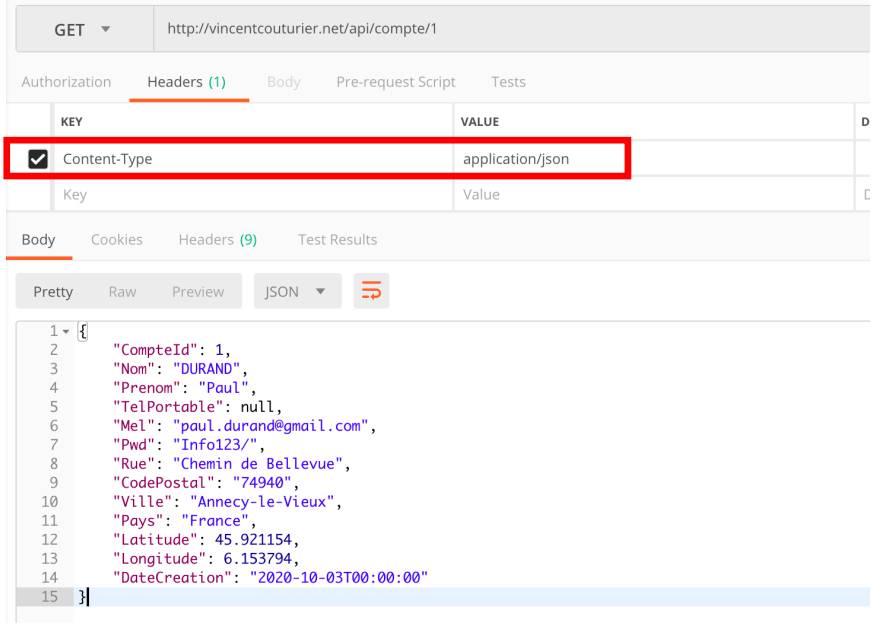
TP3 Partie 3 – Client MVVM

L'objectif de cette partie est juste de réviser la création d'un client MVVM (demandée en DS). Vous pouvez donc juste faire la première partie de ce TP (Sections 3.1 et 3.2b).
Les plus rapides pourront faire tout ce TP ou passer au TP4 - Token JWT, selon ce qu'ils préfèrent.

Pour les développeurs Xamarin :

- Le WS Compte est disponible ici : http://vincentcouturier.net/api/compte
- **Exemples d'appels :**
  http://vincentcouturier.net/api/compte/1 ou
  http://vincentcouturier.net/api/compte?id=1



  http://vincentcouturier.net/api/compte?email=vincent.vivant@gmail.com

  http://vincentcouturier.net/api/film/1 ou http://vincentcouturier.net/api/film?id=1
  http://vincentcouturier.net/api/film?titre=Lucy
- Pour générer la classe Model correspondant au compte, vous pouvez utiliser le site http://json2csharp.com/
- Pour récupérer un Json, ajouter `MonHttpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));`
- Pour le POST :

```
var content = JsonConvert.SerializeObject(MonCompte);
var buffer = System.Text.Encoding.UTF8.GetBytes(content);
var MonCompteACreer = new ByteArrayContent(buffer);
MonCompteACreer.Headers.ContentType = new
MediaTypeHeaderValue("application/json");
var httpResponse = await MonHttpClient.PostAsync("URL ou NomControlleur",
MonCompteACreer);
…
```

  Autre possibilité : https://stackoverflow.com/questions/23585919/send-json-via-post-in-c-sharp-and-receive-the-json-returned
- Installer les package *MvvmLightLibsStd10* et *NewtonSoft.Json*

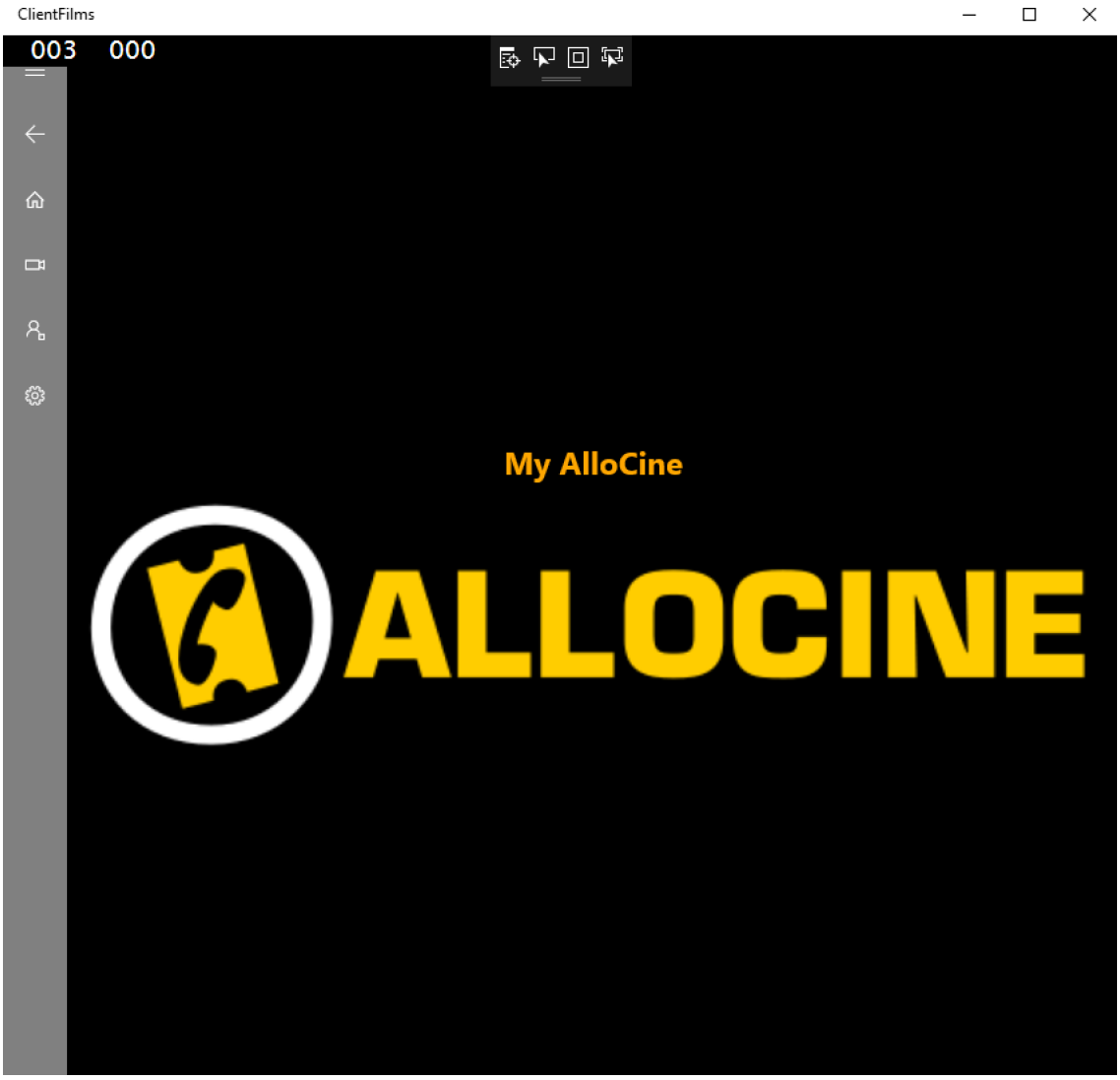
Créer un nouveau projet **Application vide Universal Windows**.

