FILE → NEW APP

May 12th, 2017

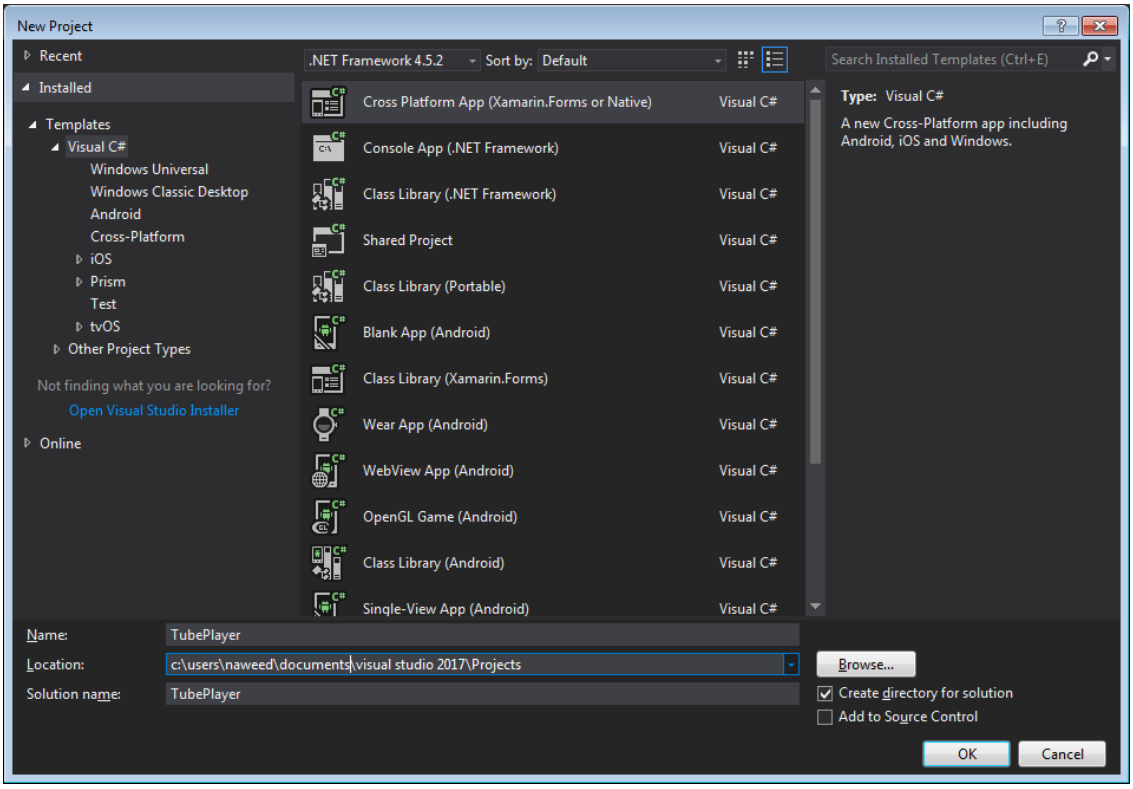
Prepared by

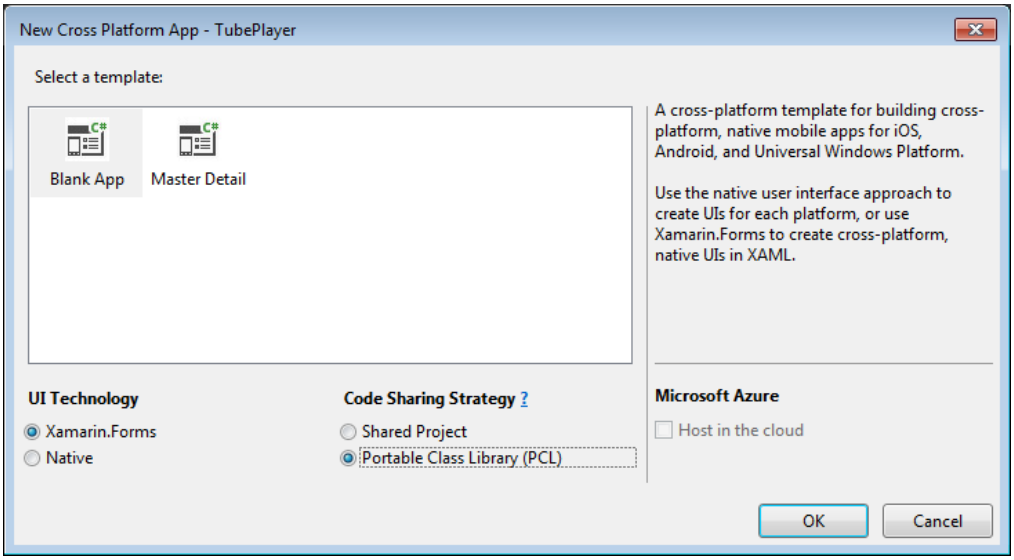
Naweed Akram

naweed@xgeno.com

Step 01:

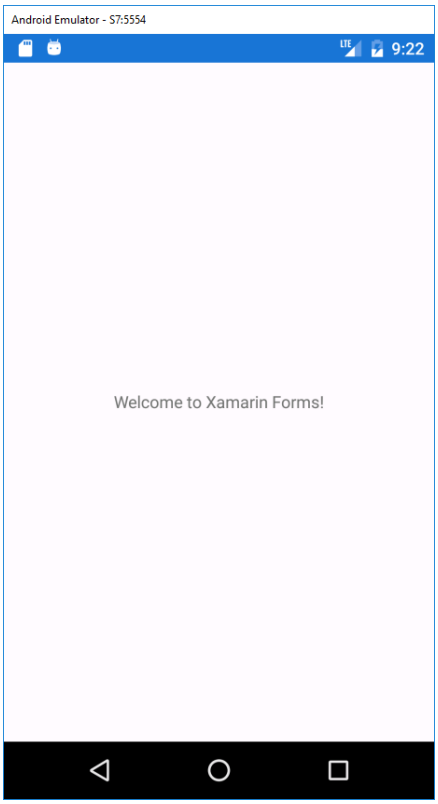
Create a new VS 2017 **Cross Platform App (Xamarin Forms or Native)** Project and name it **TubePlayer**.





Step 02:

Run the application (Android or iOS) to make sure it runs fine and the following Welcome page is displayed in Emulator or Device.



We will do all the work in TubePlayer (Portable) project to acheive 100% code reuse.

