Logotipo, nome da empresa

Descrição gerada automaticamente

Apostila – 08

**[Programação Para Dispositivos Móveis – PDM]**

**Cursos:**

**Tecnologia em Análise e Desenvolvimento de Sistemas**

**Ciência da Computação**

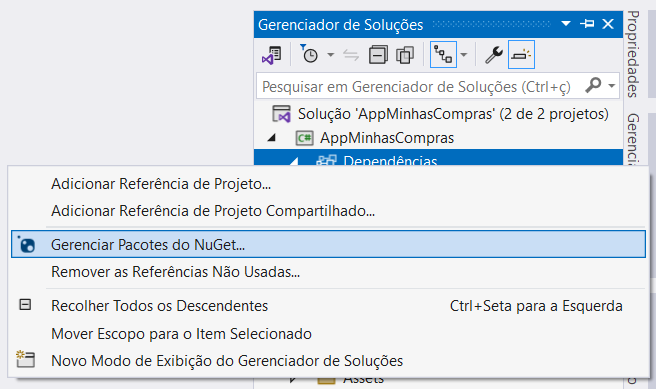
***Professor Msc Rafael Soares***

## Criando um CRUD no Abdroid com Xamarin.Forms

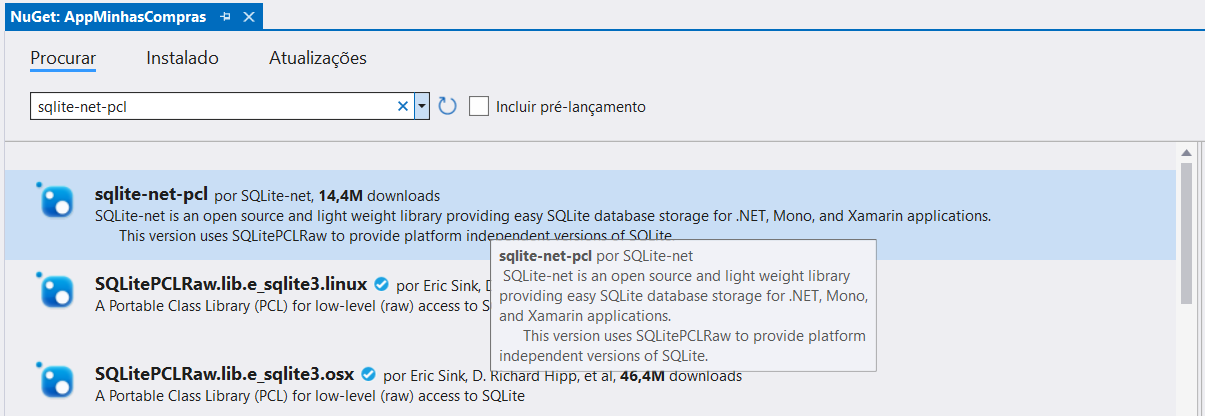
Vamos criar um novo projeto para entendermos estes conceitos, para isto, vamos retornar para a **IDE** do **Visual Studio 2022**, basta clicar no ícone do **Visual Studio 2022**, na barra de tarefas do **Windows**, e vai aparecer uma miniatura do **Visual Studio 2022**, conforme vimos na **figura 15**, se o mesmo estiver fechado, para abrir novamente, basta clicar sobre o ícone do programa que se encontra na barra de tarefas, conforme vimos na **figura 3**, e vai abrir a janela de itens recentes, conforme vimos na **figura 8**, para criar um novo projeto, temos três opções: poderíamos usar na janela de itens recentes a opção **Create a new project**, conforme vimos na **figura 22**, ou clicar no link **Continuar sem código (Continue without code)**, e aguardar a abertura da área do **IDE – Ambiente de Desenvolvimento Integrado do Visual Studio 2022 Community**, em seguida vá até o menu **File**, e selecione o item **New Project...**, ou pressionar ao mesmo tempo as teclas **Ctrl+Shift+N**, conforme vimos na **figura 23**, independentemente da forma escolhida para criar um novo projeto, vai abrir a janela **Create a new Project (Criar um novo projeto)**, no item **All Project type (Todos os Tipos de projeto)**, clique na setinha para baixo e selecione **Mobile**, em seguida, na lista **Filtering by: Mobile (Filtrado por: Mobile)**, selecione o item **Mobile App(Xamarin.Forms)** (**Aplicativo Móvel(Xamarin.Forms))**, conforme vimos na **figura 24**, clique no botão **Next (Próximo)**, e vai abrir a janela **Configure your new Project (Configurar seu novo Projeto)**,conforme vimos na **figura 25**, com a janela **Configure your new Project (Configurar seu novo Projeto)** aberta, no item **Project name:**, digite **AppMinhasCompras**, no item **Location**, vamos direcionar nosso projeto para que o mesmo possa trabalhar com controle de versão em conjunto com o **GitHub**, ou seja, vamos salvar na pasta **ProjetosLivro**, que nada mais é do que nosso repositório de controle de versão, clique no botão com três pontinhos, selecione **Win10(C:)**, no meu caso está com este nome, em seguida selecione a pasta **GitHub**, em seguida selecione a pasta **ProjetosLivro**, conforme vimos na **figura 184**, clique no botão **Selecionar pasta**, deixe marcado o item **Place solution and project in the same directory**, suas informações devem ficar conforme vimos na **figura 26**, em seguida, clique no botão **Create**. Vai abrir a janela **New Mobile App**, no item **Select a template for your app**, selecione o item **Blank**, no item **I plan to develop for:**, marques as opções **Android**, conforme vimos na **figura 27**, clique no botão **Create**, aguarde e vejam que vai aparecer a nossa **IDE** ou área de desenvolvimento, conforme vimos na **figura 28**.

