University of Zakho

Faculty of Science

Computer Department

4/22/2022

[Mobile application]

[Shopping App]

Prepared by:

1-Saad Zaid   2-Haval khalid

3-Omer Mikael   4-Sahir Shkur

Supervised by:

Mr.Yousif

What is Xamarin.Forms?

With Xamarin, you can build cross-platform apps based on .NET and C#. No more Java and Objective-C or Swift. Everything you can do with these languages and platforms, can be done with Xamarin in C#.

This is possible because of the Mono project.

If you have been around long enough, you might remember Mono. It was the Linux variant of the .NET framework. Basically, what the Mono project did was implement each new .NET API or feature in their own Mono framework, making it the first fully featured, cross-platform .NET solution. The people who have worked on Mono are mostly also the people that founded Xamarin.

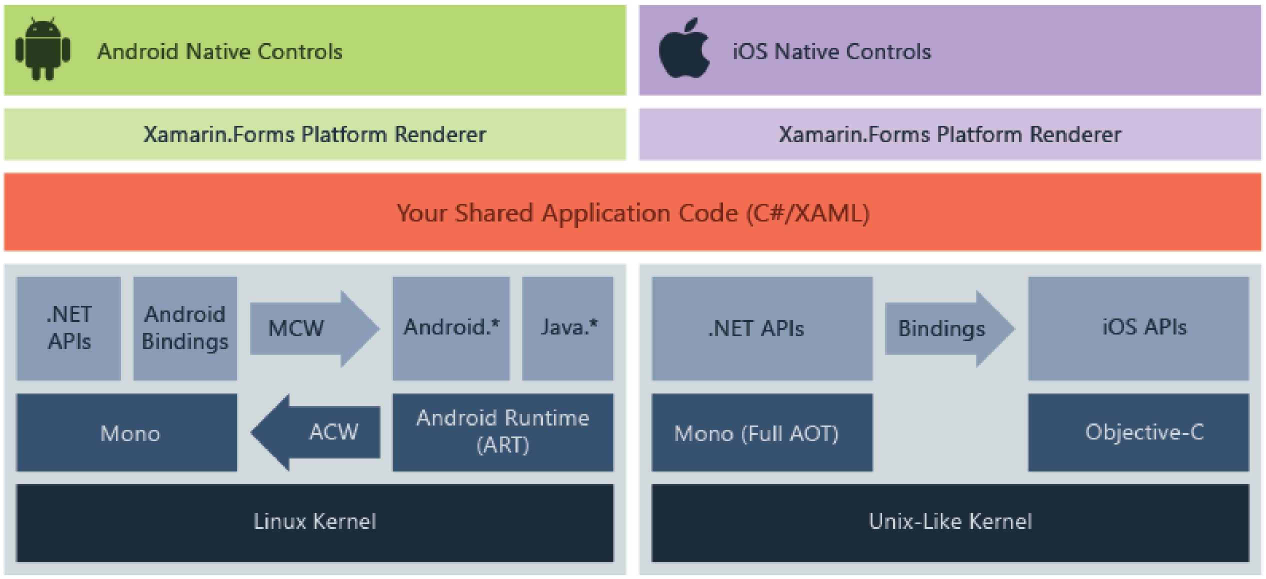
The solutions Xamarin offers you is based on binding libraries that project the native APIs onto their .NET equivalents. This allows you to write cross-platform apps based on a technology that you already know and love. The remaining problem that was left when Xamarin was introduced is that the user interface still needed to be implemented in a native way.

For Android, this means with AXML files which contain some kind of XML dialect, and for iOS you have to use Storyboards, which are also XML files under the hood. To overcome this last hurdle in true cross-platform development, **Xamarin.Forms** was created.

With Xamarin.Forms, you can define your UI through C# code or XAML in an abstract way. For instance, you define a Button object and the Xamarin.Forms libraries will translate this to their native equivalent. This way, you can develop your interfaces cross platform, but that same interface will still look native on each platform.

If you still need more information on what Xamarin.Forms is exactly and how it works internally, I would recommend to read up in the Microsoft Docs: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/.

How Xamarin.Forms works:



Xamarin.Forms provides a consistent API for creating UI elements across platforms.

This API can be implemented in either XAML or C# and supports databinding for patterns such as Model-View-ViewModel (MVVM).

At runtime, Xamarin.Forms utilizes platform renderers to convert the cross-platform UI elements into native controls on Xamarin.Android, Xamarin.iOS and UWP. This allows developers to get the native look, feel and performance while realizing the benefits of code sharing across platforms.