Step 03:

In **App.xaml.cs** code-behind, make the following change to the constructor:

```
public App()
{
   InitializeComponent();

   MainPage = new NavigationPage(new TubePlayer.MainPage());
}
```

Step 04:

Open **MainPage.xaml** and remove the **Welcome to Xamarin Forms** label and add the following:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:TubePlayer"
             x:Class="TubePlayer.MainPage"
             Title="Video Search">

    <Grid HorizontalOptions="FillAndExpand" VerticalOptions="FillAndExpand">
        <Grid.RowDefinitions>
            <RowDefinition Height="50" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <!-- Search Bar will come here -->

        <!-- Video ListView will come here -->

        <!-- Progress Bar - Hidden -->
        <StackLayout x:Name="stkBusy"
                     IsVisible="false"
                     BackgroundColor="Transparent"
                     Padding="12"
                     HorizontalOptions="Center"
                     VerticalOptions="Center"
                     Grid.RowSpan="2">
            <ActivityIndicator x:Name="indBusy" IsRunning="false" Color ="Black" />
            <Label Text="LOADING ..." HorizontalOptions="Center" TextColor="Black"/>
        </StackLayout>
    </Grid>

</ContentPage>
```

Step 05:

Create a new class file **"DataModels.cs"** and add the following class definitions:

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

namespace TubePlayer
{
    public class YoutubeVideos
    {
        public List<YoutubeVideo> items { get; set; }
    }

    public class YoutubeVideo
    {
        public Id id { get; set; }
        public Snippet snippet { get; set; }
```

```
    public string VideoID
    {
      get { return id.videoId; }
    }

    public string ThumbNail
    {
      get { return snippet.thumbnails.@default.url; }
    }

    public string ThumbNailHigh
    {
      get { return snippet.thumbnails.high.url; }
    }

    public string VideoTitle
    {
      get { return snippet.title; }
    }

    public string VideoDescription
    {
      get { return snippet.description; }
    }

    public string VideoReleaseDate
    {
      get { return "Released: " + Convert.ToDateTime(snippet.publishedAt).ToString("MMM dd, yyyy"); }
    }

}

public class Id
{
    public string videoId { get; set; }
}

public class Snippet
{
    public string publishedAt { get; set; }
    public string title { get; set; }
    public string description { get; set; }
    public Thumbnails thumbnails { get; set; }
}

public class Thumbnails
{
    public YoutubeImage @default { get; set; }
    public YoutubeImage medium { get; set; }
    public YoutubeImage high { get; set; }
}

public class YoutubeImage
{
    public string url { get; set; }
    public int width { get; set; }
    public int height { get; set; }
}

public class YoutubeVideos2
{
    public List<YoutubeVideo2> items { get; set; }
}

public class YoutubeVideo2
```

```
  {
    public string id { get; set; }
    public Snippet snippet { get; set; }

    public string VideoID
    {
      get { return id; }
    }

    public string ThumbNail
    {
      get { return snippet.thumbnails.@default.url; }
    }

    public string ThumbNailHigh
    {
      get { return snippet.thumbnails.high.url; }
    }

    public string VideoTitle
    {
      get { return snippet.title; }
    }

    public string VideoDescription
    {
      get { return snippet.description; }
    }

    public string VideoReleaseDate
    {
      get { return "Released: " + Convert.ToDateTime(snippet.publishedAt).ToString("MMM dd, yyyy"); }
    }
  }
}
```

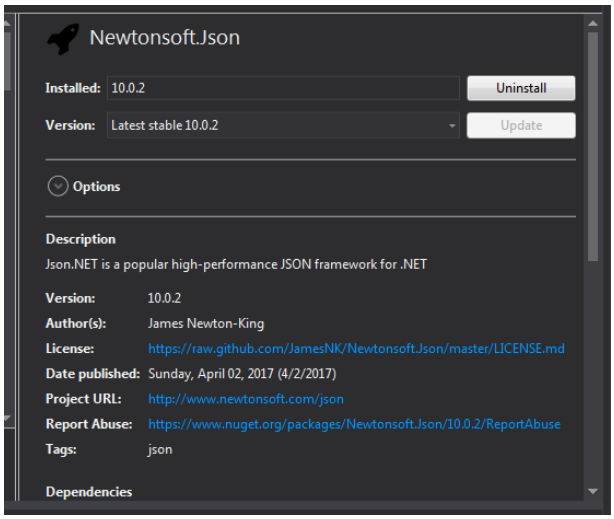
Step 06:

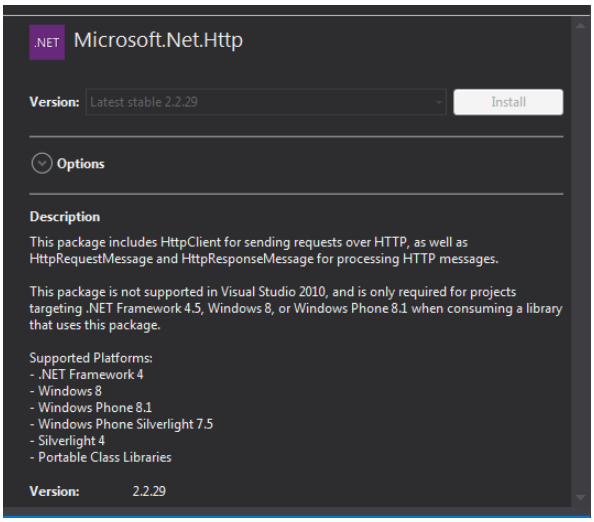
Open **MainPage.xaml.cs** file and add the following new methods:

```
    protected override void OnAppearing()
    {
      base.OnAppearing();

      LoadVideos("Xamarin");
    }

    public async Task LoadVideos(string searchQuery)
    {
      stkBusy.IsVisible = true;
      indBusy.IsRunning = true;

      var searchURL =
"https://www.googleapis.com/youtube/v3/search?part=snippet&type=video&key=AIzaSyBni3RFS7oOIBTHhibN6age5ABtVtgLW7o&maxResults=20&q=" + Uri.EscapeDataString(searchQuery);

      var httpClient = new HttpClient();
      var jsonResult = await httpClient.GetStringAsync(searchURL);
      var videosResult = JsonConvert.DeserializeObject<YoutubeVideos>(jsonResult);

      this.BindingContext = videosResult;

      stkBusy.IsVisible = false;
      indBusy.IsRunning = false;
    }
```

Step 07:

Resolve the namespaces by adding the following two NuGet packages to the **portable project:**





