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

1. Introdução

Este documento contém instruções detalhadas de como criar um aplicativo com Xamarin.Forms e trabalhar com dados externos por meio de consumo a Web Services em protocolo HTTP e formato de mensagem JSON.

2. Projeto

O projeto se concentra em apresentar como criar aplicativos que possam ler e escrever dados externos por meio do ASP.NET Web API. O projeto consiste de duas páginas. A página Lista Professores, lista todos os professores registrados na tabela PROFESSORES da base de dados existente no Azure SQL Database. A página Novo Professor, permite cadastrar novos registros nesta tabela.

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.AplicativoFIAP 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.

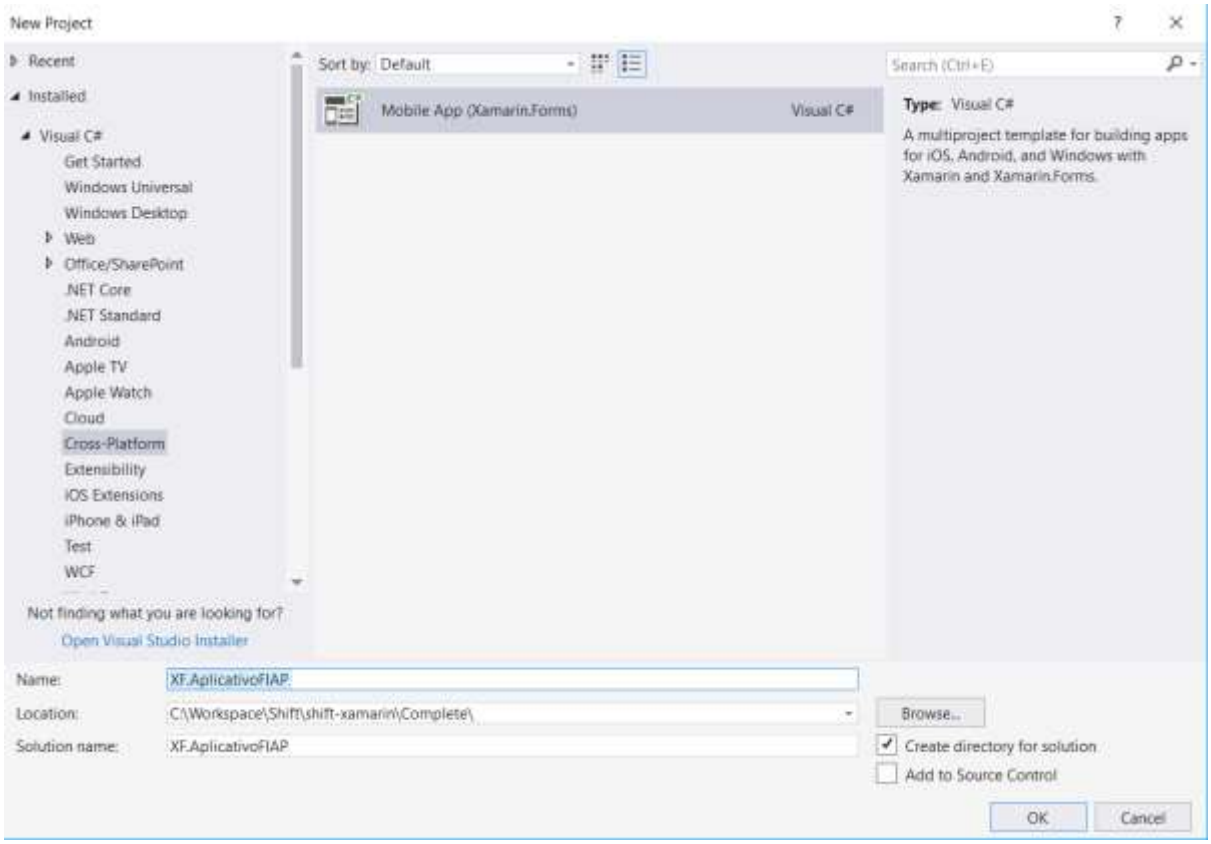


Figure 1. Janela Novo Projeto.

