TP .Net Maui n° 3 : Utilisation de Web Service

Objectif : Utiliser des appels à des Web Services à travers .Net Maui

1 - Création du projet

Lancez « Visual Studio » et créez un nouveau projet de type « Application .NET MAUI ».
Lancez l'exécution en sélectionnant « Windows Machine ». Testez l'application.

Allez sous « Windows Machine » et faites « Gestionnaire d'appareils Android », pour lancer votre simulateur Android (Si il n'est pas configuré, faites « Nouveauté », sélectionnez « Pixel 2 (+Store), x86, Android 9 - API28 » et laissez les autres paramètres par défaut, faites ensuite « Créer ».)

Ajoutez un répertoire « Views » et un répertoire « Models » à votre projet.
Ajoutez en ressources les images fournies (Moodle).
Modifiez la couleur Primary à #005067 et la couleur Secondary à #E84D0D (Couleurs officielles charte de 3iL).
Dans View, ajoutez une nouvelle page « .NET MAUI ContentPage (XAML) » que vous nommez « vTemperature.xaml ». Toujours dans View, ajoutez une 2° page « .NET MAUI ContentPage (XAML) » que vous nommez « vMeteo.xaml ».

Ouvrez « AppShel.xaml » et remplacez la balise « `ShellContent` » par avec 2 boutons « Température » pointant sur « vTemperature » et « Météo » pointant sur « vMeteo » :

```
<TabBar>
        <ShellContent
        Title="Temepérature"
        ContentTemplate="{DataTemplate local:vMeteo}"
        Icon="icon_temp" />
        …
```

Pensez à modifier « `xmlns:local="..."` » en « `xmlns:local="clr-namespace:TPMaui3.Views"` »

2 - Appel simplifié d'un Service Web

Ouvrez « vTemperature.xaml », mettez les propriétés « `Spacing="20" Margin="10"` » au « `VerticalStackLayout` ».
Ajoutez-y un bouton « Afficher » centré horizontalement, et 2 labels centrés horizontalement (`HorizontalOptions`), le premier affichant « `Température` » avec « `FontSize="Large"` » et « `TextColor="{StaticResource Primary}"` », le deuxième nommé « `lbTemperature` » avec pour celui-ci « `HeightRequest="240" FontSize="Large"` » et la couleur « `Secondary` ».
Ouvrez « vTemperature.xaml.cs ». Créez une méthode « `LireTemperature()` » que vous appellerez dans la méthode de votre bouton.
Dans cette méthode :
- créez une variable de type « `Uri` » instanciée par
« `"http://meteorestsrvmobile.lab3il.fr/RestServiceMeteo.svc/xml/1"` »

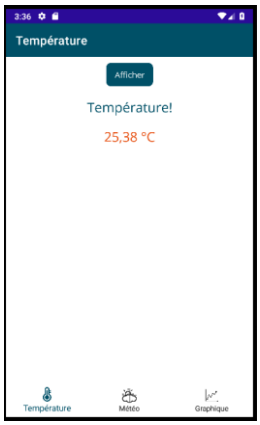
- créez un « try/catch » avec pour le catch l'affichage du message d'erreur dans une boîte de dialogue (la variable d'erreur étant définie par « `catch (Exception ex)` ») :

```
await this.DisplayAlert("Error", ex.Message, "OK");
```

- créez une variable de type « `HttpClient` » instanciée par « `new();` » qui va permettre de réaliser la requête
- créez une variable de type « `HttpResponseMessage` » qui va contenir la réponse de la requête
- appelez la méthode « `GetAsync` » de votre variable « `HttpClient` » pour récupérer en réponse votre objet de type « `HttpResponseMessage` »
- testez ensuite la valeur de « `IsSuccessStatusCode` » de votre objet réponse dans une boucle « if ».

Si cette valeur est vrai :

- chargez dans une chaîne de caractères la valeur renvoyée par « `Content.ReadAsStringAsync()` » de l'objet réponse
- traiez la chaîne obtenue pour extraire la valeur de la température (subtring démarrant à IndexOf de `"<XMLDataResult>"` plus sa longueur, et substring de 0 à IndexOf de `"</XMLDataResult>"`)
- chargez la valeur obtenue dans le label « `lbTemp` »



Pour une utilisation sur Android, il est nécessaire d'ouvrir les droits pour atteindre une Url en *http* car seules les *https* sont autorisées. Pour cela, ouvrez « MainApplication.cs » et remplacez la ligne «`[Application]` » par :

```
#if DEBUG
[Application(UsesCleartextTraffic = true)]
#else
[Application]
#endif
```

