

¿Como utilizar el MasterDetailPage en Xamarin?

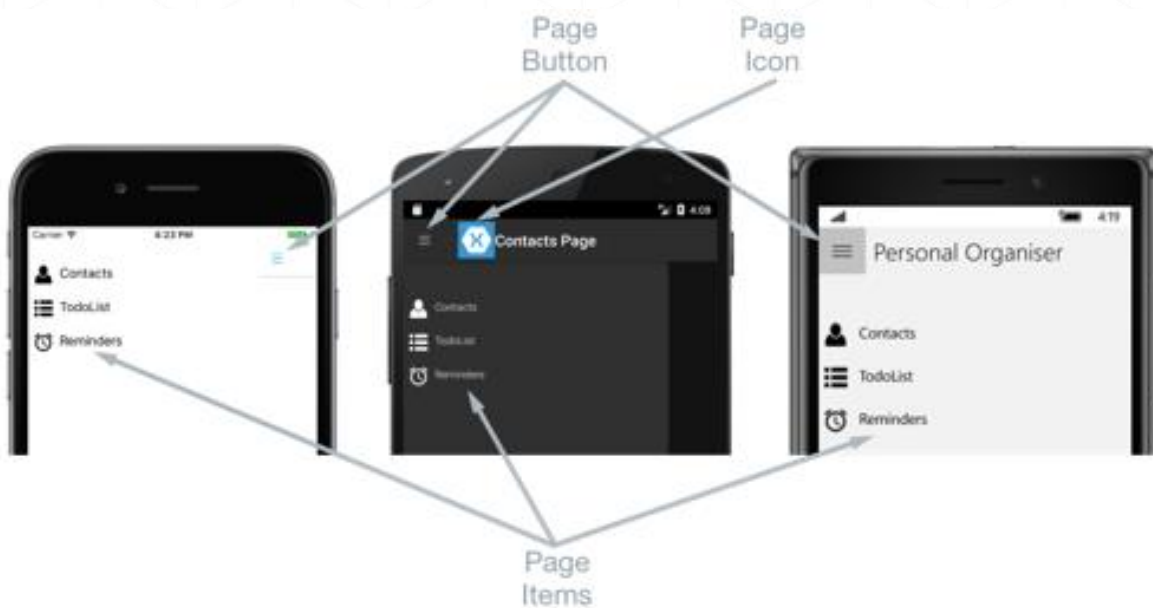


¿Antes de iniciar que es un MasterDetailPage?

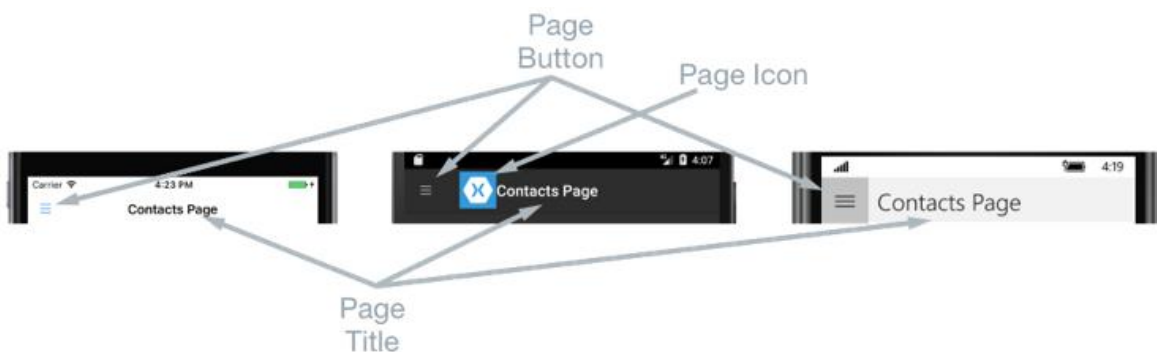
Es una página que administra dos páginas relacionadas de información, una página maestra que presenta elementos y una página de detalles que presenta detalles sobre los elementos de la página maestra.

Normalmente, una página maestra presenta una lista de elementos, como se muestra en las siguientes capturas de pantalla:

- La ubicación de la lista de elementos es idéntica en cada plataforma; al seleccionar uno de los elementos, se le lleva a la página de detalles correspondiente.



- La MasterDetailPage tiene variaciones dependiendo de la Plataforma




Ya que conoces sus funciones manos al código.

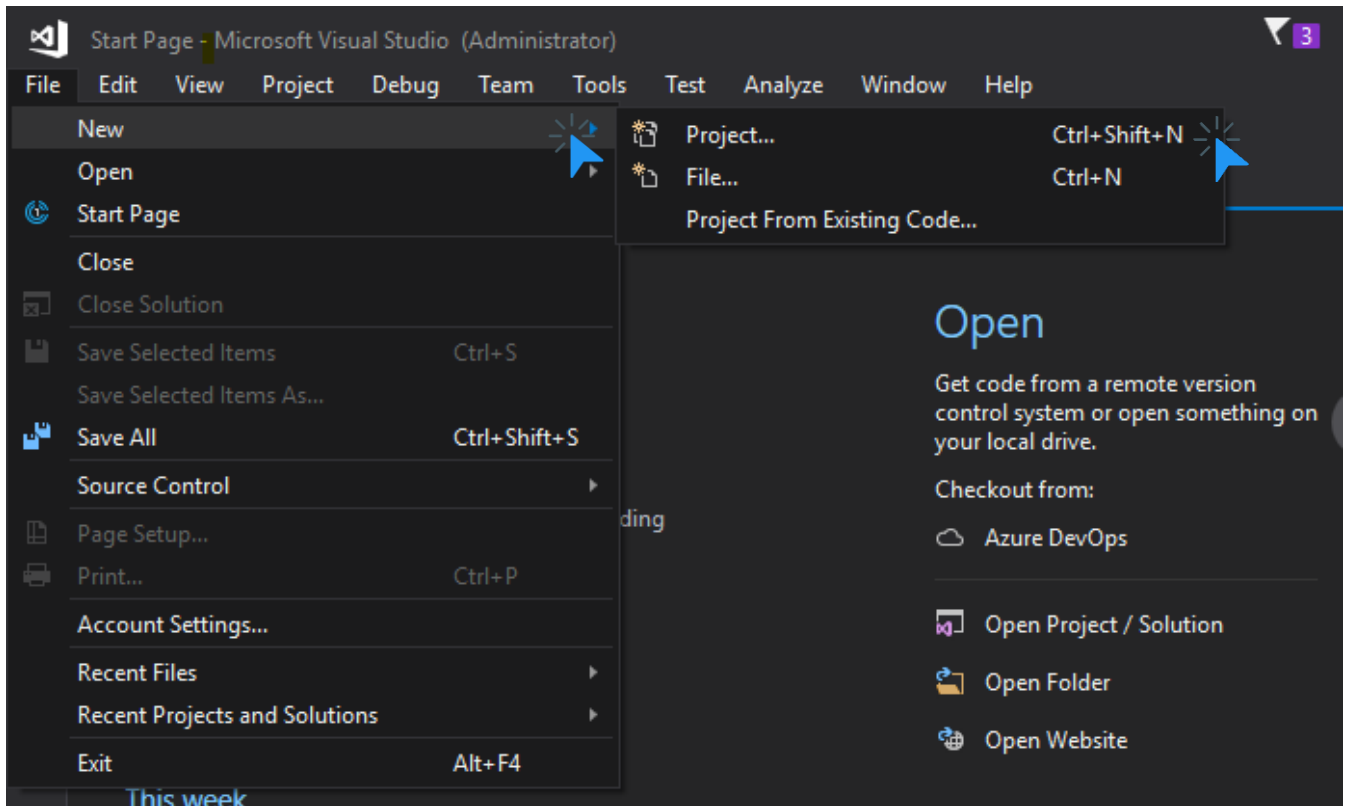


Pasos para crear la MasterDetailPage

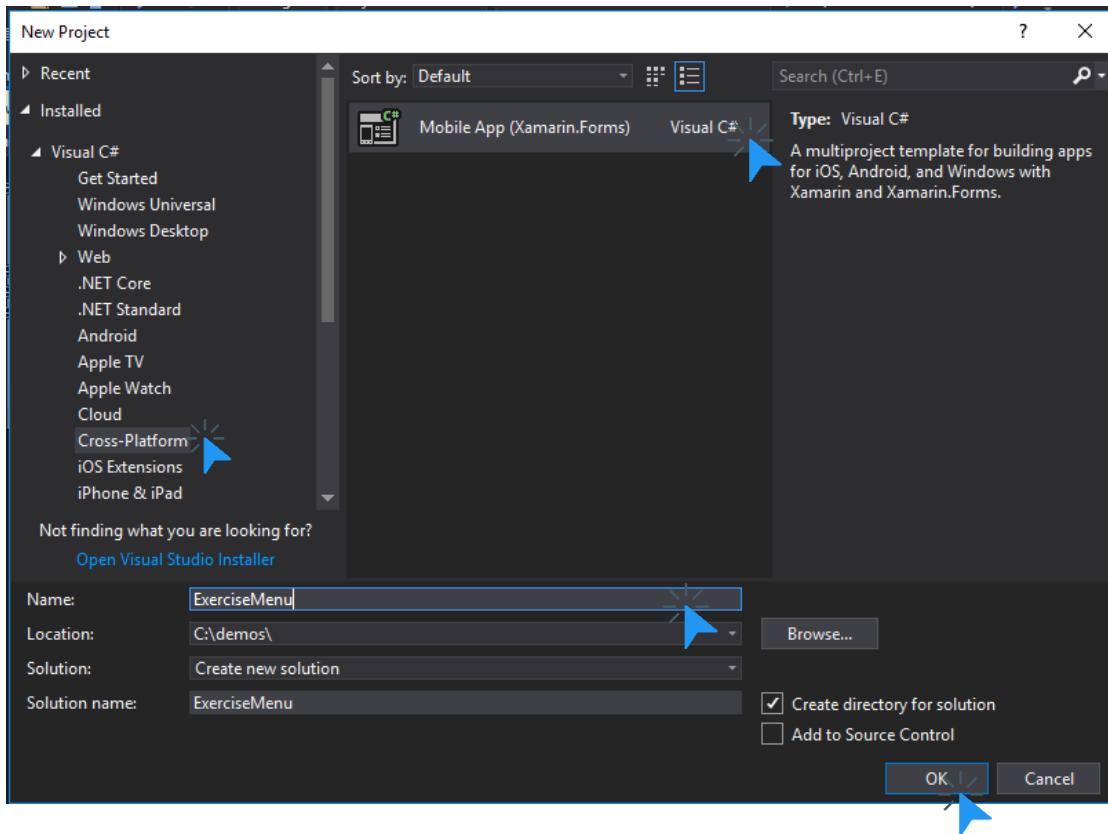
Existen distintas formas para crear una MasterDetailPage a continuación mostraremos 2 forma para hacerlo.

Primera forma

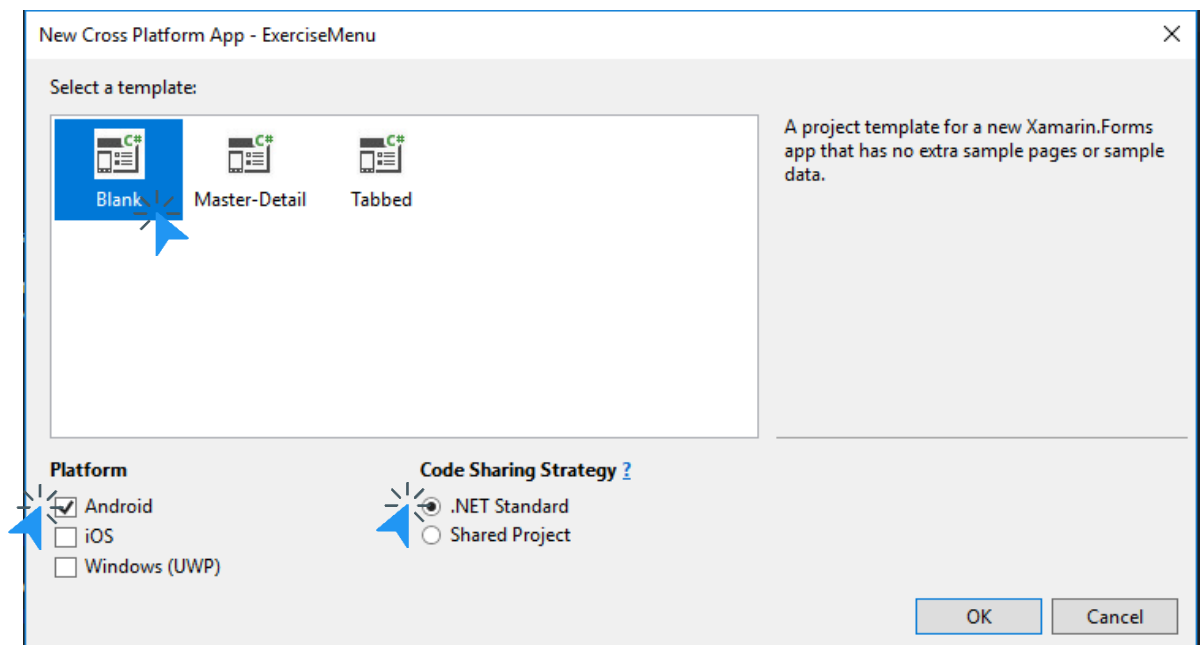
-  En la primera forma veremos su funcionamiento, iniciamos Visual Studio y creamos un nuevo proyecto.



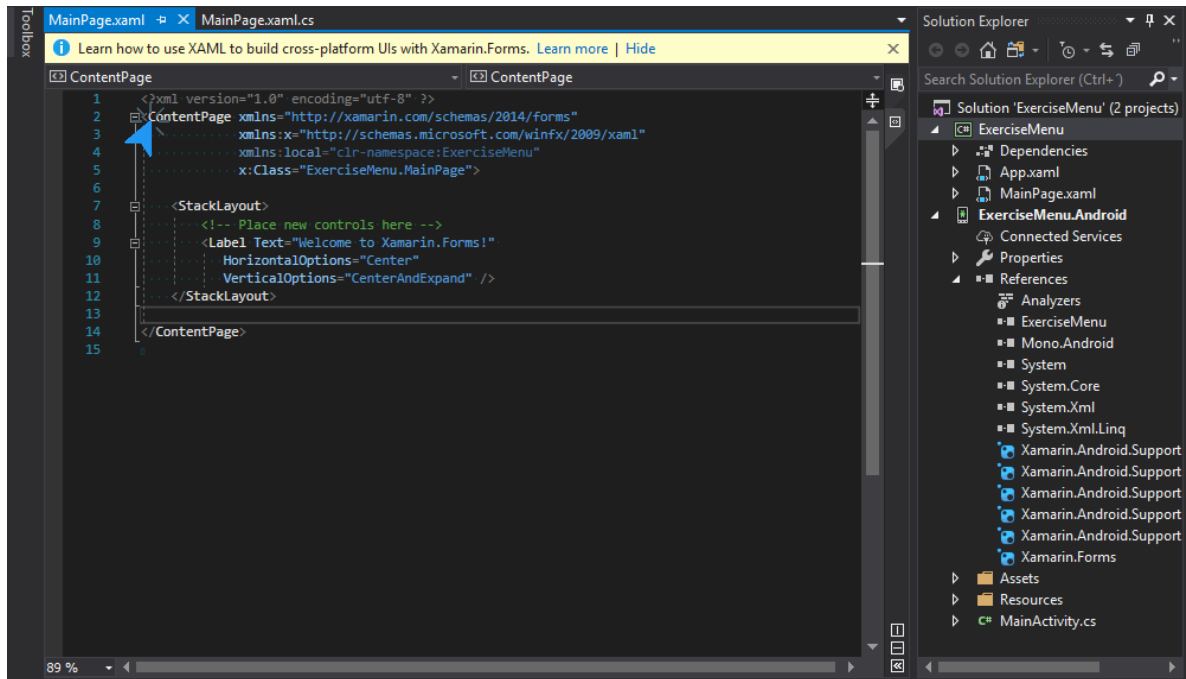
Seleccionamos Cross-Platform, lo nombramos como **ExersiceMenu** y damos OK.



