# Template Windows Phone 8 en C#/XAML

# « Be a Journalist » !

  
*Des journalistes commentent Roland-Garros (D. PALLAGES/AFP)*

## But du template

On va construire la base d’une application qui va vous permettre de devenir un commentateur sportif ! Ca pourra vous servir pour tous les grands évènements, que ce soit du foot, du rugby, ou pour Roland Garros 2013 qui commence bientôt !

Vous allez pouvoir écrire vos commentaires en live sur les matchs, et tous vos utilisateurs pourront alors suivre le match que vous regardez grâce à vos commentaires ! Futurs « T. Roland », c’est pour vous !

C’est donc parti pour notre appli Windows Phone 8 où l'on va s’appuyer sur la plate-forme Microsoft Azure, qui va gérer toute la partie Cloud pour nous (et ce n’est pas rien !).

## Temps estimé pour réaliser ce template : **30/45 minutes**

## Difficulté : **débutant** / **intermédiaire**

Pré-requis : On va utiliser Visual Studio 2012, mais si vous ne le possédez pas, ça marche tout aussi bien sur [la version gratuite « Express »](http://www.microsoft.com/france/visual-studio/essayez/express.aspx).

Une fois installé, assurez-vous aussi d’avoir installé [le SDK Windows Phone](https://dev.windowsphone.com/en-us/downloadsdk) pour développer des applis WP8, et [le dernier SDK Azure](http://www.windowsazure.com/en-us/downloads/?fb=fr-fr).  
Assurez-vous aussi d’être réveillé et d’avoir 30 minutes devant vous, ça suffira pour l’instant ;)

### Et c’est parti !!

## 0/ Organisation générale de l’application

L’idée étant de sortir une application simple mais fonctionnelle, on va volontairement se limiter à une application faite en seulement deux pages :

* Une page principale pour tous les utilisateurs, qui va afficher tous les commentaires des matchs récupérés depuis Azure.
* Une page admin, réservée aux commentateurs, accessible avec un identifiant unique type Microsoft Account, pour éditer les commentaires (ajout /suppression).

Voilà pour la structure, difficile de faire plus simple !

Et tous nos commentaires seront stockés dans Azure, pour que chacun puisse les voir.

On va donc d’abord mettre en place la partie cloud (Azure), puis on va tranquillement construire nos deux pages. On ne perd pas de temps, c’est parti direction : le nuage.

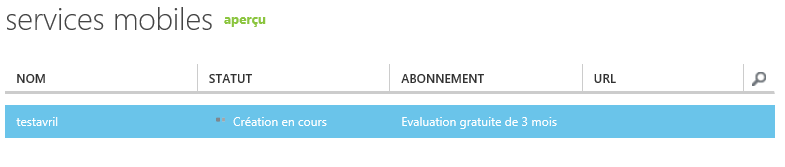
## 1/ Préparer Azure

Nous allons utiliser au cours de ce tutoriel Azure Mobile Services, donc vous allez avoir besoin d’un compte Azure. Si vous n'en possédez pas encore, vous pouvez vous en créer un rapidement en suivant [ce tutoriel](http://blogs.msdn.com/b/windowsazurefrance/archive/2012/08/29/tuto-comment-souscrire-gratuitement-224-windows-azure.aspx).

Une fois cette étape faite, rendez-vous sur <https://manage.windowsazure.com/>

Sur ce portail Azure, cliquez sur « Services Mobiles » dans le menu de gauche, puis sur « Créer un Service Mobile ».

Suivez ensuite les étapes en choisissant les options qui vous correspondent (une url, l’emplacement de la BDD etc…). Votre instance se crée alors, cela peut prendre une minute ou deux si Azure doit aussi vous créer votre base de donnée.

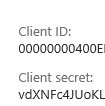


Une fois l’installation terminée cliquez sur votre instance puis allez sur l’onglet « données » en haut de la page.

Une fois sur cet onglet, vous devez avoir le message « Vous n’avez aucune table. », cliquez alors sur « ajouter une table » juste en dessous de ce message ou sur le bouton « créer » en bas de la page, appelez alors votre table : « *CommentItem* » et laissez les permissions comme prédéfinies, on ne les changera pas ici.

