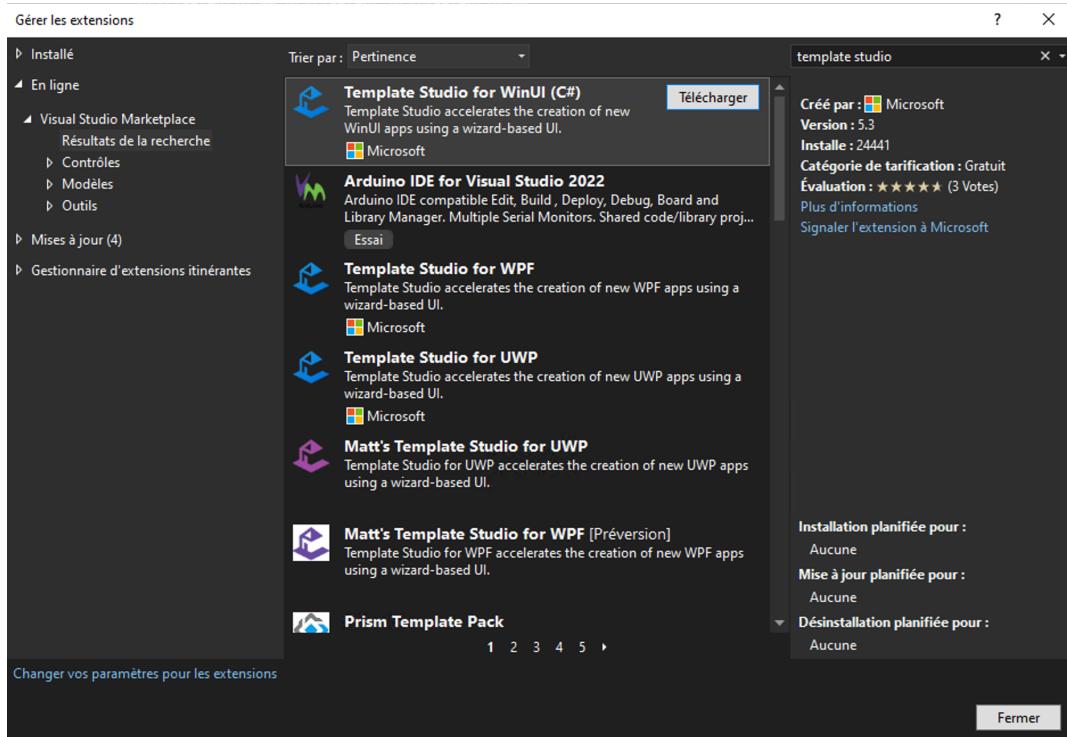


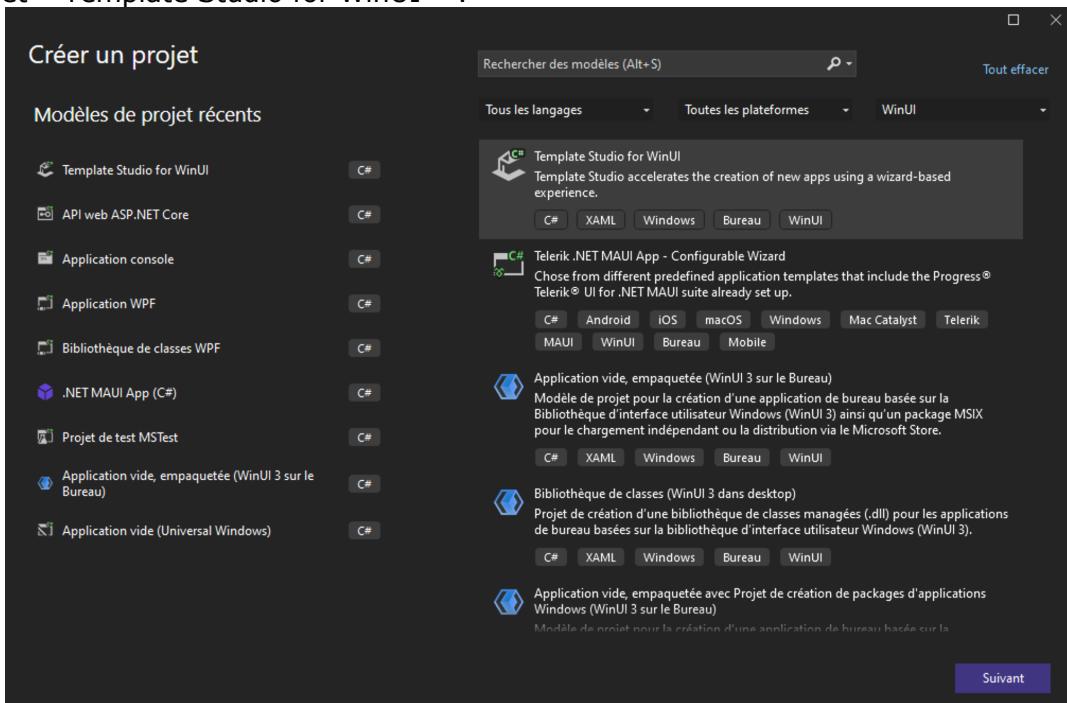
Vous pourrez soit utiliser l'extension « Template Studio for WinUI » (voir ci-dessous), soit créer des pages ou fenêtres vierges, comme en TP1 et 2.

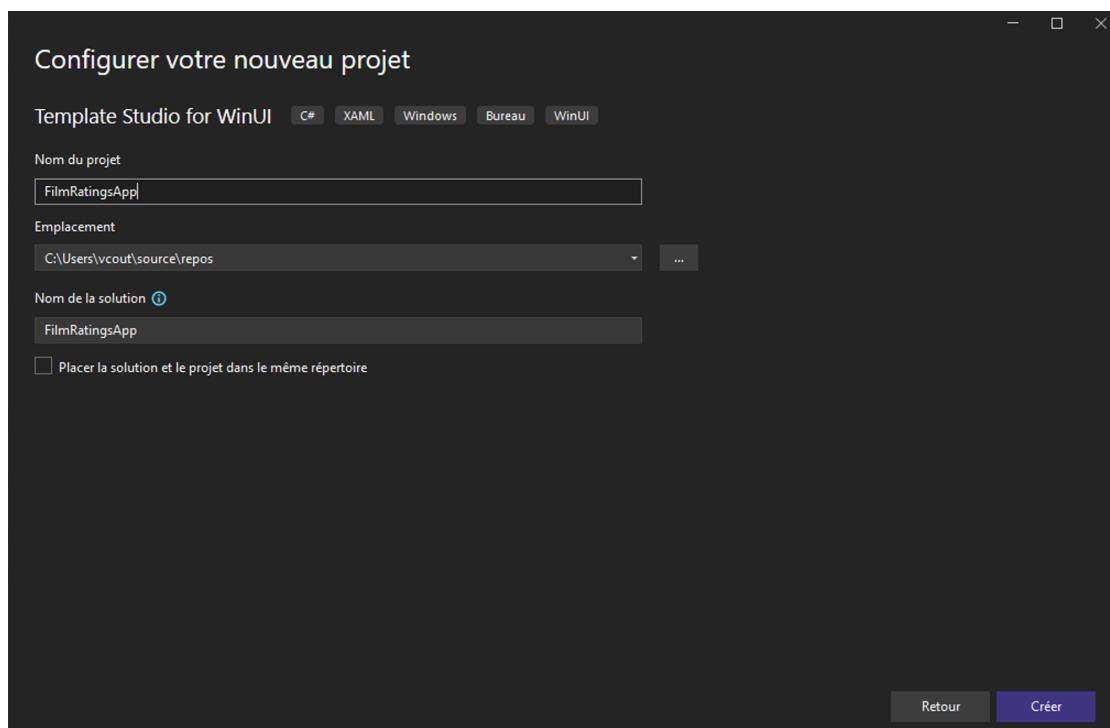
1. Installation de l'extension « Template Studio for WinUI » et création du projet

Installer l'extension « Template Studio for WinUI » (menu *Extensions > Gérer les extensions*) et redémarrer Visual Studio :

