Preuves

Conception et Programmation Orientées Objets (C#, .NET)

Documents

- Je sais concevoir un diagramme de classes qui représente mon application [sur 9 points]

Voir Documentation C#

- Je sais réaliser un diagramme de paquetages qui illustre bien l'isolation entre les parties de mon application [sur 2 points]

Voir Documentation C#

- Je sais décrire mes deux diagrammes en mettant en valeur et en justifiant les éléments essentiels [sur 9 points]

Voir Documentation C#

Programmation

- Je maîtrise les bases de la programmation C# (classes, structures, instances…) [sur 2 points]

Classes: voir n'importe quel fichier .cs
Enum: voir LauncherName.cs
Instance : voir n'importe quel fichier .cs
autres : voir n'importe quel fichier .cs
Exemple :

```
public Jeu(string nom, string dossier, string exec, LauncherName launcher = LauncherName.Autre) : base(nom)
{
    Dossier = dossier;
    Exec = exec;
    Launcher = launcher;
}

public Jeu(string nom, string dossier, string exec, string image, string icone, string note, string description, LauncherName launcher = LauncherName.Autre, bool isManuallyAdded = false) : this(nom, dossier, exec, launcher)
{
    Image = image;
    Icone = icone;
    Note = note;
    Description = description;
    IsManuallyAdded = isManuallyAdded;
}
```

- Je sais utiliser l'abstraction à bon escient (héritage, interfaces, polymorphisme) [sur 3 points]

Jeu et Launcher hérite tous les deux de Element qui est abstract(heritage). Nous avons une collection d'Element dans notre manager (polymorphisme). Nous avons une interface IPersistance qui assure que les classes qui l'implemente ait les fonctions suivantes: Load et Save(interfaces)

```
public abstract class Element : IEquatable<Element>, IComparable<Element>
{
    public string Nom { get; set; } = null;
    protected Element(string nom)
    {
        Nom = nom;
    }
```

```
public class Jeu : Element, INotifyPropertyChanged
{
    private string exec = null, description = null, image = null, note = "", icone = null;
    public bool IsManuallyAdded { get; set; } = false;
    public string Dossier { get; set; } = null;
```

```
public class Launcher : Element
{
    private int nbjeux = 0;
    public int NbJeux
```

```
public class Data
{
    public IList<Element> Elements { get; }
```

- Je sais gérer des collections simples (tableaux, listes…) [sur 2 points]

La plupart de nos collections sont des collections simples
Exemple : ligne 13 de Data.cs

```
public class Data
{
    public IList<Element> Elements { get; }
```

- Je sais gérer des collections avancées (dictionnaires) [sur 2 points]

Nous nous servons de dictionnaires dans les classes OriginSearcher et RiotSearcher (paquet Modele)
Exemple : ligne 11 de RiotSearcher.cs

```
IDictionary<string, string> dossierToNom = new Dictionary<string, string>();
```

- Je sais contrôler l'encapsulation au sein de mon application [sur 5 points]

Le paquet Modele expose à l'extérieur uniquement les types et la façade(manager), tout le reste est soit private,protected ou internal.
Exemple : Manager.cs

```
public string Pattern { get; set; } = "Rechercher";
private readonly Data data;

private IPersistance Persistance { get; set; }

public event PropertyChangedEventHandler PropertyChanged;
```

- Je sais tester mon application [sur 4 points]

Nous disposons de jeux de test unitaires et fonctionnels (projets « Test Fonctionnels » et « Test Unitaires »)

```
[Fact]
public void AjoutJeu()
{
    Modele.AjoutJeu(Jeu);
    Assert.True(Modele.Affichage.Contains(Jeu));
}

[Fact]
public void ModifDetail()
{
    var expected = "descriptionmodif";
    Modele.ElementSelected = Jeu;
    Modele.ModifDetail(Jeu.Image, Jeu.Description + "modif", Jeu.Exec, Jeu.Icone);//on test que description car les
    Assert.Equal(expected, Jeu.Description);
}

[Fact]
public void Recherche()
{
    Modele.Pattern = "a";
    var jeuxFiltrer = Modele.Affichage;
    Assert.True(jeuxFiltrer.All(j => j.Nom.Contains(Modele.Pattern, StringComparison.OrdinalIgnoreCase)));
}
```

