

Laboratório

Olá Xamarin!

Versão: 1.0.0

Abril de 2017

Angelo Belchior

@angelobelchior

William S. Rodriguez

@williamsrodz



Introdução

O objetivo desse LAB é criar um aplicativo Xamarin.Forms e implementar notificações por Push via Azure Notification Hubs

Objetivos

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

1. Configurar o projeto;
2. Configurar o recebimento de notificações Push;
3. Testar o recebimento de notificações Push.

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.
- Para o iOS, você precisará de uma [associação ao Programa do Desenvolvedor da Apple](#) e um dispositivo iOS físico. O [simulador do iOS não dá suporte a notificações por push](#).

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

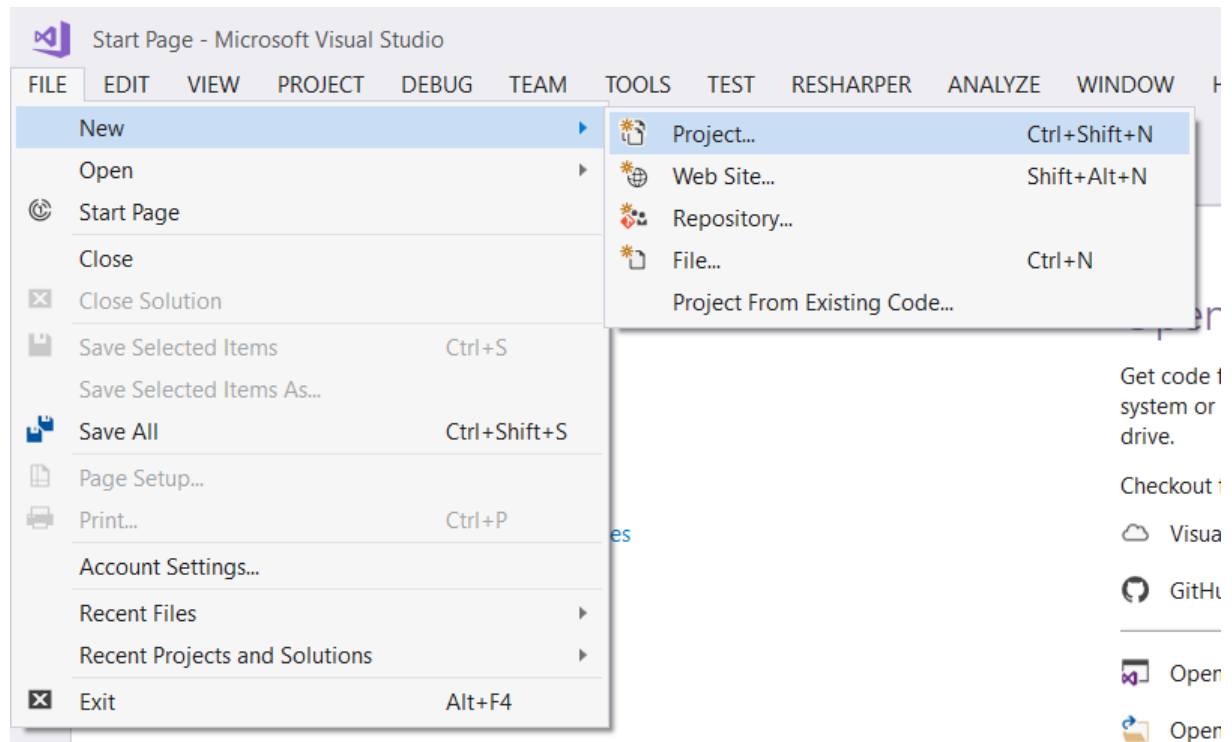
Exercício 1: Criando e Configurando o Projeto Xamarin.Forms

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

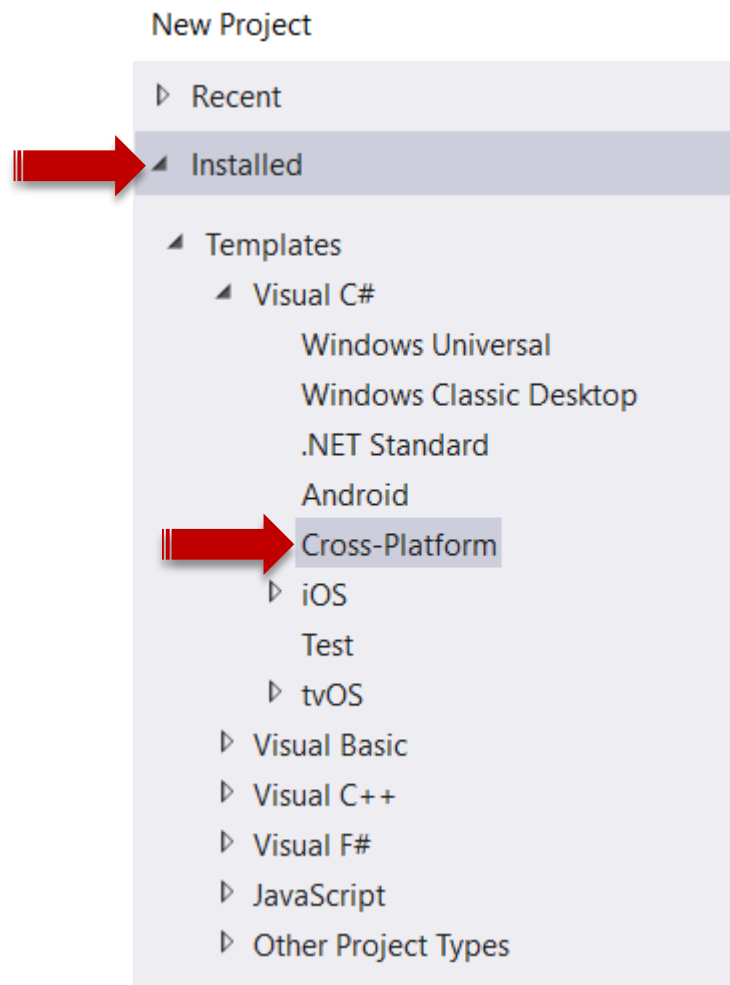
Tarefa 1. Criar o projeto Xamarin.Forms.

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

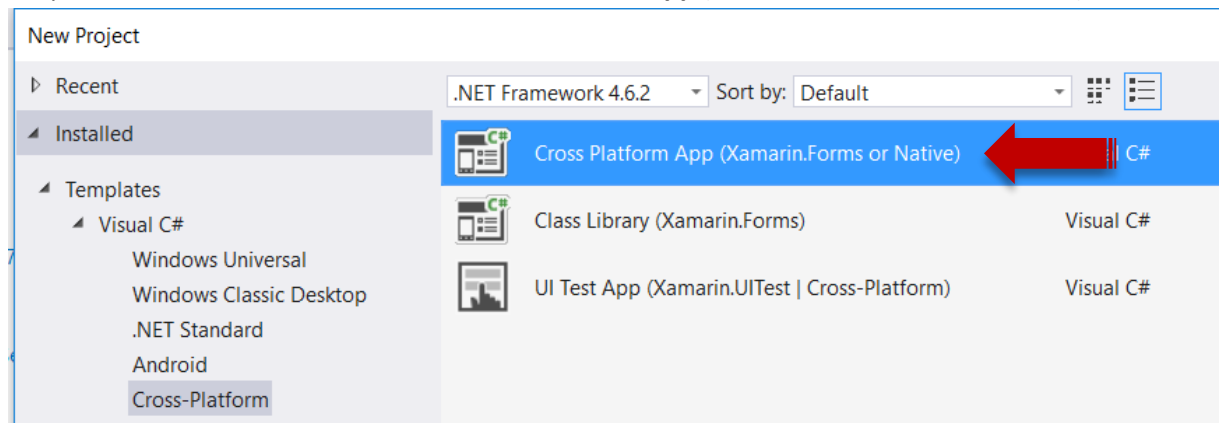
1. Selecione **File > New > Project** no Visual Studio.



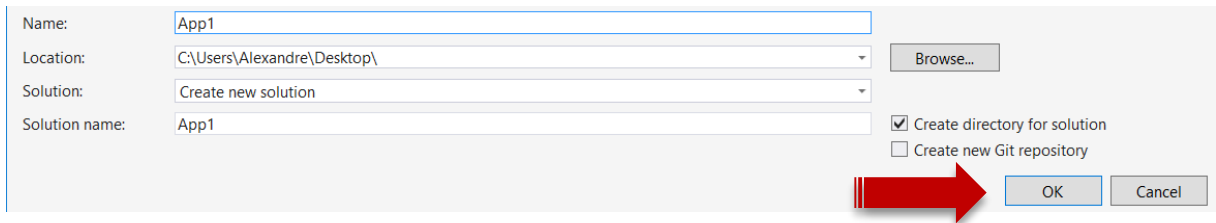
2. No painel esquerdo da janela **New Project** selecione **Visual C# > Cross-Platform** para indicar que deseja criar um aplicativo para multi-plataforma.



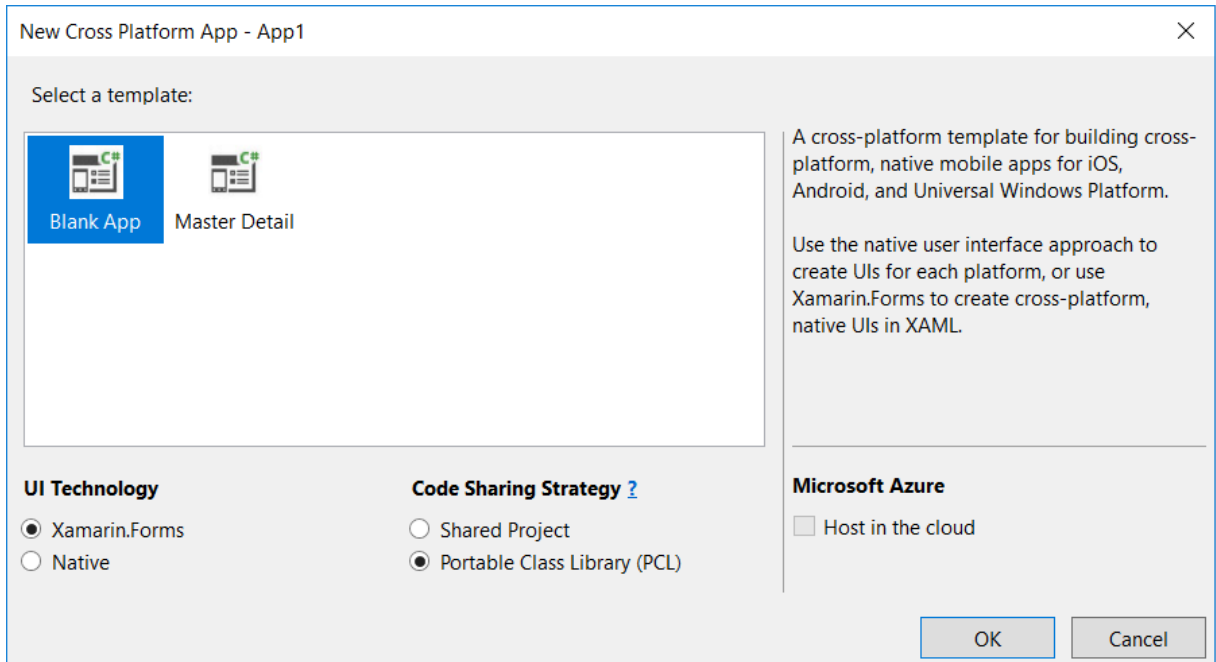
3. No painel direito, selecione o modelo **Cross Platform App (Xamarin.Forms or Native2019)**.