3 - Appel d'un Service Web avec désérialisation

- Désérialisation par package Newtonsoft

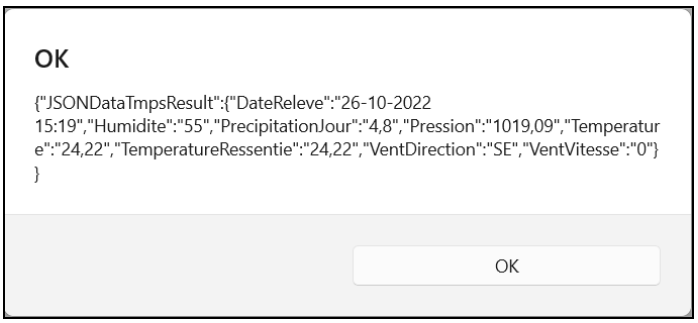
Ouvrez « vMeteo.xaml », mettez les propriétés « `Spacing="20" Margin="10"` » au « `VerticalStackLayout` ».
Ajoutez-y 2 labels centrés horizontalement (`HorizontalOptions`), le premier affichant « `Météo` » avec « `FontSize="Large"` » et « `TextColor="{StaticResource Primary}"` », le deuxième nommé « `lbTemperature` » avec pour celui-ci « `HeightRequest="240" FontSize="Large"` » et la couleur « `Secondary` ».
Ouvrez « vMeteo.xaml.cs ». Créez une méthode « `Meteo()` » que vous appellerez dans le constructeur de la fenêtre.
Dans cette méthode :

- créez une variable de type « `Uri` » instanciée par «`"http://meteorestsrvmobile.lab3il.fr/RestServiceMeteo.svc/jsontmps"` »
- créez un « try/catch » avec pour le catch l'affichage du message d'erreur
- créez une variable de type « `HttpClient` » instanciée par « `new();` » qui va permettre de réaliser la requête
- créez une variable de type « `HttpResponseMessage` » qui va contenir la réponse de la requête
- appelez la méthode « `GetAsync` » de votre variable « `HttpClient` » pour récupérer en réponse votre objet de type « `HttpResponseMessage` »
- testez ensuite la valeur de « `IsSuccessStatusCode` » de votre objet réponse dans une boucle « if ».

- si celle-ci est vraie, chargez dans un objet de type « `var` » la valeur renvoyée par « `Content.ReadAsStringAsync()` » de l'objet réponse
- affichez la valeur retournée dans une boîte de dialogue :

```
await this.DisplayAlert("Reçu :", VariableObjet.ToString(), "OK");
```

Testez votre application. Vous obtenez une chaîne de caractère Json qui peut être compliquée à traiter dans certains cas.



Pour éviter un traitement lourd, vous allez « désérialiser » cette réponse Json.
Après l'appel à « `ReadAsStringAsync` », faites un premier traitement de la chaîne obtenue pour supprimer l'entête « JSONDataTmpsResult » en appliquant :

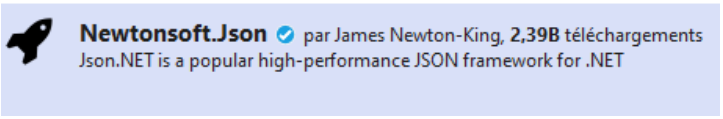
```
.Replace("{\"JSONDataTmpsResult\":", "");   et   .Replace("}}", "}");
```

Dans le répertoire « Models », ajoutez un fichier de classe C# nommée « CMeteo.cs ». Dans cette classe ajoutez les 8 données renvoyées en Json comme ceci :

```
public string Temperature { get; set; }
```

Respectez bien le fait que les noms des propriétés doivent correspondre à leur homologue du Json (affichez si nécessaire le Json dans un navigateur en copiant l'Url dans la barre d'adresse).

Dans les menus de VisualStudio, faites « Projet/Gérer les packages NuGet… ». Dans la fenêtre sélectionnez « Parcourir » et tapez « Newtonsoft ». Sélectionnez ensuite le package « Newtonsoft.Json » et dans la partie droite cliquez sur « Installer ». Ce package est un PlugIn .Net permettant de traiter de façon assez simple les paquets de données Json.



Lorsque que le package est installé, vous pouvez refermer l'onglet NuGet.

