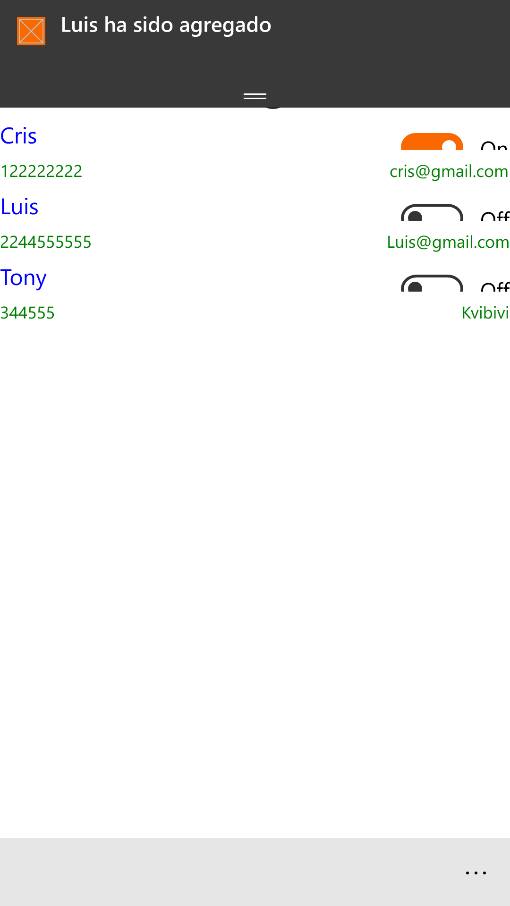
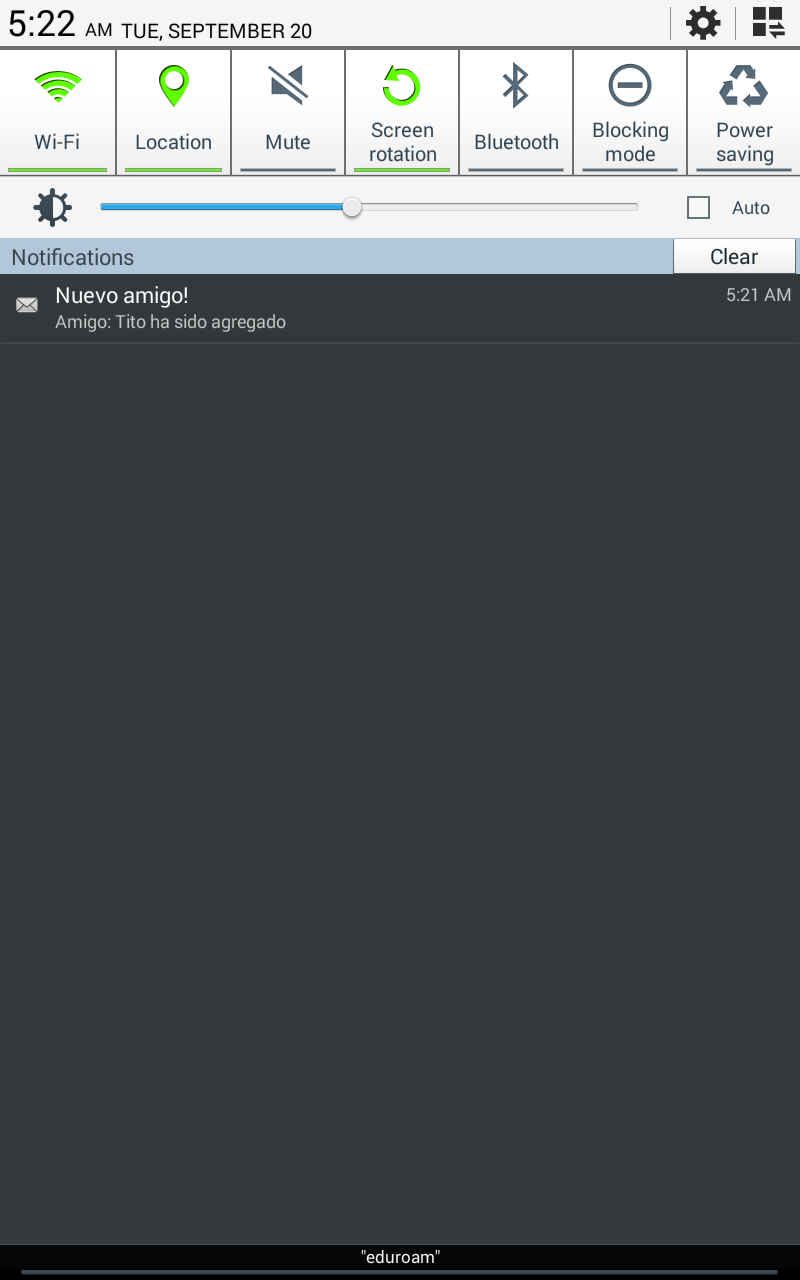
**Práctica 06 – Azure y Notificaciones Push**