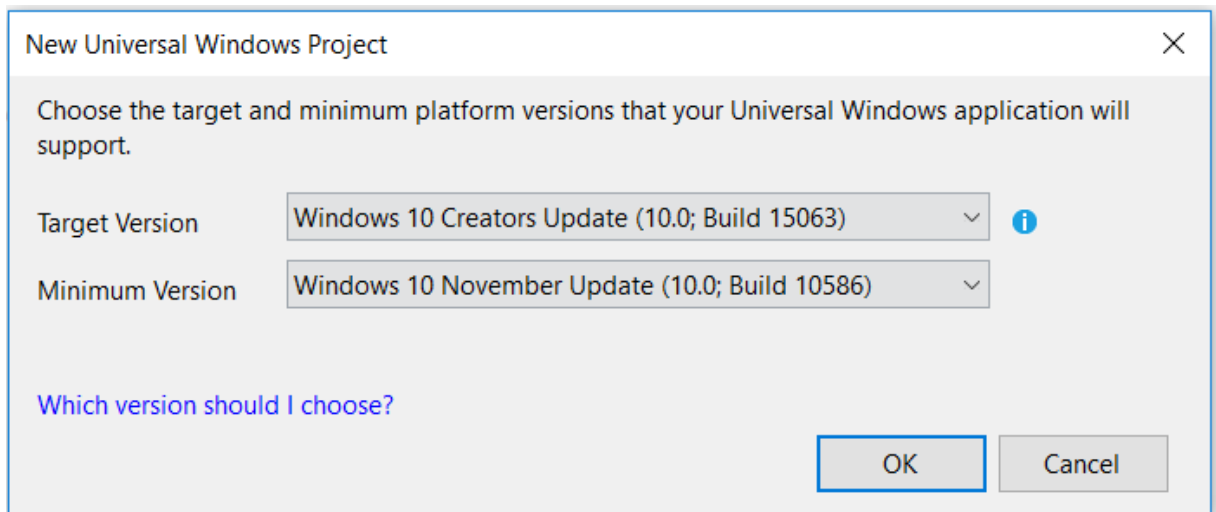
4. Forneça o nome, local e clique em **OK** para criar o projeto.



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

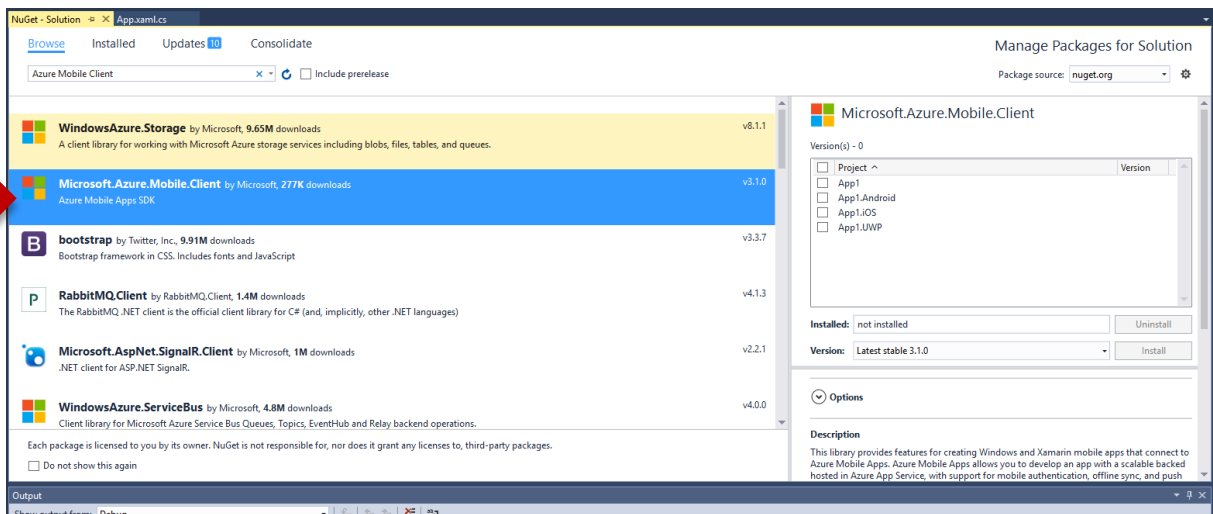


7. Aguarde a criação do projeto, ignorando qualquer instrução de conectar com o Mac.

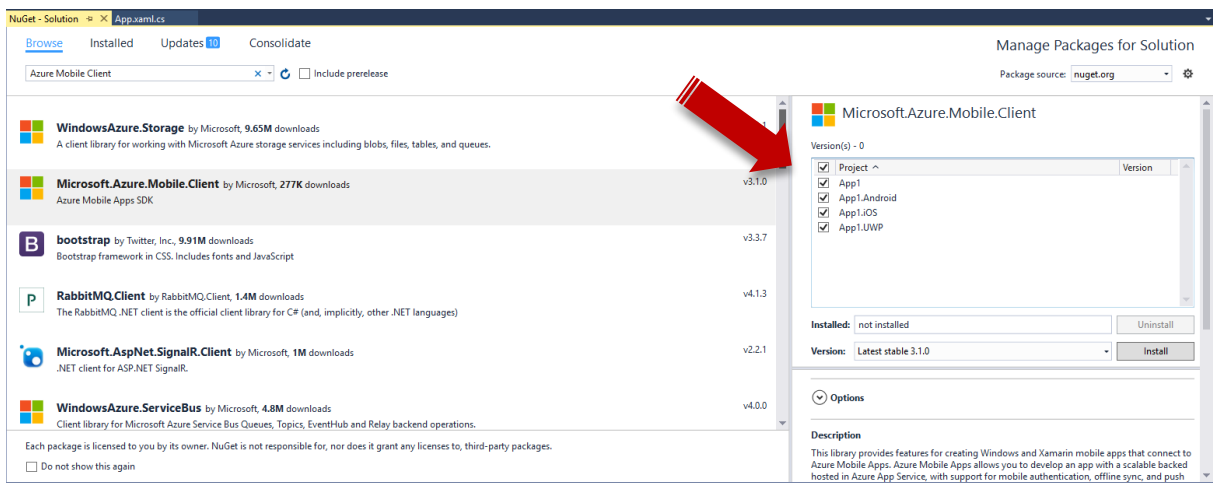
Tarefa 2. Incluir dependências do Azure Mobile Apps no seu projeto Xamarin.Forms

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

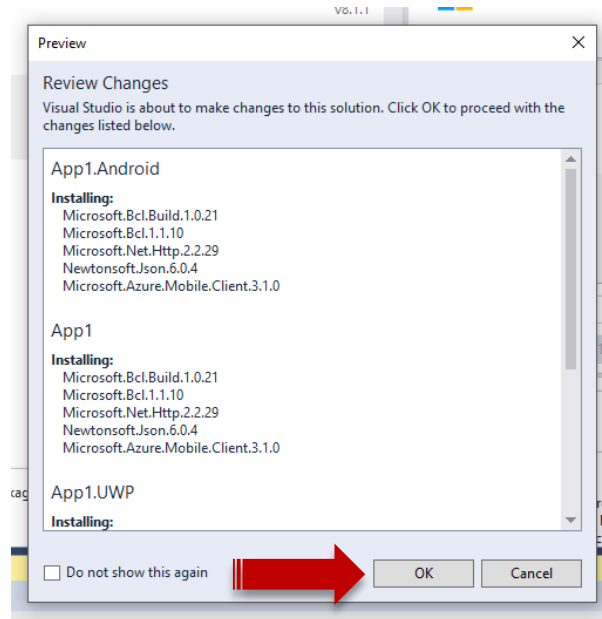
1. Clique com o botão direito na sua **Solution**, escolha a opção **Manage NuGet Packages for Solution**, clique na guia **Browse**, no campo **Search**, digite **Azure Mobile Client**, pressione **Enter** e localize o pacote **Microsoft.Azure.Client**.



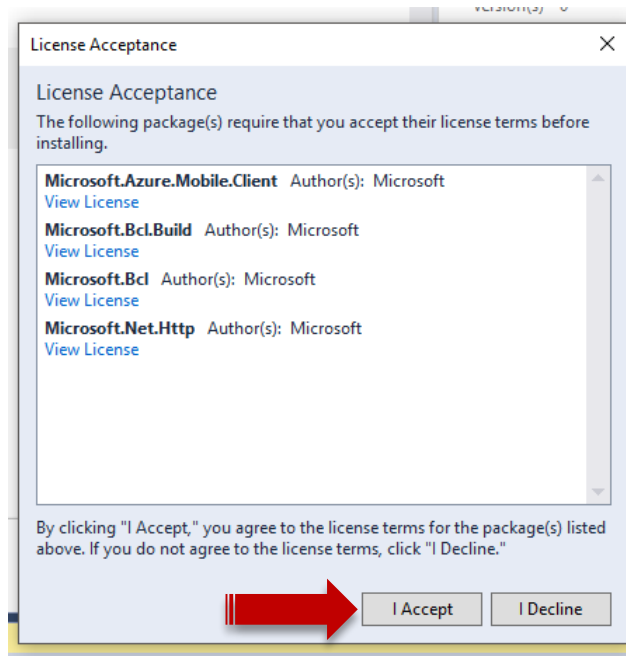
2. Na janela de detalhes do pacote, selecione todos os projetos e clique em **Install**. Isso vai iniciar o processo de download e adição das dependências do **Azure Mobile** nos projetos da sua **Solution**.