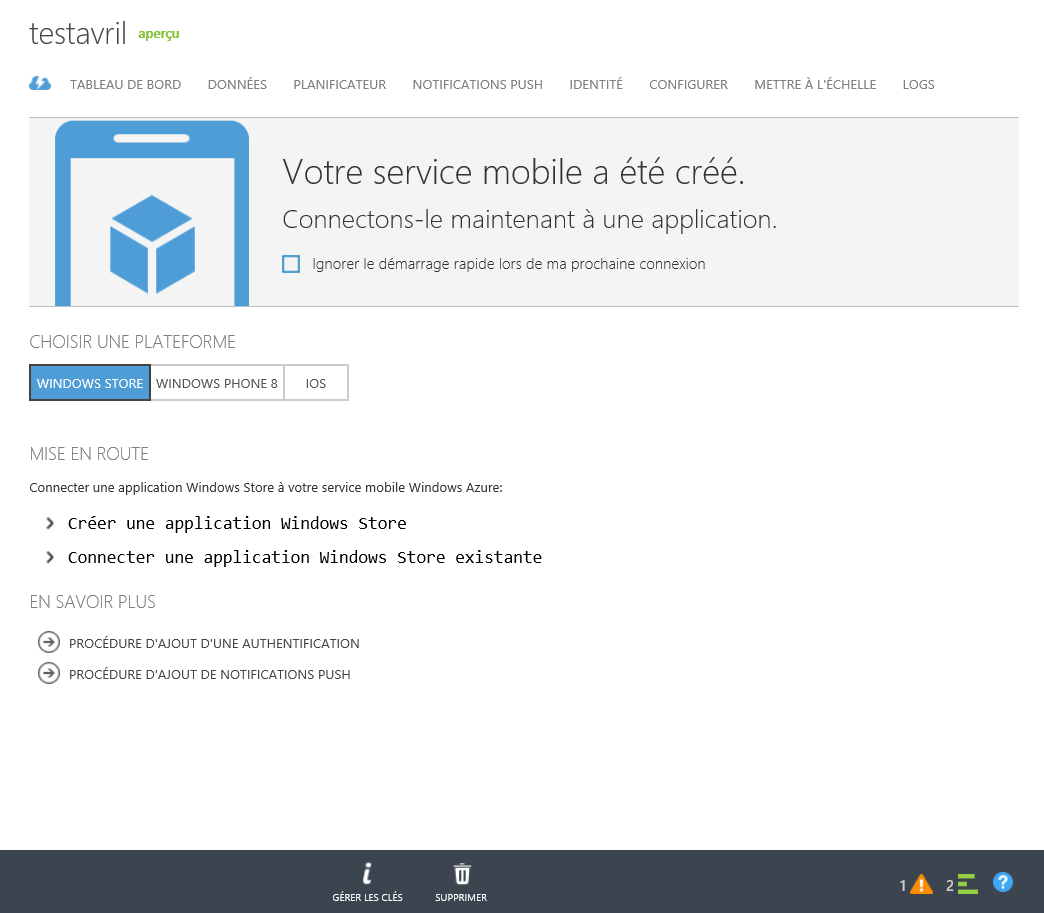
Nous allons préparer notre instance pour la création de l’application en configurant dès à présent notre système de connexion pour les administrateurs. Pour ce faire, rendez-vous sur la page suivante : <http://dev.live.com>

Connectez-vous, puis créez sur l’onglet « My Apps » et créez une nouvelle application, entrez alors dans le champ « Redirect domain » l’url de votre instance Azure (qui est disponible sur le tableau de bord de votre instance Mobile Services) puis cochez « Yes » à la question « Mobile client app ». Récupérez ensuite vos identifiants d’application :



Retournez alors sur votre instance Azure Mobile Services, allez dans l’onglet « identité » et rentrez le « Client ID » dans le champ « ID CLIENT » et le « Client secret » dans le champ « Question secrète du client » et enfin, cliquez sur « Enregistrer » en bas de la page.