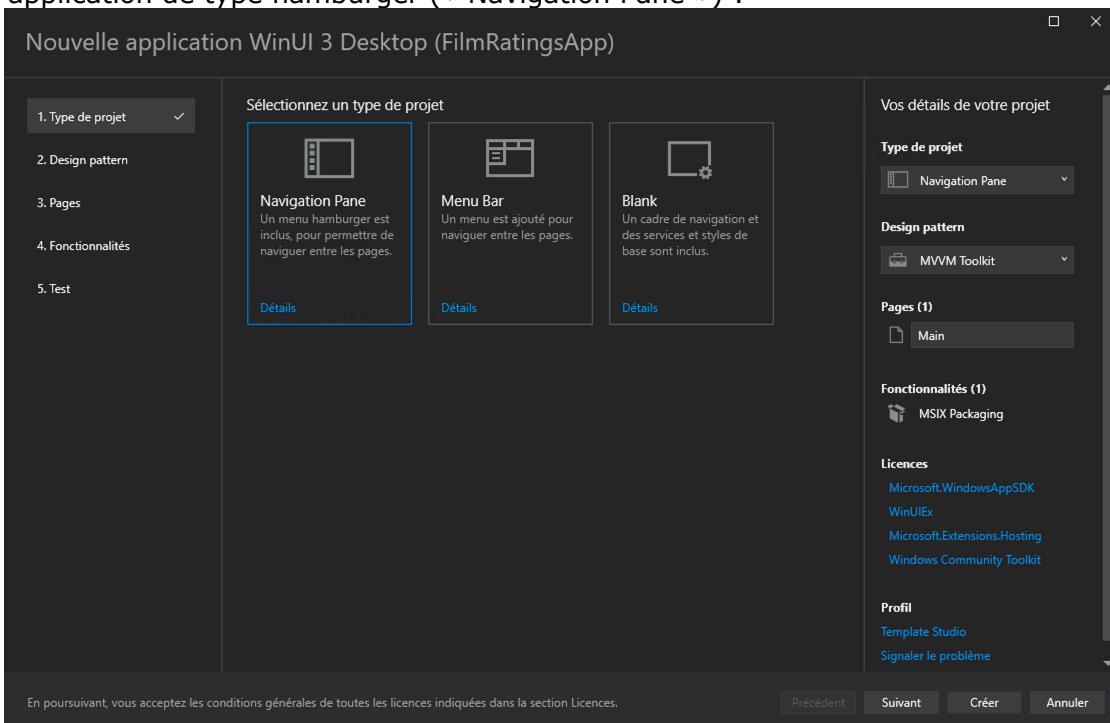


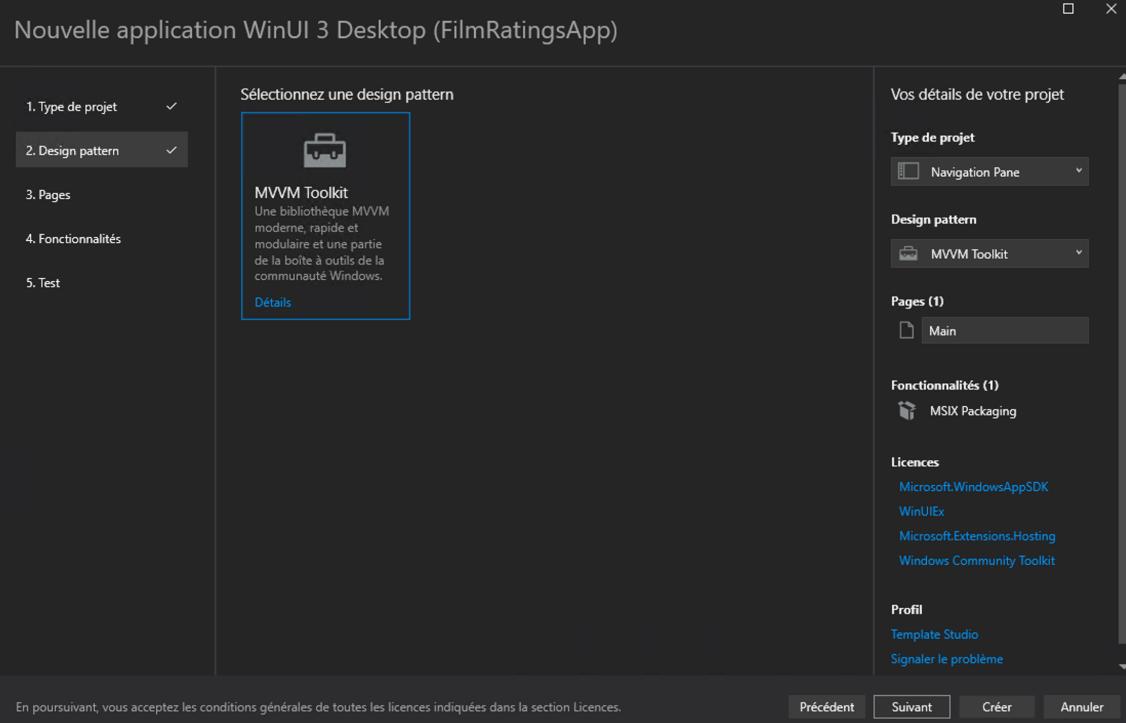
Créer un projet « Template Studio for WinUI » :



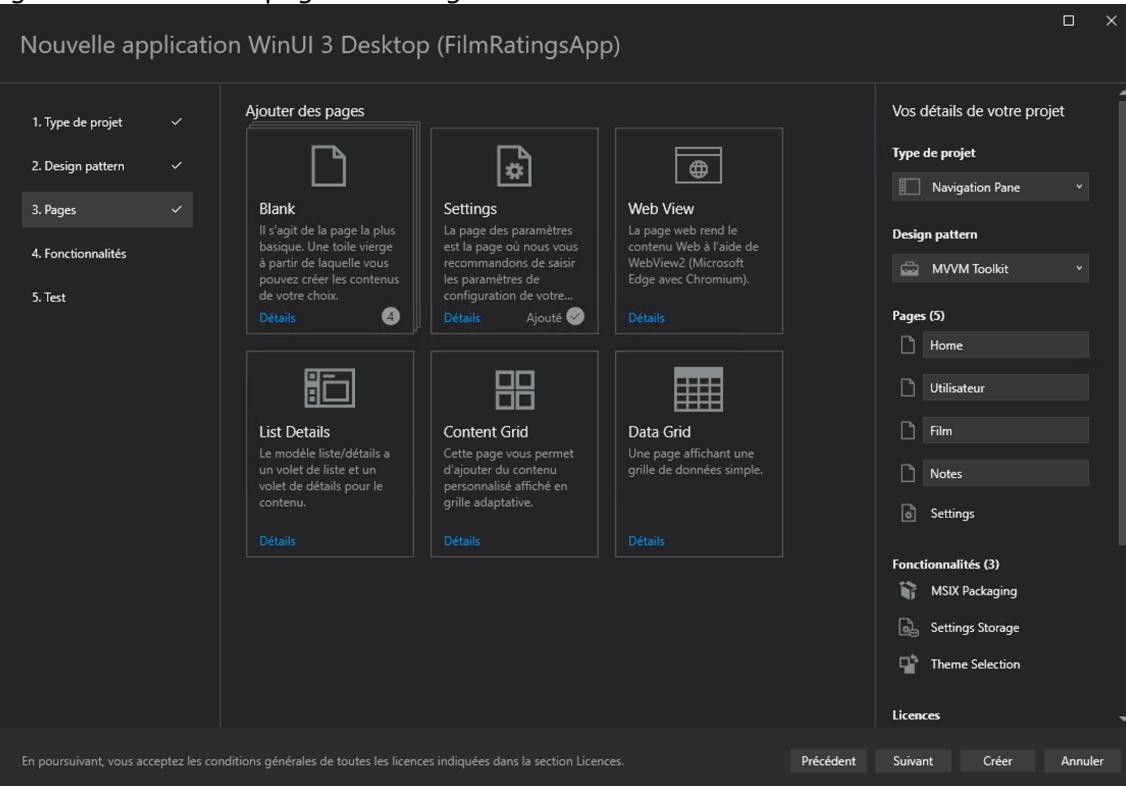


Ajouter une application de type hamburger (« Navigation Pane ») :