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



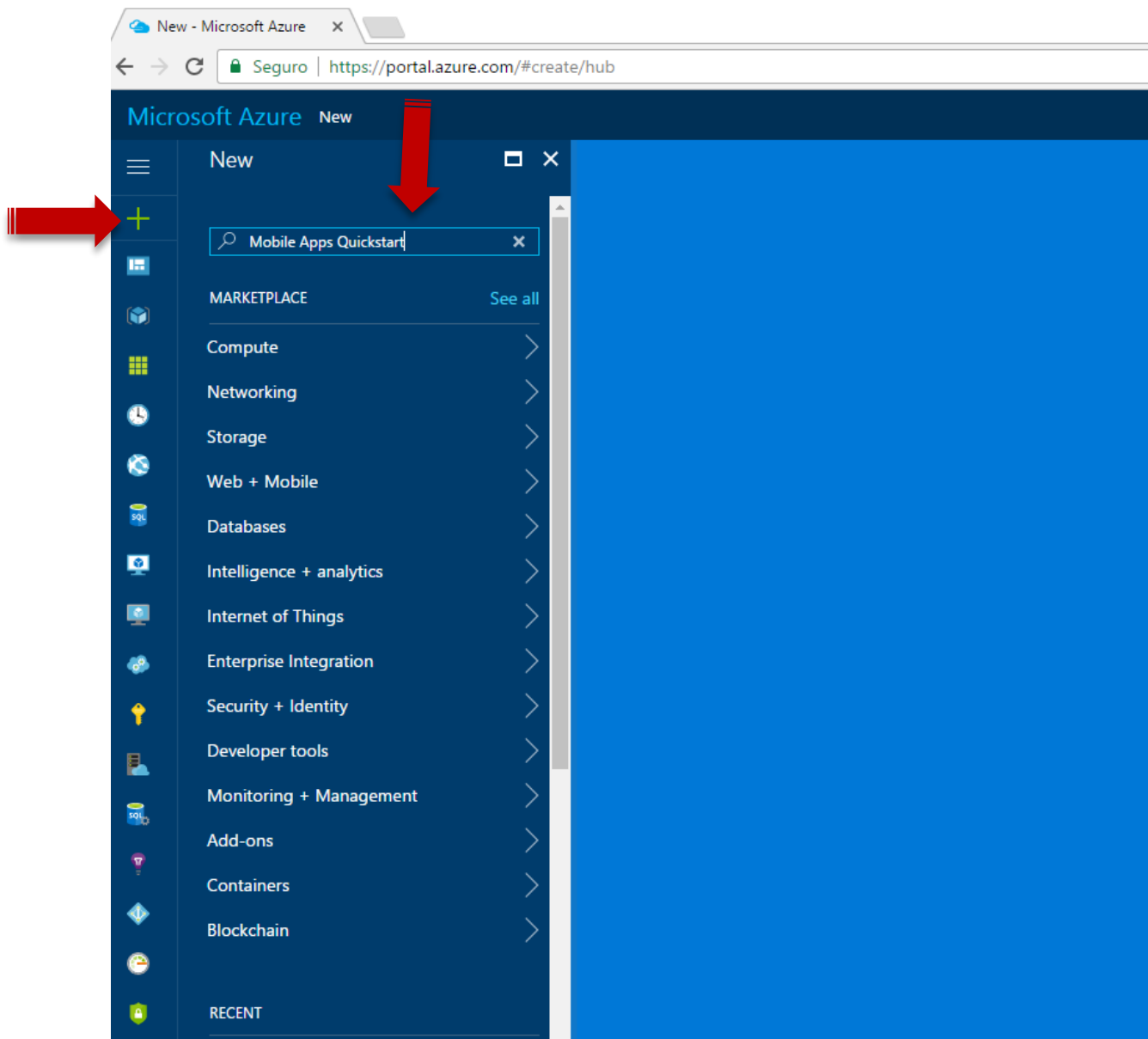
4. Clique em **I Accept** para aceitar as licenças.



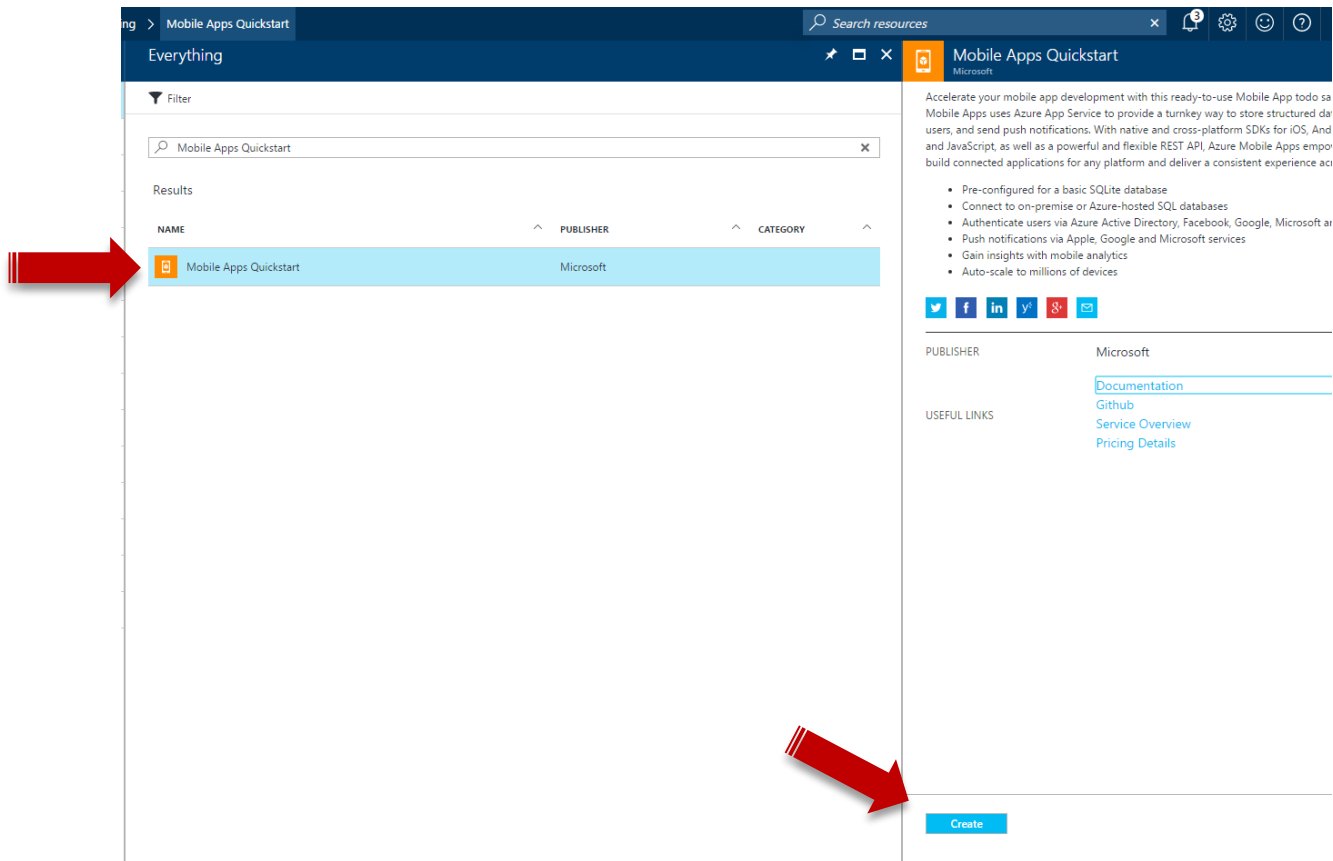
Exercício 2: Configurando o seu backend no Azure Mobile App

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**.

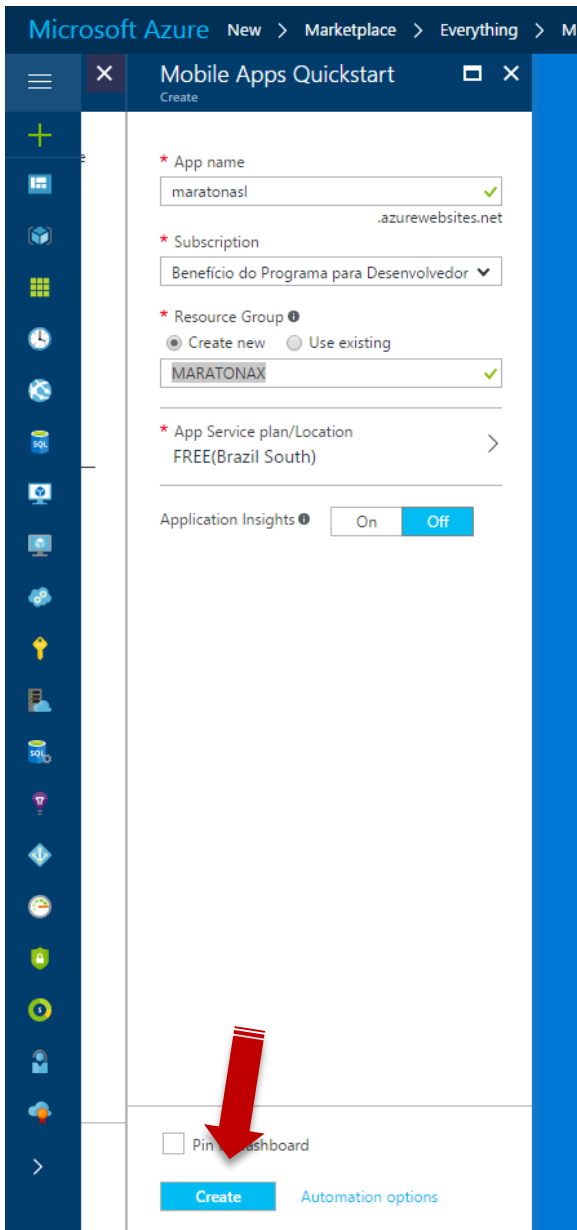


- Na blade posterior, selecione o template **Mobile Apps Quickstart** e clique no botão **Create**, para criar o seu backend.