1. ***Installation des packages NuGet***

Installer les packages « MvvmLight » et « Microsoft.AspNet.WebApi.Client ».

2. ***Page d'accueil***

Visuel indicatif de la page d'accueil, qui sera modifiée par la suite :



Au démarrage de l'application, une page `HomePage` est chargée. Celle-ci contient, par exemple, un lien vers l'image « AlloCine » https://www.citationbonheur.fr/wp-content/uploads/2018/05/logo-allocine.png.
Code XAML de la `HomePage` :

```
<RelativePanel>
        <Image x:Name="logoAlloCine" Source="https://www.citationbonheur.fr/wp-
content/uploads/2018/05/logo-allocine.png"
RelativePanel.AlignHorizontalCenterWithPanel="True"
RelativePanel.AlignVerticalCenterWithPanel="True"/>
        <TextBlock x:Name="NomAppli" Text="My AlloCine"
RelativePanel.Above="logoAlloCine" RelativePanel.AlignHorizontalCenterWithPanel="True"
FontWeight="Bold" FontSize="24" Foreground="Orange"/>
</RelativePanel>
```

Code XAML de la `RootPage` (contenant le menu Hamburger) :

```
<SplitView x:Name="MySplitView" DisplayMode="CompactOverlay"  IsPaneOpen="False"
              CompactPaneLength="50" OpenPaneLength="180" Background="{ThemeResource
ApplicationPageBackgroundThemeBrush}">
        <SplitView.Pane>
            <StackPanel Background="Gray">
                <Button x:Name="HamburgerButton" FontFamily="Segoe MDL2 Assets"
Content="&#xE700;" Width="50" Height="50" Background="Transparent"/>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="Back" FontFamily="Segoe MDL2 Assets"
Content="&#xE72B;" Width="50" Height="50" Background="Transparent"/>
                    <TextBlock Text="Précédent" FontSize="18" VerticalAlignment="Center"
/>
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="Home" FontFamily="Segoe MDL2 Assets"
Content="&#xE80F;" Width="50" Height="50" Background="Transparent"/>
                    <TextBlock Text="Home" FontSize="18" VerticalAlignment="Center" />
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="Movies" FontFamily="Segoe MDL2 Assets"
Content="&#xE714;" Width="50" Height="50" Background="Transparent"/>
                    <TextBlock Text="Films" FontSize="18" VerticalAlignment="Center" />
```

```
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="Account" FontFamily="Segoe MDL2 Assets"
Content="&#xE8CF;" Width="50" Height="50" Background="Transparent"/>
                    <TextBlock Text="Compte" FontSize="18" VerticalAlignment="Center" />
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="Settings" FontFamily="Segoe MDL2 Assets"
Content="&#xE713;" Width="50" Height="50" Background="Transparent"/>
                    <TextBlock Text="Paramètres" FontSize="18" VerticalAlignment="Center"
/>
                </StackPanel>
            </StackPanel>
        </SplitView.Pane>
        <SplitView.Content>
            <Grid>
            </Grid>
        </SplitView.Content>
</SplitView>
```

Code du bouton « Précédent » :

```
Frame myFrame = this.MySplitView.Content as Frame;
if (myFrame.CanGoBack)
{
      myFrame.GoBack();
}
```

Code du bouton « Hamburger » : cf. TP1.

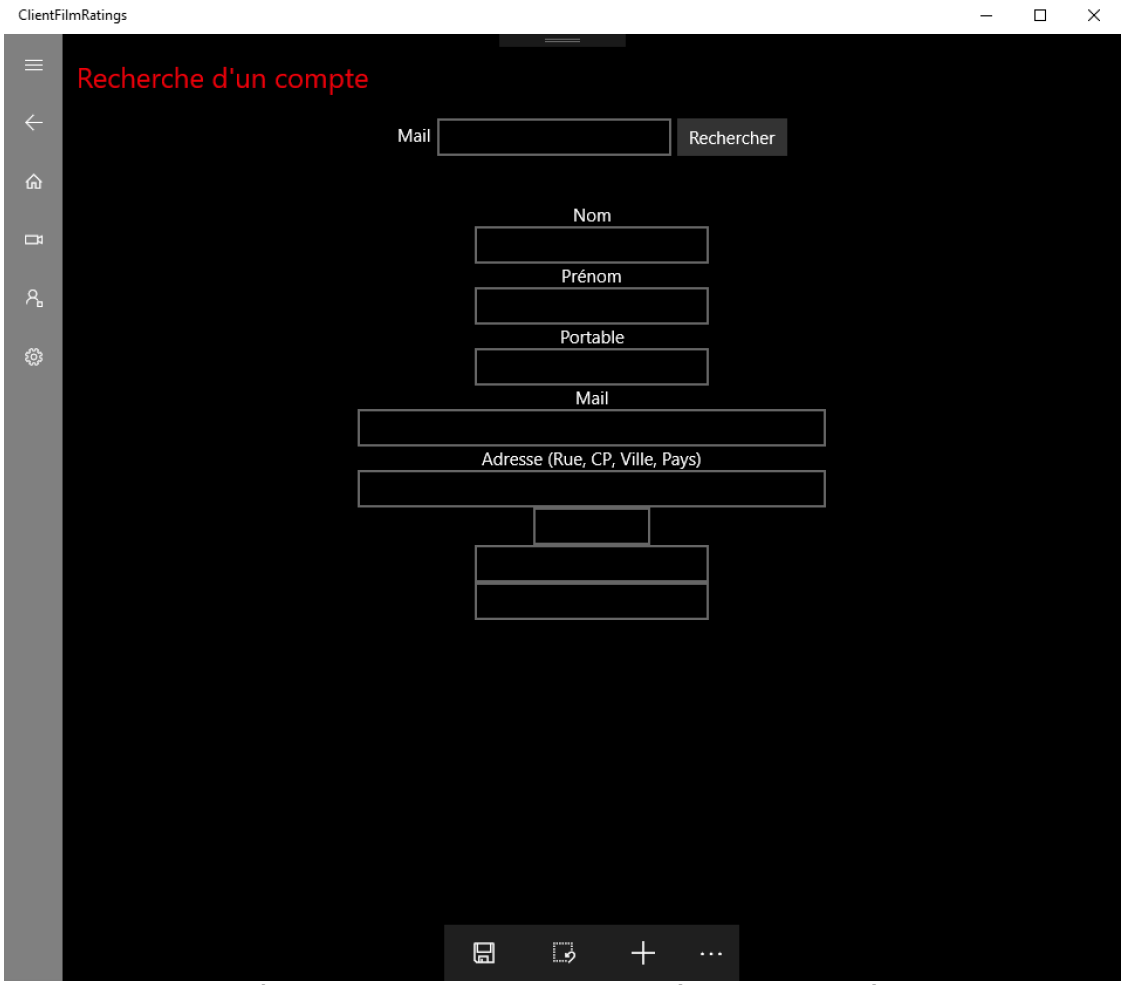
3. ***Partie « Compte »***

Ajouter les 4 classes métier du WS (`Compte.cs`, `Film.cs`, `Favori.cs`) dans un dossier `Models`. Ne conserver que les properties (sans les annotations).