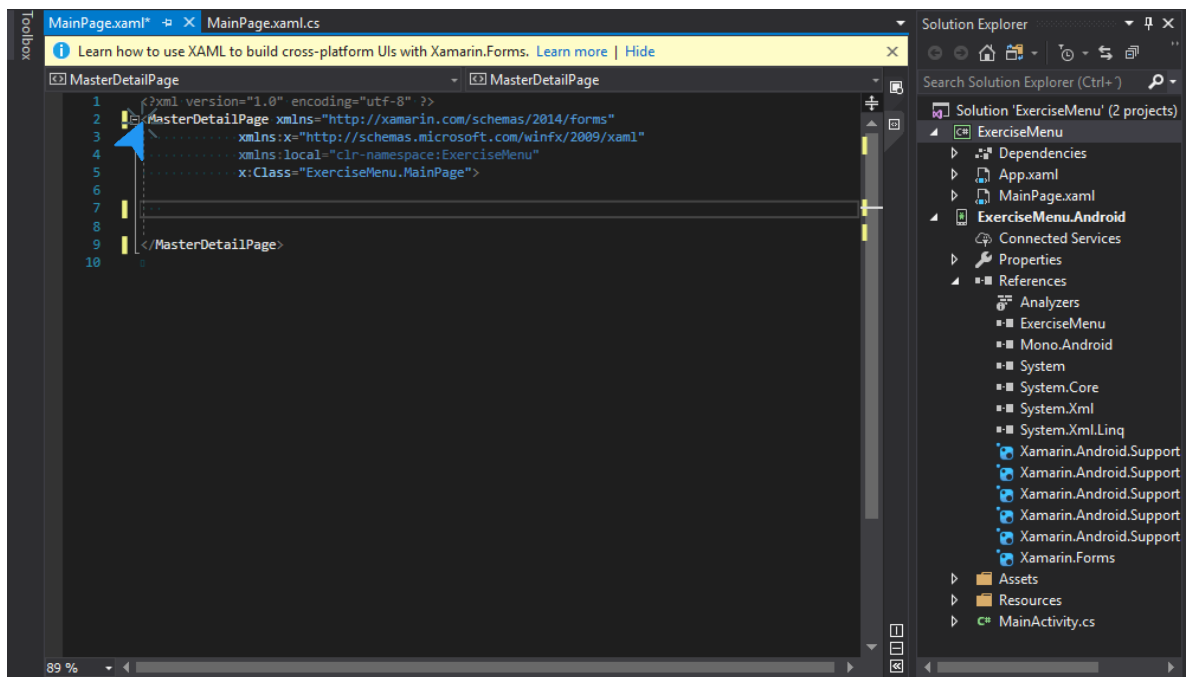
Seleccionamos el modelo Blank, plataforma Android, como .Net Standard y damos OK.



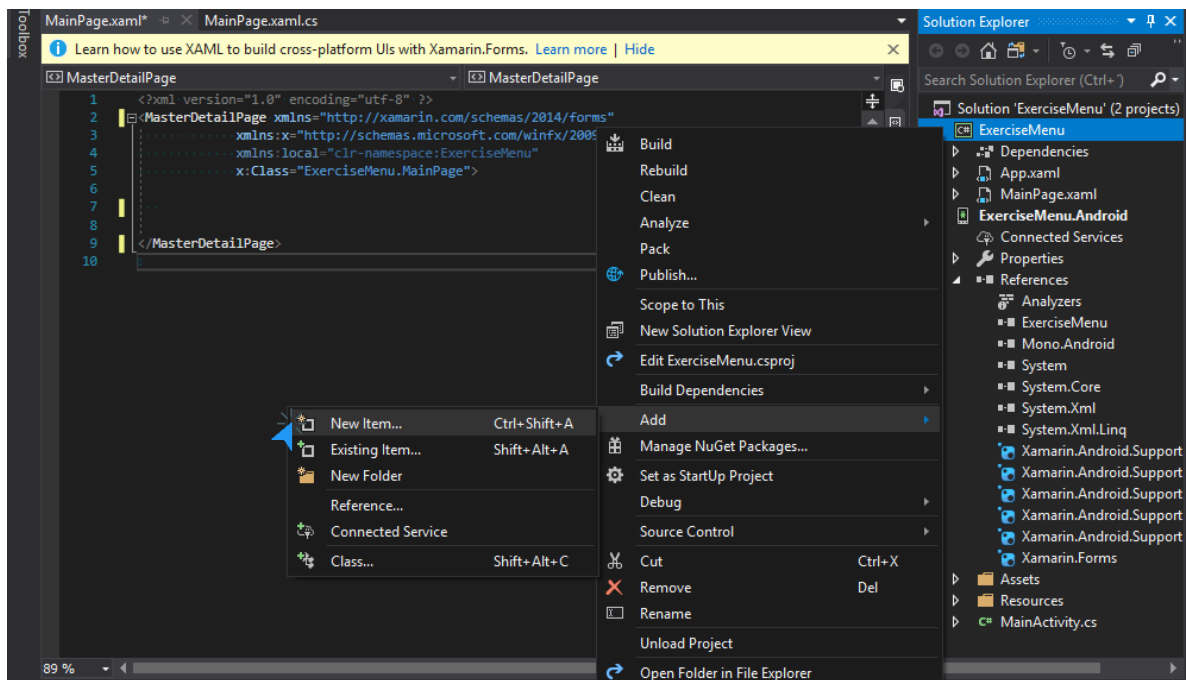
- El primer paso para implementar la **MasterDetailPage** será modificar la **MainPage** la cual dejará de ser una ContentPage y pasará a ser una MasterDetailPage.



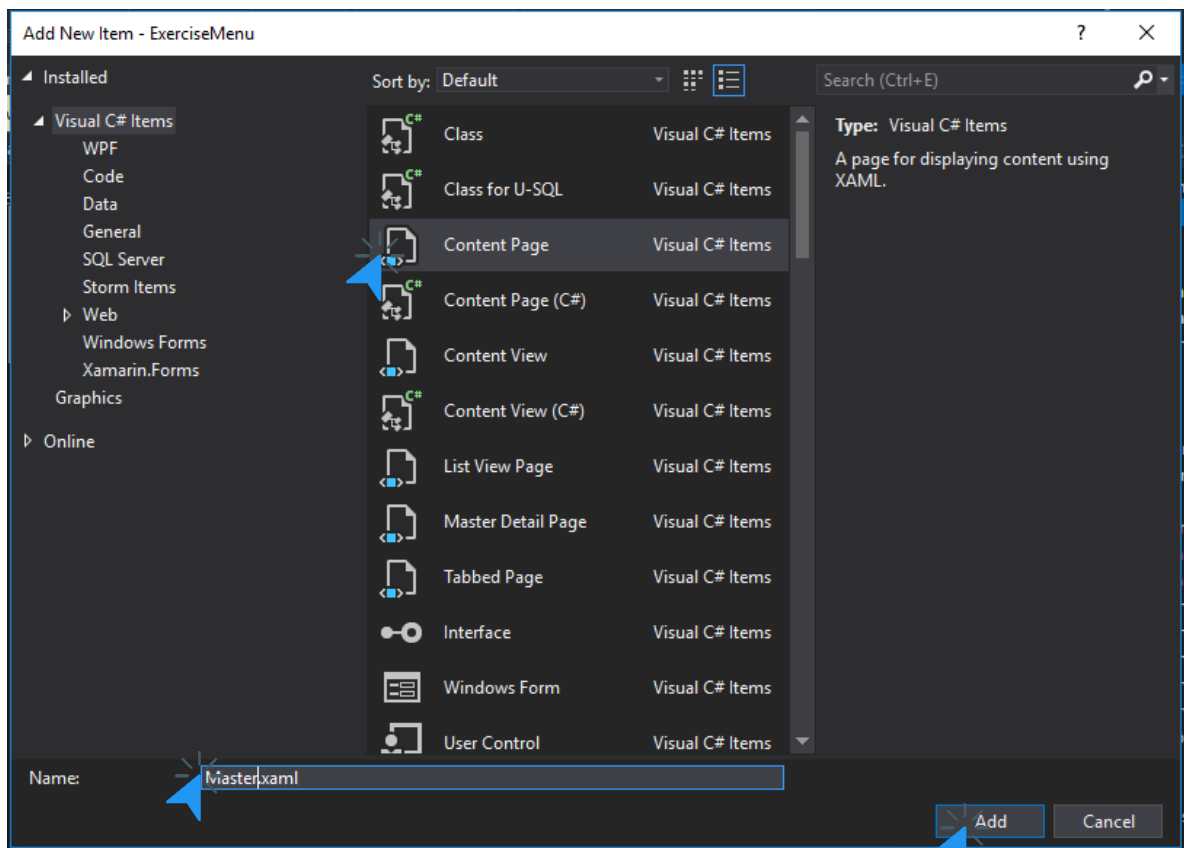
- Eliminamos el StackLayout y cambiamos la ContentPage por MasterDetailPage, quedando así.



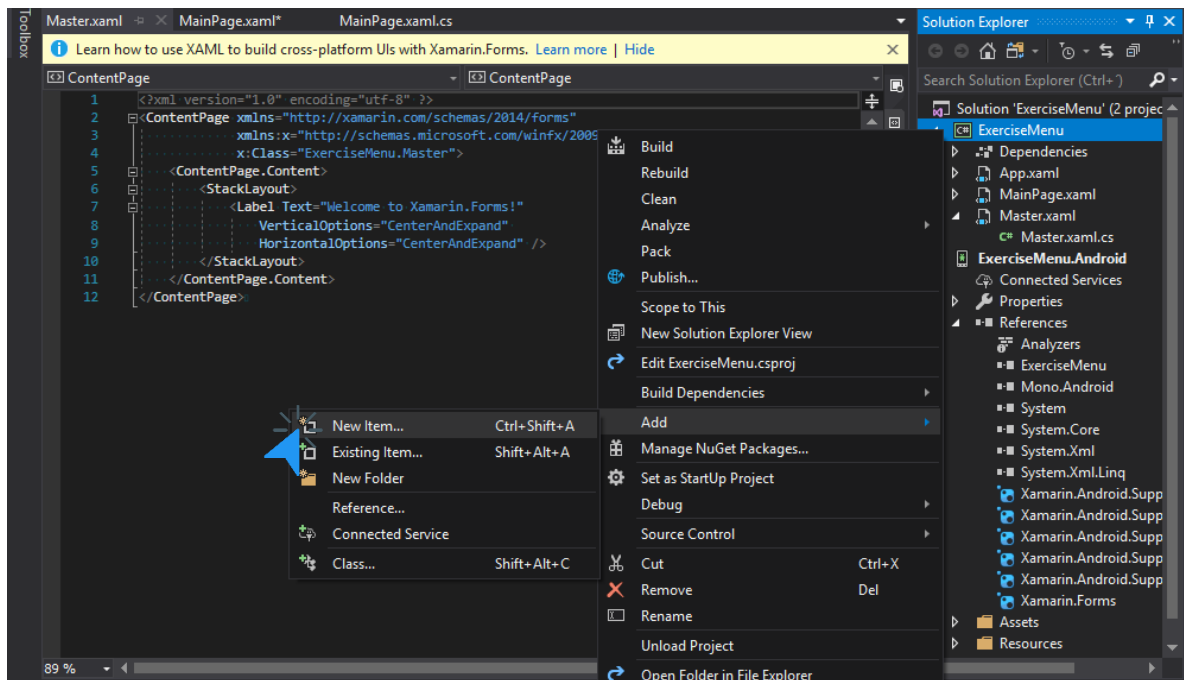
➤ Agregamos un New Ítem.



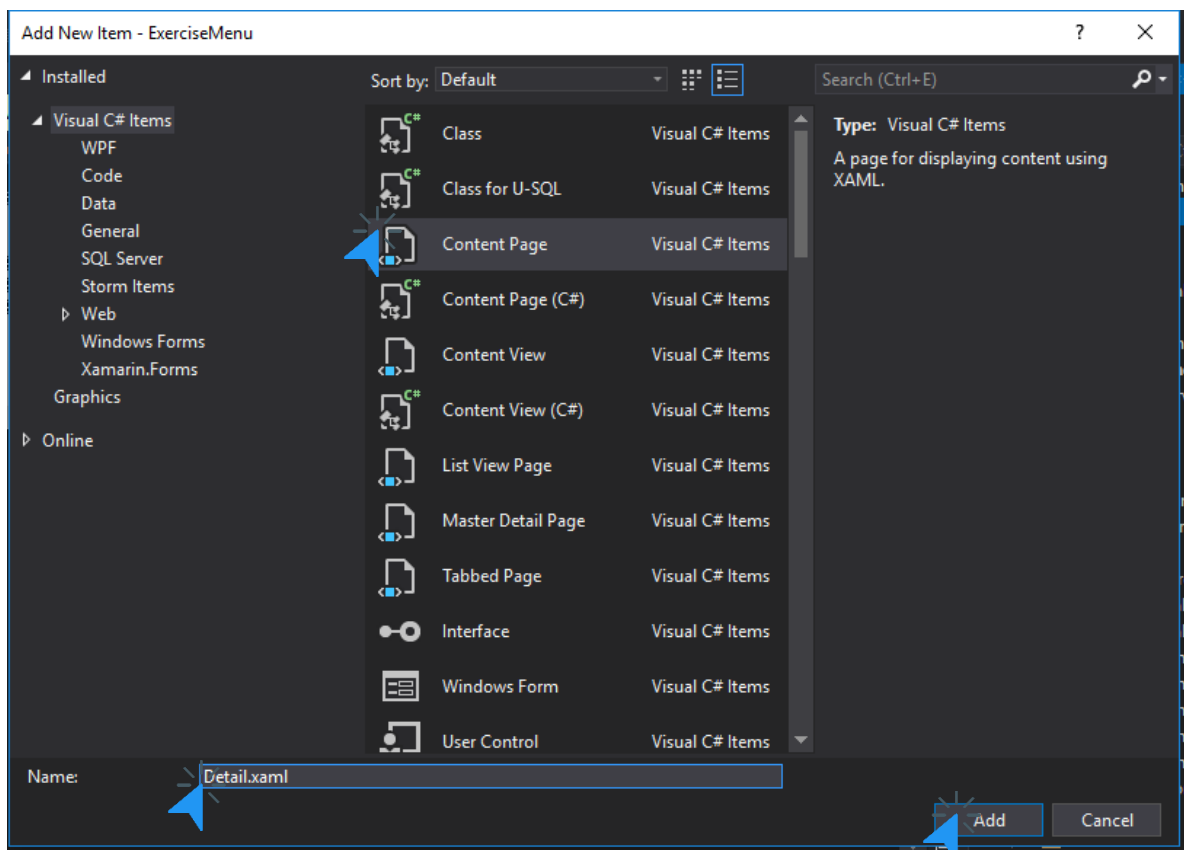
➤ El cual será una Content Page nombrada como **Master**.



