Macoratti.net

Xamarin Android - Consumindo REST com HttpClient

Neste artigo vou mostrar como consumir serviços REST usando a classe **HttpClient** no Xamarin Android com o Visual Studio 2015 e a linguagem C#.

Vamos iniciar com resumo sobre REST...

O que é REST ?

Representational State Transfer (REST) é um estilo de arquitetura para a construção de serviços na web. As solicitações REST são feitas em HTTP usando os mesmos verbos HTTP que os navegadores da Web usam para recuperar páginas da Web e enviar dados para os servidores. Os verbos são:

GET - esta operação é usada para recuperar dados do serviço da web.
POST - esta operação é usada para criar um novo item de dados no serviço web.
PUT - esta operação é usada para atualizar um item de dados no serviço da web.
PATCH - esta operação é usada para atualizar um item de dados no serviço da Web, descrevendo um conjunto de instruções sobre como o item deve ser modificado.
DELETE - esta operação é usada para apagar um item de dados no serviço web.

As APIs de serviço da Web que aderem ao REST são chamadas de **APIs RESTful** e são definidas usando:

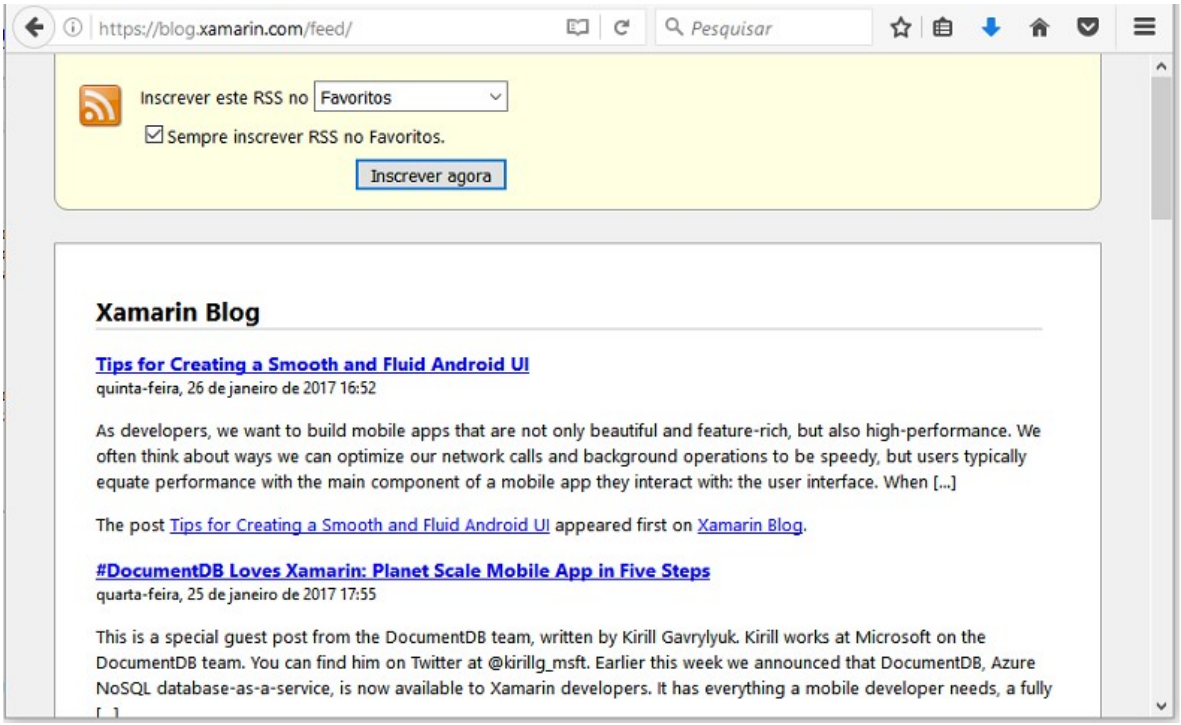
1- Uma URI base.
2- Métodos HTTP, como GET, POST, PUT, PATCH ou DELETE.
3- Um tipo de mídia para os dados, como JavaScript Object Notation (JSON).