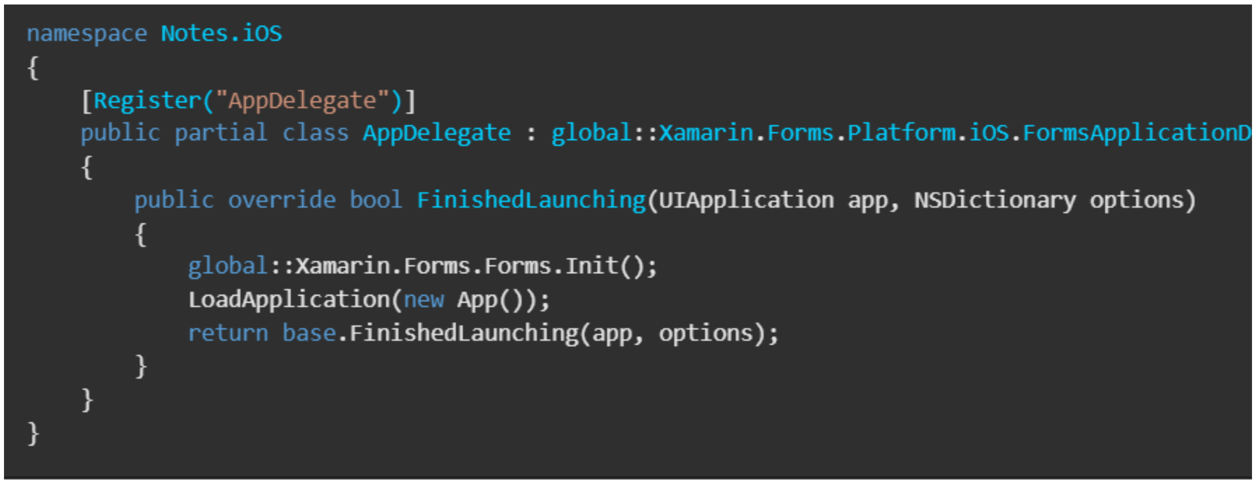
Xamarin.Forms applications typically consist of a shared .NET Standard library and individual platform projects. The shared library contains the XAML or C# views and any business logic such as services, models or other code. The platform projects contain any platform-specific logic or packages the application requires.

Xamarin.Forms uses the Xamarin platform to run .NET applications natively across platforms. For more information about the Xamarin platform, see [What is Xamarin?](#).

Launch the application on each platform:

iOS:

To launch the initial Xamarin.Forms page in iOS, the Notes.iOS project defines the AppDelegate class that inherits from the FormsApplicationDelegate class:

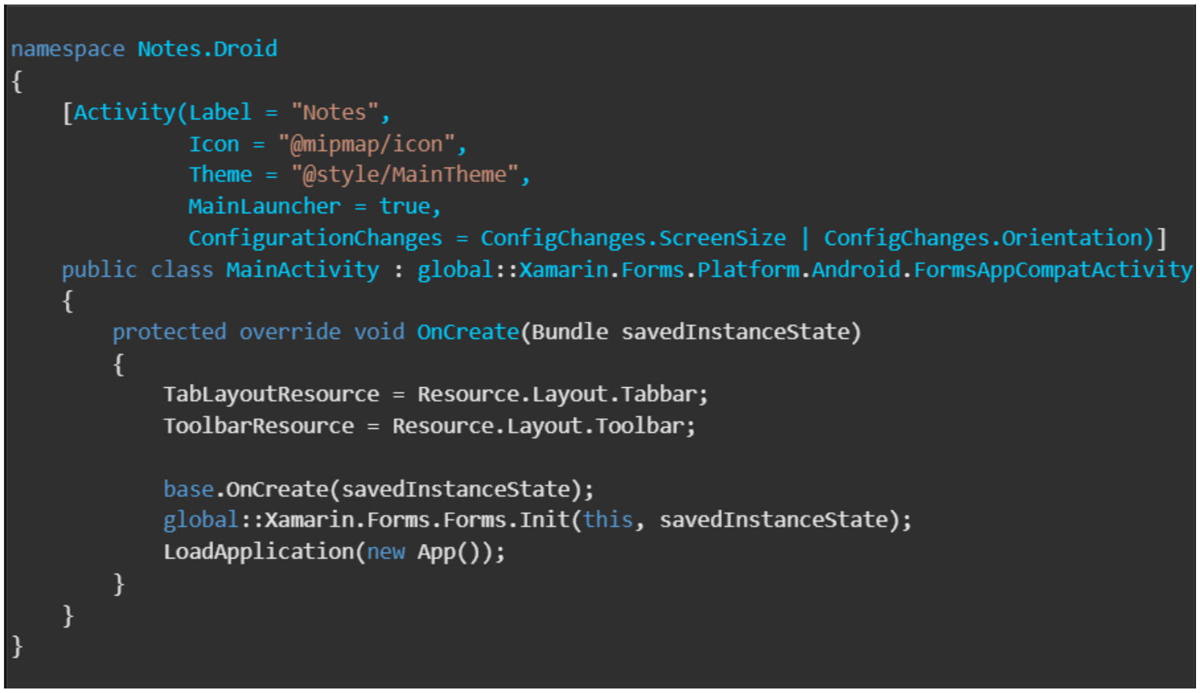


```
namespace Notes.iOS
{
    [Register("AppDelegate")]
    public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationD
    {
        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();
            LoadApplication(new App());
            return base.FinishedLaunching(app, options);
        }
    }
}
```

The FinishedLaunching override initializes the Xamarin.Forms framework by calling the Init method. This causes the iOS-specific implementation of Xamarin.Forms to be loaded in the application before the root view controller is set by the call to the LoadApplication method.