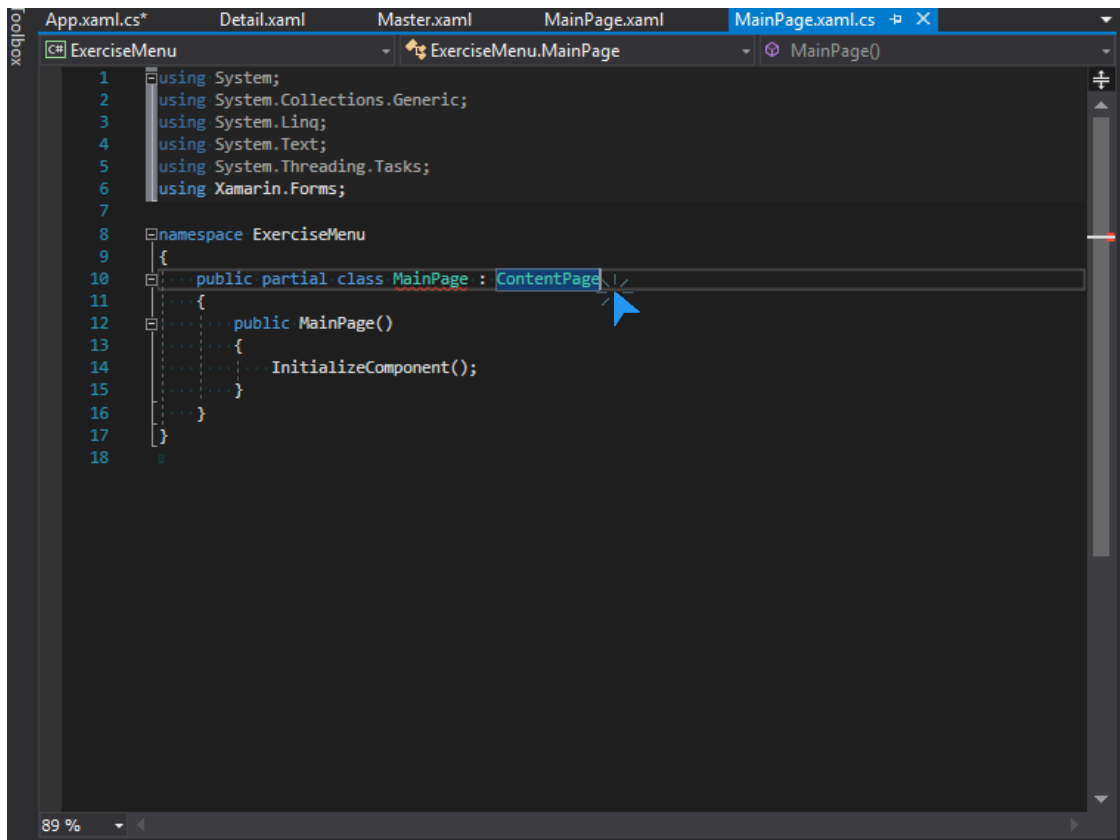
➡ Agregamos otro New Item.



➡ El cual será una Content Page nombrada como **Detail**.



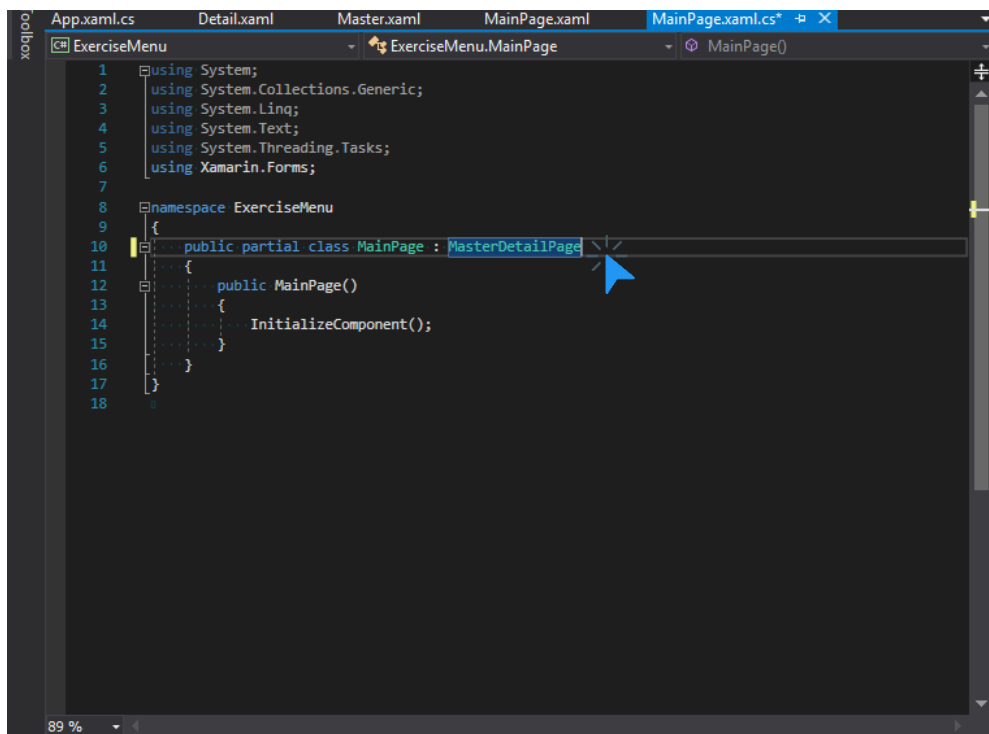
✚ Nos dirigimos a la **MainPage** cambiamos su ContentPage a MasterDetailPage.



The screenshot shows the Visual Studio IDE with the file **MainPage.xaml.cs** open. The code defines a partial class **MainPage** that inherits from **ContentPage**. A blue arrow points to the **ContentPage** text in the inheritance declaration on line 10.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using Xamarin.Forms;
7
8 namespace ExerciseMenu
9 {
10     public partial class MainPage : ContentPage
11     {
12         public MainPage()
13         {
14             InitializeComponent();
15         }
16     }
17 }
18
```

✚ Quedará así.

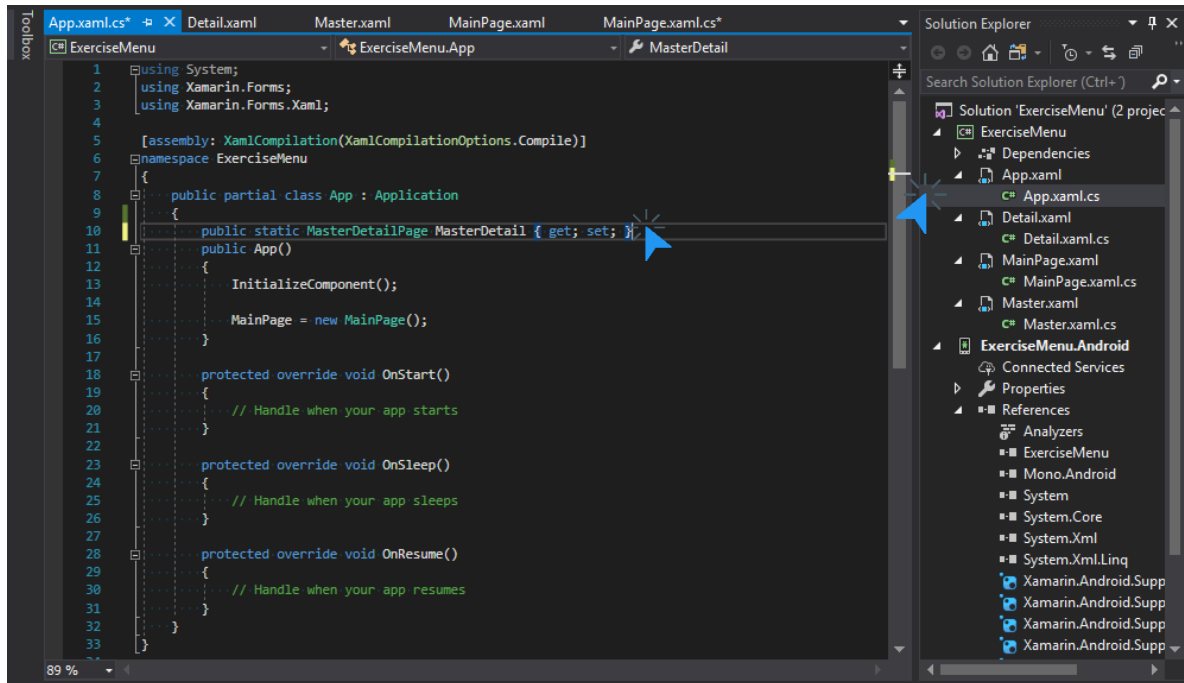


The screenshot shows the same Visual Studio IDE with **MainPage.xaml.cs** open. The code is identical to the previous one, but the inheritance declaration on line 10 now shows **MainPage** inheriting from **MasterDetailPage**. A blue arrow points to the **MasterDetailPage** text.

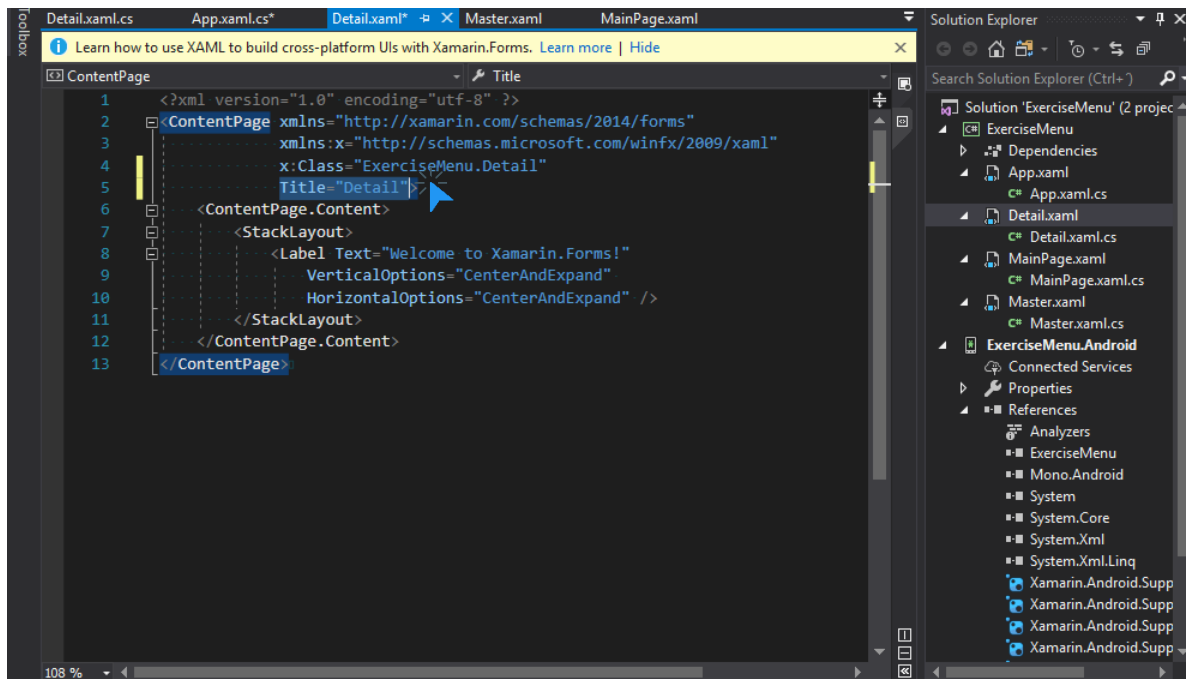
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using Xamarin.Forms;
7
8 namespace ExerciseMenu
9 {
10     public partial class MainPage : MasterDetailPage
11     {
12         public MainPage()
13         {
14             InitializeComponent();
15         }
16     }
17 }
18
```



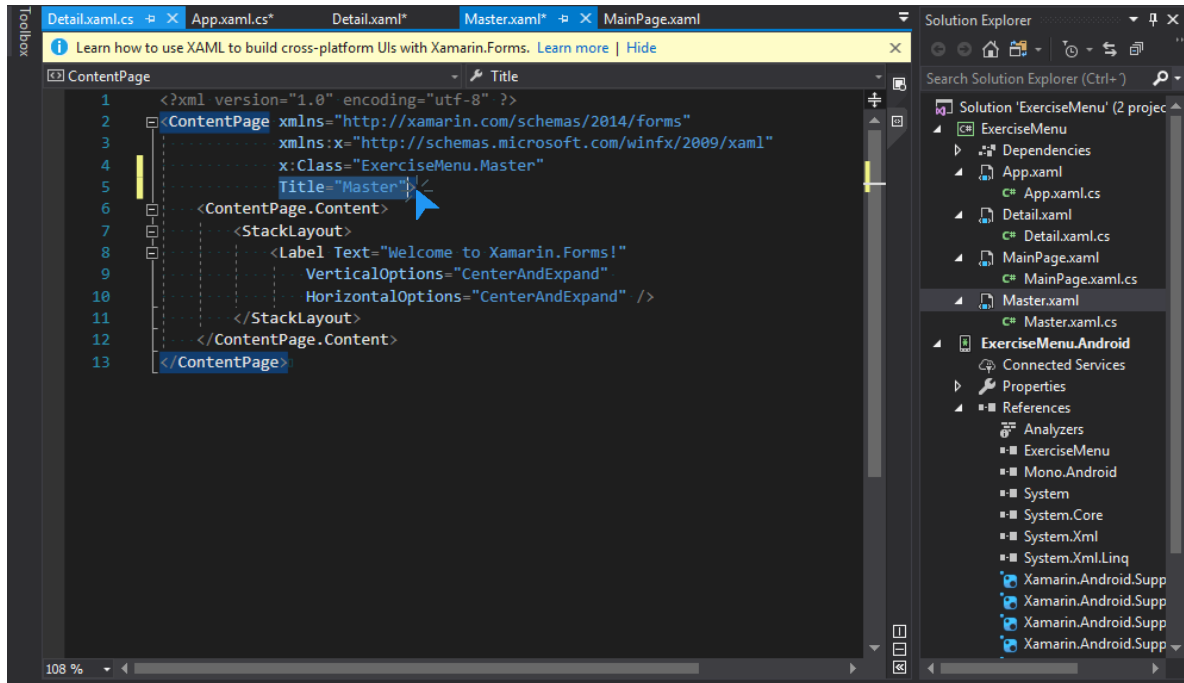
En **App.xaml.cs** agregamos una propiedad estática.



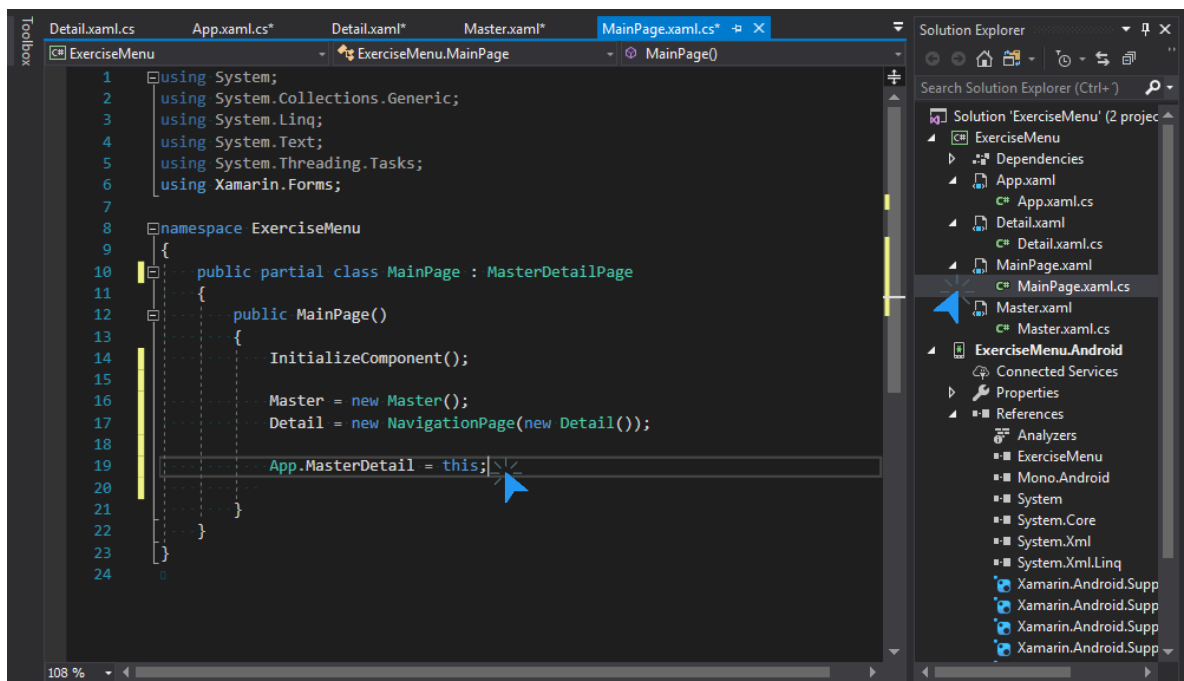
En la **ContentPage Detail**, agregamos un Title como Detail.