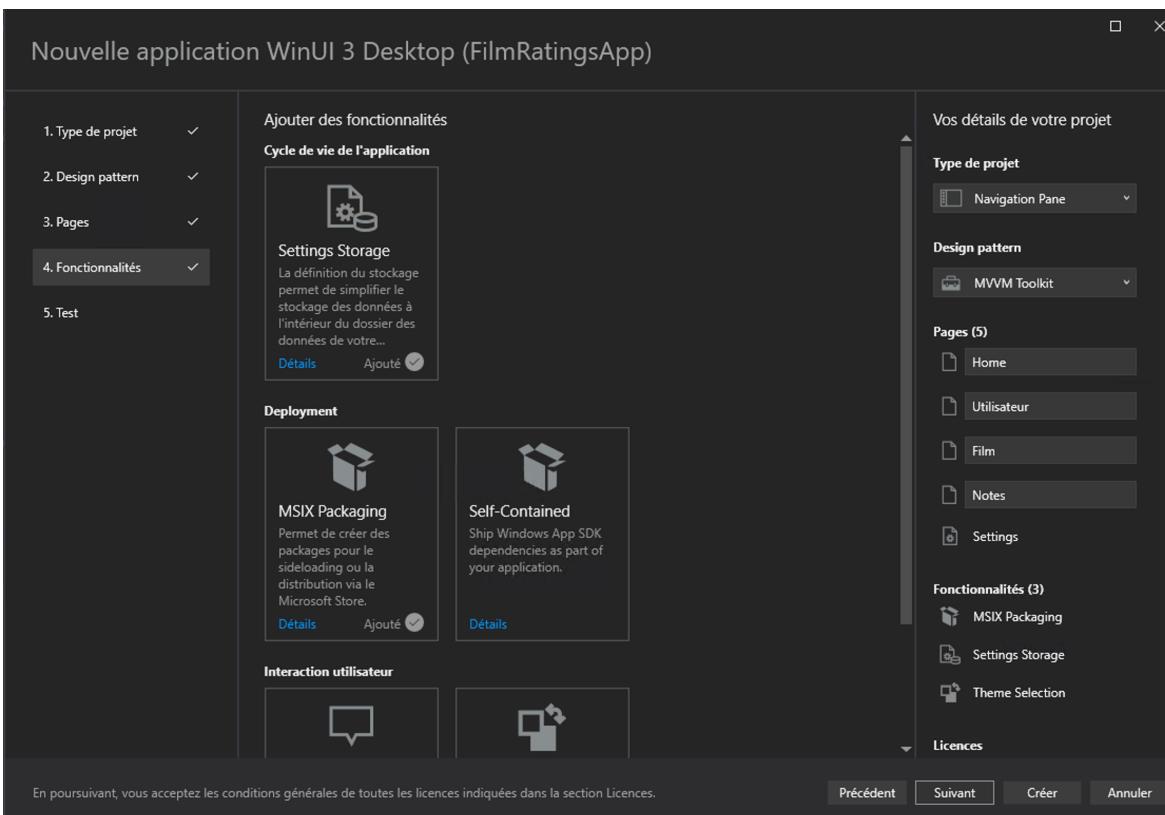




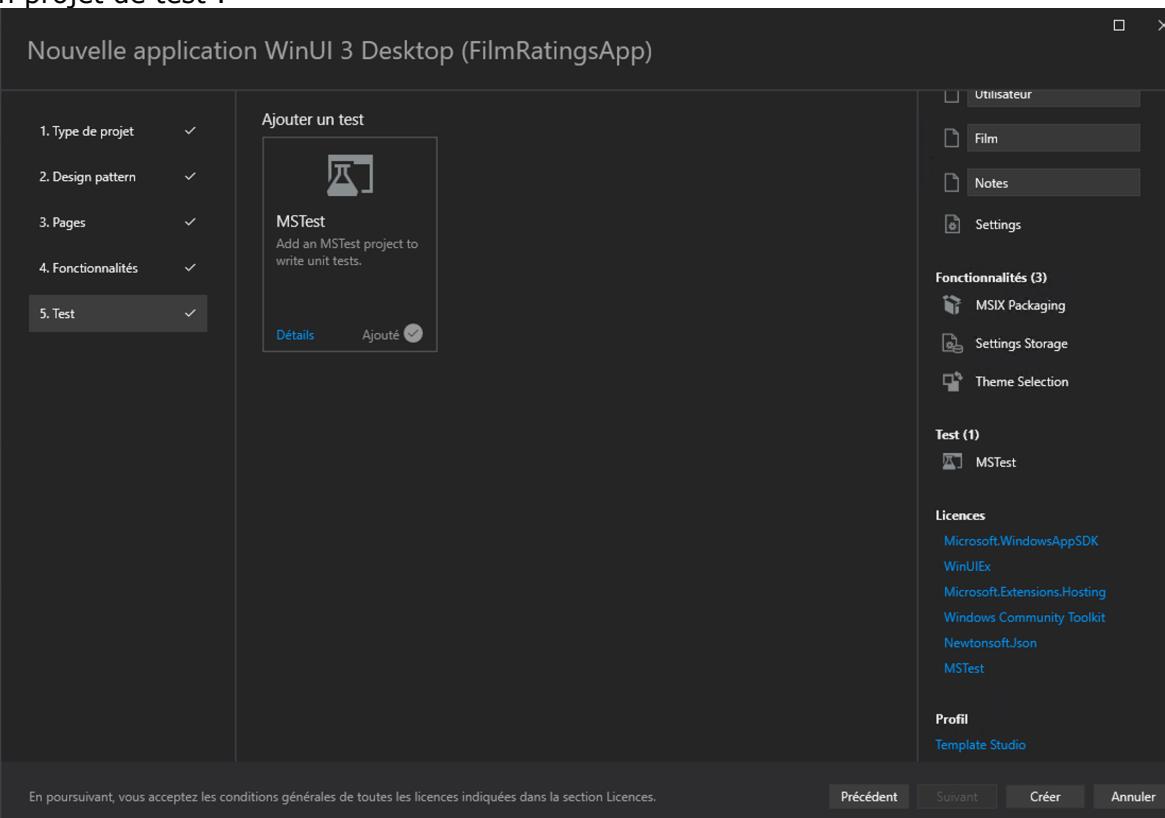
Ajouter 4 pages « Blank » et 1 page « Settings » :



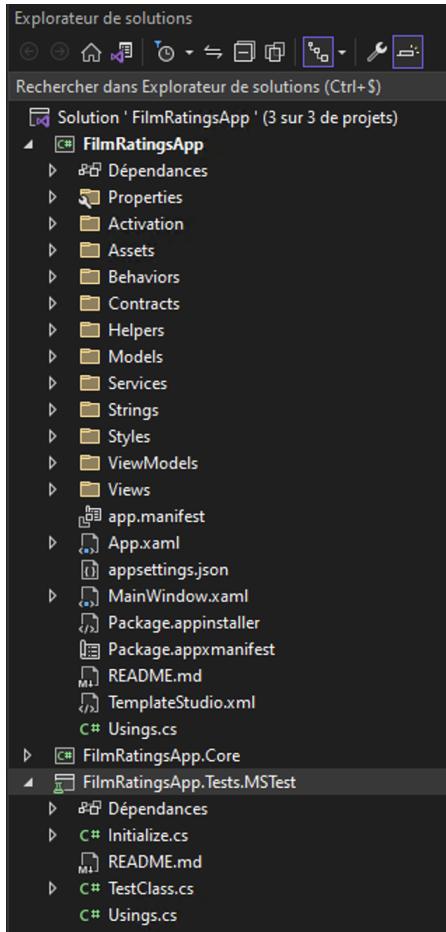
Pas de fonctionnalités supplémentaires :



Ajouter un projet de test :



3 projets sont créés :