Add the two **using** statements in **MainPage.xaml.cs**:

```
using System.Net.Http;
using Newtonsoft.Json;
```

Step 08:

In **MainPage.xaml**, add the following ListView:

```
<ListView x:Name="lstVideos"
        ItemsSource="{Binding items}"
        RowHeight="132"
        Grid.Row="1">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <Grid Padding="6,6,6,6">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="150" />
                        <ColumnDefinition Width="*" />
                    </Grid.ColumnDefinitions>

                    <Grid.RowDefinitions>
                        <RowDefinition Height="32" />
                        <RowDefinition Height="80" />
                        <RowDefinition Height="8" />
                    </Grid.RowDefinitions>

                    <Image Source="{Binding ThumbNail}" Grid.Column="0" Grid.Row="0" Grid.RowSpan="2" />
                    <Label Text="{Binding VideoTitle}" TextColor="#336699" FontSize="14"
                        Grid.Column="1" Grid.Row="0" />
                    <Label Text="{Binding VideoDescription}" HorizontalOptions="EndAndExpand"
                        TextColor="#503026" FontSize="10" Grid.Column="1" Grid.Row="1" />
                </Grid>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Run the application to see the result. It will display **XAMARIN** videos from Youtube.

Step 09:

In **MainPage.xaml** and **MainPage.xaml.cs**, add the following Search bar and also the code behind:

```
<!-- Search Bar will come here -->
<SearchBar x:Name="btnSearch" Grid.Row="0" Placeholder="Search for Videos ..."
        HorizontalOptions="Fill" SearchButtonPressed="btnSearch_SearchButtonPressed" />


private void btnSearch_SearchButtonPressed(object sender, EventArgs e)
{
   LoadVideos(btnSearch.Text);
}
```

Run the application and search for videos of your choice.

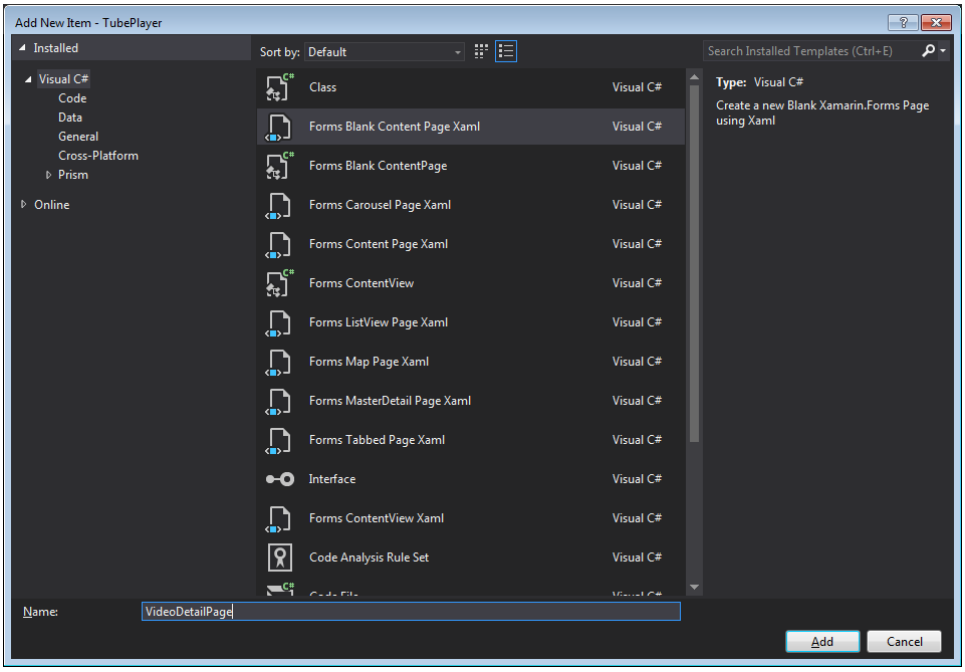
Step 10:

In **MainPage.xaml** and **MainPage.xaml.cs**, add the following to Navigate to the Video Details Page:

```
<!-- Video ListView will come here -->
<ListView x:Name="lstVideos"
        ItemsSource="{Binding items}"
        RowHeight="132"
        Grid.Row="1"
        ItemTapped="lstVideos_ItemTapped">

private void lstVideos_ItemTapped(object sender, ItemTappedEventArgs e)
{
   var selectedVideo = e.Item as YoutubeVideo;