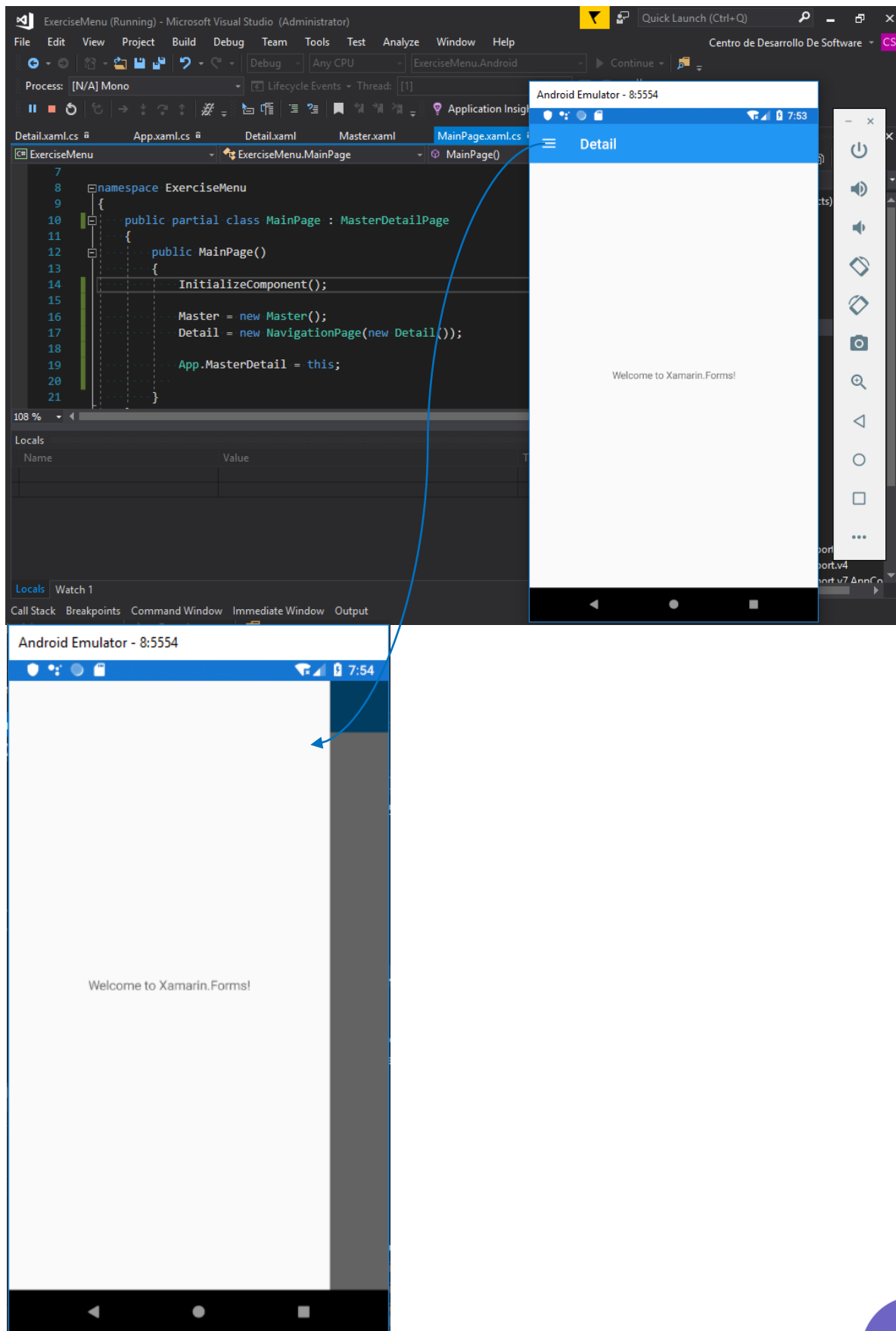
De igual forma en la **ContentPage Master** un Title como Master.



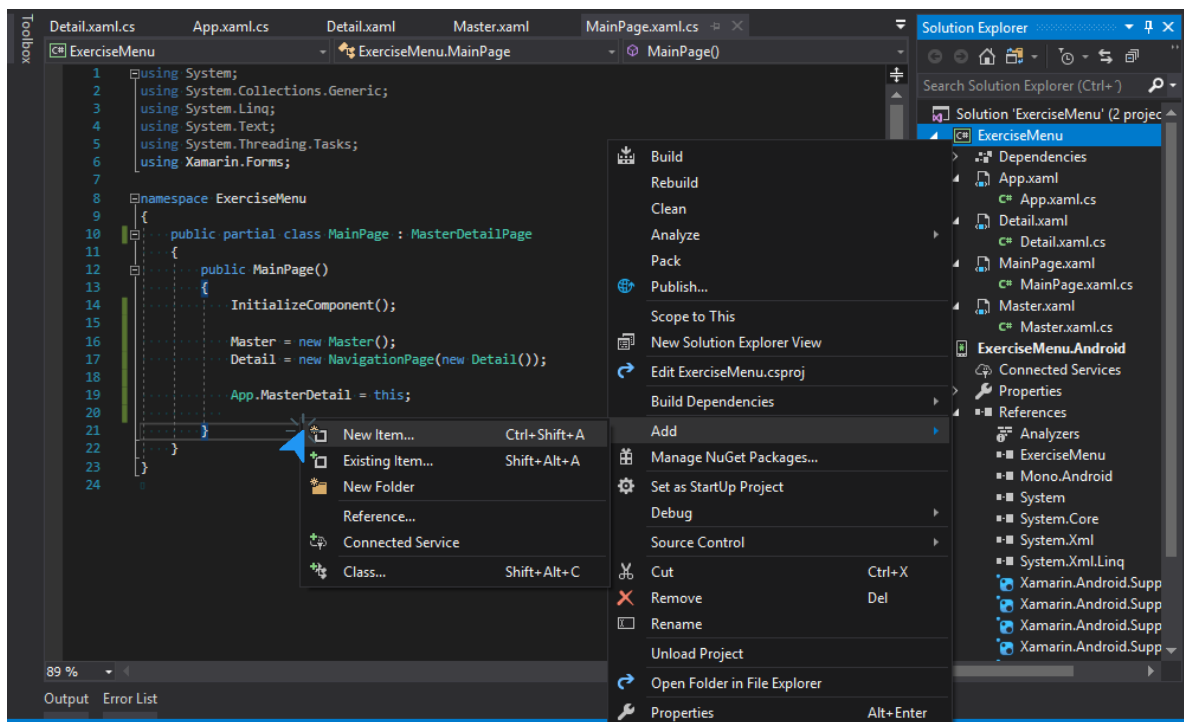
En **Mainpage.xaml.cs** definiremos la Master y la Detail.



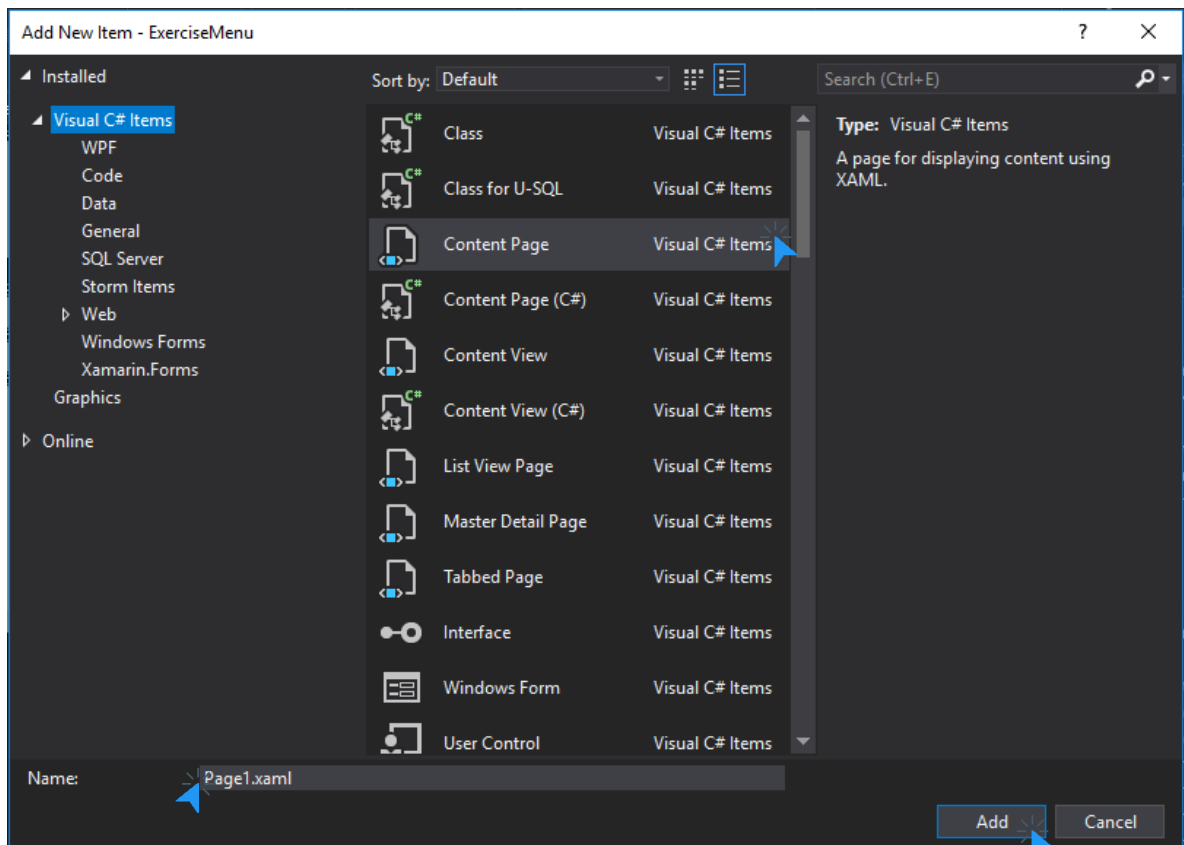
🚦 Solo con estos pasos ya es posible visualizar el funcionamiento de la MasterDetailPage.



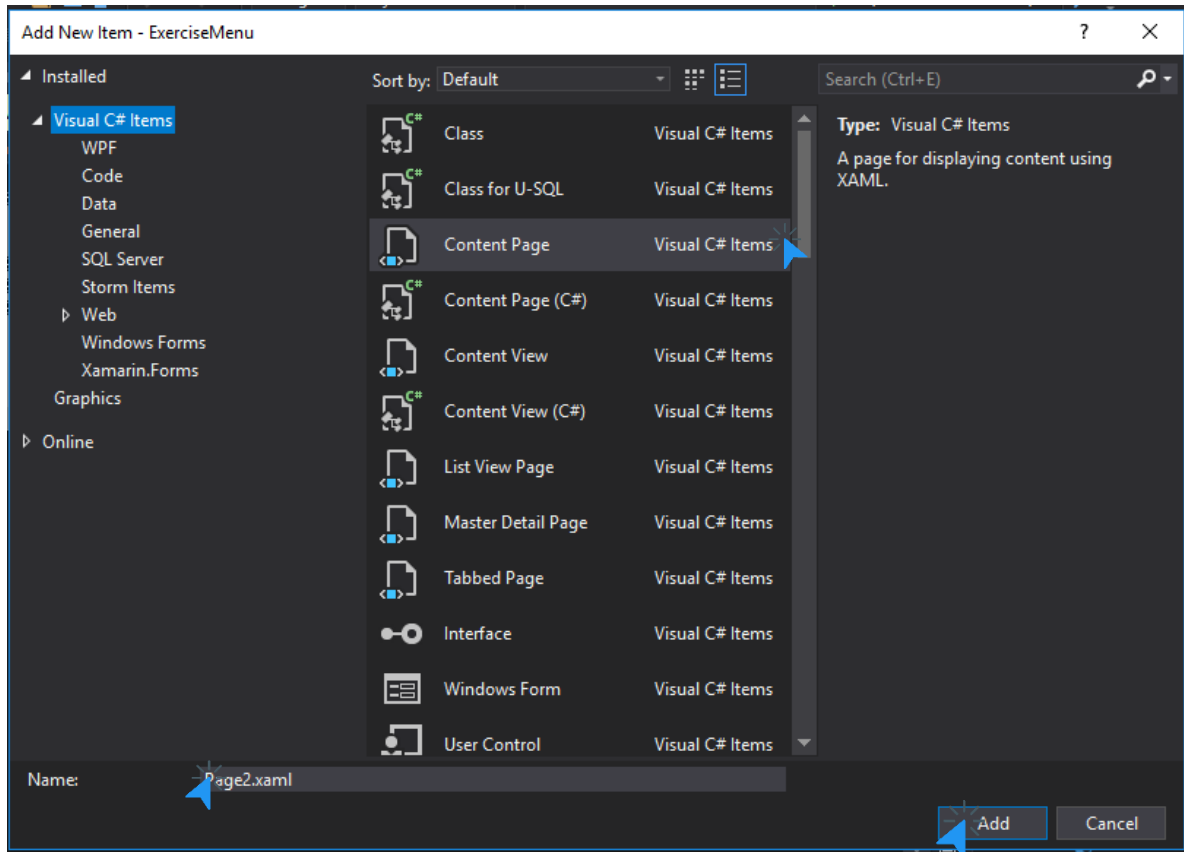
➤ Procederemos agregando 2 New Item.



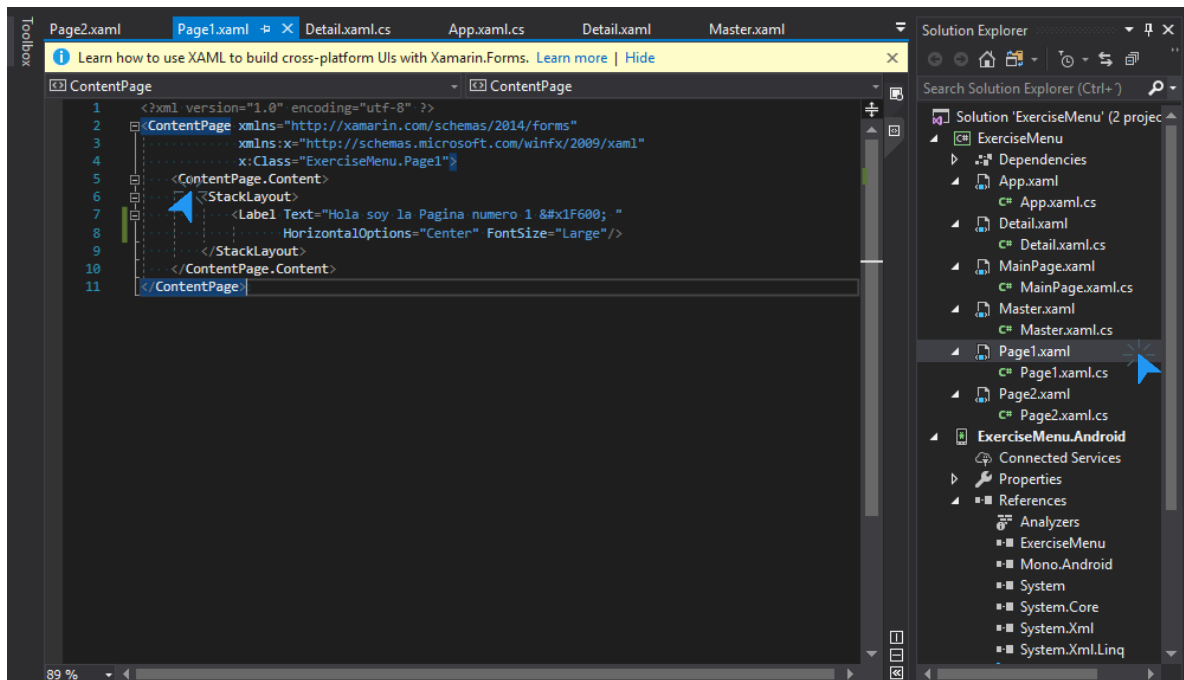
➤ La primera será una Content Page nombrada como **Page1**.