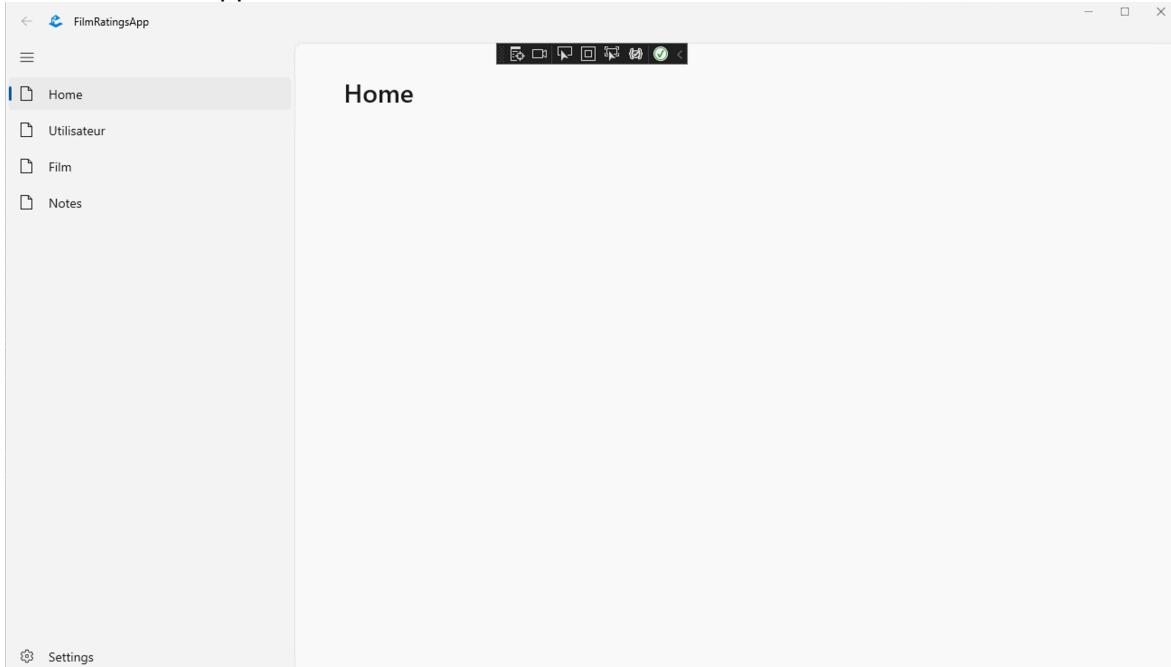
Le 1^{er} projet dans lequel vous ajouterez votre code.

Le 2^{ème} projet (.Core) que vous ne modifierez pas.

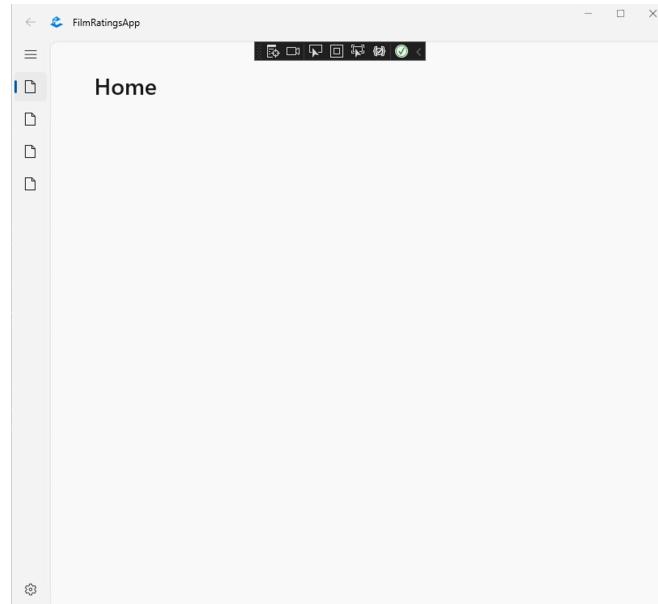
Le projet de tests unitaires.

Penser à ajouter la balise <WindowsSdkPackageVersion>10.0.19041.38</WindowsSdkPackageVersion> au fichier XML projet.

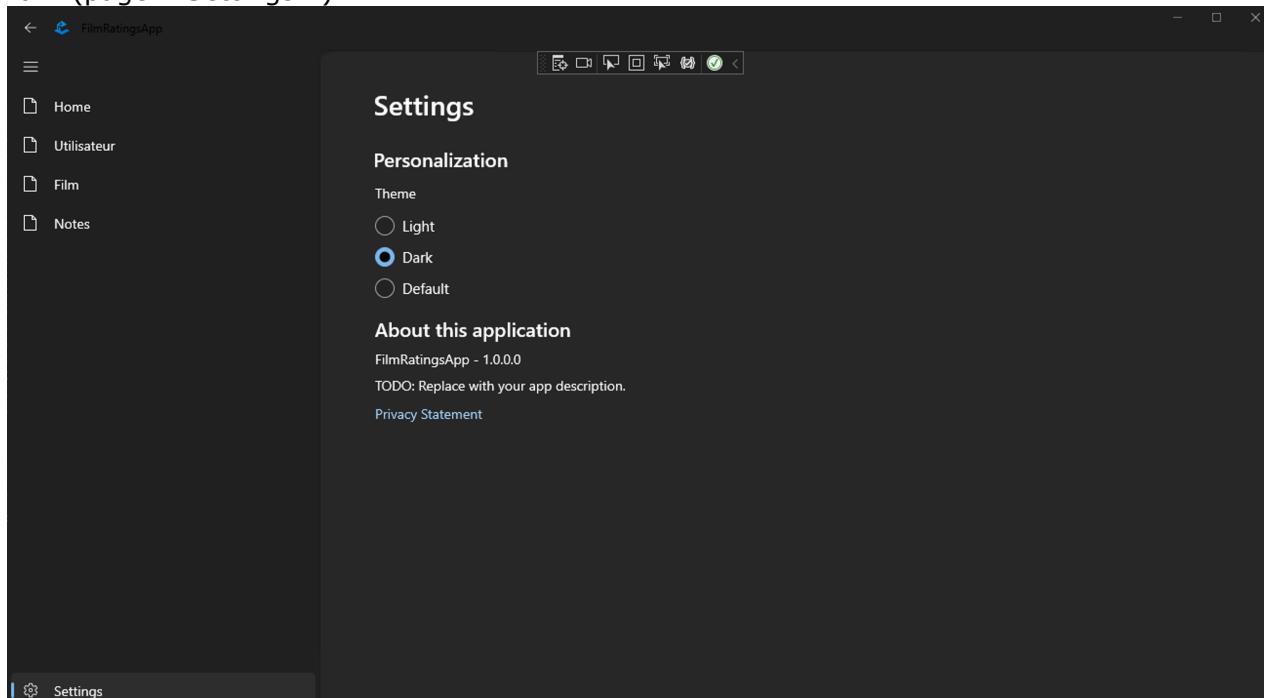
Tester le lancement de l'application.



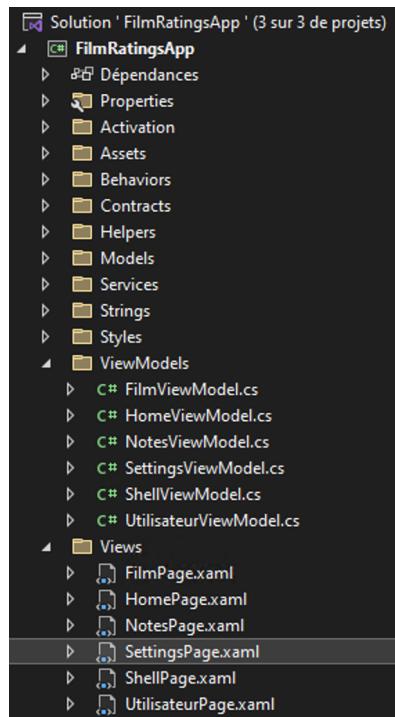
L'application est responsive-design :



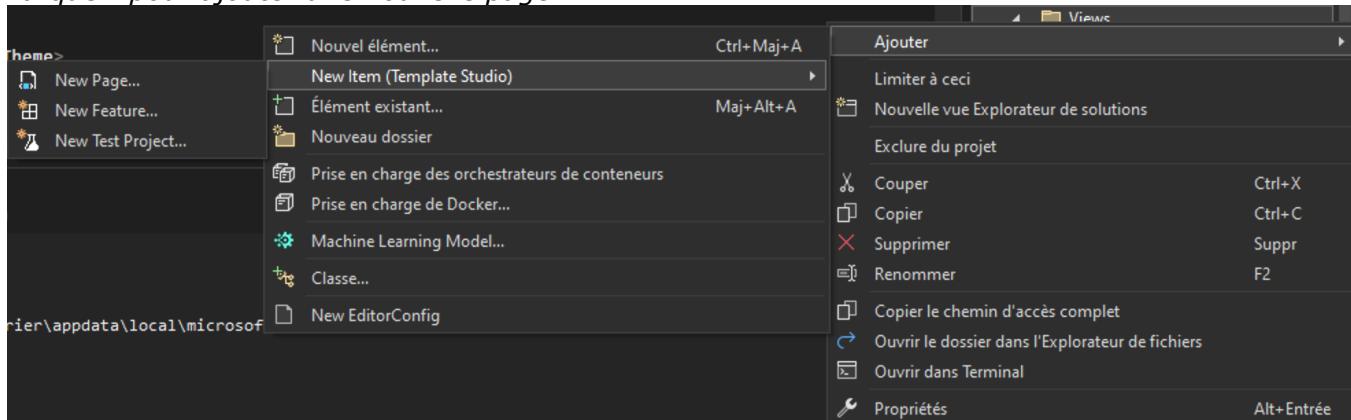
Mode Dark (page « Settings ») :