   this.Navigation.PushAsync(new VideoDetailPage(selectedVideo.VideoID));
}
```

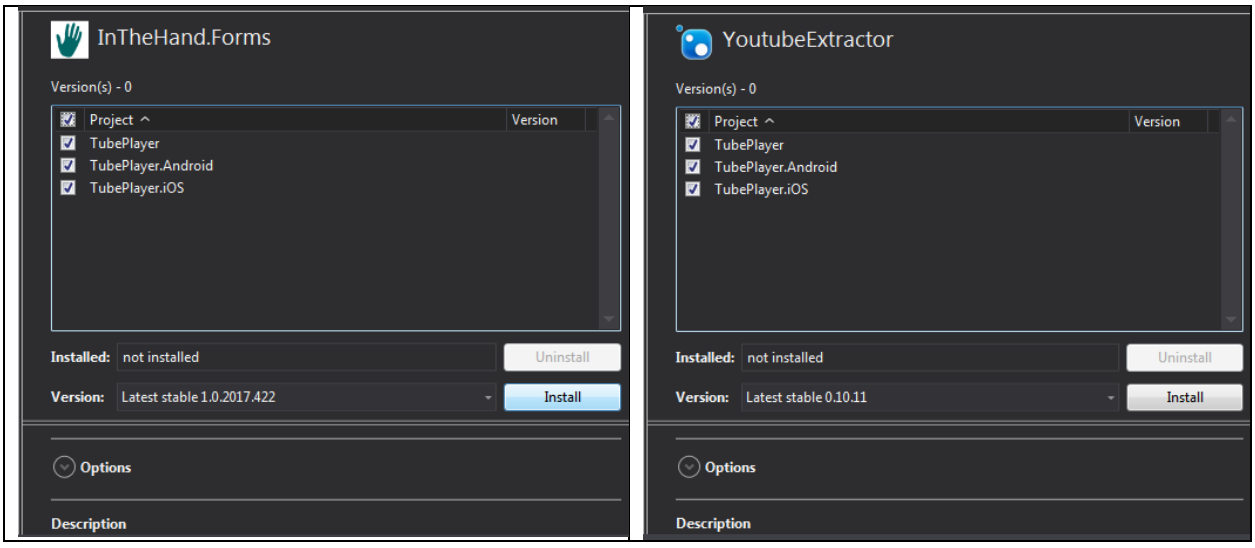
Step 11:

Add a new **Blank Forms Blank Content Page Xaml** called " **VideoDetailPage.xaml**":



Step 12:

Add the following two NuGet package to the **whole solution**:



Step 13:

Add the following code to **VideoDetailPage.xaml**:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:video="clr-namespace:InTheHand.Forms;assembly=InTheHand.Forms"
             x:Class="TubePlayer.VideoDetailPage"
             Title="Video Details">

    <Grid HorizontalOptions="FillAndExpand" VerticalOptions="FillAndExpand">
        <Grid.RowDefinitions>
            <RowDefinition Height="200" />
            <RowDefinition Height="45" />
            <RowDefinition Height="23" />
            <RowDefinition Height="20" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <Image x:Name="imgPoster" Source="{Binding ThumbNailHigh}" Grid.Row="0" HorizontalOptions="FillAndExpand"
VerticalOptions="FillAndExpand" Aspect="AspectFill" />

        <video:MediaElement x:Name="mdPlayer" Grid.Row="0" HorizontalOptions="FillAndExpand" VerticalOptions="FillAndExpand"
AreTransportControlsEnabled="True" IsVisible="False" />

        <ContentView Padding="8,0,8,0" Grid.Row="1">
            <Button x:Name="btnPlay" Text="PLAY VIDEO" FontSize="18" Clicked="btnPlay_Clicked" />
        </ContentView>

        <ContentView Padding="8,0,8,0" Grid.Row="2">
            <Label Text="{Binding VideoTitle}" TextColor="#336699" FontSize="18"  LineBreakMode="CharacterWrap"  />
        </ContentView>

        <ContentView  Grid.Row="3" Padding="8,0,8,0">
            <Label Text="{Binding VideoReleaseDate}" TextColor="#555555" FontSize="16" />
        </ContentView>

        <ContentView  Grid.Row="4" Padding="8,0,8,0">
            <Label Text="{Binding VideoDescription}" HorizontalOptions="EndAndExpand" TextColor="#555555" FontSize="14"
LineBreakMode="CharacterWrap"  />
        </ContentView>

        <StackLayout x:Name="stkBusy"
                     IsVisible="false"
                     BackgroundColor="Transparent"
                     Padding="12"
                     HorizontalOptions="Center"
                     VerticalOptions="Center"
                     Grid.RowSpan="5">
            <ActivityIndicator x:Name="indBusy" IsRunning="false" Color ="Black" />
            <Label Text="LOADING ..." HorizontalOptions="Center" TextColor="Black"/>
        </StackLayout>
    </Grid>

</ContentPage>
```

Step 14:

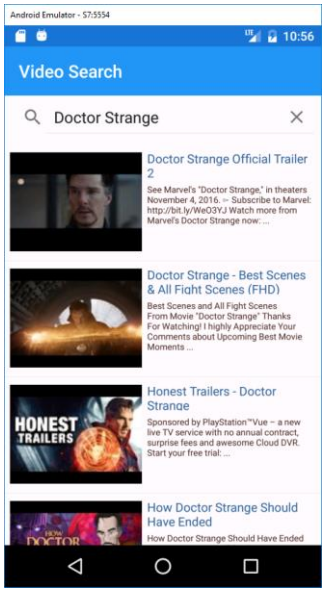
Add the following code to **VideoDetailPage.xaml.cs**:

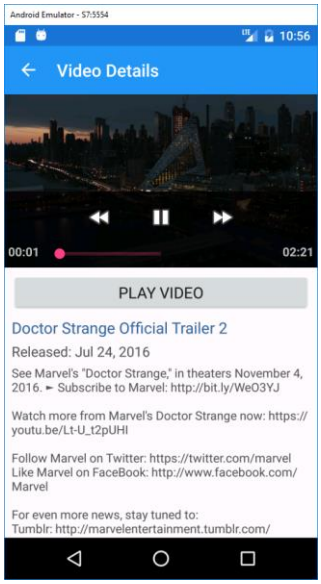
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

using Newtonsoft.Json;
using System.Net.Http;
using YoutubeExtractor;


namespace TubePlayer
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class VideoDetailPage : ContentPage
   {
      private string videoID;
      public YoutubeVideo2 TheVideo { get; set; }

      public VideoDetailPage()
      {
         InitializeComponent();
      }

      public VideoDetailPage(string youtubeID)
      {
         InitializeComponent();

         videoID = youtubeID;
      }

      protected override void OnAppearing()
      {
         base.OnAppearing();

         LoadVideo();
      }

      public async Task LoadVideo()
      {
         stkBusy.IsVisible = true;
         indBusy.IsRunning = true;

         var videoURL =
"https://www.googleapis.com/youtube/v3/videos?part=snippet&key=AIzaSyBni3RFS7oOIBTHhibN6age5ABtVtgLW7o&maxResults=20&id=" +
videoID;

         var httpClient = new HttpClient();
         var jsonResult = await httpClient.GetStringAsync(videoURL);
         var videosResult = JsonConvert.DeserializeObject<YoutubeVideos2>(jsonResult);
         TheVideo = videosResult.items.First();

         this.BindingContext = TheVideo;

         stkBusy.IsVisible = false;
         indBusy.IsRunning = false;
```

```
        }

        private void btnPlay_Clicked(object sender, EventArgs e)
        {
            PlayVideo();
        }

        public async void PlayVideo()
        {
            try
            {
                stkBusy.IsVisible = true;
                indBusy.IsRunning = true;


                var videoURL = "https://www.youtube.com/watch?v=" + TheVideo.VideoID;

                var videoLinks = DownloadUrlResolver.GetDownloadUrls(videoURL);

                var video720 = videoLinks.First(_i => _i.VideoType == VideoType.Mp4);

