

| | |
|-------------------|---|
| Disciplina | Desenvolvimento Mobile Cross-Platform (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 final os campos: Nome, RM e E-mail referente ao aluno selecionado por meio de um objeto classe (ViewModel), o qual está ligado com o objeto classe que representa a entidade (Model) aluno que contém as instruções de base de dados e lógica de negócio.

2.1. Passo 1 – Criar o Projeto

Para criar um novo projeto Xamarin.Forms, abra o Visual Studio 2017 e clique em **File, New e Project**. Será apresentada uma janela com todos os templates instalados, expanda a opção Templates, Visual C# e clique em Cross-Platform.

Na parte superior da janela, verifique se a versão do .NET selecionado seja a .NET Framework 4.6. Em seguida, marque o tipo de projeto Blank App (Xamarin.Forms Portable). Ainda na janela novo projeto, insira o nome do projeto, XF.MVVMBasic e informe a localização de sua preferência para criar o projeto. Se a opção “Criar diretório para solução” estiver marcado, o Visual Studio criará um novo diretório para o projeto (csproj), separado da solução (sln). Para criar o projeto, clique no botão OK e o Visual Studio deverá criar os seis projetos referente ao template selecionado.

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

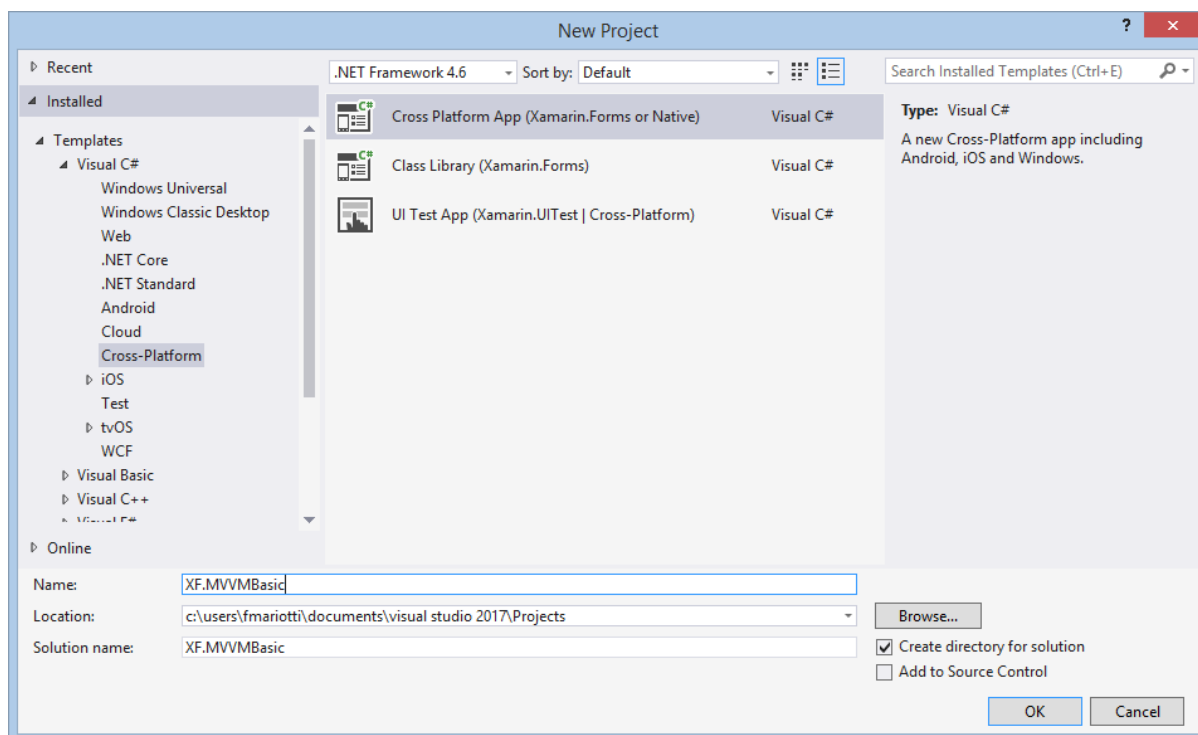


Figure 1. Janela Novo Projeto.

