Macoratti.net Xamarin Android - Criando uma View de Login

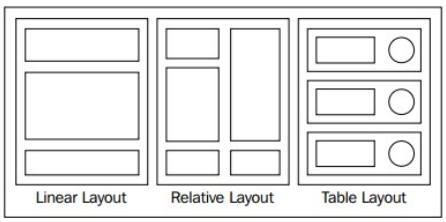
Neste artigo vou mostrar como criar uma view de **Login** para o Xamarin Android usando o Visual Studio Community 2015 e a linguagem C#.

Tudo o que você vê em um aplicativo Android é uma **View** : botões, etiquetas, caixas de texto e botões de rádio são todos exemplos de **views**.

O Android oferece um conjunto abrangente de ***widgets*** de interface que podem ser usados para construir uma rica experiência do usuário. Todos estes elementos são subtipos de views e podem ser organizados em layouts sofisticados que utilizam vários tipos de **ViewGroups**. Todos os **widgets** de interface podem ser encontrados no pacote **android.widget** no Application Framework.

O Application Framework possui um número ode subclasses de **ViewGroup**, cada uma fornece uma única forma de organizar o conteúdo da interface.

Os layouts mais comuns podem ser vistos na figura abaixo:



Cada layout pode ser usado para um propósito especifico.

Layout	Descrição	Cenário
Linear Layout	Organize seus '*filhos*' em uma única linha horizontal ou vertical e cria uma barra de rolagem quando necessário.	Indicado quando os *widgets* fluem horizontal ou verticalmente
Relative Layout	Organiza os objetos '*filhos*' relativamente uns aos outros ou e relação ao pai.	Indicado quando as posições dos *widgets* podem ser melhor descritas em relação a outro elemento (à esquerda) ou à área de fronteira do pai (lado direito, ou centrado)
Table Layout	Organiza os seus '*filhos*' em linhas e colunas	Indicado quando as posições dos *widgets* naturalmente se encaixam em linhas e colunas.

Dessa forma podemos criar diversos layouts usando os recursos do Xamarin, como podemos também usar templates prontos que se ajustam a nossa necessidade.

Neste artigo eu mostro como criar uma **View** simples para usar com uma interface de **Login** do usuário.

Recursos usados:

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

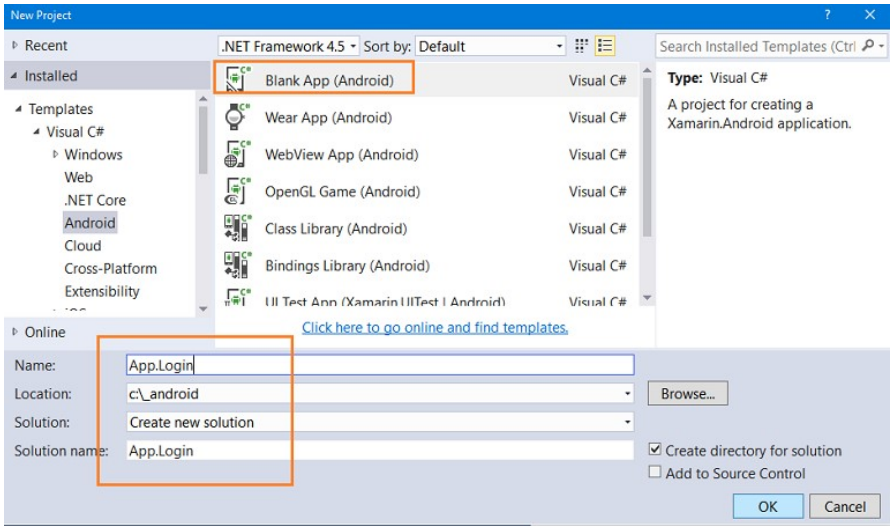
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no VS Community com Xamarin

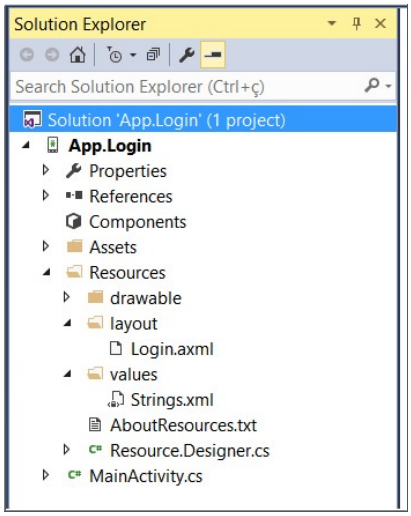
Abra o VS Community 2015 e clique em **New Project**;

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

Informe o nome **App.Login** 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;

Nota : A pasta **Drawable** contém recursos como imagens png, jpg, etc., usadas no aplicativo. Ela contém múltiplas pastas especificas para cada resolução possível em uma aplicação Android. Numa aplicação típica Android você vai acabar encontrando as pastas: Drawable-LDPI, Drawable-mdpi, Drawable-hdpi, Drawable-xhdpi, Drawable-xxhdpi, etc.

