Laboratório

Examinando a anatomia de um aplicativo Xamarin.Forms

Versão: 1.0.0
Outubro de 2016

Miguel Muñoz Serafín
@msmdotnet

Suporte: mmunoz@mail.ticapacitacion.com

CONTEÚDO

INTRODUÇÃO

EXERCÍCIO 1: EXPLORANDO A ESTRUTURA DE UM APLICATIVO XAMARIN.FORMS

Tarefa 1. Criar o aplicativo Xamarin.Forms.

Tarefa 2. Examinar os arquivos de código gerados.

RESUMO

Introdução

Uma interface de usuário moderna é construída por vários tipos de objetos visuais. Dependendo do sistema operacional, esses objetos visuais podem ter nomes diferentes, tais como Control, Element, View ou Widget. No entanto, todos eles têm a responsabilidade de fazer o trabalho de apresentação, interação, ou ambos.

Em Xamarin.Forms, os objetos que aparecem na tela são chamados de **Elementos Visuais**. Os elementos visuais são agrupados em 3 categorias principais:

- Page
- Layout
- View

A API Xamarin.Forms define classes chamadas **VisualElement**, **Page**, **Layout** e **View**. Essas classes e seus descendentes formam a espinha dorsal da interface do usuário Xamarin.Forms. **VisualElement** é excepcionalmente importante na classe Xamarin.Forms. Um objeto **VisualElement** é qualquer coisa que ocupe espaço na tela.

Um aplicativo Xamarin.Forms consiste em uma ou mais páginas. Uma página normalmente ocupa toda a tela (ou, pelo menos, uma área muito grande dela). Alguns aplicativos consistem apenas de uma página, enquanto outros permitem que você navegue entre várias páginas. Na maioria das vezes, os aplicativos trabalham simplesmente com um tipo de página chamada **ContentPage**.

Em cada página, os elementos visuais são organizados em uma hierarquia de pai e filho. O filho de um **ContentPage** geralmente é um elemento recipiente (*Layout*) que permite organizar os elementos visuais. Alguns recipientes têm apenas um filho, enquanto a maioria dos recipientes têm vários filhos que são organizados dentro dele. Estes filhos podem ser outros recipientes ou **Views** (Visualizações). Diferentes tipos de recipientes organizam seus filhos em um Stack, em uma grade bidimensional ou uma forma livre.

O termo **View** no Xamarin.Forms denota tipos familiares de objetos de apresentação e interação: texto, bitmaps, botões, campos de entrada de texto, sliders, switches, barras de progresso, data e seletores tempo e outros objetos de criação própria. Esses objetos são chamados frequentemente de Controles ou Widgets em outros ambientes de programação.

Neste laboratório, vamos explorar a estrutura de um aplicativo multiplataforma Xamarin.Forms. Em um próximo laboratório, analisaremos as diversar opções para compartilhar código em um aplicativo multiplataforma e explorararemos a *View **Label*** para exibir texto em um aplicação multiplataforma desenvolvida com Xamarin.Forms.

Objetivos

Ao finalizar este laboratório, os participantes serão capazes de:

- Descrever a estrutura de um aplicativo Xamarin.Forms.

Requisitos

Para a realização deste laboratório é necessário contar com o seguinte:

- Um equipamento de desenvolvimento com sistema operacional Windows 10 e Visual Studio 2015 Community, Professional ou Enterprise com a plataforma Xamarin.
- Um equipamento Mac com a plataforma Xamarin.

Tempo estimado para completar este laboratório: **60 minutos**.

Exercício 1: Explorando a estrutura de um aplicativo Xamarin.Forms

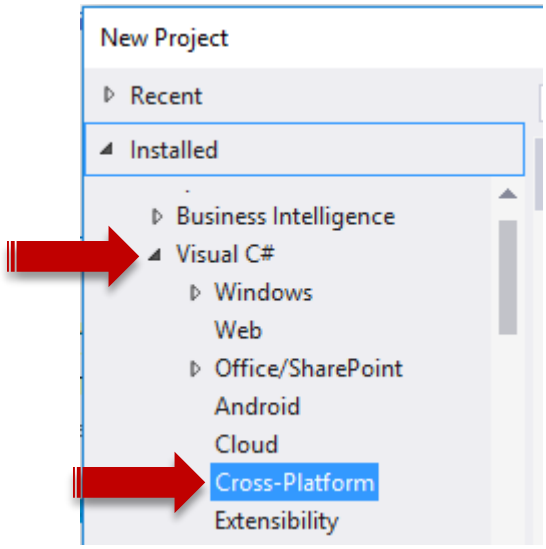
Neste exercício, você criará um aplicativo Xamarin.Forms e examinará a estrutura de arquivos de cada projeto que formam parte de uma solução uma multiplataforma Xamarin.Forms.

Tarefa 1. Criar um aplicativo Xamarin.Forms.

Nesta tarefa, você criará um aplicativo Xamarin.Forms usando o Microsoft Visual Studio e um modelo padrão. O processo irá criar uma solução que contém 6 projetos:

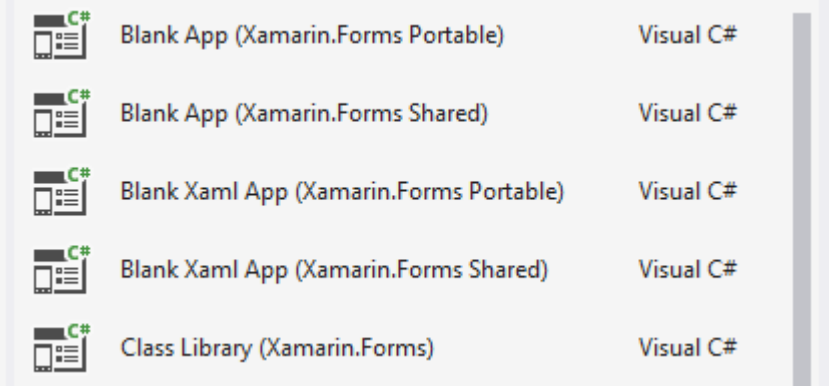
- 5 projetos específicos para cada plataforma. iOS, Android, UWP, o Windows 8.1 e Windows Phone.
- 1 projeto comum para a maior parte do código compartilhado do aplicativo.