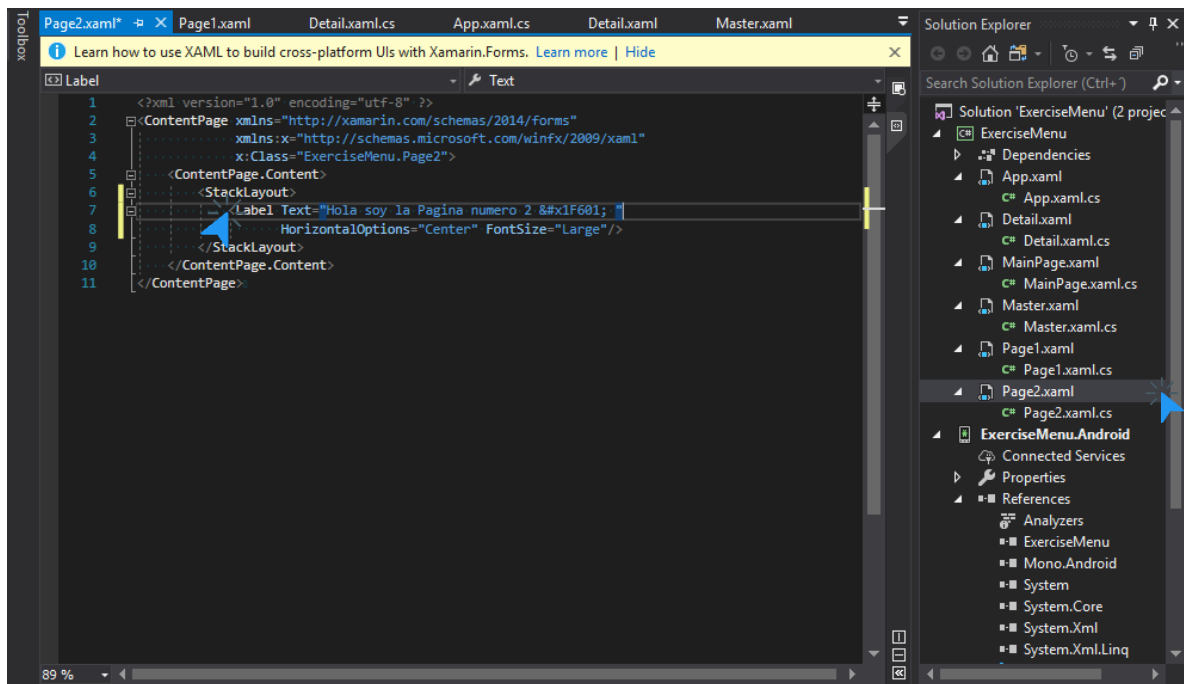
- La segunda será una Content Page nombrada como **Page2**.



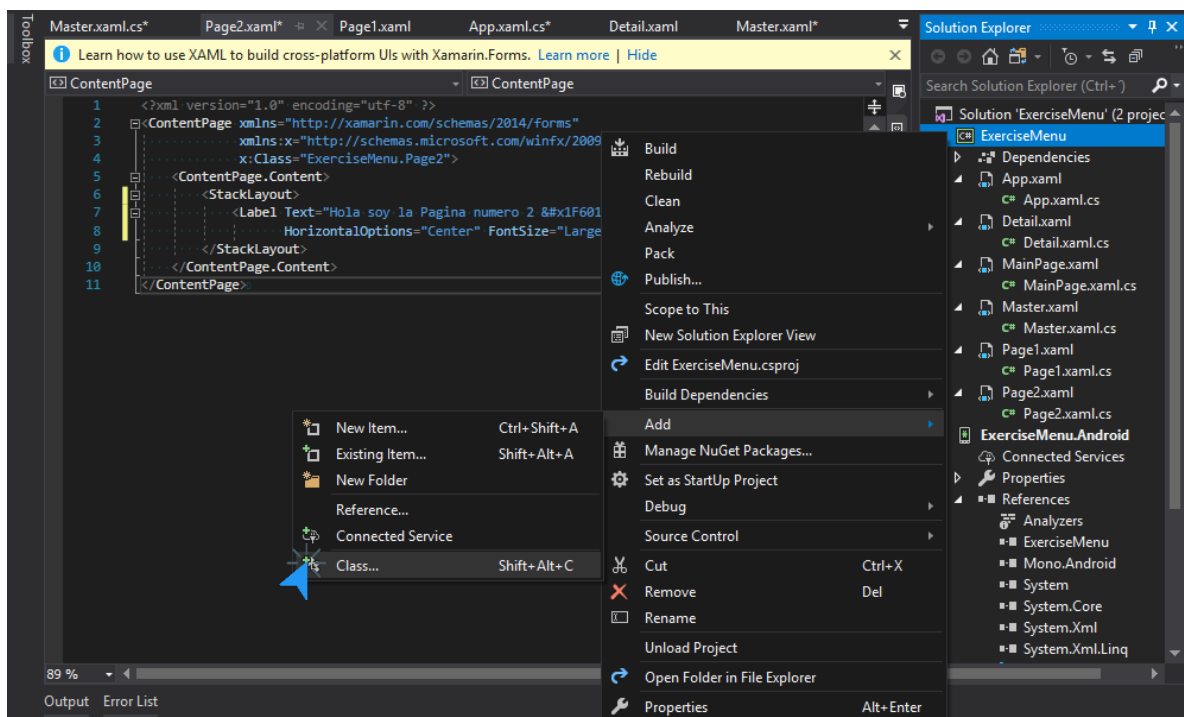
- Al terminar de agregar la **Page1** y la **Page2** nos dirigiremos a la **Page1** y agregamos el siguiente código.



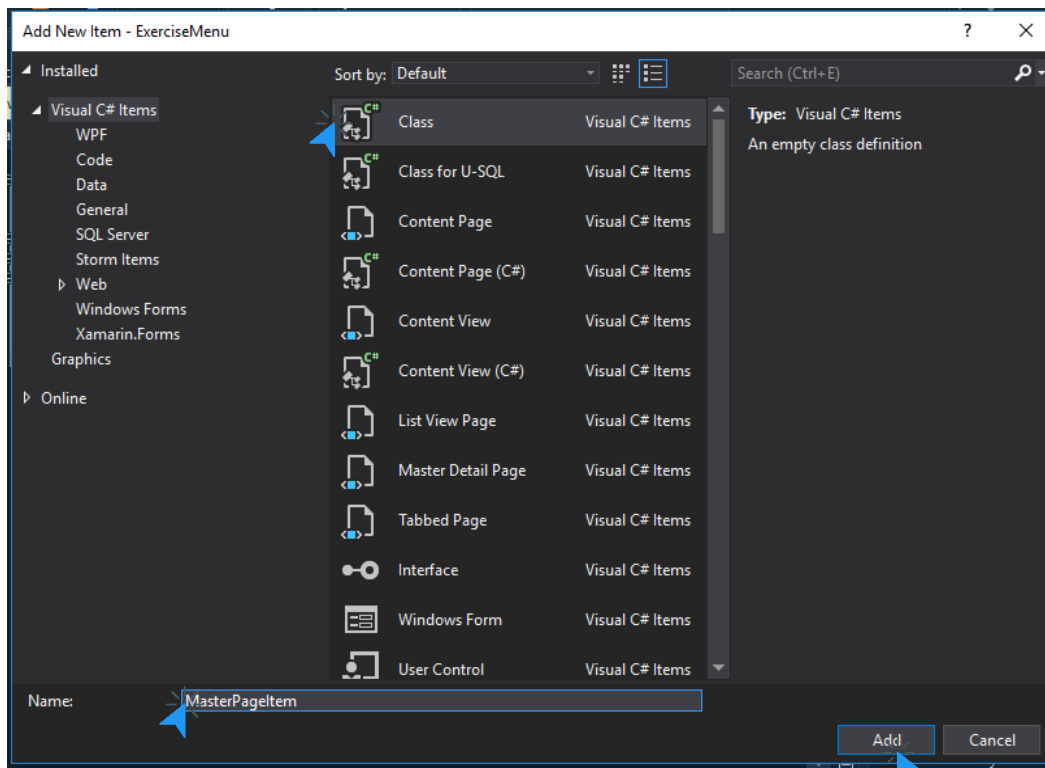
De igual forma modificaremos la **Page2**.



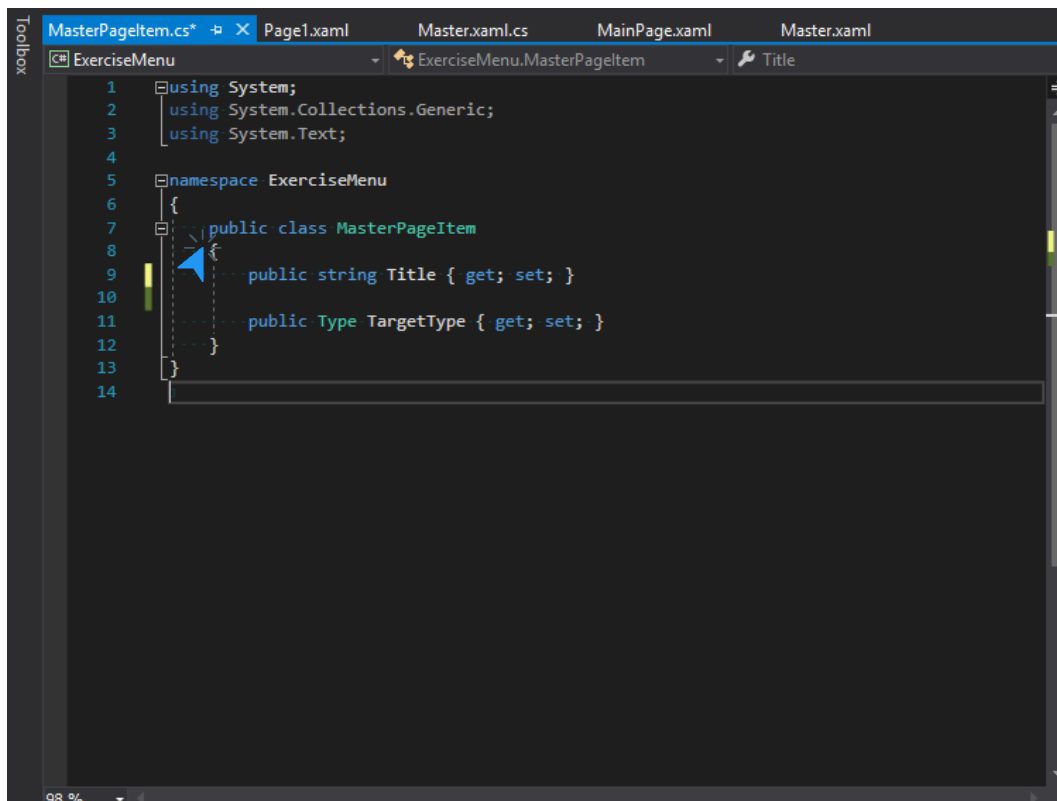
Agregaremos una nueva clase.



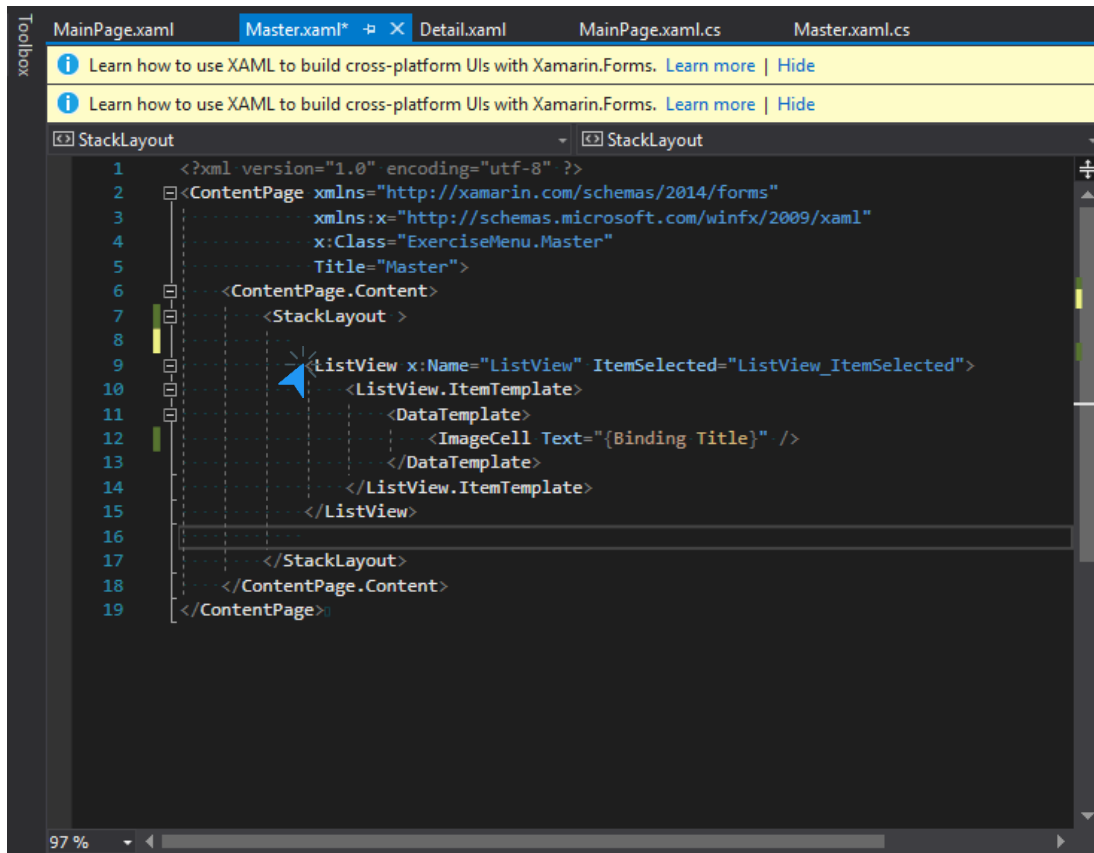
Seleccionamos Class y la nombramos como **MasterPageItem**.



Le agregamos el siguiente código el cual contendrá el Title y el TargetType.

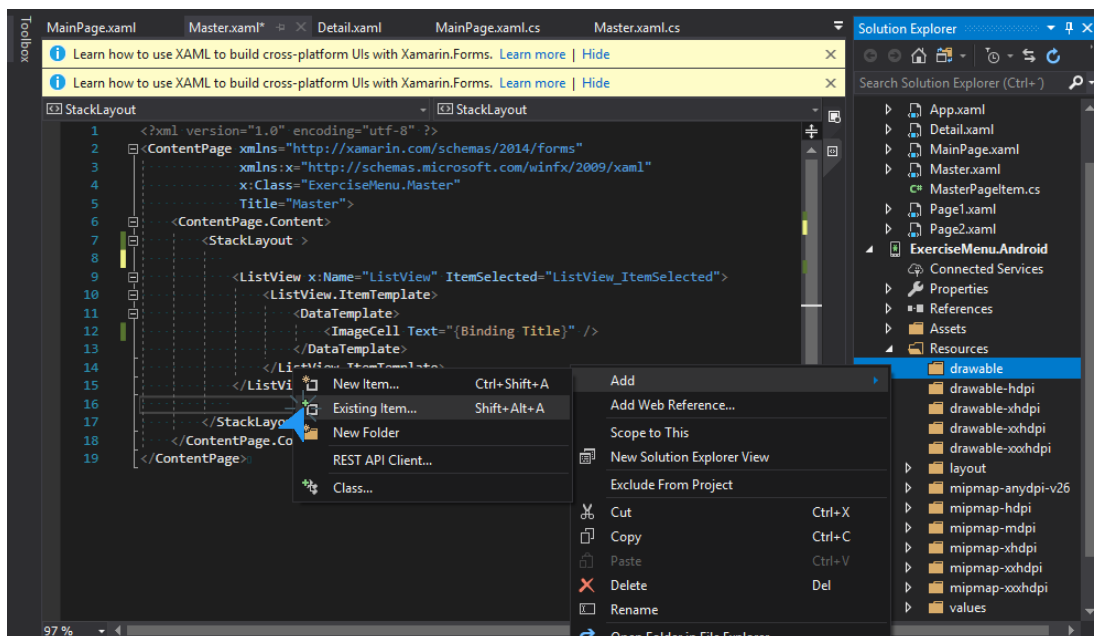


- En **Master.xaml** agregaremos un ListView el cual nombraremos como **ListView**, creando un evento ItemSelected.