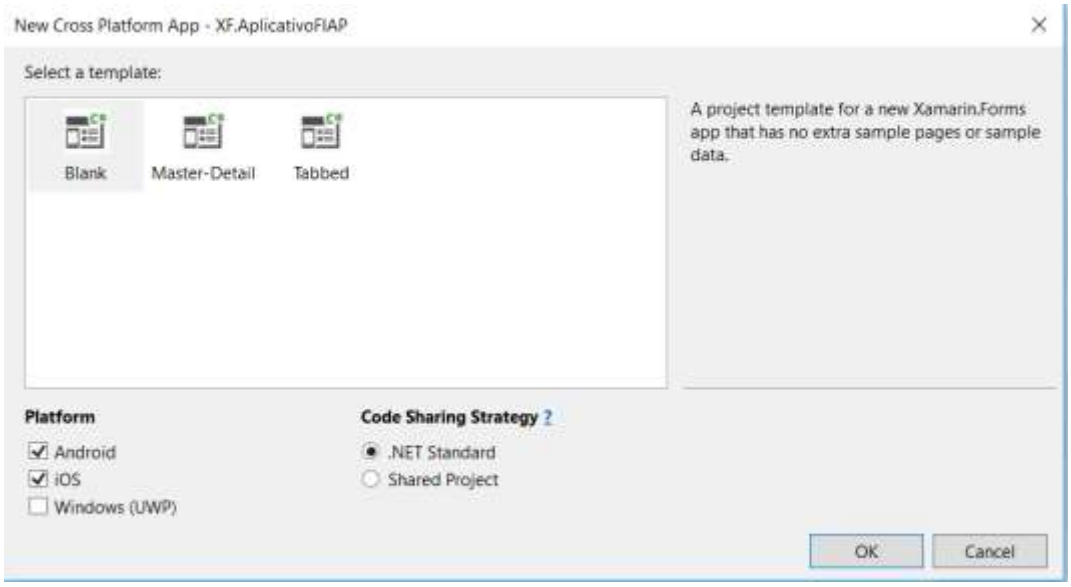


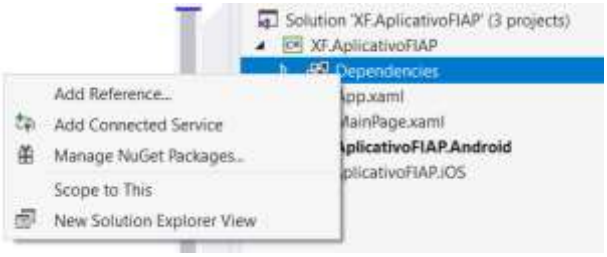
Figure 2. Janela Novo Projeto – Selecione o tipo de projeto.

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.

2.2. Passo 2 – Instalando o componente Newtonsoft.Json

Para instalar o componente Newtonsoft.Json em projetos PCL, clique com o botão direito do mouse em *Dependencies* e escolha a opção **Manage NuGet Packages**.



Na janela de gestão do NuGet, clique na guia Browse e pesquise pelo componente “Newtonsoft.Json”. Antes de instalar, se certifique que a versão selecionada é a mesma apresentada na Figura 3.

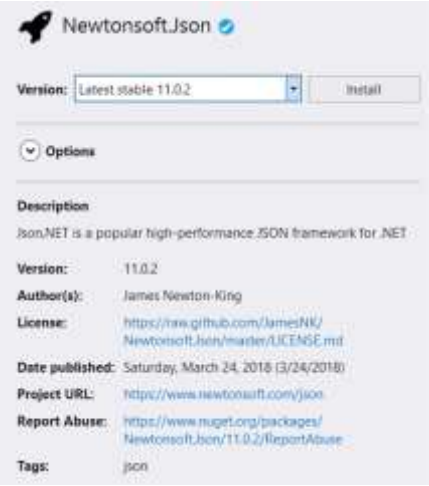


Figure 3. Newtonsoft.Json criada por James Newton-King.

NOTA: existem várias versões similares de parse JSON e XML disponíveis no NuGet, logo, certifique-se que instalou a versão mencionada.