1. No Visual Studio, selecione **File > New > Project**.
2. No painel esquerdo da janela **New Project**, selecione **Visual C # > Cross-Platform**.

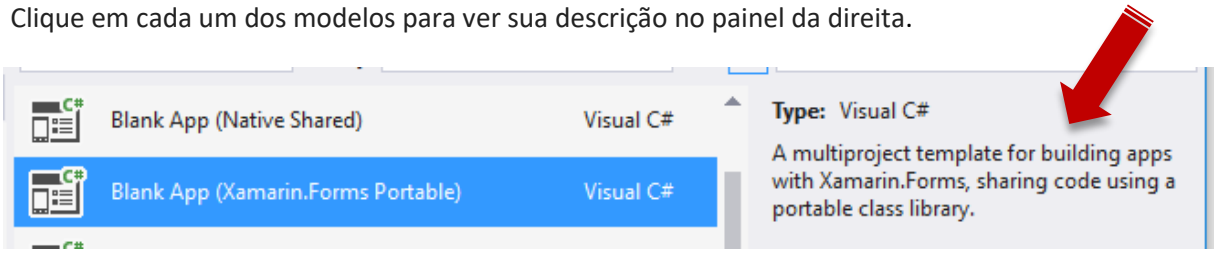


3. No painel central da janela **New Project,** você pode ver diferentes modelos de solução disponíveis incluindo 5 Xamarin.Forms:

    a. Blank App (Xamarin.Forms Portable)
    b. Blank App (Xamarin.Forms Shared)
    c. Blank Xaml App (Xamarin.Forms Portable)
    d. Blank Xaml App (Xamarin.Forms Shared)
    e. Class Library (Xamarin.Forms)



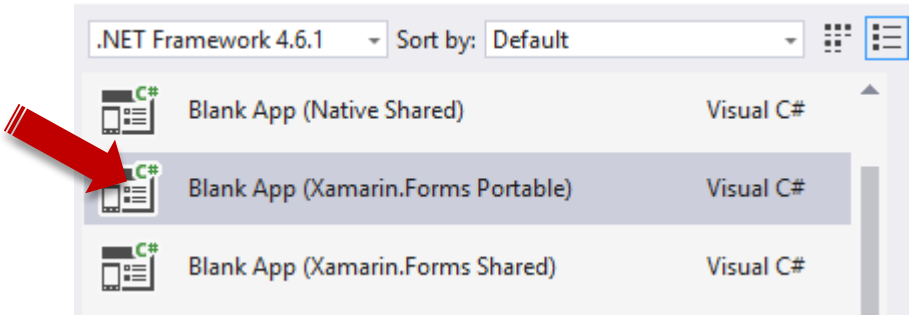
Clique em cada um dos modelos para ver sua descrição no painel da direita.



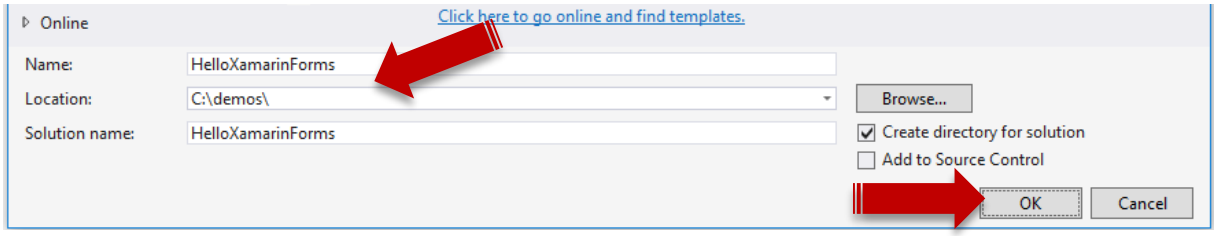
O termo **"Portable"** neste contexto refere-se a uma biblioteca de classes portátil (Portable Class Library – PCL). Todo o código comum do aplicativo torna-se uma DLL que é referenciada por todos os projetos de plataformas individuais.

O termo **"Shared"** neste contexto refere-se a um projeto de recursos partilhados (Shared Asset Project – SAP). Este projeto contém arquivos de código e outros arquivos que são compartilhados com cada projeto de plataforma, tornando-se essencialmente parte de cada projeto em cada plataforma.

