

Laboratório

Olá Xamarin!

Versão: 1.0.0
Abril de 2017

William S. Rodriguez
@williamsrodz



Introdução

O objetivo desse LAB é criar um aplicativo Xamarin.Forms e implementar autenticação com social login via Facebook e Azure Mobile Apps.

Objetivos

Neste LAB você vai integrar a sua aplicação Xamarin.Forms com Azure Mobile Services. As metas são:

1. Configurar o projeto;
2. Autenticar o usuário usando redes sociais;
3. Obtenha informações adicionais do usuário autenticado.

Requisitos

Para a realização deste laboratório é necessário o seguinte:

- Uma máquina de desenvolvimento com o sistema operacional Windows 10 e Visual Studio 2015 o 2017 Community, Professional ou Enterprise.
- Assinatura do Azure.

Tempo estimado para completar este laboratório: **60 minutos**.

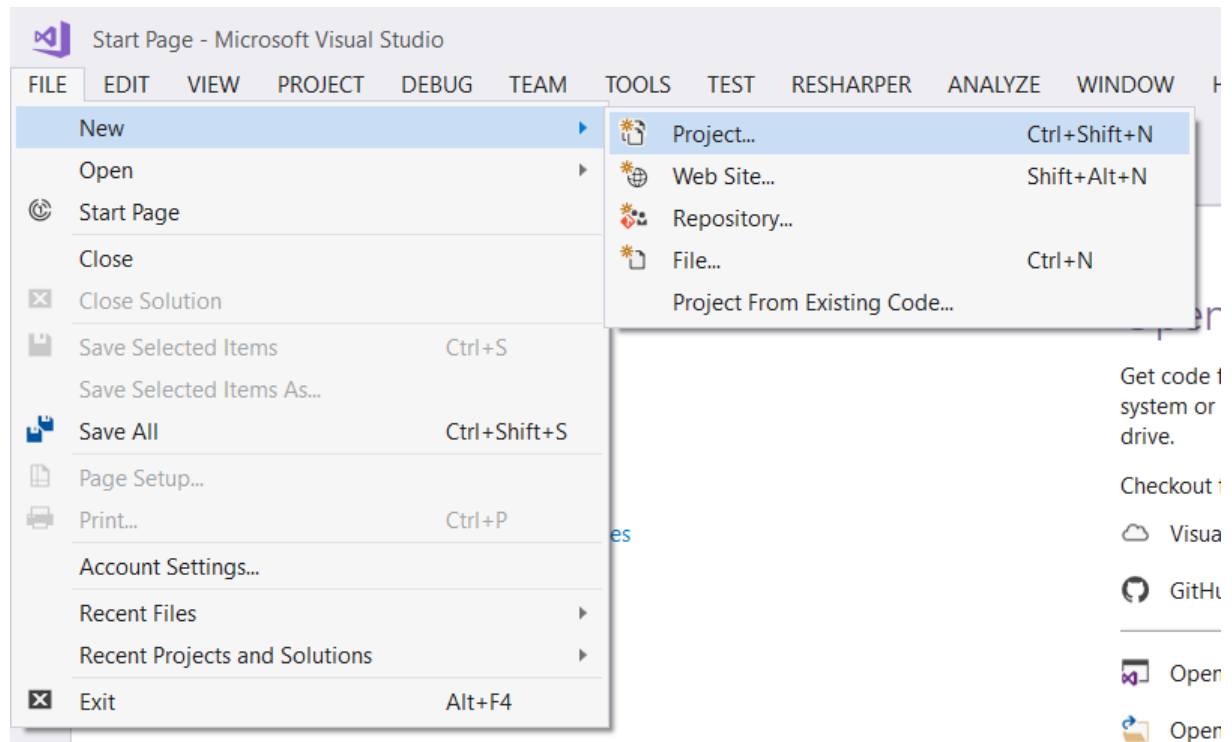
Exercício 1: Criando e Configurando o Projeto Xamarin.Forms

Para criar um projeto no Visual Studio que dê suporte ao MVVM para Android, iOS e Window, existe um template perfeito para você!

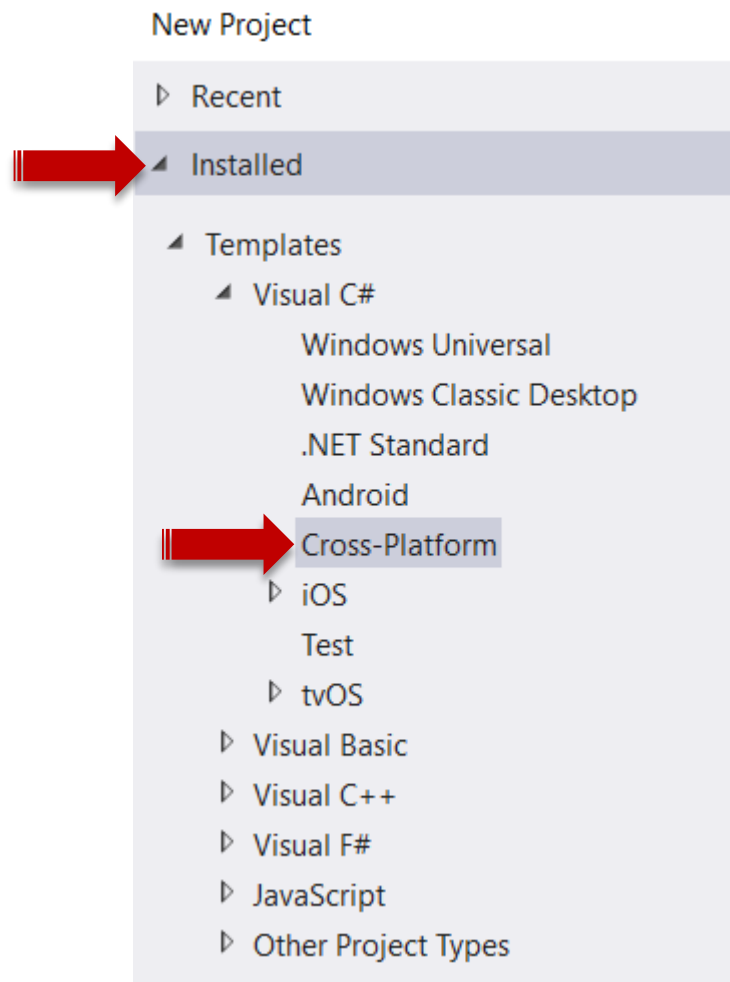
Tarefa 1. Criar o projeto Xamarin.Forms.

Execute os seguintes passos para criar um aplicativo Xamarin.Forms a partir do Visual Studio.

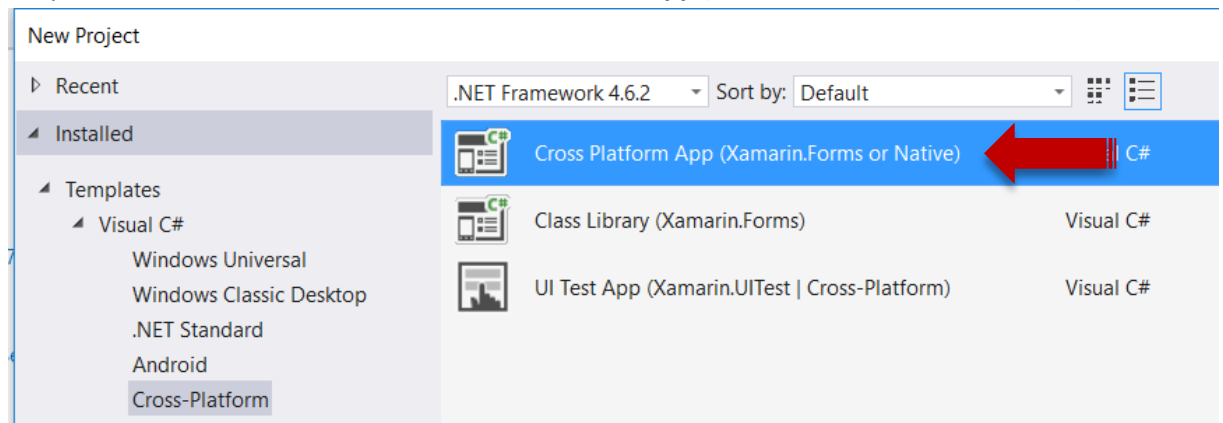
1. Selecione **File > New > Project** no Visual Studio.



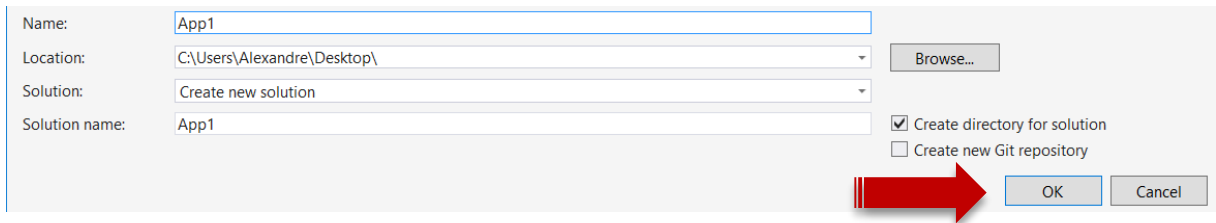
2. No painel esquerdo da janela **New Project** selecione **Visual C# > Cross-Platform** para indicar que deseja criar um aplicativo para multi-plataforma.



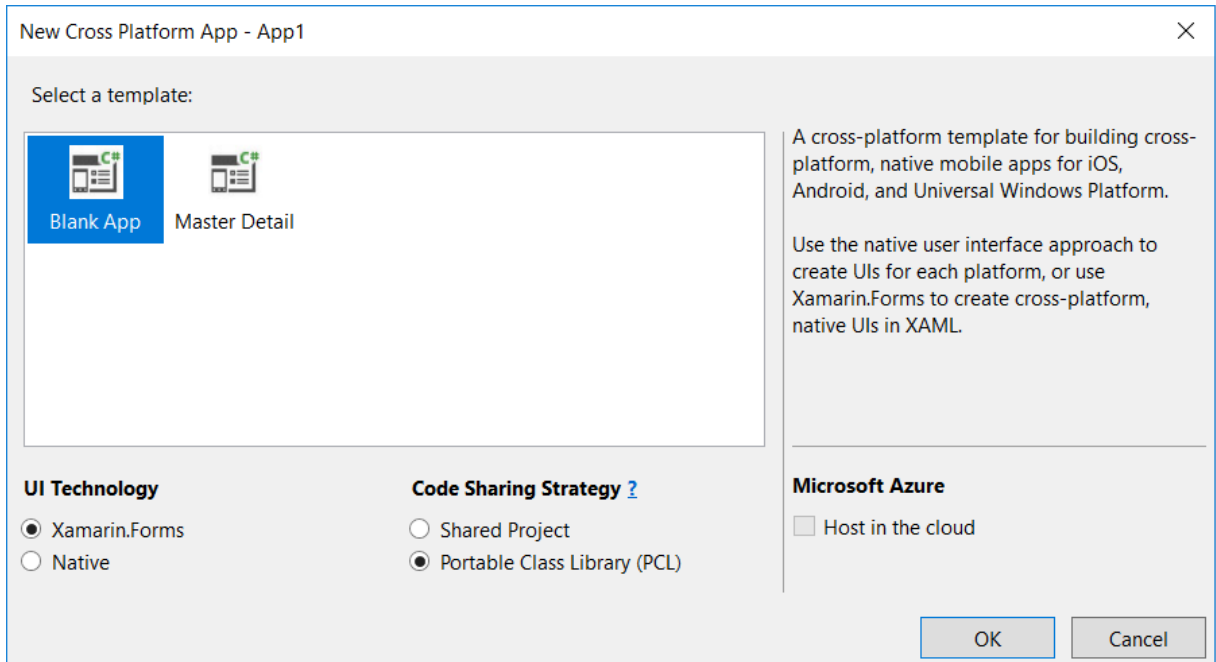
3. No painel direito, selecione o modelo **Cross Platform App (Xamarin.Forms or Native2019)**.