Dans un dossier `Services`, créer la classe `WSService`, permettant d'appeler le WS.

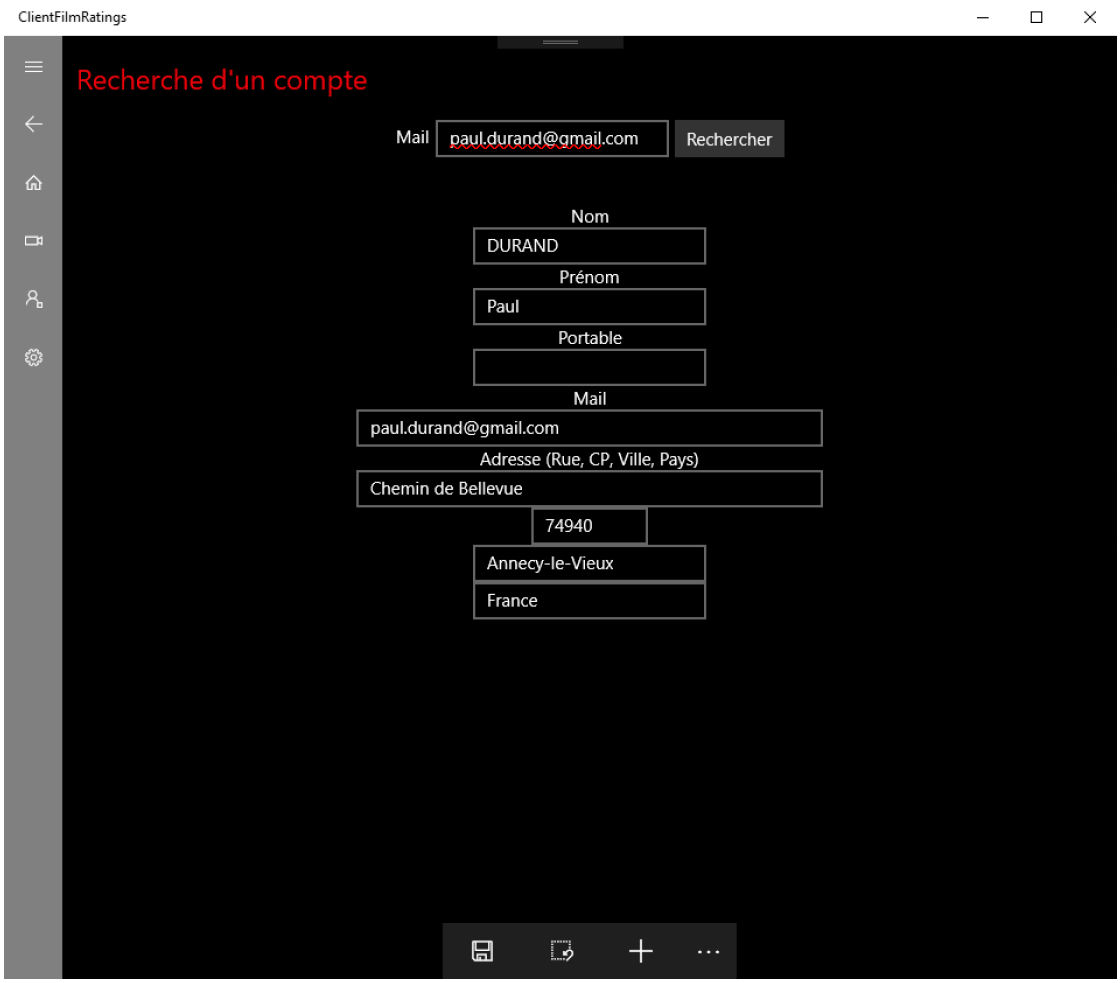
3.1. Recherche d'un compte (utilisateur) et modification

L'écran suivant (vous pourrez améliorer le visuel) permet d'afficher les informations d'un compte utilisateur à partir de son email. Il est ensuite possible, si on l'utilisateur le souhaite, de modifier les informations affichées puis de les sauvegarder (dans la BD du WS).
Cette page est chargée quand on clique sur le bouton « Compte » du menu hamburger.



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

Clic sur le bouton « Rechercher » :



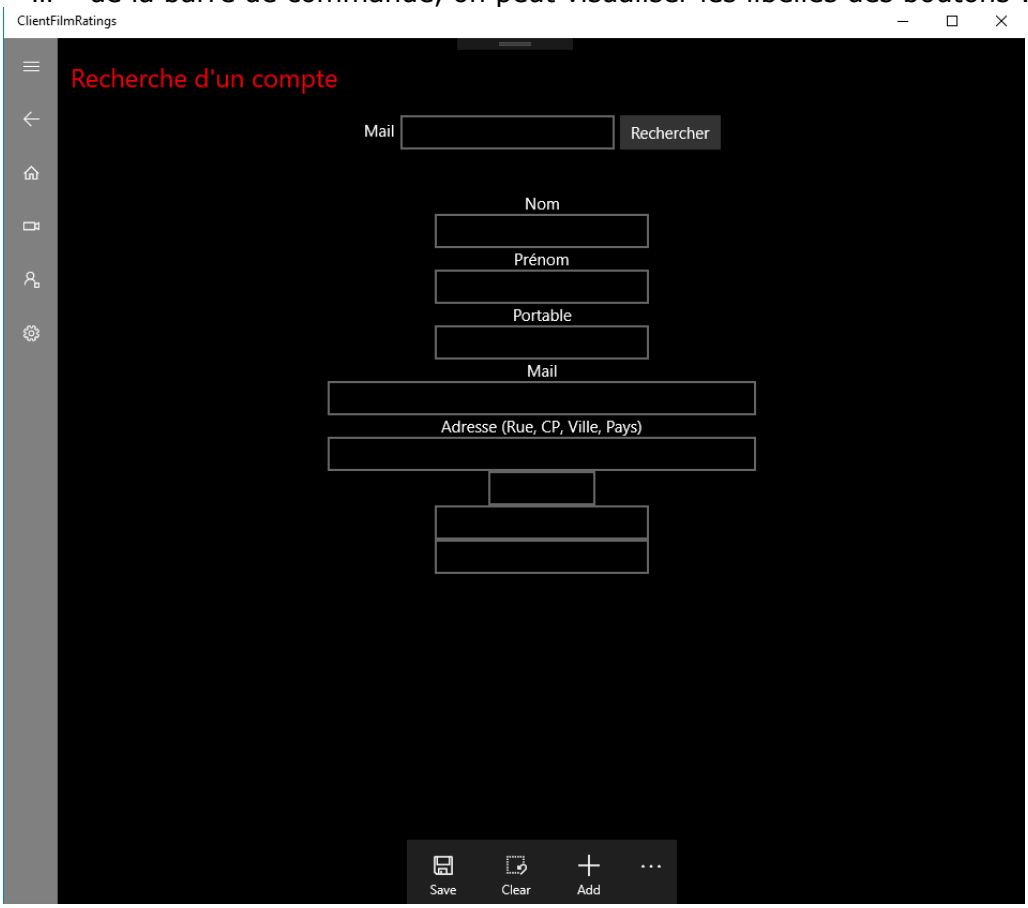
Indications :

