Disciplina	Desenvolvimento de Aplicações Multiplataforma (Xamarin)
Professor	Flávio Secchieri Mariotti

1. Introdução

Este documento contém instruções detalhadas de como criar uma aplicação com Xamarin.Forms baseado no padrão arquitetural MVVM. Além de, introduzir os princípios básicos deste padrão.

2. Projeto

O projeto criado nesta atividade apresenta conceitos básicos para implementação de aplicativos baseado em arquitetura MVVM. O projeto consiste de um objeto de visualização View que apresenta para o usuário os campos: Nome, RM e E-mail referente a entidade Aluno representado por meio de um objeto classe com as características de ViewModel, o qual está ligado com o objeto classe que representa o Model, instância com as regras de persistência e lógica de negócio.

2.1. Passo 1 – Criar o Projeto

Para criar um novo projeto Xamarin.Forms, abra o Visual Studio e clique em **File, New** e **Project**. Será apresentado uma janela com todos os templates instalados, expanda a opção Templates, Visual C# e clique em Cross-Platform.

Selecione o template localizado no Cross-Platform, Mobile App (Xamarin.Forms). Ainda na janela novo projeto, insira o nome do projeto, XF.MVVMBasic e informe a localização de sua preferência para criar o aplicativo. Se a opção “Criar diretório para solução” estiver marcado, o Visual Studio criará um novo diretório para o projeto (csproj), separado da solução (sln). Clique no botão OK e em seguida, marque o tipo de projeto Blank, verifique se as plataformas Android e iOS estão marcadas e selecione como estratégia de compartilhamento de código o modelo .NET Standard equivalente a arquitetura PCL.

Caso tenha seguido todas as instruções corretamente, a janela Novo Projeto estará conforme Figura 1.