- Je sais utiliser LINQ [sur 1 point]

Nous utilisons beaucoup LINQ l'endroit où on l'utilise le plus est la fonction de filtration dans la classe GameSearcher.

```
protected string Filter(string[] executables, string nom = null, LauncherName launcher = LauncherName.Autre)
{
    int archi = Environment.Is64BitOperatingSystem ? 64 : 32;
    if (launcher == LauncherName.Riot)//Riot est un peu speciale (peu de jeux)(launcher....etc) donc hardcodage de ceux la
    {
        return executables.First(e => e.Contains("LeagueClient.exe") || e.Contains("VALORANT.exe") || e.Contains("LoR.exe"));
    }
    IEnumerable<string> res = executables.Where(e => FilterIndesirables(e)); //apllication du filtre
    if (!res.Any()) //si tout a disparu dans le filtre
    {
        return executables[0];
    }
    else if (nom != null && res.Count() > 1) //si un nom est defini on prend les executables contenant le nom (si il y en a)
    {
        var temp = res.Where(e => Path.GetFileName(e).Contains(nom, StringComparison.OrdinalIgnoreCase));
        res = temp.Any() ? temp : res;
        temp = res.Where(e => Path.GetFileName(e).Contains(nom.Replace(" ", ""), StringComparison.OrdinalIgnoreCase));
        res = temp.Any() ? temp : res;
    }
    if (res.Count() > 1 && res.Any(e => e.Contains("bin", StringComparison.OrdinalIgnoreCase))) //preferer les exe contenu dans un dossier bin ou binaries (si il y en a)
    {
        res = res.Where(e => e.Contains("bin", StringComparison.OrdinalIgnoreCase));
    }
    if (res.Count() > 1 && res.Any(e => e.Contains(archi.ToString()))) //preferer les exe contenu dans un dossier 64 ou 32 bit en fonction de l'architecture supporté (si il y en a) (on
    {
        res = res.Where(e => e.Contains(archi.ToString(), StringComparison.OrdinalIgnoreCase));
    }
    return res.First();
}

protected bool FilterIndesirables(string executable)
{
    return !(executable.Contains("Unins", StringComparison.OrdinalIgnoreCase) || executable.Contains("Crash", StringComparison.OrdinalIgnoreCase) || executable.Contains("Helper", String
        || executable.Contains("AntiCheat", StringComparison.OrdinalIgnoreCase) || executable.Contains("Downloader", StringComparison.OrdinalIgnoreCase) || executable.Contains("Upload")
        || executable.Contains("Unreal", StringComparison.OrdinalIgnoreCase) || executable.Contains("Redist", StringComparison.OrdinalIgnoreCase) || executable.Contains("egodumper", Str
        || executable.Contains("mono.exe", StringComparison.OrdinalIgnoreCase)); //traitement des cas communs
}
```

- Je sais gérer les évènements [sur 1 point]

Nous utilisons les événements dans notre manager pour notifier la vue de changement, et dans Jeu pour notifier le changement d'une propriété (via l'événement NotifyPropertyChanged)
Exemple:ligne 84 de Manager.cs

```
public void AjoutJeu(LauncherName launcher, string exec)
{
    if (File.Exists(exec))
    {
        data.AjoutJeu(launcher, exec);
        NotifyPropertyChanged("Affichage");
        NotifyPropertyChanged("JeuLauncherSelected");
    }
}
```

IHM : Interface Homme-Machine (XAML, WPF)

Documents

- Je sais décrire le contexte de mon application pour qu'il soit compréhensible par tout le monde [sur 4 points]