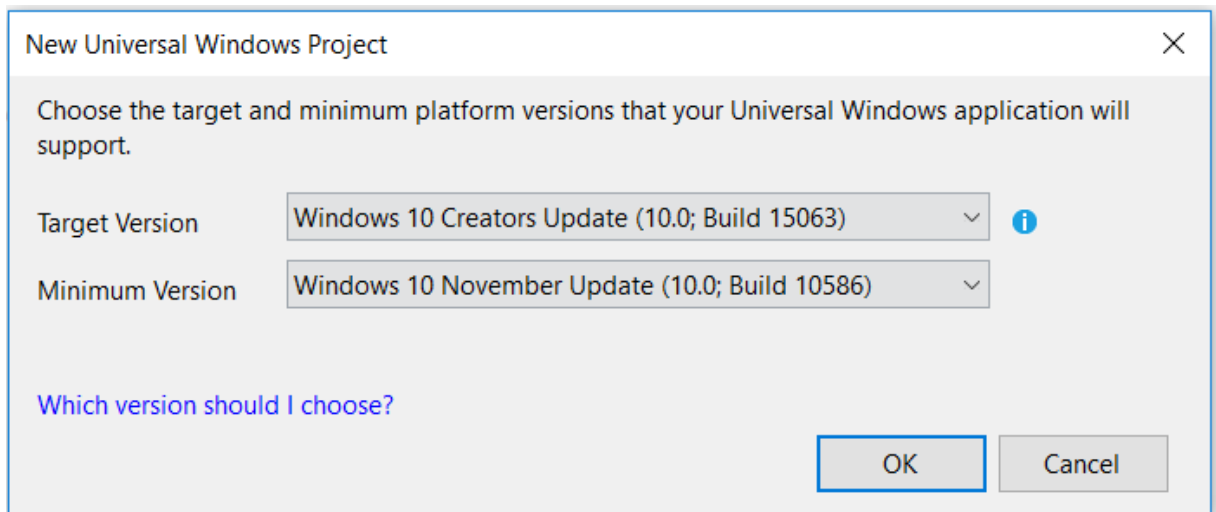
4. Forneça o nome, local e clique em **OK** para criar o projeto.



5. Selecione o tipo de **UI Technology** como **Xamarin.Forms** e a estratégia de compartilhamento de código(**Code Sharing Strategy**) como **Portable Class Library (PCL)**:



6. Se o seu projeto pedir uma versão do UWP, apenas confirme com a já pré-selecionada

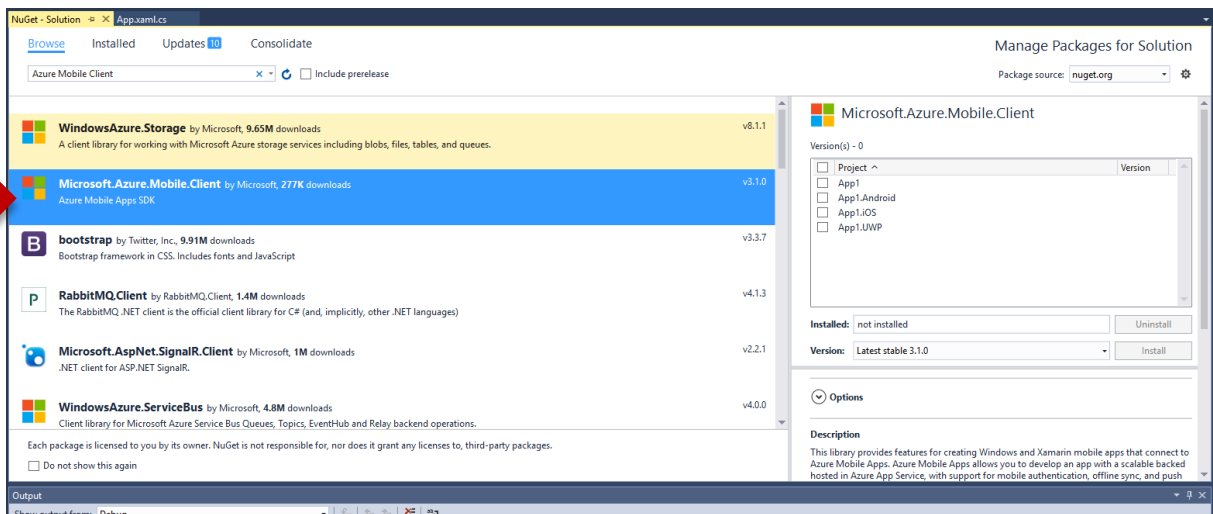


7. Aguarde a criação do projeto, ignorando qualquer instrução de conectar com o Mac.

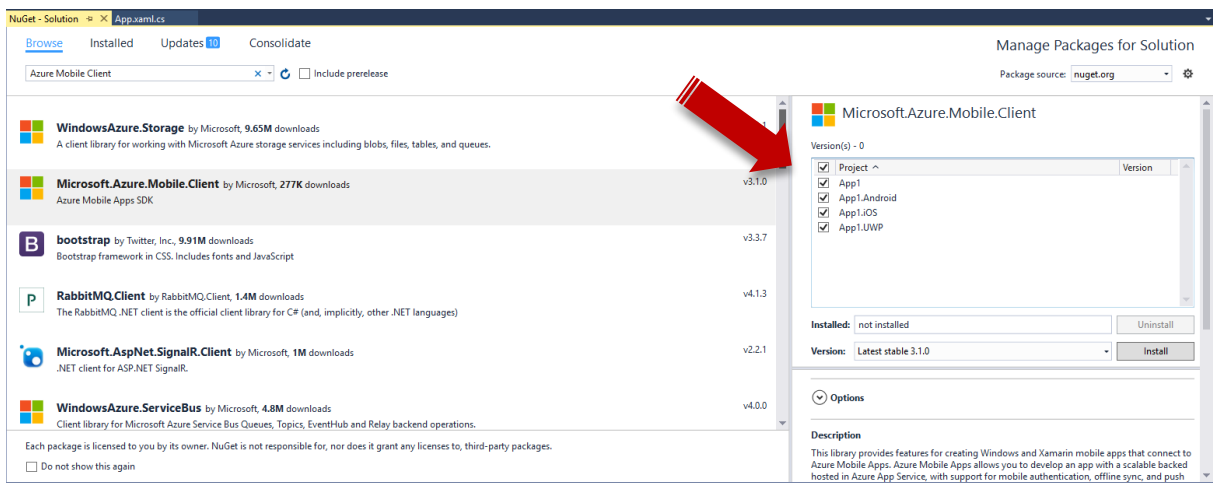
Tarefa 2. Incluir dependências do Azure Mobile Apps no seu projeto Xamarin.Forms

Execute os seguintes passos para adicionar as dependências do Azure Mobile apps no seu projeto.

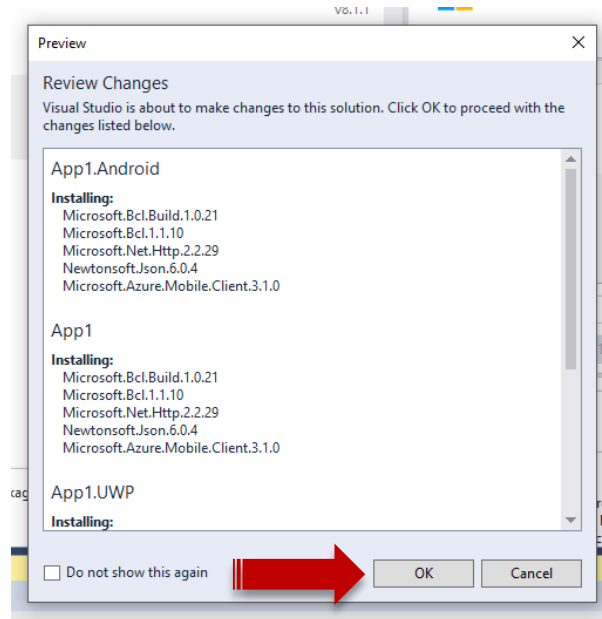
1. Clique com o botão direito na sua **Solution**, escolha a opção **Manage NuGet Packages for Solution**, clique na guia **Browse**, no campo **Search**, digite **Azure Mobile Client**, pressione **Enter** e localize o pacote **Microsoft.Azure.Client**.



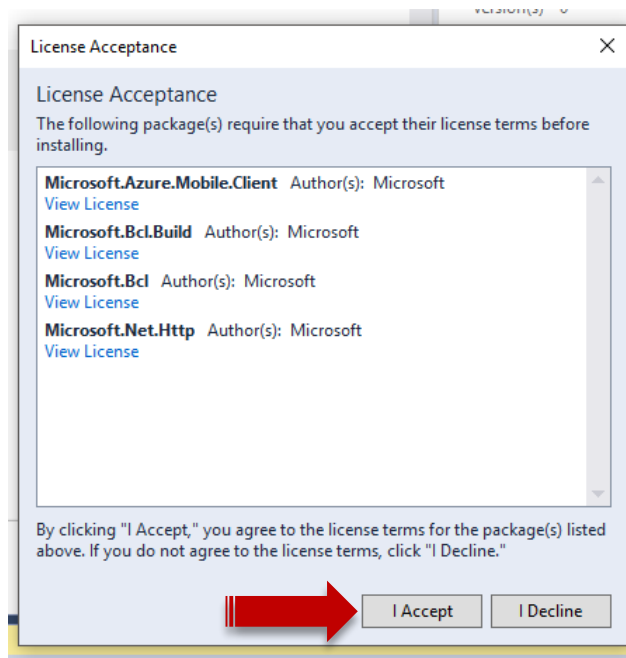
2. Na janela de detalhes do pacote, selecione todos os projetos e clique em **Install**. Isso vai iniciar o processo de download e adição das dependências do **Azure Mobile** nos projetos da sua **Solution**.