- Ajouter une méthode dans la classe `WSService` permettant d'appeler la méthode GET correspondant à la recherche d'un compte en fonction de l'email de l'utilisateur (…/`compte/getbyemail/toto@titi.fr`). Vous pourrez utiliser la méthode `string.Concat` pour construire la chaine 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. Dans ce cas, créer un dossier `fr-FR` et non `en-US` dans le dossier `Strings`.
- En MVVM, chaque Textbox sera bindée à un attribut d'une property de type `Compte`. Exemple : `<TextBox x:Name="TxtNom" Text="{Binding CompteSearch.Nom, Mode=TwoWay}" … />`

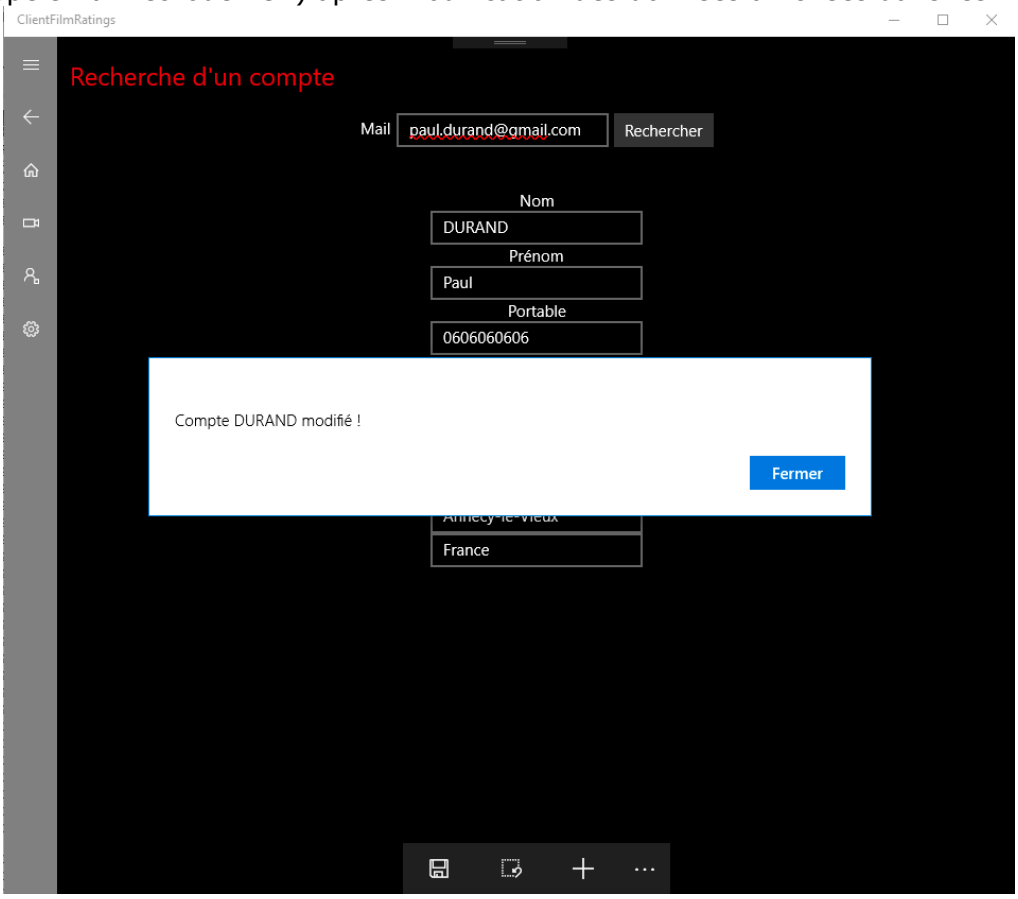
Barre de commandes en bas de la page :

```
      <CommandBar x:Name="AppCommandBar" RelativePanel.AlignBottomWithPanel="True"
      RelativePanel.AlignHorizontalCenterWithPanel="True">
            <CommandBar.PrimaryCommands>
                <AppBarButton Name="Save"
                              Icon="Save"
                              Label="Save" Command="{Binding
BtnModifyCompteCommand}"></AppBarButton>
                <AppBarButton Name="Clear"
                              Icon="ClearSelection"
                              Label="Clear" Command="{Binding
BtnClearCompteCommand}"></AppBarButton>
                <AppBarButton Name="Add"
                              Icon="Add"
                              Label="Add" Command="{Binding
BtnAddCompteCommand}"></AppBarButton>
            </CommandBar.PrimaryCommands>
        </CommandBar>
```

En cliquant sur « … » de la barre de commande, on peut visualiser les libellés des boutons :