2.3. Passo 3 – Criando o objeto Persistência (Patterns Repository)

Crie uma nova pasta no projeto 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 Professor. Clique com o botão direito do mouse sobre a basta Model e crie novo objeto do tipo Class com o nome ProfessorRepository.cs, **Add**, **New Item**, **Code** e selecione o

template Class. Implemente a classe de persistência de acordo com o código-fonte localizado na Tabela 1.

Tabela 1. Instruções C# da classe ProfessorRepository.cs

namespace XF.AplicativoFIAP.Model { public class Professor { public int Id { get; set; } public string Nome { get; set; } public string Titulo { get; set; } } public static class ProfessorRepository { private static List<Professor> professoresSqlAzure; public static async Task<List<Professor>> GetProfessoresSqlAzureAsync() { if (professoresSqlAzure != null) return professoresSqlAzure; var httpRequest = new HttpClient(); var stream = await httpRequest.GetStreamAsync("http://apiaplicativofiap.azurewebsites.net/ api/professors"); var professorSerializer = new DataContractJsonSerializer(typeof(List<Profe ssor>)); professoresSqlAzure = (List<Professor>)professorSerializer.ReadObj ect(stream); return professoresSqlAzure; } public static async Task<bool> PostProfessorSqlAzureAsync(Professor profAdd) { if (profAdd == null) return false; var httpRequest = new HttpClient(); httpRequest.BaseAddress = new Uri("http://apiaplicativofiap.azurewebsites. net/"); httpRequest.DefaultRequestHeaders.Accept.Cle ar();	- Classe concreta referente da entidade professor, a qual será o objeto de transferência. - Classe ProfessorRepository, objeto que contém todos os métodos para persistência do banco de dados por meio do consumo de API (Web Services). - Método GetProfessoresSqlAzureAsync, implementa as instruções de recuperação da tabela Professor. Para isso, utilizamos a classe HttpClient para realizar a requisição no endereço (URI) do serviço e quando recebemos a resposta em formato Json, utilizamos o DataContractJsonSerializer para serialização da entidade (lista de professores). - Para realização de inclusão, implementamos o método PostProfessorSqlAzureAsync(Professor). Da mesma forma que foi implementado no método de recuperação, por meio do HttpClient efetuamos a requisição no endereço (URI) do serviço, em seguida, usamos a classe JsonConvert do componente Newton.Soft para realizar a serialização da entidade (json) enviada como resposta da requisição. Com isso, convertemos a resposta de texto (json) em uma instância da classe complexa (professor). - Por fim, o método DeleteProfessorSqlAzureAsync(string),

`` httpRequest.DefaultRequestHeaders.Accept.Add (new System.Net.Http.Headers.MediaTypeWithQuality HeaderValue("application/json")); string profJson = Newtonsoft.Json.JsonConvert.SerializeObject(profAdd); var response = await httpRequest.PostAsync("api/professors", new StringContent(profJson, System.Text.Encoding.UTF8, "application/json")); if (response.IsSuccessStatusCode) return true; return false; } public static async Task<bool> DeleteProfessorSqlAzureAsync(string profId) { if (string.IsNullOrWhiteSpace(profId)) return false; var httpRequest = new HttpClient(); httpRequest.BaseAddress = new Uri("http://apiaplicativofiap.azurewebsites. net/"); httpRequest.DefaultRequestHeaders.Accept.Cle ar(); httpRequest.DefaultRequestHeaders.Accept.Add (new System.Net.Http.Headers.MediaTypeWithQuality HeaderValue("application/json")); var response = await httpRequest.DeleteAsync(string.Format("api/p rofessors/{0}", profId)); if (response.IsSuccessStatusCode) return true; return false; } } } ```	implementa a chamada do serviço (API) através do HttpClient, passando como parâmetro QueryString o Id do Professor que deverá ser removido da base.

3. Desafio

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

1. Implemente as classes da camada View para página de Lista de Professores e Novo Professor;
2. Implemente a classe ViewModel e assegure que o projeto seja escrito de acordo com as orientações do padrão MVVM;
3. Implemente as funcionalidades de CRUD para essa entidade: Criar, Ler, Atualizar e Remover;
4. Implemente a funcionalidade de pesquisa na lista de Professores;