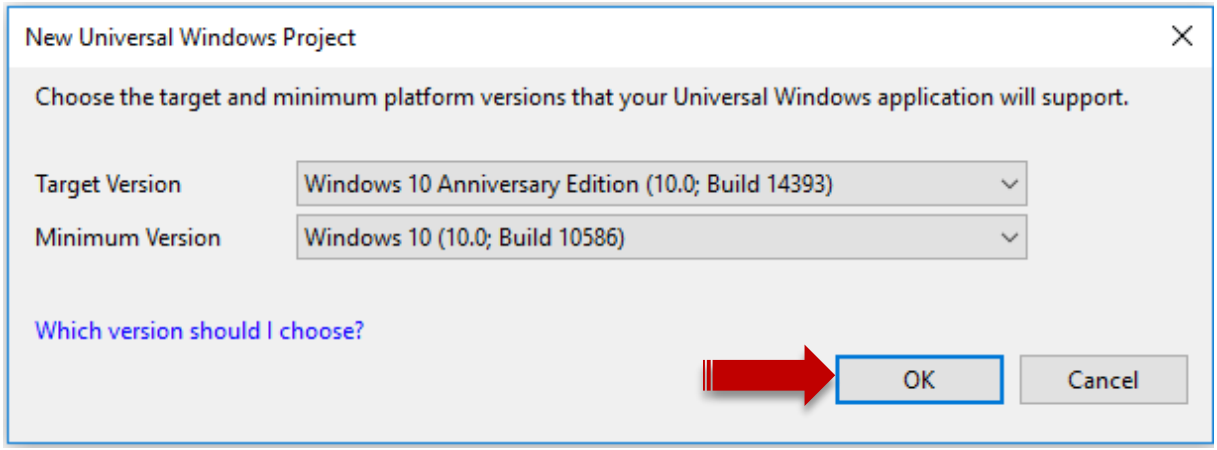
4. Selecione a opção **Blank App (Xamarin.Forms Portable)**.



5. Forneça o nome, local e clique em OK para criar o projeto.

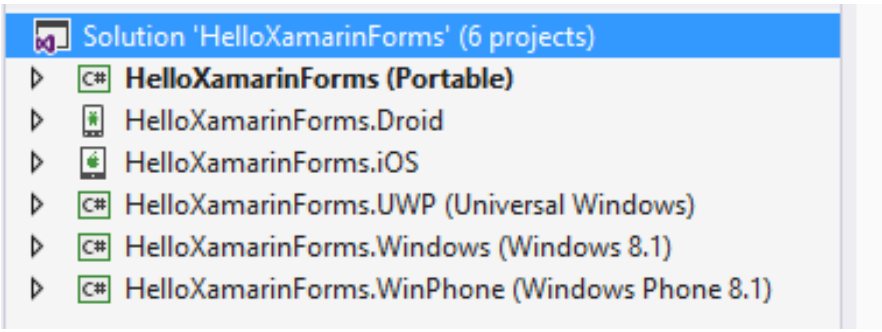


6. Clique no botão **OK** na caixa de diálogo **New Universal Windows Project** para aceitar as versões sugeridas para o aplicativo UWP que será criada.



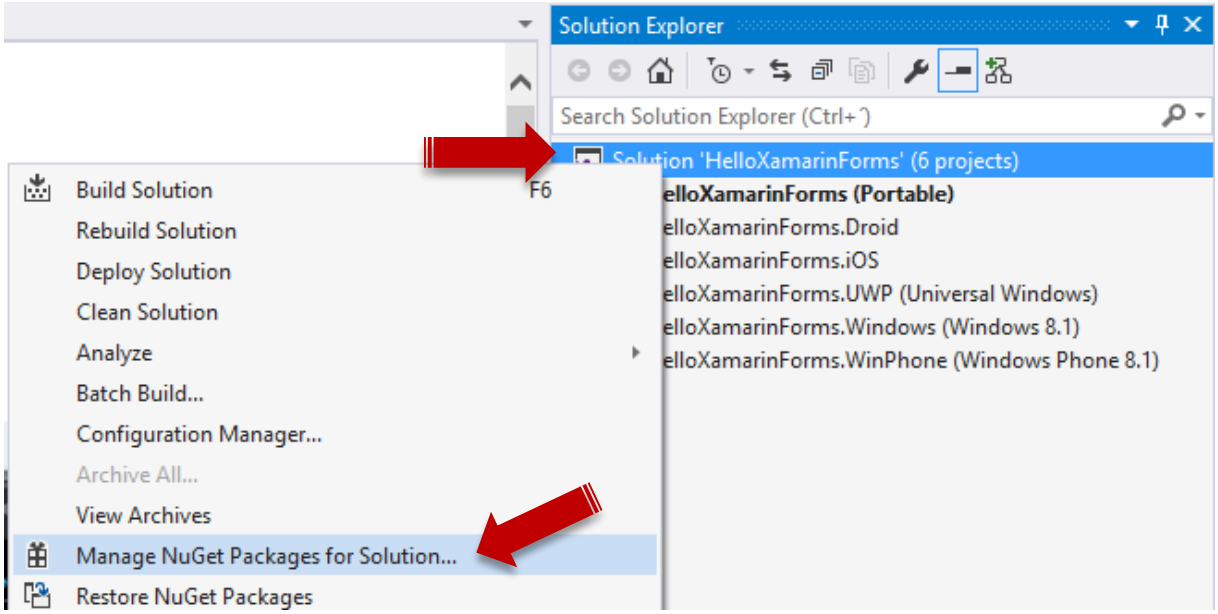
Seis projetos são criados. Para uma solução chamada HelloXamarinForms, esses projetos são:

a. Um projeto chamado **HelloXamarinForms** PCL que é referenciado por outros 5 projetos.
b. Um projeto de aplicativo para o Android chamado **HelloXamarin.Droid**.
c. Um projeto de aplicativo para iOS chamado **HelloXamarin.iOS**.
d. Um projeto de aplicativo para UWP de Windows 10 e Windows Mobile 10 chamado **HelloXamarinForms.UWP**.
e. Um projeto de aplicativo para o Windows 8.1 chamado **HelloXamarinForms.Windows**.
f. Um projeto de aplicativo para o Windows Phone 8.1 chamado **HelloXamarinForms.WinPhone**.

