Xamarin Forms

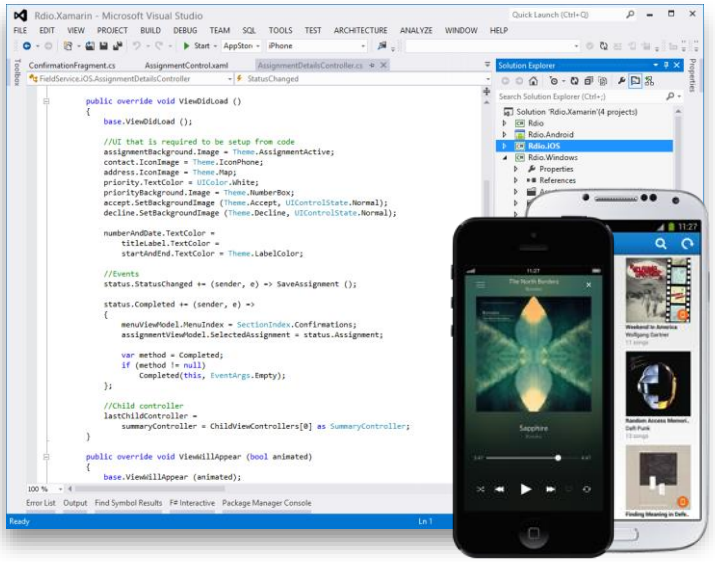
RUI FLEXA - MCPD

Uma base de código para três plataformas

A plataforma Xamarin permite a utilização do C# para a criação de aplicações que podem ser construídas para diferentes plataformas, como iOS e Android. A ideia é que o mesmo código seja capaz de criar aplicações nativas para essas plataformas. Além disso, o Xamarin também permite a criação de aplicações desktop para OS X bem como para o Windows 10 (Universal Windows Platform)

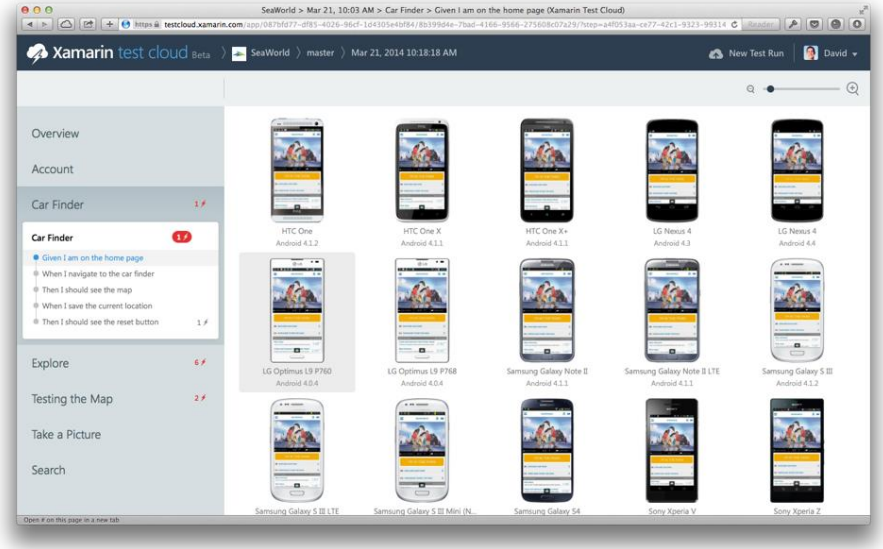
Xamarin Platform

Crie aplicações nativas para iOS, Android, Mac and Windows apps com Visual Studio e C#