Toutes les pages sont stockées dans le dossier *Views* et les View Models associés dans le dossier *View Models* :



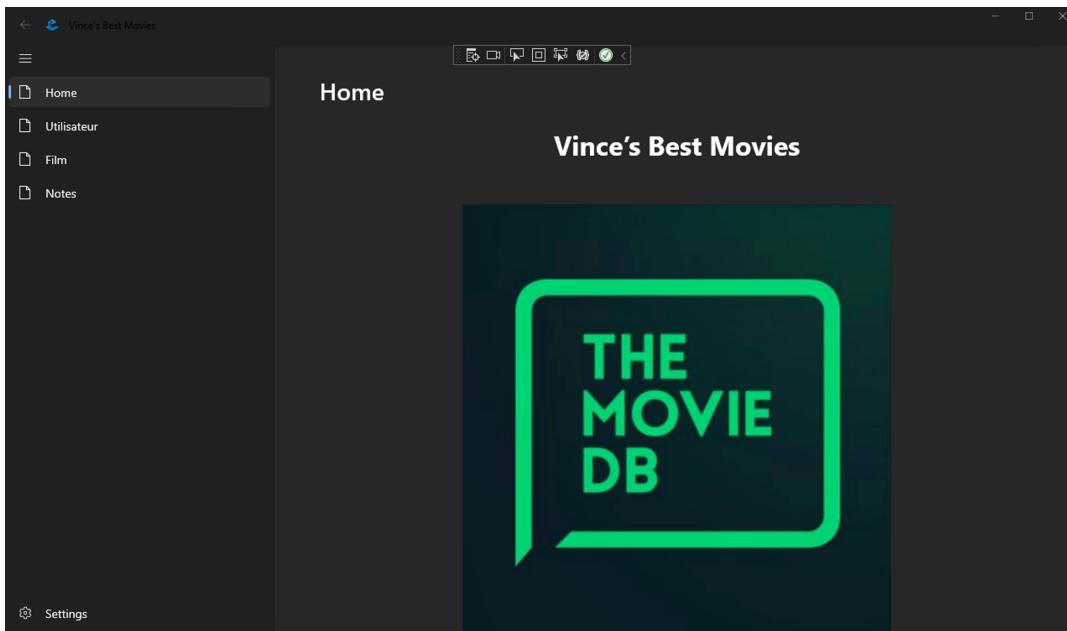
Remarque : pour ajouter une nouvelle page :



Créer un repo sur GitHub.

2. Page d'accueil

Visuel indicatif de la page d'accueil :



Au démarrage de l'application, une page `HomePage` est chargée. Celle-ci contient, par exemple, un lien vers l'image « The Movie DataBase » :

https://play-lh.googleusercontent.com/I03niAyss5tFXAQP176P0Jk5rg_A_hfKPNqzC4gb15WjLPjo5I-f7oIZ9Dqxw2wPBAg

Code XAML de la `HomePage` :

```
<RelativePanel>
    <TextBlock x:Name="NomAppli" Text="Vince's Best Movies"
        RelativePanel.AlignHorizontalCenterWithPanel="True"
        FontWeight="Bold" FontSize="32" Foreground="White"/>
    <Image x:Name="logo" Source="https://play-lh.googleusercontent.com/I03niAyss5tFXAQP176P0Jk5rg_A_hfKPNqzC4gb15WjLPjo5I-f7oIZ9Dqxw2wPBAg"
        RelativePanel.Below="NomAppli"
        RelativePanel.AlignHorizontalCenterWithPanel="True"
        Margin="0, 50, 0, 0"/>
</RelativePanel>
```

Remarque : le titre de la page `Home` est contenu dans le fichier `Ressources.resw`. De même que le nom de l'application.

Nom	Valeur	Commentaire
AppDescription	FilmRatingsApp	
AppDisplayName	Vince's Best Movies	
Settings_About.Text	About this application	
Settings_AboutDescription.Text	Une super App WinUI pour gérer mes films préférés.	
Settings_Personalization.Text	Personalization	
Settings_Theme.Text	Theme	
Settings_Theme_Dark.Content	Dark	
Settings_Theme_Default.Content	Default	
Settings_Theme_Light.Content	Light	
SettingsPage_PrivacyTermsLink.Content	Privacy Statement	
SettingsPage_PrivacyTermsLink.NavigateUri	https://YourPrivacyUrlGoesHere/	
Shell_Film.Content	Film	
Shell_Home.Content	Home	
Shell_Notes.Content	Notes	
Shell_Utilisateur.Content	Utilisateur	

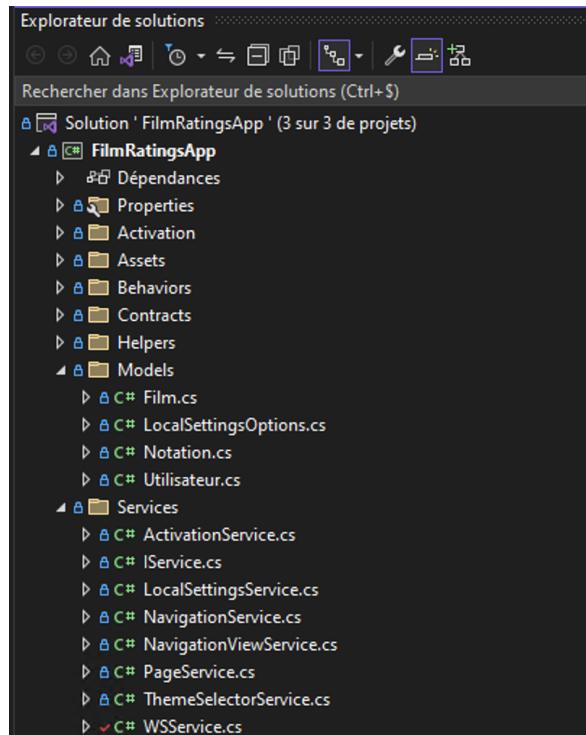
3. Partie « Utilisateur »

3.1. Modèles

Ajouter les 3 classes métier de l'API (`Utilisateur.cs`, `Film.cs`, `Notation.cs`) dans le dossier `Models`. Ne conserver que les properties (sans les annotations). Supprimer le mot clé `virtual` et les `=null!`;

3.2. Web service

Dans le dossier Services, créer la classe WSService et l'interface IService, permettant d'appeler le WS.