Retournons maintenant sur la page d’accueil de notre instance :



Choisissez Windows Phone 8 en plateforme puis « Connecter une application Windows Phone 8 existante » et récupérez le code dans l’onglet « 2/ Connecter l’application » (copiez le, on va le coller dans 2 minutes !).

## 2/ Page principale : affichage des commentaires

Retour sur Terre, il est maintenant temps de se mettre au code C#/XAML de notre application !

1. **Préparation du projet**

Au début, rien de difficile, on va juste préparer notre projet en faisant le lien App <-> Azure. On crée donc un nouveau projet (ici sous Visual Studio 2012) de type « Application Windows Phone », ce qui va nous créer un projet avec tout le minimum pour notre application.

Pour le lier à Azure, rien de plus simple : on rajoute d’abord une référence à « Windows Azure Mobile Services Managed Client » afin de bénéficier des fonctions d’Azure. Quand vous rajoutez une référence, allez dans l’onglet « Windows Phone », puis « Extensions » pour la trouver.

Et du coup, il nous reste à ajouter le lien en question, que l’on placera dans le fichier « App.xaml.cs », au début, avec les autres déclarations de variables :

public static PhoneApplicationFrame RootFrame { get; private set; }

public static MobileServiceClient MobileService = new MobileServiceClient(

"https://avrilw8.azure-mobile.net/",

"ytokgjjo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

(C’est ici qu’il faut coller votre bout de code !)

Et voilà, on est connecté au cloud ! Impressionnant non ? Notre variable « Mobile Service » va désormais nous servir à faire le lien Terre <-> Nuage, rien que ça !

1. **Préparer notre interface (en XAML)**

Et oui, on ne va pas récupérer nos commentaires si on pas d’endroits où les mettre ! Pour les accueillir, rien de tel qu’un LongListSelector, qui permettra d’afficher les commentaires sous forme d’une liste. Question design, on laisse votre imagination débordante s’en charger. Pour l’instant on va donc se contenter de placer cette liste dans notre fichier MainPage.xaml, en lui donnant le nom « ListItems » :

[…]

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<phone:LongListSelector x:Name="ListItems" Margin="0,0,-12,0" ItemsSource="{Binding Items}">

<phone:LongListSelector.ItemTemplate>

<DataTemplate>

<StackPanel Margin="0,0,0,17" Orientation="Horizontal">

<Border BorderThickness="1" BorderBrush="White" Background="White" Margin="10" CornerRadius="4" Width="430">

<TextBlock Text="{Binding Contenu}" Margin="5" TextWrapping="Wrap" Foreground="Black"/>

</Border>

</StackPanel>

</DataTemplate>

</phone:LongListSelector.ItemTemplate>

</phone:LongListSelector>

</Grid>

[…]

1. **Définition d’un commentaire**

Un dernier point pour préparer le terrain : définir ce qu’est un « commentaire » ! On en parle beaucoup, mais ça correspond à quoi ? On vous propose donc cette classe « CommentItem » avec 4 paramètres, classe que vous ferez évoluer par la suite selon vos envies ;)

public class CommentItem

{

public int Id { get; set; }

[DataMember(Name = "sport")]

public string Sport { get; set; }

[DataMember(Name = "match")]

public string Match { get; set; }

[DataMember(Name = "comment")]

public string Contenu { get; set; }

}

1. **Obtenir les commentaires sur Azure**

Ca y est, on a donc de quoi accueillir nos commentaires. Il serait peut-être temps de les récupérer !

Alors direction le code behind de notre page principale (MainPage.xaml.cs), où l’on va rajouter une petite fonction qui va se charger d’aller chercher nos commentaires et de les afficher.