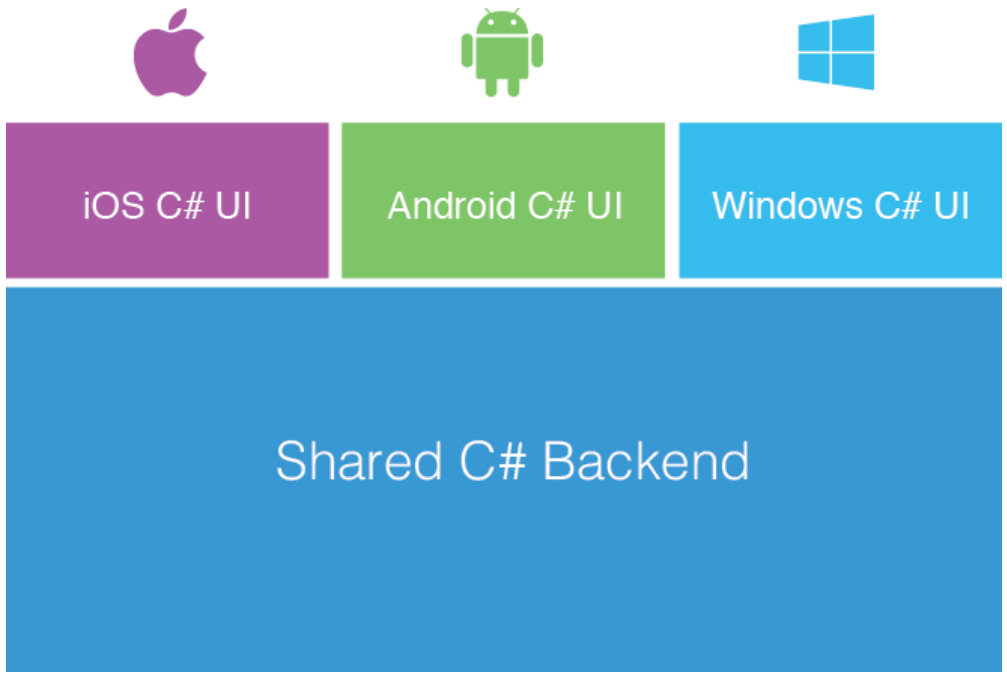
Xamarin test cloud

Teste automaticamente sua aplicação para milhares de dispositivos móvies

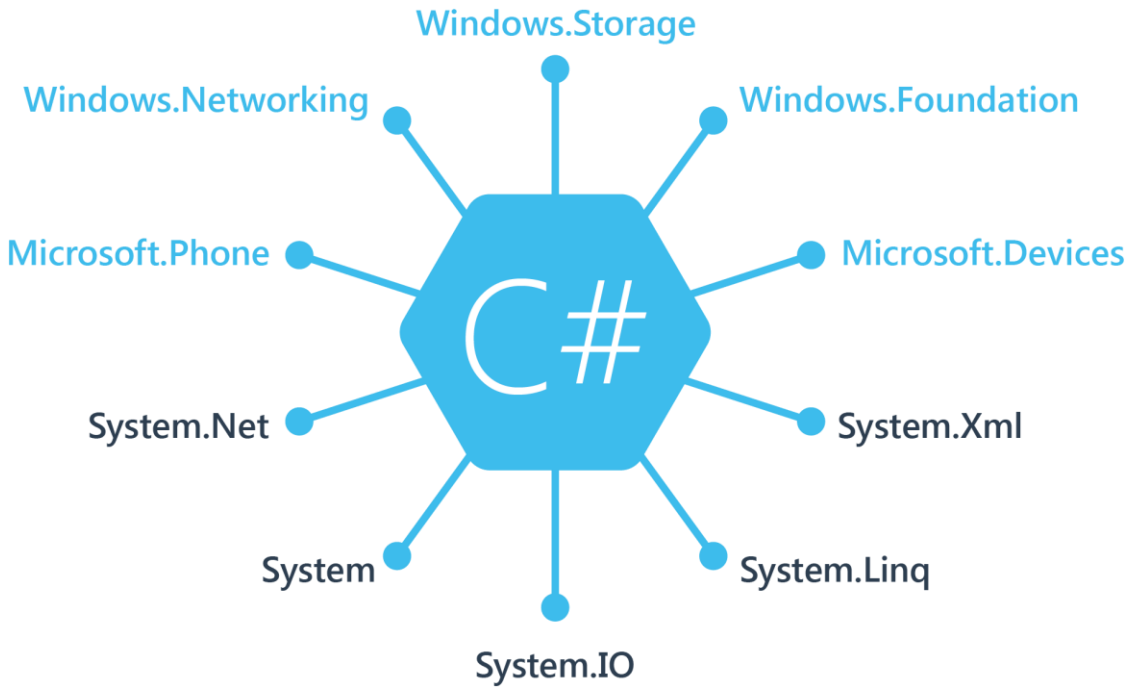


Abordagem única do Xamarin

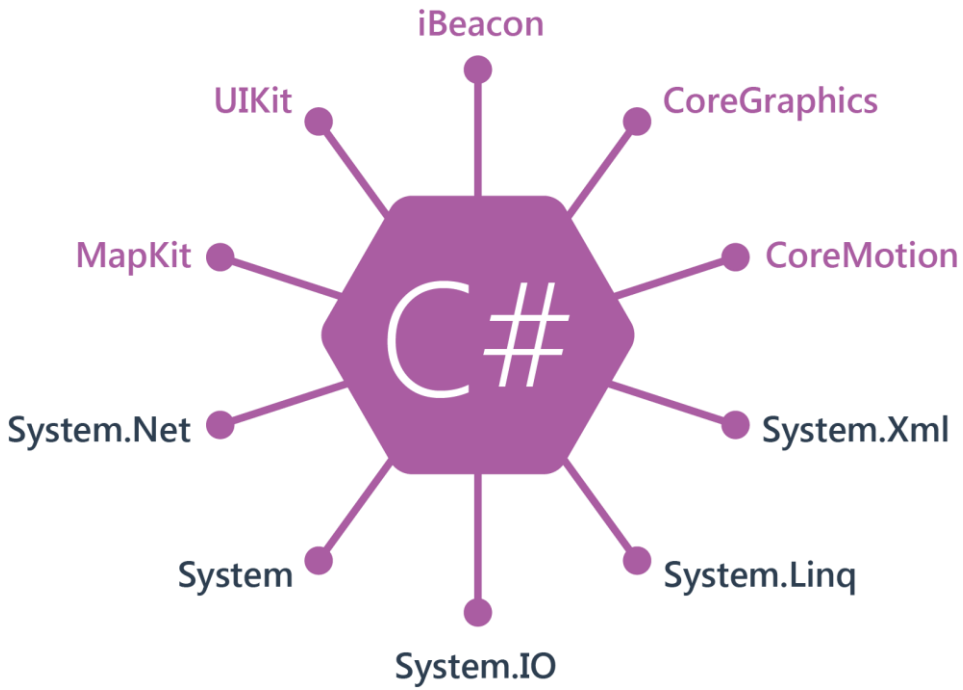
Nativo com compartilhamento de código



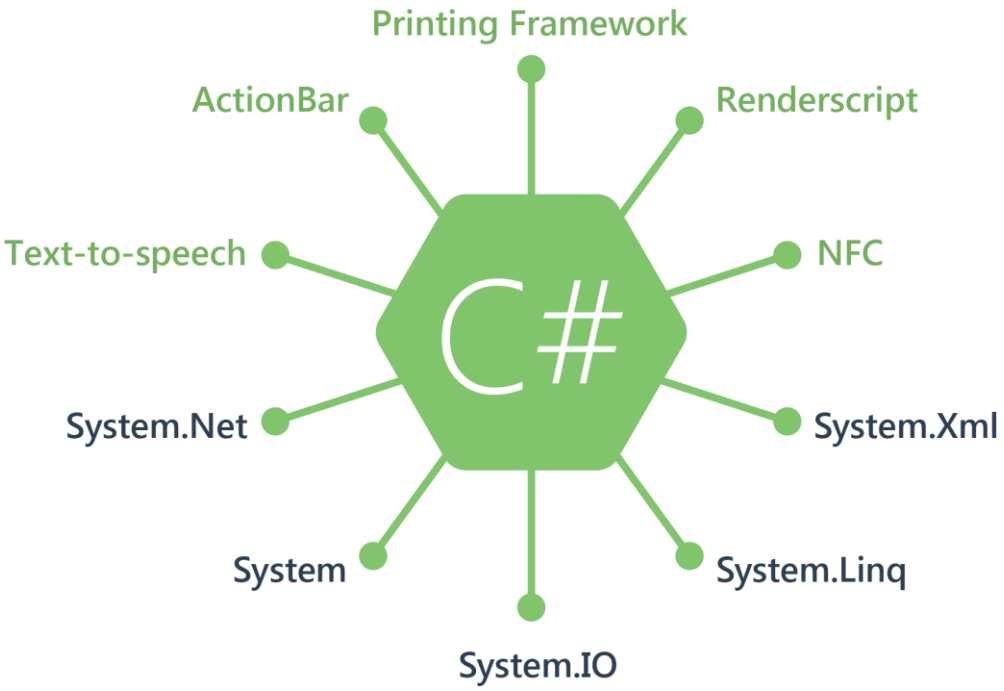
Windows APIs



iOS APIs com 100% de cobertura