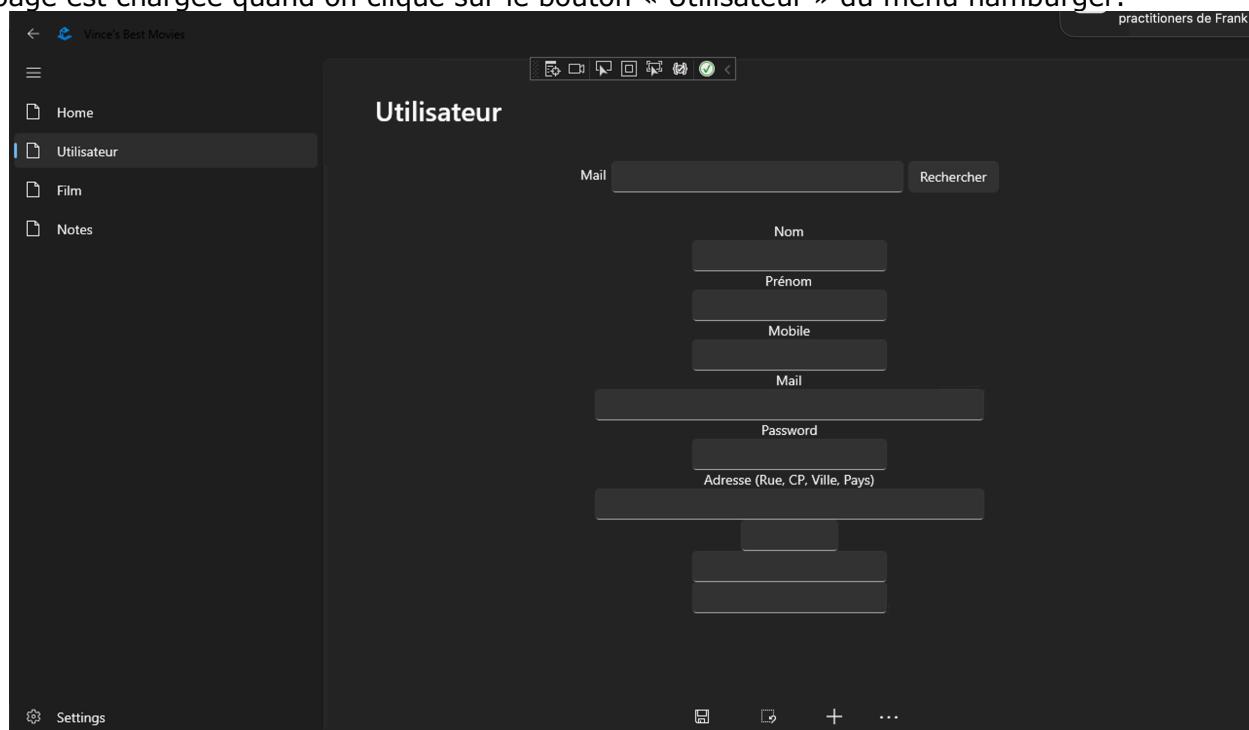


Vous devriez pouvoir récupérer le code créé en TP2.

3.3. CRUD Utilisateur

L'écran suivant (vous pourrez améliorer le visuel) permet d'afficher les informations d'un utilisateur à partir de son email. Il est ensuite possible, si on le souhaite, de modifier les informations affichées puis de les sauvegarder (dans la BD de l'API).

Cette page est chargée quand on clique sur le bouton « Utilisateur » du menu hamburger.



Code XAML (à améliorer) :

```
<RelativePanel x:Name="panelSearch" Margin="0,10,0,10" Orientation="Horizontal" RelativePanel.AlignHorizontalCenterWithPanel="True">
    <TextBlock x:Name="lblSearchMail" Margin="0,4,5,0" Text="Mail"/>
    <TextBox x:Name="txtSearchMail" Margin="0,0,5,0" Text="{Binding SearchMail, Mode=TwoWay}" Width="300"/>
    <Button x:Name="btnRechercher" Command="{Binding BtnSearchUtilisateurCommand}" Content="Rechercher"/>
</StackPanel>
```

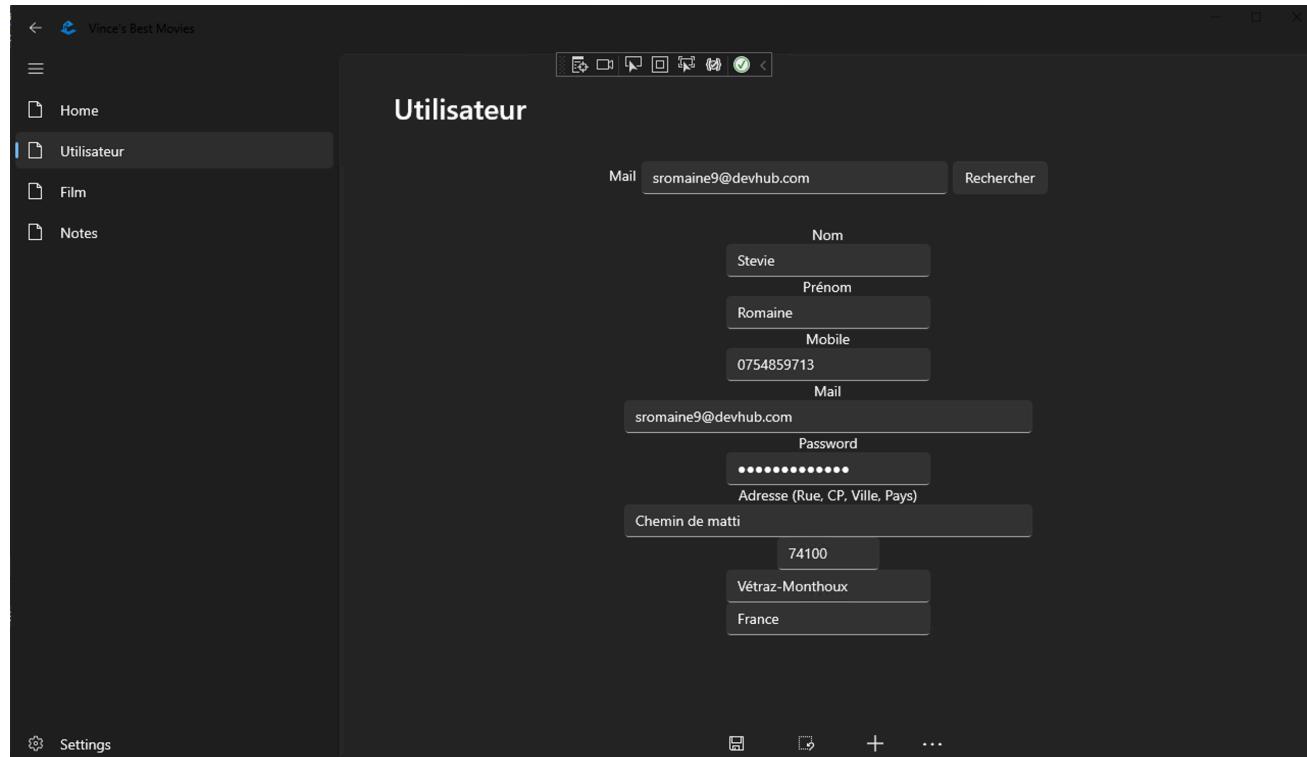
```

<TextBlock x:Name="lblNom" Text="Nom" Margin="0,20,0,0" RelativePanel.AlignHorizontalCenterWithPanel="True" RelativePanel.Below="panelSearch"/>
<TextBox x:Name="txtNom" Text="{Binding UtilisateurSearch.Nom, Mode=TwoWay}" Width="200" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="lblNom"/>
<TextBlock x:Name="lblPrenom" Text="Prénom" RelativePanel.AlignHorizontalCenterWithPanel="True" RelativePanel.Below="txtNom"/>
<TextBox x:Name="txtPrenom" Text="{Binding UtilisateurSearch.Prenom, Mode=TwoWay}" Width="200" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="lblPrenom"/>
<TextBlock x:Name="lblMobile" Text="Mobile" RelativePanel.AlignHorizontalCenterWithPanel="True" RelativePanel.Below="txtPrenom"/>
<TextBox x:Name="txtMobile" Text="{Binding UtilisateurSearch.Mobile, Mode=TwoWay}" Width="200" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="lblMobile"/>
<TextBlock x:Name="lblMail" Text="Mail" RelativePanel.AlignHorizontalCenterWithPanel="True" RelativePanel.Below="txtMobile"/>
<TextBox x:Name="txtMail" Text="{Binding UtilisateurSearch.Mail, Mode=TwoWay}" Width="400" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="lblMail"/>
<TextBlock x:Name="lblPassword" Text="Password" RelativePanel.AlignHorizontalCenterWithPanel="True" RelativePanel.Below="txtMail"/>
<PasswordBox x:Name="txtPassword" Password="{Binding UtilisateurSearch.Pwd, Mode=TwoWay}" Width="200"
RelativePanel.AlignHorizontalCenterWithPanel="True" RelativePanel.Below="lblPassword"/>
<TextBlock x:Name="lblAdresse" Text="Adresse (Rue, CP, Ville, Pays)" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="txtPassword"/>
<TextBox x:Name="txtRue" Text="{Binding UtilisateurSearch.Rue, Mode=TwoWay}" Width="400" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="lblAdresse"/>
<TextBox x:Name="txtCP" Text="{Binding UtilisateurSearch.CodePostal, Mode=TwoWay}" Width="100" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="txtRue"/>
<TextBox x:Name="txtVille" Text="{Binding UtilisateurSearch.Ville, Mode=TwoWay}" Width="200" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="txtCP"/>
<TextBox x:Name="txtPays" Text="{Binding UtilisateurSearch.Pays, Mode=TwoWay}" Width="200" RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.Below="txtVille"/>
<CommandBar x:Name="AppCommandBar" RelativePanel.AlignBottomWithPanel="True" RelativePanel.AlignHorizontalCenterWithPanel="True">
<CommandBar.PrimaryCommands>
<AppBarButton Name="Save"
Icon="Save"
Label="Save Change" Command="{Binding BtnModifyUtilisateurCommand}"/></AppBarButton>
<AppBarButton Name="Clear"
Icon="ClearSelection"
Label="Clear" Command="{Binding BtnClearUtilisateurCommand}"/></AppBarButton>
<AppBarButton Name="Add"
Icon="Add"
Label="Add" Command="{Binding BtnAddUtilisateurCommand}"/></AppBarButton>
</CommandBar.PrimaryCommands>
</CommandBar>
</RelativePanel>

