de la classe *LesSkins*. Pour on va faire appel aux méthodes *InitListeEquipements*() et *InitListePhotos*() de la classe Form1 qui vont remettre à jour la *ListBox* de photos des skins et la *ListBox* de noms des Skins pour y afficher le nouveau skin venant d'être ajouté. Ensuite c'est le même principe pour Pioche et Planneur. Si le *dialogResult* n'est pas égal à OK, on ne fait rien car aucun équipement n'a été ajouté (clic sur le bouton Annuler ou sur la croix de la fenêtre *SaisieDlg*).

```
private void BValider_Click(object sender, EventArgs e)
{   if (RBSkin.Checked == true)
    {
        sk.Photo = ph;
        sk.DateParution = dp;
        coché = "Skin";
    }
    else
    { if (RBPioche.Checked == true)
        {
            pi.Photo = ph;
            pi.DateParution = dp;
            coché = "Pioche";
        }
        else
        { pla.Photo = ph;
          pla.DateParution = dp;
            coché = "Planneur";
        }
     }
    this.DialogResult = DialogResult.OK;
    this.Close();

}
```

Cette méthode se situe dans la classe *SaisieDlg*. Elle correspond à l'évènement déclenché lors du clic sur le bouton Valider. Tout d'abord intéressons nous aux attributs de la classe *SaisieDlg*, en voici quelques-uns:

```
public partial class SaisieDlg : Form
{
    private Skin sk;
    private Pioche pi;
    private Planneur pla;
    private Image ph;
    private DateTime dp;
    private String coché;
```

Ces attributs seront remplis au fur et à mesure avec les différentes valeurs saisies et sélectionnées par l'utilisateur dans notre fenêtre.

Une fois cela fait il cliquera sur le bouton *Valider*. Si à ce moment là le *RadioButton RBSkin* est coché cela signifie que l'utilisateur vient de saisir un nouveau skin. On affecte donc à notre attribut *Sk* de type Skin l'image contenue dans l'attribut *ph* qui contient la photo ayant été sélectionnée par l'utilisateur ultérieurement et également la date contenue dans l'attribut *dp* qui contient la date sélectionnée par l'utilisateur. On définit l'attribut *coché* sur « Skin » pour que *Form1* sache qu'on lui envoie un objet de type Skin. Ensuite c'est le même principe pour pioche et planneur.

A la fin on définit le *DialogResult* sur OK pour indiquer au *Form1* que l'utilisateur a quitté la fenêtre en validant, et que donc il y a un équipement à ajouter. Ensuite on ferme la fenêtre.

```
private void BAnnuler_Click(object sender, EventArgs e)
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
}
```

Cette méthode se trouve dans la classe *SaisieDlg*. Elle correspond à l'évènement déclenché lors du clic sur le bouton Annuler. Ici on définit le *DialogResult* sur *Cancel* pour indiquer au *Form1* que l'utilisateur a quitté la fenêtre en annulant et que par conséquent il n'y a pas d'équipement à ajouter. Puis on ferme la fenêtre.

Je vais également expliquer plus en détail la partie *TabPage2* intitulée *Profil* notamment les méthodes *button1_Click*() et *InitIcone*() ainsi que les méthodes *Bvalider_Click*() et *InitListeAvatar*() de la classe *AvatarDlg*.

```
private void button1_Click(object sender, EventArgs e)
{
    AvatarDlg ad = new AvatarDlg();
    DialogResult res = ad.ShowDialog();
    if (res == DialogResult.OK)
    {
        PBAvatar.Image = ad.Avatar;
        InitIcone();
    }
}
```

Cette méthode se situe dans *Form1*. Elle correspond à l'évènement déclenché lors du clic sur le bouton *button1* intitulé « Changer d'avatar ». Tout d'abord on commence par créer une nouvelle instance de la classe *AvatarDlg*. Puis on affiche la fenêtre tout en récupérant la valeur de sa propriété *DialogResult*. Si cette valeur est définie à OK, cela signifie que l'utilisateur a sélectionné une image et qu'il veut donc changer son avatar. On va donc affecter à la propriété *Image* de la *PictureBox PBAvatar* l'image sélectionnée. Cette image se situe dans la propriété *Avatar* de la classe *AvatarDlg*. On fait également appel à la méthode *InitIcone*() de la classe *Form1* qui va mettre à jour l'icône de la *TabPage2* en utilisant l'image sélectionnée.

```
public void InitIcone() {
    ImageList imageList1 = new ImageList();
    imageList1.Images.Add("key1", PBAvatar.Image);
    tabControl1.ImageList = imageList1;
    tabPage2.ImageKey = "key1";
}
```