- 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

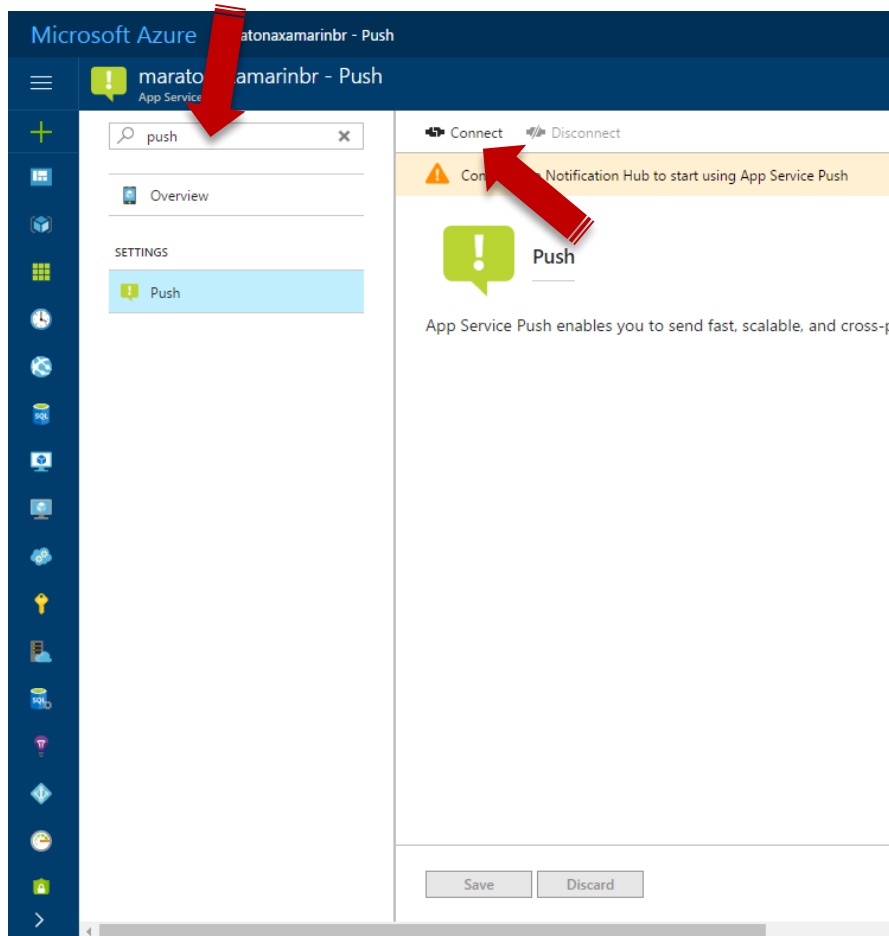
[Automation options](#)

Aguarde o seu backend ser criado e configurado.

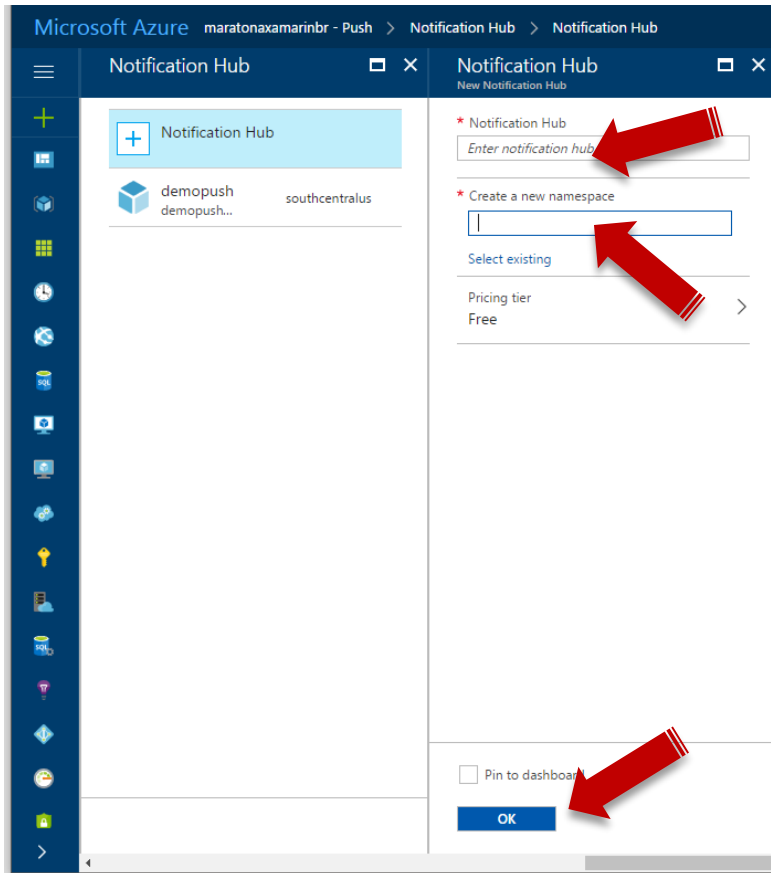
Exercício 3: Configurar um Hub de Notificação no Azure

O Azure Mobile Apps utiliza o Azure Notification Hub para enviar por push, portanto, você deve configurar um hub de notificação para seu aplicativo móvel.

1. No [portal do Azure](#), vá para o Mobile App que você acabou de criar, no campo de busca digite **Push**.
2. Clique em **Conectar** para adicionar um recurso de hub de notificação ao aplicativo. Você pode criar um hub ou conectar-se a um existente.



3. Informe um nome para o seu Hub de Notificação, e clique em OK.



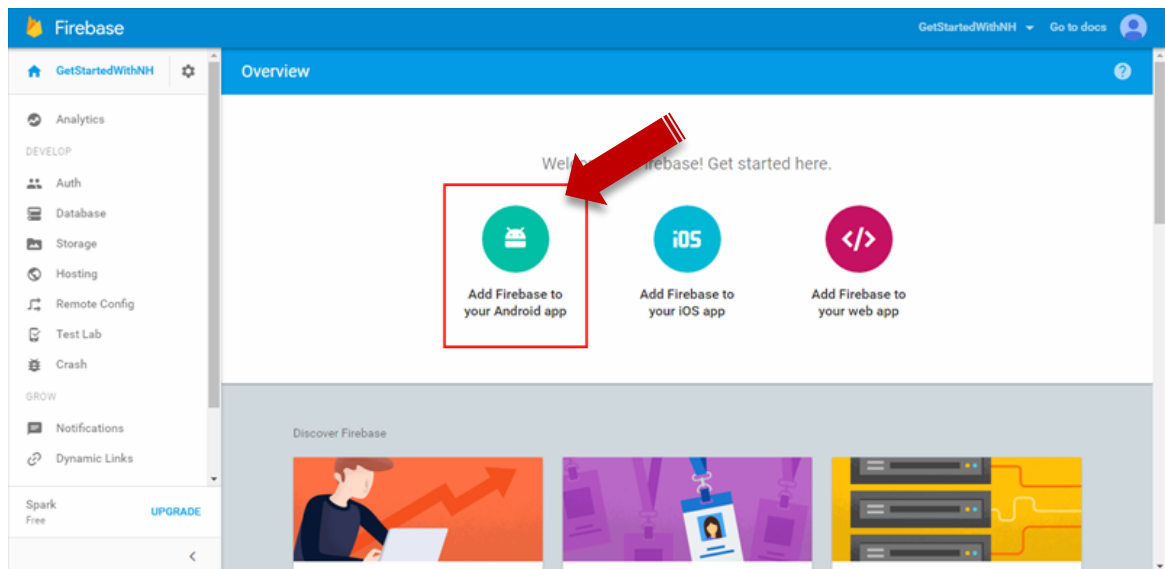
Agora você se conectou a um hub de notificação para o projeto, no próximo exercício, você vai configurar esse hub de notificação para se conectar a um PNS (Sistema de Notificação de Plataforma) a fim de enviar notificações via Push para dispositivos.

Exercício 4: Configurar e executar o projeto do Android

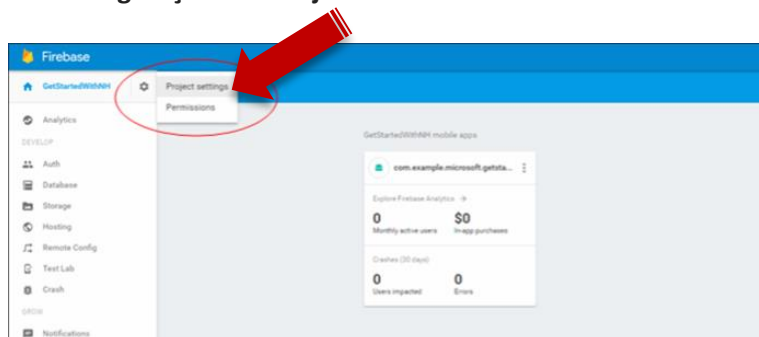
Conclua este exercício para habilitar notificações por push para o projeto Android no Xamarin.Forms.

Tarefa 1 Habilitar FCM (Firebase)

1. Faça login no [console do Firebase](#). Crie um novo projeto do Firebase se você ainda não tiver um.
2. Depois que o projeto for criado, clique em **Adicionar Firebase ao seu aplicativo Android** e siga as instruções fornecidas.



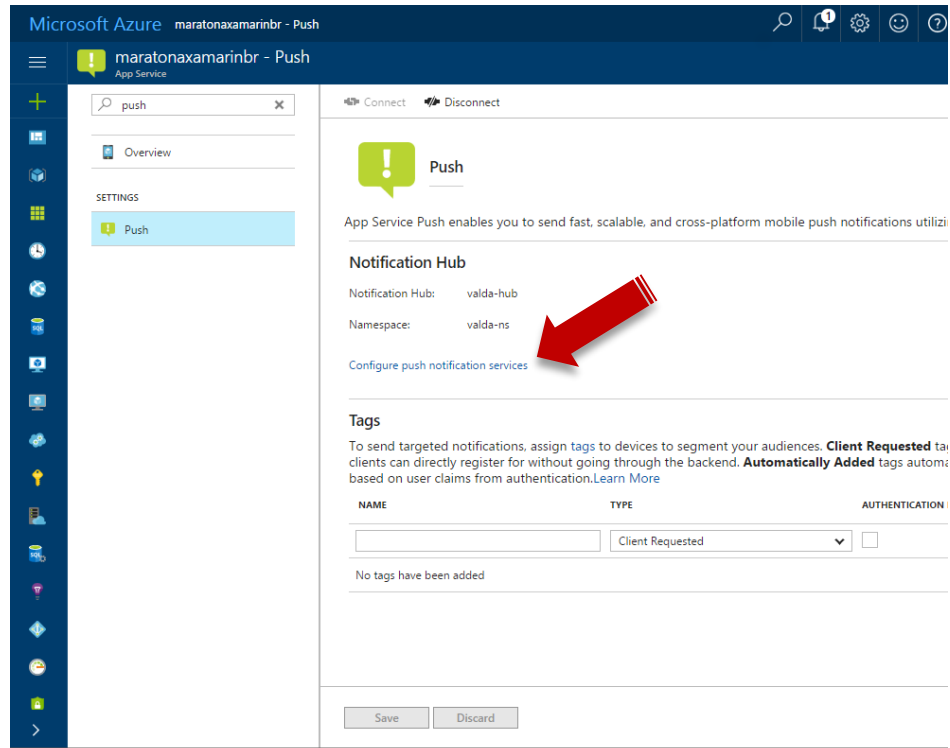
3. No Console do Firebase, clique na engrenagem para seu projeto e, em seguida, clique em **Configurações do Projeto**.



