



## Aula 01 – Programação de Dispositivos Móveis

Professor: Me. Wellington Tuler Moraes  
Curso: ADS – IFSP – Campus Cubatão

## Bibliografia

- **C# 6 and .NET Core 1.0**
  - by Mark J. Price
  - Publisher: PacktPublishing
  - Release Date: March 2016
- **Creating Mobile Apps with Xamarin.Forms**
  - by Charles Petzold
  - Publisher: Microsoft Press
  - Release Date: 2016

## Bibliografia

- **XamarinMobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals**
  - by Dan Hermes
  - Publisher: Apress
  - Release Date: July 2015

## Panorama da plataforma Mobile

- Rápido potencial para mudanças.
- Duas principais plataformas:
  - Família Apple com iPhones e iPads rodando iOS.
  - Sistema operacional Android, desenvolvido pela Google baseado no kernel Linux para celulares e tablets.
- Terceira:
  - Microsoft Windows Phone e Windows 10 Mobile.

## Problemas para Desenvolvedores

- Interfaces.
  - Facilidade para uso com toque, porém diferentes navegações, menus e aplicações.
- Ambientes de desenvolvimento.
  - iOS: Xcode no Mac.
  - Android: Android Studio em diversas plataformas.
  - Windows: Visual Studio no PC.
- Interfaces de programação (API).
  - No iPhone ou iPad, é uma view chamada **UISwitch**.
  - Dispositivos Android é uma ferramenta (widget) **Switch**.
  - Na API de execução Windows, é um controle chamado **ToggleSwitch**.

## Problemas para Desenvolvedores

- Diferentes linguagens de programação.
  - Objective-C para iPhone and iPad.
  - Java para dispositivos Android.
  - C# for Windows.
- Objective-c, Java e C# são primos por serem orientados a objetos e derivados da linguagem C.
- Se tornaram primos distantes.

## Uso do C#

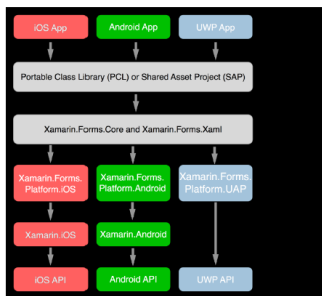
- Associado ao Microsoft .NET Framework.
- Baixo nível .NET provê os tipos básicos (int, double, string e etc..)
- Porém as bibliotecas de classe C# provê suporte para diferentes tipos
  - Math
  - Debugging
  - Reflection
  - Collections
  - Globalization
  - File I/O
  - Networking
  - Security
  - Threading
  - Web services
  - Data handling
  - XML and JSON reading and writing

**Exemplos =>**

## Compartilhando código

- Única Linguagem para diferentes plataformas.
- Arquitetura MVC (Model-View-Controller).
  - Mais recentemente MVVM (Model-View-ViewModel).
- MVVM ajuda na multiplataforma separando o código de determinada plataforma em Views.
- Parte independente de plataforma pode ser isolada em um projeto no Visual Studio - VS.
  - Códigos que acessam uma PCL (Portable Class Library) possível incluir até DLLs.

## Plataforma Compartilhada

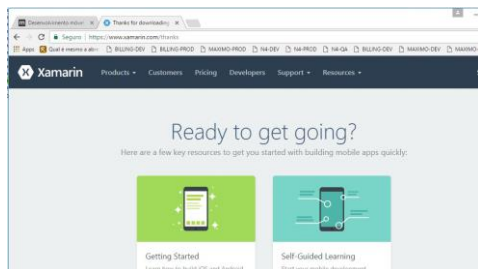


## Criando aplicações para dispositivos móveis

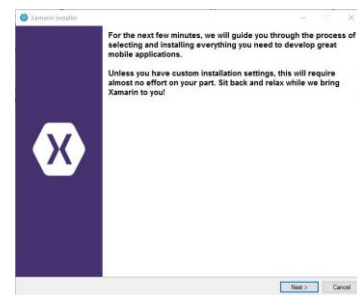
- Xamarin: Adequado para programadores C# que desejam escrever programas usando um único código base para as mais populares plataformas:
  - iOS.
  - Android.
  - Windows.
- Xamarin.Forms permite compartilhar e escrever interfaces em C# e XAML (Extensible Application Markup Language) – mapeiam os controles nativos das plataformas.



