HMIN309 - Développement logiciel pour l'embarqué

TP 1 - Introduction à Xamarin.forms

Pré-requis

Ce TP fait suite au précédant TP 1 « introduction à la plateforme Xamarin ». Avant de commencer, récupérer le code du *projet partagé* **TP1-ProjetPartagé-init.zip** sur l'ENT.

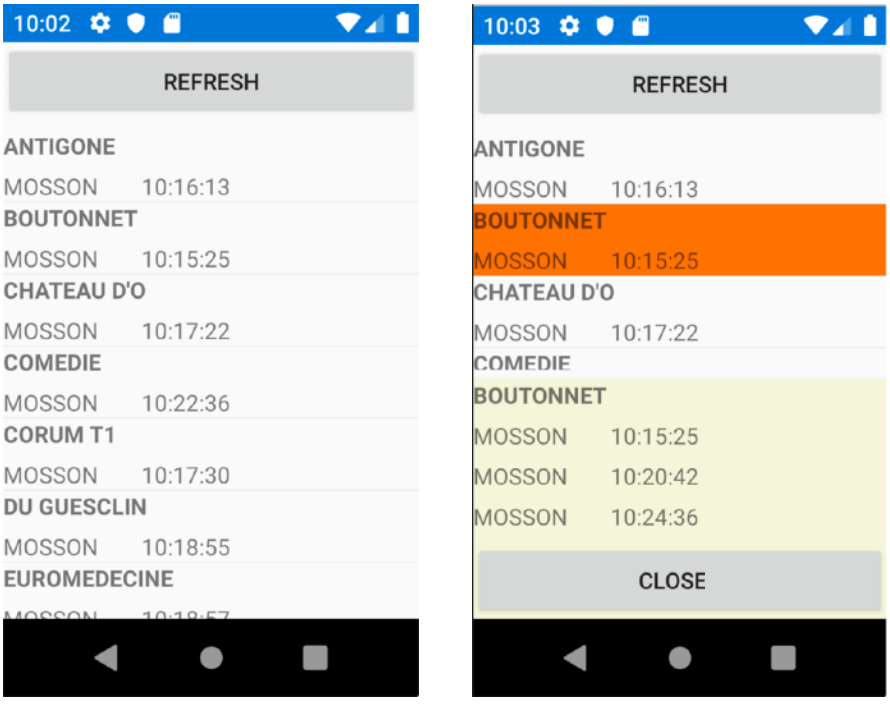
Dans ce TP, nous allons concevoir l'interface graphique d'une application mobile (iOS ou Android) qui affichera en temps réel les horaires de TRAM du service TAM de Montpellier.

Mise en place

Ouvrez Visual Studio et créez un nouveau projet. Sélectionnez *Multiplateforme* > *Application* > *Application Forms vide* et appuyez sur *suivant*. Nommez votre application TP2. Veillez à bien sélectionnez comme plateformes cible *Android* et *iOS*. Afin coché l'option *Utiliser une bibliothèque partagée*. Appuyez *suivant*, indiquez l'emplacement de votre nouveau projet et appuyez *Créer.*

Votre nouveau projet (ou *solution*) est dispose maintenant de trois projets :
- **TP2** : le projet qui va contenir le code commun de nos applications, et l'interface graphique.
- **TP2.Android** : le projet qui contient le code spécifique à l'application Android.
- **TP2.iOS** : le projet qui contient le code spécifique à l'application iOS.

Récupérez le projet partagé **TP1-ProjetPartage** (ENT) et ajoutez le à votre solution (*clique droit* > *ajouter…* > *ajouter projet existant*). N'oubliez pas d'ajouter également les références vers le projet **TP1-ProjetPartage** dans les projets iOS et Android.



a **b**

Figure 1 - Interface Graphique de l'application cliente

Exercice 1 - Modéliser de l'interface graphique

Contrairement à Xamarin, *Xamarin.forms* permet de partager le code de l'interface graphique entre nos différentes applications, en plus du code de la logique métier. Dans ce TP, nous allons continuer sur code de la séance précédente en concevant l'interface graphique associé.

Question 1 : La figure 1 présente l'interface graphique de notre client mobile. La partie **b** de la figure montre l'action qui est réalisé lorsque que l'on appuie sur le nom d'un arrêt de tram. Décrivez pour cette interface la liste des éléments graphiques utilisés que vous observez.

Question 2 : Décrivez la relation hiérarchique en entre tous ces éléments, et imaginez ainsi quels layouts ont pu être utilisés pour mettre les éléments en relation.

Exercice 2 - Récupération des données en Temps Réel

URLs pour télécharger les données avec votre application :
- **Officiel** : http://data.montpellier3m.fr/sites/default/files/ressources/TAM_MMM_TpsReel.csv
- **Snapshot** : https://gite.lirmm.fr/hlad/hmin309-tp1-xamarin/raw/master/horaireTram.csv
- **Petit snapshop**: https://gite.lirmm.fr/hlad/hmin309-tp1-xamarin/raw/master/smallHoraire.csv

Nous allons récupérer en temps réel les données de le *TAM* pour afficher les horaires de passages des trois prochains trams, pour chaque arrêt.

Question 3 : Dans la classe **MainPage**, Créez deux propriétés tableaux :
- **_trams** de type `ObservableCollection<TP1ProjetPartage.TamSchedule>`
- **Trams** de type public `ObservableCollection<TP1ProjetPartage.TamSchedule>`

Les Objets Observables vont notifier automatiquement l'UI pour demander une mise à jour du contenu à afficher à chaque fois que ceux-ci sont modifier.