Android:

To launch the initial Xamarin.Forms page in Android, the Notes.Android project includes code that creates an Activity with the MainLauncher attribute, with the activity inheriting from the FormsAppCompatActivity class:



```
namespace Notes.Droid
{
    [Activity(Label = "Notes",
              Icon = "@mipmap/icon",
              Theme = "@style/MainTheme",
              MainLauncher = true,
              ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)]
    public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate(savedInstanceState);
            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
            LoadApplication(new App());
        }
    }
}
```

The OnCreate override initializes the Xamarin.Forms framework by calling the Init method. This causes the Android-specific implementation of Xamarin.Forms to be loaded in the application before the Xamarin.Forms application is loaded.

User interface:

There are several control groups used to create the user interface of a Xamarin.Forms application:

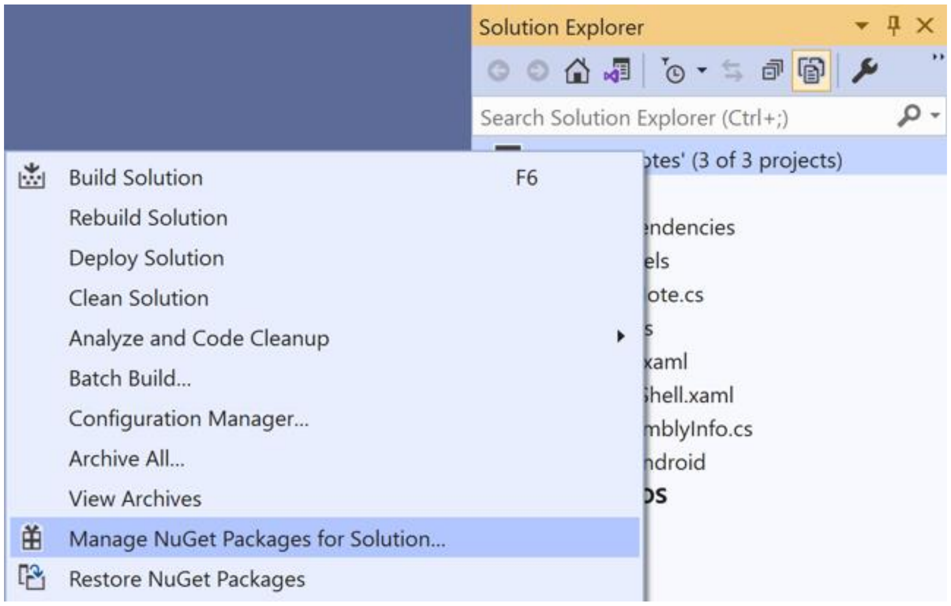
1. **Pages** – Xamarin.Forms pages represent cross-platform mobile application screens. The Notes application uses the ContentPage class to display single screens. For more information about pages, see Xamarin.Forms Pages.
2. **Views** – Xamarin.Forms views are the controls displayed on the user interface, such as labels, butto,\ns, and text entry boxes. The finished Notes application uses the CollectionView, Editor, and Button views. For more information about views, see Xamarin.Forms Views.
3. **Layouts** – Xamarin.Forms layouts are containers used to compose views into logical structures. The Notes application uses the StackLayout class to arrange views in a vertical stack, and the Grid class to arrange buttons horizontally. For more information about layouts, see Xamarin.Forms Layouts.

What are the Top Goals of Developers with Xamarin.Forms:

- Xamarin.Forms are for developers with the following main goals:
- They can Share UI layout and design across platforms.
- Developers Share code, test, and business logic across platforms.
- They can write cross-platform apps in C# with Visual Studio.

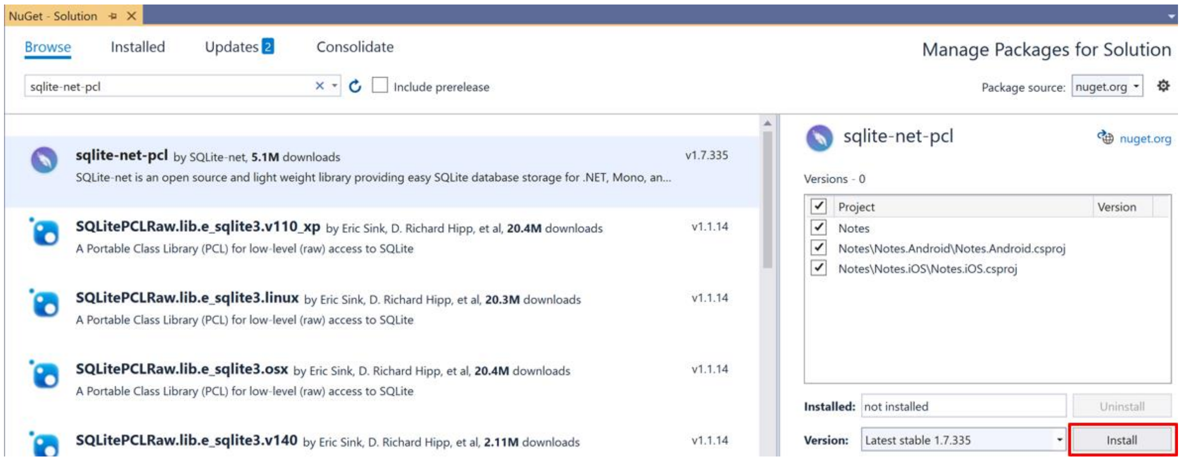
Sqlite-net-pcl :

1. Launch Visual Studio and open the Notes solution.
2. In **Solution Explorer**, right-click the **Notes** solution and select **Manage NuGet Packages for Solution...**:



3. In the **NuGet Package Manager**, select the **Browse** tab, and search for the **sqlite-net-pcl** NuGet package.

   In the **NuGet Package Manager**, select the correct **sqlite-net-pcl** package, check the **Project** checkbox, and click the **Install** button to add it to the solution:



sqlite-net-pcl

sqlite-net-pcl by SQLite-net, 5.1M downloads v1.7.335
SQLite-net is an open source and light weight library providing easy SQLite database storage for .NET, Mono, an...