                //Code to play Video
                mdPlayer.Source = new Uri(video720.DownloadUrl);
                mdPlayer.IsVisible = true;
            }

            catch
            {
            }
            finally
            {
                stkBusy.IsVisible = false;
                indBusy.IsRunning = false;
            }
        }
    }
}
```

Step 15:

Run the application to see the result.



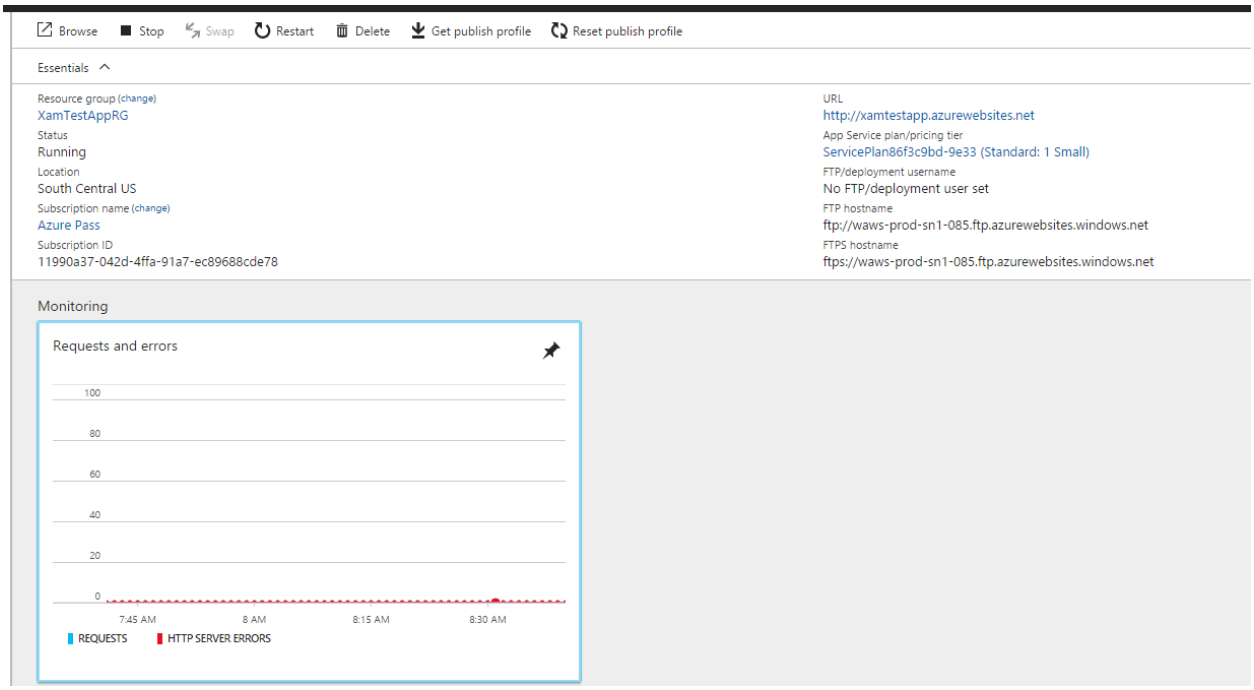


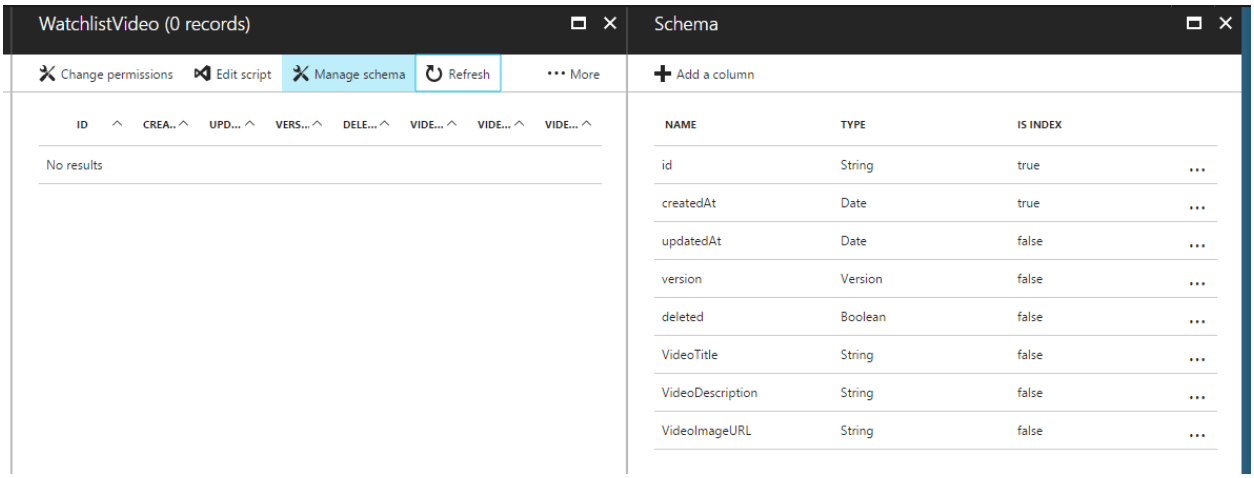
PART 2

(ADDING AZURE CONNECTIVITY)

Step 16:

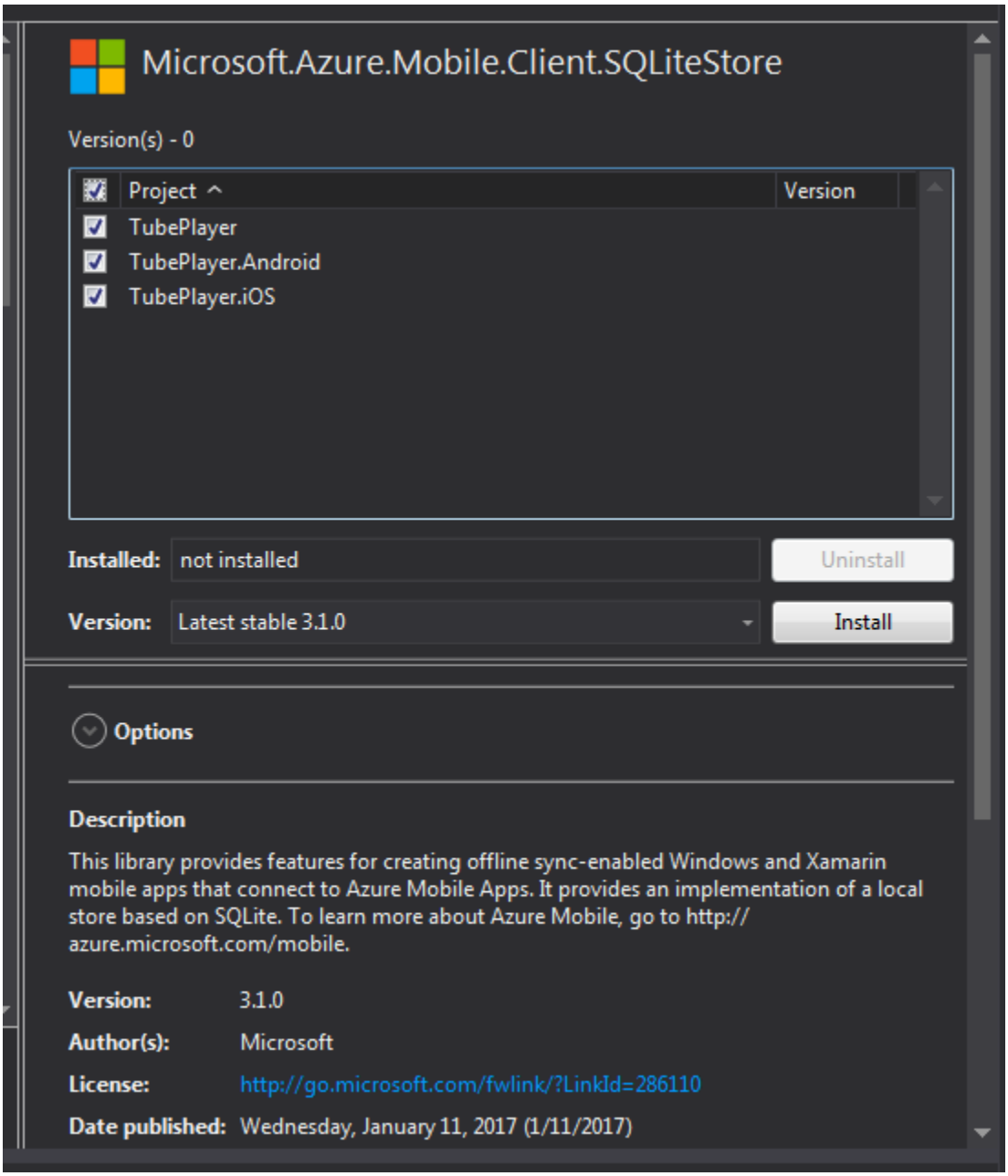
Create a new **Mobile App** (give it the name of your choice) and a new **Easy Table** called “WatchlistVideo”.





Step 17:

Add the **Azure Mobile SQLiteStore** NuGet package to the **whole solution**.



Step 18:

Initialize the Azure Mobile Client.

For iOS, add the following code to the **FinishedLaunching** method of the **AppDelegate** class:

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options)
{
    global::Xamarin.Forms.Forms.Init();

    Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();

    LoadApplication(new App());

    return base.FinishedLaunching(app, options);
}
```