## Referenciando o SQLite no Projeto via Nuget

Abra a janela **Solution Explorer**, pressione ao mesmo tempo as teclas **Ctrl+Alt+L**, e deve abrir a janela **Solution Explorer** conforme vimos na **figura 30**. Em seguida, abra o projeto **AppMinhasCompras**, clique com o botão direito do mouse sobre o item **Dependências**, e selecione o item de menu **Gerenciar Pacotes do NuGet...**, conforme figura abaixo:



Na janela **Manage Nuget Packages for Solution**, clique no item **Porcurar(Browse)**, na caixa de pesquisa, digite **sqlite-net-pcl**, aguarde a pesquisa e selecione o item **sqlite-net-pcl por SQLite-net, 14,4M**, conforme figura abaixo:

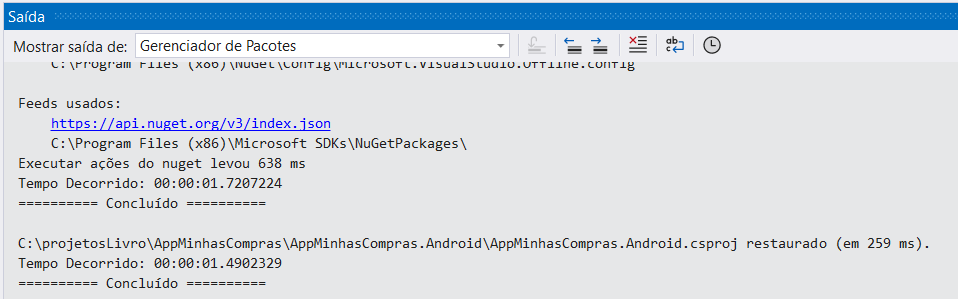


Na janela do lado direito, no item **Version:**, clique no botão **Install**, vai abrir a janela **Preview Changes**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Clique no botão **Aplicar**, e aguarde o término da instalação, conforme figura abaixo:



Agora, vamos criar algumas pastas, que iremos utilizar na nossa aplicação, para isto, clique com o botão direito do mouse no projeto comum **AppMinhasCompras**, em seguida, selecione o item de menu **Adicionar**, em seguida, selecione o item de menu **Nova Pasta**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

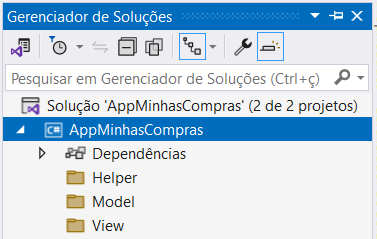
Descrição gerada automaticamente

No nome da pasta digite **Model**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Pressione a tecla **Enter** do teclado. Salve seu projeto. Repita o procedimento anterior e crie uma passa chamada **View** e outra pasta chamada **Helper**, vejam como ficou ao final, conforme figura abaixo:

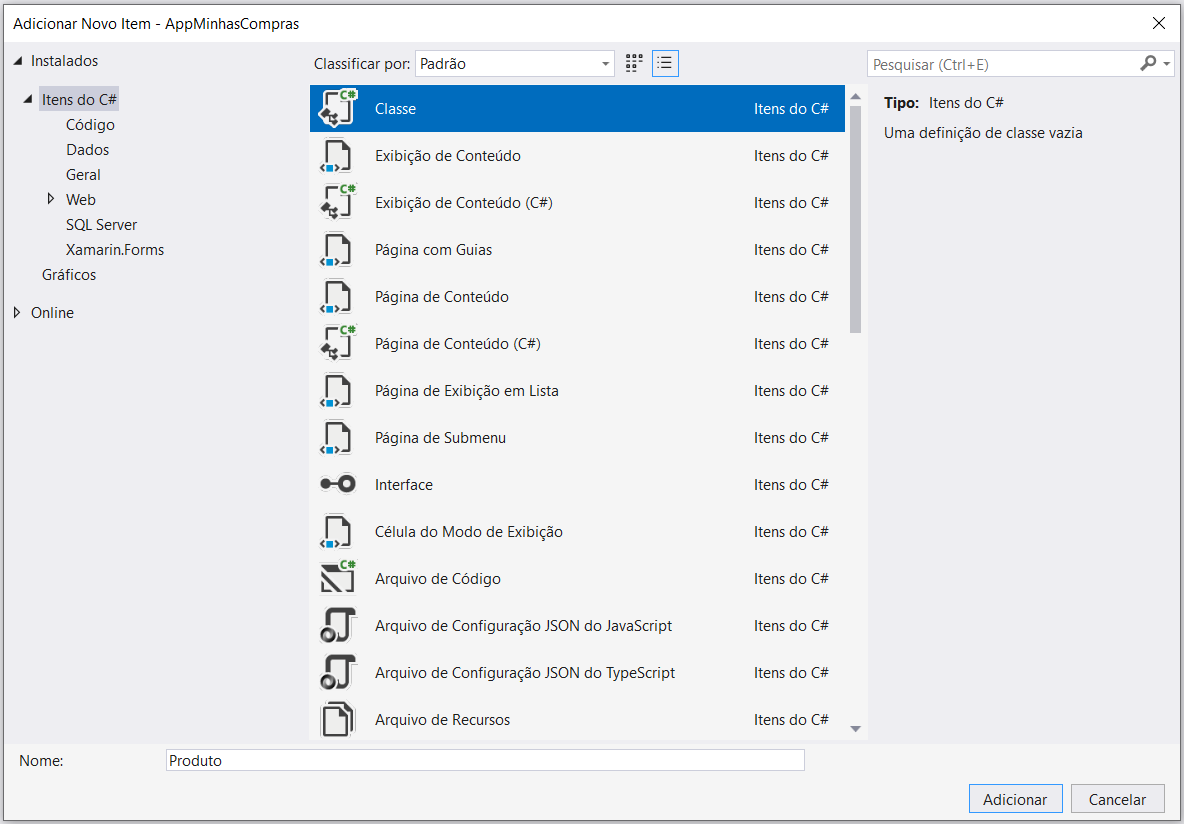


Agora vamos criar nossa **Classe Produto** dentro da pasta **Model**, para isto selecione com o mouse a pasta **Model**, que está dentro do projeto principal, dentro da janela **Solution Explorer**, em seguida selecione o item **Adicionar**, em seguida selecione o item **Classe...,** conforme figura abaixo:

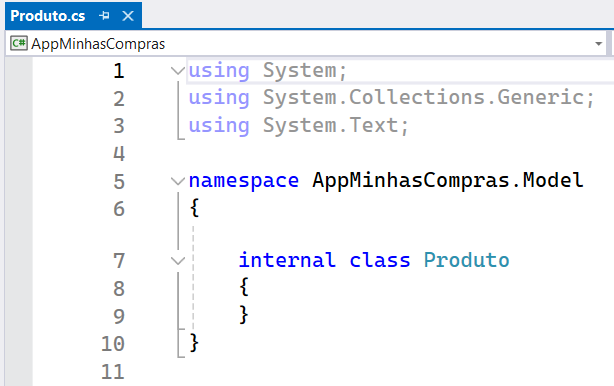
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

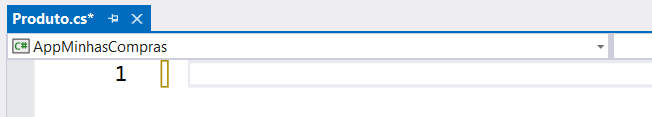
Vai abrir a janela **Adicionar Novo Item – AppMinhasCompras**, o tipo **Classe** vai estar selecionado, no item **Nome:**, digite **Produto**, conforme figura abaixo:



Clique no botão **Adicionar**, vai aparecer a área de código da classe **Produto**, conforme figura abaixo:



Pressione ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, agora pressione a tecla **Delete**, do teclado, e o cursor do mouse vai ficar piscando na **linha 1**, conforme figura abaixo:



Neste local, digite o código abaixo:

**/\*\***

**\* A biblioteca SQLite é chamada aqui para que as anotações de chave primária**

**\* e de autoIncremento possam ser usadas na propriedade proID**

**\*/**

**using SQLite;**

**namespace AppMinhasCompras.Model**

**{**

**/\*\***

**\* A classe model Produto define como os dados serão armazenados no SQLite (arquivo db3)**

**\* e como serão transportados entre a View e o SQLite.**

**\*/**

**public class Produto**

**{**

**[PrimaryKey, AutoIncrement]**

**public int ProID { get; set; }**

**public string ProDescricao { get; set; }**

**public double ProQuantidade { get; set; }**

**public double ProPreco { get; set; }**

**}**

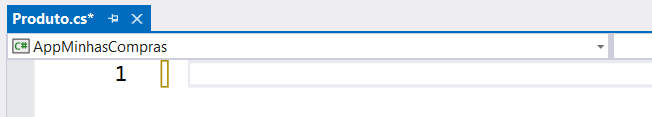
**}**

Agora, vamos adicionar na pasta **Helper**, a classe que vai fazer o nosso **CRUD**, para isto clique com o botão direito do mouse sobre a pasta **Helper**, que está dentro do projeto principal, dentro da janela **Solution Explorer**, em seguida selecione o item **Adicionar**, em seguida selecione o item **Classe...,** conforme vimos na figura anterior. Vai abrir a janela **Adicionar Novo Item – AppMinhasCompras**, o tipo **Classe** vai estar selecionado, no item **Nome:**, digite **SQLiteDatabaseHelper**, conforme vimos na figura anterior. Clique no botão **Adicionar**, vai aparecer a área de código da classe **SQLiteDatabaseHelper**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Pressione ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, agora pressione a tecla **Delete**, do teclado, e o cursor do mouse vai ficar piscando na **linha 1**, conforme figura abaixo:



Neste local, digite o código abaixo:

**/\*\***

**\* Arquivo Model para criação da Tabela e Transporte de Dados**

**\*/**

**using AppMinhasCompras.Model;**

**/\*\***

**\* Classes da Bilioteca do SQLite para acesso aos dados e criação da estrutura da tabela.**

**\*/**

**using SQLite;**

**/\*\***

**\* Classes como List.**

**\*/**

**using System.Collections.Generic;**

**/\*\***

**\* Criação de Threads e Tarefas para execução assíncrona (sem congelar a tela do app) de ações.**

**\*/**

**using System.Threading.Tasks;**

**namespace AppMinhasCompras.Helper**

**{**

**/\*\***

**\* Definição da classe SQLiteDatabaseHelper que funciona como uma abstração de acesso ao arquivo db3**

**\* do SQLite. A classe contém as informações de "conexão" e os métodos para realizar o CRUD (Create,**

**\* Read, Update e Delete).**

**\* Observe que na classe todos os métodos são Async, isso significa que todos são executados via Threads**

**\* o que, em teoria) não trava a interface do app enquanto os dados são lidos/gravados no arquivo db3.**

**\*/**

**public class SQLiteDatabaseHelper**

**{**

**/\*\***

**\* Campo da classe que armazena a "conexão" com o arquivo db3.**

**\* Isso siginifica que o arquivo db3 é aberto e armazenado aqui para que**

**\* essa classe possa usar os métodos da classe do SQLite para gravar**

**\* e ler dados do arquivo.**

**\*/**

**readonly SQLiteAsyncConnection \_conn;**

**/\*\***

**\* Método construtor da classe que recebe um parâmetro chamado path para**

**\* "conectar" ao arquivo db3.**

**\*/**

**public SQLiteDatabaseHelper(string path)**

**{**

**/\*\***

**\* Abrindo uma nova "conexão" com o arquivo db3 através do caminho recebido.**

**\* note a utilização da biblioteca SQLite "instalada" no projeto via pacote Nuget**

**\*/**

**\_conn = new SQLiteAsyncConnection(path);**

**/\*\***

**\* Criação da tabela com base no Model Produto (mais detalhes no arquivo Produto.cs na pasta Model)**

**\* Note que apesar do Async na criação da tabela é chamado o método Wait() que define a espera**

**\* da criação da tabela (se ela ainda não existir) antes de efetuar as outras operações, por exemplo,**

**\* insert.**

**\*/**

**\_conn.CreateTableAsync<Produto>().Wait();**

**}**

**/\*\***

**\* Método que faz a inseração de um novo registro na tabela. Veja que o método recebe uma Model**

**\* preenchida com os dados a serem inseridos. Observe que o método tem um retorno do tipo int**

**\* (número de linhas inseridas) sendo executado via Task (tarefa sendo executada de forma assíncrona).**

**\*/**

**public Task<int> Insert(Produto p)**

**{**

**return \_conn.InsertAsync(p);**

**}**

**/\*\***

**\* Método implementado com uso da estratégia de escrever o código SQL. Neste método podemos**

**\* ver a abstração que o SQLite faz, onde podemos digitar código SQL para manipulação do**

**\* arquivo db3. O método também recebe uma model preenchida para atualizar no db3 e o retorno**

**\* em forma de Task é uma lista de todos os registros atualizados.**

**\*/**

**public Task<List<Produto>> Update(Produto p)**

**{**

**string sql = "UPDATE Produto SET ProDescricao=?, ProQuantidade=?, ProPreco=? WHERE ProID= ? ";**

**return \_conn.QueryAsync<Produto>(sql, p.ProDescricao, p.ProQuantidade, p.ProPreco, p.ProID);**

**}**

**/\*\***

**\* Método que faz o retorno de todas as linhas contidas no arquivo db3 referentes**

**\* a tabela Produto. Veja que o método executa a listagem de forma assíncrona.**

**\*/**

**public Task<List<Produto>> GetAll()**

**{**

**return \_conn.Table<Produto>().ToListAsync();**

**}**

**/\*\***

**\* Método que remove um registro do arquivo db3 de forma assíncrona. Este método recebe**

**\* como parâmetro a proID do registro a ser removido. Observe o uso da LINQ no processo de**

**\* remoção.**

**\*/**

**public Task<int> Delete(int id)**

**{**

**return \_conn.Table<Produto>().DeleteAsync(i => i.ProID == id);**

**}**

**/\*\***

**\* Método para realizar uma busca na tabela com base uma string. O método recebe um**

**\* parâmetro do tipo string e por meio do SQL Like faz uma busca em um determinado campo**

**\* É retornada uma List de Produtos por meio de uma Task. A execução do SQL segue a mesma**

**\* linha utilizada no método update.**

**\*/**

**public Task<List<Produto>> Search(string q)**

**{**

**string sql = "SELECT \* FROM Produto WHERE Descricao LIKE '%" + q + "%' ";**

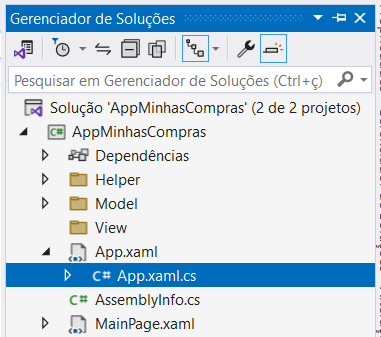
**return \_conn.QueryAsync<Produto>(sql);**

**}**

**}**

**}**

Agora vamos abrir a Classe **App.xaml.cs**, para isto basta clicar na aba com o nome da Classe. Se a mesma não estiver aberta, para abri-la, basta clicar duas vezes sobre o nome da Classe **App.xaml.cs**, que está dentro do nosso projeto principal, na janela **Solution Explorer**, conforme vimos na **Figura 259**. Clique na setinha que tem ao lado do arquivo **App.xaml**, em seguida clique duas vezes sobre a Classe **App.xaml.cs**, conforme figura abaixo:



Vai abrir a área de código da Classe **App.xaml.cs**. Pressione ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, agora pressione a tecla **Delete**, do teclado, e o cursor do mouse vai ficar piscando na **linha 1**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Neste local, digite o código abaixo:

**using System;**

**using Xamarin.Forms;**

**// Classe que gerencia o acesso ao SQLite**

**using AppMinhasCompras.Helper;**

**// Recurso da .NET para abstrair o acesso ao armazenamento interno do dispositivo.**

**using System.IO;**

**namespace AppMinhasCompras**

**{**

**public partial class App : Application**

**{**

**/\*\***

**\* Campo estático que contém a instância da classe que abstrai os métodos de gerenciamento**

**\* do SQLite. Leia mais sobre a classe no arquivo SQLiteDatabaseHelper.cs**

**\*/**

**static SQLiteDatabaseHelper database;**

**/\*\***

**\* Propriedade que define a forma de acesso a instância de SQLiteDatabaseHelper. A propriedade**

**\* é somente leitura, isto é, não é possível atribuir um valor a este campo. No comento que**

**\* o campo é chamado uma instância da classe SQLiteDatabaseHelper é criada (implementação get).**

**\*/**

**public static SQLiteDatabaseHelper Database**

**{**

**get**

**{**

**/\*\***

**\* Se o campo database for nulo, significa que ainda não foi atribuída uma instância de**

**\* SQLiteDatabaseHelper a ele, então uma nova instância será criada e esta mesma será usada**

**\* em todo tempo de execução do arquivo.**

**\*/**

**if (database == null)**

**{**

**/\*\***

**\* Para criar uma instância de SQLiteDatabaseHelper devemos o caminho do arquivo db3**

**\* (arquivo que contém as definições "DDL" e os dados propriamente ditos) no SQLite).**

**\* Devemos notar que essa abstração é necessária pois estamos em uma ferramenta multiplataforma**

**\* e isso significa que há um caminho diferente no Windows, Android e iOS e com o uso das**

**\* classes do System.IO podemos abstrair esse caminho.**

**\*/**

**string path = Path.Combine(**

**Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData),**

**"arquivo.db3"**

**);**

**/\*\***

**\* Criando uma instância de SQLiteDatabaBaseHelper como caminho até o arquivo db3 mencionado acima.**

**\*/**

**database = new SQLiteDatabaseHelper(path);**

**}**

**return database;**

**}**

**}**

**public App()**

**{**

**InitializeComponent();**

**/\*\***

**\* Habilitando o recurso de navegação entre páginas e definindo a página**

**\* de listagem (dentro da pasta View) como a tela inicial do App.**

**\*/**

**MainPage = new NavigationPage(new View.Listagem());**

**}**

**protected override void OnStart()**

**{**

**}**

**protected override void OnSleep()**

**{**

**}**

**protected override void OnResume()**

**{**

**}**

**}**

**}**

Observem que todo projeto **Xamarin.Forms** que criamos, automaticamente já vem com uma página de visualização ou **view**, vinculada, que é o arquivo **MainPage.xaml**, neste projeto não vamos precisar deste arquivo, vamos deletá-lo, para isto abra a janela **Solution Explorer**, pressione ao mesmo tempo as teclas **Ctrl+Alt+L**, e deve abrir a janela **Solution Explorer** conforme vimos na **figura 30**. Em seguida, abra o projeto **AppMinhasCompras**, para isto clique na setinha que tem ao lado do projeto **AppMinhasCompras**, em seguida clique com o botão esquerdo do mouse sobre o arquivo **MainPage.xaml**, e selecione o item de menu **Delete**, conforme **figura 109**:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Ou pressione a tecla **Delete** do teclado, vai parecer uma janela perguntando se queremos deletar permanentemente o arquivo, conforme **figura abaixo**:

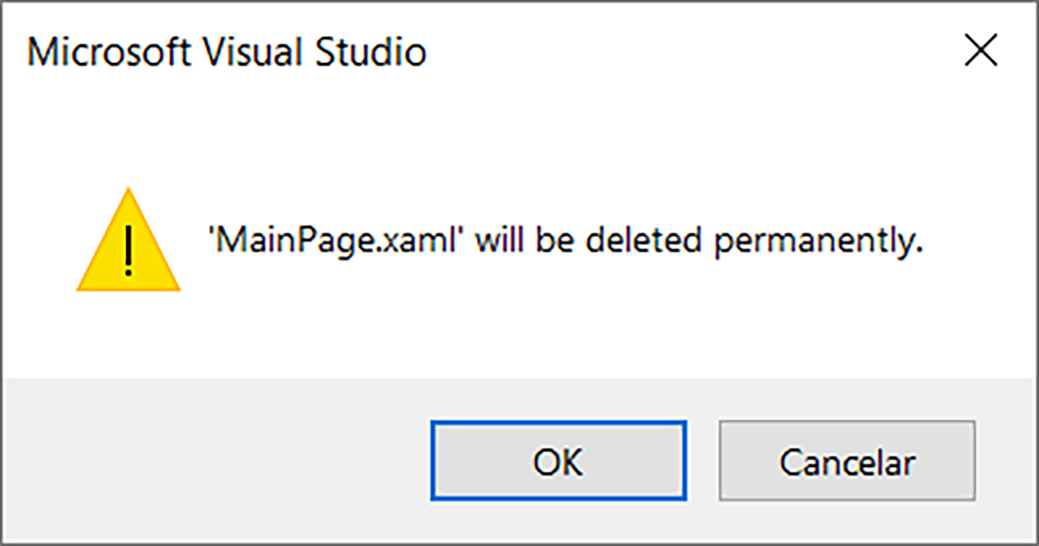


Figura 5

Clique no botão **OK**. Com a janela **Solution Explorer** aberta, clique com o botão esquerdo do mouse sobre a pasta **Views**, em seguida selecione o item de menu **Adicionar**, e selecione o item **Novo Item…**, ou pressione ao mesmo tempo as teclas **Ctrl+Shift+A**, conforme **figura abaixo**:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Vai abrir a janela **Adicionar Novo Item - AppMInhasCompras**, no menu do lado esquerdo, clique na setinha preta do item **Instalados**, para expandi-lo, em seguida clique na setinha preta do item **Itens do C#**, e selecione o item **Xamarin.Forms**, no menu do lado direito, selecione o item **Página de Conteúdo (Content Page)**, no item **Nome:**, digite **Listagem**, conforme **figura 114**:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Clique no botão **Adicionar**, e vai abrir o arquivo **Listagem.xaml**, conforme **figura abaixo**:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Repita o procedimento acima, e crie as páginas **NovoProduto**, **EditarProduto**, vejam como ficou nossa pasta **View**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Vamos retornar ao arquivo **Listagem.xaml**, se o mesmo estiver fechado, vamos abri-lo, para isto abra a janela **Solution Explorer**, pressione ao mesmo tempo as teclas **Ctrl+Alt+L**, e deve abrir a janela **Solution Explorer** conforme vimos na **figura 30**. Em seguida, abra o projeto **AppMinhasCompras**, para isto clique na setinha que tem ao lado do projeto **AppMinhasCompras**, em seguida clique duas vezes com o botão direito do mouse sobre o arquivo **Listagem.xaml**, e vai abrir o mesmo, agora pressione ao mesmo tempo as teclas **Ctrl+A** e vai selecionar todo o código, conforme vimos na **figura 116**. Pressione a tecla **Delete** do teclado, no local onde o cursor do mouse está piscando, na **linha 1**, digite o código abaixo:

**<?xml version="1.0" encoding="utf-8" ?>**

**<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"**

**xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"**

**x:Class="AppMinhasCompras.View.Listagem"**

**Title="Lista de Compras">**

**<ContentPage.ToolbarItems>**

**<ToolbarItem Text="Somar" Clicked="ToolbarItem\_Clicked\_Somar" />**

**<ToolbarItem Text="Novo" Clicked="ToolbarItem\_Clicked\_Novo" />**

**</ContentPage.ToolbarItems>**

**<ContentPage.Content>**

**<StackLayout>**

**<SearchBar x:Name="txtBbusca"**

**Placeholder="Qual o produto?"**

**TextChanged="txtBbusca\_TextChanged" />**

**<RefreshView x:Name="refCarregando" IsRefreshing="true">**

**<ListView x:Name="lstProdutos" ItemSelected="lstProdutos\_ItemSelected">**

**<ListView.Header>**

**<Grid RowDefinitions="Auto"**

**ColumnDefinitions="\*, Auto, Auto">**

**<Label Grid.Row="0" Grid.Column="0" Text="Descrição" />**

**<Label Grid.Row="0" Grid.Column="1" Text="Quantidade" />**

**<Label Grid.Row="0" Grid.Column="2" Text="Preço" />**

**</Grid>**

**</ListView.Header>**

**<ListView.ItemTemplate>**

**<DataTemplate>**

**<ViewCell>**

**<ViewCell.ContextActions>**

**<MenuItem Text="Excluir" Clicked="MenuItem\_Clicked\_Excluir" />**

**</ViewCell.ContextActions>**

**<Grid RowDefinitions="Auto"**

**ColumnDefinitions="\*, Auto, Auto">**

**<Label Grid.Row="0" Grid.Column="0" Text="{Binding ProDescricao}" />**

**<Label Grid.Row="0" Grid.Column="1" Text="{Binding ProQuantidade}" />**

**<Label Grid.Row="0" Grid.Column="2" Text="{Binding ProPreco}" />**

**</Grid>**

**</ViewCell>**

**</DataTemplate>**

**</ListView.ItemTemplate>**

**</ListView>**

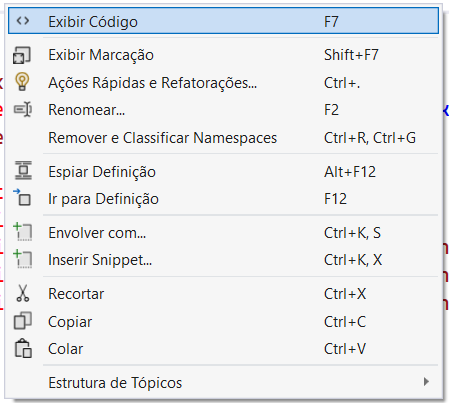
**</RefreshView>**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

Salve seu projeto, conforme vimos nas **figuras 48 e 49**. Vamos abrir o arquivo **Listagem.xaml.cs**, par isto, clique com o botão direito do mouse sobre o código, e selecione o item **Exibir Código**, conforme figura abaixo:



E vai abrir o mesmo, agora pressione ao mesmo tempo as teclas **Ctrl+A** e vai selecionar todo o código, conforme vimos na **figura 116**. Pressione a tecla **Delete** do teclado, no local onde o cursor do mouse está piscando, na **linha 1**, digite o código abaixo:

**using AppMinhasCompras.Model;**

**using System;**

**using System.Collections.Generic;**

**using System.Collections.ObjectModel;**

**using System.Linq;**

**using Xamarin.Forms;**

**using Xamarin.Forms.Xaml;**

**namespace AppMinhasCompras.View**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class Listagem : ContentPage**

**{**

**/\*\***

**\* A ObservableCollection é uma classe que armazena um array de objetos do tipo de Produto.**

**\* Utilizamos essa classe quando estamos apresentando um array de objetos ao usuário. Diferencial**

**\* dessa classe é que toda vez que um item é add, removido ou modificado no array de objetos a interface**

**\* gráfica também é atualizada. Assim as modificações feitas no array sempre estão na vista do usuário.**

**\*/**

**ObservableCollection<Produto> listaProdutos = new ObservableCollection<Produto>();**

**public Listagem()**

**{**

**InitializeComponent();**

**/\*\***

**\* Referenciando a fonte itens (a serem mostrados ao usuário) a ListView é a ObservableCollection**

**\* definida acima. Fazendo essa definição no construtor estamos amarrando a fonte de dados da ListView assim**

**\* que ela é criada.**

**\*/**

**lstProdutos.ItemsSource = listaProdutos;**

**}**

**/\*\***

**\* Tratamento do evento de clique no ToolBarItem que fará a navegação da tela de listagem**

**\* até a leta de cadastro de novo produto. A navegação está envolvida em um try catch**

**\* e se algum problema acontecer a mensação da exceção será mostrada ao usuário via DisplayAlert**

**\*/**

**private void ToolbarItem\_Clicked\_Novo(object sender, EventArgs e)**

**{**

**try**

**{**

**Navigation.PushAsync(new NovoProduto());**

**} catch(Exception ex)**

**{**

**DisplayAlert("Atenção", ex.Message, "OK");**

**}**

**}**

**/\*\***

**\* Método que faz a soma dos itens da ObservableCollection, isto é,**

**\* a soma do subtotal (preco x quantidade) de cada um dos itens do array de objetos**

**\*/**

**private void ToolbarItem\_Clicked\_Somar(object sender, EventArgs e)**

**{**

**/\*\***

**\* Uso da LINQ do C# para fazer a soma de cada um dos itens do array de objetos.**

**\*/**

**double soma = listaProdutos.Sum(i => i.ProPreco \* i.ProQuantidade);**

**string msg = "O total da compra é: " + soma;**

**DisplayAlert("Atenção", msg, "OK");**

**}**

**/\*\***

**\* Método executado quando a página é exibida ao usuário.**

**\*/**

**protected override void OnAppearing()**

**{**

**/\*\***

**\* Se a ObservableCollection estiver vazia é executado para obter todas as linhas do db3**

**\*/**

**if (listaProdutos.Count == 0)**

**{**

**/\*\***

**\* Inicializando a Thread que irá buscar o array de objetos no arquivo db3**

**\* via classe SQLiteDatabaseHelper encapsulada na propriedade Database da**

**\* classe App.**

**\*/**

**System.Threading.Tasks.Task.Run(async () =>**

**{**

**/\*\***

**\* Retornando o array de objetos vindos do db3, foi usada uma variável tem do tipo**

**\* List para que abaixo no foreach possamos percorrer a lista temporária e add**

**\* os itens à ObservableCollection**

**\*/**

**List<Produto> temp = await App.Database.GetAll();**

**foreach (Produto item in temp)**

**{**

**listaProdutos.Add(item);**

**}**

**/\*\***

**\* Após carregar os registros para a ObservableCollection removemos o loading da tela.**

**\*/**

**refCarregando.IsRefreshing = false;**

**});**

**}**

**}**

**/\*\***

**\* Trata o evento Clicked do MenuItem da ViewCell.ContextActions perguntando ao usuário**

**\* se ele realmente deseja remover aquele item do arquivo db3**

**\*/**

**private async void MenuItem\_Clicked\_Excluir(object sender, EventArgs e)**

**{**

**/\*\***

**\* Reconhecendo qual foi a linha do ListView que disparou o evento de exclusão.**

**\*/**

**MenuItem disparador = (MenuItem)sender;**

**/\*\***

**\* Obtendo qual foi o produto que estava anexado no BindingContext**

**\*/**

**Produto produtoSelecionado = (Produto)disparador.BindingContext;**

**/\*\***

**\* Perguntando ao usuário se ele realmente deseja remover. Note o await para aguardar**

**\* a resposta do usuário antes de prosseguir com o código.**

**\*/**

**bool confirmacao = await DisplayAlert("Tem Certeza que quer", " excluir o Produto?", "Sim", "Não");**

**if(confirmacao)**

**{**

**/\*\***

**\* Removendo o registro do db3 via método Delete da SQLiteDatabaseHelper**

**\*/**

**await App.Database.Delete(produtoSelecionado.ProID);**

**/\*\***

**\* Removendo o item da ObservableCollection também, que é automaticamente**

**\* removida da visão do usuário na ListView também.**

**\*/**

**listaProdutos.Remove(produtoSelecionado);**

**}**

**}**

**/\*\***

**\* Trata o evento TextChanged da SearchBar recebendo os novos valores digitados**

**\*/**

**private void txtBbusca\_TextChanged(object sender, TextChangedEventArgs e)**

**{**

**/\*\***

**\* Obtendo o valor que foi digitado no Search**

**\*/**

**string buscou = e.NewTextValue;**

**System.Threading.Tasks.Task.Run(async () =>**

**{**

**List<Produto> temp = await App.Database.Search(buscou);**

**/\*\***

**\* Limpando a ObservableCollection antes de add os itens vindos da busca.**

**\*/**

**listaProdutos.Clear();**

**foreach (Produto item in temp)**

**{**

**listaProdutos.Add(item);**

**}**

**refCarregando.IsRefreshing = false;**

**});**

**}**

**/\*\***

**\* Trata o evento ItemSelected da ListView navegando para a página de detalhes.**

**\*/**

**private void lstProdutos\_ItemSelected(object sender, SelectedItemChangedEventArgs e)**

**{**

**/\*\***

**\* Forma contraída de definir o BindingContext da página EditarProduto como sendo o**

**\* Produto que foi selecionado na ListView (item da ListView) e em seguida já**

**\* redicionando na navegação.**

**\*/**

**Navigation.PushAsync(new EditarProduto**

**{**

**BindingContext = (Produto)e.SelectedItem**

**});**

**}**

**}**

**}**

Vamos retornar ao arquivo **NovoProduto.xaml**, se o mesmo estiver fechado, vamos abri-lo, para isto abra a janela **Solution Explorer**, pressione ao mesmo tempo as teclas **Ctrl+Alt+L**, e deve abrir a janela **Solution Explorer** conforme vimos na **figura 30**. Em seguida, abra o projeto **AppMinhasCompras**, para isto clique na setinha que tem ao lado do projeto **AppMinhasCompras**, em seguida clique duas vezes com o botão direito do mouse sobre o arquivo **NovoProduto.xaml**, e vai abrir o mesmo, agora pressione ao mesmo tempo as teclas **Ctrl+A** e vai selecionar todo o código, conforme vimos na **figura 116**. Pressione a tecla **Delete** do teclado, no local onde o cursor do mouse está piscando, na **linha 1**, digite o código abaixo:

**<?xml version="1.0" encoding="utf-8" ?>**

**<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"**

**xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"**

**x:Class="AppMinhasCompras.View.NovoProduto"**

**Title="Novo Produto">**

**<ContentPage.ToolbarItems>**

**<ToolbarItem Text="Salvar" Clicked="ToolbarItem\_Clicked\_Salvar" />**

**</ContentPage.ToolbarItems>**

**<ContentPage.Content>**

**<StackLayout>**

**<Label Text="Descrição:" />**

**<Entry x:Name="txtDescricao" />**

**<Label Text="Quantidade:" />**

**<Entry x:Name="txtQuantidade" />**

**<Label Text="Preço:" />**

**<Entry x:Name="txtPreco" />**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

Salve seu projeto, conforme vimos nas **figuras 48 e 49**. Vamos abrir o arquivo **NovoProduto.xaml.cs**, par isto, clique com o botão direito do mouse sobre o código, e selecione o item **Exibir Código**, conforme figura abaixo:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