SQLitePCLRaw.lib.e_sqlite3.v110_xp by Eric Sink, D. Richard Hipp, et al, 20.4M downloads v1.1.14
A Portable Class Library (PCL) for low-level (raw) access to SQLite

SQLitePCLRaw.lib.e_sqlite3.linux by Eric Sink, D. Richard Hipp, et al, 20.3M downloads v1.1.14
A Portable Class Library (PCL) for low-level (raw) access to SQLite

SQLitePCLRaw.lib.e_sqlite3.osx by Eric Sink, D. Richard Hipp, et al, 20.4M downloads v1.1.14
A Portable Class Library (PCL) for low-level (raw) access to SQLite

SQLitePCLRaw.lib.e_sqlite3.v140 by Eric Sink, D. Richard Hipp, et al, 2.11M downloads v1.1.14

sqlite-net-pcl nuget.org

Versions - 0

✓	Project	Version
✓	Notes	
✓	Notes\Notes.Android\Notes.Android.csproj	
✓	Notes\Notes.iOS\Notes.iOS.csproj	

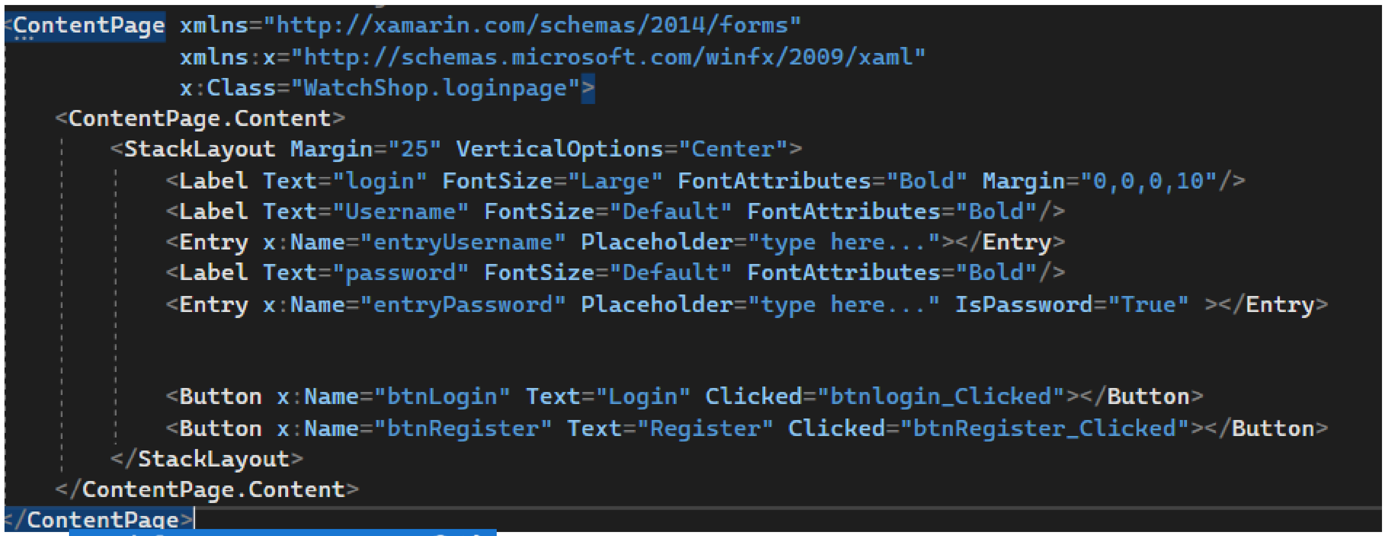
Installed: not installed

Version: Latest stable 1.7.335

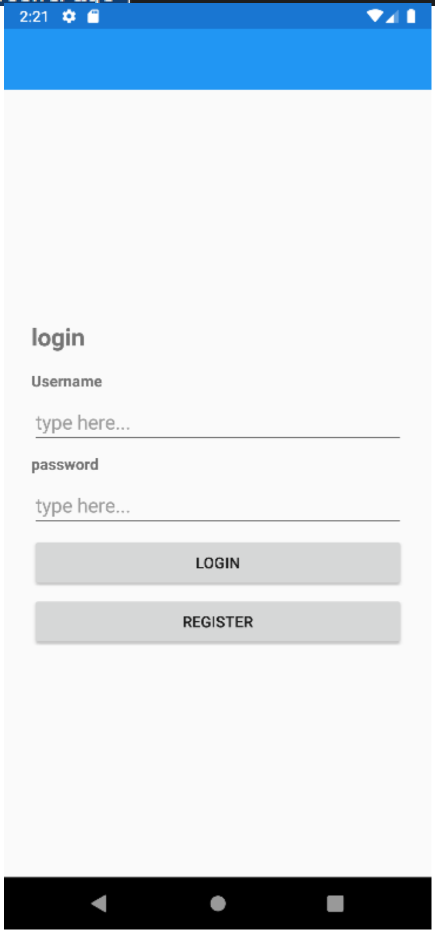
This package will be used to incorporate database operations into the application, and will be added to every project in the solution.