Voir Documentation IHM

- Je sais dessiner des sketchs pour concevoir les fenêtres de mon application [sur 4 points]

Voir Documentation IHM

- Je sais enchaîner mes sketchs au sein d'un story-board [sur 4 points]

Voir Documentation IHM

- Je sais concevoir un diagramme de cas d'utilisation qui représente les fonctionnalités de mon application [sur 5 points]

Voir Documentation IHM

- Je sais concevoir une application ergonomique [sur 2 points]

Explication dans la Documentation IHM

- Je sais concevoir une application avec une prise en compte de l'accessibilité [sur 1 point]

Explication dans la Documentation IHM

Programmation

- Je sais choisir mes layouts à bon escient [sur 1 points]

Voir n'importe quel fichier .***xaml***
Exemple:ligne 20 de MainWindow.xaml

```
    <DockPanel>
        <Grid DockPanel.Dock="Top">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <StackPanel Margin="10,0,0,0" Grid.Column="0" Orientation="Horizon
                <Image Margin="0,0,5,0" Height="35" Width="35" Source="/Icones
                <TextBox Text="{Binding Pattern, Mode=TwoWay, UpdateSourceTrig
                         VerticalAlignment="Center" Width="120" x:Name="BarreD


            </StackPanel>
            <Image Grid.Column="1" Height="40" Width="40" Source="/Icones;Comp
              Margin="5,5,5,5" MouseLeftButtonUp="OuvrirParametre" VerticalAl
        </Grid>
        <Button DockPanel.Dock="Bottom" Height="20"
```

- Je sais choisir mes composants à bon escient [sur 1 point]

Voir n'importe quel fichier *.xaml*
Exemple: AjoutDetailWindow.xaml

```
<StackPanel Grid.ColumnSpan="2" Grid.Row="0" Style="{StaticResource SPNomValueExplorer}">
    <TextBlock Text="Lien vers l'exec. : "/>
    <TextBox x:Name="textBoxLienExecutable" Text="{Binding Executable}"/>
    <Image MouseLeftButtonDown="ChercherExec"/>
</StackPanel>
<StackPanel Grid.ColumnSpan="2" Grid.Row="1" Style="{StaticResource SPNomValueExplorer}">
    <TextBlock Text="Lien vers l'icone : "/>
    <TextBox x:Name="textBoxLienIcone" Text="{Binding Icone}"/>
    <Image MouseLeftButtonDown="ChercherIcone"/>
</StackPanel>
<StackPanel Grid.ColumnSpan="2" Grid.Row="2" Style="{StaticResource SPNomValueExplorer}" Margin="0,0,0,13">
    <TextBlock Text="Lien vers l'image: "/>
    <TextBox x:Name="textBoxLienImage" Text="{Binding Image}"/>
    <Image MouseLeftButtonDown="ChercherImage"/>
</StackPanel>
```

- Je sais créer mon propre composant [sur 2 points]

Voir les fichiers contenus dans l'arborescente : « Projet\Vues\User Controls » et « Projet\Vues\windowParts»
Exemple :VignetteJeu.xaml

```
<UserControl x:Class="Vues.User_Controls.VignetteJeu"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Vues.User_Controls"
             xmlns:conv="clr-namespace:Vues.Converters"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800" MouseLeftButtonDown="VignetteClicked">
    <UserControl.Resources>
        <conv:StringToImage x:Key="StringToImageConv"/>
    </UserControl.Resources>
    <DockPanel Margin="40">
        <Image DockPanel.Dock="Top" Source="{Binding Icone, Converter={StaticResource StringToImageConv}}" Width="100"/>
        <TextBlock DockPanel.Dock="Bottom" Text="{Binding Nom}" HorizontalAlignment="Center" TextTrimming="CharacterEllipsis">
            <TextBlock.ToolTip>
                <ToolTip>
                    <TextBlock Text="{Binding Nom}"/>
                </ToolTip>
            </TextBlock.ToolTip>
        </TextBlock>
    </DockPanel>
</UserControl>
```