Cette méthode se situe dans *Form1*. Elle va nous permettre d'initialiser l'icône de la *TabPage2*. Pour cela nous devons créer une nouvelle instance de liste d'images *ImageList* qui contiendra toutes les icônes pour nos différentes pages (ici la liste n'aura qu'un seul élément comme on veut ne mettre une icône qu'à une seule page). On ajoute l'image que l'on veut comme icône dans notre *ImageList* : on récupère l'image de la *PictureBox PBAvatar* et on fixe un identifiant à l'image, ici ce sera key1. Ensuite on doit affecter notre *ImageList* à la propriété *ImageList* de notre *TabControl*. Et enfin on affecte l'icône à notre *tabPage*2 en affectant à la propriété *ImageKey* l'identifiant de notre image.

```
public void InitListeAvatar()
{
   //Ajout d'images des ressources dans l'attribut liste d'image
    iml.Images.Add(Properties.Resources.avatarpardefaut);
    iml.Images.Add(Properties.Resources.b1);
    iml.Images.Add(Properties.Resources.b2);
    iml.Images.Add(Properties.Resources.b3);
    iml.Images.Add(Properties.Resources.b4);
    iml.Images.Add(Properties.Resources.b5);
    iml.Images.Add(Properties.Resources.b6);
    iml.Images.Add(Properties.Resources.b7);
    iml.Images.Add(Properties.Resources.b8);
    iml.Images.Add(Properties.Resources.b9);
    iml.Images.Add(Properties.Resources.b10);
    iml.Images.Add(Properties.Resources.b11);
    iml.Images.Add(Properties.Resources.b12);
    iml.Images.Add(Properties.Resources.b13);
    iml.Images.Add(Properties.Resources.b14);
    iml.Images.Add(Properties.Resources.b15);
    iml.Images.Add(Properties.Resources.b16);
    iml.Images.Add(Properties.Resources.b17);
    iml.Images.Add(Properties.Resources.b18);
    iml.Images.Add(Properties.Resources.b19);
    iml.Images.Add(Properties.Resources.b20);
    iml.Images.Add(Properties.Resources.b21);
    iml.Images.Add(Properties.Resources.b21);
    iml.Images.Add(Properties.Resources.b22);
    iml.Images.Add(Properties.Resources.b23);
    iml.Images.Add(Properties.Resources.b24);
    iml.Images.Add(Properties.Resources.b25);
    //initialisation de la taille des images dans la liste d'image
    iml.ImageSize = new Size(100,100);

    //ajout de la liste d'image dans la ListView1
    listView1.LargeImageList = iml;

    //ajout d'un libellé à chaque image dans la listeView1
    for (int i=0;i<25;i++)
    {
        listView1.Items.Add(new ListViewItem("", i));

    }
}
```

Cette méthode se trouve dans la classe *AvatarDlg*. Elle va permettre d'initialiser le composant *ListView*. Pour cela on commence par ajouter dans notre ImageList iml (attribut de la classe) des images que l'on a placé dans les ressources de l'application. Ensuite on définit la taille que l'on veut pour nos images. Ensuite on ajoute notre liste d'images à notre composant *ListView* par la propriété *LargeImageList*. Et enfin on ajoute un libellé et un index à chaque image car une image dans une *listView* doit avoir un libellé et un index.

```
private void Bvalider_Click(object sender, EventArgs e)
{
    this.avatar =iml.Images[listView1.FocusedItem.Index];
    this.DialogResult = DialogResult.OK;
    this.Close();
}
```

Cette méthode se trouve dans *AvatarDlg*. Elle correspond à l'évènement déclenché lors du clic sur le bouton Valider. On commence par récupérer l'image choisie dans la *listView*. Pour cela on récupère l'index sélectionné dans la *listView* et l'image correspondante dans *l'ImageList*. Ensuite on définit le *DialogResult* à OK et on ferme la fenêtre.

VI. Bilan

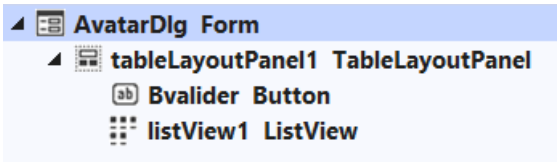
J'ai essayé de rendre l'application plus sympathique en explorant les différentes options offertes par les composants (changement de la font, de la couleur de fond, du curseur etc).