El objetivo de esta práctica es integrar Push Notifications al backend de Azure creado en la práctica anterior (P05 – Azure Mobile Services) en nuestra aplicación multiplataforma. En este caso, la notificación será administrada y enviada por el servicio respectivo a la aplicación que se esté ejecutando: Para **Android** será a través del **Google Cloud Messaging (GCM)**, mientras que para **iOS** será el **Apple Push Notification Service (APNS)** y finalmente para **Windows** será el **Microsoft Push Notification Service (MPNS).** Para esta práctica necesitas:

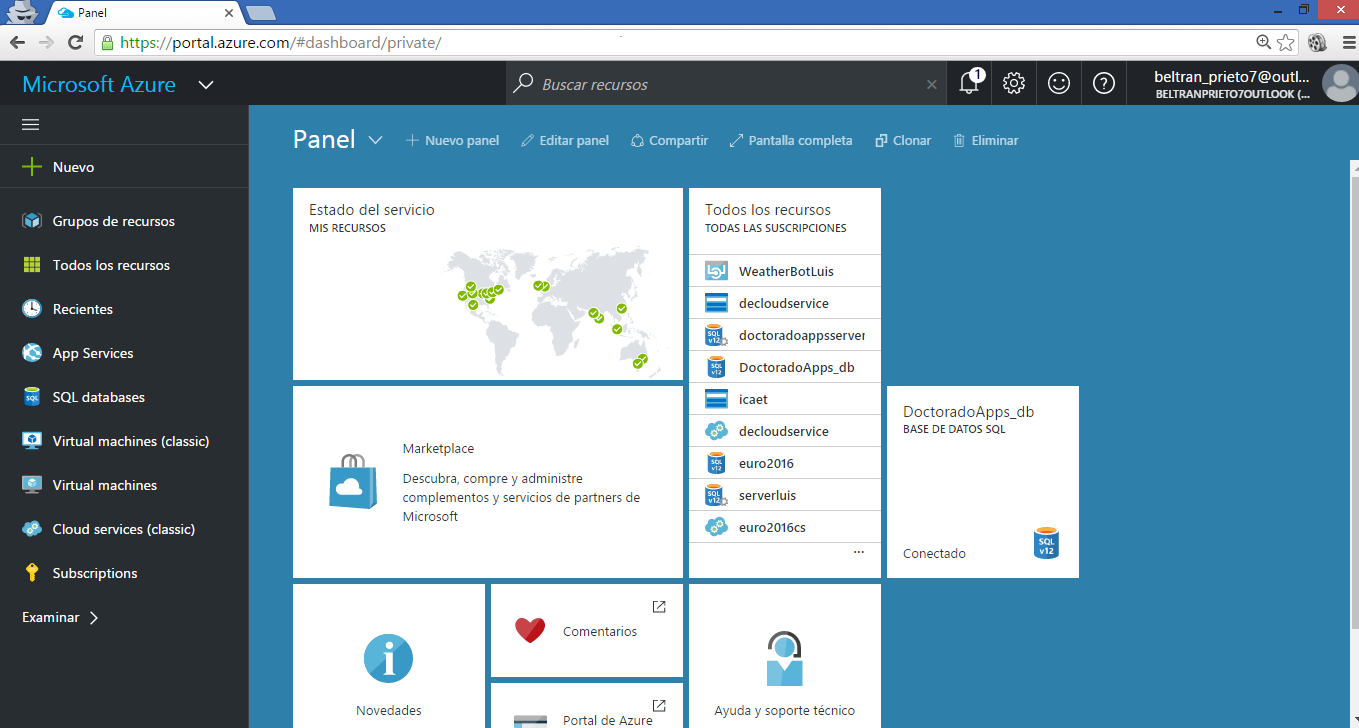
* Una cuenta de Azure
* Opcional: Una cuenta de Google Play Developer
* Opcional: Una cuenta de Windows Store (Windows Dev Center)
* Haber realizado la práctica 5 (si no la has realizado, no te preocupes, descárgala de aquí: <https://github.com/icebeam7/P05-MisAmigosAzureMobile>)