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

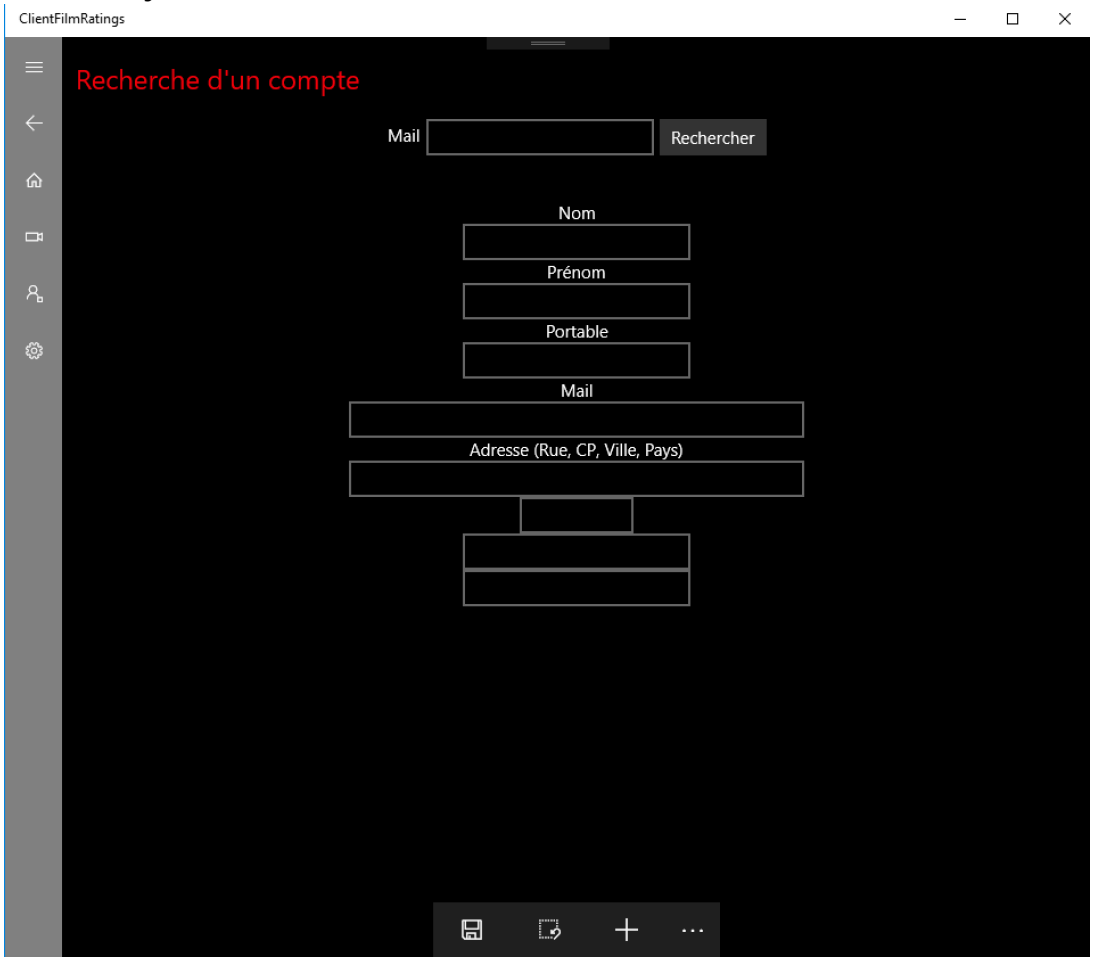


Attention, il faudra sortir du champ que vous avez modifié pour valider sa modification.

Le bouton « Add » permet d'ajouter un compte (Cf. section suivante). Code pour naviguer vers la page de création d'un compte :

```
RootPage r = (RootPage)Window.Current.Content;
SplitView sv = (SplitView)(r.Content);
(sv.Content as Frame).Navigate(typeof(AddComptePage));
```

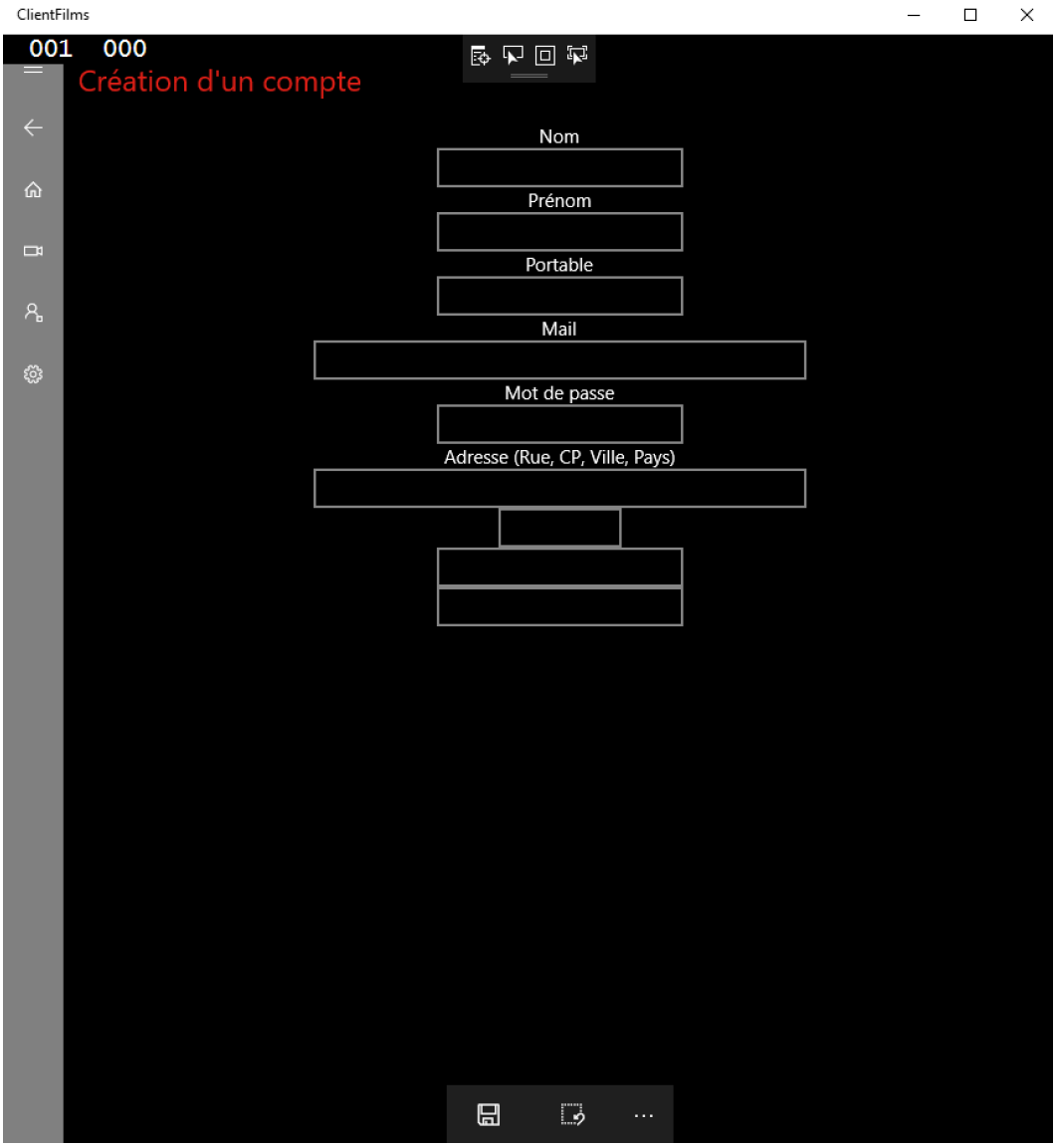
Le bouton « Clear » permet de réinitialiser les champs. Le binding étant fait sur un objet de type `Compte`, il suffira de mettre cet objet à `null`. Résultat :



