

## Macoratti.net Xamarin Android - CRUD

### Básico com SQLite - I



Neste tutorial vou mostrar como implementar as operações de manutenção de dados - **CRUD** - usando um banco de dados **SQLite** no Xamarin Android com **VS 2015 Community** e a linguagem **C#**.

Curso C# Vídeo Aulas  
Do básico ao intermediário

Por um preço justo

O **SQLite** é uma biblioteca de software escrito em C desenvolvido por **D. Richard Hipp** em 2000 originalmente como parte de um contrato com a Marinha dos EUA.

Hoje, ele consiste de mais ou menos 184.000 linhas de código e é de domínio público, por isso pode ser usado por qualquer pessoa.

Embora você possa encontrar referências ao SQLite como sendo um SGBD, ele é, estritamente falando, apenas esta biblioteca. O recipiente no qual a biblioteca é compilada (*uma classe, um framework, ou um SGBD de pleno direito*) fornece as funcionalidades de DBMS maiores.

Como o **SQLite** não requer administração, ele funciona bem em dispositivos que devem operar sem o suporte de um especialista. Por isso ele é muito usado em celulares, televisores, consoles de jogos, câmeras, relógios, utensílios de cozinha, termostatos, automóveis, máquinas-ferramentas, aviões, sensores remotos, zangões, dispositivos médicos, e robôs: a "*internet das coisas*".

Assim, um banco de dados SQL completo com várias **tabelas, índices, triggers e views**, está contido em um único arquivo de disco. O formato de arquivo de banco de dados é multiplataforma - você pode copiar livremente um banco de dados entre os sistemas de 32 bits e de 64 bits ou entre arquiteturas de *big-endian* e *little-endian*.

A versão atual do SQLite é o **SQLite 3**.

Neste tutorial eu vou criar uma aplicação **Xamarin Android** que usa o banco de dados(*vou chamar assim*) **SQLite** para gerenciar informações de **alunos** usando as operações **CRUD - Create, Read, Update e Delete**.

#### 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**;

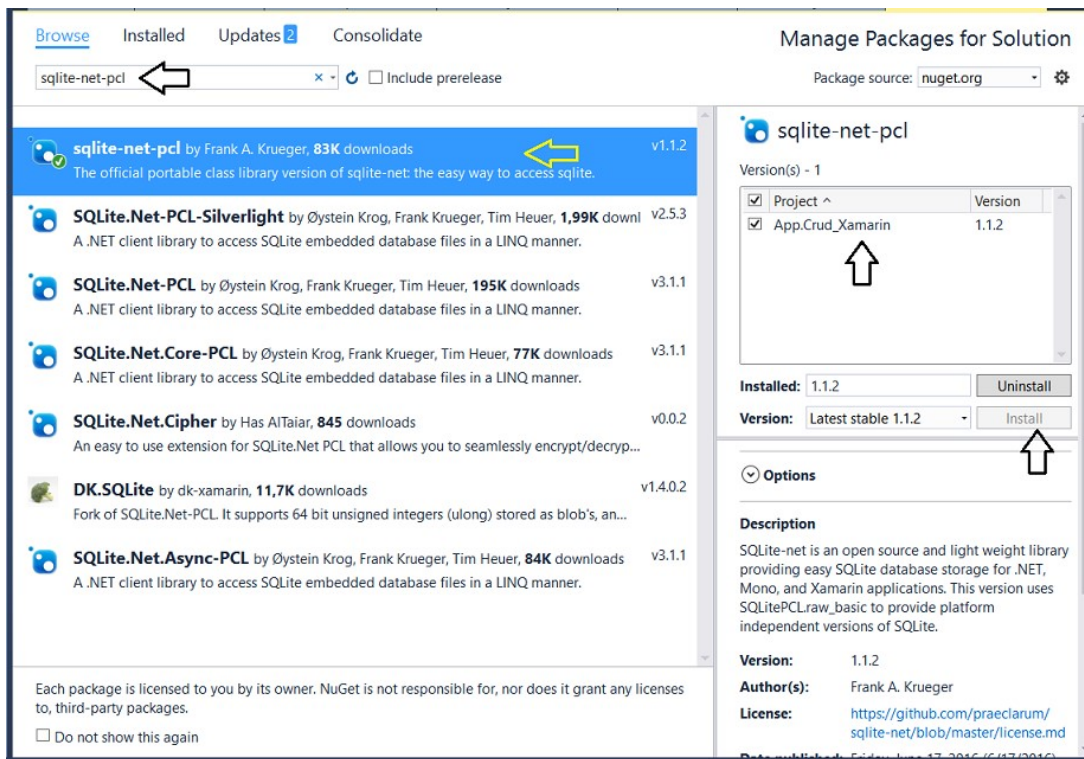
Selecione a linguagem Visual C# e o template **Android -> Blank App(Android)**