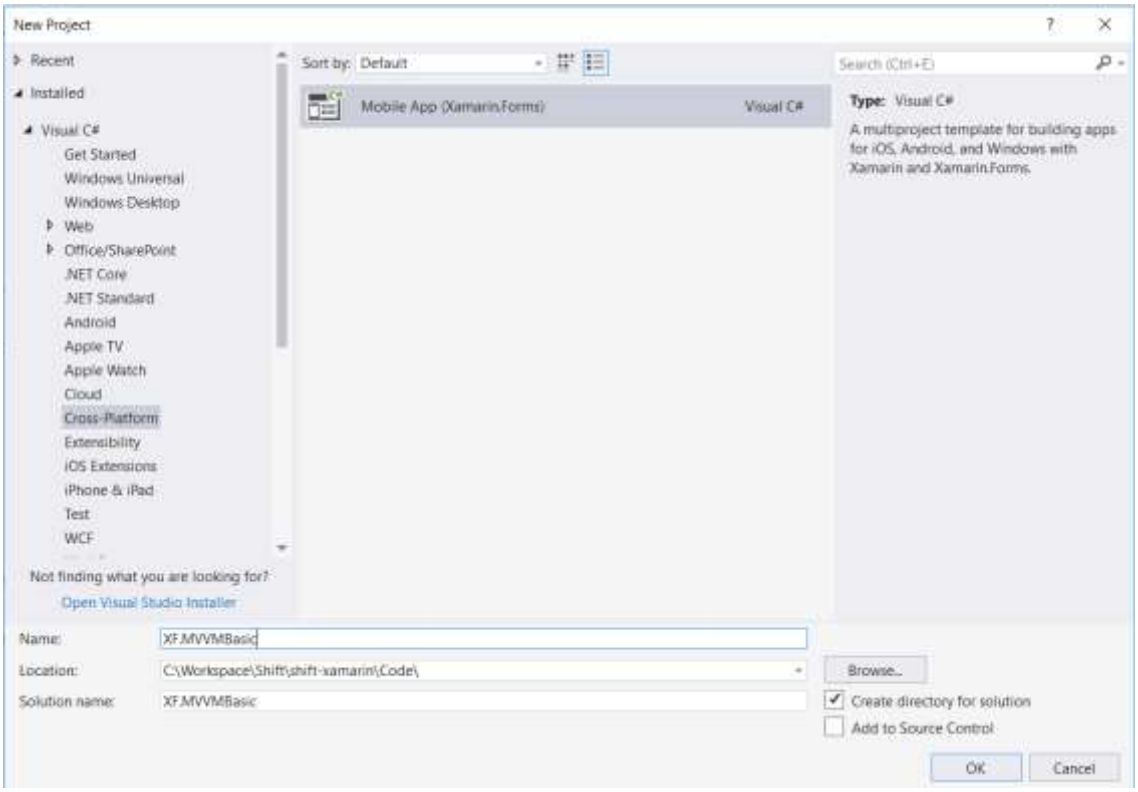

Figure 1. Janela Novo Projeto.

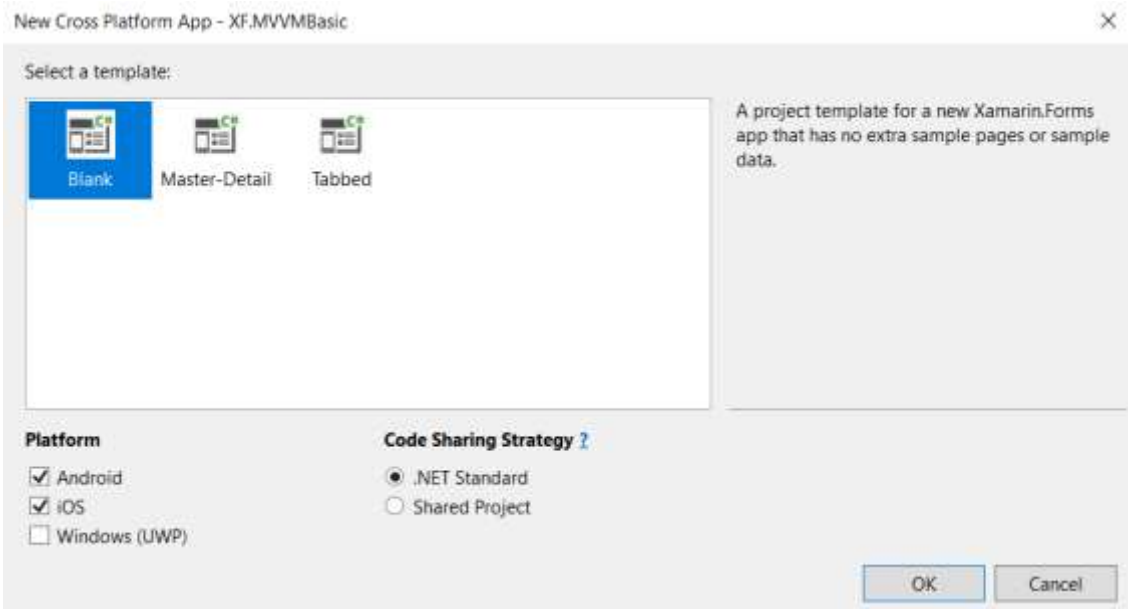

Figure 2. Janela Novo Projeto – Selecionar o tipo de projeto.

NOTA: o projeto criado é do tipo Portable Class Library (PCL) o que permite utilizar bibliotecas de classes portáteis, tais como, SQLite, Json.NET ou ReactiveUI em todas as plataformas por meio de DI. Diferentemente do tipo Shared Asset Project

(SAP) que anexa qualquer arquivo em um único projeto e compartilha automaticamente em todas as plataformas (código, imagens e qualquer outra mídia em iOS, Android e Windows). Saiba mais em: Introduction to Portable Class Libraries e Shared Projects.

2.2. Passo 2 – Criando o objeto View

Existe uma convenção entre os desenvolvedores .NET e Xamarin o que também é considerado como boas práticas de programação para implementação de projetos baseado na arquitetura MVVM, em outras palavras, modo de organização dos objetos e características que permite distribuir responsabilidades de acordo com as classes do tipo: Model, View e ViewModel.

Crie uma nova pasta no projeto XF.MVVMBasic com o nome View, para isso, clique com o botão direito do mouse no projeto citado e selecione a opção **Add, New Folder**. Na pasta criada, acrescente um novo item do tipo página, para isso, clique com o botão direito do mouse sobre a pasta View e escolha a opção novo item do tipo Content Page com o nome AlunoView.xaml. **Add, New Item, Xamarin.Forms** e selecione o template Content Page, conforme ilustrado pela Figura 3.