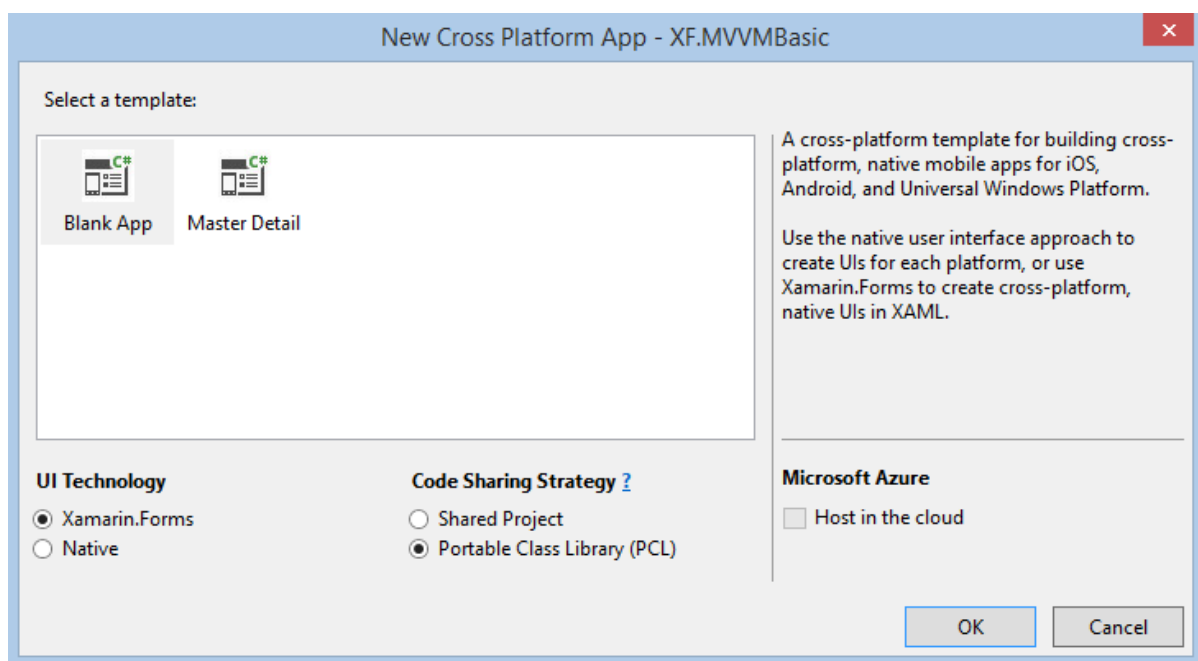


Figure 2. Janela Novo Projeto – Selecionar o tipo de projeto.

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 permite anexar qualquer arquivo em um único projeto e compartilhar 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](#).

Durante a criação do dos projetos, será apresentado a tela de configuração do Xamarin Mac Agent Instructions, o que permite conectar o PC com uma máquina MacOS, conforme Figura 3.