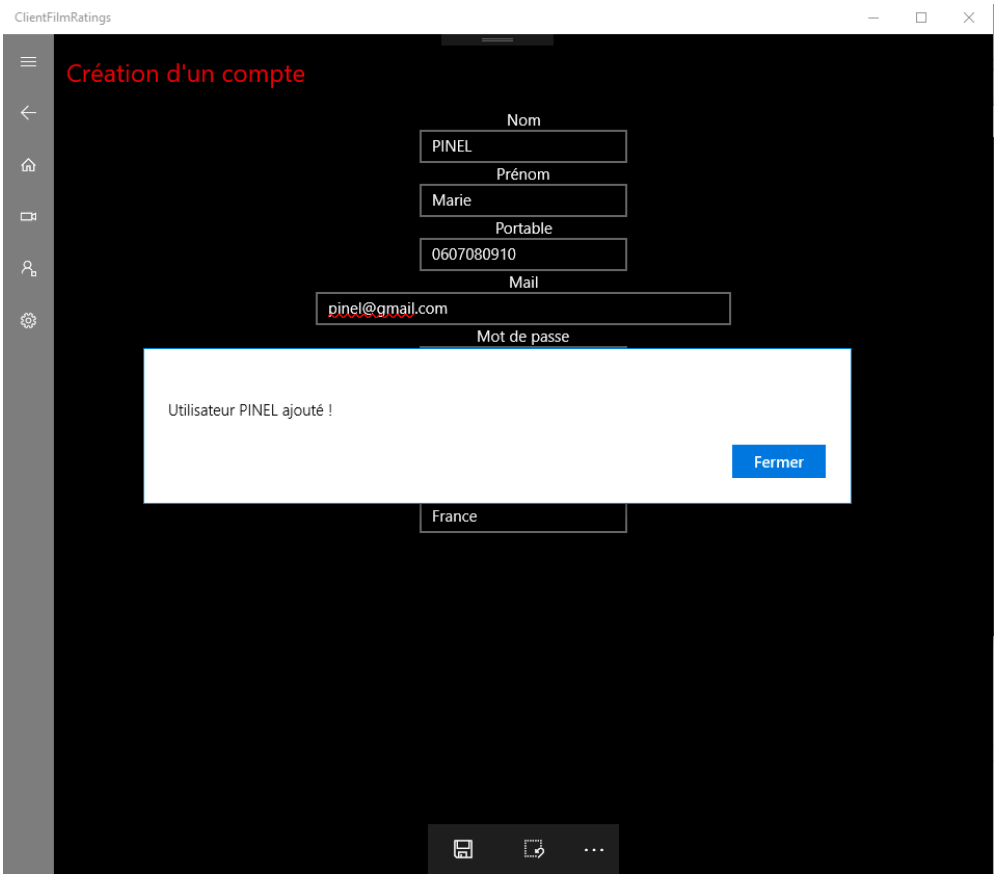
3.2. *Ajout d'un compte*

Ajouter une méthode dans la classe `WSService` permettant d'appeler la méthode POST du contrôleur compte. Utiliser la méthode `PostAsJsonAsync` de la classe `HttpClient`.

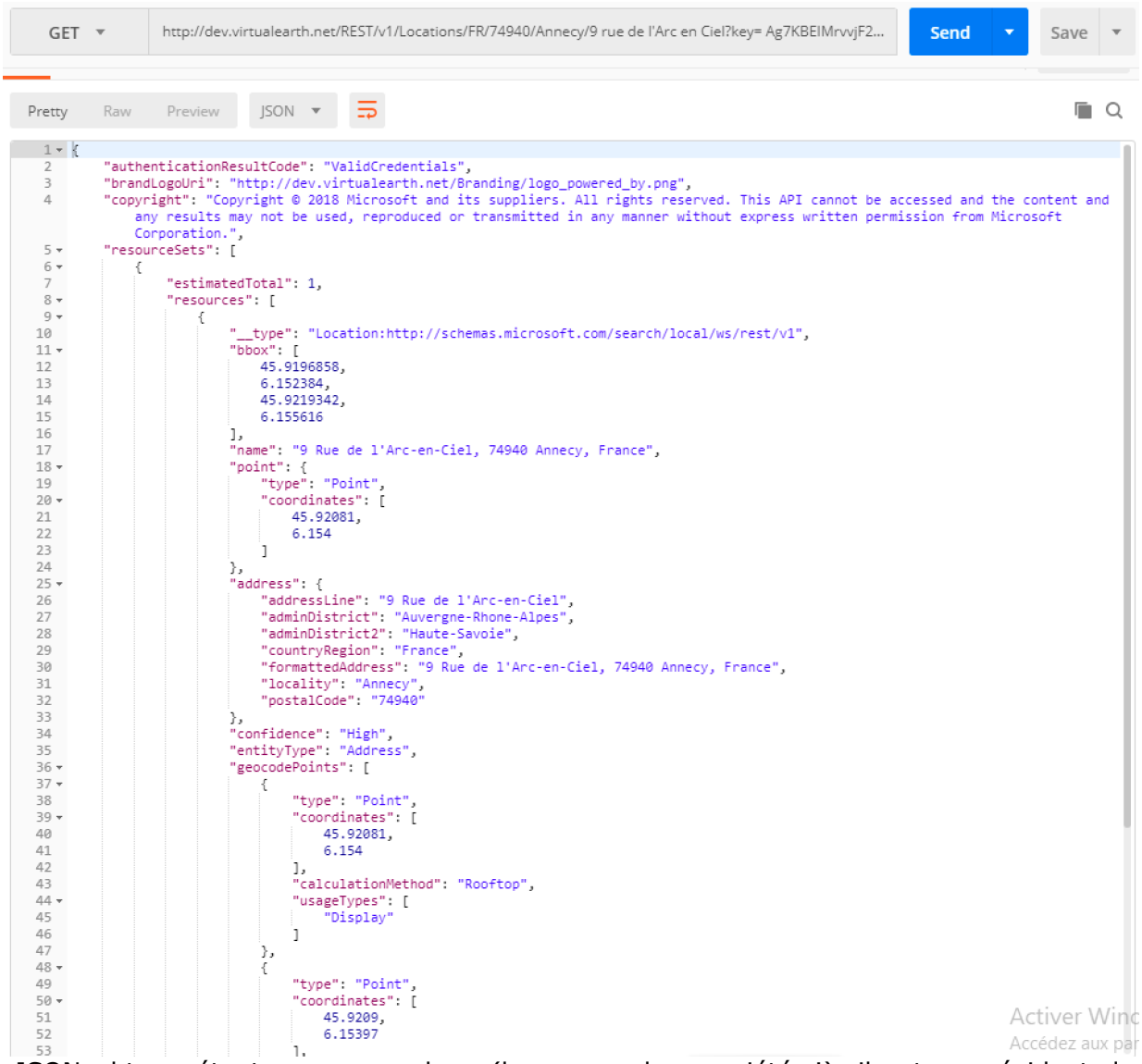
Exemple de visuel (**à retravailler !!!**) :



a. Créer d'abord le code permettant d'ajouter un compte (sans calcul des coordonnées GPS).
   Indications :
   - **Penser à initialiser l'objet de type `Compte` 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).
   - Pour le mot de passe, vous utiliserez un contrôle `PasswordBox`. La propriété à binder est `Password`. https://docs.microsoft.com/fr-fr/windows/uwp/controls-and-patterns/password-box