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             x:Class="ExerciseMenu.Master"
5             Title="Master">
6     <ContentPage.Content>
7         <StackLayout>
8             <ListView x:Name="ListView" ItemSelected="ListView_ItemSelected">
9                 <ListView.ItemTemplate>
10                     <DataTemplate>
11                         <ImageCell Text="{Binding Title}" />
12                     </DataTemplate>
13                 </ListView.ItemTemplate>
14             </ListView>
15         </StackLayout>
16     </ContentPage.Content>
17 </ContentPage>
```

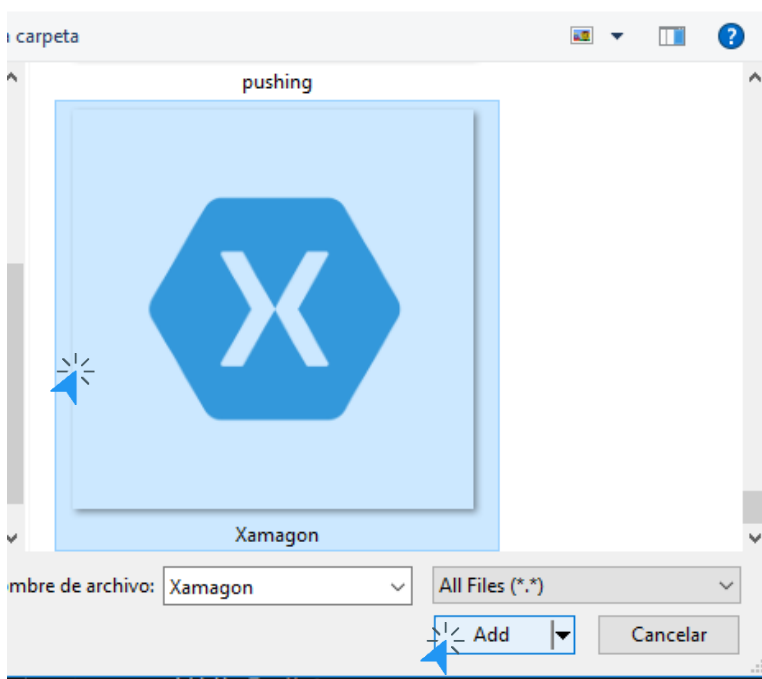
- Procederemos a agregar un icono, nos dirigimos a Resources y en drawable agregamos un Existing Item.



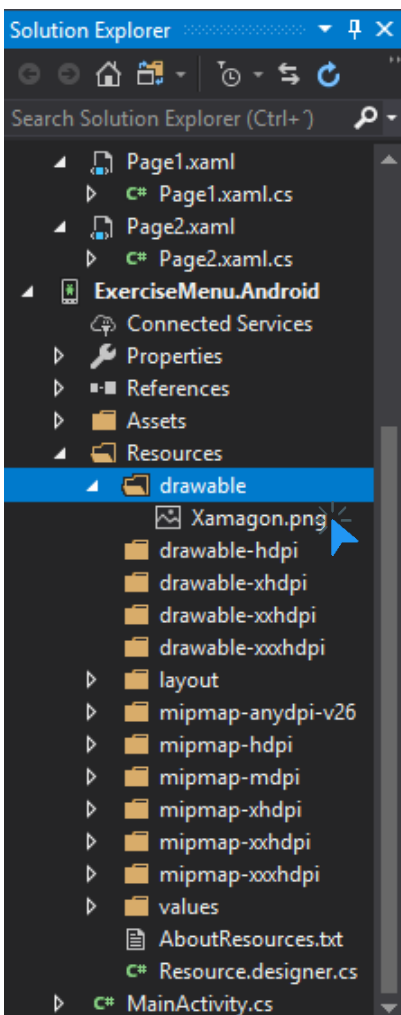
```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             x:Class="ExerciseMenu.Master"
5             Title="Master">
6     <ContentPage.Content>
7         <StackLayout>
8             <ListView x:Name="ListView" ItemSelected="ListView_ItemSelected">
9                 <ListView.ItemTemplate>
10                     <DataTemplate>
11                         <ImageCell Text="{Binding Title}" />
12                     </DataTemplate>
13                 </ListView.ItemTemplate>
14             </ListView>
15         </StackLayout>
16     </ContentPage.Content>
17 </ContentPage>
```



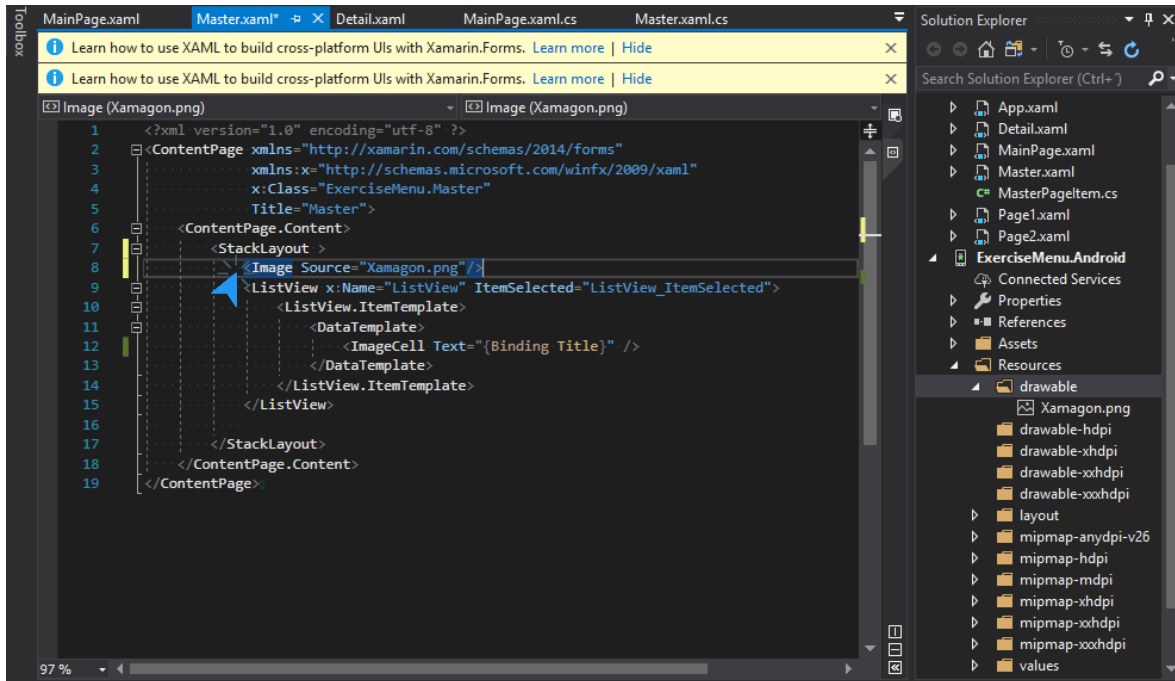
Seleccionamos un icono que tengamos con anterioridad.



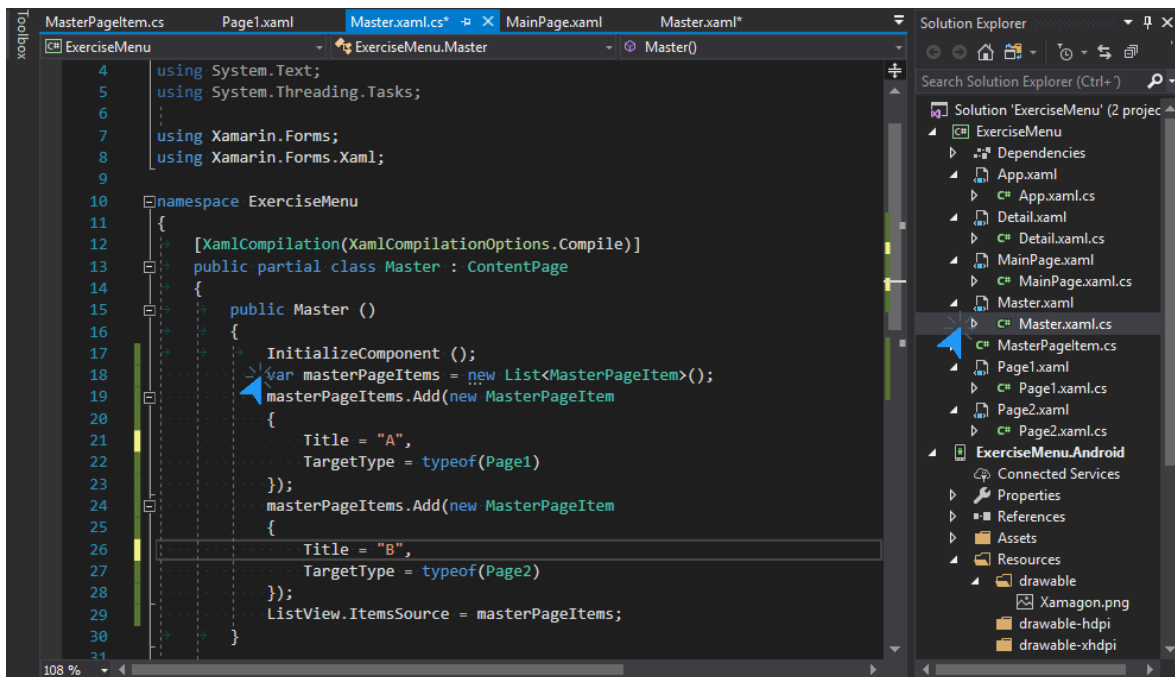
Debemos recordar cómo ha sido nombrado para utilizarlo posteriormente.



➤ Agregamos arriba del ListView un Image en el cual agregaremos el icono anterior.



➤ En **Master.xaml.cs** definiremos el contenido del ListView.



En el evento del ItemSelected se hará posible la navegación del ListView.

The screenshot shows the Visual Studio IDE with the file `Master.xaml.cs` open. The code implements the `ItemSelected` event handler for a `ListView`. The `MasterPageItem` class is defined with `TargetType` properties for `Page1` and `Page2`. The `ListView_ItemSelected` method is implemented as follows:

```
22 TargetType = typeof(Page1)
23 });
24 masterPageItems.Add(new MasterPageItem
25 {
26     Title = "B",
27     TargetType = typeof(Page2)
28 });
29 ListView.ItemsSource = masterPageItems;
30 }
31
32 private void ListView_ItemSelected(object sender, SelectedItemChangedEventArgs e)
33 {
34     var item = e.SelectedItem as MasterPageItem;
35     if (item != null)
36     {
37         App.MasterDetail.Detail.Navigation.
38             PushAsync((Page)Activator.CreateInstance(item.TargetType));
39         ListView.SelectedItem = null;
40         App.MasterDetail.IsPresented = false;
41     }
42 }
43
44 }
```

The Solution Explorer on the right shows the project structure for 'ExerciseMenu' (2 projects), including `ExerciseMenu` and `ExerciseMenu.Android`. The `ExerciseMenu` project contains `App.xaml`, `Detail.xaml`, `MainPage.xaml`, and `Master.xaml`. The `ExerciseMenu.Android` project contains `Connected Services`, `Properties`, `References`, `Assets`, and `Resources`.

Nos dirigiremos a **Detail.xaml** y agregamos un Label.

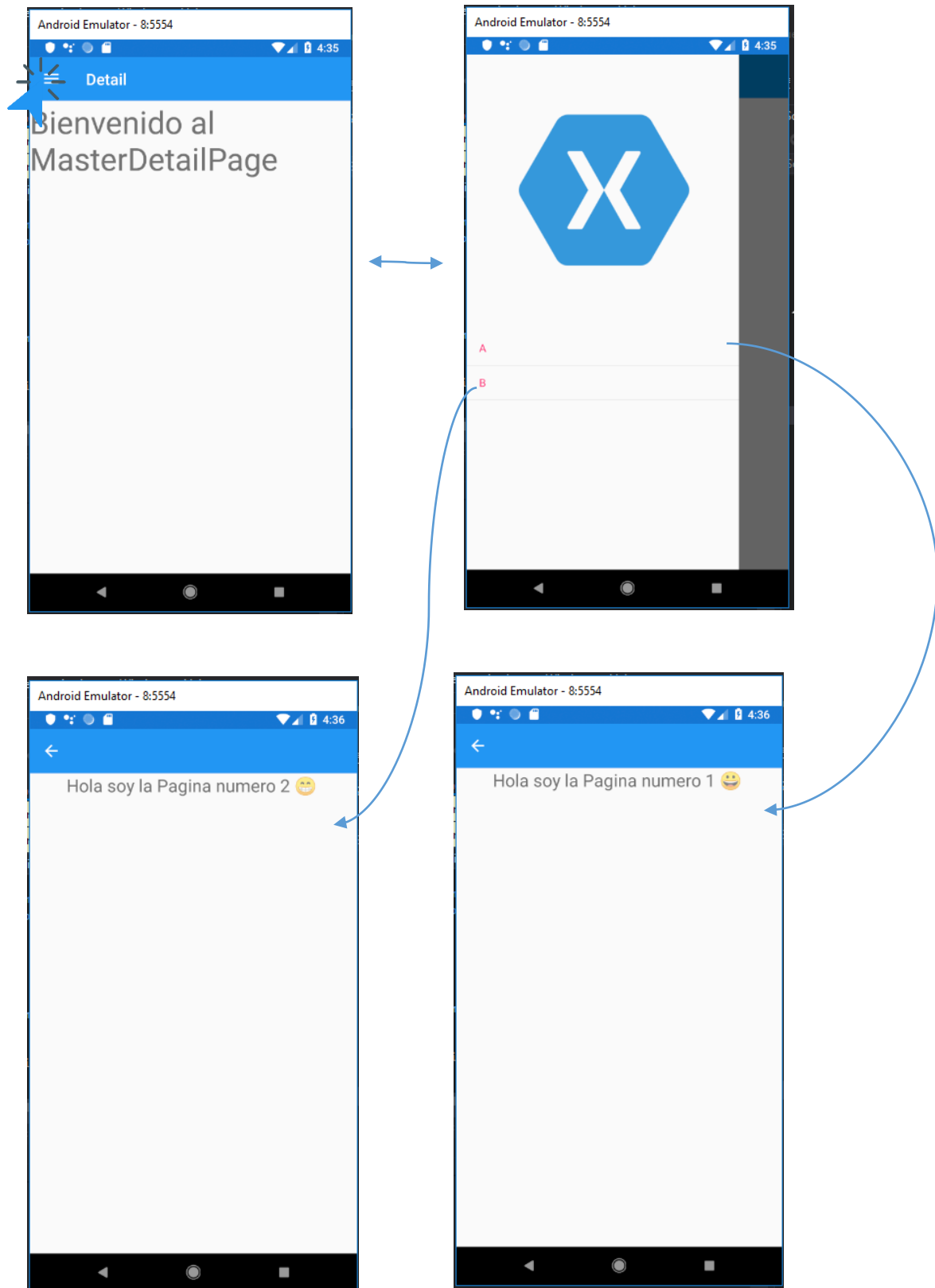
The screenshot shows the Visual Studio IDE with the file `Detail.xaml` open. The XAML code defines the `Detail` page, which is a `ContentPage` with a `StackLayout` containing a `Label`. The `Label` has the text "Bienvenido al MasterDetailPage" and a font size of 40. The `ContentPage` has a title of "Detail".