A simplicidade do REST ajudou a torná-lo o principal método para acessar serviços da Web em aplicativos móveis.

Para mostrar como podemos usar consimir um serviço REST vou criar um exemplo bem simples que acessa serviços REST disponibilizados nos seguintes endereços:

- **http://jsonplaceholder.typicode.com/posts** - retorna dados no formato Json;

- **https://blog.xamarin.com/feed/** - retorna dados no formato XML;



Além de enviar requisições para obter informações (GET) vamos também postar informações usando o POST. Vamos realizar essas tarefas usando a classe **HttpClient**.

A classe **HttpClient** é a classe principal usada para enviar e receber mensagens HTTP através de **HttpRequestMessage** e **HttpResponseMessage.** Se você já utilizou anteriormente as classes WebClient e HttpWebRequest vai notar que a classe HttpClient possui diferenças importantes a saber:

1. Uma instância HttpClient é usada para configurar extensões, definir *headers* padrão, cancelar *requests* e mais;
2. Você pode emitir tantos pedidos quantos quiser através de uma única instância HttpClient;
3. HttpClients não estão vinculados a um determinado servidor HTTP ou host; você pode enviar qualquer solicitação HTTP usando a mesma instância HttpClient;
4. Você pode derivar de HttpClient para criar clientes especializados para determinados sites ou padrões;
5. A classe HttpClient usa o novo padrão orientada a tarefa (Task) para lidar com solicitações assíncronas, possibilitando assim, gerenciar e coordenar de forma mais fácil solicitações pendentes;

Recursos usados:

- **Visual Studio Community 2015 ou Xamarin Studio**
- **Xamarin**
- **RestSharp - http://restsharp.org/**
- **Emulador Android virtual ou físico (veja como emular usando o Vysor)**

Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no VS Community 2015

Abra o VS 2015 Community e clique em **New Project;**

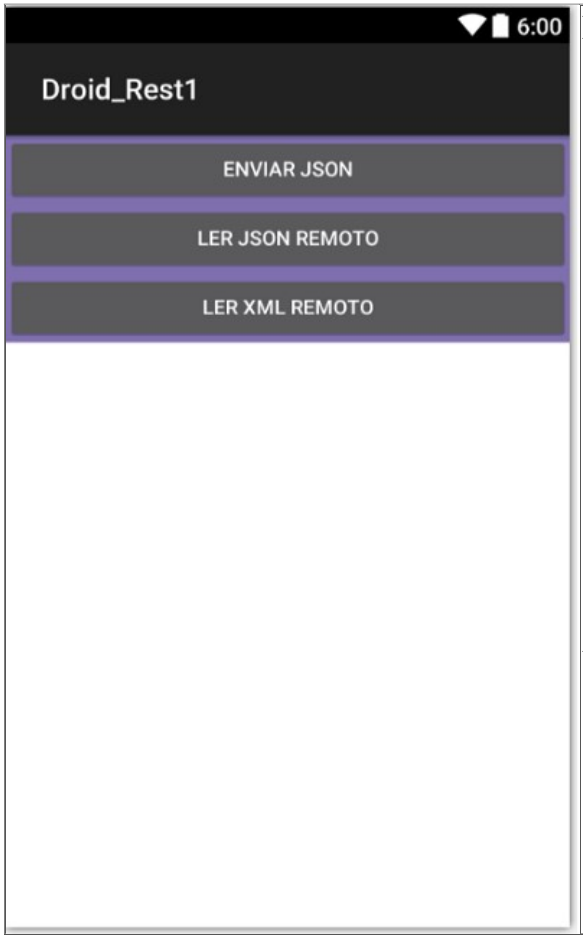
Selecione a linguagem Visual C# e o template **Android -> Blank App(Android)**

Informe o nome um nome adequado ao seu projeto, eu vou usar o nome **Droid_Rest1**, e clique no botão **OK**;