Login Page:

Login is a process of logging in to an application, database, or device etc. Most of the login pages require an email and password to be entered by the users.

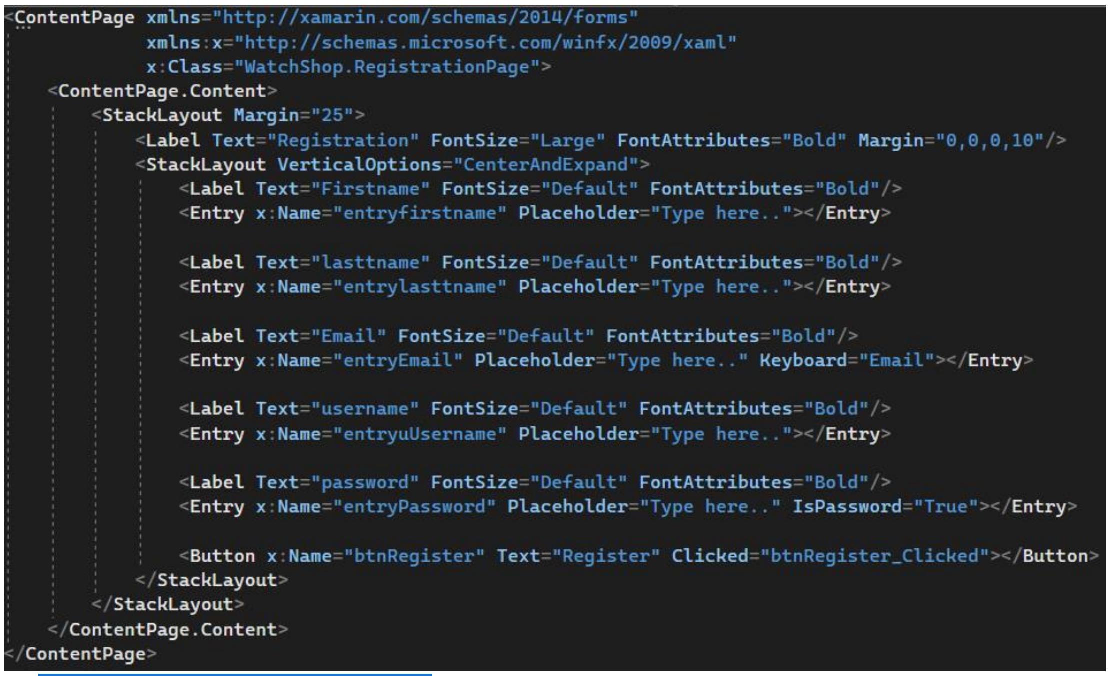


```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="WatchShop.loginpage">
    <ContentPage.Content>
        <StackLayout Margin="25" VerticalOptions="Center">
            <Label Text="login" FontSize="Large" FontAttributes="Bold" Margin="0,0,0,10"/>
            <Label Text="Username" FontSize="Default" FontAttributes="Bold"/>
            <Entry x:Name="entryUsername" Placeholder="type here..."></Entry>
            <Label Text="password" FontSize="Default" FontAttributes="Bold"/>
            <Entry x:Name="entryPassword" Placeholder="type here..." IsPassword="True" ></Entry>


            <Button x:Name="btnLogin" Text="Login" Clicked="btnlogin_Clicked"></Button>
            <Button x:Name="btnRegister" Text="Register" Clicked="btnRegister_Clicked"></Button>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

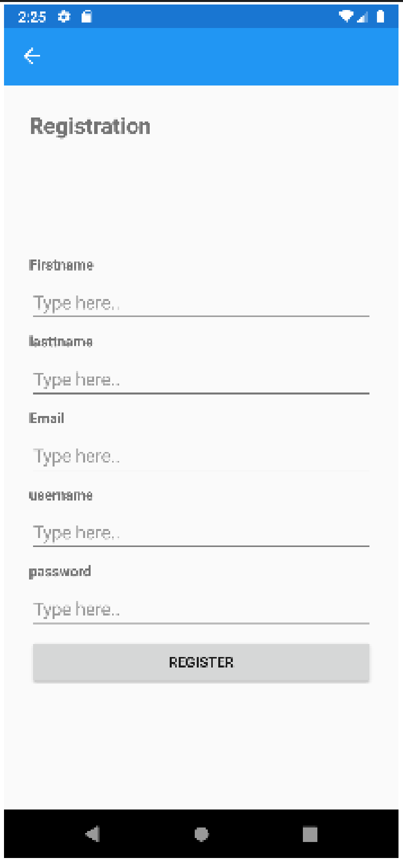


SignUp & Registration:

SignUp is a process of signing in to an application. Another term used for SignUp is registration. A user can provide his/her credentials to register himself/herself into the application. A signup form requires a username, email, password, DOB, gender and other such details for registration.