- Je sais personnaliser mon application en utilisant des ressources et des styles [sur 2 points]

Voir App.XAML, ce fichier contient quelques styles utilisé dans des fenêtres comme AjoutDetailWindow ou AjoutJeuWindow
Exemple: App.xaml

```
<Style x:Key="SPNomValueExplorer" TargetType="StackPanel">
    <Style.Resources>
        <Style TargetType="TextBlock">
            <Setter Property="VerticalAlignment" Value="Center"></Setter>
            <Setter Property="HorizontalAlignment" Value="Left"></Setter>
            <Setter Property="Width" Value="110"></Setter>
        </Style>
        <Style TargetType="TextBox" BasedOn="{StaticResource MaterialDesignTextBoxBase}">
            <Setter Property="Width" Value="500"></Setter>
            <Setter Property="VerticalAlignment" Value="Center"></Setter>
        </Style>
        <Style TargetType="Image">
            <Setter Property="Margin" Value="5"></Setter>
            <Setter Property="Width" Value="30"></Setter>
            <Setter Property="Source" Value="/Icones;Component/parcourir.png"></Setter>
        </Style>
    </Style.Resources>
    <Setter Property="Orientation" Value="Horizontal"></Setter>
    <Setter Property="HorizontalAlignment" Value="Center"></Setter>
</Style>
```

- Je sais utiliser les DataTemplate (locaux et globaux) [sur 2 points]

Nous n'avons pas de DataTemplate globaux
Nous utilisons des DataTemplate DetailLauncher.*xaml*, FolderExplorerView.xaml et MasterDetail.xaml
Exemple : MasterDetail.xaml

```
<ListBox.Resources>
    <DataTemplate DataType="{x:Type modele:Jeu}">
        <User_Controls:JeuUc Width="{Binding Path=ActualWidth, ElementName=ListeJeu}"/>
    </DataTemplate>
    <DataTemplate DataType="{x:Type modele:Launcher}">
        <User_Controls:LauncherUc />
    </DataTemplate>
</ListBox.Resources>
```

- Je sais intercepter les évènements de la vue [sur 2 points]

Tous les boutons ont au moins une méthode abonné à l'événement click.
Une méthode est abonné à l'événement TextChanged de la TextBox de recherche afin d'actualiser la recherche a chaque caractère entré.
Exemple : ligne 59 *MainWindow*.xaml.cs

```
private void Recherche(object sender, TextChangedEventArgs e)//appeler quand le texte de la barre de recherche change
{
    if ((App.Current as App).Manager.SearchActivated)
    {
        (App.Current as App).Manager.UpdateRecherche();
    }
}
```

- Je sais notifier la vue depuis des évènements métier [sur 2 points]

Nous utilisons les événements dans notre manager pour notifier la vue de changement, et dans Jeu pour notifier le changement d'une propriété (via l'événement NotifyPropertyChanged)
Exemple : Manager.cs

```
public void AjoutJeu(LauncherName launcher, string exec)
{
    if (File.Exists(exec))
    {
        data.AjoutJeu(launcher, exec);
        NotifyPropertyChanged("Affichage");
        NotifyPropertyChanged("JeuLauncherSelected");
    }
}
```

- Je sais gérer le DataBinding sur mon master [sur 2 points]

Nous avons du DataBinding sur toutes nos fenêtres et UserControl
Exemple : MasterDetail.xaml

```
<ListBox x:Name="ListeJeu" Grid.Column="0" BorderThickness="0" ItemsSource="{Binding Affichage}"
```

- Je sais gérer le DataBinding sur mon détail [sur 2 points]

Nous avons du DataBinding sur toutes nos fenêtres et UserControl
Exemple : DetailJeu.xaml