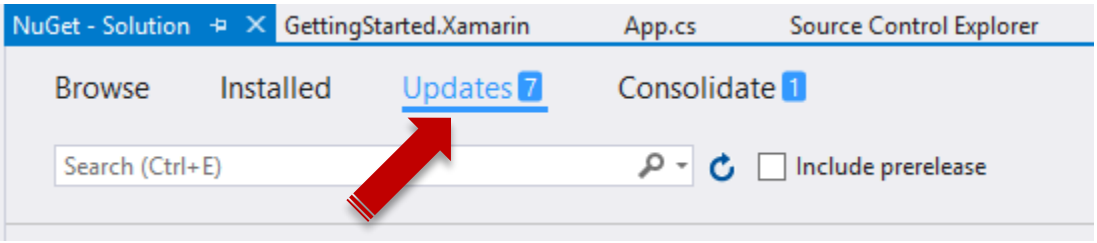


Quando você cria uma nova solução Xamarin.Forms, as bibliotecas Xamarin.Forms e outras bibliotecas auxiliares são automaticamente transferidas a partir do gerenciador de pacotes NuGet. O Visual Studio armazena essas bibliotecas em um diretório chamado **packages** dentro do diretório da solução. No entanto, a versão específica da biblioteca Xamarin.Forms que é baixada é especificada dentro do modelo da solução e, portanto, uma nova versão poderia estar disponível.

7. Selecione a opção **Manage NuGet Packages** for Solution no menu de contexto do nome da solução.

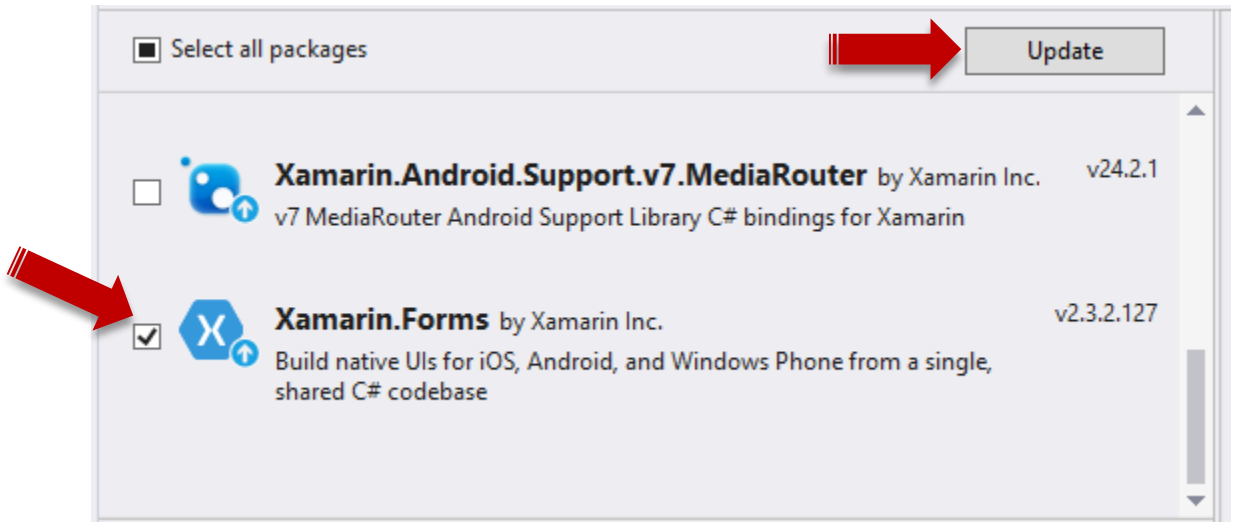


8. Clique na opção **Updates** para ver as atualizações disponíveis.



Visual Studio pode indicar que há atualizações para o pacote NuGet Xamarin.Forms e todas as suas dependências, no entanto, Xamarin.Forms está configurado para dependências de versões específicas. Portanto, embora o Visual Studio indique que há novas versões disponíveis de pacotes Xamarin.Android.Support, Xamarin.Forms não é necessariamente compatível com estas novas versões.

9. Selecione a opção **Xamarin.Forms** e clique em **Update** para iniciar a atualização.



É provável que seja solicitado aceitar o acordo de licenciamento e reiniciar Visual Studio para concluir a instalação.

Se você selecionar todos os pacotes, a seguinte mensagem de erro pode aparecer, devido ao problema de compatibilidade de verões mencionado anteriormente:

Unable to resolve dependencies. 'Xamarin.Android.Support.Design 24.2.1' is not compatible with 'Xamarin.Forms 2.3.2.127 constraint: Xamarin.Android.Support.Design (= 23.3.0)'