3. Na tela de revisão de alterações clique em **OK**.

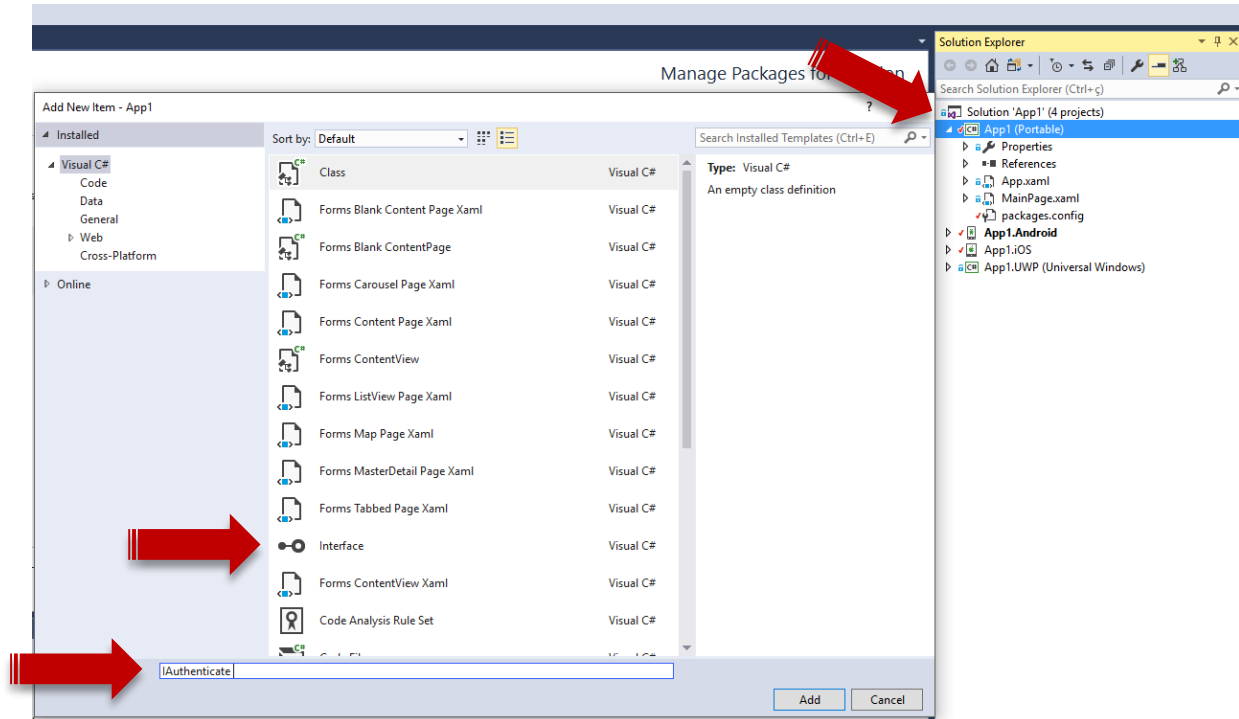


4. Clique em **I Accept** para aceitar as licenças.



Tarefa 3. Implemente a interface **IAuthenticate** para cada plataforma compatível com seu aplicativo.

1. Em seu projeto PCL, adicione uma interface chamada **IAuthenticate**.

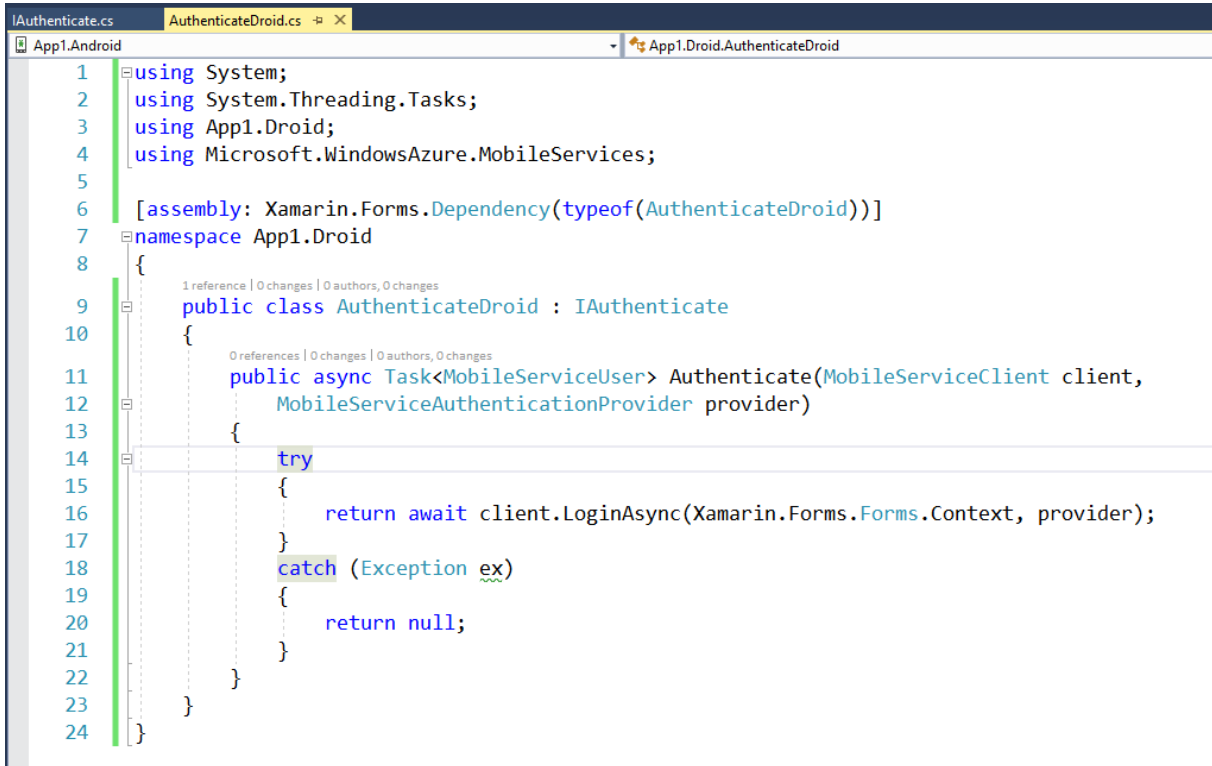


2. Adicione as seguintes instruções na interface **IAuthenticate**



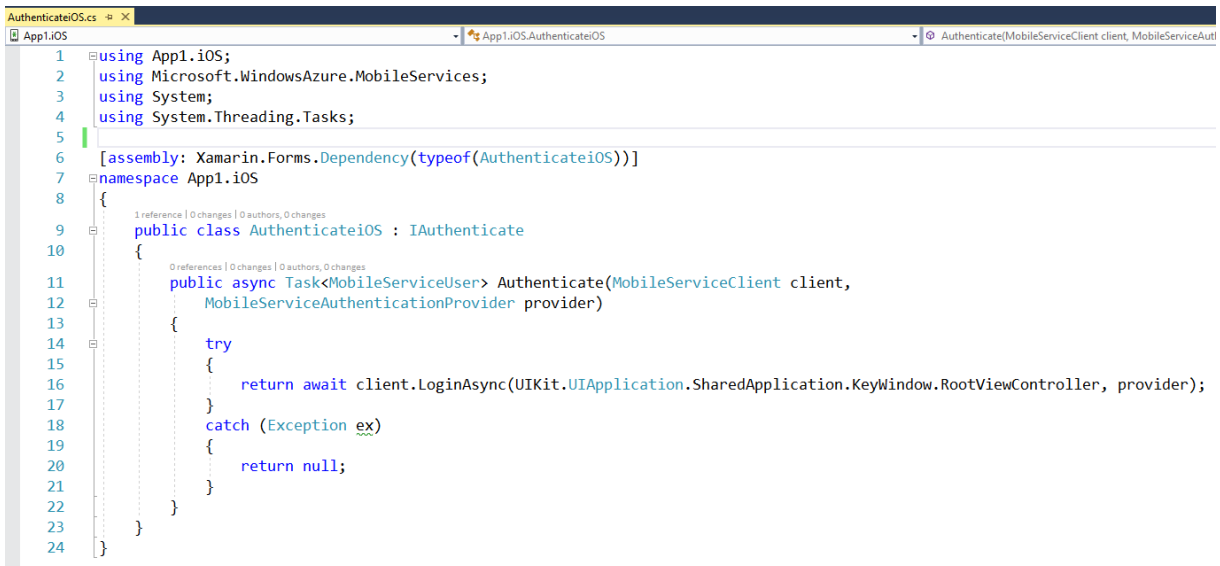
Agora vamos implementar os métodos de autenticação específicos de cada plataforma, herdando da interface **IAuthenticate**.

3. No projeto **Android** adicione uma nova classe chamada **AuthenticateDroid.cs** implementando o seguinte método:



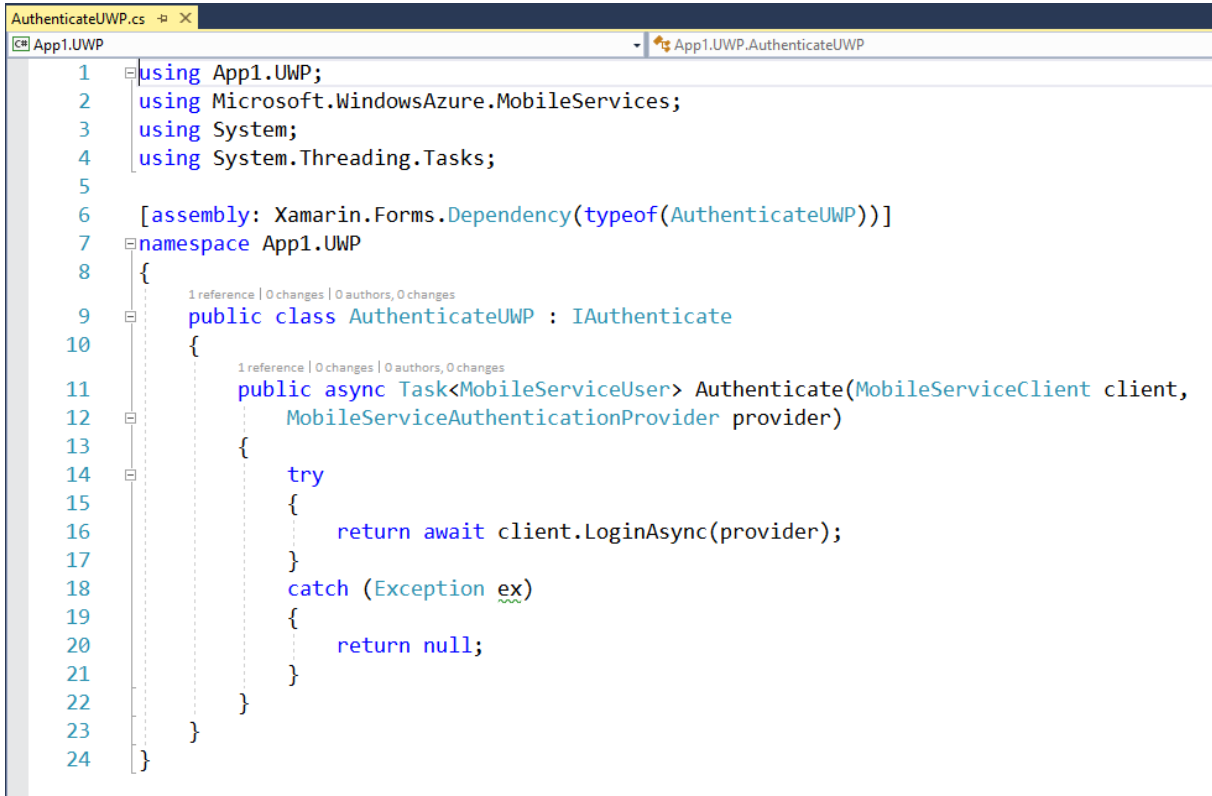
```
1 using System;
2 using System.Threading.Tasks;
3 using App1.Droid;
4 using Microsoft.WindowsAzure.MobileServices;
5
6 [assembly: Xamarin.Forms.Dependency(typeof(AuthenticateDroid))]
7 namespace App1.Droid
8 {
9     public class AuthenticateDroid : IAuthenticate
10     {
11         public async Task<MobileServiceUser> Authenticate(MobileServiceClient client,
12             MobileServiceAuthenticationProvider provider)
13         {
14             try
15             {
16                 return await client.LoginAsync(Xamarin.Forms.Forms.Context, provider);
17             }
18             catch (Exception ex)
19             {
20                 return null;
21             }
22         }
23     }
24 }
```