For Android, add the following to the **OnCreate** method of your **MainActivity**:

```
protected override void OnCreate(Bundle bundle)
{
    TabLayoutResource = Resource.Layout.Tabbar;
    ToolbarResource = Resource.Layout.Toolbar;

    base.OnCreate(bundle);

    global::Xamarin.Forms.Forms.Init(this, bundle);

    Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();

    LoadApplication(new App());
}
```

Step 19:

Add the following class to the **DataModels.cs** file:

```
public class WatchlistVideo
{
    public string VideoTitle { get; set; }
    public string VideoDescription { get; set; }
    public string VideoImageURL { get; set; }

    [Newtonsoft.Json.JsonProperty("Id")]
    public string Id { get; set; }

    [Microsoft.WindowsAzure.MobileServices.Version]
    public string AzureVersion { get; set; }
}
```

Step 20:

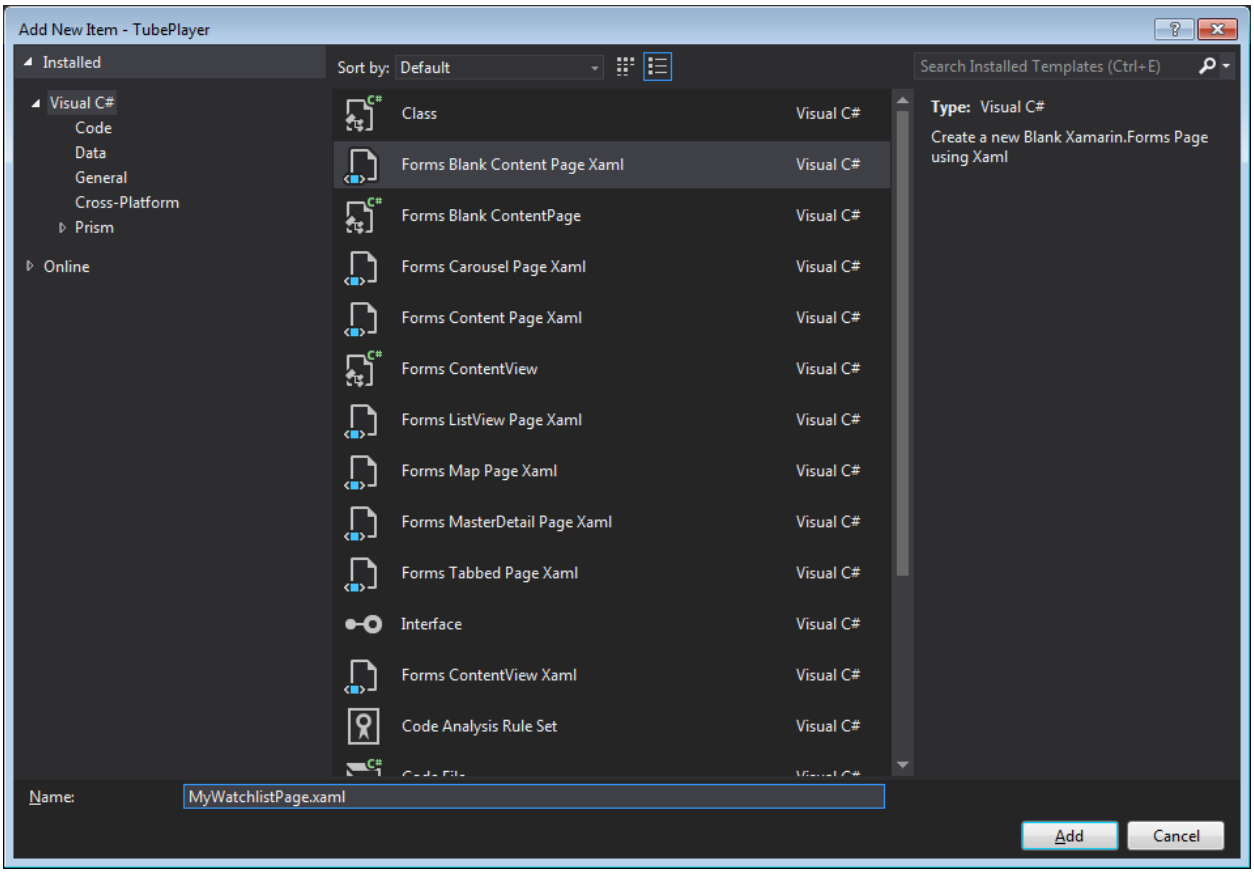
Create a new class called **AzureDataService.cs** and add the following code:

```
using Microsoft.WindowsAzure.MobileServices;
using Microsoft.WindowsAzure.MobileServices.SQLiteStore;
using Microsoft.WindowsAzure.MobileServices.Sync;
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TubePlayer
{
  public class AzureDataService
  {
    public MobileServiceClient MobileService { get; set; }

    IMobileServiceSyncTable<WatchlistVideo> watchlistTable;


    public async Task Initialize()
    {
      //Create our client
      MobileService = new MobileServiceClient("https://<YouAppService>.azurewebsites.net");

      var path = "videos.db";
      path = Path.Combine(MobileServiceClient.DefaultDatabasePath, path);

      //setup our local sqlite store and intialize our table
      var store = new MobileServiceSQLiteStore(path);

      store.DefineTable<WatchlistVideo>();

      await MobileService.SyncContext.InitializeAsync(store);

      //Get our sync table that will call out to azure
      watchlistTable = MobileService.GetSyncTable<WatchlistVideo>();
    }

    public async Task<List<WatchlistVideo>> GetWatchlistVideos()
    {
      //Initialize & Sync
      await Initialize();
      await SyncWatchlist();
      return await watchlistTable.ToListAsync();
    }

    public async Task AddVideoToWatchlist(WatchlistVideo video)
    {
      await Initialize();

      await watchlistTable.InsertAsync(video);

      //Synchronize Watchlist
      await SyncWatchlist();
    }

    public async Task SyncWatchlist()
    {
```

```
        //pull down all latest changes and then push current coffees up
        await watchlistTable.PullAsync("allWatchlist", watchlistTable.CreateQuery());
        await MobileService.SyncContext.PushAsync();
      }
    }
}
```