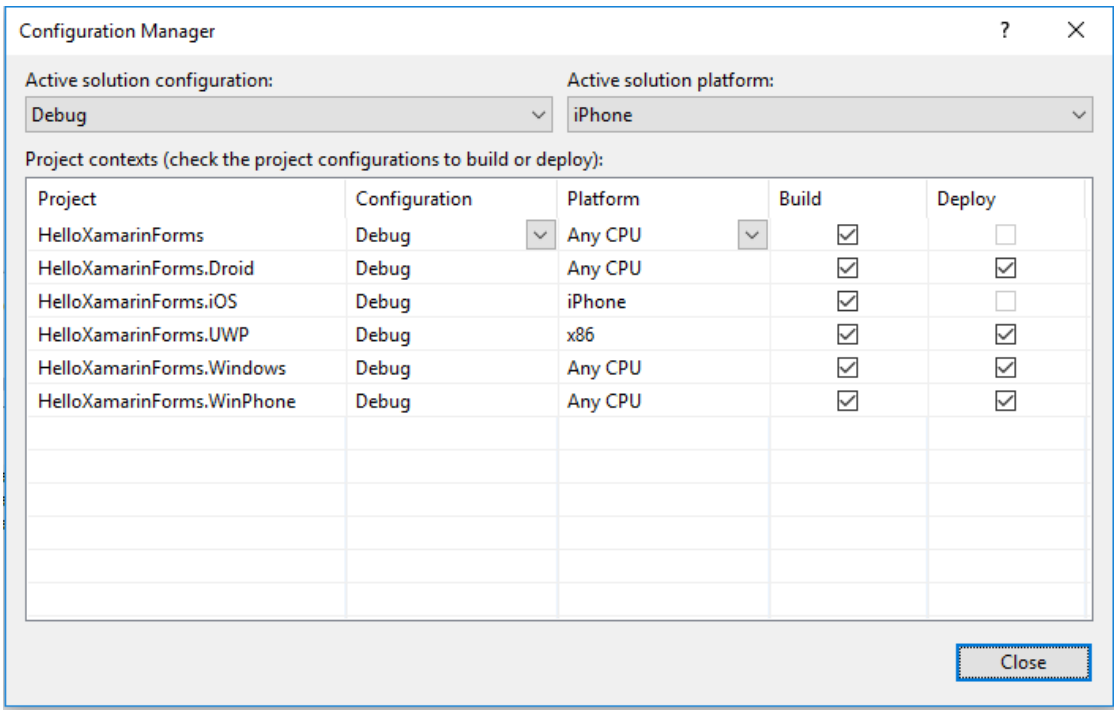
Você pode verificar o seguinte link para obter mais informações sobre a mensagem de erro.

https://developer.xamarin.com/guides/xamarin-forms/troubleshooting/

10. Uma vez que a atualização for concluída, selecione **Build > Configuration Manager**.

11. Na caixa de diálogo **Configuration Manager** poderá ver o projeto PCL e os outros 5 projetos de aplicativo.

    Verifique se a caixa de seleção **Build** está selecionada para todos os projetos e que a caixa de seleção **Deploy** está selecionada para todos os projetos de aplicativo (a menos que a caixa seja cinza).



Observe que na coluna **Platform**:

- O projeto PCL tenha estabelecido a única opção disponível **Any CPU**.
- O projeto Android tenha estabelecido a única opção disponível **Any CPU**.
- O projeto iOS pode assumir os valores **iPhone** ou **iPhoneSimulator** dependendo de como testaremos o aplicativo.
- O projeto UWP pode ter o valor **x86** para implantar o aplicativo para o desktop do Windows ou para um emulador. Você também pode ter o valor **ARM** para implantar o aplicativo a um telefone.
- O projeto Windows pode ter o valor **x86** para implantar o aplicativo para o desktop do Windows ou para um emulador. Você também pode ter o valor **ARM** para implantar o aplicativo para um tablet. Você pode ter até o valor **x64** para implantar o aplicativo para plataformas de 64 bits ou de **Any CPU** para implementar o aplicativo sobre as plataformas de desktop do Windows, emulador, tablet ou plataforma de 64 bits.
- O projeto WinPhone pode levar o valor **x86** para implantar o aplicativo para um emulador, o valor **ARM** para implantar o aplicativo em um telefone real ou Qualquer valor **Any CPU** para implantar o aplicativo, tanto no emulador e telefone real.

12. Clique em **Close** para fechar a caixa de diálogo **Configuration Manager**.

13. Opcionalmente, se quiser exibir a barra de ferramentas iOS e Android para acessar os dispositivos e emuladores correspondentes, você pode ativar as opções de **View > Toolbars > iOS** e **View > Toolbars > Android**.

14. Selecione a opção **Set As StartUp Project** a partir do menu de contexto do projeto Android para estabelecer o aplicativo Android como projeto de inicialização.

15. Exiba o aplicativo no emulador de sua preferência.

    Você vai ver o aplicativo implantado no emulador Android selecionado.

16. Volte ao Visual Studio e pare a execução.

17. Escolha a opção **Set As StartUp Project** a partir do menu de contexto do projeto iOS iOS para estabelecer o aplicativo como projeto de inicialização.

18. Exiba o aplicativo no emulador de sua escolha.

No Mac, você vai ver o aplicativo implantado no emulador iOS que você selecionou.

19. Volte ao Visual Studio e pare a execução.
20. Selecione a opção Set As Startup Project do menu contextual do projeto UWP para estabelecer o aplicativo UWP como projeto de início.
21. Implante o aplicativo no emulador de Windows Mobile de sua escolha.

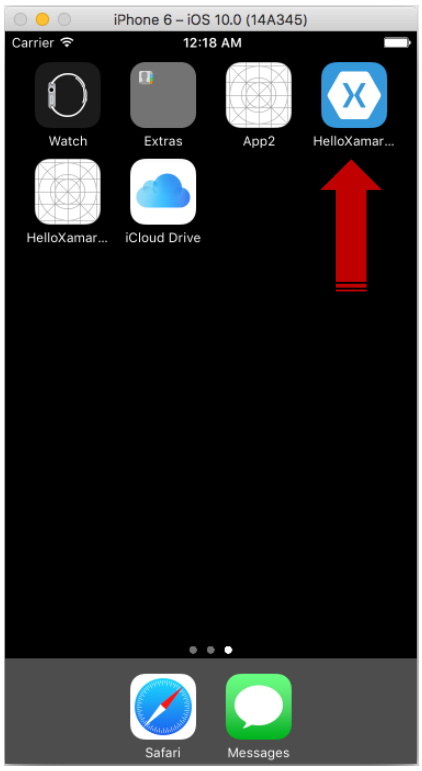
    Você vai ver o aplicativo implantado no emulador Windows Mobile selecionado.