Abra o arquivo **Main.axml** na pasta **Resources/layout** no modo **Designer** e inclua a partir da ToolBox os seguintes controles:

- **3 Buttons - enviarDados, lerJson e lerXml**
- **1 TextView - textView**

Abaixo vemos o leiaute no emulador do Xamarin e ao lado o respectivo código XML gerado :

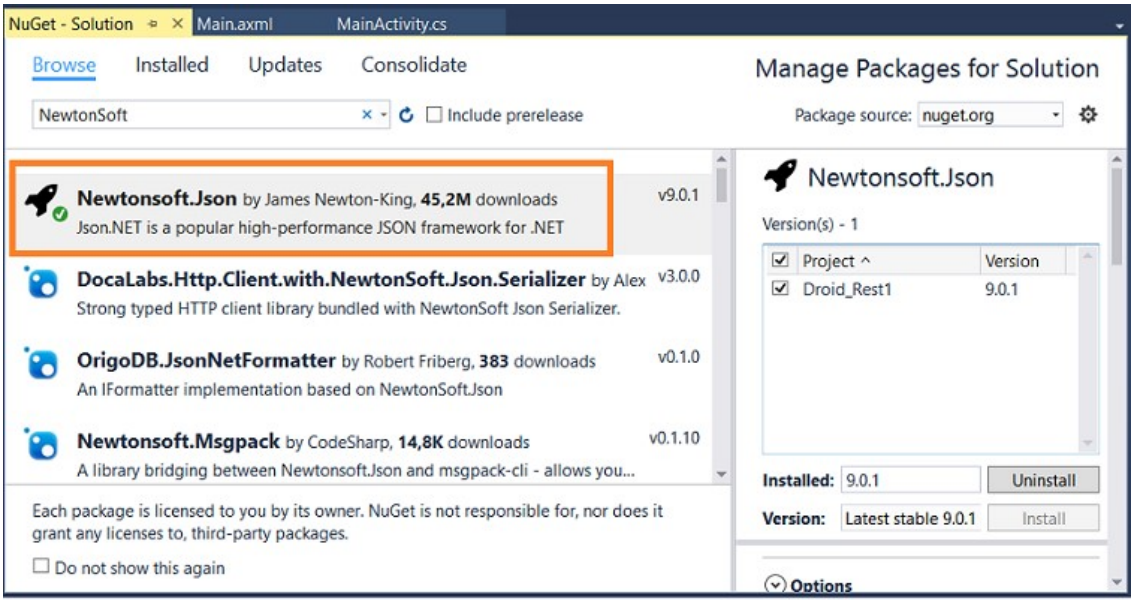



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:background="#7f6faf"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:text="@string/sendData"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/enviarDados" />
    <Button
        android:text="@string/readJson"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/lerJson" />
    <Button
        android:text="@string/readXml"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/lerXml" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:background="@android:color/white"
        android:textColor="@android:color/black"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        android:layout_weight="1" />
</LinearLayout>
```

Vamos agora incluir as referências às bibliotecas **System.Net.Http , Newntonsoft.Json** que iremos usar para acessar o serviço REST e tratar os dados no formato Json.

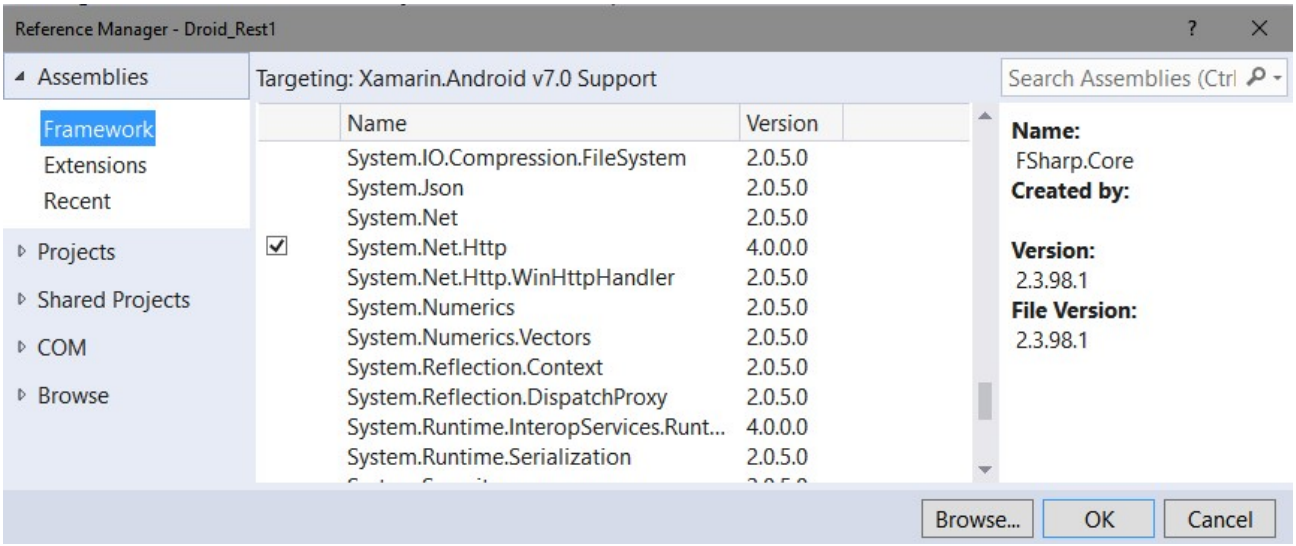
No menu **Tools** clique em **Nuget Package Manager** e a seguir em Manage Nuget Packages for Solution;

Selecione a guia **Browse** e digite **NewTonSoft**. Localize a biblioteca e instale-a no projeto:



A seguir clique com o botão direito do mouse sobre o projeto e a seguir em **Add References;**

Selecione o item **Assemblies** e marque o item **System.Net.Http** e clique em OK;



Criando as classes Post e Rss

Vamos criar as classes **Post** e **Rss** definindo os dados conforme a estrutura que iremos receber via Json e XML.