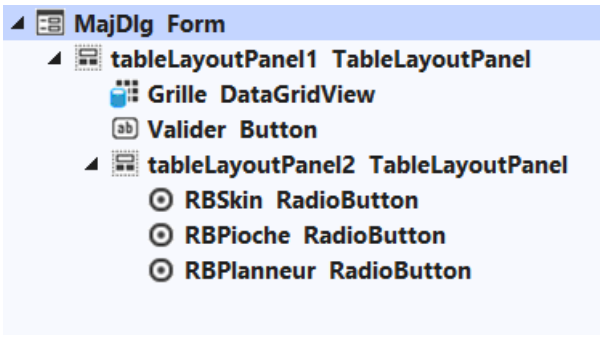
Je pense que mon programme aurait pu être amélioré en créant une classe générique LesEquipements<T> qui aurait regroupé les classes LesSkins, LesPioches et LesPlanneurs.

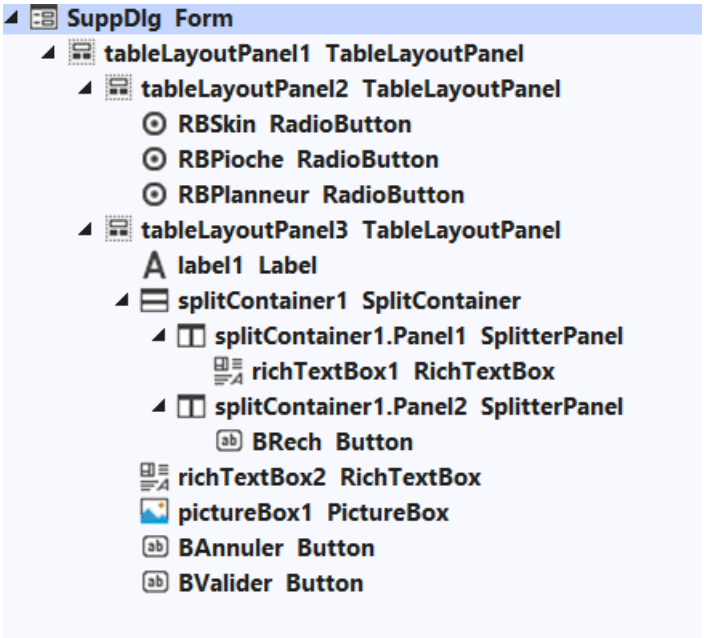
Mon interface aussi aurait pu être améliorée pour notamment éviter la répétition des *RadioButton* dans chaque fenêtre et dans le code également.