4. No projeto **iOS** adicione uma nova classe chamada **AuthenticateiOS.cs** implementando o seguinte método:



```
1 using App1.iOS;
2 using Microsoft.WindowsAzure.MobileServices;
3 using System;
4 using System.Threading.Tasks;
5
6 [assembly: Xamarin.Forms.Dependency(typeof(AuthenticateiOS))]
7 namespace App1.iOS
8 {
9     public class AuthenticateiOS : IAuthenticate
10     {
11         public async Task<MobileServiceUser> Authenticate(MobileServiceClient client,
12             MobileServiceAuthenticationProvider provider)
13         {
14             try
15             {
16                 return await client.LoginAsync(UIKit.UIApplication.SharedApplication.KeyWindow.RootViewController, provider);
17             }
18             catch (Exception ex)
19             {
20                 return null;
21             }
22         }
23     }
24 }
```

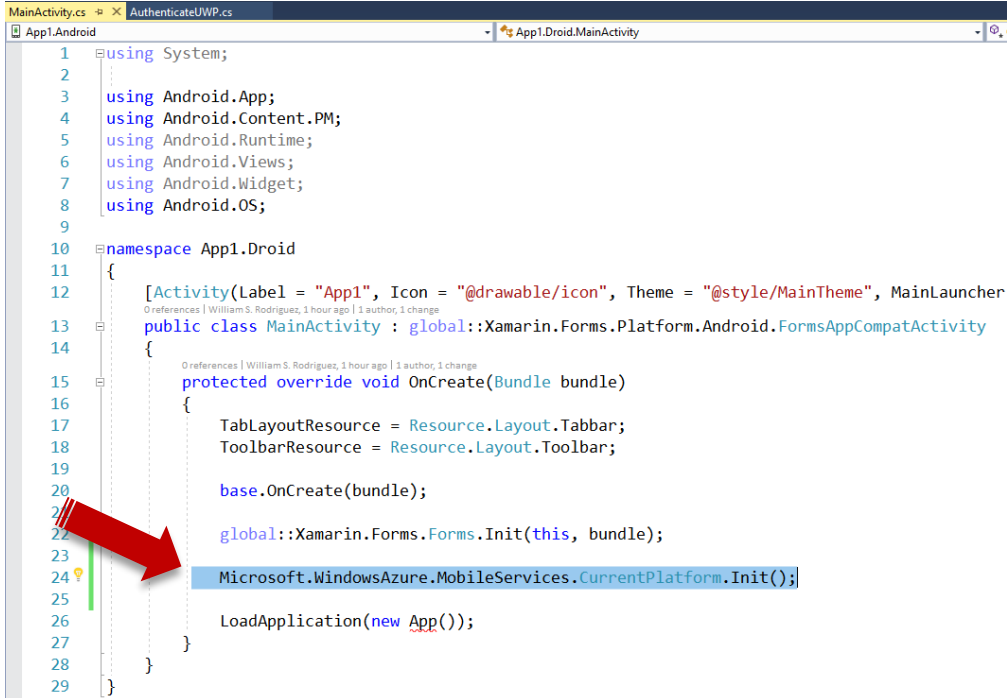
5. No projeto **UWP** adicione uma nova classe chamada **AuthenticateUWP.cs** implementando o seguinte método:



```
1 using App1.UWP;
2 using Microsoft.WindowsAzure.MobileServices;
3 using System;
4 using System.Threading.Tasks;
5
6 [assembly: Xamarin.Forms.Dependency(typeof(AuthenticateUWP))]
7 namespace App1.UWP
8 {
9     1 reference | 0 changes | 0 authors, 0 changes
10     public class AuthenticateUWP : IAuthenticate
11     {
12         1 reference | 0 changes | 0 authors, 0 changes
13         public async Task<MobileServiceUser> Authenticate(MobileServiceClient client,
14             MobileServiceAuthenticationProvider provider)
15         {
16             try
17             {
18                 return await client.LoginAsync(provider);
19             }
20             catch (Exception ex)
21             {
22                 return null;
23             }
24         }
25     }
26 }
```

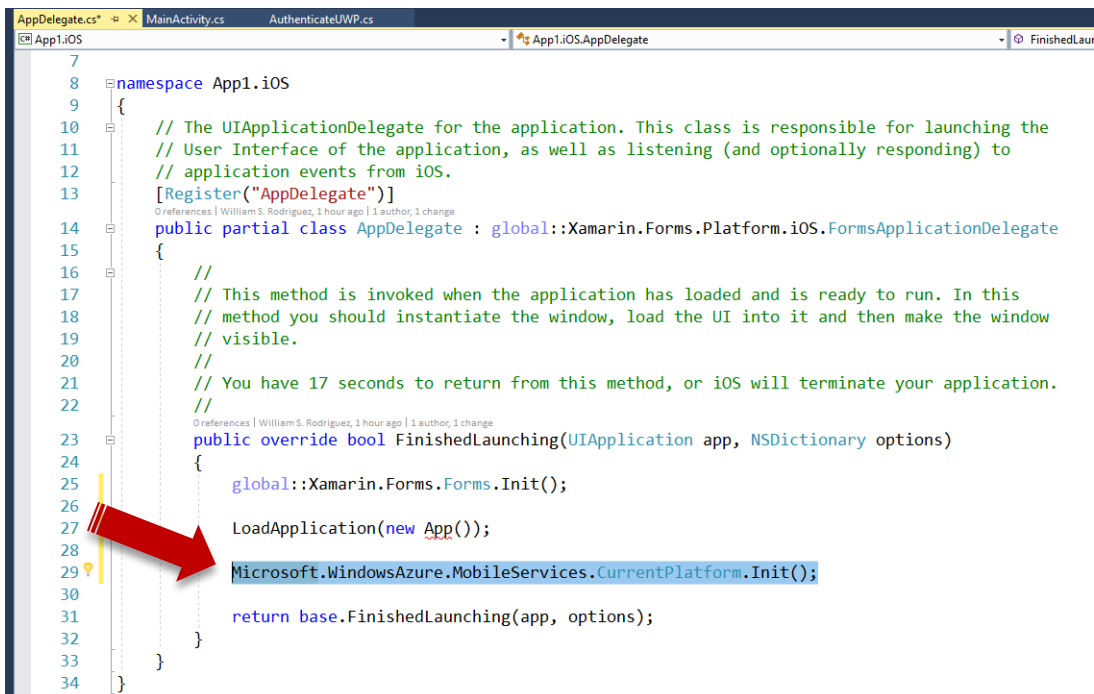
Agora você precisa inicializar a SDK do Azure nas plataformas Android e iOS.

6. No aplicativo Android abra o arquivo **MainActivity.cs** e no método **OnCreate** adicione a seguinte instrução:



```
1 using System;
2
3 using Android.App;
4 using Android.Content.PM;
5 using Android.Runtime;
6 using Android.Views;
7 using Android.Widget;
8 using Android.OS;
9
10 namespace App1.Droid
11 {
12     [Activity(Label = "App1", Icon = "@drawable/icon", Theme = "@style/MainTheme", MainLauncher
13     public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
14     {
15         protected override void OnCreate(Bundle bundle)
16         {
17             TabLayoutResource = Resource.Layout.Tabbar;
18             ToolbarResource = Resource.Layout.Toolbar;
19
20             base.OnCreate(bundle);
21
22             global::Xamarin.Forms.Forms.Init(this, bundle);
23
24             Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();
25
26             LoadApplication(new App());
27         }
28     }
29 }
```

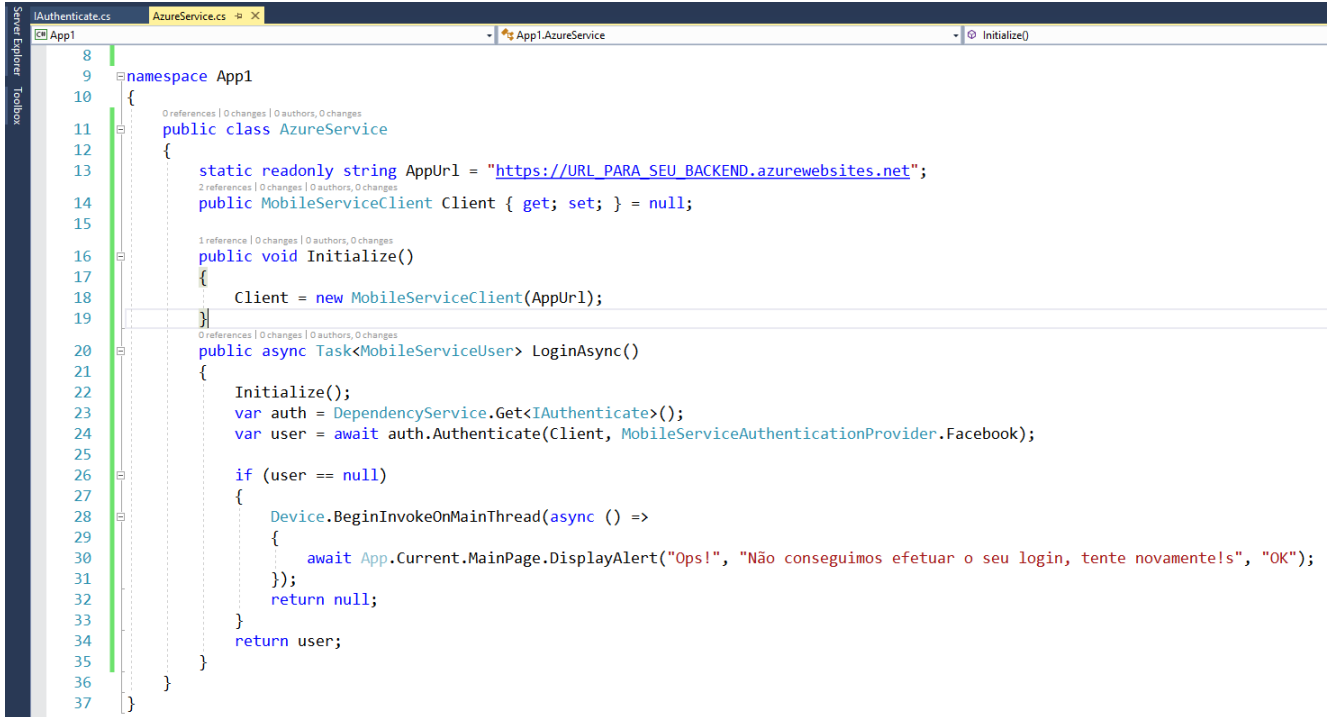
7. No aplicativo iOS abra o arquivo **AppDelegate.cs** e no método **FinishedLaunching** adicione a seguinte instrução:



```
7
8 namespace App1.iOS
9 {
10     // The UIApplicationDelegate for the application. This class is responsible for launching the
11     // User Interface of the application, as well as listening (and optionally responding) to
12     // application events from iOS.
13     [Register("AppDelegate")]
14     public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
15     {
16         //
17         // This method is invoked when the application has loaded and is ready to run. In this
18         // method you should instantiate the window, load the UI into it and then make the window
19         // visible.
20         //
21         // You have 17 seconds to return from this method, or iOS will terminate your application.
22         //
23         public override bool FinishedLaunching(UIApplication app, NSDictionary options)
24         {
25             global::Xamarin.Forms.Forms.Init();
26
27             LoadApplication(new App());
28
29             Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();
30
31             return base.FinishedLaunching(app, options);
32         }
33     }
34 }
```