Android APIs com 100% de cobertura



Multiplataforma

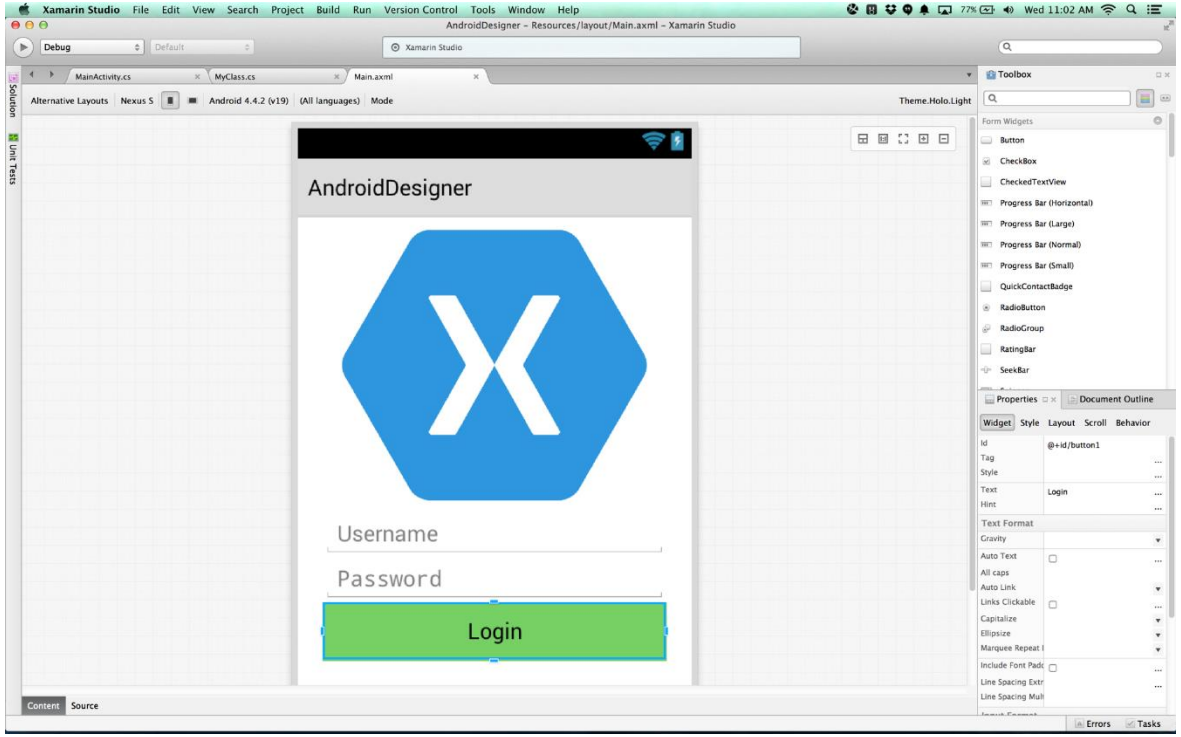
- Qualquer coisa que você pode fazer em Objective-C, Swift ou Java pode ser feito em C # com Xamarin usando Visual Studio ou Xamarin Studio em um Mac

Xamarin Designer for Android

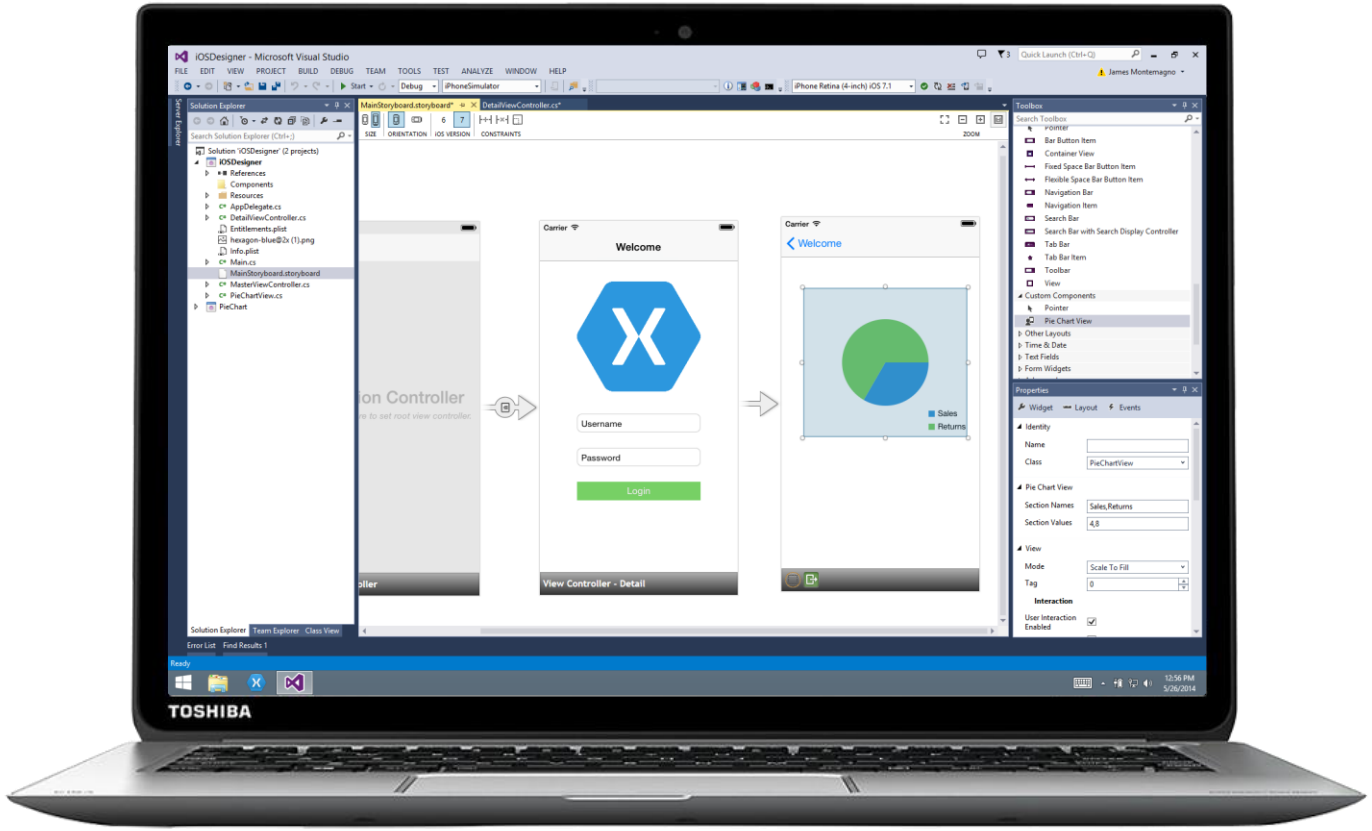
Totalmente integrado no Xamarin Studio & Visual Studio