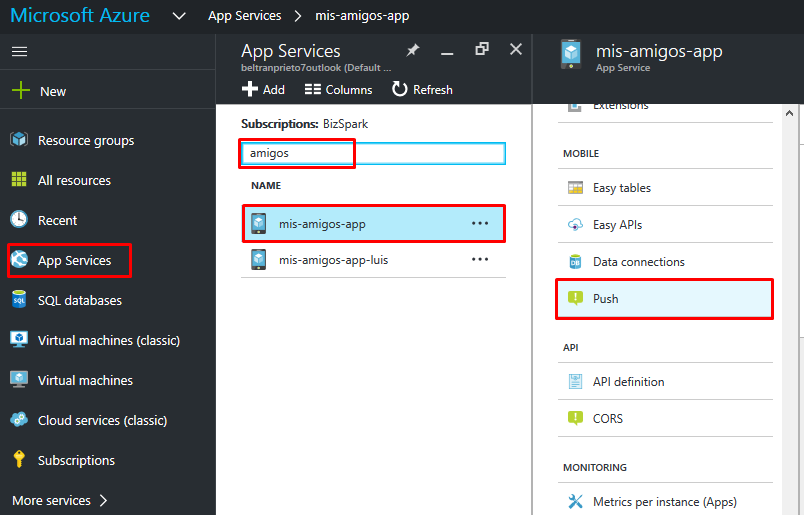
La práctica se basó en el siguiente tutorial:

<https://azure.microsoft.com/en-us/documentation/articles/app-service-mobile-xamarin-forms-get-started-push/>

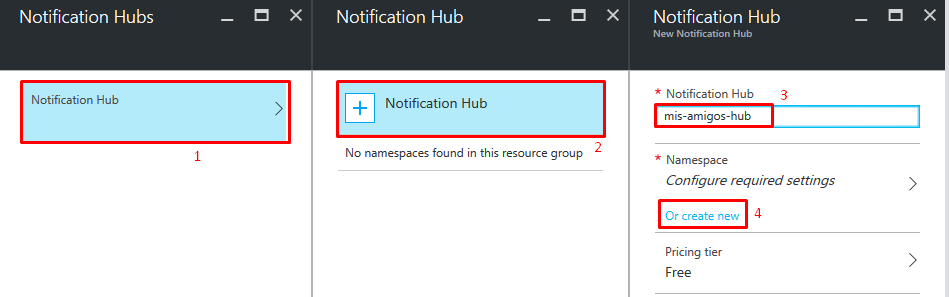
**Paso 1:** Ingresa al portal de Azure e inicia sesión (<http://portal.azure.com>).