```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="WatchShop.RegistrationPage">
    <ContentPage.Content>
        <StackLayout Margin="25">
            <Label Text="Registration" FontSize="Large" FontAttributes="Bold" Margin="0,0,0,10"/>
            <StackLayout VerticalOptions="CenterAndExpand">
                <Label Text="Firstname" FontSize="Default" FontAttributes="Bold"/>
                <Entry x:Name="entryfirstname" Placeholder="Type here.."></Entry>

                <Label Text="lasttname" FontSize="Default" FontAttributes="Bold"/>
                <Entry x:Name="entrylasttname" Placeholder="Type here.."></Entry>

                <Label Text="Email" FontSize="Default" FontAttributes="Bold"/>
                <Entry x:Name="entryEmail" Placeholder="Type here.." Keyboard="Email"></Entry>

                <Label Text="username" FontSize="Default" FontAttributes="Bold"/>
                <Entry x:Name="entryuUsername" Placeholder="Type here.."></Entry>

                <Label Text="password" FontSize="Default" FontAttributes="Bold"/>
                <Entry x:Name="entryPassword" Placeholder="Type here.." IsPassword="True"></Entry>

                <Button x:Name="btnRegister" Text="Register" Clicked="btnRegister_Clicked"></Button>
            </StackLayout>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

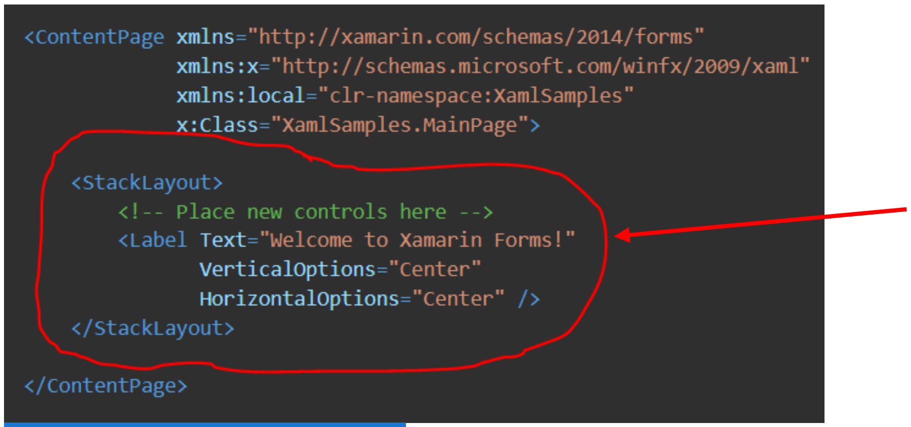


Main Page:

Both **App.xaml** and **App.xaml.cs** contribute to a class named App that derives from Application. Most other classes with XAML files contribute to a class that derives from ContentPage; those files use XAML to define the visual contents of an entire page. This is true of the other two files in the **XamlSamples** project:

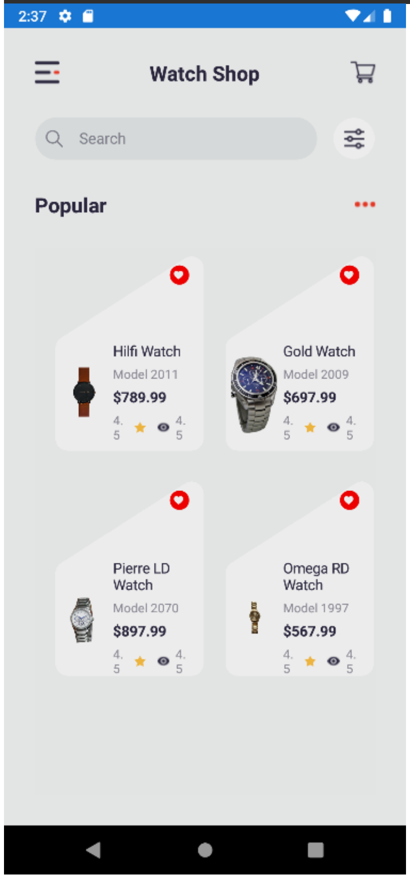
- **MainPage.xaml**, the XAML file; and
- **MainPage.xaml.cs**, the C# code-behind file.

The **MainPage.xaml** file looks like this (although the formatting might be a little different):