Edição multiresolução

Fácil alternância entre design e Android XML

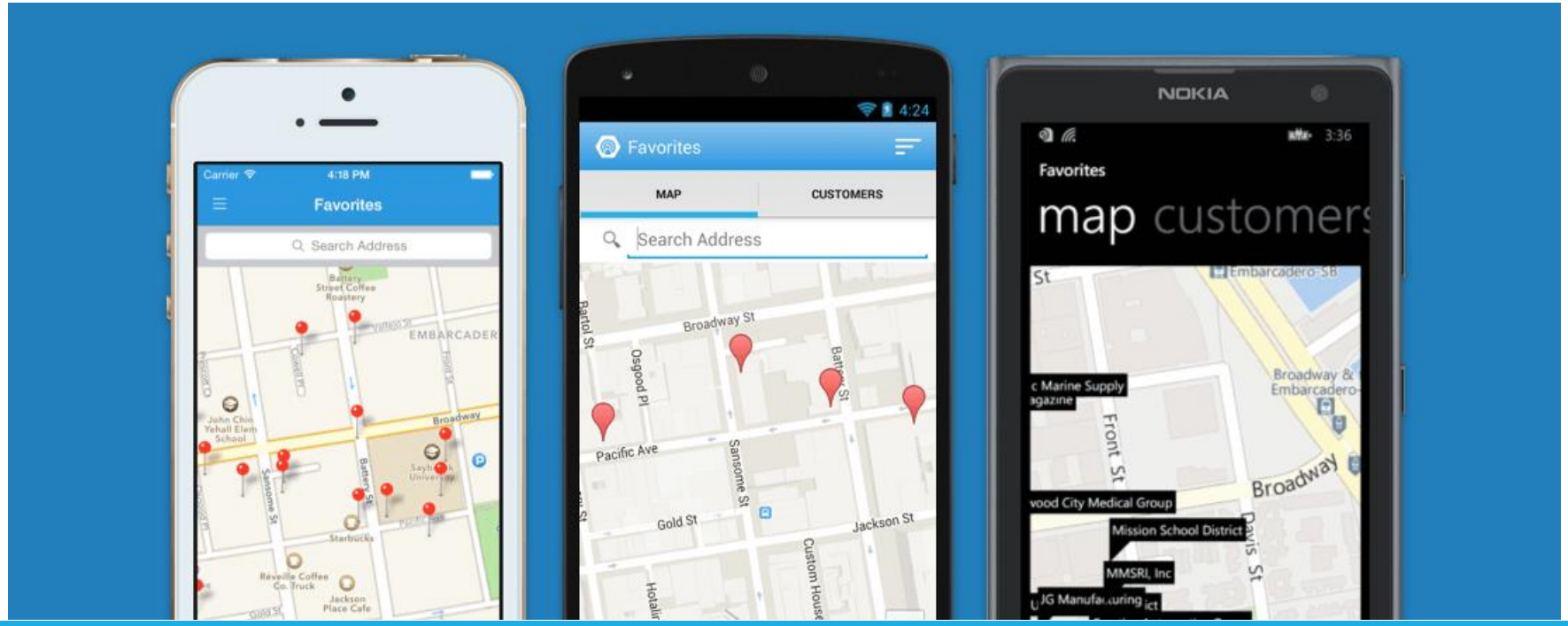


Xamarin Design para iOS



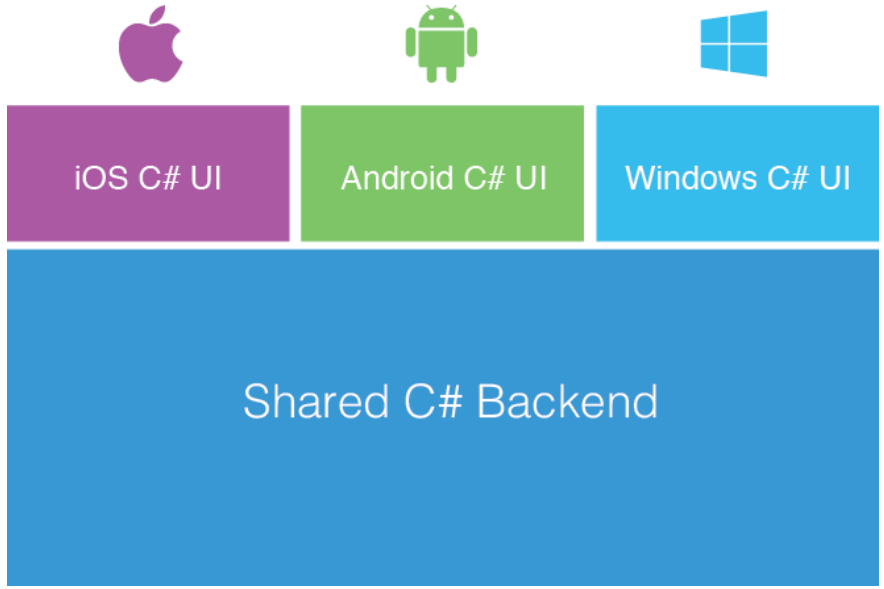
Xamarin.Forms

Crie UIs nativas para iOS, Android e UWP a partir de uma única base de código C # compartilhada