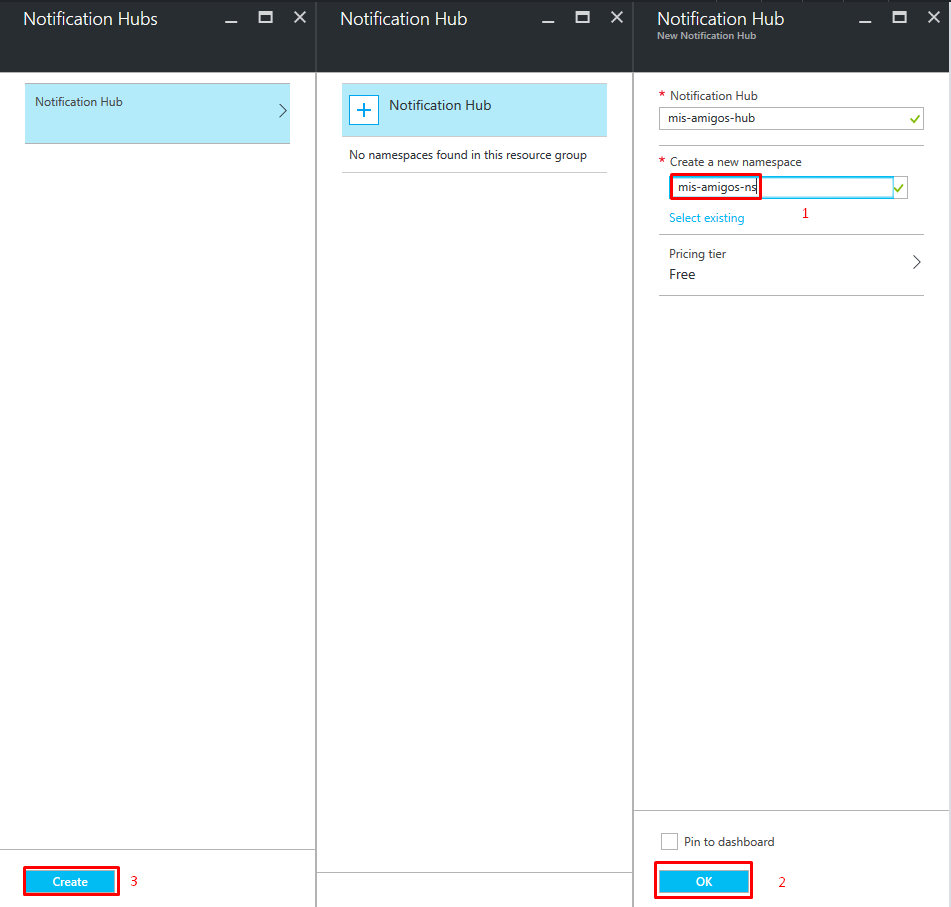
**Paso 2:** Selecciona la categoría App Services (menú izquierdo) y busca (localiza) el servicio creado en la sesión anterior: **mis-amigos-app**. Selecciónalo y se desplegará un nuevo blade, del cual seleccionarás la opción **Push** de la categoría **Mobile**.

****

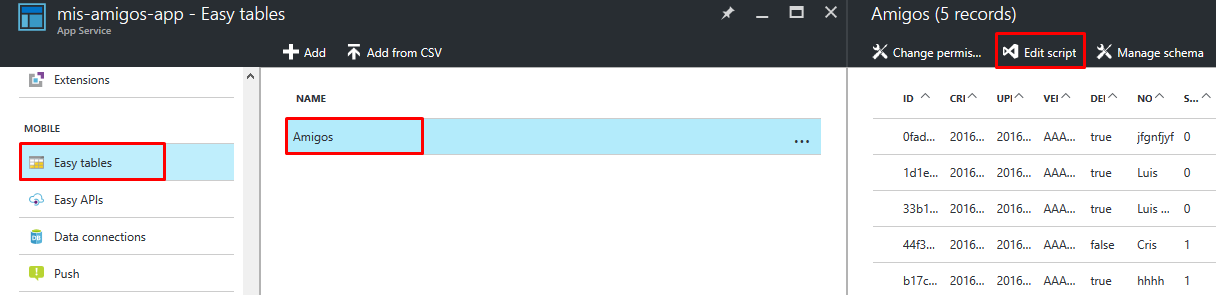
**Paso 3:** Selecciona **Notification Hub** 🡪 **Notification Hub (agregar)** y escribe un nombre para el hub, por ejemplo **mis-amigos-hub**. Da clic en **Create new** en la sección del Namespace.