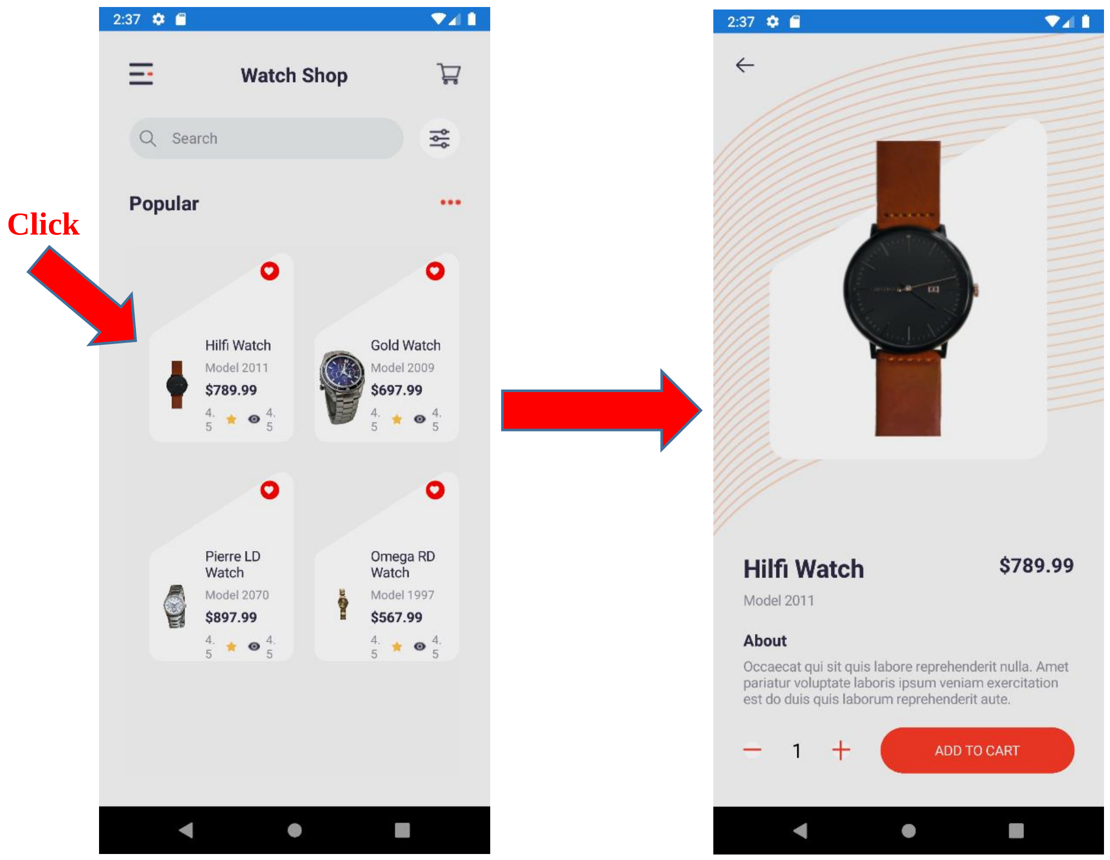
```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:XamlSamples"
             x:Class="XamlSamples.MainPage">

    <StackLayout>
        <!-- Place new controls here -->
        <Label Text="Welcome to Xamarin Forms!"
               VerticalOptions="Center"
               HorizontalOptions="Center" />
    </StackLayout>

</ContentPage>
```

We Change here



FlyoutPage(Details Page):

A flyout page typically displays a list of items, as shown in the following screenshots:



Occaecat qui sit quis labore reprehenderit nulla. Amet pariatur voluptate laboris ipsum veniam exercitation est do duis quis laborum reprehenderit aute.

The location of the list of items is identical on each platform, and selecting one of the items will navigate to the corresponding detail page. In addition, the flyout page also features a navigation bar that contains a button that can be used to navigate to the active detail page:

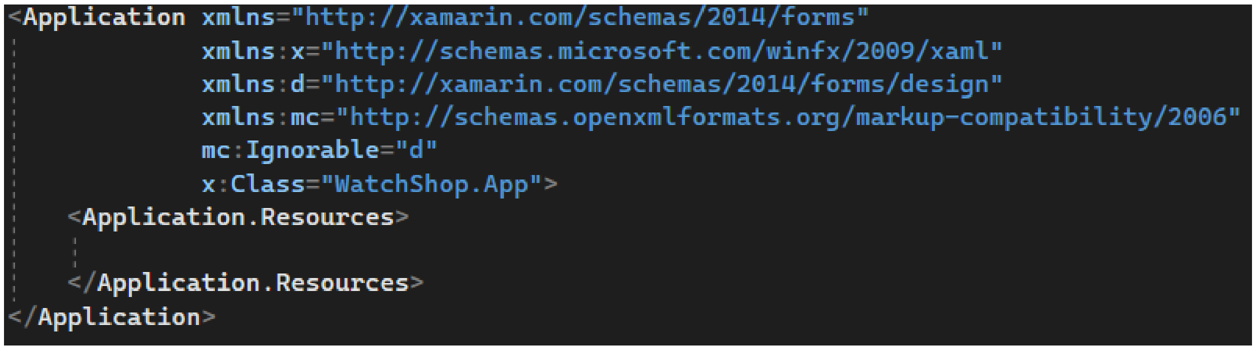
- On iOS, the navigation bar is present at the top of the page and has a button that navigates to the detail page. In addition, the active detail page can be navigated to by swiping the flyout to the left.
- On Android, the navigation bar is present at the top of the page and displays a title, an icon, and a button that navigates to the detail page. The icon is defined in the [Activity] attribute that decorates the MainActivity class in the Android platform-specific project. In addition, the active detail page can be navigated to by swiping the flyout page to the left, by tapping the detail page at the far right of the screen, and by tapping the Back button at the bottom of the screen.
- On the Universal Windows Platform (UWP), the navigation bar is present at the top of the page and has a button that navigates to the detail page.