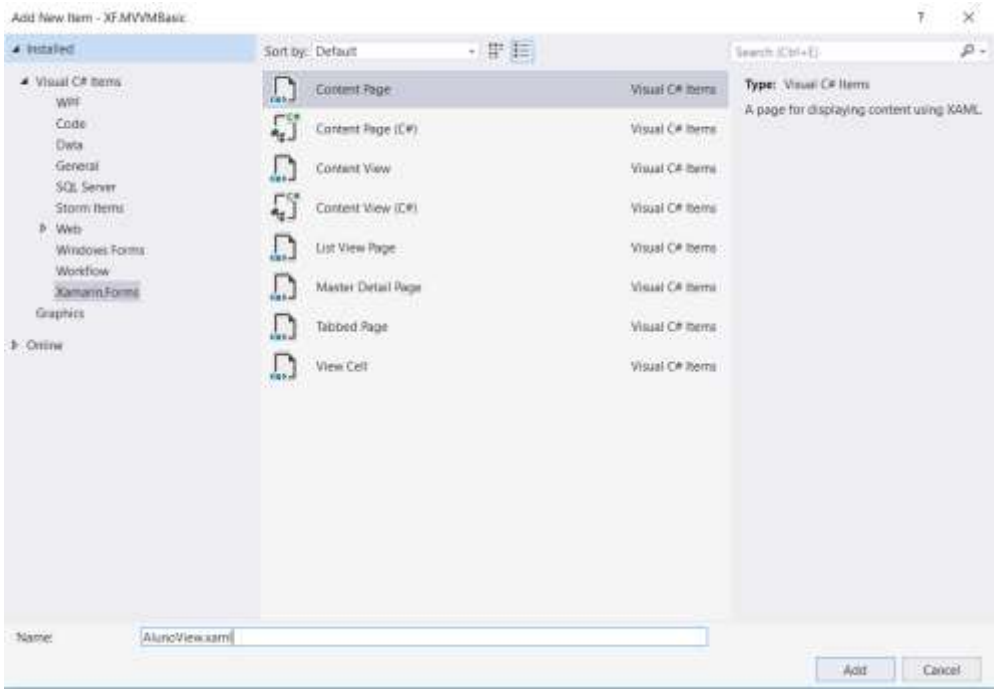

Figure 3. Janela novo item.

Neste exemplo, a implementação da página de aluno foi dividida em duas partes, visualização (xaml) e código-fonte (codebehind). Insira o código localizado na Tabela 1 no arquivo XAML e o código-fonte localizado na Tabela 2 no codebehind (.cs).

Tabela 1. Instruções XAML da página AlunoView.xaml

`<ContentPage.Content>`
`  <StackLayout Padding="20">`
`  <Label Text="Aluno" Font="20" />`
`  <Label Text="{Binding Nome}" />`
`  <Label Text="{Binding RM}" />`
`  <Label Text="{Binding Email}" />`
`  </StackLayout>`
`</ContentPage.Content>`
	StackLayout – elemento de layout que permite organizar os objetos de visualização de modo horizontal ou vertical.
	Label – elemento de visualização que cria uma etiqueta de texto.

Tabela 2. Instruções C# da página AlunoViewModel.cs

`AlunoViewModel vmAluno;`
 	
`public AlunoView()`
`{`
`  var aluno = AlunoViewModel.GetAluno();`
`  vmAluno = new AlunoViewModel(aluno);`
`  BindingContext = vmAluno;`
 	
`  InitializeComponent();`
`}`
	No construtor da página, foi criado uma instância que efetua a ligação entre a visualização e o objeto ViewModel referente ao modelo aluno.

2.3. Passo 3 – Criando o objeto Model

Crie uma nova pasta no projeto XF.MVVMBasic com o nome Model, para isso, clique com o botão direito do mouse no projeto citado e selecione a opção **Add**, **New Folder**. Crie um objeto do tipo Class, relativo a classe de entidade de domínio do Aluno. Clique com o botão direito do mouse sobre a pasta Model e crie novo item do tipo Class com o nome Aluno.cs, **Add**, **New Item**, **Code** e selecione o template Class.