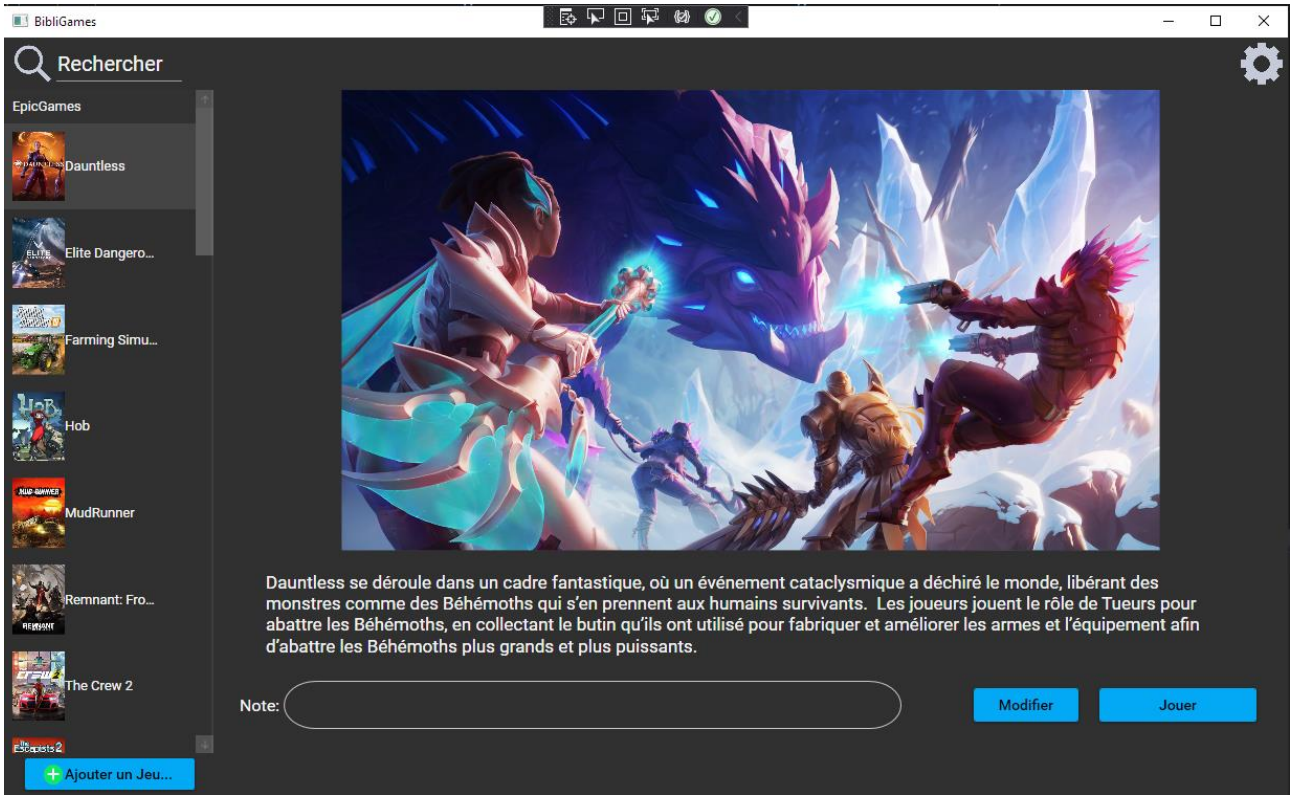
```
<ScrollViewer DockPanel.Dock="Bottom" VerticalScrollBarVisibility="Auto" Margin="0,20,0,0" MaxHeight="100">
    <TextBlock x:Name="textBlockDescription" TextWrapping="Wrap" Text="{Binding Description}"
               FontSize="17" Margin="50,0,50,0"/>
</ScrollViewer>
<Image DockPanel.Dock="Top" Source="{Binding Image, Converter={StaticResource StringToImageConv}}" Stretch="Uniform" />
```

- Je sais gérer le DataBinding et les Dependency Property sur mes UserControl [sur 2 points]

Nous utilisons pas de Dependency Property

- Je sais développer un Master/Detail [sur 2 points]