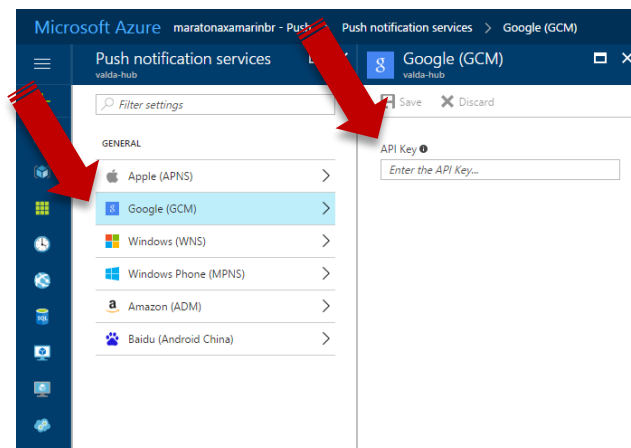
4. Clique na guia **Cloud Messaging** nas configurações do projeto e copie o valor da **Chave do servidor** e da **ID do remetente**. Esses valores serão usados mais tarde para configurar a Política de Acesso do hub de notificação e o manipulador de notificações no aplicativo.

Tarefa 2 Configurar o back-end para enviar solicitações por push usando o FCM

1. No seu projeto Mobile App no Azure, clique em Push, e em seguida clique em **Configure Push Notification Services**;



2. Clique em Google (GMC) e cole a chave de servidor que você copiou anteriormente no campo API Key.

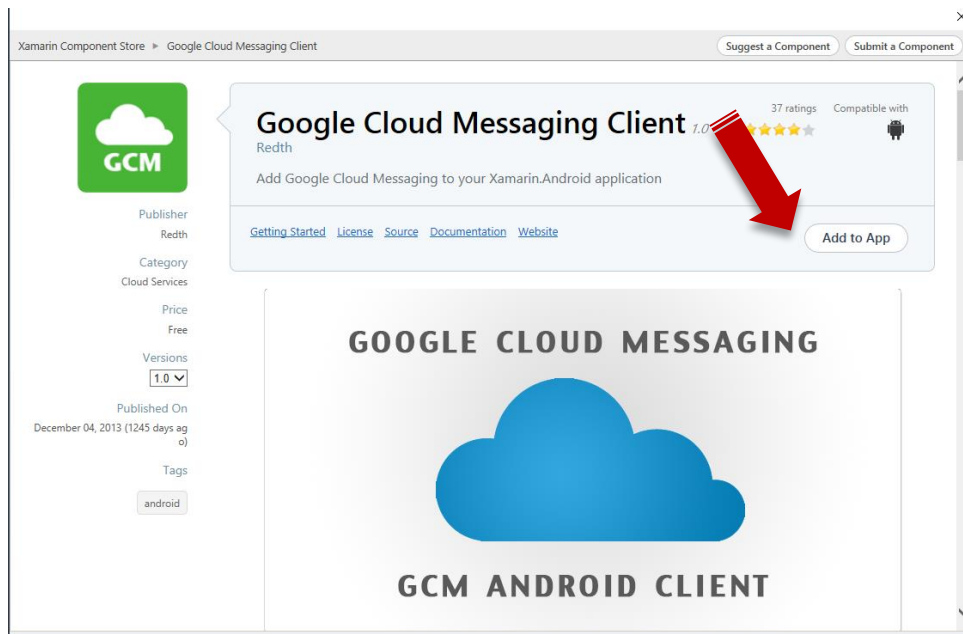


3. Clique em salvar e o serviço agora está configurado para trabalhar com o **Firestore Cloud Messaging**.

Exercício 5: Adicionar notificações por push ao projeto Droid

Com o back-end configurado com o FCM, é possível adicionar componentes e códigos ao cliente para registrá-lo no FCM. Também é possível se registrar para receber notificações por push com os Hubs de Notificação do Azure por meio do back-end dos Aplicativos Móveis e receber as notificações.

1. No projeto **Droid**, clique com o botão direito do mouse na pasta **Componentes** e clique em **Obter Mais Componentes...**
2. Pesquise o componente **Cliente do Google Cloud Messaging** e adicione-o ao projeto. Esse componente oferece suporte a notificações por push para um projeto Android Xamarin.



3. Abra o arquivo de projeto **MainActivity.cs** e adicione a seguinte instrução à parte superior do arquivo:

```
using Gcm.Client;
```

4. Adicione o código a seguir ao método **OnCreate**, após a chamada a **LoadApplication**:

```
try
{
    // Check to ensure everything's set up right
    GcmClient.CheckDevice(this);
    GcmClient.CheckManifest(this);

    // Register for push notifications
    System.Diagnostics.Debug.WriteLine("Registering...");
    GcmClient.Register(this, PushHandlerBroadcastReceiver.SENDER_IDS);
}
catch (Java.Net.MalformedURLException)
{
    CreateAndShowDialog("There was an error creating the client. Verify the
URL.", "Error");
}
catch (Exception e)
{
    CreateAndShowDialog(e.Message, "Error");
}
```

5. Adicione um novo método auxiliar **CreateAndShowDialog** , desta maneira:

```
private void CreateAndShowDialog(String message, String title)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.SetMessage (message);
    builder.SetTitle (title);
    builder.Create().Show ();
}
```

6. Adicione o código a seguir à classe **MainActivity** :

```
// Create a new instance field for this activity.
static MainActivity instance = null;

// Return the current activity instance.
public static MainActivity CurrentActivity
{
    get
    {
        return instance;
    }
}
```



```
}
```

Isso expõe a instância **MainActivity** atual e, portanto, podemos executar no thread principal da interface do usuário.

7. Inicialize a variável `instance` no início do método **OnCreate** como mostrado a seguir.

```
// Set the current instance of MainActivity.  
instance = this;
```

8. Adicione um novo arquivo de classe ao projeto **Droid** chamado `GcmService.cs` e verifique se as instruções **using** a seguir estão presentes na parte superior do arquivo:

```
using Android.App;  
using Android.Content;  
using Android.Media;  
using Android.Support.V4.App;  
using Android.Util;  
using Gcm.Client;  
using Microsoft.WindowsAzure.MobileServices;  
using Newtonsoft.Json.Linq;  
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Text;
```

9. Adicione as seguintes solicitações de permissão na parte superior do arquivo, após as instruções **using** e antes da declaração **namespace**.

```
[assembly: Permission(Name = "@PACKAGE_NAME@.permission.C2D_MESSAGE")]  
[assembly: UsesPermission(Name = "@PACKAGE_NAME@.permission.C2D_MESSAGE")]  
[assembly: UsesPermission(Name =  
"com.google.android.c2dm.permission.RECEIVE")]  
[assembly: UsesPermission(Name = "android.permission.INTERNET")]  
[assembly: UsesPermission(Name = "android.permission.WAKE_LOCK")]  
//GET_ACCOUNTS is only needed for android versions 4.0.3 and below  
[assembly: UsesPermission(Name = "android.permission.GET_ACCOUNTS")]
```

10. Adicione a seguinte definição de classe ao namespace.

```
[BroadcastReceiver(Permission =
Gcm.Client.Constants.PERMISSION_GCM_INTENTS)]
[IntentFilter(new string[] { Gcm.Client.Constants.INTENT_FROM_GCM_MESSAGE },
Categories = new string[] { "@PACKAGE_NAME@" })]
[IntentFilter(new string[] {
Gcm.Client.Constants.INTENT_FROM_GCM_REGISTRATION_CALLBACK }, Categories =
new string[] { "@PACKAGE_NAME@" })]
[IntentFilter(new string[] {
Gcm.Client.Constants.INTENT_FROM_GCM_LIBRARY_RETRY }, Categories = new
string[] { "@PACKAGE_NAME@" })]
public class PushHandlerBroadcastReceiver :
GcmBroadcastReceiverBase<GcmService>
{
    public static string[] SENDER_IDS = new string[] { "<PROJECT_NUMBER>" };
}
```

11. Substitua a classe **GcmService** vazia pelo código a seguir, que usa o novo receptor de difusão:

```
[Service]
public class GcmService : GcmServiceBase
{
    MobileServiceClient client = new MobileServiceClient("MOBILE_APP_URL");

    public static string RegistrationID { get; private set; }

    public GcmService()
        : base(PushHandlerBroadcastReceiver.SENDER_IDS){}
}
```

12. Adicione o código a seguir à classe **GcmService**. Isso substitui o manipulador de eventos **OnRegistered** e implementa um método **Register**.

```
protected override void OnRegistered(Context context, string registrationId)
{
    Log.Verbose("PushHandlerBroadcastReceiver", "GCM Registered: " +
registrationId);
    RegistrationID = registrationId;

    var push = client.GetPush();

    MainActivity.CurrentActivity.RunOnUiThread(() => Register(push, null));
}

public async void Register(Microsoft.WindowsAzure.MobileServices.Push push,
IEnumerable<string> tags)
{
    try
    {
        const string templateBodyGCM =
"\{\"data\":{\\"message\":\\"$(messageParam)\\"}\}";

        JObject templates = new JObject();
        templates["genericMessage"] = new JObject
        {
            {"body", templateBodyGCM}
        };

        await push.RegisterAsync(RegistrationID, templates);
        Log.Info("Push Installation Id", push.InstallationId.ToString());
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex.Message);
        Debugger.Break();
    }
}
```

Observe que esse código usa o parâmetro **messageParam** no registro do modelo.

13. Adicione o seguinte código que implementa **OnMessage**:

```
protected override void OnMessage(Context context, Intent intent)
{
    Log.Info("PushHandlerBroadcastReceiver", "GCM Message Received!");

    var msg = new StringBuilder();

    if (intent != null && intent.Extras != null)
    {
        foreach (var key in intent.Extras.KeySet())
            msg.AppendLine(key + "=" + intent.Extras.Get(key).ToString());
    }

    //Store the message
    var prefs = GetSharedPreferences(context.PackageName,
    FileCreationMode.Private);
    var edit = prefs.Edit();
    edit.PutString("last_msg", msg.ToString());
    edit.Commit();

    string message = intent.Extras.GetString("message");
    if (!string.IsNullOrEmpty(message))
    {
        createNotification("New todo item!", "Todo item: " + message);
        return;
    }

    string msg2 = intent.Extras.GetString("msg");
    if (!string.IsNullOrEmpty(msg2))
    {
        createNotification("New hub message!", msg2);
        return;
    }

    createNotification("Unknown message details", msg.ToString());
}

void createNotification(string title, string desc)
{
    //Create notification
    var notificationManager = GetSystemService(Context.NotificationService)
    as NotificationManager;

    //Create an intent to show ui
    var uiIntent = new Intent(this, typeof(MainActivity));

    //Use Notification Builder
    NotificationCompat.Builder builder = new
    NotificationCompat.Builder(this);
```

```
//Create the notification
//we use the pending intent, passing our ui intent over which will get
called
//when the notification is tapped.
var notification =
builder.SetContentIntent(PendingIntent.GetActivity(this, 0, uiIntent, 0))
        .SetSmallIcon(Android.Resource.Drawable.SymActionEmail)
        .SetTicker(title)
        .SetContentTitle(title)
        .SetContentText(desc)

        //Set the notification sound

.SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification))

        //Auto cancel will remove the notification once the user touches
it
        .SetAutoCancel(true).Build();

//Show the notification
notificationManager.Notify(1, notification);
}
```

Isso manipula as notificações recebidas e as envia para o gerenciador de notificações a ser exibido.

