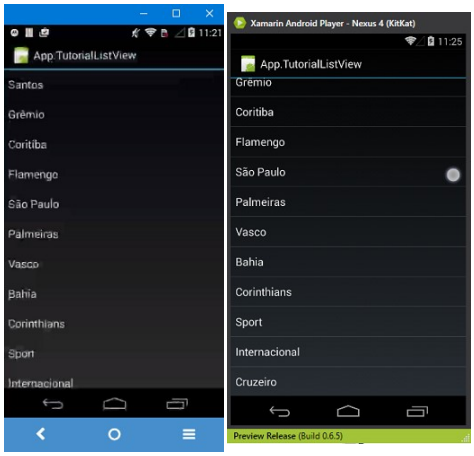
Macoratti.net Xamarin Android - Apresentando e usando o componente ListView

Neste artigo vou apresentar os conceitos básicos sobre o **ListView** e sua utilização em aplicações **Android** usando o Visual Studio com Xamarin e a linguagem C#.

Um **ListView** nada mais é que uma lista de itens, certo ?

Um **ListView** no Android é uma **view** que agrupa diversos itens e os exibe em uma lista vertical rolável. Os itens da lista são automaticamente inseridos na lista usando um **Adpater** que obtém o conteúdo de uma fonte de dados como um *array, um banco de dados*, etc.

Conforme mostra a figura abaixo: *(1-tela do meu motorola emulada no Vysor e 2-tela do emulador Android virtual KitKat)*



Um **ListView** consiste nas seguintes partes:

Rows - A representação visível dos dados na lista.
Adapter - Uma classe não visual que vincula a fonte de dados para a exibição da lista.
Fast Scrolling - Um identificador que permite ao usuário rolar o comprimento da lista.
Section Index - Um elemento de interface de usuário que flutua sobre as linhas de rolagem para indicar onde na lista as linhas atuais estão localizadas.

Um **Adapter (adaptador)** serve como ponte entre os componentes de interface do usuário (UI) e a fonte de dados que preenche os dados nestes componentes. Ele trata e envia os dados para a respectiva view que pode obter esses dados exibindo-os a seguir em diferentes tipos de views como **spinner, list view, grid view, etc.**

As subclasses **ListView e GridView** são subclasses de **AdapterView,** e, elas podem ser preenchidas pela vinculação com um **Adapter**, que recupera dados de uma fonte externa e cria uma **view** que representa cada entrada de dados.

O Xamarin Android fornece várias subclasses de adaptador que são úteis para recuperar diferentes tipos de dados e **views** para uma **AdapterView** *(isto é ListView ou GridView).*

Os adaptadores mais comuns são ArrayAdapter, BaseAdapter, CursorAdapter, SimpleCursorAdapter, SpinnerAdapter e WrapperListAdapter.

Vamos ver na prática como exibir uma lista de itens de uma fonte de dados usando o **ListView** em uma aplicação Android criada no Visual Studio com Xamarin.

Recursos usados:

- **Visual Studio Community 2015 ou Xamarin Studio**
- **Xamarin**
- **Emulador Android virtual ou físico (veja como emular usando o Vysor)**

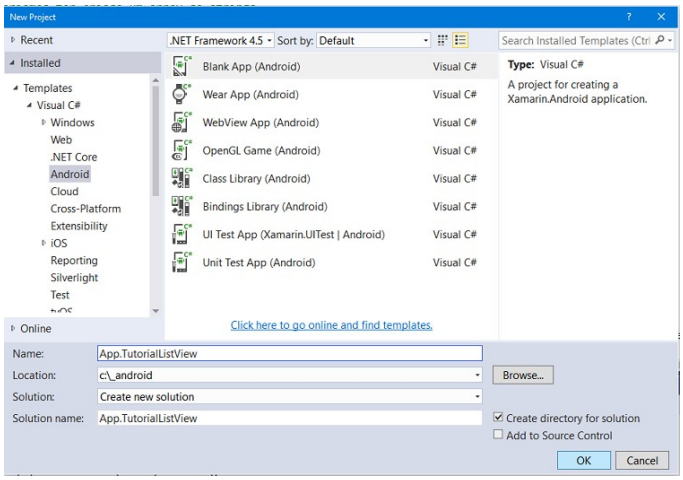
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no Visual Studio 2015 Community