```
1 ?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             x:Class="ExerciseMenu.Detail"
5             Title="Detail">
6     <ContentPage.Content>
7         <StackLayout>
8             <Label Text="Bienvenido al MasterDetailPage"
9                 FontSize="40" HorizontalOptions="Center" VerticalOptions="Center" />
10        </StackLayout>
11    </ContentPage.Content>
12 </ContentPage>
```


The Solution Explorer on the right shows the project structure for 'ExerciseMenu' (2 projects), including `ExerciseMenu` and `ExerciseMenu.Android`. The `ExerciseMenu` project contains `App.xaml`, `Detail.xaml`, `MainPage.xaml`, and `Master.xaml`. The `ExerciseMenu.Android` project contains `Connected Services`, `Properties`, `References`, `Assets`, and `Resources`.




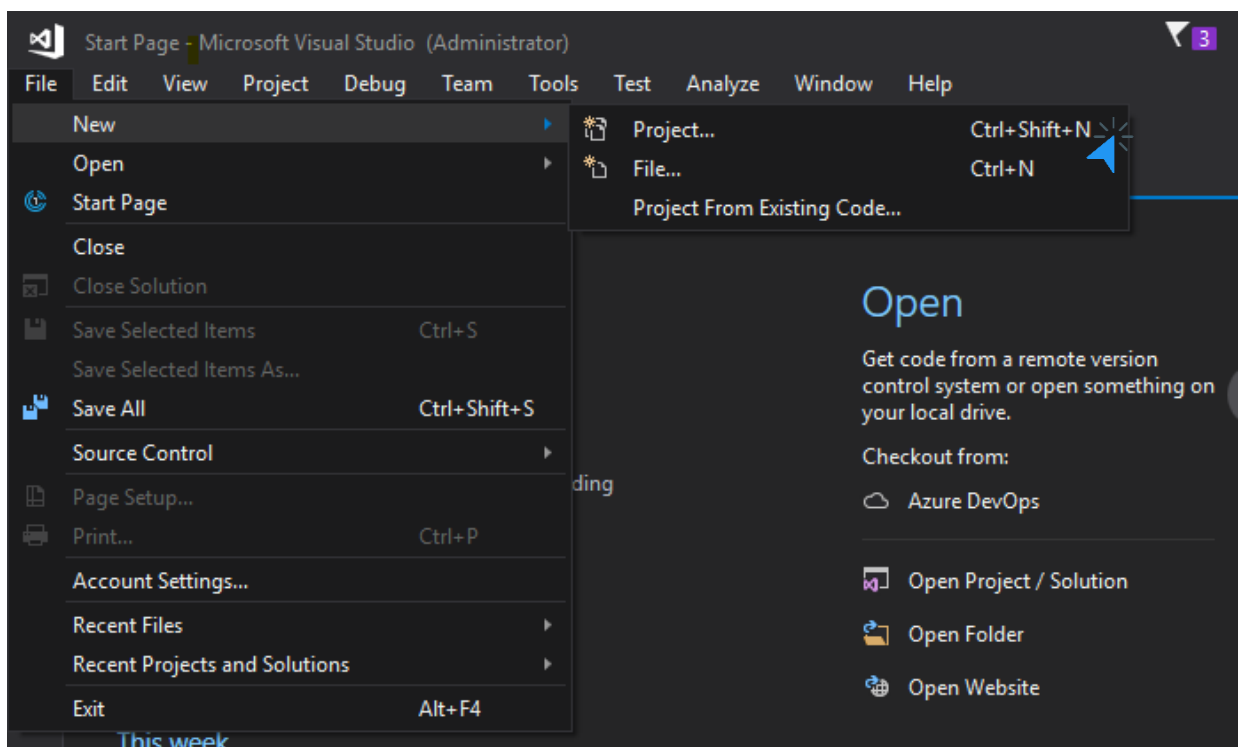
Finalmente terminaría así.



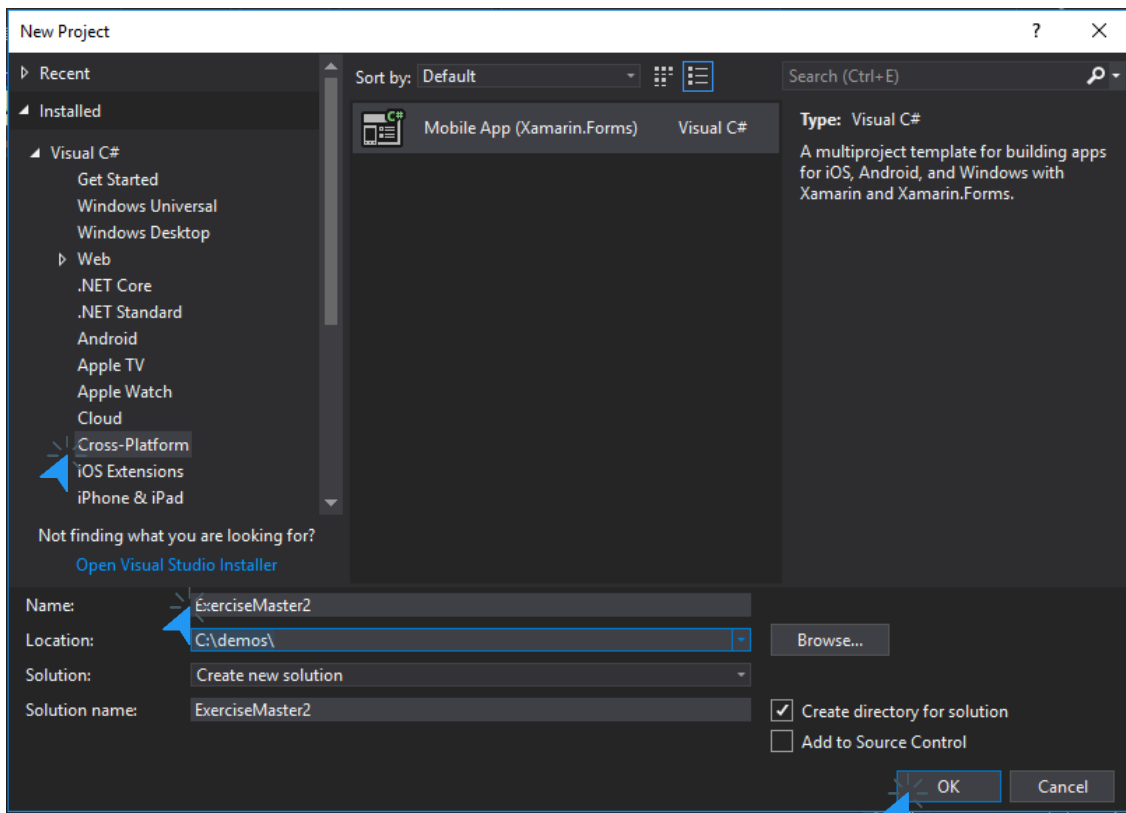
Segunda forma

 En la segunda forma es posible observar lo más básico de un MasterDetailPage por medio de Buttons.

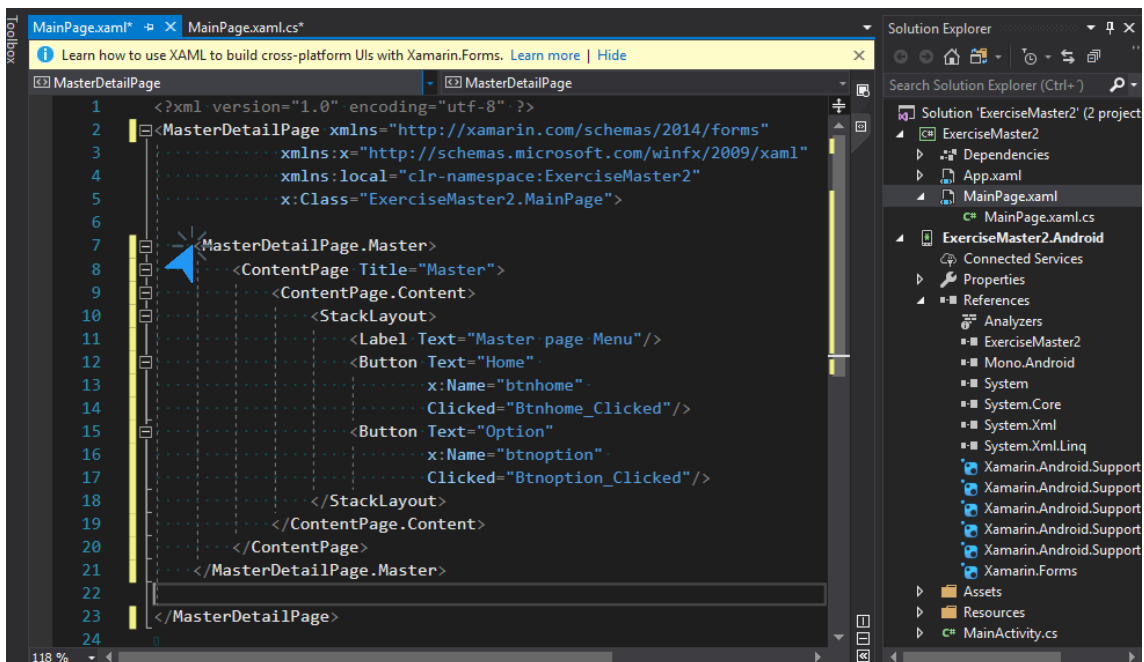
 Crearemos un nuevo proyecto en Visual Studio.



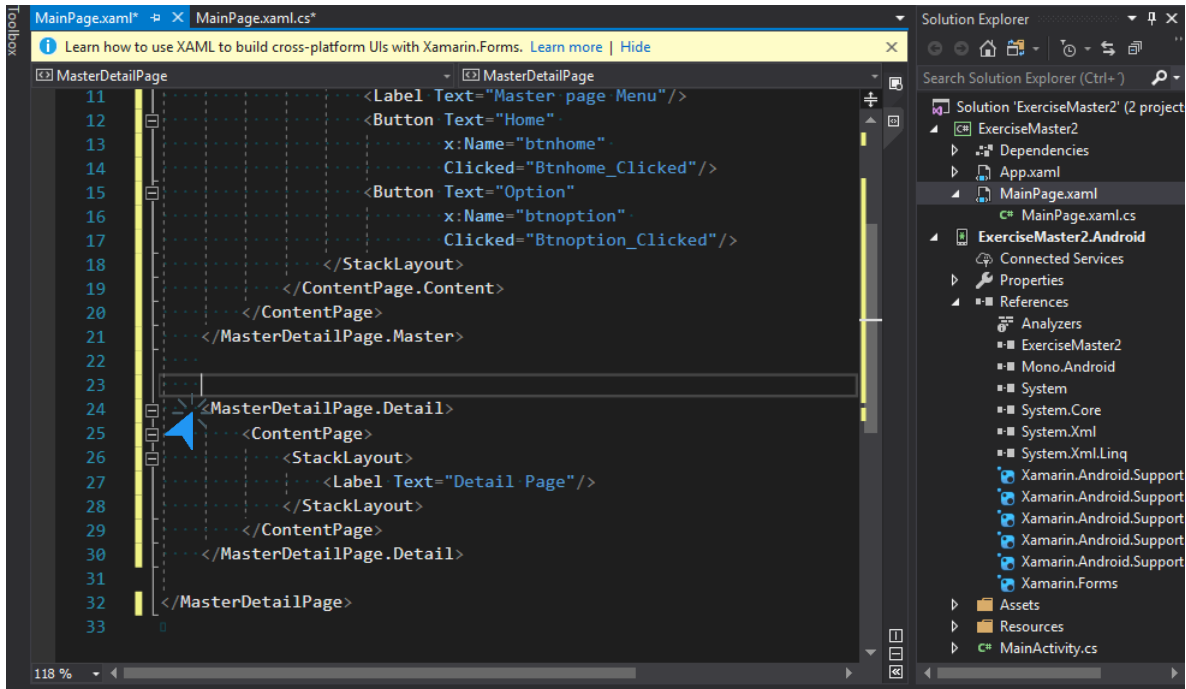