Working with App.xaml:

App.xaml is the declarative starting point of your application. Visual Studio will automatically create it for you when you start a new WPF application, including a Code-behind file called App.xaml.cs. They work much like for a Window, where the two files are partial classes, working together to allow you to work in both markup (XAML) and Code-behind.

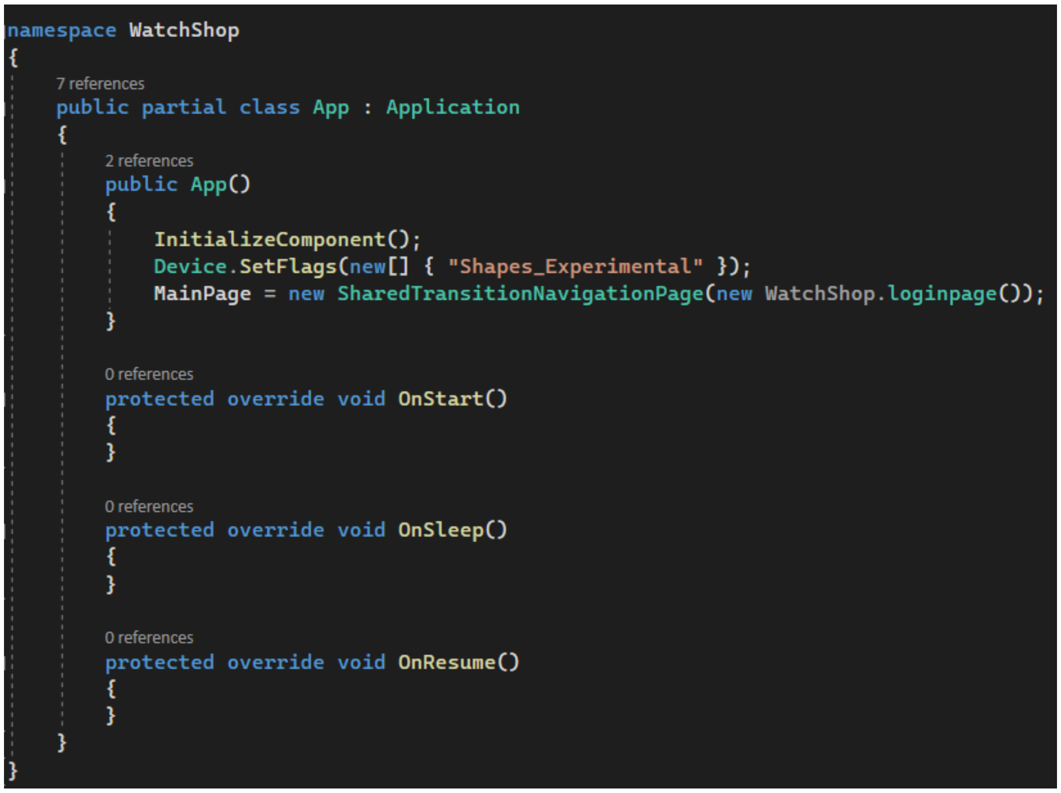
App.xaml.cs extends the Application class, which is a central class in a WPF Windows application. .NET will go to this class for starting instructions and then start the desired Window or Page from there. This is also the place to subscribe to important application events, like application start, unhandled exceptions and so on.

App.xaml structure:



```
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="WatchShop.App">
    <Application.Resources>

    </Application.Resources>
</Application>
```

App.xaml.cs structure:



```
namespace WatchShop
{
    7 references
    public partial class App : Application
    {
        2 references
        public App()
        {
            InitializeComponent();
            Device.SetFlags(new[] { "Shapes_Experimental" });
            MainPage = new SharedTransitionNavigationPage(new WatchShop.loginpage());
        }

        0 references
        protected override void OnStart()
        {
        }

        0 references
        protected override void OnSleep()
        {
        }

        0 references
        protected override void OnResume()
        {
        }
    }
}
```

Xamarin Advantages:

1. Native User Experience

2. Single Technological Stack

3. Shareable Code

4. Time and Costs Saving

4. Simplified Maintenance

5. Testing

6. Technical Support by Microsoft

Our Experience with Xamarin:

SaM Solutions' Xamarin experts have been providing services for more than five years and have successfully completed more than ten projects. Our competencies include:

- Business needs and requirements analysis
- App development
- Support and maintenance
- UI/UX design
- Legacy native mobile apps improvement and performance optimization
- Mobile testing services

One of our clients needed a solution with two separate mobile apps for two specific user groups to collaborate. Developing different apps for iOS and Android would have been time-consuming and would have resulted in high expenses along the way. That's why the client opted for Xamarin cross-platform development.

References:

https://docs.microsoft.com/en-us/xamarin

https://www.udemy.com/course/xamarin-forms-free

What is Xamarin - Javatpoint

Build Login in Xamarin with Xamarin.Forms | Okta Developer

How to connect .cs to .xaml.cs & .xaml (microsoft.com)

Advantages Of Using Xamarin For Your Mobile Projects (goodworklabs.com)

What is Xamarin? And know the Advantages & Disadvantages of Xamarin.Forms (quinterocorp.com)