14. **GcmServiceBase** também exige que você implemente os métodos de manipulador **OnUnRegistered** e **OnError**, o que pode ser feito da seguinte maneira:

```
protected override void OnUnRegistered(Context context, string
registrationId)
{
    Log.Error("PushHandlerBroadcastReceiver", "Unregistered RegistrationId :
" + registrationId);
}

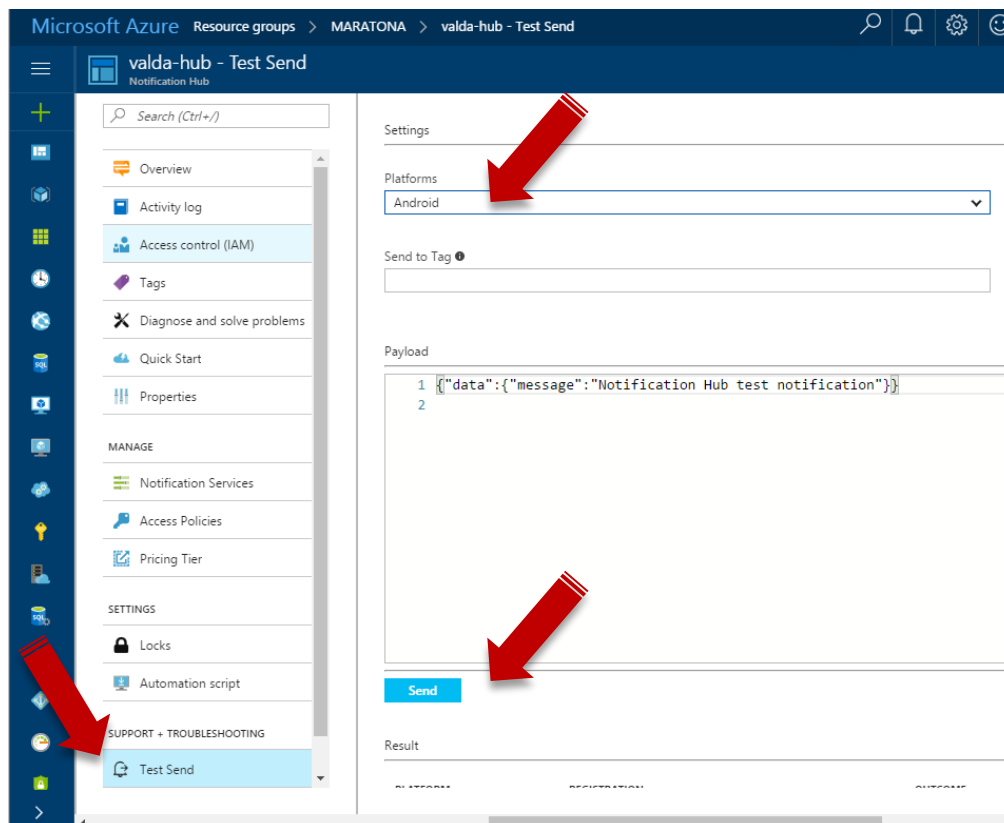
protected override void OnError(Context context, string errorId)
{
    Log.Error("PushHandlerBroadcastReceiver", "GCM Error: " + errorId);
}
```

Agora, você está pronto para testar as notificações por push no aplicativo em execução em um dispositivo ou no emulador Android.

Exercício 6: Testar notificações por push em seu aplicativo Android

Atenção: As duas primeiras etapas serão necessárias apenas quando o teste estiver sendo feito em um emulador que não tenha instalado o Google Play Services.

1. Verifique se você está implantando ou depurando em um dispositivo virtual que tem as APIs do Google definidas como o destino, como mostrado abaixo no gerenciador de Dispositivo Virtual do Android.
2. Adicione uma conta do Google ao dispositivo Android clicando em **Aplicativos > Configurações > Adicionar conta**. Depois, siga os prompts para adicionar uma conta existente do Google ao dispositivo ou para criar uma nova.
3. No Visual Studio ou Xamarin Studio, clique com o botão direito do mouse no projeto **Droid** e clique em **Definir como projeto de inicialização**.
4. Clique em **Executar** para criar o projeto e iniciar o aplicativo no emulador ou no dispositivo Android.
5. No portal do Azure, localize o seu Hub de Notificação criado, clique em **Test Send**, escolha a **plataforma de destino** (no nosso caso Android), e clique em **Send**.



6. No emulador, verifique se uma notificação é recebida.

Exercício 7: Configurar e executar o projeto do iOS (opcional)

Atenção: Esta seção trata da execução do projeto do iOS Xamarin para dispositivos iOS. Você poderá ignorá-la se não estiver trabalhando com dispositivos iOS.

- [Registrar uma ID para seu aplicativo](#). Crie uma ID de aplicativo explícita (não uma ID de aplicativo de caractere curinga) e, para **ID do Pacote**, use a ID do pacote exata em seu projeto de início rápido do Xcode. É também crucial que você marque a opção **Notificações por Push**.
- Em seguida, [para se preparar para configurar notificações por push](#), crie um certificado SSL de "Desenvolvimento" ou "Distribuição".

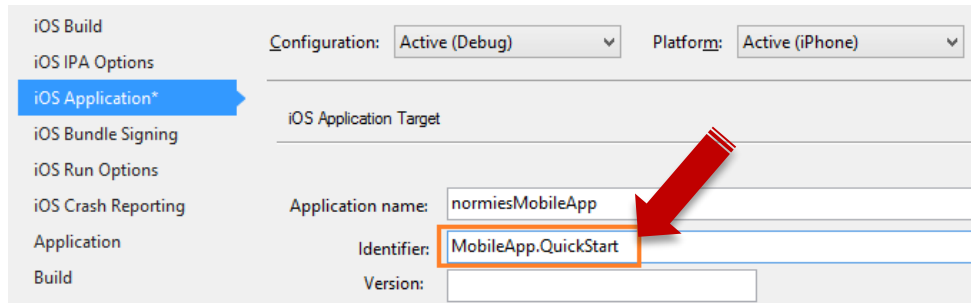
Tarefa 01, configurar o hub de notificação para APNS

1. Em seu Mac, inicie **Acesso ao Conjunto de Chaves**. Na barra de navegação à esquerda, em **Categoria**, abra **Meus Certificados**. Encontre o certificado SSL que você baixou na seção anterior e divulgue seu conteúdo. Selecione apenas o certificado (não selecione a chave privada) e [exporte-o](#).
2. No [portal do Azure](#), clique em **Procurar Tudo > Serviços de Aplicativos** e clique no back-end dos Aplicativos Móveis. Em **Configurações**, clique em **Push do Serviço de Aplicativo** e clique no nome do hub de notificação. Vá para **Serviços de Notificação por Push da Apple > Carregar Certificado**. Carregue o arquivo .p12 selecionando o **Modo** correto (dependendo do certificado de cliente SSL anterior ser de produção ou de área restrita).
3. Salve as alterações.

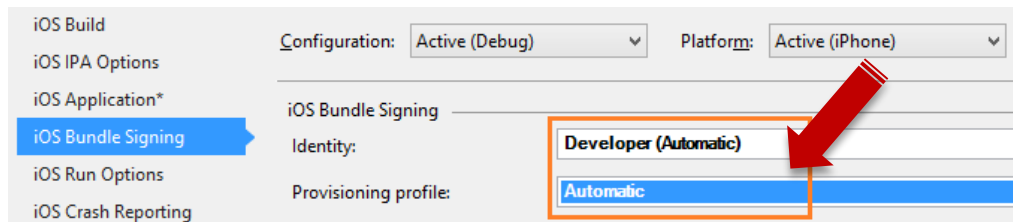
Agora, seu serviço móvel está configurado para funcionar com notificações por push no iOS.

Tarefa 02, configurar o projeto iOS no Visual Studio

1. No Visual Studio, clique com o botão direito do mouse no projeto e clique em **Propriedades**.
2. Nas páginas de propriedades, clique na guia **Aplicativo iOS** e atualize o **Identificador** com a ID que você criou anteriormente.



3. Na guia **Assinatura do Pacote iOS**, selecione a identidade e o perfil de provisionamento correspondentes que você acabou de configurar para este projeto.



Isso garantirá que o projeto use o novo perfil para a assinatura de código. Para obter a documentação oficial de provisionamento do dispositivo Xamarin, consulte [Provisionamento do dispositivo Xamarin](#).

4. Clique duas vezes em Info.plist para abri-lo e habilite **RemoteNotifications** em **Modos de Segundo Plano**.

Tarefa 03, adicionar as notificações por push ao seu aplicativo iOS

1. No projeto **iOS**, abra AppDelegate.cs e adicione a instrução a seguir à parte superior do arquivo de código.

```
using Newtonsoft.Json.Linq;
```

2. Na classe **AppDelegate**, adicione também uma substituição ao evento **RegisteredForRemoteNotifications** a fim de registrar para o recebimento notificações:


```
public override void RegisteredForRemoteNotifications(UIApplication
application,
    NSData deviceToken)
{
    const string templateBodyAPNS =
        "{\"aps\":{\"alert\":\"$(messageParam)\"}}";

    JObject templates = new JObject();
    templates["genericMessage"] = new JObject
    {
        {"body", templateBodyAPNS}
    };

    // Register for push with your mobile app
    MobileServiceClient client = new MobileServiceClient("URL DO SEU APP");
    var push = client.GetPush();
    push.RegisterAsync(deviceToken, templates);
}
```

3. Em **AppDelegate**, adicione também a seguinte substituição para o manipulador de eventos **DidReceiveRemoteNotification**:

```
public override void DidReceiveRemoteNotification(UIApplication application,
    NSDictionary userInfo, Action<UIBackgroundFetchResult>
completionHandler)
{
    NSDictionary aps = userInfo.ObjectForKey(new NSString("aps")) as
NSDictionary;

    string alert = string.Empty;
    if (aps.ContainsKey(new NSString("alert")))
        alert = (aps[new NSString("alert")] as NSString).ToString();

    //show alert
    if (!string.IsNullOrEmpty(alert))
    {
        UIAlertView avAlert = new UIAlertView("Notification", alert, null,
"OK", null);
        avAlert.Show();
    }
}
```

Este método trata as notificações recebidas enquanto o aplicativo está em execução.

4. Na classe **AppDelegate**, adicione o seguinte código ao método **FinishedLaunching**:

```
// Register for push notifications.
var settings = UIUserNotificationSettings.GetSettingsForTypes(
    UIUserNotificationType.Alert
    | UIUserNotificationType.Badge
    | UIUserNotificationType.Sound,
    new NSSet());

UIApplication.SharedApplication.RegisterUserNotificationSettings(settings);
UIApplication.SharedApplication.RegisterForRemoteNotifications();
```

Isso habilita o suporte para notificações remotas e solicitações de registro por push.