Seleccionamos Cross-Plataforma, lo nombramos como **ExersiceMenu2** y damos OK.



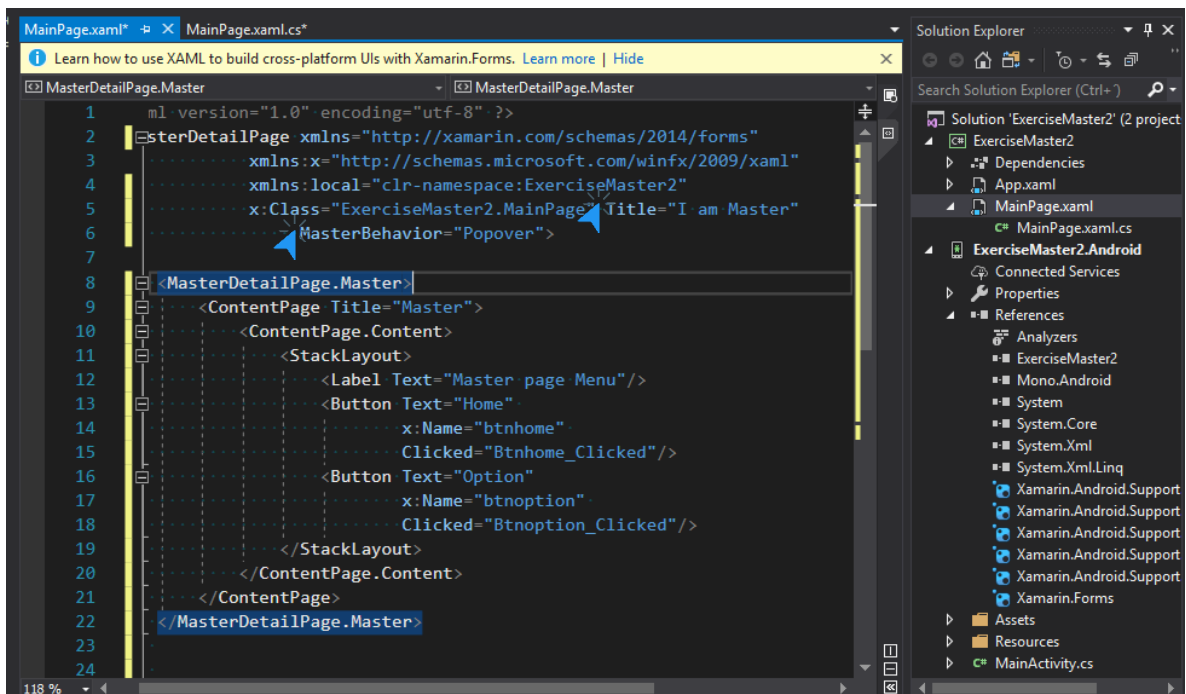
Modificamos la **MainPage** para que pase de ser una ContentPage a una MasterDetailPage, agregando el Master a la vez 1 Label y 2 Buttons.



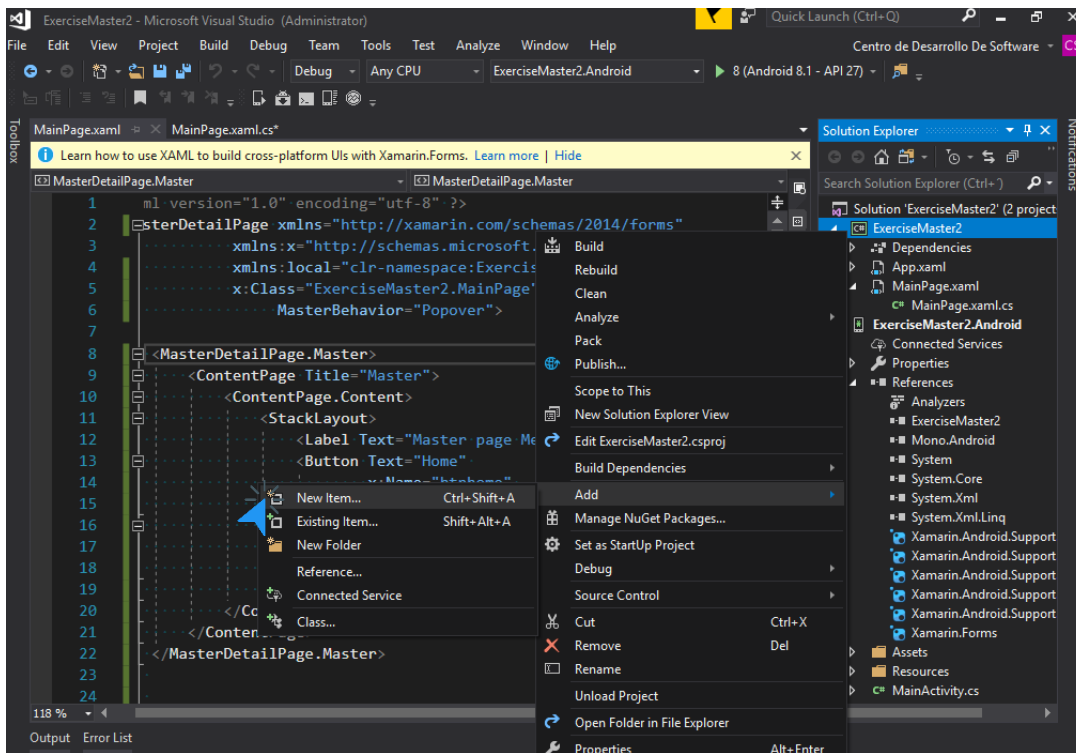
➦ Luego agregamos el Detail, todo esto en la **MainPage**.



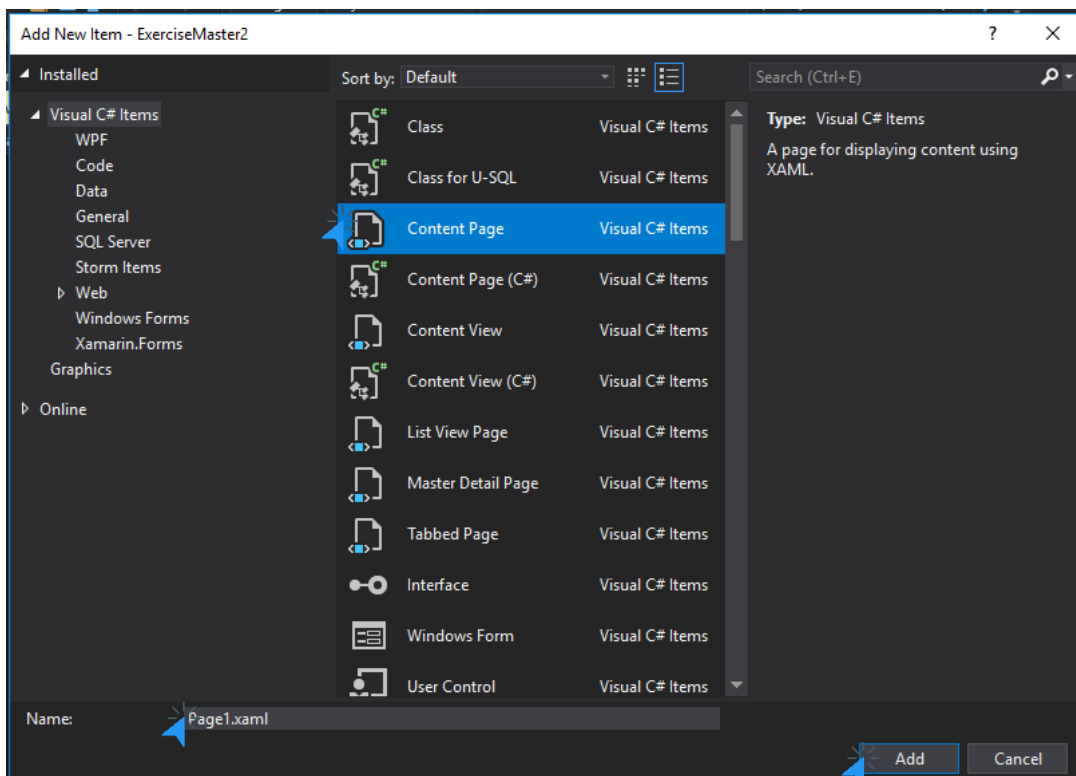
➦ También modificamos agregamos el Title y el MasterBehavior.



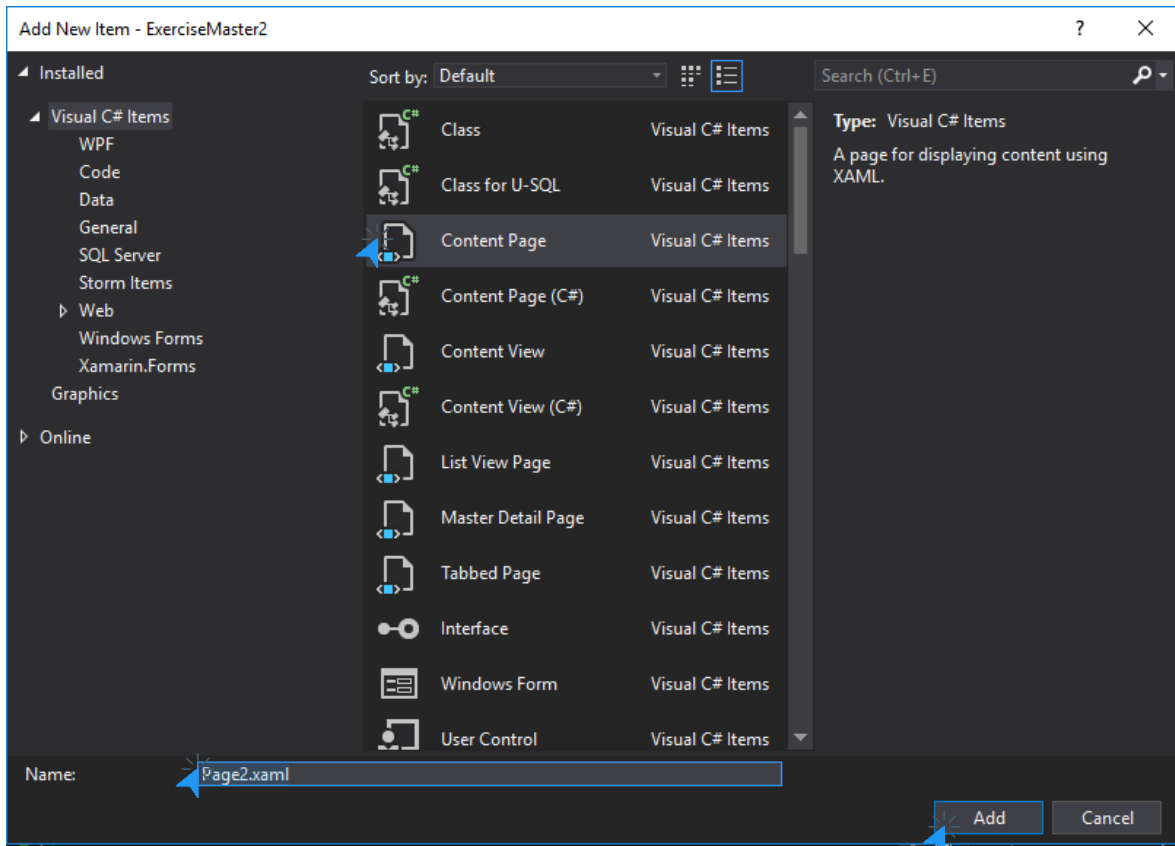
✚ Agregamos 2 New Item los cuales servirán como home y options.



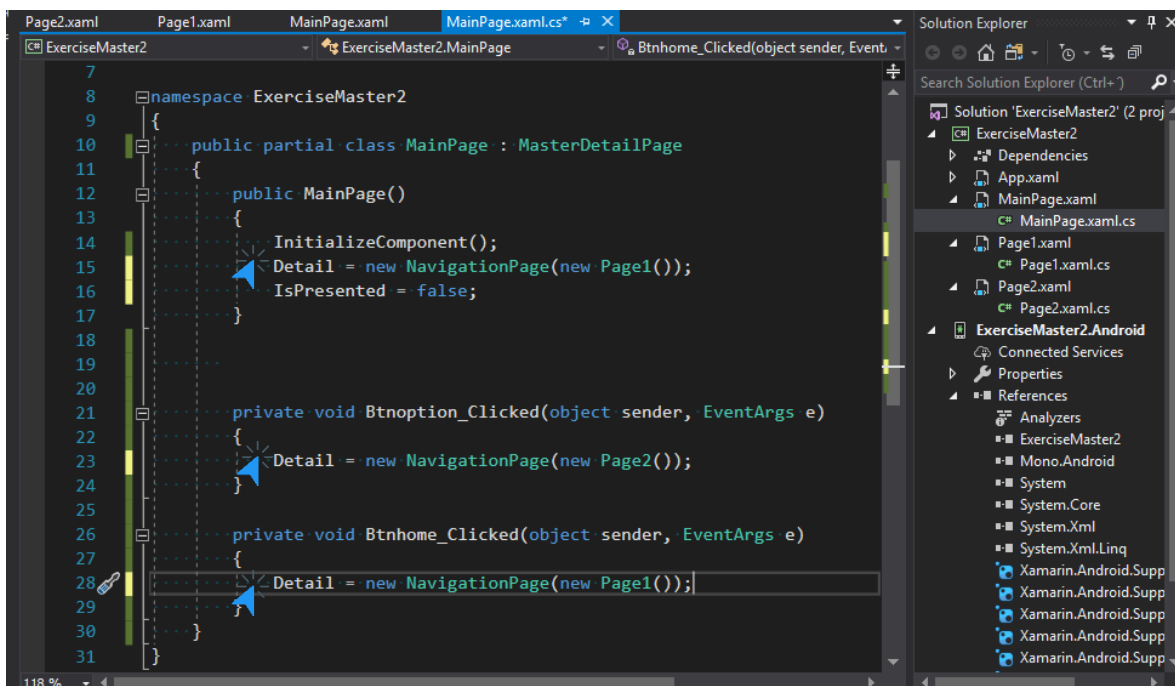
✚ Seleccionamos Content Page y lo nombramos como **Page1**.



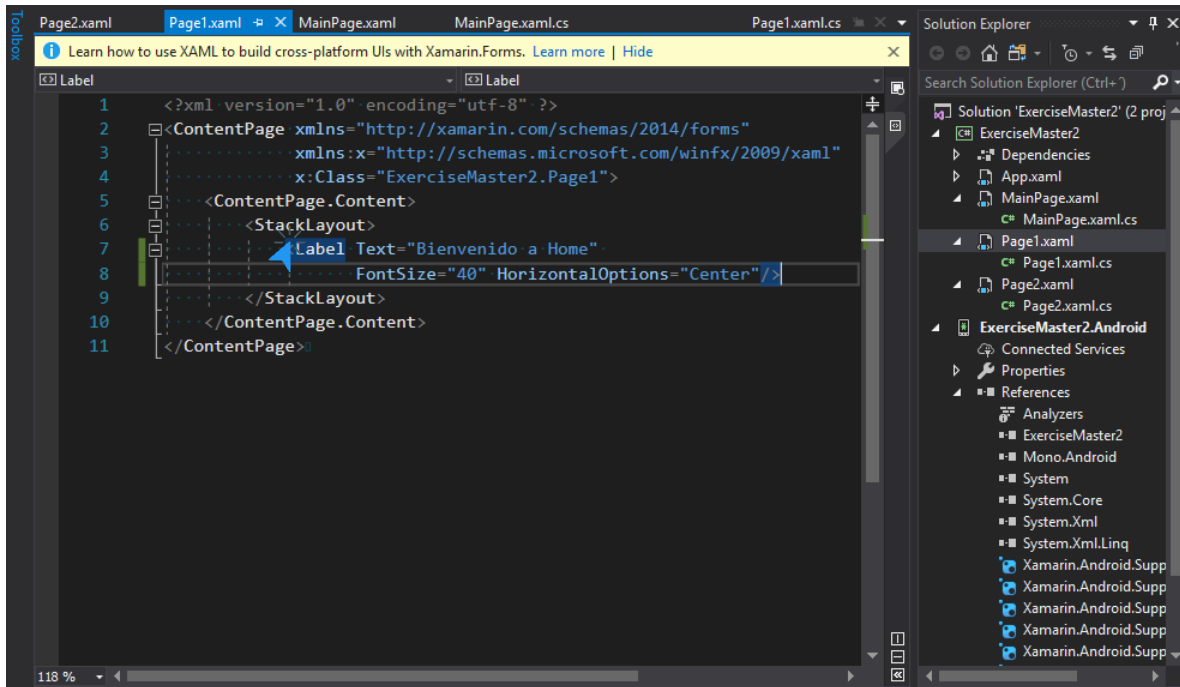
La siguiente también será una Content Page y la nombraremos como **Page2**.



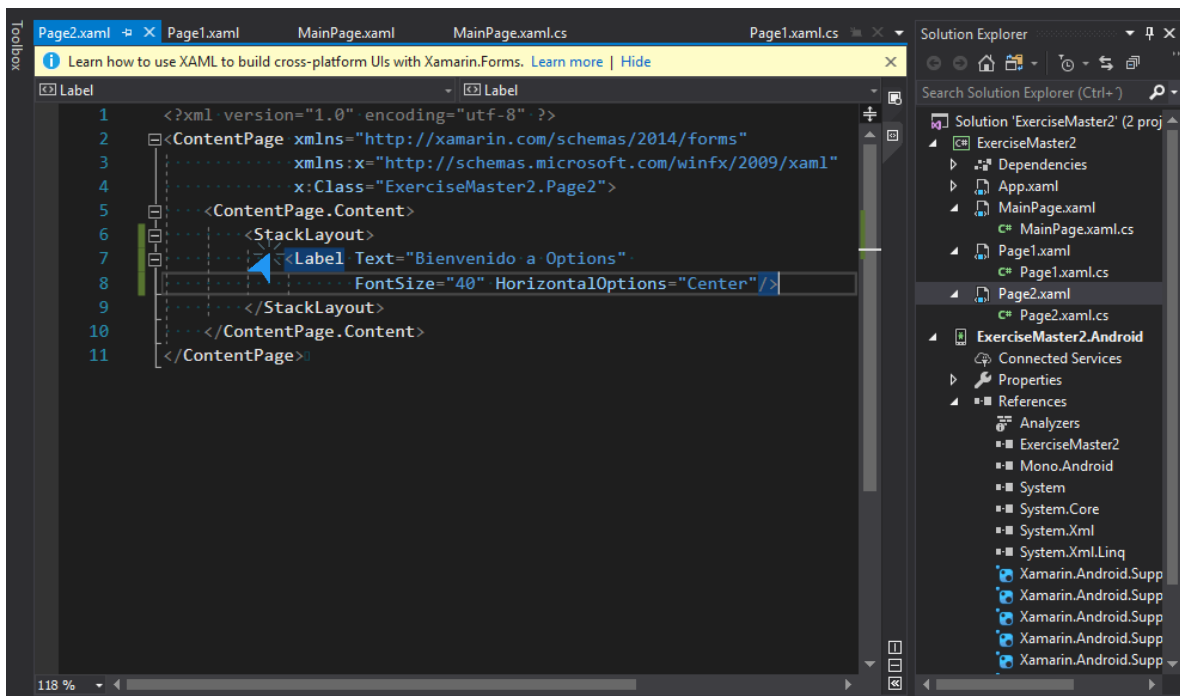
En **MainPage.xaml.cs** definiremos el Detail, además navegaremos por medio de los eventos Buttons.



En **Page1** agregaremos el siguiente Label.



En **Page2** agregaremos el siguiente Label.



✚ El resultado final sería similar a la primera forma, aunque utilizando Buttons.