E vai abrir o mesmo, agora pressione ao mesmo tempo as teclas **Ctrl+A** e vai selecionar todo o código, conforme vimos na **figura 116**. Pressione a tecla **Delete** do teclado, no local onde o cursor do mouse está piscando, na **linha 1**, digite o código abaixo:

**using AppMinhasCompras.Model;**

**using System;**

**using Xamarin.Forms;**

**using Xamarin.Forms.Xaml;**

**namespace AppMinhasCompras.View**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class NovoProduto : ContentPage**

**{**

**public NovoProduto()**

**{**

**InitializeComponent();**

**}**

**/\*\***

**\* Trata o evento Clicked do ToolbarItem**

**\*/**

**private async void ToolbarItem\_Clicked\_Salvar(object sender, EventArgs e)**

**{**

**try**

**{**

**/\*\***

**\* Preenchendo o model Produto com os dados informados na interface gráfica.**

**\*/**

**Produto p = new Produto**

**{**

**ProDescricao = txtDescricao.Text,**

**ProQuantidade = Convert.ToDouble(txtQuantidade.Text),**

**ProPreco = Convert.ToDouble(txtPreco.Text),**

**};**

**/\*\***

**\* Chamando o método insert da SQLiteDatabaseHelper para fazer a inseração do**

**\* novo registro no arquivo db3 com os dados da model preenchida acima. O await**

**\* denota que o código deve esperar o insert para prosseguir.**

**\*/**

**await App.Database.Insert(p);**

**/\*\***

**\* Avisando o usuário que deu certo.**

**\*/**

**await DisplayAlert("Produto Cadastrado", " com Sucesso !!!!", "OK");**

**/\*\***

**\* Navegando para a tela de listagem.**

**\*/**

**await Navigation.PushAsync(new Listagem());**

**}**

**catch (Exception ex)**

**{**

**await DisplayAlert("Atenção", ex.Message, "OK");**

**}**

**}**

**}**

**}**

Vamos retornar ao arquivo **EditarProduto.xaml**, se o mesmo estiver fechado, vamos abri-lo, para isto abra a janela **Solution Explorer**, pressione ao mesmo tempo as teclas **Ctrl+Alt+L**, e deve abrir a janela **Solution Explorer** conforme vimos na **figura 30**. Em seguida, abra o projeto **AppMinhasCompras**, para isto clique na setinha que tem ao lado do projeto **AppMinhasCompras**, em seguida clique duas vezes com o botão direito do mouse sobre o arquivo **EditarProduto.xaml**, e vai abrir o mesmo, agora pressione ao mesmo tempo as teclas **Ctrl+A** e vai selecionar todo o código, conforme vimos na **figura 116**. Pressione a tecla **Delete** do teclado, no local onde o cursor do mouse está piscando, na **linha 1**, digite o código abaixo:

**<?xml version="1.0" encoding="utf-8" ?>**

**<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"**

**xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"**

**x:Class="AppMinhasCompras.View.EditarProduto"**

**Title="{Binding ProDescricao}">**

**<ContentPage.ToolbarItems>**

**<ToolbarItem Text="Salvar" Clicked="ToolbarItem\_Clicked\_Salvar" />**

**</ContentPage.ToolbarItems>**

**<ContentPage.Content>**

**<StackLayout>**

**<Label Text="Descrição:" />**

**<Entry x:Name="txtDescricao" Text="{Binding ProDescricao}" />**

**<Label Text="Quantidade:" />**

**<Entry x:Name="txtQuantidade" Text="{Binding ProQuantidade}" />**

**<Label Text="Preço:" />**

**<Entry x:Name="txtPreco" Text="{Binding ProPreco}" />**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

Salve seu projeto, conforme vimos nas **figuras 48 e 49**. Vamos abrir o arquivo **EditarProduto.xaml.cs**, par isto, clique com o botão direito do mouse sobre o código, e selecione o item **Exibir Código**, conforme figura abaixo:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

E vai abrir o mesmo, agora pressione ao mesmo tempo as teclas **Ctrl+A** e vai selecionar todo o código, conforme vimos na **figura 116**. Pressione a tecla **Delete** do teclado, no local onde o cursor do mouse está piscando, na **linha 1**, digite o código abaixo:

**using AppMinhasCompras.Model;**

**using System;**

**using Xamarin.Forms;**

**using Xamarin.Forms.Xaml;**

**namespace AppMinhasCompras.View**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class EditarProduto : ContentPage**

**{**

**public EditarProduto()**

**{**

**InitializeComponent();**

**}**

**private async void ToolbarItem\_Clicked\_Salvar(object sender, EventArgs e)**

**{**

**try**

**{**

**/\*\***

**\* Obtém qual foi o Produto anexado no BindingContext da página no momento que**

**\* ela foi criada e enviada para navegação.**

**\*/**

**Produto produtoAnexado = BindingContext as Produto;**

**/\*\***

**\* Preenchendo a model de acordo com os valores dos Entry. Note que recuperamos a Id**

**\* do BindingContext, como feito acima.**

**\*/**

**Produto p = new Produto**

**{**

**//Id = ((Produto) BindingContext).Id,**

**ProID = produtoAnexado.ProID,**

**ProDescricao = txtDescricao.Text,**

**ProQuantidade = Convert.ToDouble(txtQuantidade.Text),**

**ProPreco = Convert.ToDouble(txtPreco.Text),**

**};**

**/\*\***

**\* Método para atualizar o registro no arquivo db3. Note que o método recebe um model**

**\* preenchido e neste deve conter o Id para que seja feito o Where do Update.**

**\*/**

**await App.Database.Update(p);**

**await DisplayAlert("Produto Alterado", " com Sucesso !!!!", "OK");**

**await Navigation.PushAsync(new Listagem());**

**}**

**catch (Exception ex)**

**{**

**await DisplayAlert("Atenção", ex.Message, "OK");**

**}**

**}**

**}**

**}**