22. Volte ao Visual Studio e pare a execução.

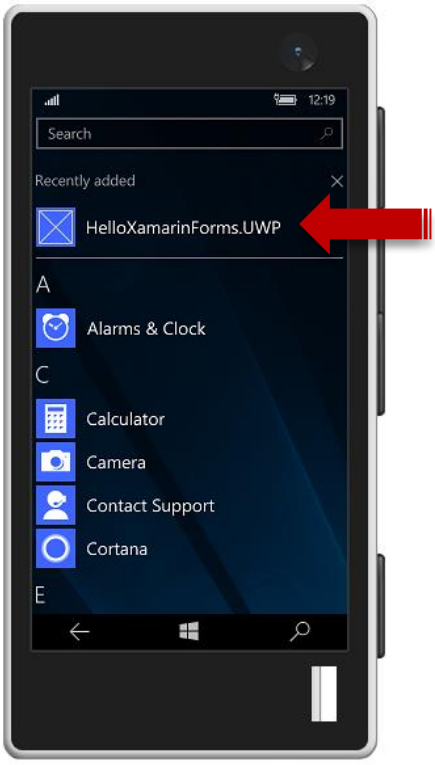
Por padrão, todas as plataformas estão habilitados para mudanças de orientação. Você pode girar qualquer um dos emuladores e você percebe que o texto é ajustado para o novo centro da tela.

O aplicativo não só é executado no dispositivo ou emulador. A aplicação aparece com os outros aplicativos no telefone ou emulador e pode ser executado a partir de lá.

Android



iOS



Windows 10 Mobile

Se você não gosta do ícone do aplicativo ou a forma como o nome do aplicativo é exibida, você pode mudar isso nos projetos individuais de cada plataforma.

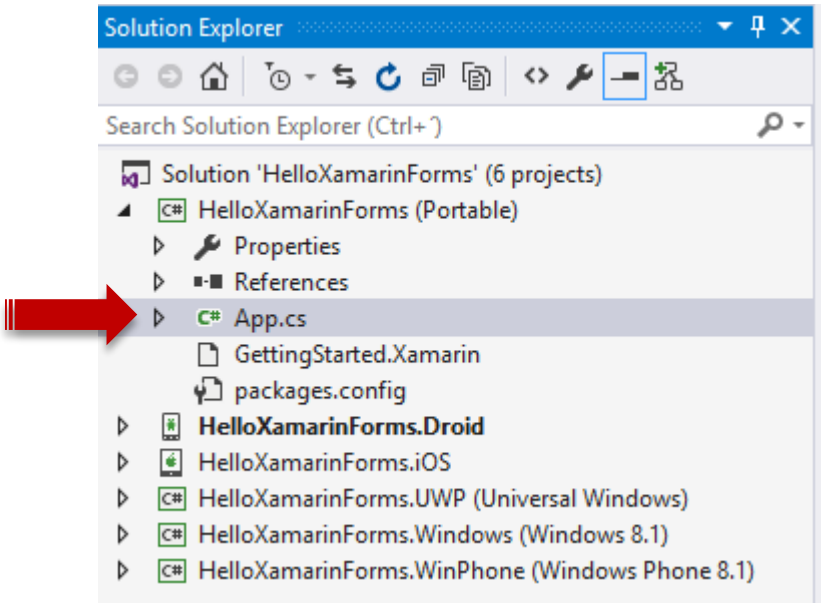
Tarefa 2. Examine os arquivos de código gerados.

Claramente, o programa criado pelo modelo Xamarin.Forms é muito simples, portanto, representa uma excelente oportunidade para examinar os arquivos de código gerados, descobrir como eles se relacionam e como eles funcionam.

O projeto PCL é o projeto que receberá a maior parte de nossa atenção quando estivermos escrevendo uma aplicação Xamarin.Forms. Em algumas circunstâncias, o código neste projeto poderia exigir algum código especial de certas plataformas.

1. Vamos começar com o código que é responsável por apresentar o texto que você vê na tela. Esta é a classe **App** no PCL projeto comum.

   Abra o arquivo **App.cs** localizado dentro do projeto PCL.



2. Examine a definição da classe App.

   Você pode notar que o espaço para nome é o mesmo que o nome do projeto.

```
namespace HelloXamarinForms
{
    public class App : Application
    {
```

   Você também pode perceber que a classe é definida como pública e derivada da classe **Xamarin.Forms.Application**.

3. Examine o código do construtor.

A única responsabilidade do construtor é estabelecer o valor da propriedade **MainPage** da classe **Application** para um objeto do tipo **Page**.

```
public App()
{
    // The root page of your application
    var content = new ContentPage
    {
        Title = "HelloXamarinForms",
        Content = new StackLayout
        {
            VerticalOptions = LayoutOptions.Center,
            Children = {
                new Label {
                    HorizontalTextAlignment = TextAlignment.Center,
                    Text = "Welcome to Xamarin Forms!"
                }
            }
        }
    };

    MainPage = new NavigationPage(content);
}
```

A classe **ContentPage** deriva da classe **Page** e é muito comum utilizá-la em aplicativos Xamarin.Forms de uma página. **ContentPage** ocupa a maior parte da tela, porque, dependendo da plataforma, pode dividir o espaço com barras de ferramentas ou de estado.