Tarefa 4. Implemente a classe de autenticação no seu projeto Portable.

1. Adicione uma classe chamada **AzureService** e implemente os seguintes métodos:



```
1  namespace App1
2  {
3      static readonly string AppUrl = "https://URL_PARA_SEU_BACKEND.azurewebsites.net";
4      public MobileServiceClient Client { get; set; } = null;
5
6      public void Initialize()
7      {
8          Client = new MobileServiceClient(AppUrl);
9      }
10
11     public async Task<MobileServiceUser> LoginAsync()
12     {
13         Initialize();
14         var auth = DependencyService.Get<IAAuthenticate>();
15         var user = await auth.Authenticate(Client, MobileServiceAuthenticationProvider.Facebook);
16
17         if (user == null)
18         {
19             Device.BeginInvokeOnMainThread(async () =>
20             {
21                 await App.Current.MainPage.DisplayAlert("Ops!", "Não conseguimos efetuar o seu login, tente novamente!", "OK");
22             });
23             return null;
24         }
25         return user;
26     }
27 }
```

Exercício 2: Adicionando controles de autenticação no seu Projeto Xamarin.Forms

Tarefa 1. Inclua um botão e um label na sua página MainPage.

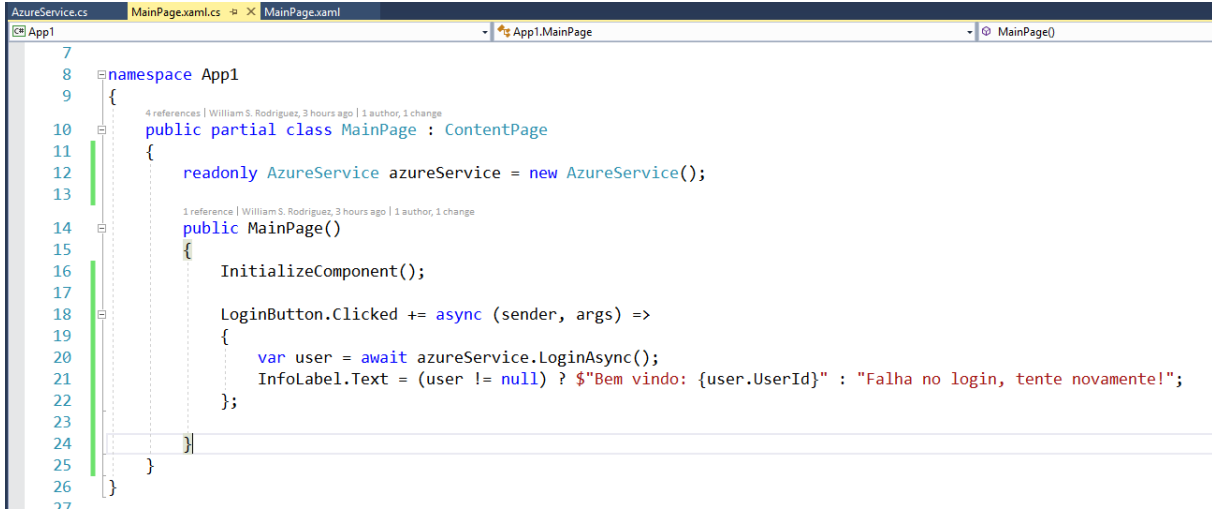
1. Abra o arquivo **MainPage.xaml** e altere o layout adicionando os seguintes controles:



```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             xmlns:local="clr-namespace:App1"
5             x:Class="App1.MainPage">
6      <StackLayout
7          HorizontalOptions="Center"
8          VerticalOptions="CenterAndExpand">
9          <Button Text="Logar com Facebook"
10               BackgroundColor="Blue"
11               FontSize="Medium"
12               TextColor="White"
13               x:Name="LoginButton"></Button>
14
15          <Label TextColor="Black"
16               FontSize="Medium"
17               x:Name="InfoLabel" />
18      </StackLayout>
19 </ContentPage>
```

Tarefa 2. Inclua os eventos de click no seu botão.

1. Abra o arquivo **MainPage.cs** e inclua os seguintes métodos:

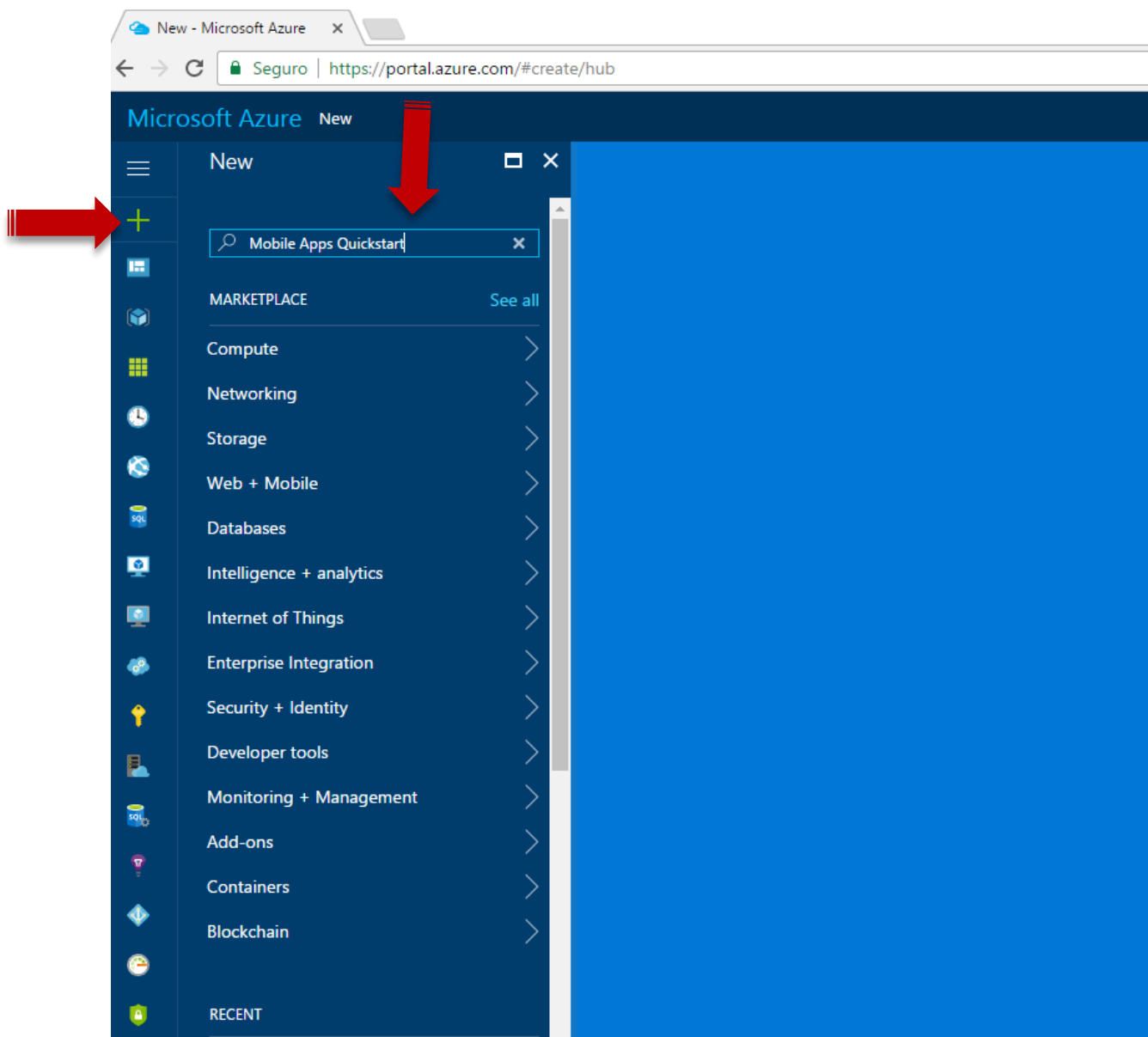


```
7
8 namespace App1
9 {
10     4 references | William S. Rodriguez, 3 hours ago | 1 author, 1 change
11     public partial class MainPage : ContentPage
12     {
13         readonly AzureService azureService = new AzureService();
14
15         1 reference | William S. Rodriguez, 3 hours ago | 1 author, 1 change
16         public MainPage()
17         {
18             InitializeComponent();
19
20             LoginButton.Clicked += async (sender, args) =>
21             {
22                 var user = await azureService.LoginAsync();
23                 InfoLabel.Text = (user != null) ? $"Bem vindo: {user.UserId}" : "Falha no login, tente novamente!";
24             };
25         }
26     }
27 }
```