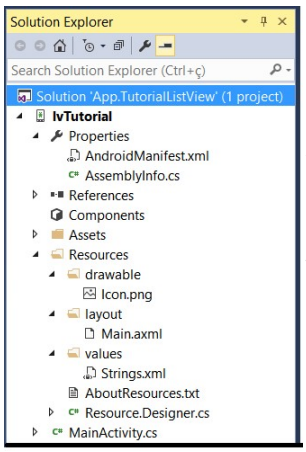
Abra o VS 2015 Community e clique em **New Project;**

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

Informe o nome **App.TutorialListView** e clique no botão **OK**;



Será criada uma solução com a seguinte estrutura:



- **Properties** - Contém o arquivo AndroidManifest.xml que descreve as funcionalidades e requisitos da sua aplicação Android, e o arquivo AssemblyInfo.cs contém informação sobre o projeto como número de versão e build.

- **References** - Contém as bibliotecas Mono.Android, System.Core e todas as bibliotecas usadas no seu projeto;

- **Components** - Contém componentes de terceiros ou desenvolvidos por você usados no seu projeto.

A maioria dos componentes está disponíveis diretamente do **Xamarin Component Store** e são **free** *(não todos)* e prontos para serem usados; (Para incluir um componente clique com o botão direito sobre **Components** e a seguir em Get More Components);

- **Assets e Resources** - Contém arquivos que não são código, como imagens, sons, arquivos XML e qualquer outro recurso que sua aplicação for usar. Os arquivos externos colocados na pasta **Assets** são facilmente acessíveis em tempo de execução através do Asset Manager.

Já os arquivos colocados na pasta **Resources** precisam ser declarados e mantidos em uma lista com os IDs dos recursos que você desejar usar em tempo de execução.

De forma geral, todas a imagens, ícones, sons e outros arquivos externos são colocados na pasta **Resources** enquanto que dicionários e arquivos XML são postos na pasta **Assets**;

Na subpasta **layout** temos os arquivos .axml que definem as views usadas no projeto;

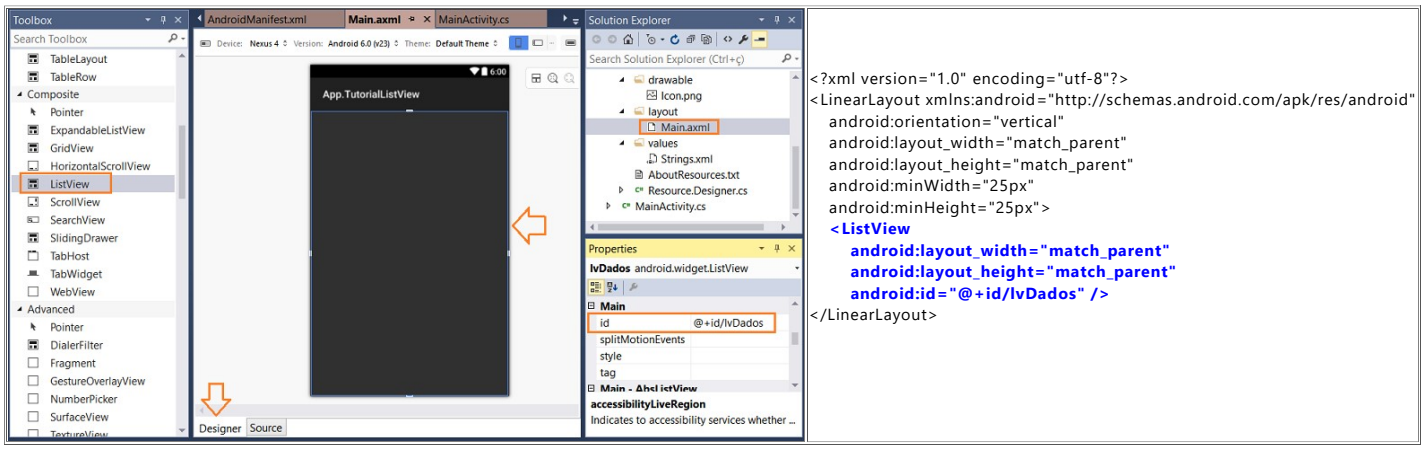
Na subpasta **values** temos o arquivo Strings.xml onde definimos as strings usadas no projeto;