On crée simplement une variable « commentTable » qui va contenir notre table de commentaires (celle stockée sur Azure), que l’on déclare au début de notre classe, ainsi qu’une variable « items » qui contiendra la collection de commentaire triées :

private IMobileServiceTable<CommentItem> commentTable = App.MobileService.GetTable<CommentItem>();

private MobileServiceCollectionView<CommentItem> items;

Il nous reste plus qu’à créer une fonction « RefreshCommentsItems », qui va faire le travail en deux petites lignes !

private void RefreshCommentsItems()

{

items = commentTable

.OrderByDescending(x => x.Id)

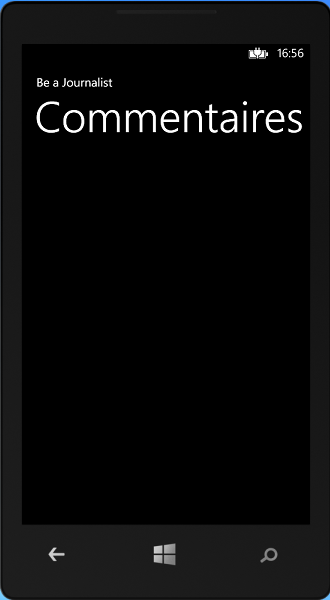
.ToCollectionView();

ListItems.ItemsSource = items;

}

Et voilà !! En deux lignes, on a récupéré nos commentaires, on les a triés par ID pour les avoir dans le bon ordre, et on les a affichées dans notre « ListItems » ! Tout bête non ?!

Et je vous laisse le plaisir de lancer votre application et de découvrir tous vos commentaires sportifs directement affichées sur votre Windows Phone !



Admirez la beaut….Euhh…. Comment ça c’est vide ?? Ah mais oui, on a encore rien écrit !

Au temps pour moi, on règle ca tout de suite !

## 3/ Page principale : partie administration

Bon allez, courage, on s’y remet, vous ne le regretterez pas. On va maintenant s’attaquer à la partie d’administration pour rajouter nos commentaires du match Nadal-Federer !

1. **Accès à la partie administration**

Toujours pour faire simple, l’accès à cette partie se fera via l’application bar, ce genre de menu en bas de l’application :

App Bar in default size

Nous, on ne va pas rajouter autant de boutons, on va juste en mettre un, qui permettra d’accéder à la partie privée. Pour cela, ultra-classique, on place le code suivant en bas de notre fichier XAML (MainPage.xaml) :

<phone:PhoneApplicationPage.ApplicationBar>

<shell:ApplicationBar IsVisible="True" IsMenuEnabled="False">

<shell:ApplicationBar.Buttons>

<shell:ApplicationBarIconButton x:Name="admin" Text="admin" IconUri="/images/admin.png" Click="admin\_Click"/>

</shell:ApplicationBar.Buttons>

</shell:ApplicationBar>

</phone:PhoneApplicationPage.ApplicationBar>

1. **Lancement de l’authentification**

Comme vous le voyez, le bouton que l’on vient de rajouter est abonné à l’évènement « admin\_click », qui va nous permettre de déclencher l’authentification. Afin de ne pas bloquer l’interface, cette fonction sera asynchrone. Voilà donc le code behind correspondant au clic :

private async void admin\_Click(object sender, EventArgs e)

{

await Authenticate();

}

Et je vous le donne en mille sur la suite : on va s’occuper de la méthode… « Authentificate » !

Et pour ça, une seule ligne suffit !! (Merci Azure !)

/// <summary>

/// Permet de lancer l'authentification asynchrone

/// </summary>

private async System.Threading.Tasks.Task Authenticate()

{

Await App.MobileService.LoginAsync(MobileServiceAuthenticationProvider.MicrosoftAccount);

}

Et là, surprise, je vous laisse tester l’application : au clic sur notre application bar apparait un bel écran de connexion à votre compte Microsoft, comme ça, sans rien faire de plus !

Et maintenant ?

Si l’utilisateur rentre un login/mot de passe Microsoft valide, alors la fonction continue. Sinon, cela lève une exception (*InvalidOperationException*). Mais on ne veut pas laisser l’accès à notre partie admin à tous les gens qui possède un compte Microsoft !