A classe **ContentPage** define uma propriedade chamada **Content**, para a qual estabelecemos o conteúdo da página. Geralmente, o conteúdo é um espaço que comporta diversos **Views**. No caso do código gerado, é um **StackLayout** que acomoda seus filhos em uma pilha.

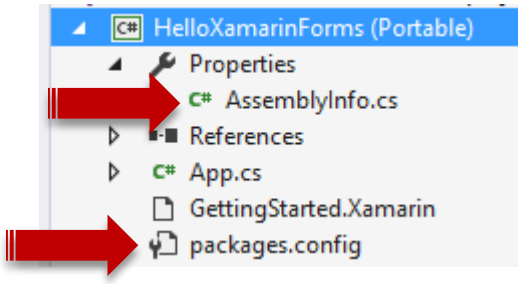
A propriedade **VerticalOptions** do **StackLayout** permite especificar o alinhamento vertical do conteúdo.

No código gerado, **StackLayout** contém um único filho que é uma **View Label**. A classe Label deriva da classe **View** e é usada em aplicativos Xamarin.Forms para exibir textos.

A propriedade **HorizontalTextAlignment** de Label especifica o alinhamento horizontal do conteúdo armazenado na propriedade **Text**.

Na vida real, em nossos aplicativos Xamarin.Forms de uma única página, geralmente definimos nossa própria classe que deriva de **ContentPage**. O construtor da classe App estabelece em sua propriedade **MainPage** uma instância da classe que definimos.

4. Observe que projeto **PCL** contém o arquivo AssemblyInfo.cs com informação de **Assembly**. Contém também o arquivo packages.config, que lista os pacotes NuGet exigidos pelo programa.



5. Abra a secção **References** do projeto PCL e note que existem ao menos 4 bibliotecas que o projeto PCL requer.

    - .NET
    - Xamarin.Forms.Core
    - Xamarin.Forms.Platform
    - Xamarin.Forms.Xaml

Os 5 projetos de aplicativo tem seus próprios recursos (assets) em forma de ícones e metadados. É importante prestar atenção a esses recursos ao publicar o aplicativo na plataforma, mas enquanto você está aprendendo a desenvolver aplicativos Xamarin.Forms pode ignorar por um momento esses recursos. Você pode também manter contraídos os projetos desses aplicativos, visto que não precisa se preocupar muito com o seu conteúdo.

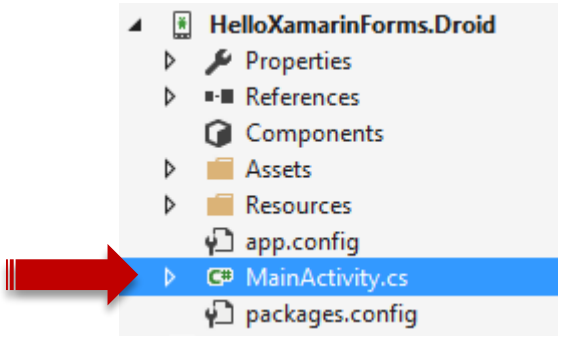
Vamos ver o que há dentro desses projetos de aplicativos.

6. Examine a seção References de cada projeto de aplicativo. Lá você verá a referência para o projeto comum PCL, assim como vários Assemblies .NET, os Assemblies Xamarin.Forms listados anteriormente e os Assemblies adicionais Xamarin.Forms aplicáveis a cada plataforma:

    - Xamarin.Forms.Platform.Android
    - Xamarin.Forms.Platform.iOS
    - Xamarin.Forms.Platform.UAP (no mostrado explicitamente no projeto UWP)
    - Xamarin.Forms.Platform.WinRT
    - Xamarin.Forms.Platform.WinRT.Tablet
    - Xamarin.Forms.Platform.WinRT.Phone

    Cada uma dessas bibliotecas define um método estático **Forms.Init** no namespace **Xamarin.Forms** que inicializa o sistema Xamarin.Forms para essa plataforma particular. O código de inicialização em cada plataforma deve fazer uma chamada para esse método. O código de inicialização de cada plataforma também deve instanciar a classe **App** do projeto PCL.
    Vamos ver como se faz o código de inicialização para cada plataforma.

7. Abra o arquivo **MainActivity.cs** do projeto Android.



Em um aplicativo Android, a classe típica **MainActivity** deve ser derivada de uma classe Xamarin.Forms chamada **FormsAppCompatActivity** definida na Assembly **Xamarin.Forms.Platform.Android.**

```
[Activity(Label = "HelloXamarinForms", Icon = "@drawable/icon",
    Theme = "@style/MainTheme", MainLauncher = true,
    ConfigurationChanges =
    ConfigChanges.ScreenSize | ConfigChanges.Orientation)]

public class MainActivity :
    global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
```

O atributo definido na classe **MainActivity** indica que a atividade não será recriada quando o telefone muda a sua orientação (vertical para horizontal e vice-versa) ou quando você mudar o tamanho da tela.