No menu **Project** clique em **Add Class** e informe o nome **Post**. A seguir digite o código abaixo nesta classe:

```
using Newtonsoft.Json;

namespace Droid_Rest1
{
    public class Post
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public string Title { get; set; }

        [JsonProperty("body")]
        public string Content { get; set; }

        public override string ToString()
```

```
        {
            return string.Format(
                "Post Id: {0}\nTitle: {1}\nBody: {2}",
                Id, Title, Content);
        }
    }
}
```

Repita o procedimento e informe o código a seguir para classe **Rss**:

```
using System.Collections.Generic;
using System.Net;
using System.Xml;
using System.Xml.Serialization;

namespace Droid_Rest1
{
    [XmlRoot(ElementName = "rss")]
    public class Rss
    {
        [XmlElement("channel")]
        public Channel Channel { get; set; }
    }

    [XmlRootAttribute(ElementName = "channel")]
    public class Channel
    {
        [XmlElement("title")]
        public string Title { get; set; }

        [XmlElement(ElementName = "item", Type = typeof(Item))]
        public List<Item> Items { get; set; }
    }

    [XmlRootAttribute(ElementName = "item")]
    public class Item
    {
        [XmlElement("guid")]
        public string Guid { get; set; }

        [XmlElement("title")]
        public string Title { get; set; }

        [XmlElement("description")]
        public XmlCDataSection DescriptionCData { get; set; }

        public string Content
        {
            get { return WebUtility.HtmlDecode(DescriptionCData.Data); }
        }

        public override string ToString()
        {
            return string.Format(
                "Post Url: {0}\nTitle: {1}\nBody: {2}",
                Guid, Title, Content);
        }
    }
```