Nous avons développé une application de type Master/Detail



Projet Tuteuré S2

Documents

- Je sais mettre en avant dans mon diagramme de classes la persistance de mon application [sur 4 points]

Voir Documentation Projet Tutoré

- Je sais mettre en avant dans mon diagramme de classes ma partie personnelle [sur 4 points]

Voir Documentation Projet Tutoré

- Je sais mettre en avant dans mon diagramme de paquetages la persistance de mon application [sur 2 points]

Voir Documentation Projet Tutoré

- Je sais réaliser une vidéo de 1 à 3 minutes qui montre la démo de mon application [sur 10 points]

Voir Documentation Projet Tutoré

Programmation

- Je sais coder la persistance au sein de mon application [sur 3 points]

Notre application sauvegarde les changement effectué par l'utilisateur.

- Je sais coder une fonctionnalité qui m'est personnelle [sur 3 points]

Voir Documentation Projet Tutoré

- Je sais documenter mon code [sur 2 points]

Nous avons documenter notre code
Exemple :GameSearcher.cs

```
protected string Filter(string[] executables, string nom = null, LauncherName launcher = LauncherName.Autre)
{
    int archi = Environment.Is64BitOperatingSystem ? 64 : 32;
    if (launcher == LauncherName.Riot)//Riot est un peu speciale (peu de jeux)(launcher....etc) donc hardcodage de ceux la
    {
        return executables.First(e => e.Contains("LeagueClient.exe") || e.Contains("VALORANT.exe") || e.Contains("LoR.exe"));
    }
    IEnumerable<string> res = executables.Where(e => FilterIndesirables(e)); //apllication du filtre
    if (!res.Any()) //si tout a disparu dans le filtre
    {
        return executables[0];
    }
    else if (nom != null && res.Count() > 1) //si un nom est defini on prend les executables contenant le nom (si il y en a)
    {
        var temp = res.Where(e => Path.GetFileName(e).Contains(nom, StringComparison.OrdinalIgnoreCase));
        res = temp.Any() ? temp : res;
        temp = res.Where(e => Path.GetFileName(e).Contains(nom.Replace(" ", ""), StringComparison.OrdinalIgnoreCase));
        res = temp.Any() ? temp : res;
    }
    if (res.Count() > 1 && res.Any(e => e.Contains("bin", StringComparison.OrdinalIgnoreCase))) //preferer les exe contenu dans un dossier bin ou binaries (si il y en a)
    {
        res = res.Where(e => e.Contains("bin", StringComparison.OrdinalIgnoreCase));
    }
    if (res.Count() > 1 && res.Any(e => e.Contains(archi.ToString()))) //preferer les exe contenu dans un dossier 64 ou 32 bit en fonction de l'architecture supporté (si il y en a) (on
    {
        res = res.Where(e => e.Contains(archi.ToString(), StringComparison.OrdinalIgnoreCase));
    }
    return res.First();
}

protected bool FilterIndesirables(string executable)
{
    return !(executable.Contains("Unins", StringComparison.OrdinalIgnoreCase) || executable.Contains("Crash", StringComparison.OrdinalIgnoreCase) || executable.Contains("Helper", String
        || executable.Contains("AntiCheat", StringComparison.OrdinalIgnoreCase) || executable.Contains("Downloader", StringComparison.OrdinalIgnoreCase) || executable.Contains("Upload")
        || executable.Contains("Unreal", StringComparison.OrdinalIgnoreCase) || executable.Contains("Redist", StringComparison.OrdinalIgnoreCase) || executable.Contains("egodumper", Str
        || executable.Contains("mono.exe", StringComparison.OrdinalIgnoreCase)); //traitement des cas communs
}
```