```

Nous verrons un peu plus loin à quoi servent les boutons situés en bas de l'écran...

Clic sur le bouton « Rechercher » :



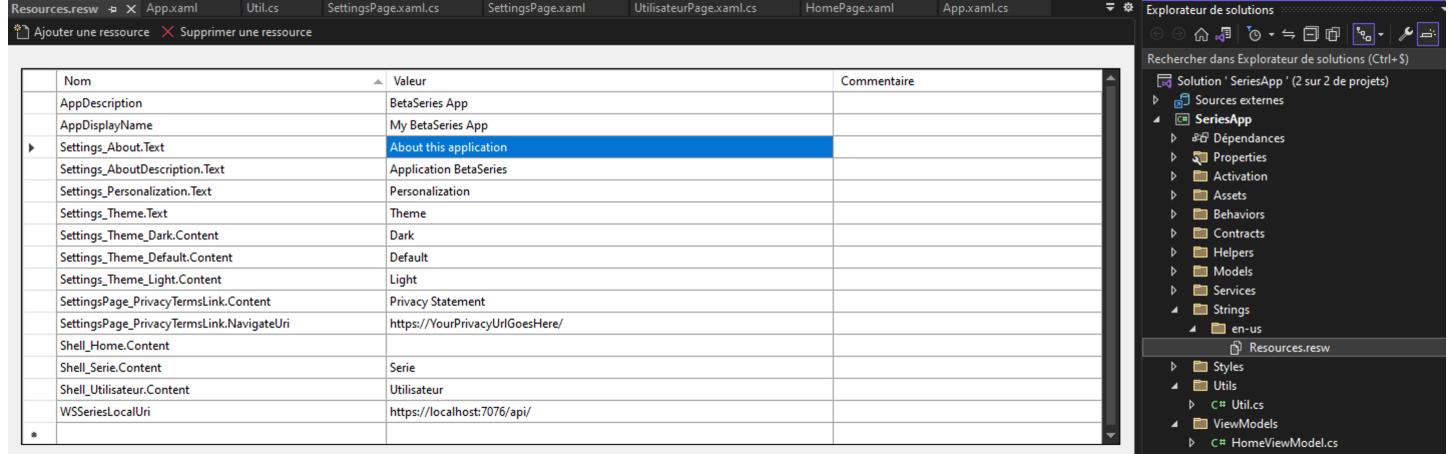
Indications :

Vincent COUTURIER

- Ajouter une méthode dans la classe `WSService` permettant d'appeler la méthode GET correspondant à la recherche d'un utilisateur en fonction de son email (`.../utilisateurs/getbyemail/toto@titi.fr`). Vous pourrez utiliser la méthode `string.Concat` pour construire la chaîne d'appel du WS.
- Vous pourrez stocker le lien vers le WS dans les ressources de l'application. Cf. <https://docs.microsoft.com/fr-fr/windows/uwp/app-resources/localize-strings-ui-manifest>.

```
var loader = new Windows.ApplicationModel.Resources.ResourceLoader();
```

```
string Uri = loader.GetString("MonURI");
```



- En MVVM, chaque `TextBox` sera bindée à un attribut d'une `property` de type `Utilisateur`. Exemple :
`<TextBox x:Name="TxtNom" Text="{Binding UtilisateurSearch.Nom, Mode=TwoWay}" ... />`
- Vous remarquerez que le VM hérite de `ObservableRecipient` :

```
public class UtilisateurViewModel : ObservableRecipient
```

<https://learn.microsoft.com/fr-fr/dotnet/communitytoolkit/mvvm/observablerecipient>

On peut remplacer cet héritage par `ObservableObject` comme en TP2.

- Pour définir le `DataContext` :

```
public UtilisateurPage()
{
    ViewModel = App.GetService<UtilisateurViewModel>();
    this.DataContext = ViewModel;
    InitializeComponent();
}
```

- Pour le mot de passe, nous avons utilisé un contrôle `PasswordBox`. La propriété à binder est `Password`. <https://docs.microsoft.com/fr-fr/windows/uwp/controls-and-patterns/password-box>
<https://learn.microsoft.com/fr-fr/uwp/api/windows.ui.xaml.controls.passwordbox>
- Pour l'affichage des messages PopUp `ContentDialog`, le `MainRoot` sera défini à l'aide d'une `property` dans le fichier `App.xaml.cs` :

```
public static FrameworkElement MainRoot
{
    get
    {
        return MainWindow.Content as FrameworkElement;
    }
}
```

Barre de commandes en bas de la page : en cliquant sur « ... » de la barre de commande, on peut visualiser les libellés des boutons :