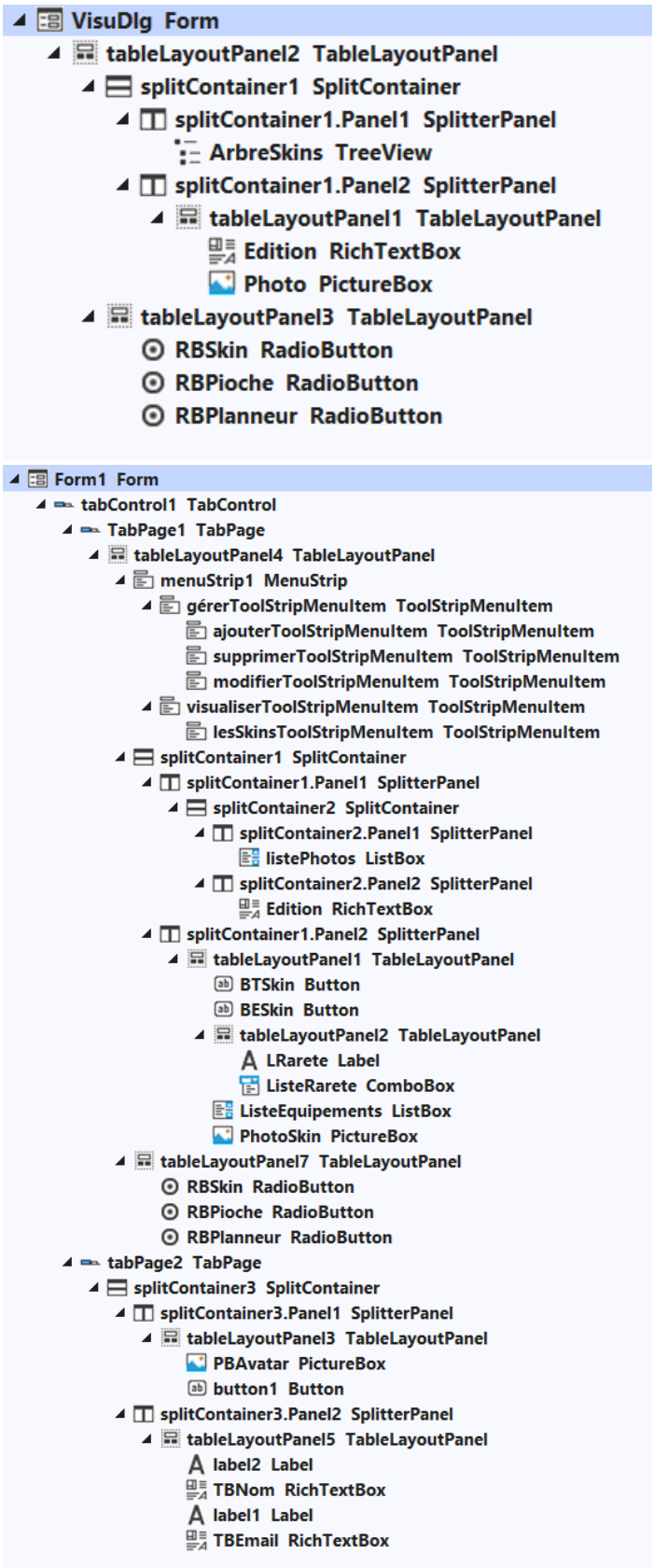
VII. Annexes

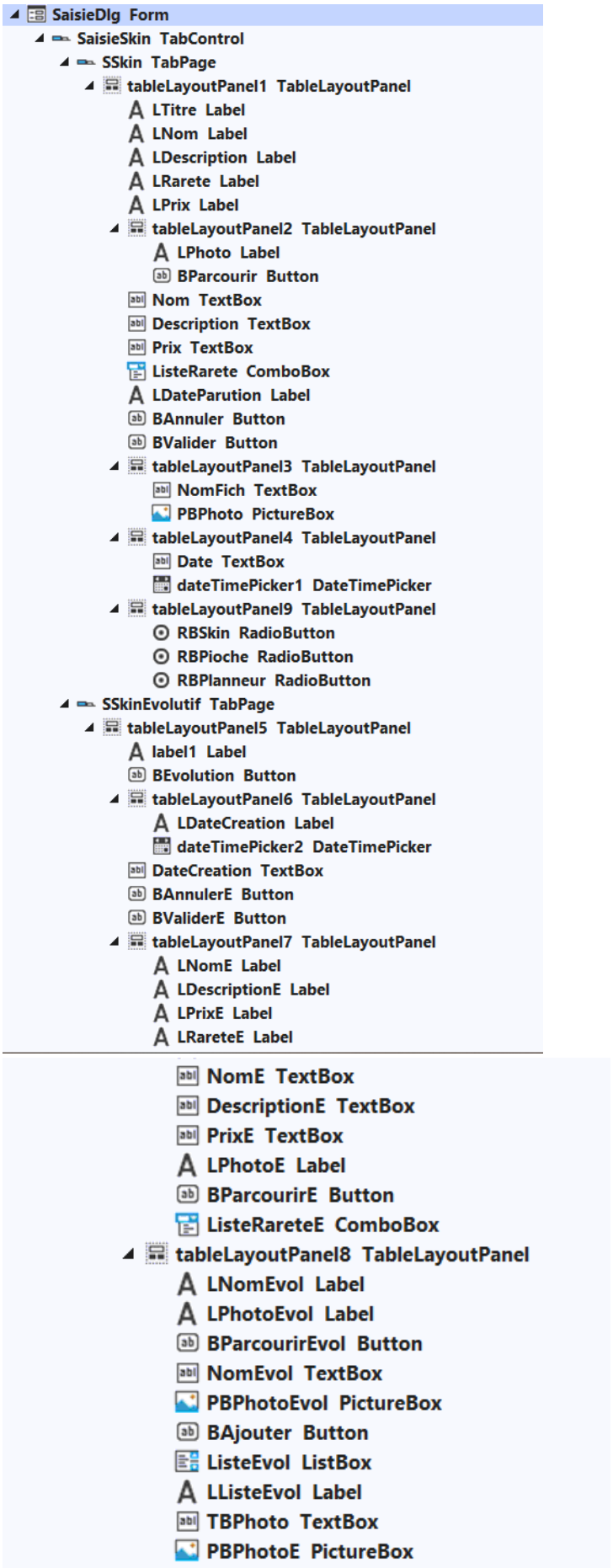
Structures des différentes fenêtres :