Alors du coup, on va simplement vérifier qu’il s’agit bien de vous (ou d’un tierce). Si c’est le cas, on envoie l’utilisateur à la page Admin, sinon… à vous de voir ! ;)

Donc, juste après, toujours dans notre méthode « Authentificate » :

if (App.user.UserId.Equals(MY\_MICROSOFT\_ID))

{

// ("Bienvenue dans votre espace !");

this.NavigationService.Navigate(new Uri("/Admin.xaml", UriKind.Relative));

}

else

// "Désolé, ce compte n'est pas administrateur";

Avec la variable MY\_MICROSOFT\_ID qui sera définie en haut avec l’ID du compte admin (le vôtre ou celui voulu), par ex :

private string MY\_MICROSOFT\_ID = "MicrosoftAccount:15ebaa53ba0c8a3f1ce7d0e3fa4d94c8";

Et comment je le trouve moi, **mon** ID Microsoft ??  
Malheureusement, pas de solution miracle pour l’instant, du coup, je vous laisse le trouver en rajoutant une ligne comme celle-ci, juste avant la condition « if » précédente :

MessageBox.Show(App.MobileService.CurrentUser.UserId);

Ça vous donnera votre Microsoft ID, n’oubliez pas de l’enlever après par contre ;)

## 4/ Page admin

Ca y est, vous êtes authentifié comme Admin, et vous arrivez donc sur votre page à VOUS, futur « T. Roland » !

Qu’est-ce qu’on va y faire ?! Et bien on peut commencer par l’essentiel : pouvoir y rajouter et supprimer des commentaires !

1. **Une touche de design, très légère**

Oui parce que pour rajouter et supprimer des commentaires, même si vous serez le seul à le faire, il vous faut au moins un affichage fonctionnel. Alors faisons simple, avec deux pages pivot, une pour ajouter des commentaires, une pour en supprimer :

<!--LayoutRoot est la grille racine où tout le contenu de la page est placé-->

<Grid x:Name="LayoutRoot" Background="Transparent">

<!--Contrôle Pivot-->

<phone:Pivot Title="Be a Journalist">

<phone:PivotItem Header="Ajout">

< <StackPanel>

<StackPanel>

<TextBlock Text="Tapez ci-dessous un nouveau commentaire. Une fois ajouté, il sera visible par tous vos utilisateurs !" TextWrapping="Wrap" />

<TextBox x:Name="TextComment" HorizontalAlignment="Left" Height="200" TextWrapping="Wrap" Text="Votre commentaire sportif ici..." VerticalAlignment="Top" Width="456" FontSize="22" GotFocus="TextComment\_GotFocus" AcceptsReturn="True" />

<Button Name="Badd" Content="Ajouter" Click="Badd\_Click"/>

</StackPanel>

</StackPanel>

</phone:PivotItem>

<phone:PivotItem Header="Suppression">

<phone:LongListSelector x:Name="ListItems" Margin="0,0,-12,0" ItemsSource="{Binding Items}" SelectionChanged="ListItems\_SelectionChanged">

<phone:LongListSelector.ItemTemplate>

<DataTemplate>

<StackPanel Margin="0,0,0,17" Orientation="Horizontal">

<Border BorderThickness="1" BorderBrush="White" Background="White" Margin="10" CornerRadius="4" Width="430">

<TextBlock Text="{Binding Contenu}" Margin="5" TextWrapping="Wrap" Foreground="Black"/>

</Border>

</StackPanel>

</DataTemplate>

</phone:LongListSelector.ItemTemplate>

</phone:LongListSelector>

</phone:PivotItem>

</phone:Pivot>

</Grid>

Oui, ça a l’air un peu long et compliqué, mais regardez de plus près, il n’y a vraiment rien de sorcier !

Dans le premier pivot, on a en fait une simple TextBox où vous pourrez rédiger votre commentaire, et dans le second, on a juste remis la liste complète des commentaires, qu’on supprimera en cliquant dessus.

Mais voyons le concret tout de suite dans le code behind C#.