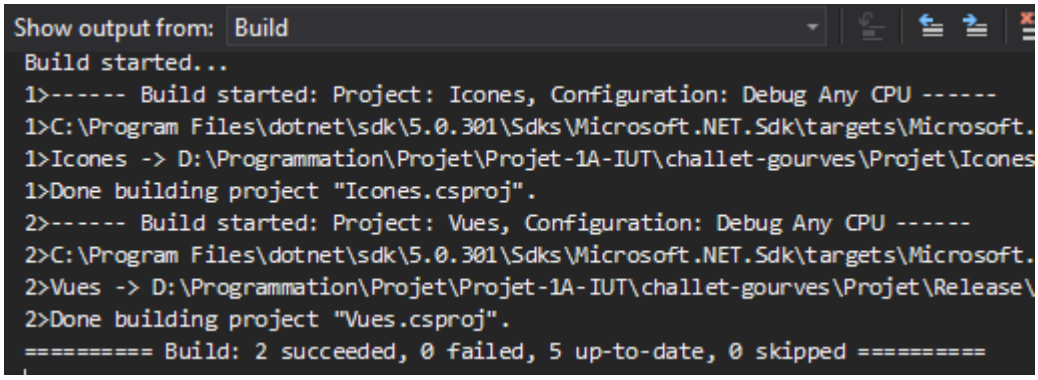
- Je sais utiliser SVN. [sur 2 points]

Nous avons utiliser SVN

Revision	Actions	Author	Date	Message
266		trchallet	vendredi 11 juin 2021 14:49:19	ajout d'une icone pour l'installation de l'application
265		yogourves	vendredi 11 juin 2021 12:21:22	fix typo sur Diagrames.mdj Debut Preuve
264		yogourves	jeudi 10 juin 2021 18:44:39	fix bug IsSaveOk
263		trchallet	jeudi 10 juin 2021 18:41:53	ajout Ctrl + p au survole de l'icone parametre
262		yogourves	jeudi 10 juin 2021 17:51:37	fix bug sur LoadElements
261		trchallet	jeudi 10 juin 2021 17:05:13	Nettoyage des using
260		yogourves	jeudi 10 juin 2021 16:59:44	fix bug sur SteamSearcher
259		yogourves	jeudi 10 juin 2021 16:25:17	mise en place de style

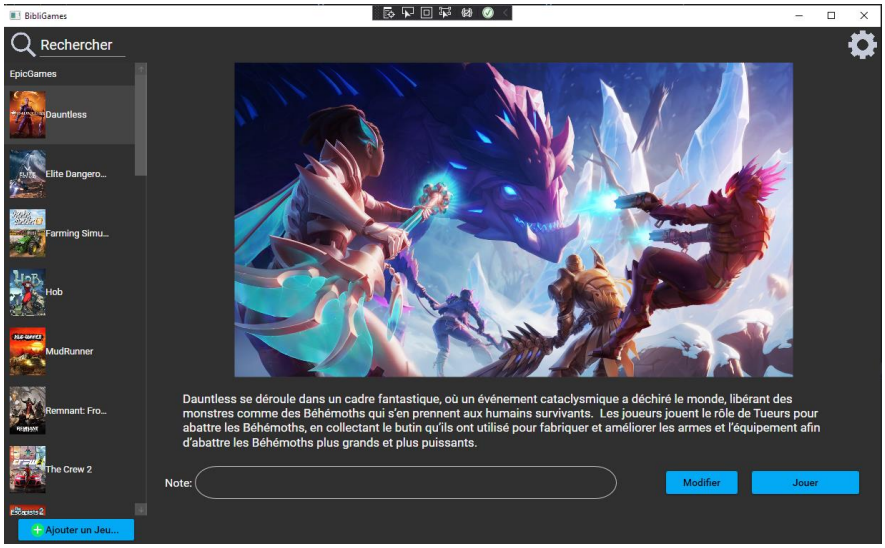
- Je sais développer une application qui compile [sur 3 point]

Notre application compile



- Je sais développer une application fonctionnelle [sur 5 point]

Notre application est fonctionnelle



- Je sais mettre à disposition un outil pour déployer mon application [sur 2 points]

Nous avons a disposition un outil pour déployer mon application