Question 4 : Créez dans la classe **MainPage.xaml.cs** une méthode **GetData()** en suivant le modèle :

```
private void GetData()
{

    string uri = "https://gite.lirmm.fr/hlad/hmin309-tp1-xamarin/raw/master/horaireTram.csv";
    //string uri = "http://data.montpellier3m.fr/sites/default/files/ressources/TAM_MMM_TpsReel.csv";

    HttpWebRequest request = WebRequest.CreateHttp(uri);

    TamScheduleManager manager;
    request.BeginGetResponse((arg) =>
    {
        // Deserialize CSV file.
        Stream stream = request.EndGetResponse(arg).GetResponseStream();

        // completez ici avec les classe CsvManager et TamScheduleManager
        // pour deserialiser le Stream (voir fichiers CsvManager.cs et TamSchedule.cs)
        //...

        //utilisez ici la mehtode download CSVFile
        //...

        // Pour chaque TamSchedule extrait du fichier CSV, on l'ajoute dans la liste des _trams
        // completez ici avec une boucle for pour ajouter les élements renvoyer à votre liste _trams.
        //...
    }, null);

}
```

BeginGetResponse(…) permet de créer un thread asynchrone en parallèle au thread principal pour télécharger des données. Vous allez tester cette méthode juste après.

Question 4 : Créez dans votre fichier MainPage.xaml le bouton **Refresh** (positionnez le en haut de la page) et associez lui un action une fonction `private void RefreshData(object sender, EventArgs e)` qui appel la fonction **getData().** Ajoutez dans la méthode **getData()** une instruction `Console.WriteLine(t.TramStop).`
Avec `t` un élément **TamSchedule** de la tables **_trams**. Cette instruction permet d'afficher la liste des arrêts de trams dans la console de Visual Studio.

Exercice 3 - ListView

Question 5 : Dans le xaml, créer un ListView, nommez la **MyTramView** et associez lui dans l'*ItemSources* par *Binding* votre propriété **Trams**. Créez ensuite à l'intérieur de *DataTemplate* un *Label* que vous allez mettre en *Binding* avec la propriété **TramStop** de la classe **TamSchedule.**

```
<ListView x:Name="MyTramView" ItemsSource="{Binding XXXX}" >
    <ListView.ItemTemplate>
        <DataTemplate>
            <Label Text="{Binding XXXXX}" />
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Figure 2 - Indication pour la création d'une listView

Question 6 : Dans le constructeur de **MainPage**, indiquez que l'*ItemSource* de **MyTramView** est **_trams**. Appelez aussi dans le constructeur la méthode **GetData()**.

Question 7 : Testez votre application (Android ou iOS). La liste se télécharge-t-elle ? Affichez-vous correctement la liste dans votre ListView ? Appuyez sur **Refresh** quand votre liste est affichée. Qu'observez vous ? Ajouter la méthode nécessaire dans votre code pour éviter l'augmentation permanente de votre liste.

Question 8 : On souhaite ajoutez un *Label* dans les items de la liste pour afficher le prochain tram (voir figure 1). Dans la classe TamSchedule, ajoutez une propriété **Prochain** qui renvoie le prochain tram, c'est-à-dire à dire le 1er élément du tableau **NextTrams**. Dans votre fichier xaml, ajouter ce *Label*. Directement en dessous du *Label* pour le *TramStop*. Qu'observez-vous ?

Question 9 : Ajouter un *ViewCell* et un layout dans le *DataTemplate* de votre *ListView* afin de résoudre le problème de la question **8.**

Question 10 : Faites visuellement ressortir le nom de l'arrêt de tram en le *Label* en gras dans la *ListView*.

Exercice 4 - Afficher les horaires de tramway en détail.

On souhaite maintenant créer un un panneau d'affichage pour les informations détaillés du tram. Comme présenté dans la figure **1.b**.

Question 11 : Créez un sous-layout (StackLayout) nommez le **InformationLayout**. Ajoutez lui 4 *Labels* pour :

- Le nom de l'arrêt (ex: Comedie)
- Le 1er tram à venir (ex: Odysseum 13:10:00)
- Le 2eme tram (ex: Odysseum 13:15:00)
- Le 3eme tram (ex: Odysseum 13:20:34)
- Un bouton **close** pour fermer le layout.

Ajoutez aussi un *backgroundColor* de couleur (celle qui vous plait le plus). En Plus, ajoutez une propriété *IsVisible* initialisé par défaut à **false**.

Question 12 : En utilisant la propriété *OnItemTapped* de *ListView*, créer une action qui a chaque fois qu'un item est sélectionné, le panneau **InformationLayout** s'affiche avec les informations correspondante.

Question 13 : Créer une action pour le bouton **close** qui ferme le panneau **InformationLayout**. Vous utiliserez la propriété *IsVisible* pour cela.

Exercice 5 - Aller plus loin

On souhaite rajouter une fonctionnalité « favoris » dans l'application afin d'afficher toujours en 1er dans la liste un arrêt de tram marqué comme **favoris.**

Question 14 : Ajoutez un bouton **Favorite** au dessus de bouton **close** dans le **InformationLayout** qui ajoute en favoris l'arrêt correspondant à l'item sélectionné. Lorsque ajouter, l'item de la listView devra changer de couleur pour indiquer le passage en favoris.

Question 15 : Ajoutez une entrée dans le dictionnaire **properties** de la classe App afin de sauvegarder l'arrêt de Tram favoris.

Question 16 : Ajoutez un trie sur la liste afin de toujours afficher en 1er l'arrêt de Tram favoris.