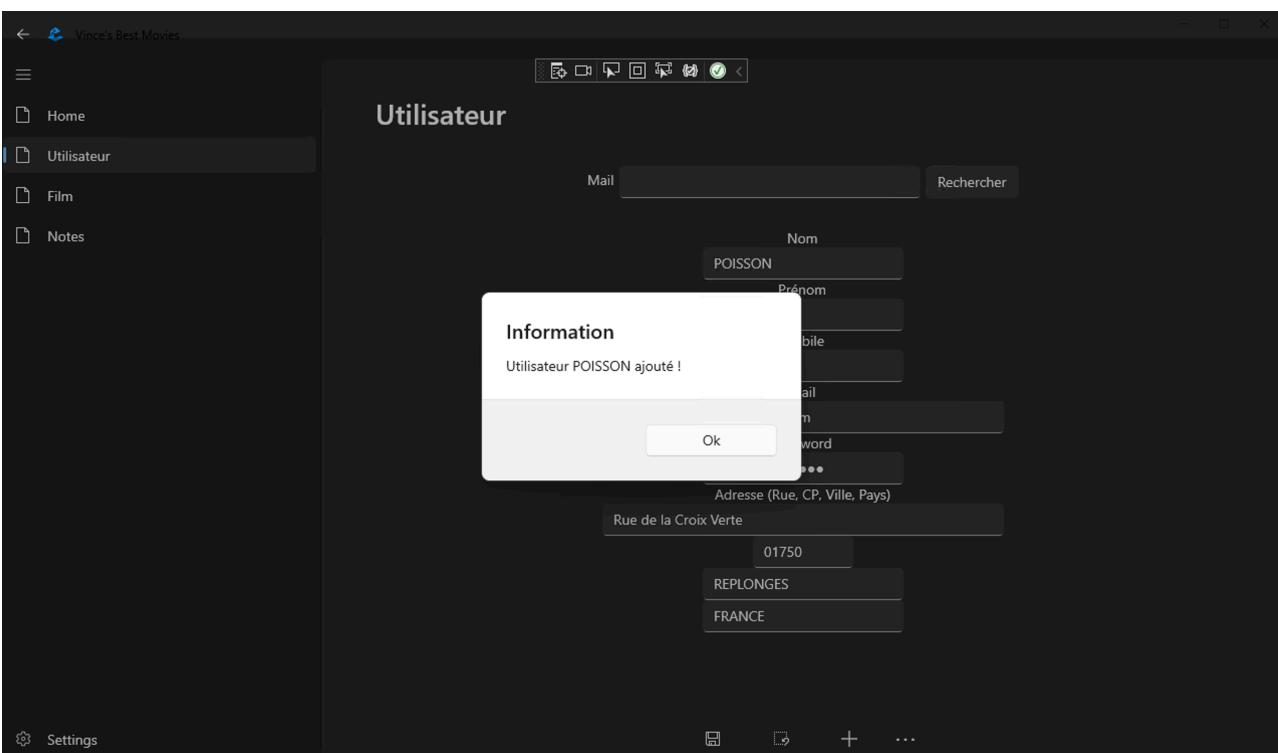
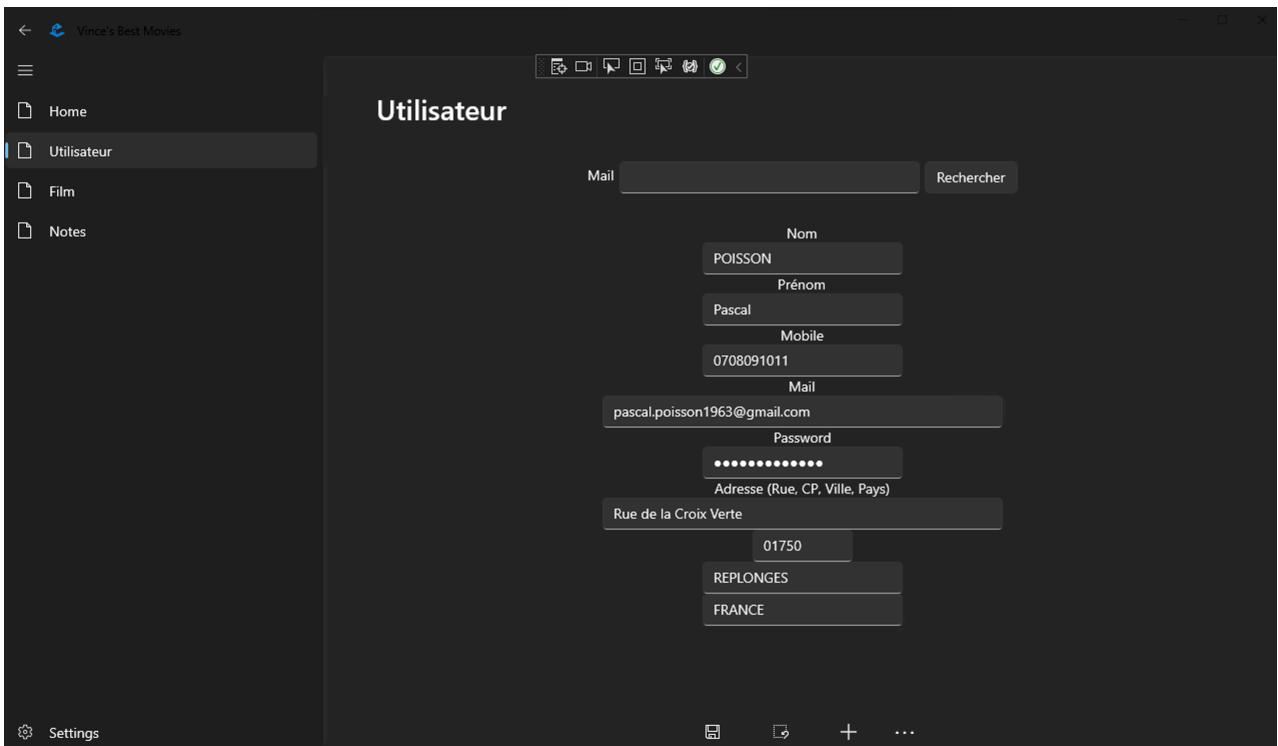
The screenshot shows a user editing interface. On the left is a sidebar with 'Home', 'Utilisateur' (selected), 'Film', and 'Notes'. The main area has a title 'Utilisateur'. It contains fields for 'Nom' (Stevie), 'Prénom' (Romaine), 'Mobile' (0754859713), 'Mail' (sromaine9@devhub.com), 'Password' (redacted), 'Adresse (Rue, CP, Ville, Pays)' (Chemin de matti, 74100, Vétraz-Monthoux, France), and 'Chemin de matti' (74100, Vétraz-Monthoux, France). At the bottom right are buttons for 'Save Change', 'Clear', 'Add', and more.

Le bouton « Save Change » permet de sauvegarder les informations de l'utilisateur (ajouter le code permettant d'appeler la méthode PUT de l'API) après modification des données affichées dans les Textbox.

This screenshot is similar to the first one, showing the user edit interface. However, a modal dialog box titled 'Information' is overlaid on the screen, containing the message 'Utilisateur Stevie modifié !'. A single 'Ok' button is visible in the dialog's footer. The background form fields are partially visible.

Attention, il faudra sortir du champ que vous avez modifié pour valider sa modification.

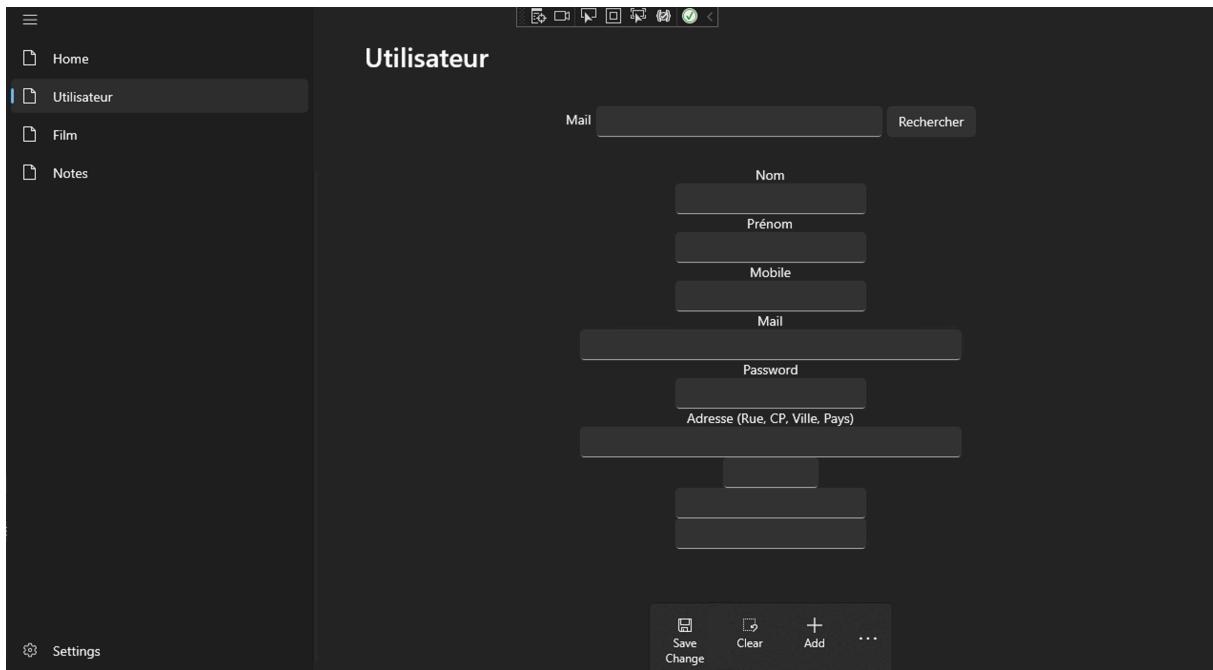
Le bouton « Add » permet d'ajouter un utilisateur dont les informations seront saisies dans le formulaire :



Indications :

- Ajouter une méthode dans la classe `WSService` permettant d'appeler la méthode POST du contrôleur utilisateur.
- **Penser à initialiser l'objet de type Utilisateur dans le constructeur du ViewModel afin d'éviter une référence nulle.**
- Le binding n'est effectif que lorsque l'on sort du champ (attention, donc pour le dernier champ à remplir).

Le bouton « Clear » permet de réinitialiser les champs. Le binding étant fait sur un objet de type `Utilisateur`, il suffira de mettre cet objet à null. Résultat :



Gérer les erreurs.

3.4. Tests unitaires Utilisateur

Ajouter les tests unitaires de l'application cliente.