Step 21:

Add a new **Blank Forms Blank Content Page Xaml** called " **MyWatchlistPage.xaml**":



Step 22:

Add the following layout to the " **MyWatchlistPage.xaml**" page:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         x:Class="TubePlayer.MyWatchlistPage"
         Title="My Watchlist">
  <Grid HorizontalOptions="FillAndExpand" VerticalOptions="FillAndExpand">
    <!-- Video ListView will come here -->
    <ListView x:Name="lstVideos"
           RowHeight="132">
      <ListView.ItemTemplate>
        <DataTemplate>
          <ViewCell>
            <Grid Padding="6,6,6,6">
              <Grid.ColumnDefinitions>
                <ColumnDefinition Width="150" />
                <ColumnDefinition Width="*" />
              </Grid.ColumnDefinitions>

              <Grid.RowDefinitions>
                <RowDefinition Height="32" />
                <RowDefinition Height="80" />
                <RowDefinition Height="8" />
              </Grid.RowDefinitions>

              <Image Source="{Binding VideoImageURL}" Grid.Column="0" Grid.Row="0" Grid.RowSpan="2" />
              <Label Text="{Binding VideoTitle}" TextColor="#336699" FontSize="14"
                   Grid.Column="1" Grid.Row="0" />
              <Label Text="{Binding VideoDescription}" HorizontalOptions="EndAndExpand"
                   TextColor="#503026" FontSize="10" Grid.Column="1" Grid.Row="1" />
            </Grid>
          </ViewCell>
        </DataTemplate>
      </ListView.ItemTemplate>
    </ListView>


    <!-- Progress Bar - Hidden -->
    <StackLayout x:Name="stkBusy"
           IsVisible="false"
           BackgroundColor="Transparent"
           Padding="12"
           HorizontalOptions="Center"
           VerticalOptions="Center"
           Grid.RowSpan="2">
      <ActivityIndicator x:Name="indBusy" IsRunning="false" Color ="Black" />
      <Label Text="LOADING ..." HorizontalOptions="Center" TextColor="Black"/>
    </StackLayout>
  </Grid>

</ContentPage>
```

Step 23:

Add the following code behind in the " **MyWatchlistPage.cs**" file:

```
protected override void OnAppearing()
{
   base.OnAppearing();
   LoadMyWatchlist();
}

public async Task LoadMyWatchlist()
{
   stkBusy.IsVisible = true;
   indBusy.IsRunning = true;

   var azureService = new AzureDataService();

   var videosResult = await azureService.GetWatchlistVideos();

   lstVideos.ItemsSource = videosResult;

   stkBusy.IsVisible = false;
   indBusy.IsRunning = false;
}
```

Step 24:

Add a new button in " **VideoDetailPage.xaml**" and "**VideoDetailPage.cs**":

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:video="clr-namespace:InTheHand.Forms;assembly=InTheHand.Forms"
        x:Class="TubePlayer.VideoDetailPage"
        Title="Video Details">

  <Grid HorizontalOptions="FillAndExpand" VerticalOptions="FillAndExpand">
    <Grid.RowDefinitions>
      <RowDefinition Height="200" />
      <RowDefinition Height="45" />
      <RowDefinition Height="45" />
      <RowDefinition Height="23" />
      <RowDefinition Height="20" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>

    <Image x:Name="imgPoster" Source="{Binding ThumbNailHigh}" Grid.Row="0" HorizontalOptions="FillAndExpand"
VerticalOptions="FillAndExpand" Aspect="AspectFill" />

    <video:MediaElement x:Name="mdPlayer" Grid.Row="0" HorizontalOptions="FillAndExpand" VerticalOptions="FillAndExpand"
AreTransportControlsEnabled="True" IsVisible="False" />

    <ContentView Padding="8,0,8,0" Grid.Row="1">
      <Button x:Name="btnPlay" Text="PLAY VIDEO" FontSize="18" Clicked="btnPlay_Clicked" />
    </ContentView>

    <ContentView Padding="8,0,8,0" Grid.Row="2">
      <Button x:Name="btnAdd" Text="ADD TO WATCHLIST" FontSize="18" Clicked="btnAdd_Clicked" />
    </ContentView>
```