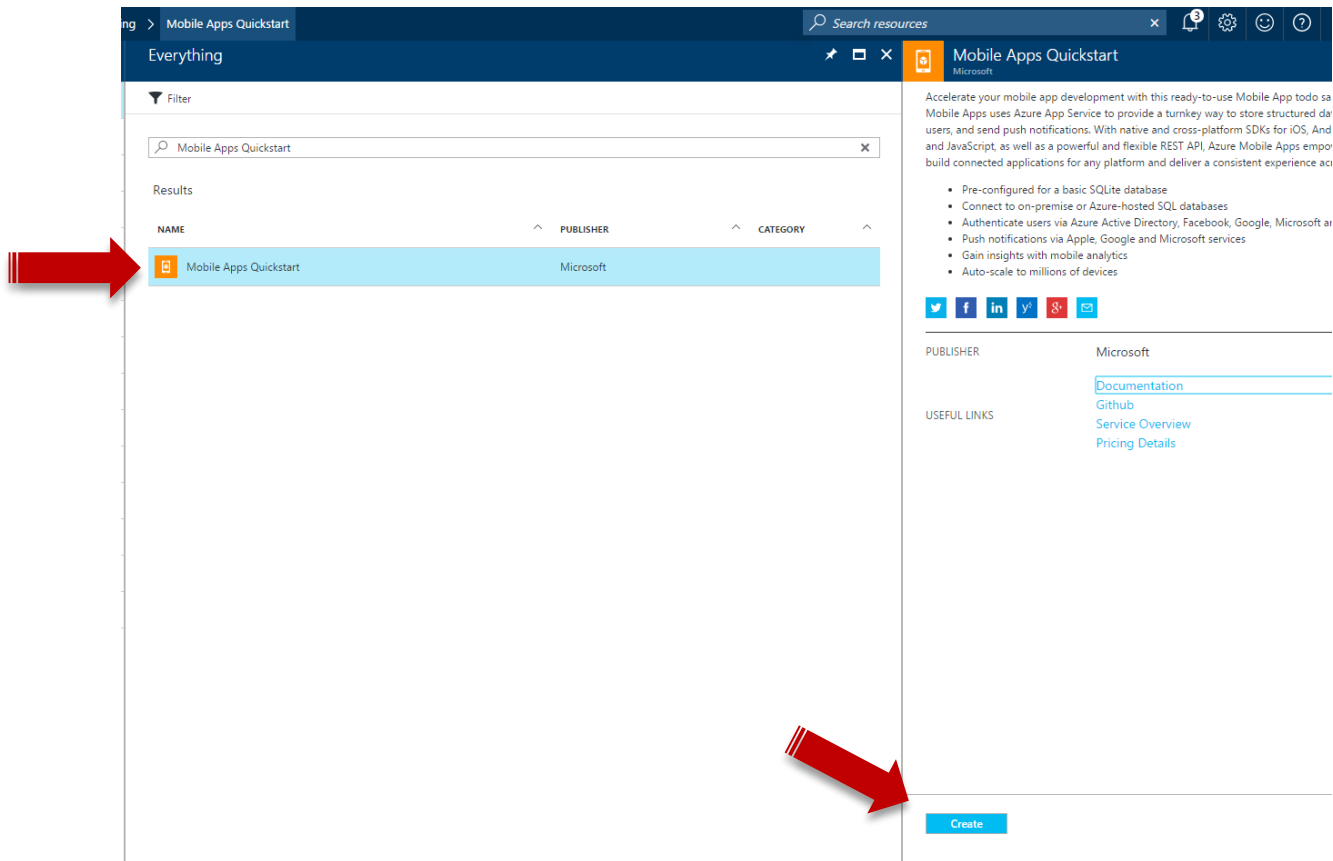
Exercício 3: Configurando o seu backend no Azure Mobile App

Tarefa 1. Crie um novo backend Mobile App no

1. Acesse **portal.azure.com**, clique no menu **+ New**, procure por **Mobile Apps Quickstart**, e pressione **Enter**.

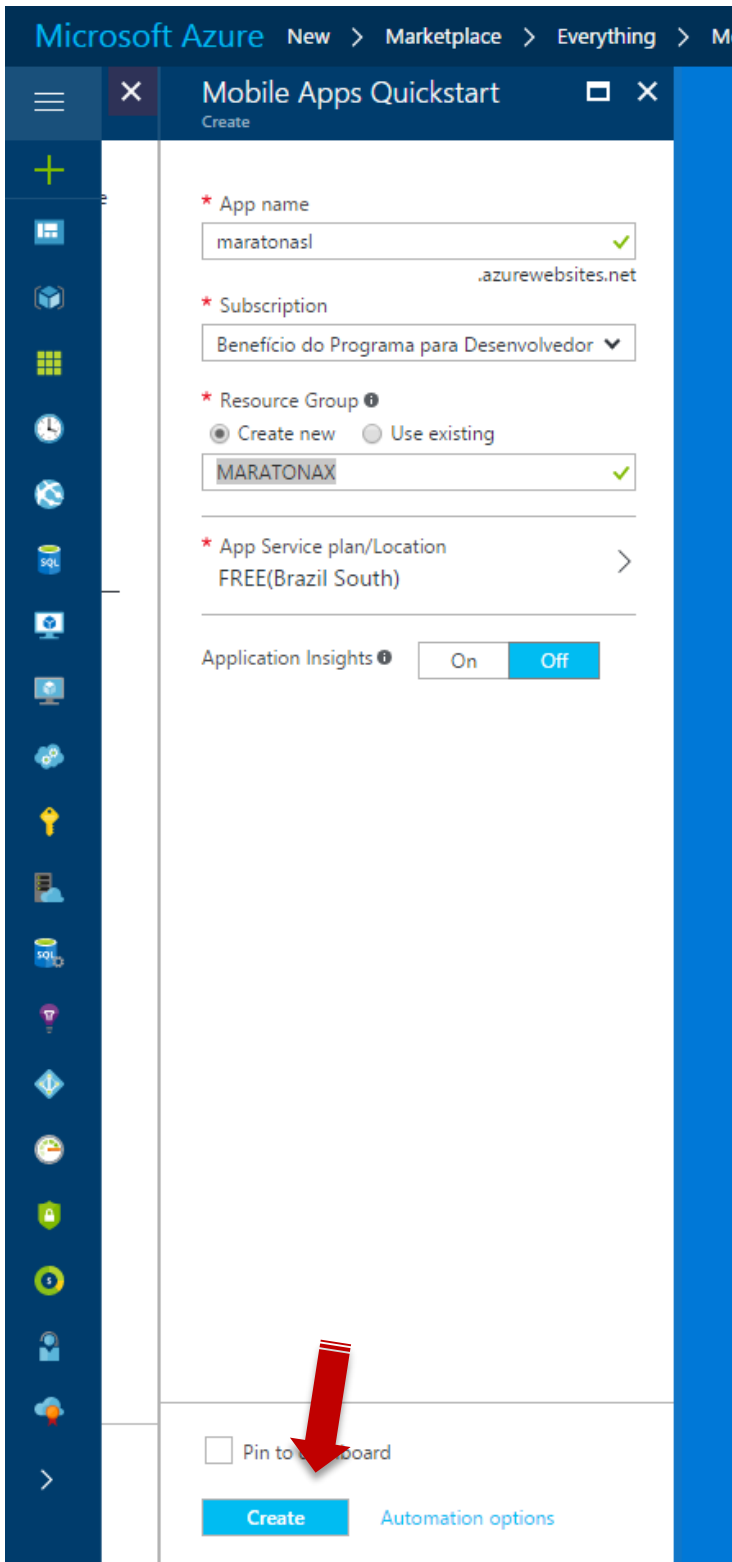


- Na blade posterior, selecione o template **Mobile Apps Quickstart** e clique no botão **Create**, para criar o seu backend.



- Na tela de criação, preencha as informações nos campos da seguinte forma:
 - App name: **Nome do seu aplicativo**
 - Subscription: **Escolha a assinatura**
 - Resource Group: Digite **MARATONAX** e escolha a opção **Create new**
 - App Service plan/Location: Mantenha a opção padrão ou crie um novo plano.

Após isso clique em **Create**.



Microsoft Azure New > Marketplace > Everything > Mo

Mobile Apps Quickstart
Create

* App name
maratonas ✓
.azurewebsites.net

* Subscription
Benefício do Programa para Desenvolvedor ▼

* Resource Group ⓘ
☒ Create new ☐ Use existing
MARATONAX ✓

* App Service plan/Location >
FREE(Brazil South)

Application Insights ⓘ

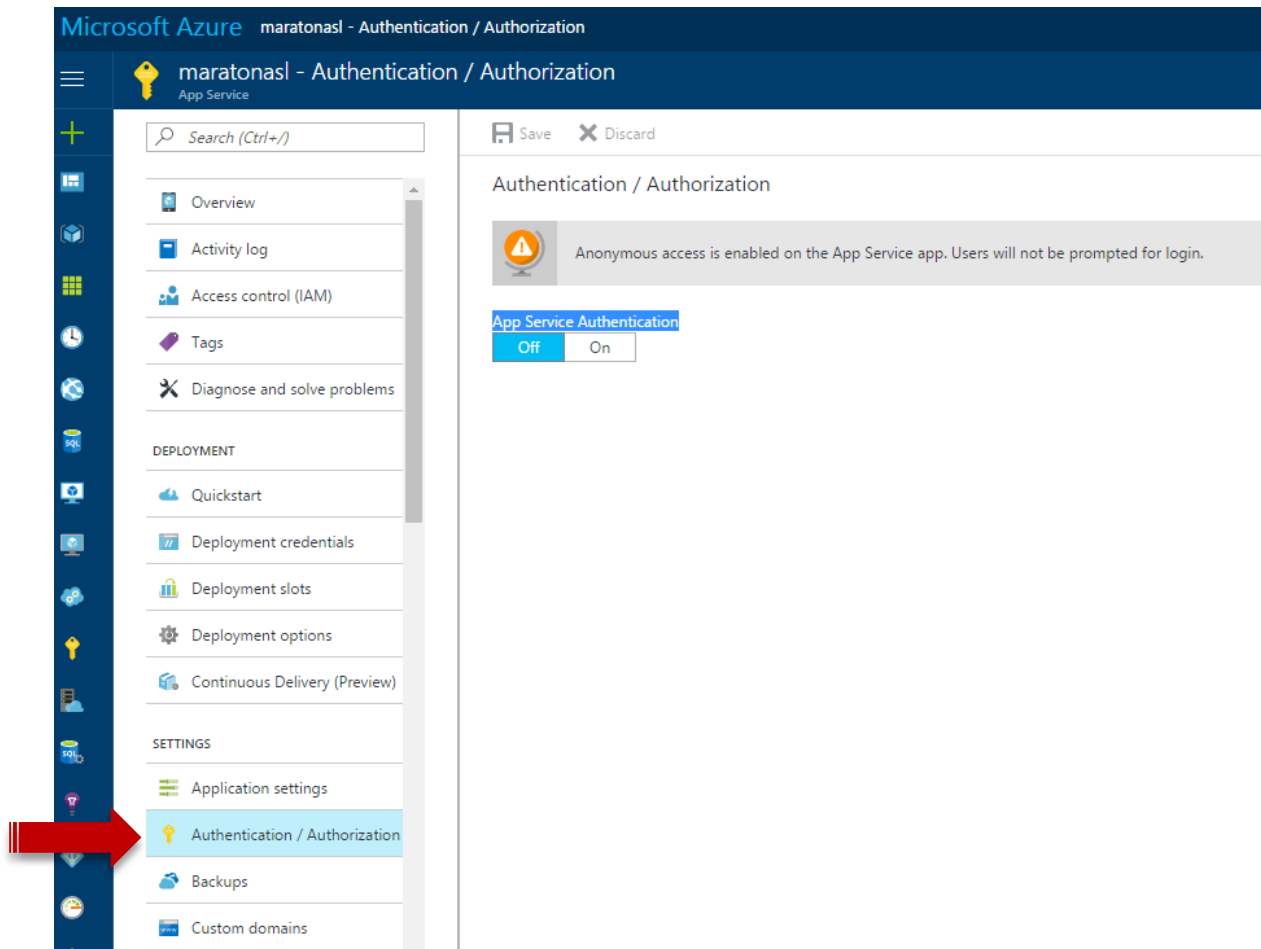
☐ Pin to dashboard

[Automation options](#)