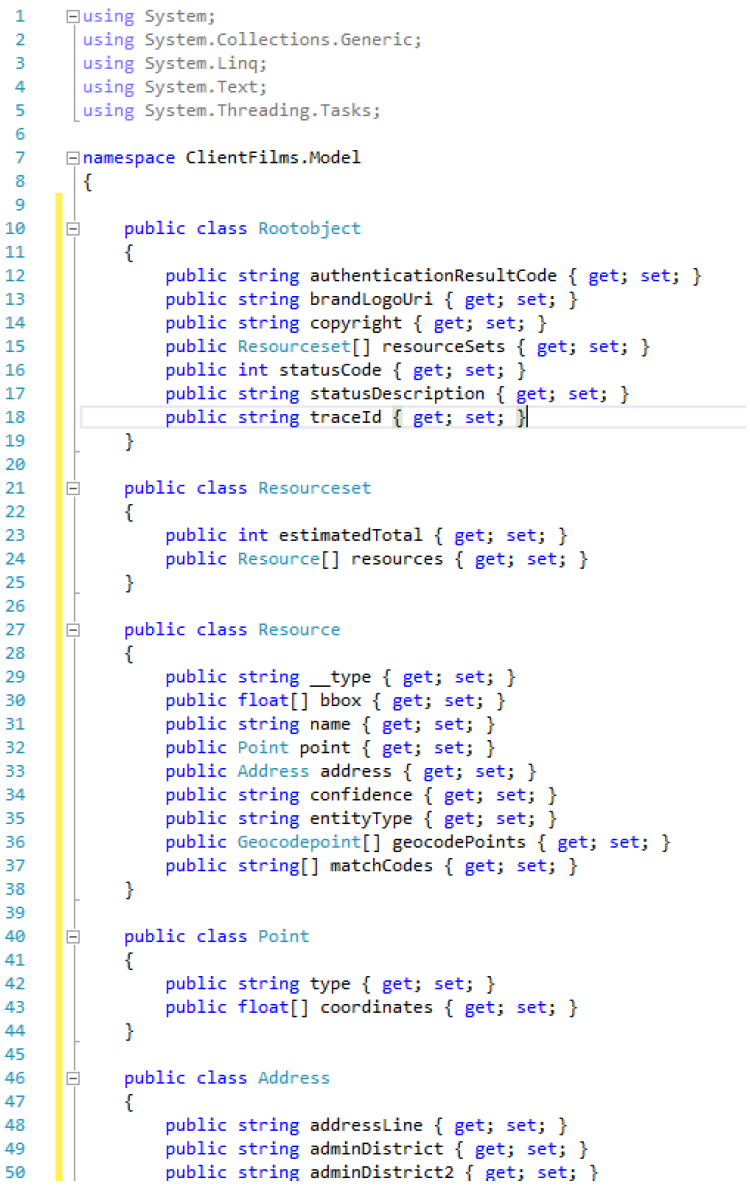


b. Modifier votre code d'ajout d'un compte afin de sauvegarder également les coordonnées GPS de l'adresse. Utiliser le WS Bing Maps afin de récupérer les coordonnées GPS de l'utilisateur saisi.
Indications :
   - Ajouter une nouvelle classe dans le dossier `Services` permettant d'utiliser le WS Bing Maps.
   - Clé Bing Maps : `Ag7KBEIMrvvjF2Kpz9Ze9UaNNoj1jkizmw-_bxWFpRaLJEXzBGNW-IFl4aHj5jd1`. Pour en générer une, voir Annexe.
   - Principes de l'appel au WS Rest Bing Maps et de la réponse obtenue : http://msdn.microsoft.com/en-us/library/ff701714.aspx
   - Les appels au WS sont différents selon les pays. Pour la France (nous ne gérerons que la France), il faut concaténer une chaine de caractères contenant le code pays (FR), le CP, la ville, la rue (adresse ligne 1) et la clé d'activation. Exemple d'appel :
     http://dev.virtualearth.net/REST/v1/Locations/FR/74940/Annecy/9 rue de l'Arc en Ciel?key= Ag7KBEIMrvvjF2Kpz9Ze9UaNNoj1jkizmw- bxWFpRaLJEXzBGNW-IFl4aHj5jd1
   - Tester l'appel précédent dans Postman. Vous obtiendrez le fichier suivant :