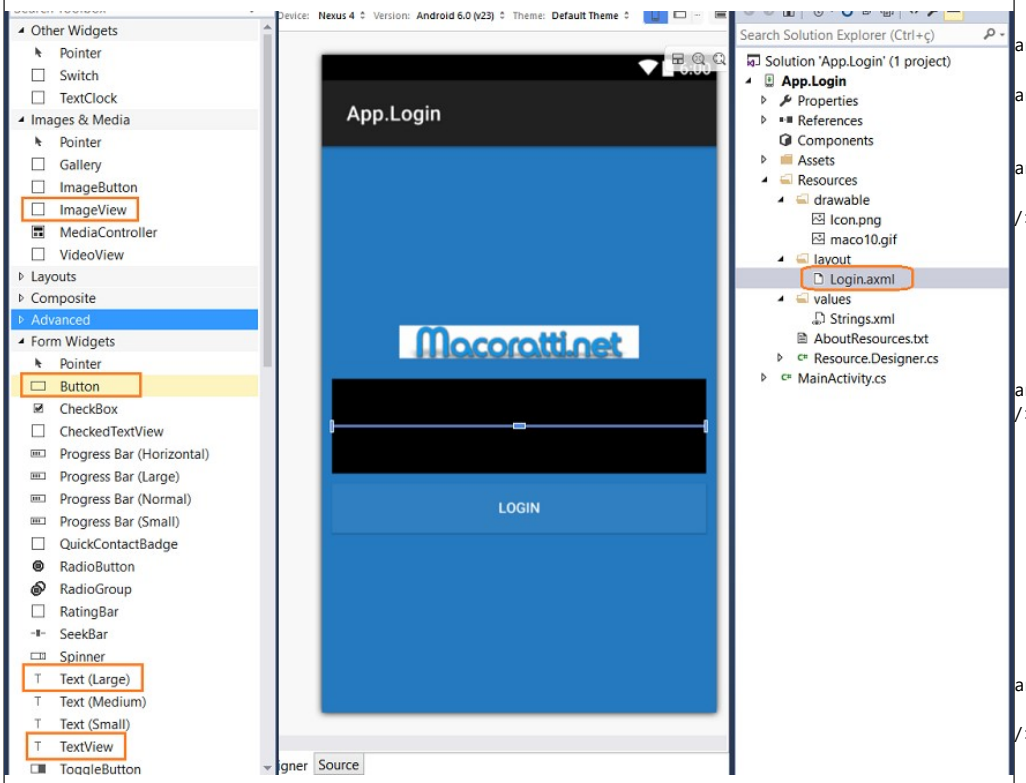
Inclua na pasta **Resources/drawable** um arquivo de imagem que deverá ser exibido na view **ImageView**. No exemplo eu estou usando o arquivo **maco10.gif**.

A seguir localize o arquivo **Main.axml** na pasta **Resources\layout** e altere o seu nome para **Login.axml.**

A seguir inclua a partir da ToolBox os seguintes controles:

- **LinearLayout**
- **ImageView**
- **EditText**
- **TextView**
- **EditText**
- **Button**

Disponha os controles conforme o leiaute da figura abaixo:




```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com
/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#2579BF">
   <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:layout_margin="10dp"
      android:orientation="vertical">
      <ImageView

android:layout_width="wrap_content"

android:layout_height="wrap_content"
         android:layout_gravity="center"

android:layout_marginBottom="20dp"
         android:src="@drawable/maco10"
/>
      <EditText
         android:id="@+id/nomeUsuario"
         android:layout_width="fill_parent"
         android:layout_height="45dp"
         android:padding="5dp"

android:background="@android:color/black"
/>
      <TextView
         android:layout_width="fill_parent"
         android:layout_height="2dp"
         android:background="#f7f7f7" />
      <EditText
         android:id="@+id/senha"
         android:layout_width="fill_parent"
         android:layout_height="45dp"
         android:padding="5dp"

android:background="@android:color/black"
         android:inputType="textPassword"
/>
      <Button
         android:id="@+id/login"
         android:layout_width="fill_parent"

android:layout_height="wrap_content"
         android:layout_below="@id/senha"
         android:layout_marginTop="10dp"
         android:background="#307FC1"
         android:text="Login"

android:textColor="@android:color/white"
/>
   </LinearLayout>
</RelativeLayout>
```

Ao lado da figura vemos o código XML gerado.

O arquivo **MainActivity**.cs, como o nome já sugere, é onde esta definida a **Activity** da aplicação Android.

Vamos agora abrir o arquivo **MainActivity.cs** e incluir o seguinte código:

```
using Android.App;
using Android.OS;
using Android.Widget;

namespace App.Login
{
   [Activity(Label = "App.Login", MainLauncher = true, Icon = "@drawable/icon")]
   public class MainActivity : Activity
   {

      protected override void OnCreate(Bundle bundle)
      {
```

```
            base.OnCreate(bundle);

            SetContentView(Resource.Layout.Login);

            EditText usuario = FindViewById<EditText>(Resource.Id.nomeUsuario);
            usuario.SetTextColor(Android.Graphics.Color.Yellow);

            EditText senha = FindViewById<EditText>(Resource.Id.senha);
            senha.SetTextColor(Android.Graphics.Color.Yellow);

            Button login = FindViewById<Button>(Resource.Id.login);

            login.Click += delegate
            {

                Toast.MakeText(this, "Botão de Login foi clicado!", ToastLength.Short).Show();

            };
        }
    }
}
```

Executando o projeto iremos obter o seguinte resultado:

Em outro artigo eu vou mostrar como podemos continuar a implementação do Login.

Aguarde mais artigos sobre o desenvolvimento de aplicativos Android usando o Visual Studio e o Xamarin.

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

(Disse Jesus) "Eu sou a videira, vós as varas; quem está em mim, e eu nele, esse dá muito fruto; porque sem mim nada podeis fazer. Se alguém não estiver em mim, será lançado fora, como a vara, e secará; e os colhem e lançam no fogo, e ardem."
João 15:5,6

- **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**
- 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

---

José Carlos Macoratti