```
}
```

Definindo o código na MainActivity

Abra o arquivo **MainActivity** na raiz do projeto e incluir o código abaixo neste arquivo.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Xml.Serialization;
using Android.App;
using Android.Widget;
using Android.OS;
using Newtonsoft.Json;

namespace Droid_Rest1
{
    [Activity(Label = "Consumindo Serviços REST", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            SetContentView (Resource.Layout.Main);
            Button lerJson = FindViewById<Button>(Resource.Id.lerJson);
            Button lerXml = FindViewById<Button>(Resource.Id.lerXml);
            Button enviarDados = FindViewById<Button>(Resource.Id.enviarDados);
            TextView textView = FindViewById<TextView>(Resource.Id.textView);

            lerJson.Click += async delegate
            {
                using (var client = new HttpClient())
                {
                    // envia a requisição GET
                    var uri = "http://jsonplaceholder.typicode.com/posts";
                    var result = await client.GetStringAsync(uri);

                    // processa a resposta
                    var posts = JsonConvert.DeserializeObject<List<Post>>(result);

                    // gera a saida
                    var post = posts.First();
                    textView.Text = "Primeiro post:\n\n" + post;
                }
            };

            lerXml.Click += async delegate
            {
                using (var client = new HttpClient())
                {
                    // envia uma requição GET
                    var uri = "http://blog.xamarin.com/feed";
```

```
            var result = await client.GetStreamAsync(uri);

            // processa a resposta
            var serializer = new XmlSerializer(typeof(Rss));
            var feed = (Rss)serializer.Deserialize(result);

            // gera a saida
            var item = feed.Channel.Items.First();
            textView.Text = "Primeiro Item:\n\n" + item;
        }
    };

    enviarDados.Click += async delegate
    {
        using (var client = new HttpClient())
        {
            // cria um novo post
            var novoPost = new Post
            {
                UserId = 12,
                Title = "Meu primeiro Post",
                Content = "Macoratti .net - Quase tudo para .NET!"
            };

            // cria o conteudo da requisição e define o tipo Json
            var json = JsonConvert.SerializeObject(novoPost);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            // envia a requisição POST
            var uri = "http://jsonplaceholder.typicode.com/posts";
            var result = await client.PostAsync(uri, content);

            // Se ocorrer um erro lança uma exceção
            result.EnsureSuccessStatusCode();

            // processa a resposta
            var resultString = await result.Content.ReadAsStringAsync();
            var post = JsonConvert.DeserializeObject<Post>(resultString);

            // exibe a saida no TextView
            textView.Text = post.ToString();
        }
    };

  }
 }
}
```