VisuDlg Form
tableLayoutPanel2 TableLayoutPanel
splitContainer1 SplitContainer
splitContainer1.Panel1 SplitterPanel
ArbreSkins TreeView
splitContainer1.Panel2 SplitterPanel
tableLayoutPanel1 TableLayoutPanel
Edition RichTextBox
Photo PictureBox
tableLayoutPanel3 TableLayoutPanel
RBSkin RadioButton
RBPioche RadioButton
RBPlanneur RadioButton
Form1 Form
tabControl1 TabControl
TabPage1 TabPage
tableLayoutPanel4 TableLayoutPanel
menuStrip1 MenuStrip
gérerToolStripMenuItem ToolStripMenuItem
ajouterToolStripMenuItem ToolStripMenuItem
supprimerToolStripMenuItem ToolStripMenuItem
modifierToolStripMenuItem ToolStripMenuItem
visualiserToolStripMenuItem ToolStripMenuItem
lesSkinsToolStripMenuItem ToolStripMenuItem
splitContainer1 SplitContainer
splitContainer1.Panel1 SplitterPanel
splitContainer2 SplitContainer
splitContainer2.Panel1 SplitterPanel
listePhotos ListBox
splitContainer2.Panel2 SplitterPanel
Edition RichTextBox
splitContainer1.Panel2 SplitterPanel
tableLayoutPanel1 TableLayoutPanel
BTSkin Button
BESkin Button
tableLayoutPanel2 TableLayoutPanel
LRarete Label
ListeRarete ComboBox
ListeEquipements ListBox
PhotoSkin PictureBox
tableLayoutPanel7 TableLayoutPanel
RBSkin RadioButton
RBPioche RadioButton
RBPlanneur RadioButton
tabPage2 TabPage
splitContainer3 SplitContainer
splitContainer3.Panel1 SplitterPanel
tableLayoutPanel3 TableLayoutPanel
PBAvatar PictureBox
button1 Button
splitContainer3.Panel2 SplitterPanel
tableLayoutPanel5 TableLayoutPanel
label2 Label
TBNom RichTextBox
label1 Label
TBEmail RichTextBox


SaisieDlg Form
SaisieSkin TabControl
SSkin TabPage
tableLayoutPanel1 TableLayoutPanel
LTitre Label
LNom Label
LDescription Label
LRarete Label
LPrix Label
tableLayoutPanel2 TableLayoutPanel
LPhoto Label
BParcourir Button
Nom TextBox
Description TextBox
Prix TextBox
ListeRarete ComboBox
LDateParution Label
BAnnuler Button
BValider Button
tableLayoutPanel3 TableLayoutPanel
NomFich TextBox
PBPhoto PictureBox
tableLayoutPanel4 TableLayoutPanel
Date TextBox
dateTimePicker1 DateTimePicker
tableLayoutPanel9 TableLayoutPanel
RBSkin RadioButton
RBPioche RadioButton
RBPlanneur RadioButton
SSkinEvolutif TabPage
tableLayoutPanel5 TableLayoutPanel
label1 Label
BEvolution Button
tableLayoutPanel6 TableLayoutPanel
LDateCreation Label
dateTimePicker2 DateTimePicker
DateCreation TextBox
BAnnulerE Button
BValiderE Button
tableLayoutPanel7 TableLayoutPanel
LNomE Label
LDescriptionE Label
LPrixE Label
LRareteE Label
NomE TextBox
DescriptionE TextBox
PrixE TextBox
LPhotoE Label
BParcourirE Button
ListeRareteE ComboBox
tableLayoutPanel8 TableLayoutPanel
LNomEvol Label
LPhotoEvol Label
BParcourirEvol Button
NomEvol TextBox
PBPhotoEvol PictureBox
BAjouter Button
ListeEvol ListBox
LListeEvol Label
TBPhoto TextBox
PBPhotoE PictureBox