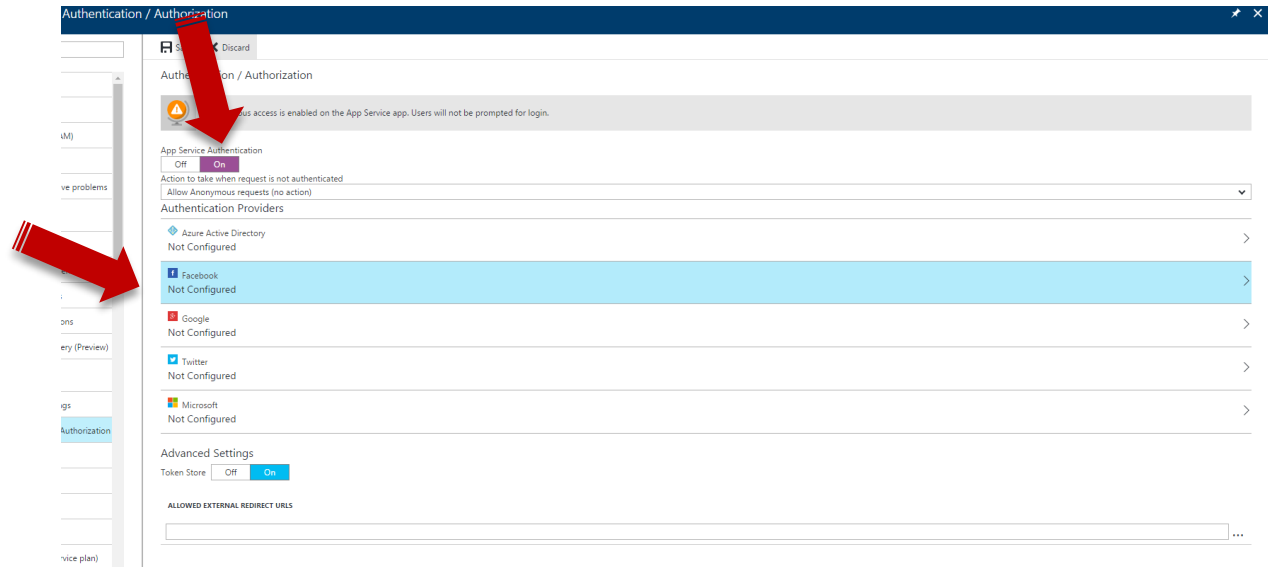
Aguarde o seu backend ser criado e configurado.

Tarefa 2. Configure a autenticação do Facebook no seu backend

1. Com o seu backend criado selecione a opção **Authentication / Authorization**.

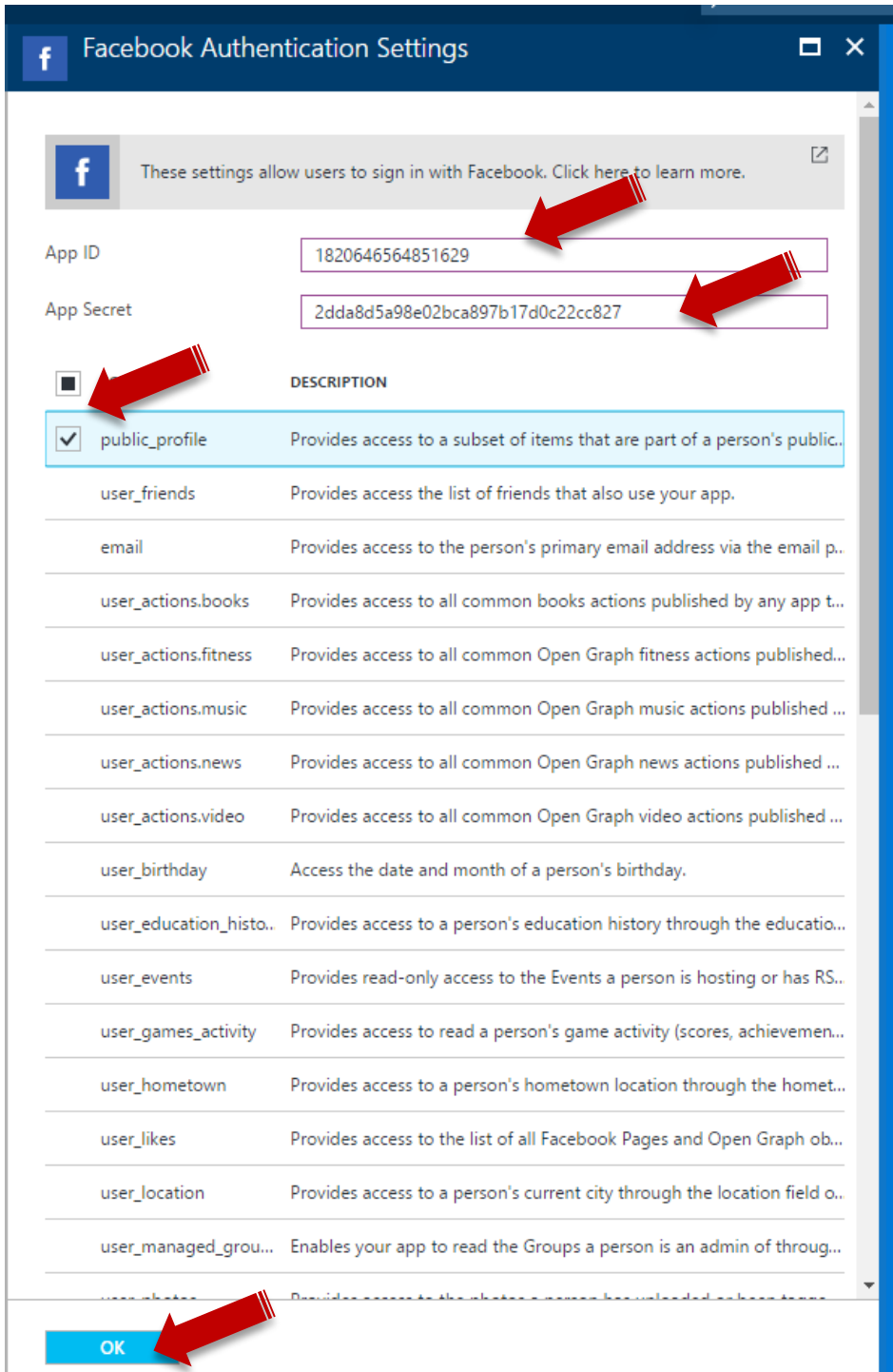


2. Ative a opção App Service **Authentication** e selecione a Opção Facebook



3. Siga o guia de configuração de aplicativo no Facebook disponível em <https://docs.microsoft.com/pt-br/azure/app-service-mobile/app-service-mobile-how-to-configure-facebook-authentication>

4. Informe o **App ID** e o **App Secret** obtidos após a configuração do aplicativo no Facebook, selecione **public_profile** e clique em **OK**.



Facebook Authentication Settings

These settings allow users to sign in with Facebook. Click here to learn more.