Informe o nome **App.Crud\_Xamarin** e clique no botão **OK**;

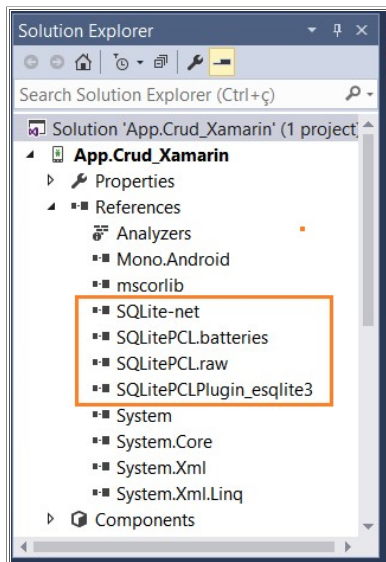
### Referenciando o SQLite no projeto via Nuget

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

Digite **sqlite-net-pcl** e selecione o pacote, o projeto e clique no botão **Install**:



Ao final você deverá ver os pacotes instalados indicados em **References**:



Assim estamos prontos para usar os recursos do SQLite no projeto.

## Definindo o modelo de domínio : a classe Aluno

Clique com o botão direito sobre a pasta **Resources** e a seguir em **Add Folder** e informe o nome **Model**.

Selecione a pasta **Model** e no menu **Project** clique em **Add Class** e informe o nome **Aluno.cs**.

A seguir defina o código da classe **Aluno** conforme o código abaixo:

using SQLite;

```
namespace
App.Crud_Xamarin.Resources.Model
{
    public class Aluno
    {
        [PrimaryKey, AutoIncrement]
```

```

    public int Id { get; set; }

    public string Nome { get; set; }
    public int Idade { get; set; }
    public string Email { get; set; }
}
}

```

O atributo o **[Primary-Key, AutoIncrement]** torna o campo Id uma chave primária autoincremental.

## Definindo a camada de acesso a dados : a classe DataBase

Clique com o botão direito sobre a pasta **Resources** e a seguir em **Add Folder** e informe o nome **DataBaseHelper**.

Selecione a pasta **DataBaseHelper** e no menu **Project** clique em **Add Class** e informe o nome **DataBase.cs**.

A seguir defina o código da classe **DataBase** conforme o código abaixo:

```

using Android.Util;
using App.Crud_Xamarin.Resources.Model;
using SQLite;
using System.Collections.Generic;
using System.Linq;

namespace App.Crud_Xamarin.Resources.DataBaseHelper
{
    public class DataBase
    {
        string pasta = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);

        public bool CriarBancoDeDados()
        {
            try
            {
                using (var conexao = new SQLiteConnection(System.IO.Path.Combine(pasta, "Alunos.db")))
                {
                    conexao.CreateTable<Aluno>();
                    return true;
                }
            }
            catch (SQLiteException ex)
            {
                Log.Info("SQLiteEx", ex.Message);
                return false;
            }
        }

        public bool InserirAluno(Aluno aluno)
        {
            try
            {
                using (var conexao = new SQLiteConnection(System.IO.Path.Combine(pasta, "Alunos.db")))
                {
                    conexao.Insert(aluno);
                    return true;
                }
            }
            catch (SQLiteException ex)
            {
                Log.Info("SQLiteEx", ex.Message);
                return false;
            }
        }

        public List<Aluno> GetAlunos()
        {
            try
            {
                using (var conexao = new SQLiteConnection(System.IO.Path.Combine(pasta, "Alunos.db")))
                {
                    return conexao.Table<Aluno>().ToList();
                }
            }
            catch (SQLiteException ex)
            {
                Log.Info("SQLiteEx", ex.Message);
                return null;
            }
        }
    }
}

```

```

    }
}

public bool AtualizarAluno(Aluno aluno)
{
    try
    {
        using (var conexao = new SQLiteConnection(System.IO.Path.Combine(pasta, "Alunos.db")))
        {
            conexao.Query<Aluno>("UPDATE Aluno set Nome=?,Idade=?, Email=? Where Id=?", aluno.Nome, aluno.Idade, aluno.Email, aluno.Id);
            return true;
        }
    }
    catch (SQLiteException ex)
    {
        Log.Info("SQLiteEx", ex.Message);
        return false;
    }
}

public bool DeletarAluno(Aluno aluno)
{
    try
    {
        using (var conexao = new SQLiteConnection(System.IO.Path.Combine(pasta, "Alunos.db")))
        {
            conexao.Delete(aluno);
            return true;
        }
    }
    catch (SQLiteException ex)
    {
        Log.Info("SQLiteEx", ex.Message);
        return false;
    }
}

public bool GetAluno(int Id)
{
    try
    {
        using (var conexao = new SQLiteConnection(System.IO.Path.Combine(pasta, "Alunos.db")))
        {
            conexao.Query<Aluno>("SELECT * FROM Aluno Where Id=?", Id);
            //conexao.Update(aluno);
            return true;
        }
    }
    catch (SQLiteException ex)
    {
        Log.Info("SQLiteEx", ex.Message);
        return false;
    }
}
}
}