## Baixando Pacote – VS 2015



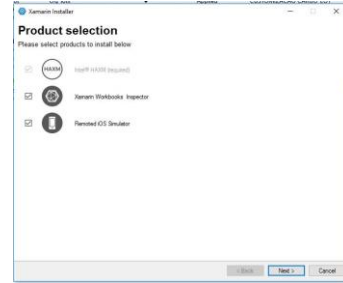
## Baixando Pacote



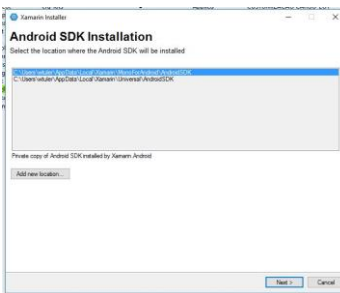
## Baixando Pacote



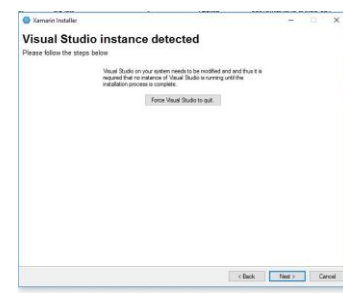
## Baixando Pacote



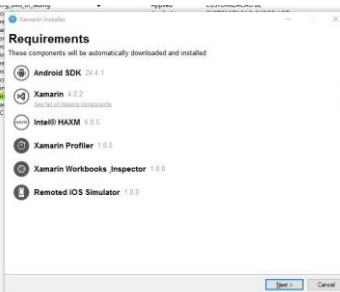
## Baixando Pacote



## Baixando Pacote



## Baixando Pacote



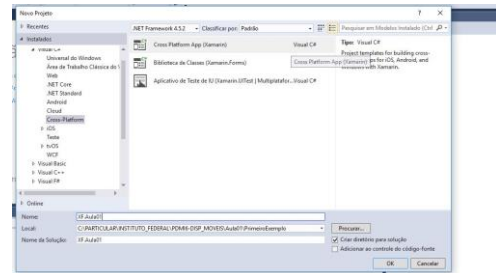
## Baixando Pacote



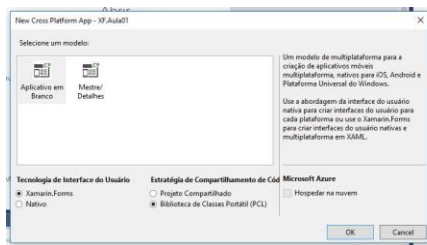
## Instalando Pacote



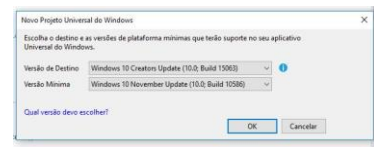
## Criando novo Projeto VS 2017



## Criando novo Projeto VS 2017

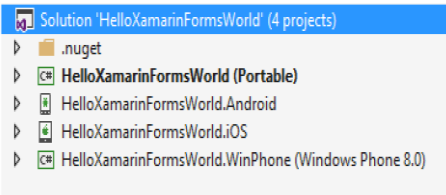


## Criando novo Projeto VS 2017

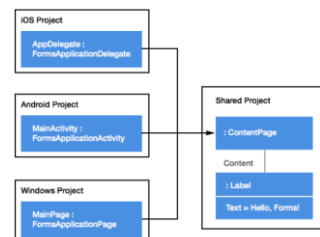


## Solution Explorer

The Solution Explorer for a cross-platform "Hello, World!" app might look like this:



## Hello, Xamarin.Forms



## Xamarin.Forms Namespace

- Kit de abstração para interface de usuário que permite aos desenvolvedores criar interface para diferentes plataformas.
- A interface utiliza controles nativos para a plataforma alvo, tendo aparência semelhante e é responsivo como aplicações nativas.
- Dependência é em uma via: Plataforma depende do projeto compartilhado, mas o projeto não depende da plataforma.

## Exemplo c#

```
public class App : Xamarin.Forms.Application
{
    public App ()
    {
        MainPage = new ContentPage
        {
            Content = new Label
            {
                Text = "Hello, Forms !",
                VerticalOptions = LayoutOptions.CenterAndExpand,
                HorizontalOptions = LayoutOptions.CenterAndExpand,
            },
        };
    }
}
```

## Plataforma iOS

```
[Register("AppDelegate")]
public class AppDelegate : FormsApplicationDelegate
{
    UIWindow window;

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

        LoadApplication (new App ());

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

## Plataforma Android

```
[Activity(Label = "HelloXamarinFormsWorld", MainLauncher = true)]
public class MainActivity : FormsApplicationActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);

        Forms.Init(this, bundle);

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

## Windows Phone

```
public partial class MainPage
{
    public MainPage()
    {
        InitializeComponent();

        Forms.Init();

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

## Resultado



## Interfaces do usuário nativas

- Os aplicativos criados com o Xamarin contêm controles de interface do usuário nativos padrão.
- Os aplicativos não têm apenas a aparência que o usuário final espera, mas também se comportam dessa forma.



## Acesso à API nativa

- Os aplicativos criados com o Xamarin têm acesso a todas as funcionalidades expostas pela plataforma e pelo dispositivo subjacentes.
- Inclui recursos específicos da plataforma, como o ARKit e o modo de várias janelas do Android.



## Desempenho Nativo

- Os aplicativos criados com o Xamarin aproveitam a aceleração de hardware específica da plataforma e são compilados para desempenho nativo.
- Isso não pode ser obtido com soluções que interpretam código em tempo de execução