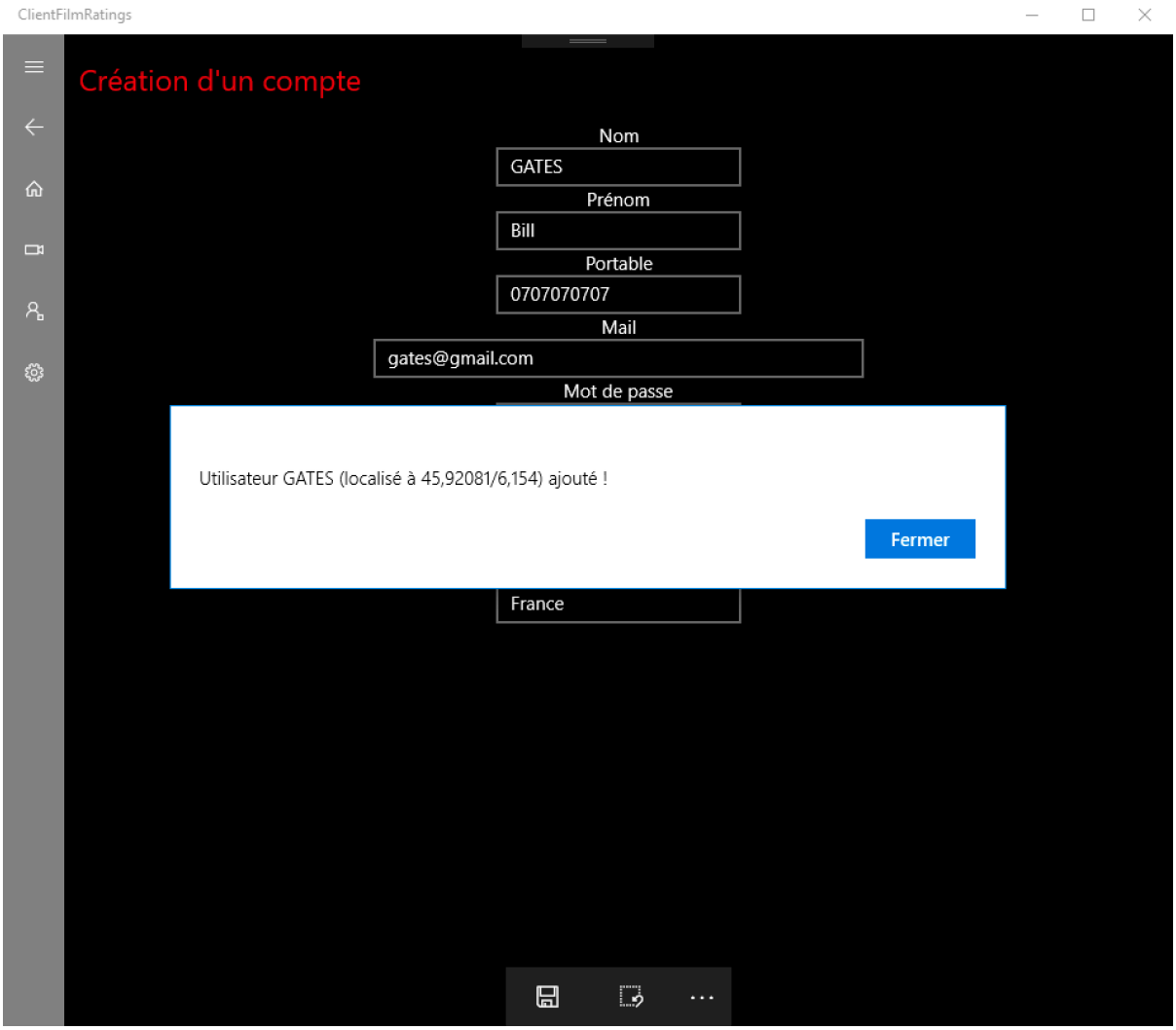

```
{
    "authenticationResultCode": "ValidCredentials",
    "brandLogoUri": "http://dev.virtualearth.net/Branding/logo_powered_by.png",
    "copyright": "Copyright © 2018 Microsoft and its suppliers. All rights reserved. This API cannot be accessed and the content and any results may not be used, reproduced or transmitted in any manner without express written permission from Microsoft Corporation.",
    "resourceSets": [
        {
            "estimatedTotal": 1,
            "resources": [
                {
                    "__type": "Location:http://schemas.microsoft.com/search/local/ws/rest/v1",
                    "bbox": [
                        45.9196858,
                        6.152384,
                        45.9219342,
                        6.155616
                    ],
                    "name": "9 Rue de l'Arc-en-Ciel, 74940 Annecy, France",
                    "point": {
                        "type": "Point",
                        "coordinates": [
                            45.92081,
                            6.154
                        ]
                    },
                    "address": {
                        "addressLine": "9 Rue de l'Arc-en-Ciel",
                        "adminDistrict": "Auvergne-Rhone-Alpes",
                        "adminDistrict2": "Haute-Savoie",
                        "countryRegion": "France",
                        "formattedAddress": "9 Rue de l'Arc-en-Ciel, 74940 Annecy, France",
                        "locality": "Annecy",
                        "postalCode": "74940"
                    },
                    "confidence": "High",
                    "entityType": "Address",
                    "geocodePoints": [
                        {
                            "type": "Point",
                            "coordinates": [
                                45.92081,
                                6.154
                            ],
                            "calculationMethod": "Rooftop",
                            "usageTypes": [
                                "Display"
                            ]
                        },
                        {
                            "type": "Point",
                            "coordinates": [
                                45.9209,
                                6.15397
                            ],
```


- Le JSON obtenu étant assez complexe (beaucoup de propriétés !), il est peu évident de le transformer en classe métier. Visual Studio intègre un outil qui va nous permettre de le faire simplement. Pour cela, ajouter une classe métier (dans `Models`) nommée `Rootobject.cs`. Supprimer le code `class Rootobject {}`. Se positionner entre les accolades {} du namespace, puis *Menu Edition > Collage spécial > Coller le code du JSON en tant que classe*. Vous obtiendrez alors ceci :



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClientFilms.Model
{

    public class Rootobject
    {
        public string authenticationResultCode { get; set; }
        public string brandLogoUri { get; set; }
        public string copyright { get; set; }
        public Resourceset[] resourceSets { get; set; }
        public int statusCode { get; set; }
        public string statusDescription { get; set; }
        public string traceId { get; set; }
    }

    public class Resourceset
    {
        public int estimatedTotal { get; set; }
        public Resource[] resources { get; set; }
    }

    public class Resource
    {
        public string __type { get; set; }
        public float[] bbox { get; set; }
        public string name { get; set; }
        public Point point { get; set; }
        public Address address { get; set; }
        public string confidence { get; set; }
        public string entityType { get; set; }
        public Geocodepoint[] geocodePoints { get; set; }
        public string[] matchCodes { get; set; }
    }

    public class Point
    {
        public string type { get; set; }
        public float[] coordinates { get; set; }
    }

    public class Address
    {
        public string addressLine { get; set; }
        public string adminDistrict { get; set; }
        public string adminDistrict2 { get; set; }
```

- Les latitude et longitude sont contenues dans le tableau `coordinates` de type `float[]` (classe `Point`). Le 1er élément du tableau correspond à la latitude, le 2nd à la longitude. Dans un souci de simplification, on considérera qu'il n'y a qu'un seul élément dans le tableau `resourceSets`, ainsi que dans le tableau `resources`.
- Vous appliquerez le pattern Singleton.
- Vous pourrez stocker le lien vers le WS dans les ressources de l'application.



c. Gérer les erreurs.

4. ***Modification de l'application cliente lourde (POUR LES PLUS RAPIDES)***

a. Crypter le mot de passe. Cf. https://docs.microsoft.com/en-us/windows/uwp/security/macs-hashes-and-signatures
Le code créé devra être réutilisable !
ATTENTION, en fonction de l'algorithme choisi, vous devrez peut-être modifier la taille de la colonne `CPT_PWD` dans la base de données (128 octets pour du SHA512, par exemple). Dans ce cas, il faudra créer une nouvelle migration et l'appliquer.

b. Réaliser les fonctionnalités suivantes pour le film (coder le contrôleur du WS et l'application cliente) :
   - Recherche d'un film (recherche approximative sur le nom pouvant ramener plusieurs résultats).
   - Visualisation d'un film (sélectionné après avoir effectué la recherche).

c. Gérer les favoris.
   - A partir de la fiche de visualisation d'un film, il sera possible de le stocker en local (dans une base SQLite) en tant que favori. Seront enregistrés localement, l'ID du film, son titre, ainsi que la date de mise en favori.
   - Si le film est en favori, il sera possible sur la fiche de visualisation de celui-ci de le supprimer de ses favoris.
   - SQLite avec UWP :
     - https://blogs.windows.com/buildingapps/2017/02/06/using-sqlite-databases-uwp-apps/#yxt7kXMaeom0hwzi.97
     - Types SQLite : https://www.sqlite.org/datatype3.html
     - Fonctions Date & Time : https://sqlite.org/lang_datefunc.html

5. ***Annexe***

Clé Bing Maps : https://www.bingmapsportal.com/