```

Definimos nesta classe os seguintes métodos :

- **CriarBancoDeDados()**
- **InserirAluno()**
- **GetAlunos()**
- **AtualizarAluno()**
- **DeletarAluno()**
- **GetAluno()**

Com esse métodos poderemos realizar as operações de gerenciamento das informações dos alunos usando o banco de dados **Alunos.db** e a tabela **Aluno**.


## Definindo a interface com o usuário no arquivo Main.xml

Abra o arquivo **Main.xml** na pasta **Resources/layout** e no modo **Designer** inclua os seguintes controles

- 3 EditText - txtNome, txtIdade, e txtEmail
- 3 Button - btnIncluir, btnEditar e btnDeletar

- 1 ListView - lvDados

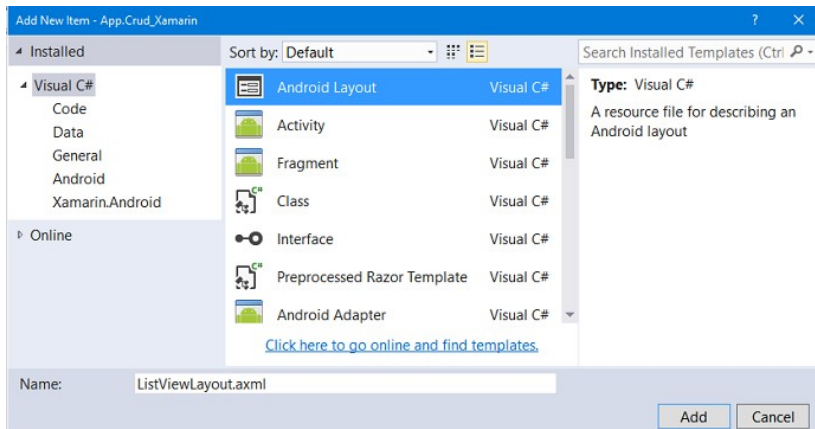
Abaixo vemos o leiaute no emulador do Xamarin exibindo a tela e ao lado o respectivo código XML gerado :

	<pre>&lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     android:orientation="vertical"     android:background="#720707"     android:layout_width="match_parent"     android:layout_height="match_parent"&gt;     &lt;EditText         android:hint="Informe o nome"         android:layout_width="match_parent"         android:layout_height="wrap_content"         android:id="@+id/txtNome" /&gt;     &lt;EditText         android:hint="Informe a Idade"         android:layout_width="match_parent"         android:layout_height="wrap_content"         android:id="@+id/txtIdade" /&gt;     &lt;EditText         android:hint="Informe o Email"         android:layout_width="match_parent"         android:layout_height="wrap_content"         android:id="@+id/txtEmail" /&gt;     &lt;LinearLayout         android:orientation="horizontal"         android:layout_width="match_parent"         android:layout_height="wrap_content"&gt;         &lt;Button             android:id="@+id/btnIncluir"             android:layout_width="wrap_content"             android:layout_height="wrap_content"             android:text="Incluir"             android:layout_weight="1" /&gt;         &lt;Button             android:id="@+id/btnEditar"             android:layout_width="wrap_content"             android:layout_height="wrap_content"             android:text="Editar"             android:layout_weight="1" /&gt;         &lt;Button             android:id="@+id/btnDeletar"             android:layout_width="wrap_content"             android:layout_height="wrap_content"             android:text="Deletar"             android:layout_weight="1" /&gt;     &lt;/LinearLayout&gt;     &lt;ListView         android:minWidth="25px"         android:minHeight="25px"         android:background="#8b7355"         android:layout_width="match_parent"         android:layout_height="match_parent"         android:id="@+id/lvDados" /&gt; &lt;/LinearLayout&gt;</pre>
--	---