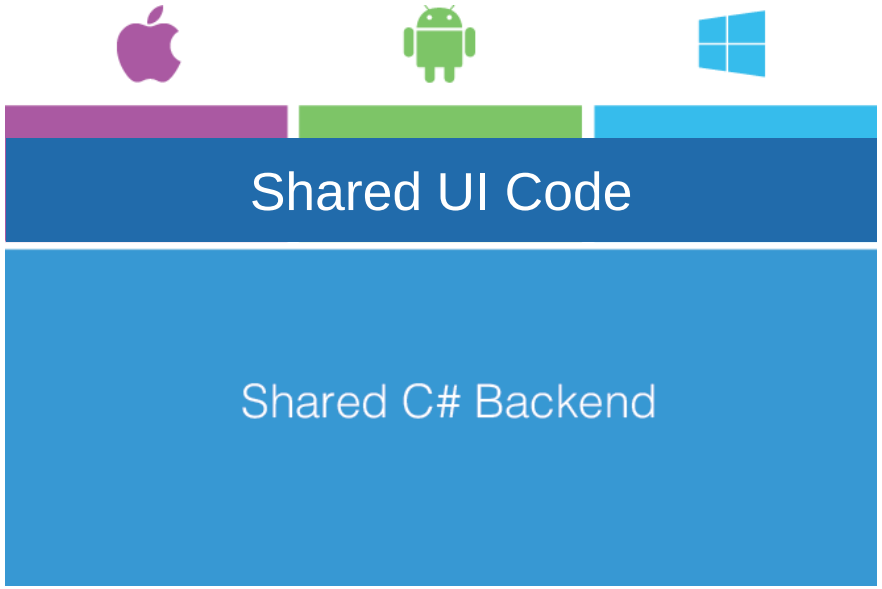


Xamarin + Xamarin.Forms

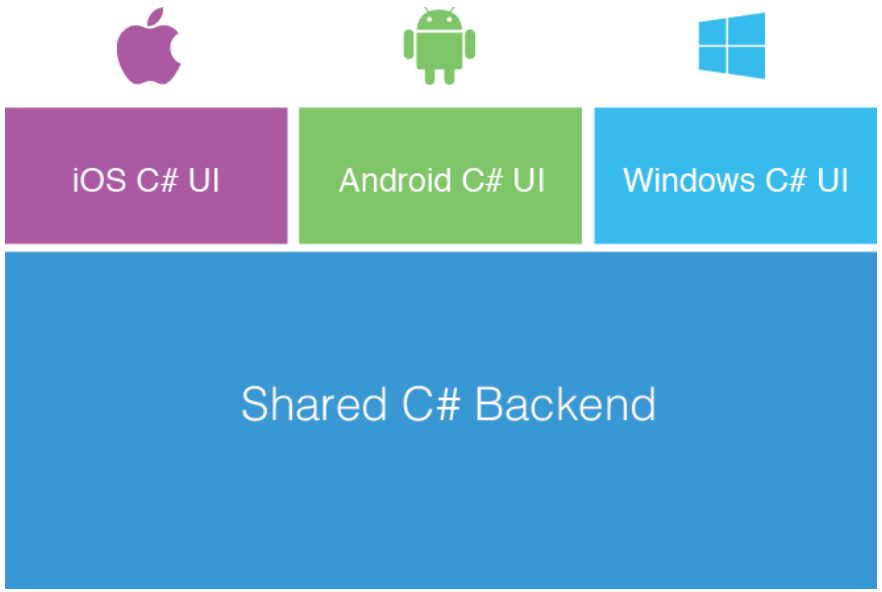
Abordagem tradicional Xamarin



Com Xamarin.Forms mais compartilhamento de código, controles nativos

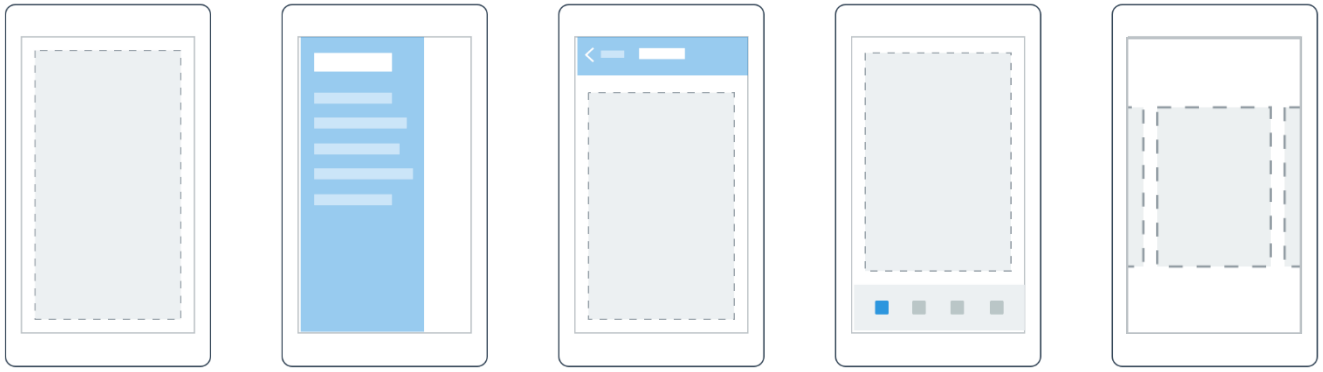


O que inclui...

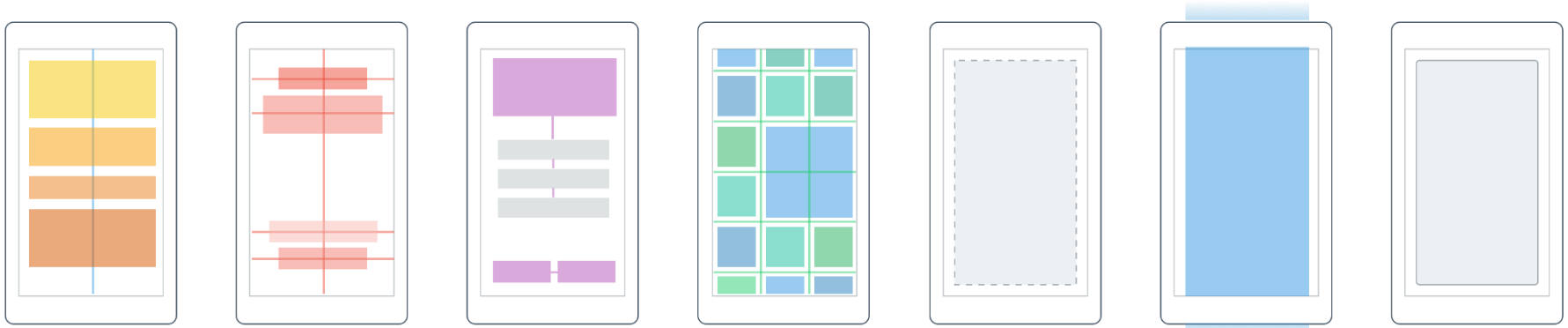


- 40+ páginas, Layouts, and controles
  - Construção via code behind ou XAML
- Two-way Data Binding
- navegação
- Animation API
- Dependency Service
- Messaging

Páginas (Pages)



Layouts


Stack
Absolute
Relative
Grid
ContentVie
w
ScrollView
Frame

Controles (Controls)

ActivityIndicator	BoxView	Button	DatePicker	Editor
Entry	Image	Label	ListView	Map
OpenGLView	Picker	ProgressBar	SearchBar	Slider
Stepper	TableView	TimePicker	WebView	EntryCell
ImageCell	SwitchCell	TextCell	ViewCell	