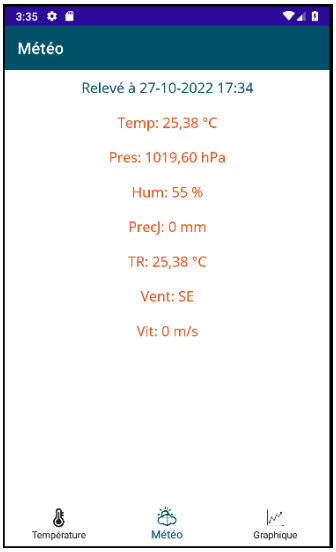
Créez ensuite une variable de type « `var` » que vous allez initialiser en appelant « `DeserializeObject` » disponible dans « `Newtonsoft.Json.JsonConvert` » en lui passant votre variable Json et en lui indiquant que le résultat est un object « CMeteo » :

```
var ... = Newtonsoft.Json.JsonConvert.DeserializeObject<CMeteo>(...);
```

Si l'objet obtenu est de type « CMeteo », il suffit d'appeler sa propriété « `Temperature` » pour l'afficher dans le Label prévu à cet effet.

Ajoutez les Labels manquant et afficher le relevé complet.

Réduisez la taille des textes des Labels avec « FontSize » à « Medium ».



4 - Graphisme avec composants

Dans View, ajoutez une nouvelle page « .NET MAUI ContentPage (XAML) » que vous nommez « vGraphe.xaml ». Ajoutez un lien sur cette page à votre barre de contrôle.

Ouvrez « vGraphe.xaml », mettez les propriétés « `Spacing="20" Margin="10"` » au « `VerticalStackLayout` ». Ajoutez-y un Label centré horizontalement (`HorizontalOptions`) affichant « `Historique` » avec « `FontSize="Large"` » et « `TextColor="{StaticResource Primary}"` », et un bouton « Afficher » centré horizontalement et attribué lui une méthode « `AfficherGraph` ».
Ajoutez toujours dans le « `VerticalStackLayout` » un « `HorizontalStackLayout` » nommé « `HSL` » avec les propriétés « `Spacing="10" Margin="10" HorizontalOptions="Center" HeightRequest="200"` ».
Dans celui-ci, ajoutez 5 Labels nommés de « lb1 » à « lb5 », sans texte, avec les propriétés « `VerticalOptions="End"` » (position en bas), « `BackgroundColor="{StaticResource Secondary}"` » (couleur de fond), « `WidthRequest` » à 20 et « `HeightRequest` » à 10.

Allez sur l'Url : http://meteorestsrvmobile.lab3il.fr/RestServiceMeteo.svc/xmlhisto
Cette-ci renvoie un historique sur 5 jours de plusieurs paramètres météo. Vous allez utiliser ce lien pour récupérer les données et les mettre sous une forme graphique.

Ouvrez « vGraphe.xaml.cs », et :
- déclarez une « `Uri` » pointant sur ce lien
- déclarez un objet « `HttpClient` » et un objet « `HttpResponseMessage` » lié au premier par « `GetAsync` »
- testez la réponse par « `IsSuccessStatusCode` » et pour Vrai récupérez la valeur obtenue dans un objet « var » (`ReadAsStringAsync`)

Ajoutez ensuite la méthode suivante à votre classe de fenêtre :

```
private double ReadValue(String sXml, String sFlag)
{
    try
    {
        String sFlag1 = "<a:" + sFlag + ">";
        String sFlag2 = "</a:" + sFlag + ">";
        sXml = sXml.Substring(sXml.IndexOf(sFlag1) + sFlag1.Length);
        sXml = sXml.Substring(0, sXml.IndexOf(sFlag2));
        sXml = sXml.Replace(",", ".");
        double dVal = Convert.ToDouble(sXml,
                    System.Globalization.CultureInfo.InvariantCulture);
        return dVal;
    }
    catch { }
    return 0;
}
```

Cette méthode va traiter la chaîne « `sXml` » et extraire la valeur associée au « `sFlag` ». Par exemple pour obtenir la valeur de « `TMn0` » il faut appeler « `ReadValue(resultxml, "TMn0")` »

Pour tracer par exemple l'évolution des températures minimales des 5 derniers jours, il vous suffit donc de récupérer les valeurs de TMn0 à TMn4 et d'affecter leur valeur aux hauteurs des Labels lb1 et ln5.
Par contre, il faudra adapter proportionnellement ces hauteurs à celle de leur conteneur « `HSL` ». Il suffit de calculer le rapport « Hauteur du conteneur » / « ordonnée maximale des valeurs de température du graphe » et de multiplier les hauteurs des Labels par cette valeur. Vous pouvez considérer l'ordonnée maximale du graphe à 20.

Si vous souhaitez afficher un axe des ordonnées avec les unités, vous pouvez utiliser une image qui représente cet axe (voir exemple sur Moodle).