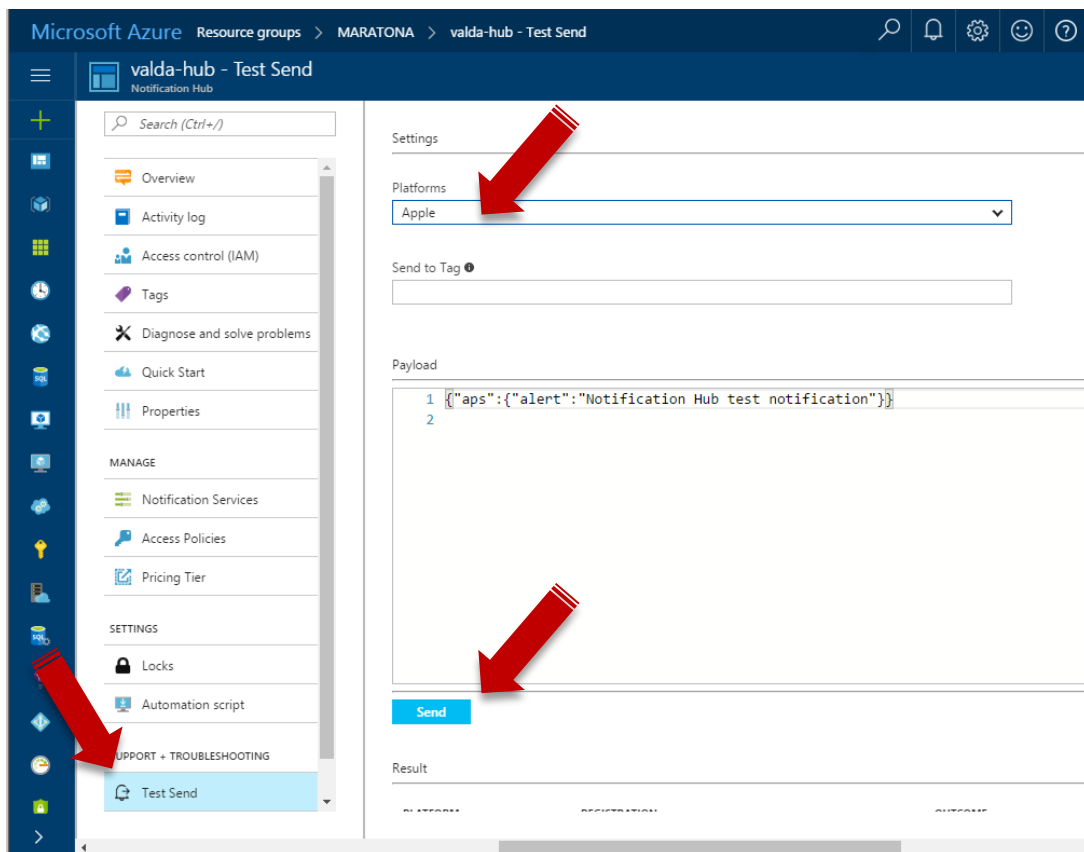
Seu aplicativo foi atualizado para oferecer suporte a notificações de push.

Exercício 8: Testar notificações por push em seu aplicativo iOS (opcional)

1. Clique com o botão direito do mouse no projeto do iOS e, depois, clique em **Definir como Projeto de Inicialização**.
2. Pressione o botão **Executar** ou **F5** no Visual Studio para criar o projeto e iniciar o aplicativo em um dispositivo iOS. Em seguida, clique em **OK** para aceitar as notificações por push.

Atenção: Você deve aceitar explicitamente as notificações por push do seu aplicativo. Essa solicitação ocorrerá apenas na primeira vez que o aplicativo for executado.

3. No portal do Azure, localize o seu Hub de Notificação criado, clique em **Test Send**, escolha a **plataforma de destino** (no nosso caso Apple), e clique em **Send**.

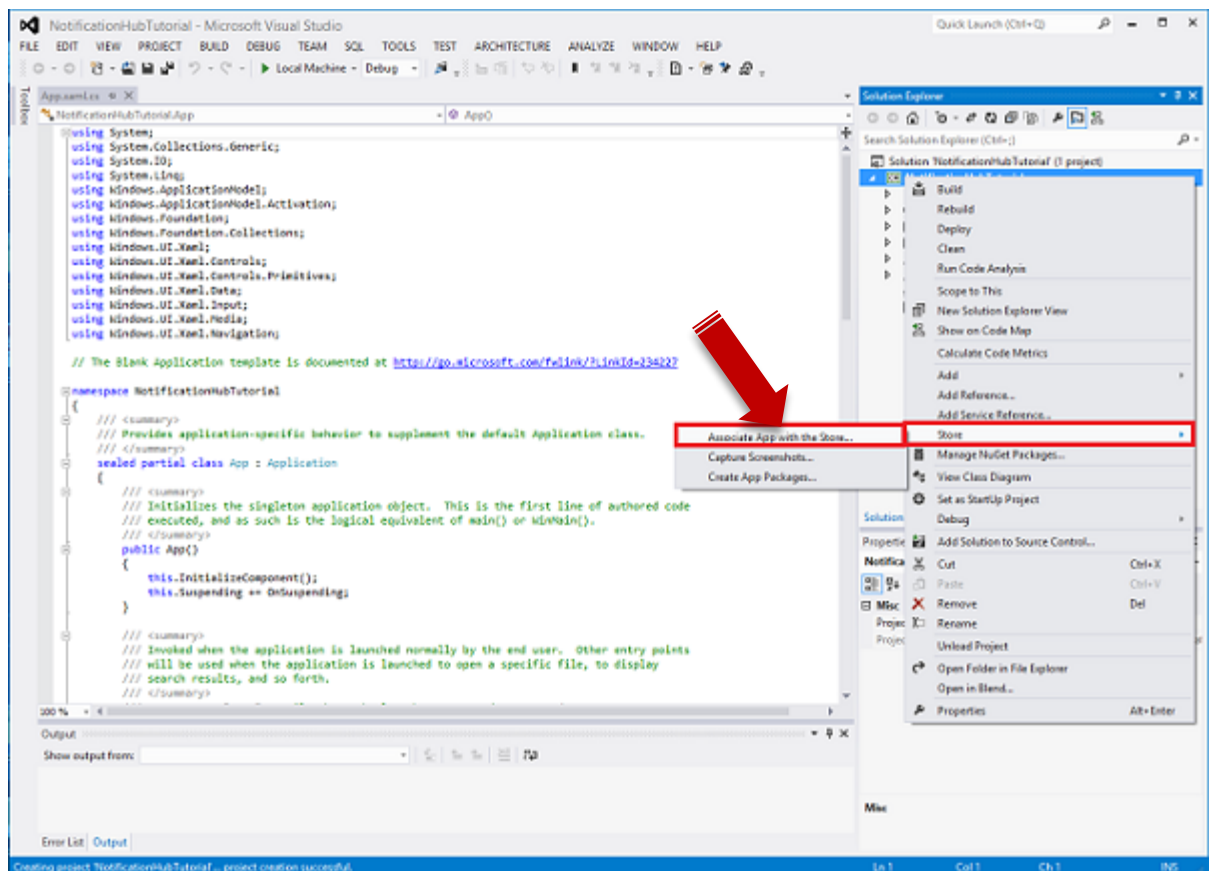


4. No simulador, verifique se uma notificação é recebida.

Exercício 9: Configurar e executar projetos do Windows (opcional)

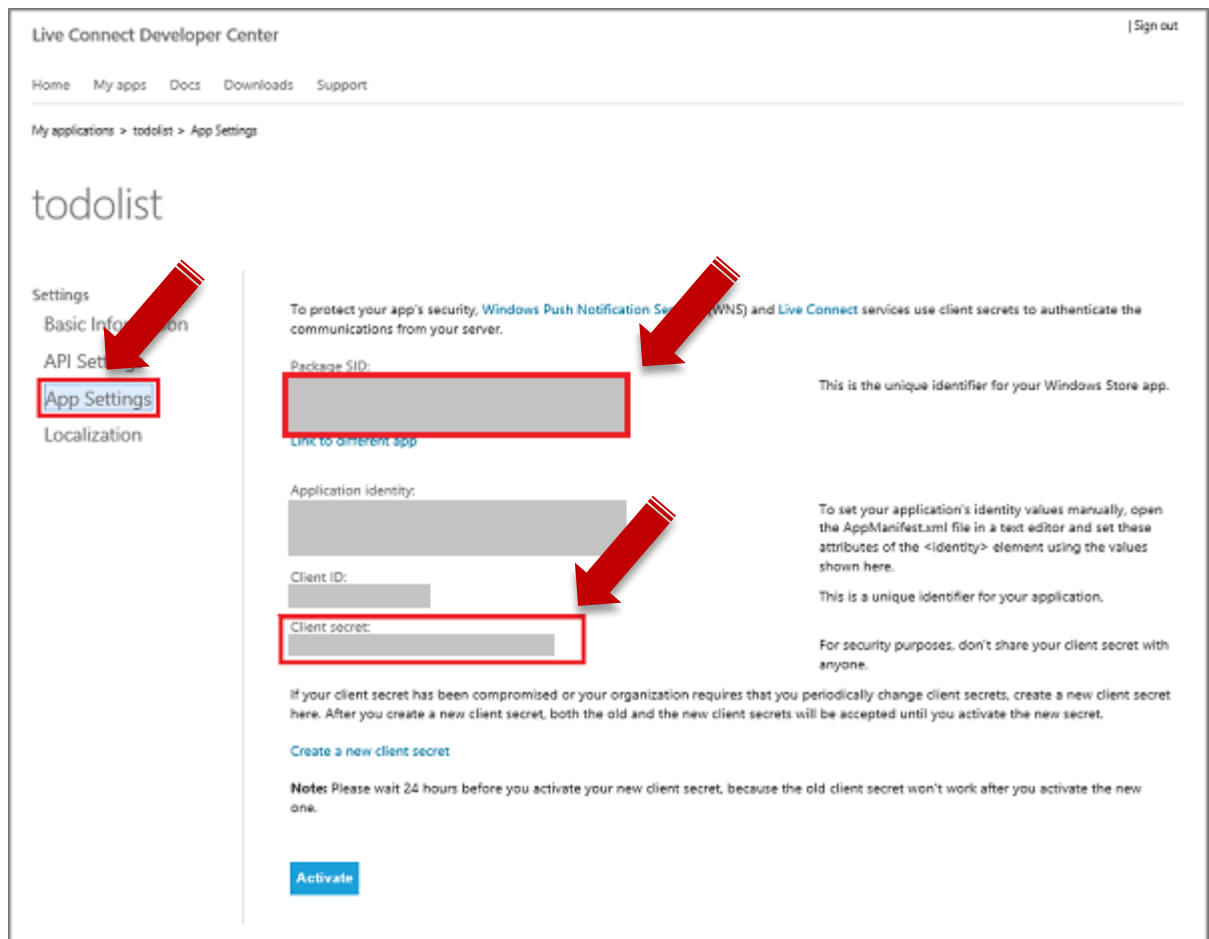
Atenção: Esta seção trata da execução dos projetos WinApp e WinPhone81 de Xamarin.Forms para dispositivos Windows. Estas etapas também oferecem suporte a projetos da Plataforma Universal do Windows (UWP). Você poderá ignorá-la se não estiver trabalhando com dispositivos Windows. Tarefa 01, registrar o aplicativo Windows para receber notificações por push com o WNS (Serviço de Notificação do Windows)

1. No Gerenciador de Soluções do Visual Studio, clique com o botão direito do mouse no projeto do aplicativo da Windows Store e clique em **Armazenar > Associar Aplicativo à Store**.