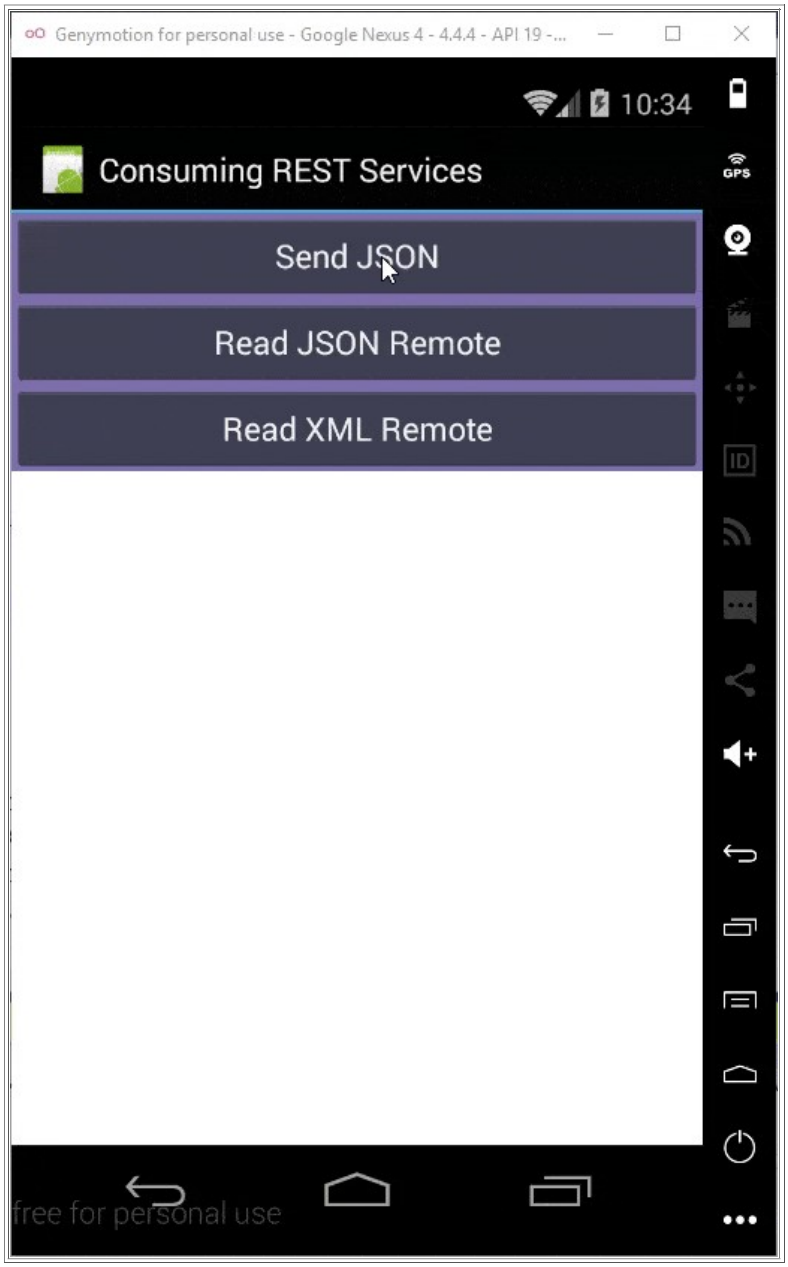
Vamos entender o código :

Estamos usando os seguintes métodos da classe **HttpClient** :

- **GetStringAsync - Usando para enviar uma requisição GET;**
- **GetStreamAsync - Usando para enviar uma requisição GET;**
- **PostAsync - Usando para enviar uma requisição POST;**

Depois deserializamos os dados recebidos usando o método **DeserializeObject** da classe **Newtonsoft** para os dados Json e do método **Deserialize** para o XML, e, exibimos o resultado no TextView.

Executando o projeto usando o emulador **Genymotion** iremos obter o seguinte resultado:



Pegue o projeto aqui : **Droid_Rest1.zip** (sem as referências)

Respondeu Jesus, e disse-lhes: Ainda que eu testifico de mim mesmo, o meu testemunho é verdadeiro, porque sei de onde vim, e para onde vou; mas vós não sabeis de onde venho, nem para onde vou.
Vós julgais segundo a carne; eu a ninguém julgo.
E, se na verdade julgo, o meu juízo é verdadeiro, porque não sou eu só, mas eu e o Pai que me enviou.
João 8:14-16

- **Curso Básico VB .NET - Vídeo Aulas**

Quer aprender C# ??

- Chegou o **Super DVD C#** com exclusivo material de suporte e vídeo aulas com curso básico sobre C#.
- **Curso C# Basico - Video Aulas**

Quer aprender os conceitos da Programação Orientada a objetos ?

- **Curso Fundamentos da Programação Orientada a Objetos com VB .NET** NEW

Quer aprender o gerar relatórios com o ReportViewer no VS 2013 ?

- **Curso - Gerando Relatórios com o ReportViewer no VS 2013 - Vídeo Aulas** NEW

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET
- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- VB.NET 2005 - Controles - Macoratti.net
- Seção de Jogos do site Macoratti .net
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net
- https://developer.xamarin.com/api/type/Android.App.AlertDialog/
- Xamarin Android - Tratando eventos de forma declarativa
- Seção Mobile/Xamarin do site Macoratti .net
- https://developer.android.com/reference/android/app/Activity.html
- https://developer.xamarin.com/api/type/Android.Widget.ProgressBar/

José Carlos Macoratti