1. **Affichage des commentaires publiés**

Là, pas de nouveautés, l’idée est juste d’afficher à nouveau la liste des commentaires comme précédemment, dans le « LongListSelector » :

private IMobileServiceTable<CommentItem> commentTable = App.MobileService.GetTable<CommentItem>();

private MobileServiceCollectionView<CommentItem> items;

private void RefreshComments()

{

items = commentTable

.OrderByDescending(x => x.Id)

.ToCollectionView();

ListItems.ItemsSource = items;

}

Ça nous sert à voir la liste des commentaires publiés, pour plus tard en supprimer en cliquant directement dessus ;)

1. **L’ajout d’un nouveau commentaire**

On va enfin pouvoir commenter notre match (il serait temps, c’est la reprise) ! Comme vous l’avez peut être remarqué dans le XAML du dessus, on a mis un évènement click sur le bouton « Ajouter ». Et bien on va l'implémenter comme ceci :

//Clic sur l'ajout d'un commentaire

private void Badd\_Click(object sender, RoutedEventArgs e)

{

//On crée un objet comment et on lance l’ajout

CommentItem comment = new CommentItem { Contenu = TextComment.Text, Sport = "Tennis", Match = "Nadal - federer" };

InsertCommentItem(comment);

}

On a créé un object « comment » qui contient le commentaire que vous venez d’écrire dans le champ « TextComment ». Du coup, il nous reste plus qu’à l’envoyer sur Azure, et là on remercie encore Azure pour faire dans la simplicité absolue :

private async void InsertCommentItem(CommmentItem commentItem)

{

await commentTable.InsertAsync(commentItem);

MessageBox.Show("Ce commentaire a bien été ajouté.");

}

Tada ! Je suis désolé, on a essayé de faire plus simple mais on n’a pas réussi !

1. **La suppression d’un commentaire existant**

Et bien pas trop de surprises, c’est à peu près la même chose que pour l’ajout : au moment où l’administrateur clique sur un commentaire de la liste, on récupère l’objet et on lance sa suppression.

private void ListItems\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

CommentItem ci = (CommentItem)e.AddedItems[0];

DeleteComment(ci);

}

Il nous reste donc à supprimer ce commentaire situé sur Azure, là encore, d’une façon assez compliquée, accrochez-vous :

/// <summary>

/// Permet de supprimer un commentaire de la base Azure, de facon asynchrone

/// </summary>

private async void DeleteComment(CommentItem ci)

{

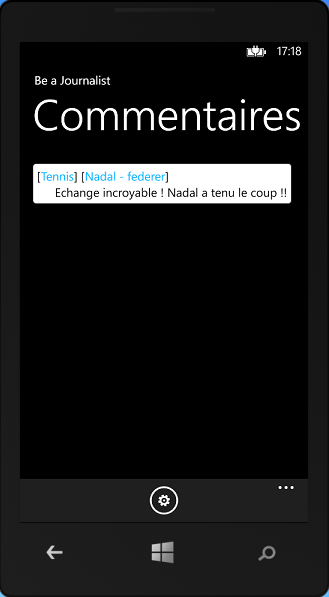
await this.commentTable.DeleteAsync(ci);

}

Encore vivant après un pavé de code pareil ?? ;)

Et bien bonne nouvelle, c’est FINI !!!

Et tiens, je ne vous laisse **même pas** la surprise du résultat :



Mais je vous laisse profiter de votre application, et surtout je vous laisse en faire une VRAIE application ! (Non parce que là, jamais ça passera la validation du Windows Phone Store ! ^^).

N’hésitez pas à télécharger le ZIP avec le template construit (légèrement retouché).

Je vous quitte sur quelques pistes qui peuvent vous aider améliorer votre application, profitez-en ! Et jetez un coup d’œil à nos Tips pour l’enrichir le plus possible !

Comme disait T. Roland : *« Après ça on peut mourir tranquille ! Enfin, le plus tard possible, mais on peut ! ».* Bon match !!

## 5/ Idées pour aller plus loin

* Faire une authentification côté serveur sur Azure afin de sécuriser votre partie administration de manière correcte
* Concevoir un vrai design « modern UI » !
* Prévoir un tri sur plusieurs sports (voir notre ZIP pour commencer)
* Avoir plusieurs administrateurs possibles, pour créer une équipe de commentateurs et commenter tous les gros matches !
* Faire une notification push à chaque nouveau commentaire (utilisez Azure !)