### Definindo o arquivo de Layout para exibir os itens no ListView : Arquivo ListViewLayout.axml

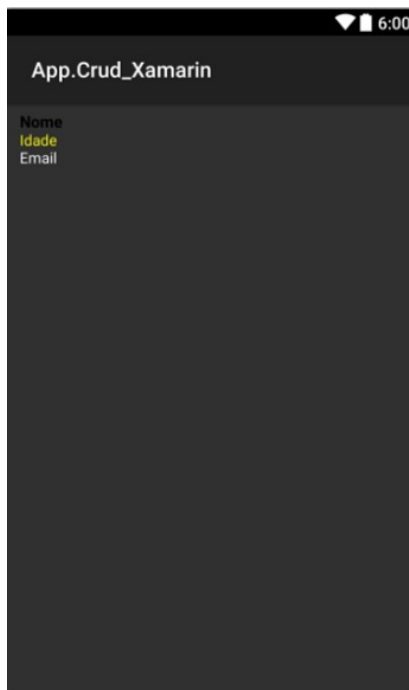
Clique com o botão direito do mouse sobre a pasta **Resources/layout** e no menu **Project** clique em **Add New Item**;

Selecione o template **Android Layout** e informe o nome **ListViewLayout.axml** e clique em **Add**;



A seguir vamos incluir a partir da **ToolBox** os seguintes controles :

- 3 **TextView** : txtvNome, txtvIdade e txtvEmail



```
<?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:padding="8dp">
    <TextView
        android:id="@+id/txtvNome"
        android:text="Nome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textSize="15dp"
        android:textStyle="bold"
        android:paddingLeft="5dp" />
    <TextView
        android:id="@+id/txtvIdade"
        android:text="Idade"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#ffff00"
        android:paddingLeft="5dp" />
    <TextView
        android:id="@+id/txtvEmail"
        android:text="Email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#ffffff"
        android:paddingLeft="5dp" />
</LinearLayout>
```

## Definindo a classe que implementa um **ListAdatper** a partir da classe **BaseAdapter**

Para personalizar o seu **ListView** vamos criar a classe **ListAdapter** que vai implementar a classe abstrata **BaseAdapter** sobrescrevendo os seguintes métodos:

- **Count** - Informa ao controle quantas linhas estão nos dados.
- **GetView** - Retorna uma View para cada linha, preenchida com dados.
- **GetItemId** - Retorna um identificador de linha (normalmente o número da linha)
- **GetItem** - Retorna informação sobre o item atual;

No menu **Project** clique em **Add Class** e informe o nome **ListAdapter.cs**;

Inclua o código abaixo nesta classe:

```
using Android.App;
using Android.Views;
using Android.Widget;
using App.Crud_Xamarin.Resources.Model;
using System.Collections.Generic;

namespace App.Crud_Xamarin.Resources
{
```

```

public class ListViewAdapter : BaseAdapter
{
    private readonly Activity context;
    private readonly List<Aluno> alunos;

    public ListViewAdapter(Activity _context, List<Aluno> _alunos)
    {
        this.context = _context;
        this.alunos = _alunos;
    }

    public override int Count
    {
        get
        {
            return alunos.Count;
        }
    }

    public override long GetItemId(int position)
    {
        return alunos[position].Id;
    }

    public override View GetView(int position, View convertView, ViewGroup parent)
    {
        var view = convertView ?? context.LayoutInflater.Inflate(Resource.Layout.ListViewLayout, parent, false);

        var lvtxtNome = view.FindViewById<TextView>(Resource.Id.txtvNome);
        var lvtxtIdade = view.FindViewById<TextView>(Resource.Id.txtvIdade);
        var lvtxtEmail = view.FindViewById<TextView>(Resource.Id.txtvEmail);

        lvtxtNome.Text = alunos[position].Nome;
        lvtxtIdade.Text = "" + alunos[position].Idade;
        lvtxtEmail.Text = alunos[position].Email;

        return view;
    }

    public override Java.Lang.Object GetItem(int position)
    {
        return null;
    }
}

```

Desses métodos o mais importante é o método **GetView()** que retorna uma view correspondendo aos dados a serem exibidos;

É dentro do método **GetView()** que vamos transformar o arquivo de layout **ListViewLayout.xml** em uma view contendo o leiaute do item da lista usando o método **inflate** da classe **LayoutInflater**.

O código é muito usado :

```
var view = convertView ?? context.LayoutInflater.Inflate(Resource.Layout.ListViewLayout, parent, false);
```

Este método cria uma nova view para cada filme adicionado ao **ListAdapter**.

Quando ele for chamado, a **View** é passada, o que normalmente é um objeto reciclado, então temos uma verificação para ver se o objeto é nulo. Se o objeto for nulo, uma **view** é instanciada e configurada com as propriedades desejadas para a apresentação dos itens.

Agora temos todas as peças prontas para podermos implementar no arquivo **MainActivity.cs** as funcionalidades do nosso CRUD.

Na [segunda parte do artigo](#) vamos implementar o CRUD.

Porque pela graça sois salvos, por meio da fé; e isto não vem de vós, é dom de Deus.

Não vem das obras, para que ninguém se glorie;

[Efésios 2:8,9](#)

[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 - 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 Video 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](#)
- [Xamarin - 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](#)

---

[José Carlos Macoratti](#)