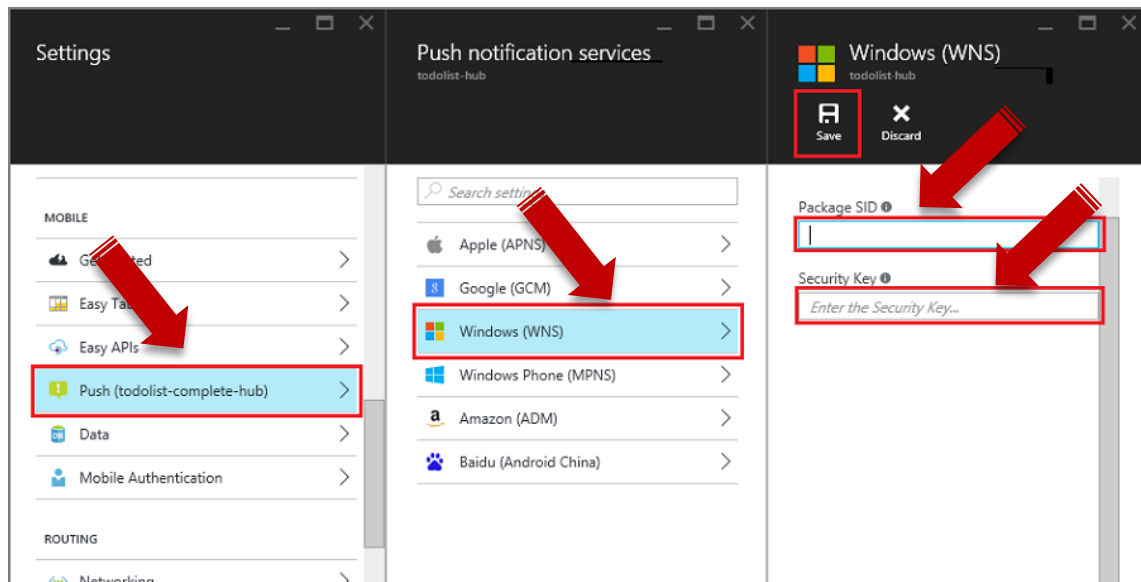
2. No assistente, clique em **Avançar** e entre com sua conta da Microsoft. Digite um nome para o seu aplicativo em **Reservar nome do aplicativo** e clique em **Reservar**.

3. Depois que o registro do aplicativo for criado com êxito, selecione o novo nome do aplicativo, clique em **Avançar** e em **Associar**. Isso adiciona as informações de registro necessárias da Windows Store para o manifesto do aplicativo.
4. Repita as etapas 1 e 3 para o projeto de aplicativo da Windows Phone Store usando o mesmo registro que você criou anteriormente para o aplicativo da Windows Store.
5. Navegue até o [Centro de Desenvolvimento do Windows](#) e entre com sua conta da Microsoft. Clique no novo registro de aplicativo em **Meus aplicativos** e expanda **Serviços > Notificações por push**.
6. Na página **Notificações por push**, clique em **site dos Live Services em WNS (Serviços de Notificação por Push do Windows)** e **Serviços Móveis do Microsoft Azure**. Anote os valores do **SID do Pacote** e o valor *atual* em **Segredo do Aplicativo**.



Tarefa 02, configurar o hub de notificação para WNS

1. No [portal do Azure](#), clique em **Procurar Tudo > Serviços de Aplicativos** e clique no back-end dos Aplicativos Móveis. Em **Configurações**, clique em **Push do Serviço de Aplicativo** e clique no nome do hub de notificação.
2. Vá para **Windows (WNS)**, digite a **Chave de segurança** (segredo do cliente) e o **SID do pacote** que você obteve no site dos Live Services e clique em **Salvar**.



O back-end agora está configurado para usar o WNS a fim de enviar notificações por push.

Tarefa 02, adicionar notificações por push ao seu aplicativo do Windows

1. No Visual Studio, abra **App.xaml.cs** em um projeto do Windows e adicione as instruções a seguir.

```
using Newtonsoft.Json.Linq;  
using Microsoft.WindowsAzure.MobileServices;  
using System.Threading.Tasks;  
using Windows.Networking.PushNotifications;
```

2. No App.xaml.cs, adicione o seguinte método **InitNotificationsAsync**:

```
private async Task InitNotificationsAsync()
```

```
{
    var channel = await PushNotificationChannelManager
        .CreatePushNotificationChannelForApplicationAsync();

    const string templateBodyWNS =
        "<toast><visual><binding template=\"ToastText01\"><text
id=\"1\">$(messageParam)</text></binding></visual></toast>";

    JObject headers = new JObject();
    headers["X-WNS-Type"] = "wns/toast";

    JObject templates = new JObject();
    templates["genericMessage"] = new JObject
    {
        {"body", templateBodyWNS},
        {"headers", headers} // Needed for WNS.
    };

    MobileServiceClient client = new MobileServiceClient("MOBILE_APP_URL");

    await client.GetPush()
        .RegisterAsync(channel.Uri, templates);
}
```

Esse método obtém o canal de notificação por push e registra um modelo para receber as notificações de modelo do hub de notificação. Uma notificação de modelo que oferece suporte a *messageParam* será entregue a esse cliente.

3. No App.xaml.cs, atualize a definição de método do manipulador de eventos **OnLaunched** adicionando o modificador `async`. Depois, adicione a seguinte linha de código ao final do método:

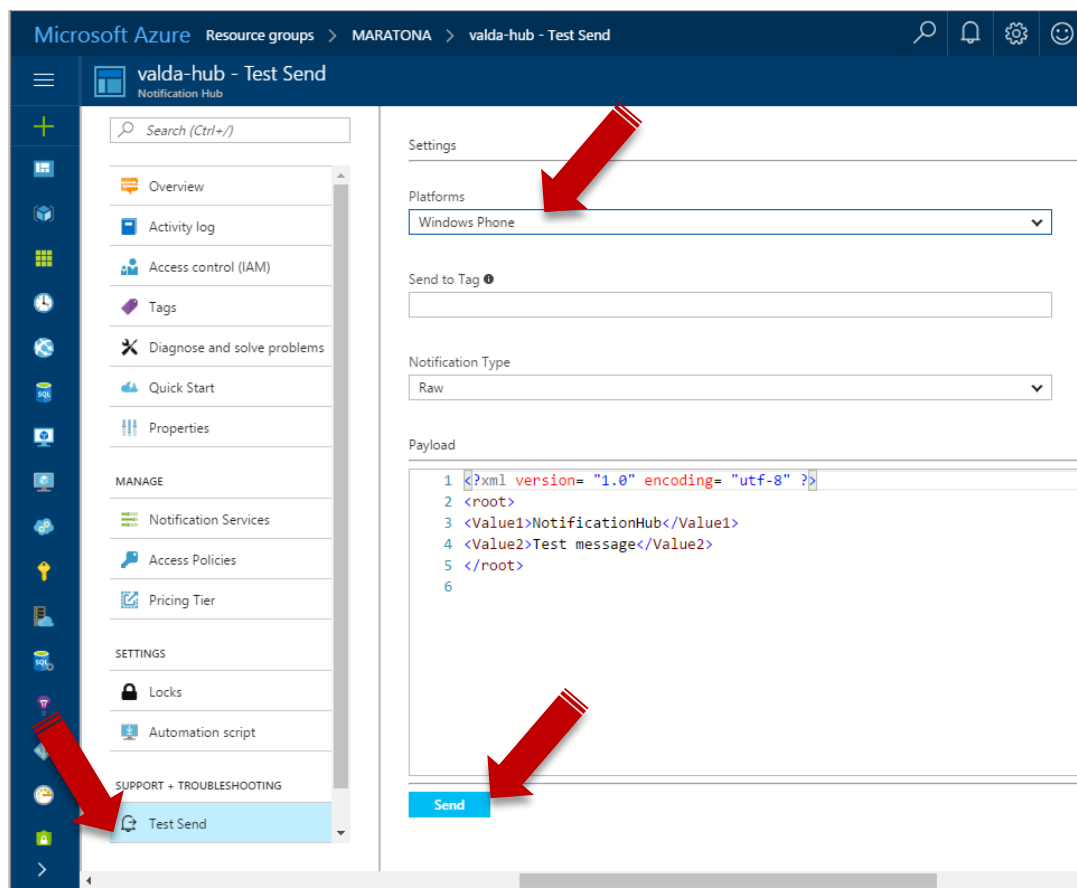
```
await InitNotificationsAsync();
```

Isso garante que o registro de notificação por push é criado ou atualizado sempre que o aplicativo é iniciado. É importante fazer isso para garantir que o canal de notificação por push WNS esteja sempre ativo.

4. No Gerenciador de Soluções do Visual Studio, abra o arquivo **Package.appxmanifest** e defina **Compatível com Notificação do Sistema** como **Sim** em **Notificações**.
5. Compile o aplicativo e verifique se não há erros. O aplicativo cliente agora deve se registrar para receber notificações de modelo do back-end dos Aplicativos Móveis.
6. Repita esta seção para cada projeto do Windows em sua solução.

Exercício 10: Testar as notificações por push em seu aplicativo para Windows (opcional)

1. No Visual Studio, clique com o botão direito do mouse no projeto do Windows e clique em **Definir como projeto de inicialização**.
2. Pressione o botão **Executar** para compilar o projeto e iniciar o aplicativo.
3. No portal do Azure, localize o seu Hub de Notificação criado, clique em **Test Send**, escolha a **plataforma de destino** (no nosso caso Windows ou Windows Phone), e clique em **Send**.



4. No simulador, verifique se uma notificação é recebida.

Resumo

Neste laboratório, você criou um projeto em branco, configurou a SDK do Azure Mobile e o Recebimento de Notificações via Push.

Agora você já pode evoluir e criar experiências 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 #Lab5 da #MaratonaXamarin Intermediária e meu aplicativo agora recebe notificações via Push!