O arquivo **MainActivity.cs** é um arquivo C# que define a atividade principal da aplicação Android.

1- Vamos Abrir o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer** incluir um novo controle **ListView** a partir da **ToolBox** e definir a seguintes propriedades:

- **id = @+id/lvDados**

Abaixo vemos o leiaute no emulador do Xamarin exibindo a tela 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:layout_width="match_parent"
  android:layout_height="match_parent"
  android:minWidth="25px"
  android:minHeight="25px">
  <ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/lvDados" />
</LinearLayout>
```

Agora podemos iniciar a implementação do código para exibir itens no ListView no arquivo **MainActivity.cs**.

Veja como deve ficar o código do arquivo **MainActivity.cs**: (*O código em azul foi o que incluímos)*

```
using Android.App;
using Android.OS;
using Android.Widget;
using System.Collections.Generic;

namespace App.TutorialListView
{
  [Activity(Label = "App.TutorialListView", MainLauncher = true, Icon = "@drawable/icon")]
  public class MainActivity : Activity
  {
    private List<string> times;
    private ListView listView;

    protected override void OnCreate(Bundle bundle)
    {
      base.OnCreate(bundle);

      SetContentView(Resource.Layout.Main);
      listView = FindViewById<ListView>(Resource.Id.lvDados);

      times = new List<string>();
      times.Add("Santos");
      times.Add("Grêmio");
      times.Add("Coritiba");
      times.Add("Flamengo");
      times.Add("São Paulo");
      times.Add("Palmeiras");
      times.Add("Vasco");
      times.Add("Bahia");
      times.Add("Corinthians");
      times.Add("Sport");
      times.Add("Internacional");
      times.Add("Cruzeiro");
      times.Add("Atletico");
      times.Add("Vitoria");

      ArrayAdapter<string> adapter = new ArrayAdapter<string>(this, Android.Resource.Layout.SimpleListItem1, times);
      listView.Adapter = adapter;

      listView.ItemClick += (sender, e) =>
```

```
        {
            using (var dialog = new AlertDialog.Builder(this))
            {
                int posicao = e.Position;
                string valor = times[posicao];
                dialog.SetTitle("Time Selecionado");
                dialog.SetMessage(valor);
                dialog.Show();
            }
        };
    }
  }
}
```

Entendendo o código :

Antes da definição do método **OnCreate** criamos a variável **times** como uma lista de strings e a variável **listView** do tipo da classe ListView:

```
private List<string> times;
private ListView listView;
```

A seguir fiz as seguintes implementações no método **OnCreate**() :

1- **SetContentView**(Resource.**Layout.Main**);

O arquivo **Resource.designer.cs** é um arquivo C# na pasta Resources que é gerado pelo Xamarin.Android e contém definições de ID para todos os recursos na App.

O método **SetContentView** define o conteúdo da atividade (**activity**) para uma **view** explicita.

No exemplo, estamos carregando a interface do usuário (*a view*) definida no arquivo **Main.axml** presente na pasta **Resources\layout**.

2- listView = **FindViewById**<Button>(Resource.Id.**lvDados**);

Usei o objeto listView do tipo ListView localizado pelo método **FindViewById** pelo Id **lvDados** a partir do arquivo XML que foi processado no método **OnCreate**.

3- Definição dos dados na lista de strings representando pela variável **times**, contendo o nome de times de futebol :