MVVM - Overview

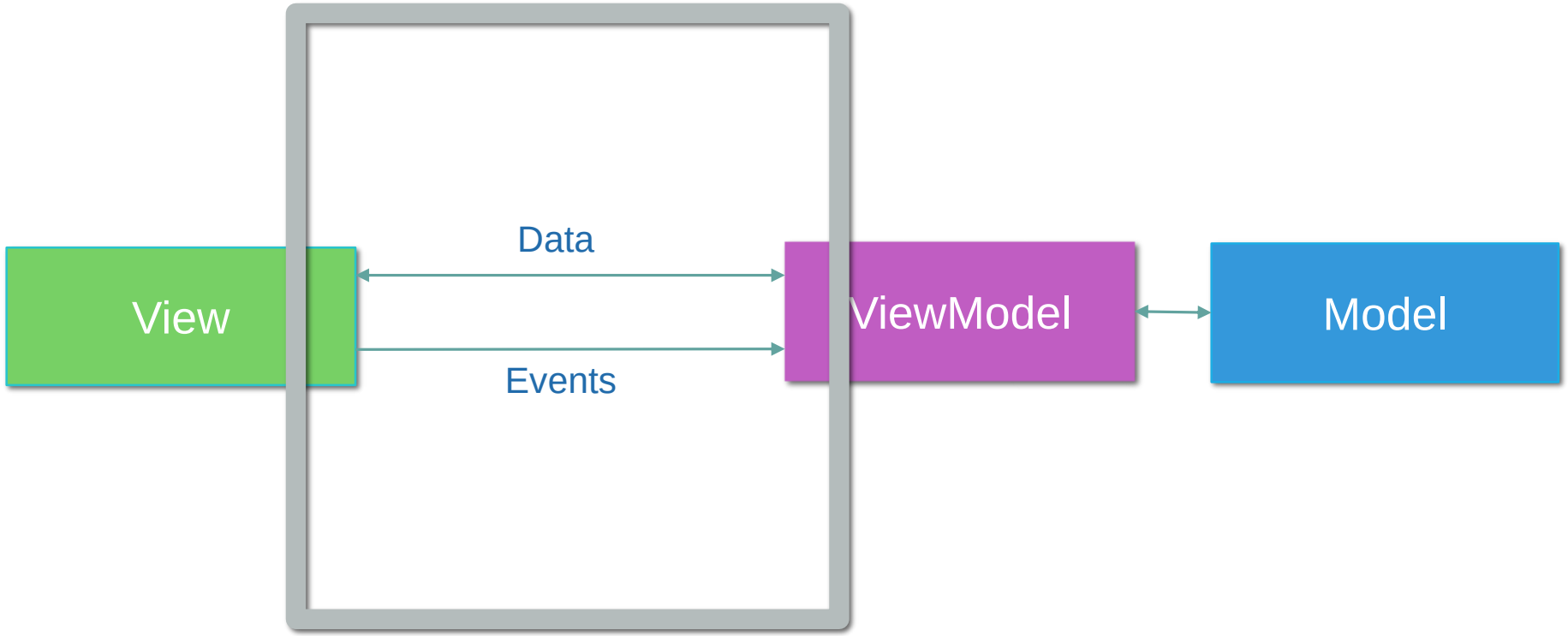
Model-View-ViewModel

Como exibir a informação

O que deve ser exibido e o fluxo de interações

Lógica de negócio / Objetos de dados

Model-View-ViewModel



Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

```
Label firstName = new Label ();
firstName.SetBinding (Label.TextProperty, "FirstName");
```

Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

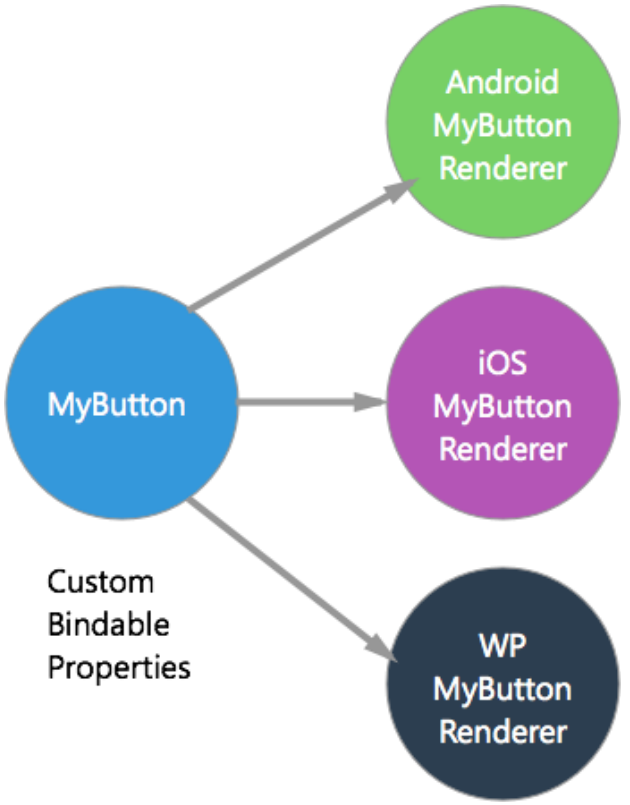
```
Label firstName = new Label ();
firstName.SetBinding (Label.TextProperty, "FirstName");
```

```
Entry firstEntry = new Entry ();
firstEntry.SetBinding<UserViewModel> (Entry.TextProperty, vm => vm.FirstName, BindingMode.TwoWay);
```

Extensibilidade

Custom Renderer

- Qualquer controle pode ser facilmente personalizado
- Adicione ou crie seus próprios controles
- Adicione Bindable Properties personalizadas



Ver mais em:
http://developer.xamarin.com/guides/cross-platform/xamarin-forms/custom-renderer/

Extensibilidade

- Dependency Service
  - Permite recursos de SDK específicos da plataforma de acesso ao código compartilhado por meio de uma implementação de interface

Extensibilidade

1.Interface

```
public interface ITextToSpeech
{
    void Speak (string text);
}
```

2.Implemente para cada plataforma

```
public class TextToSpeech_iOS : ITextToSpeech
{
    public TextToSpeech_iOS () {}

    public void Speak (string text)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (text) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

3. Registre com Dependency Service

```
[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeech_iOS))]
```

4. Acesso via código compartilhado (shared code)