Implemente a classe Aluno conforme código-fonte localizado na Tabela 3.

Tabela 3. Instruções C# da classe Aluno.cs

`public class Aluno`
`{`
`public Guid Id { get; set; }`
`public string RM { get; set; }`
`public string Nome { get; set; }`
`public string Email { get; set; }`
`}`
	Classe concreta referente da entidade aluno, a qual deverá conter as regras de negócio e acesso a base de dados.

2.4. Passo 4 – Criando o objeto ViewModel

Crie uma nova pasta no projeto XF.MVVMBasic com o nome ViewModel, clique com o botão direito do mouse no projeto citado e selecione a opção **Add**, **New Folder**. Crie um novo objeto do tipo Class, relativo a classe que conecta o modelo com a visualização, contendo o comportamento lógico e visual da interface do usuário (página). Clique com o botão direito do mouse sobre a pasta ViewModel e crie novo item do tipo Class com o nome AlunoViewModel.cs, **Add**, **New Item**, **Code** e selecione o template Class.

Implemente a classe AlunoViewModel de acordo com o código-fonte localizado na Tabela 4.

Tabela 4. Instruções C# da classe AlunoViewModel.cs

`public class AlunoViewModel`
`{`
`#region Propriedades`
`public string RM { get; set; }`
`public string Nome { get; set; }`
`public string Email { get; set; }`
`#endregion`
 	
`public AlunoViewModel(Aluno aluno)`
`{`
`this.RM = aluno.RM;`
`this.Nome = aluno.Nome;`
`this.Email = aluno.Email;`
`}`
 	
`public static Aluno GetAluno()`
`{`
`var aluno = new Aluno()`
`{`
`Id = Guid.NewGuid(),`
`RM = "542621",`
`Nome = "Anderson Silva",`
`Email = "anderson@ufc.com"`
`};`
`return aluno;`
	#region – diretriz de compilação utilizada para organizar, normalmente utilizado em arquivos extensos.
	
	Campos RM, Nome e Email, os quais serão apresentados na página (view).
	
	Método AlunoViewModel – construtor da classe AlunoViewModel que recebe como parâmetro um objeto aluno no qual passa os valores para as propriedades da classe.

} } }	Método GetAluno – exemplo simples que cria uma instância a partir do objeto aluno. Normalmente, esse tipo de método recupera os dados de uma base de dados ou por meio de Web Service.

2.5. Passo 5 – Configurando a classe App

Para finalizarmos o aplicativo, precisamos configurar a página principal do projeto e ativar o Page Stack. Implemente a classe App conforme código-fonte localizado na Tabela 5.

Tabela 5. Instruções C# da classe App.cs

public App()
{
InitializeComponent();
 	
MainPage = new NavigationPage(new View.AlunoView());
}
	Uso da classe NavigationPage para nagevar até a página AlunoView utilizando o padrão arquitetural page-based.

3. Aplicativo

Caso todos os passos tenham sido realizados conforme instruções do capítulo 2, o aplicativo deverá apresentar o aluno conforme Figura 4.

Figure 4. Aplicativo.

4. Desafio

Com o objetivo de praticar os conhecimentos adquiridos em sala de aula, implemente os desafios propostos abaixo:

1. Crie uma identidade visual utilizando os recursos de estilo;
2. Implemente no objeto ViewModel uma propriedade de coleção aluno;
3. Implemente uma nova página com o nome NovoAlunoView, a qual permitirá incluir novos registros na lista criada no requisito 2 (anterior);
4. Modifique a página AlunoView para que suporte uma lista de alunos e altere o método GetAluno localizado na classe AlunoViewModel para que retorne a lista de alunos criado no requisito 2;
5. Acrescente um botão na página AlunoView que navegue até a página de NovoAlunoView, permitindo que o usuário inclua novos alunos na lista;
6. Implemente validação no formulário de inclusão, no qual não autoriza o cadastro de novos alunos sem o campo Nome ser preenchido; e
7. Acrescente a funcionalidade de remoção de alunos.