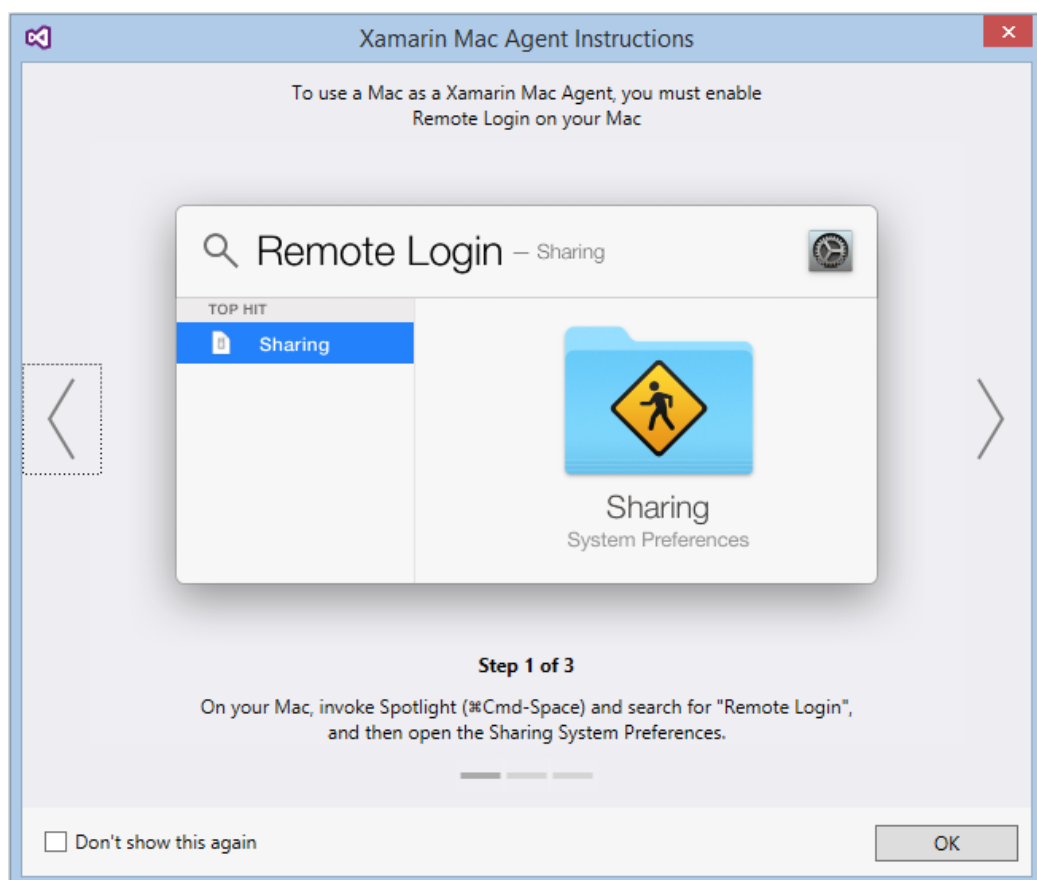


Figure 3. Xamarin Mac Agent Instructions.

Além disso, para criação do projeto Windows (UWP) também será apresentado uma nova janela, solicitando a configuração referente versão mínimo e máximo do sistema operacional que deverá oferecer suporte à aplicação XF.MVVMBasic.UWP.

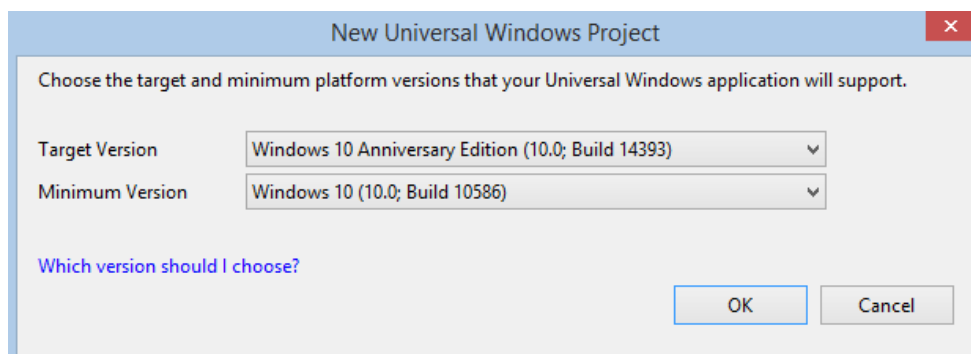


Figure 4. Selecionar a versão mínimo e máxima do Windows 10.

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, se refere na organização dos objetos em pastas específicas (Model, View e ViewModel) as quais permitem organizar os objetos de acordo com suas características.

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**. Crie um objeto do tipo Page, relativo a página de visualização do Aluno, clique com o botão direito do mouse sobre a pasta View e crie novo objeto do tipo *Content Page* com o nome AlunoView.cs. **Add, New Item, Xamarin.Forms** e selecione o template *Content Page*, conforme ilustrado pela Figura 5.