```
times = new List<string>();
        times.Add("Santos");
        times.Add("Grêmio");
        times.Add("Coritiba");
        times.Add("Flamengo");
        times.Add("São Paulo");
        times.Add("Palmeiras");
        times.Add("Vasco");
        times.Add("Bahia");
        times.Add("Corinthians");
        times.Add("Sport");
        times.Add("Internacional");
        times.Add("Cruzeiro");
        times.Add("Atletico");
        times.Add("Vitoria");
```

4 - Criação do **adapter** usando a classe **ArrayAdapter** que é uma classe concreta de **BaseAdapter** que é apoiada por um array de objetos.

Por padrão, o **ArrayAdapter** cria uma view para cada item do array chamando **toString()** em cada item e coloca o conteúdo em um **TextView**.

No exemplo estamos usando a lista de strings definida em **times,** e estamos usando um leiaute de linha existente chamado **SimpleListItem1**, que representa uma única linha de texto, para definir a aparência do ListView. Este layout de item contém um único **TextView** permitindo exibir uma única linha de texto.

Nota: *Existem diversos layouts de itens de lista incorporados ao Xamarin.Android como :* ***SimpleListItem2*** , ***TwoLineListItem*** , ***ActivityListItem*** , ***SimpleListItem2*** , ***TestListItem*** , *etc.*

```
ArrayAdapter<string> adapter = new ArrayAdapter<string>(this, Android.Resource.Layout.SimpleListItem1, times);
```

Os argumentos usados são:

- O primeiro argumento é **this** : é o contexto da aplicação;
- O segundo argumento é o leiaute definido no arquivo XML que possui o **TextView** para cada item do array. Estamos usando : **SimpleListItem1**
- O terceiro argumento é o **array** de strings que será usado para preencher o texto da view;

5 - Concluindo usamos a propriedade **Adapter** que retorna o adaptador atualmente em uso nesta **ListView** e exibe os dados na view:

```
listView.Adapter = adapter;
```

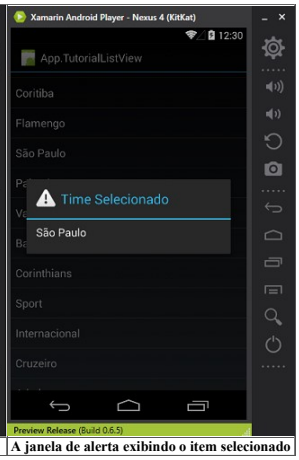
6- No evento **Click** do Item do **ListView** definimos uma janela de alerta para exibir qual o item foi selecionado :

```
listView.ItemClick += (sender, e) =>
{
    using (var dialog = new AlertDialog.Builder(this))
    {
        int posicao = e.Position;
        string valor = times[posicao];
        dialog.SetIcon(Android.Resource.Drawable.IcDialogAlert);
        dialog.SetTitle("Time Selecionado");
        dialog.SetMessage(valor);
        dialog.Show();
    }
};
```

Para saber mais sobre diálogos de alerta veja o artigo : Xamarin Android - Exibindo uma janela de Alerta com AlertDialog.Builder

Executando o projeto iremos obter o seguinte resultado:

A tela de apresentação



A janela de alerta exibindo o item selecionado

Existe uma abordagem mais simples para exibir informações em um ListView usando a classe **ListActivity** que é uma *Activity* que contém um ListView.

Aguarde em breve mais artigos sobre o componente **ListView** em aplicações Xamarin Android.

Pegue o projeto completo aqui : **App.TutorialListView2.zip** (*sem as referências*)

(Disse Jesus aos seus discípulos) : "Se vós fôsseis do mundo, o mundo amaria o que era seu, mas porque não sois do mundo, antes eu vos escolhi do mundo, por isso é que o mundo vos odeia."
João 15:19

Veja os Destaques e novidades do SUPER DVD Visual Basic (sempre atualizado) : clique e confira !

Quer migrar para o VB .NET ?

- Veja mais sistemas completos para a plataforma .NET no **Super DVD .NET** , confira...
- **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 - Vídeo 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**
- 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.Widget.ListView/
- https://developer.xamarin.com/api/property/Android.Widget.ListView.Adapter/

José Carlos Macoratti