8. Examine o código do método **OnCreate**. Neste método é realizada a chamada para o método **Forms.Init**.

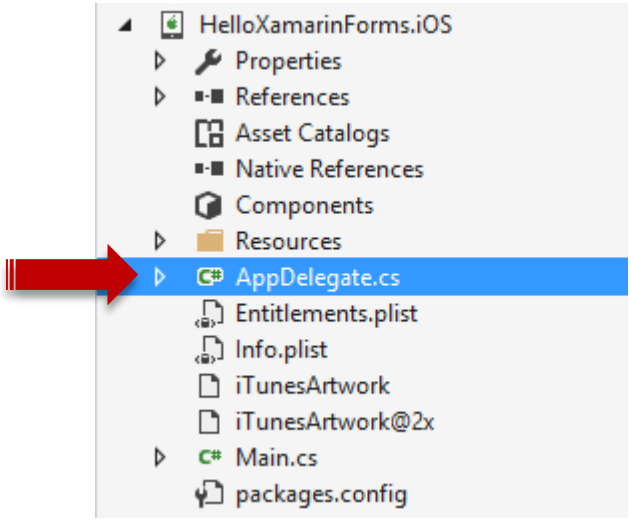
```
protected override void OnCreate(Bundle bundle)
{
    TabLayoutResource = Resource.Layout.Tabbar;
    ToolbarResource = Resource.Layout.Toolbar;

    base.OnCreate(bundle);

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

Note que a nova instância da classe **App** definida no projeto PCL é passada para o método **LoadApplication** definido por **FormsAppCompatActivity**.

9. Abra o arquivo **AppDelegate.cs** dentro do projeto iOS.



Um projeto iOS normalmente contém uma classe que deriva de **UIApplicationDelegate**. No entanto, a biblioteca **Xamarin.Forms.Platform.iOS** define uma classe base alternativa chamada **FormsApplicationDelegate**.

A seguir está o código que define a classe **AppDelegate** que deriva de **FormsApplicationDelegate**.

```
[Register("AppDelegate")]
public partial class AppDelegate :
    global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
```

10. Examine o código do método **FinishedLaunching**.

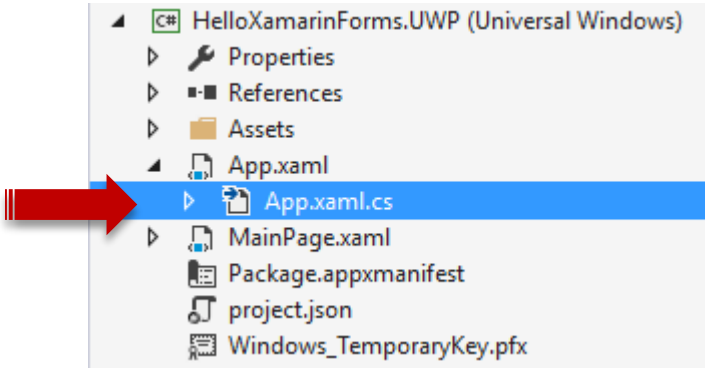
```
public override bool FinishedLaunching(UIApplication app,
    NSDictionary options)
{
    global::Xamarin.Forms.Forms.Init();
    LoadApplication(new App());

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

Este método começa com uma chamada ao método Forms.Init definido na Assembly **Xamarin.Forms.Platform.iOS**. Em seguida, chama o método **LoadApplication** definido na classe **FormsApplicationDelegate**, passando uma nova instância da classe **App** definida no projeto PCL.

O objeto **Page** atribuído à propriedade **MainPage** desse objeto App pode ser usado para criar um objeto do tipo **UIViewController**, que é responsável por gerar o conteúdo da página.

11. No projeto UWP ou em qualquer um dos projetos Windows, abra o arquivo **App.xaml.cs**.



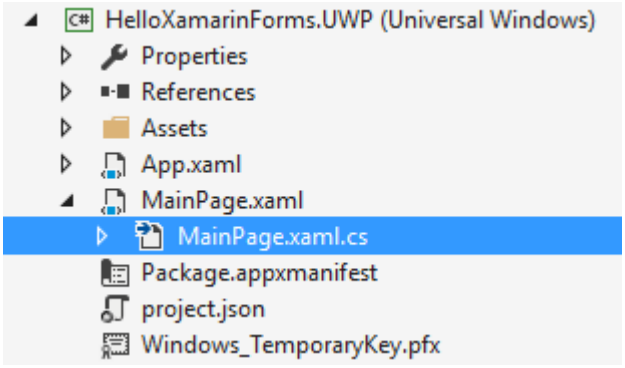
No arquivo **App.xaml.cs** é definida uma classe chamada **App** que deriva da classe **App** definida na biblioteca **Windows.UI.Xaml**.

```
sealed partial class App : Application
```

12. Examine o código do método **OnLaunched**. O código neste método chama o método **Forms.Init** da seguinte forma:

```
Xamarin.Forms.Forms.Init(e);
```

13. Abra agora o arquivo **MainPage.xaml.cs**.



Este arquivo define uma classe chamada **MainPage** que deriva de uma classe Xamarin.Forms especificada no elemento raiz do arquivo **MainPage.xaml**.

14. Examine o código para o construtor da classe **MainPage**.

```
public MainPage()
{
    this.InitializeComponent();

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

Neste método, uma nova instância da classe **App** é passada para o método **LoadApplication** definido na classe base **Xamarin.Forms.Platform.UWP.WindowsBasePage**.

Resumo

Neste laboratório, você explorou a anatomia de um aplicativo Xamarin.Forms.

Se você criou uma solução Xamarin.Forms e não quer desenvolver para uma ou mais plataformas, simplesmente elimine esses projetos.

Se mais tarde você mudar de idéia sobre esses projetos, você pode adicionar novos projetos da plataforma desejada para a solução Xamarin.Forms.

Em conclusão, não há nada especial entre os aplicativos Xamarin.Forms em comparação a aplicativos normais Xamarin ou projetos Windows, exceto as bibliotecas Xamarin.Forms.

Agora é o momento para explorar as diferenças entre as opções disponíveis para compartilhar código em uma aplicação multiplataforma com Xamarin.Forms.

Quando tiver terminado este laboratório, publique a seguinte mensagem no Twitter e no Facebook:

Eu terminei o #Lab02 na #MaratonaXamarin e conheço a estrutura de um aplicativo Xamarin.Forms!