```
speak.Clicked += (sender, e) => {
    DependencyService.Get<ITextToSpeech>().Speak("Hello from Xamarin Forms");
};
```

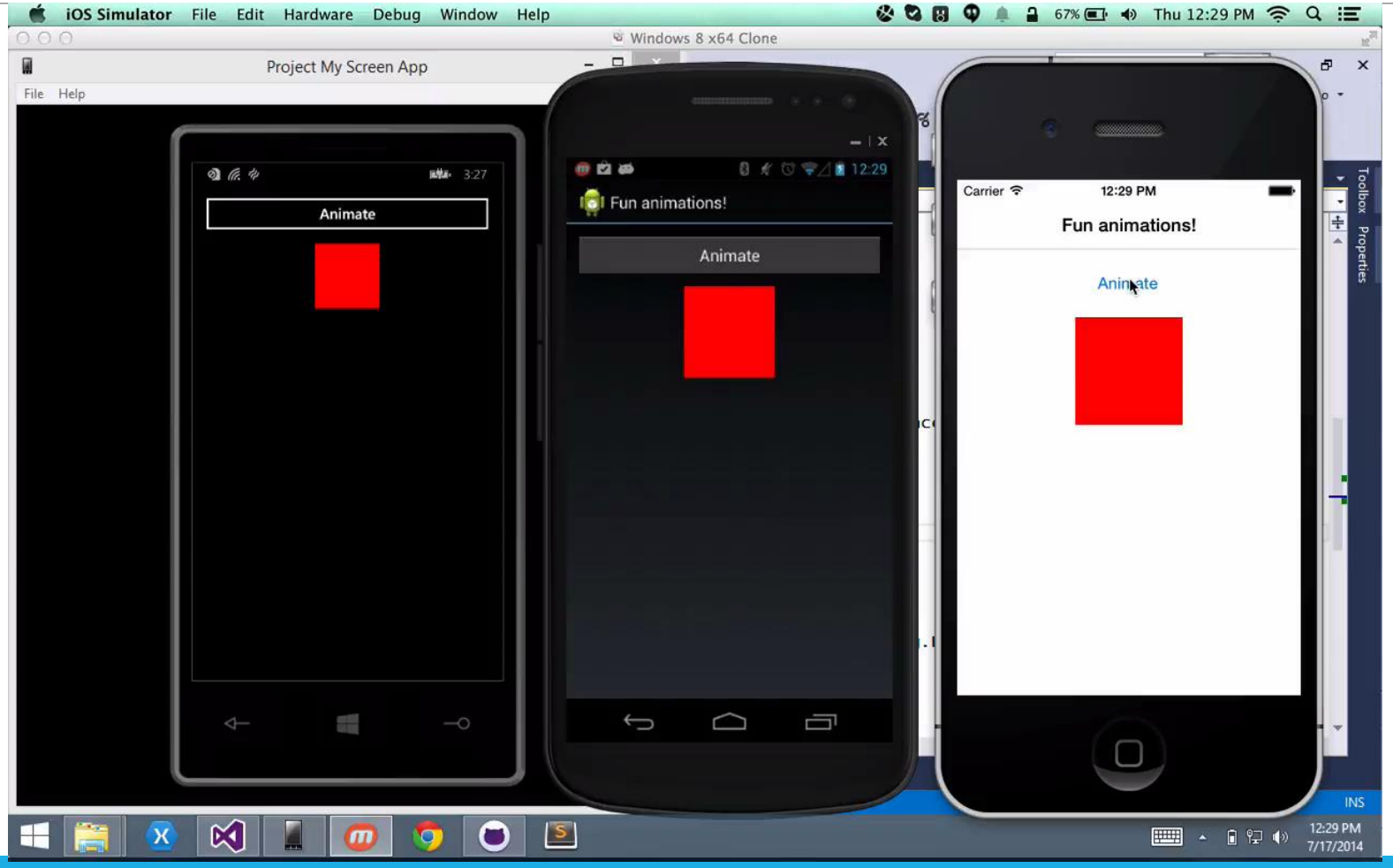
Navigation (Navegação)

- Root Page:
  - NavigationPage – Gives each page an INavigation

  - Standard Navigation
    - Navigation.PushAsync(page: nextPage);
    - Navigation.PopAsync();

  - Modal Navigation
    - Navigation.PushModalAsync(page: modalPage);
    - Navigation.PopModalAsync();

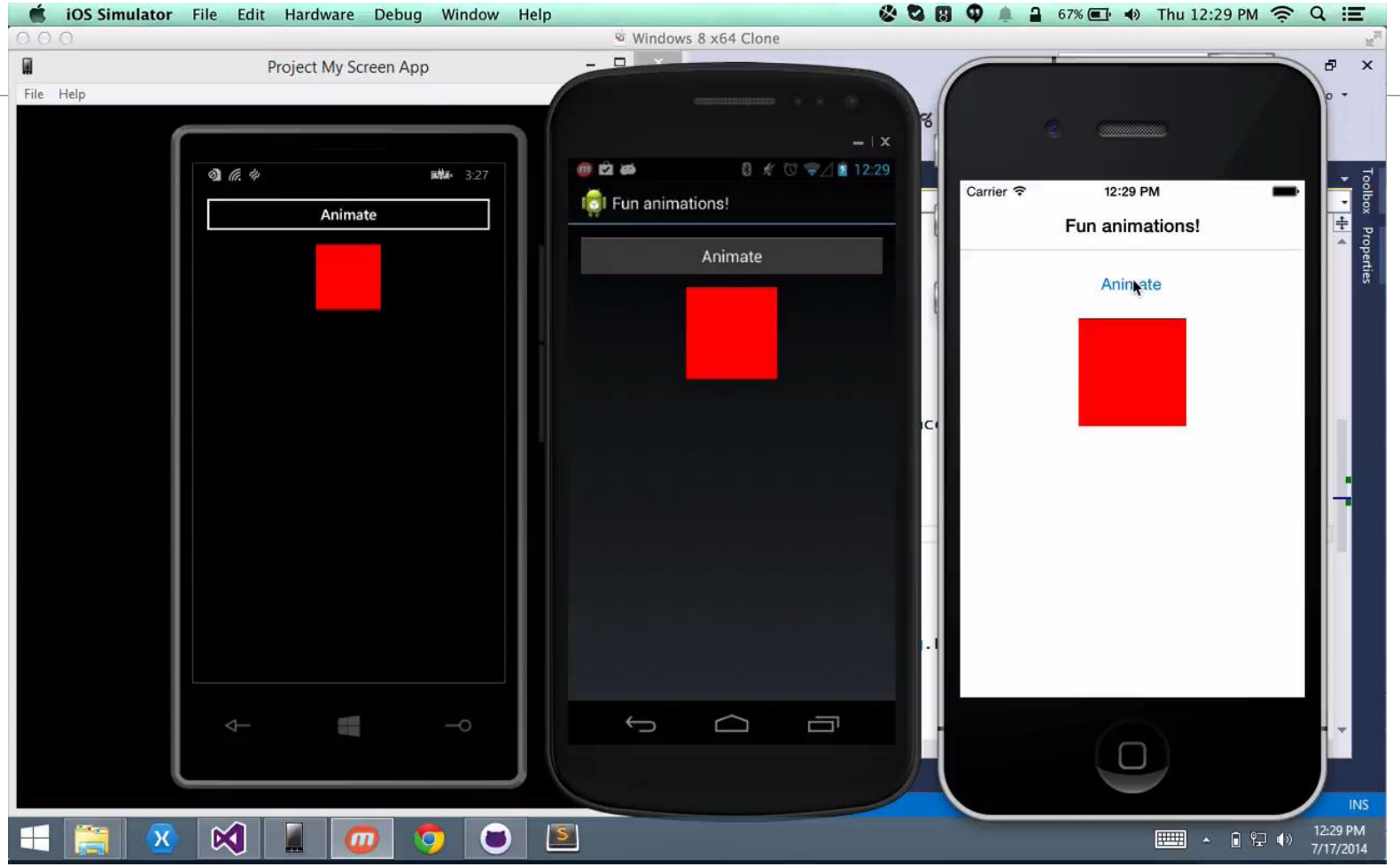
Animations API