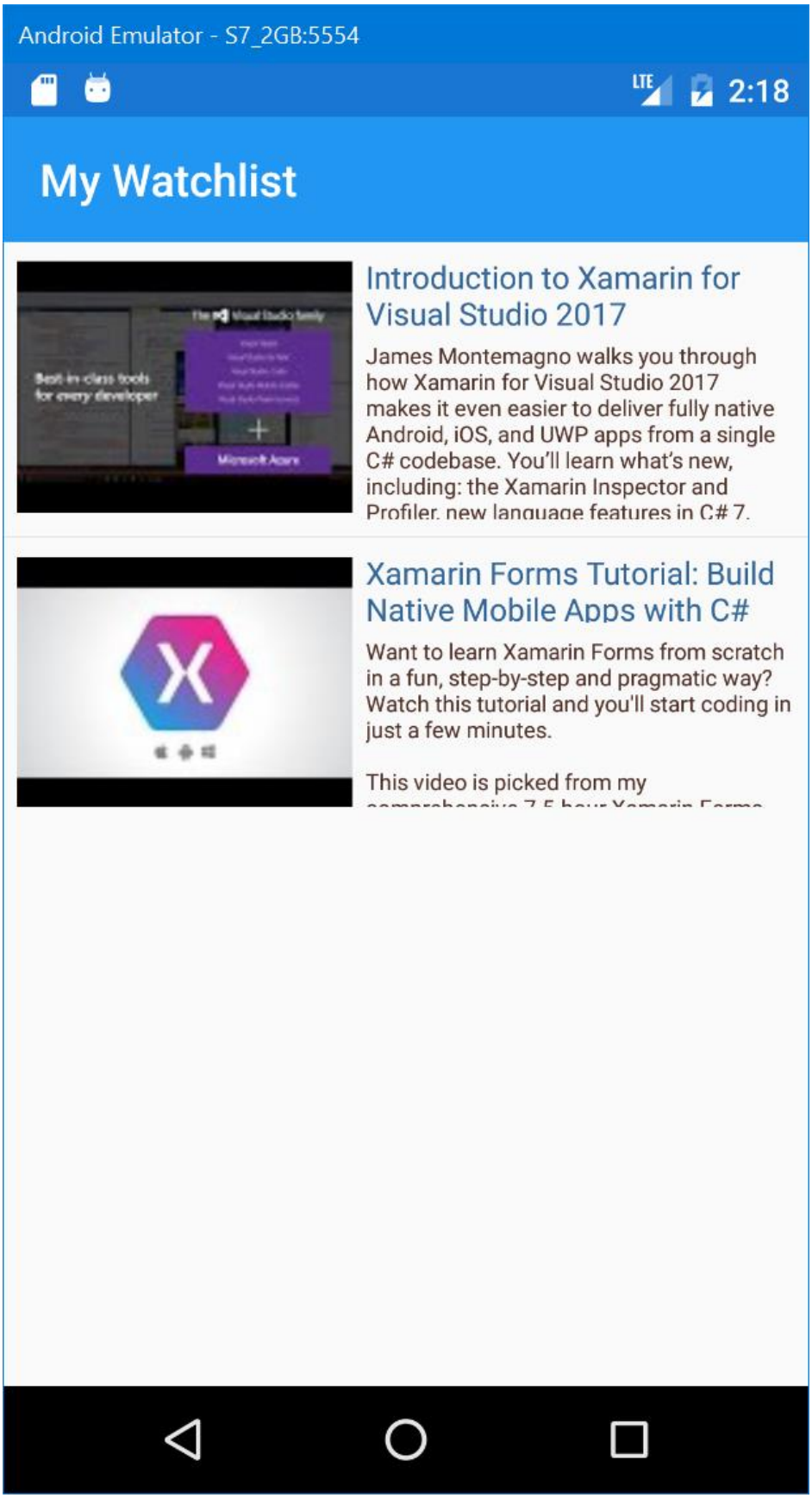
```
    <ContentView Padding="8,0,8,0" Grid.Row="3">
      <Label Text="{Binding VideoTitle}" TextColor="#336699" FontSize="18"  LineBreakMode="CharacterWrap"  />
    </ContentView>

    <ContentView  Grid.Row="4" Padding="8,0,8,0">
      <Label Text="{Binding VideoReleaseDate}" TextColor="#555555" FontSize="16" />
    </ContentView>

    <ContentView  Grid.Row="5" Padding="8,0,8,0">
      <Label Text="{Binding VideoDescription}" HorizontalOptions="EndAndExpand" TextColor="#555555" FontSize="14"
LineBreakMode="CharacterWrap"  />
    </ContentView>

    <StackLayout x:Name="stkBusy"
                 IsVisible="false"
                 BackgroundColor="Transparent"
                 Padding="12"
                 HorizontalOptions="Center"
                 VerticalOptions="Center"
                 Grid.RowSpan="6">
      <ActivityIndicator x:Name="indBusy" IsRunning="false" Color ="Black" />
      <Label Text="LOADING ..." HorizontalOptions="Center" TextColor="Black"/>
    </StackLayout>
  </Grid>

</ContentPage>
```

```
    private void btnAdd_Clicked(object sender, EventArgs e)
    {
      AddVideo();
    }

    public async void AddVideo()
    {
      stkBusy.IsVisible = true;
      indBusy.IsRunning = true;

      var videoToAdd = new WatchlistVideo();
      videoToAdd.VideoTitle = TheVideo.VideoTitle;
      videoToAdd.VideoDescription = TheVideo.VideoDescription;
      videoToAdd.VideoImageURL = TheVideo.ThumbNail;

      var azureService = new AzureDataService();
      await azureService.AddVideoToWatchlist(videoToAdd);

      stkBusy.IsVisible = false;
      indBusy.IsRunning = false;
    }
```

Step 25:

Add some videos to Watchlist. In **App.xaml.cs** code-behind, make the following change to the constructor. Run the application to see the result.

```
    public App()
    {
      InitializeComponent();
      MainPage = new NavigationPage(new TubePlayer.MyWatchlistPage());
    }
```


Android Emulator - S7_2GB:5554
LTE
2:18
My Watchlist
Best-in-class tools for every developer
Introduction to Xamarin for Visual Studio 2017
James Montemagno walks you through how Xamarin for Visual Studio 2017 makes it even easier to deliver fully native Android, iOS, and UWP apps from a single C# codebase. You'll learn what's new, including: the Xamarin Inspector and Profiler, new language features in C# 7.
Xamarin Forms Tutorial: Build Native Mobile Apps with C#
Want to learn Xamarin Forms from scratch in a fun, step-by-step and pragmatic way? Watch this tutorial and you'll start coding in just a few minutes.
This video is picked from my