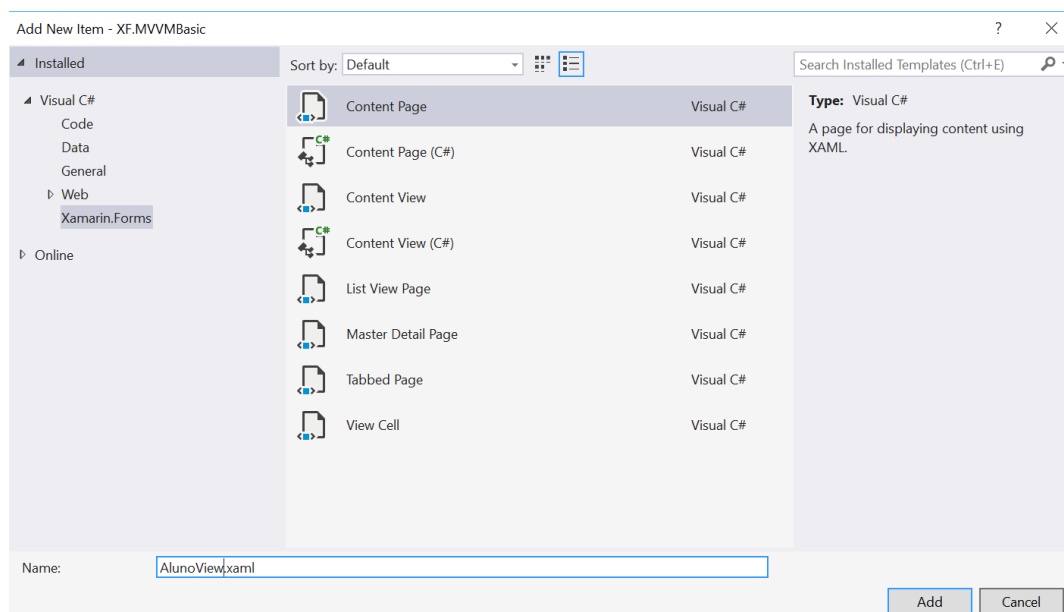


Figure 5. Janela novo item.

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

| | |
|---|--|
| <pre><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></pre> | <p>StackLayout – elemento de layout que permite organizar os objetos de visualização de modo horizontal ou vertical.</p> <p>Label – elemento de visualização que cria uma etiqueta de texto.</p> |
|---|--|

Tabela 2. Instruções C# da página AlunoViewModel.cs

| | |
|--|--|
| <pre>AlunoViewModel vmAluno; public AlunoView() { var aluno = AlunoViewModel.GetAluno(); vmAluno = new AlunoViewModel(aluno); BindingContext = vmAluno; InitializeComponent(); }</pre> | <p>No construtor da página, foi criada uma instância que efetua a ligação entre a visualização e o objeto ViewModel referente ao modelo aluno.</p> |
|--|--|

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 objeto do tipo Class com o nome Aluno.cs, **Add, New Item, Code** e selecione o template Class.

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

Tabela 3. Instruções C# da classe Aluno.cs

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

2.4. Passo 4 – Criando o objeto ViewModel

Crie uma nova pasta no projeto XF.MVVMBasic com o nome ViewModel, para isso, 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 sincroniza 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 objeto 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

| | |
|--|---|
| <pre>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" } } }</pre> | <p>#region – diretriz de compilação utilizada para organizar, normalmente utilizado em arquivos extensos.</p> <p>Campos RM, Nome e Email, os quais serão apresentados na página (view).</p> <p>Método AlunoViewModel – construtor da classe AlunoViewModel que recebe como parâmetro um objeto aluno no qual passa os valores para as propriedades da classe.</p> |
|--|---|

| | |
|---|--|
| <pre> }; return aluno; } } </pre> | <p>Método GetAluno – exemplo simples que instância e cria um objeto aluno. Normalmente, esse método deverá recuperar os dados de uma base de dados ou por meio de Web Service.</p> |
|---|--|

2.5. Passo 5 – Configurando a classe App

Para finalizarmos o aplicativo, precisamos configurar a página principal do projeto. Implemente a classe AlunoViewModel segundo o código-fonte localizado na Tabela 5.

Tabela 5. Instruções C# da classe App.cs

| | |
|---|--|
| <pre> public App() { // The root page of your application MainPage = new NavigationPage(new View.AlunoView()); } </pre> | <p>Uso da classe NavigationPage para navegar até a página AlunoView utilizando o padrão arquitetural page-based.</p> |
|---|--|

3. Aplicativo

Caso todos os passos tenham sido executados conforme instruções do capítulo 2, o aplicativo deverá apresentar o aluno conforme Figura 6.

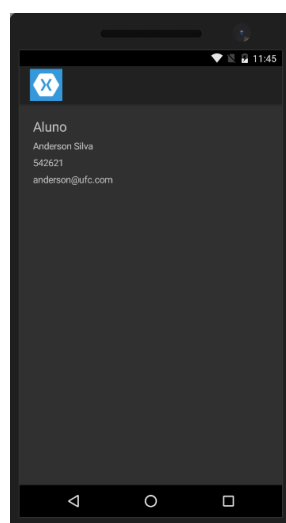


Figure 6. Aplicativo.

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 uma classe do tipo lista de alunos (coleção de alunos);
3. Implemente uma nova página (NovoAlunoView) que permite incluir novos alunos na lista de alunos criados no requisito 2;
4. Modifique a página AlunoView para que suporte uma lista de alunos e altere o método `GetAluno` localizado na classe `AlunoViewModel` para que carregue a lista de aluno criada no requisito 2;
5. Acrescente um botão na página AlunoView que navegue até a página de NovoAlunoView.