```
button.Clicked += async (sender, args) =>
{
    button.IsEnabled = false;

    box.Color = Color.Green;

    var originalPosition = box.Bounds;
    var newPosition = box.Bounds;
    newPosition.Y = contentPage.Height - box.Height;

    await box.LayoutTo(newPosition, 2000, Easing.BounceOut);

    box.FadeTo(0, 2000);
    box.Color = Color.Yellow;

    await box.ScaleTo(2, 2000);

    box.FadeTo(1, 2000);
    await box.ScaleTo(1, 2000);

    box.Color = Color.Green;

    await box.LayoutTo(originalPosition, 2000, Easing.Linear);

    box.Color = Color.Red;
    button.IsEnabled = true;

};
```

```
button.Clicked += async (sender, args) =>
{
    button.IsEnabled = false;

    box.Color = Color.Green;

    var originalPosition = box.Bounds;
    var newPosition = box.Bounds;
    newPosition.Y = contentPage.Height - box.Height;

    await box.LayoutTo(newPosition, 2000, Easing.BounceOut);

    box.FadeTo(0, 2000);
    box.Color = Color.Yellow;

    await box.ScaleTo(2, 2000);

    box.FadeTo(1, 2000);
    await box.ScaleTo(1, 2000);

    box.Color = Color.Green;

    await box.LayoutTo(originalPosition, 2000, Easing.Linear);

    box.Color = Color.Red;
    button.IsEnabled = true;

};
```

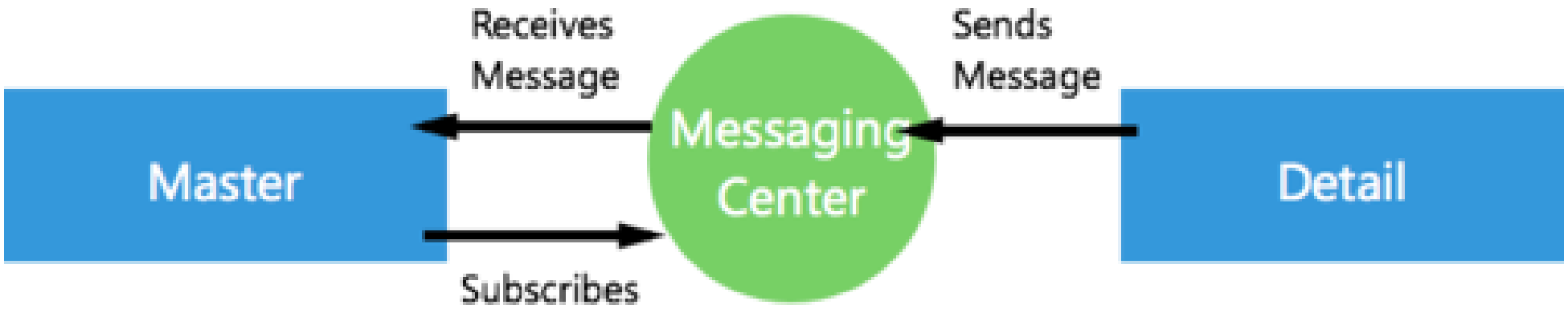
```
button.Clicked += async (sender, args) =>
{
    button.IsEnabled = false;

    box.Color = Color.Green;

    var originalPosition = box.Bounds;
    var newPosition = box.Bounds;
    newPosition.Y = contentPage.Height - box.Height;

    await box.LayoutTo(newPosition, 2000, Easing.BounceOut);

    box.FadeTo(0, 2000);
    box.Color = Color.Yellow;

    await box.ScaleTo(2, 2000);

    box.FadeTo(1, 2000);
    await box.ScaleTo(1, 2000);

    box.Color = Color.Green;

    await box.LayoutTo(originalPosition, 2000, Easing.Linear);

    box.Color = Color.Red;
    button.IsEnabled = true;

};
```


iOS Simulator File Edit Hardware Debug Window Help
67% Thu 12:29 PM
Windows 8 x64 Clone
Project My Screen App
File Help
3:27
Animate
12:29
Fun animations!
Animate
Carrier
12:29 PM
Fun animations!
Animate
Toolbox
Properties
INS
12:29 PM
7/17/2014

Messaging Center

- MessagingCenter.Subscribe<T>(object subscriber, string message, Action<T> callback);

- MessagingCenter.Send(T item, string message);



Messaging Center

Master Page

```
//Subscibe to insert expenses
MessagingCenter.Subscribe<TripExpense>(this, "AddNew", (expense) =>
{
  Expenses.Add(expense);
});
```

Detail Page:

```
MessagingCenter.Send(expense, "AddNew");
```

REFERÊNCIAS

- Documentação
  - http://developer.xamarin.com/guides/cross-platform/xamarin-forms/
- Documentação XAML
  - http://developer.xamarin.com/guides/cross-platform/xamarin-forms/xaml-for-xamarin-forms/
- Exemplos
  - https://github.com/xamarin/xamarin-forms-samples