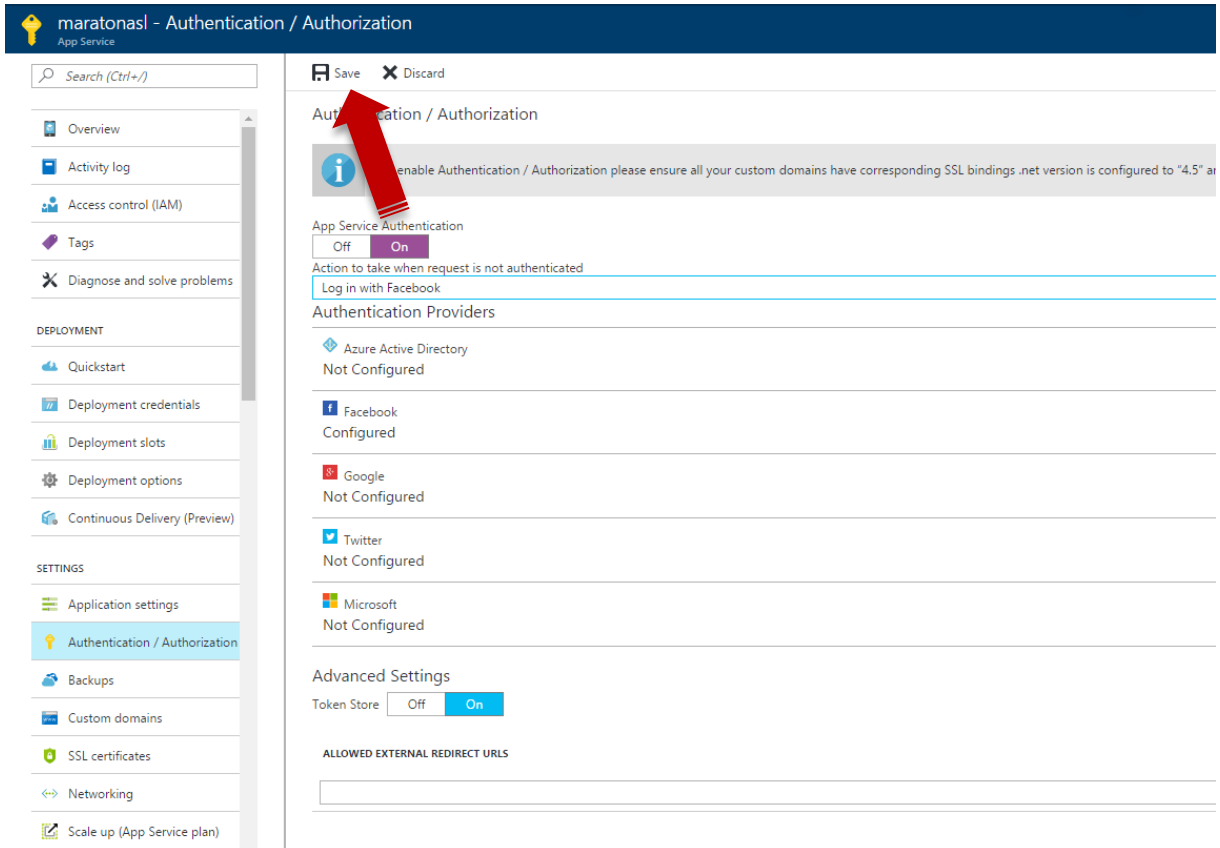
App ID: 1820646564851629

App Secret: 2dda8d5a98e02bca897b17d0c22cc827

	DESCRIPTION
<input checked="" type="checkbox"/> public_profile	Provides access to a subset of items that are part of a person's public...
<input type="checkbox"/> user_friends	Provides access the list of friends that also use your app.
<input type="checkbox"/> email	Provides access to the person's primary email address via the email p...
<input type="checkbox"/> user_actions.books	Provides access to all common books actions published by any app t...
<input type="checkbox"/> user_actions.fitness	Provides access to all common Open Graph fitness actions published...
<input type="checkbox"/> user_actions.music	Provides access to all common Open Graph music actions published ...
<input type="checkbox"/> user_actions.news	Provides access to all common Open Graph news actions published ...
<input type="checkbox"/> user_actions.video	Provides access to all common Open Graph video actions published ...
<input type="checkbox"/> user_birthday	Access the date and month of a person's birthday.
<input type="checkbox"/> user_education_histo..	Provides access to a person's education history through the educatio...
<input type="checkbox"/> user_events	Provides read-only access to the Events a person is hosting or has RS...
<input type="checkbox"/> user_games_activity	Provides access to read a person's game activity (scores, achievemen...
<input type="checkbox"/> user_hometown	Provides access to a person's hometown location through the homet...
<input type="checkbox"/> user_likes	Provides access to the list of all Facebook Pages and Open Graph ob...
<input type="checkbox"/> user_location	Provides access to a person's current city through the location field o...
<input type="checkbox"/> user_managed_grou...	Enables your app to read the Groups a person is an admin of throug...
<input type="checkbox"/> user_photos	Provides access to the photos a person has uploaded on their page...

OK

5. Após isso clique em **Save**, e pronto a autenticação está configurada.



maratonas! - Authentication / Authorization
App Service

Search (Ctrl+J)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

DEPLOYMENT

Quickstart
Deployment credentials
Deployment slots
Deployment options
Continuous Delivery (Preview)

SETTINGS

Application settings
Authentication / Authorization
Backups
Custom domains
SSL certificates
Networking
Scale up (App Service plan)

Save Discard

Authentication / Authorization

Enable Authentication / Authorization please ensure all your custom domains have corresponding SSL bindings .net version is configured to "4.5" and

App Service Authentication
Off On

Action to take when request is not authenticated
Log in with Facebook

Authentication Providers

Azure Active Directory	Not Configured
Facebook	Configured
Google	Not Configured
Twitter	Not Configured
Microsoft	Not Configured

Advanced Settings

Token Store Off On


ALLOWED EXTERNAL REDIRECT URLS

Exercício 4: Testando a autenticação social do Azure Mobile App

Tarefa 1. Altere o endereço do seu backend na classe AzureService.cs

1. Abra a classe **AzureService.cs** e altere o valor da variável **AppUrl** de para o endereço do seu backend recém criado e configurado.

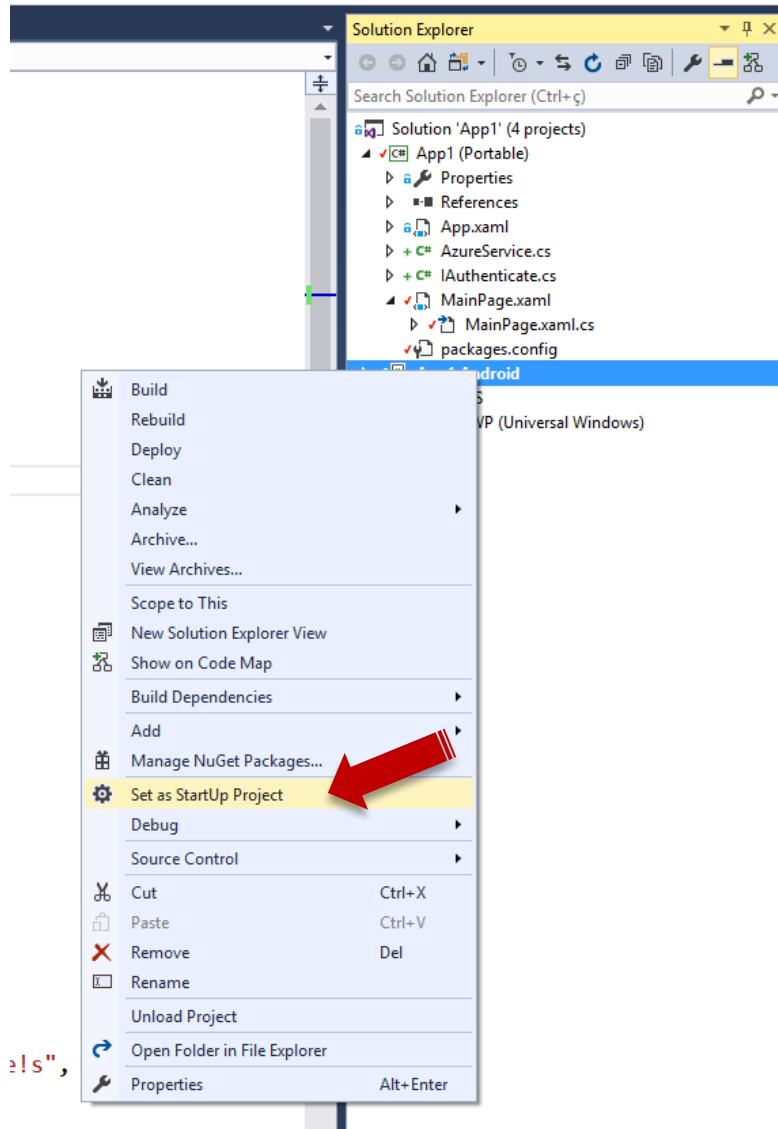
```
namespace App1
{
    2 references | 0 changes | 0 authors, 0 changes
    public class AzureService
    {
        static readonly string AppUrl = "https://URL PARA SEU BACKEND.azurewebsites.net";
        2 references | 0 changes | 0 authors, 0 changes
        public MobileServiceClient Client { get; set; } = null;
    }
}
```



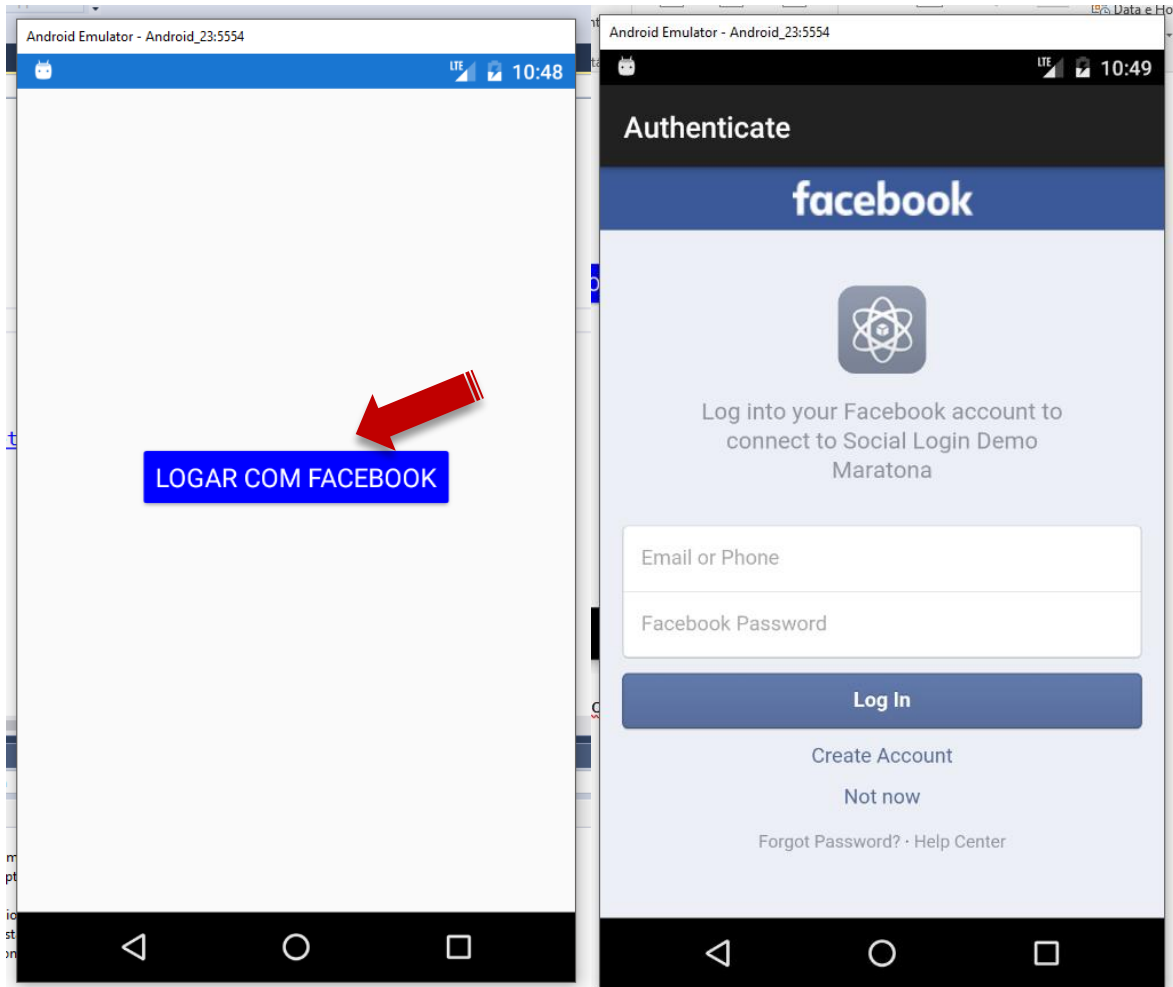
2. Salve e feche o arquivo.

Tarefa 2. Compilando o seu projeto e testando a autenticação.

1. Marque o seu **Android** projeto como **Startup Project**



2. Selecione a opção build e execute o seu projeto.
3. Clique no botão logar com facebook e na tela de autenticação informe o seu usuário se senha.



4. Se você seguiu todos os passos corretamente o resultado da autenticação será o apresentado na imagem a seguir:



Resumo

Neste laboratório, você criou um projeto em branco, configurou a SDK do Azure Mobile e por fim configurou a autenticação via rede social com Facebook.

Agora você já pode evoluir e criar experiencias fantásticas entrando no mundo maravilhoso do desenvolvimento de apps com Xamarin.Forms com Azure.

Quando tiver terminado este laboratório publique a seguinte mensagem no Twitter e no Facebook:

Eu terminei o #Lab4 da #MaratonaXamarin Intermediária e meu aplicativo está integrado ao #Azure com Social Login!