****

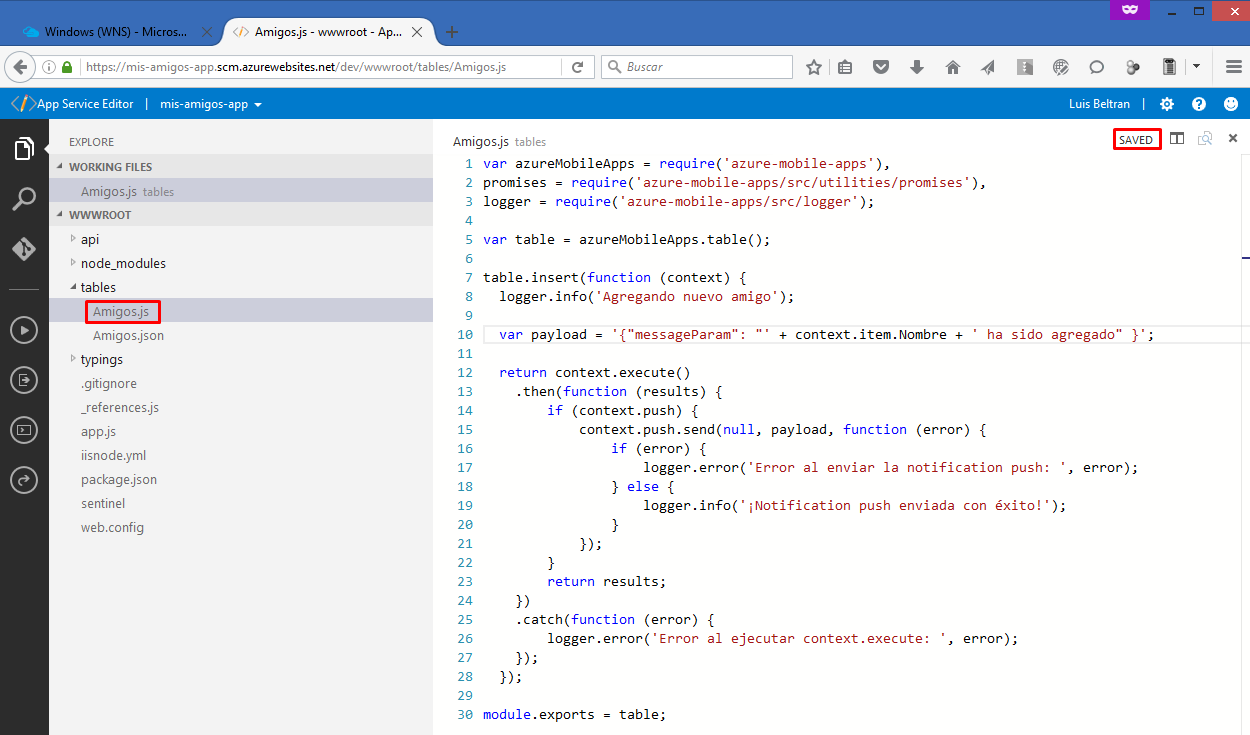
**Paso 4:** Escribe un nombre para el namespace, por ejemplo, **mis-amigos-ns**. Da clic en OK en el primer y tercer blade desplegados.

****

**Paso 5:** Dentro del mismo App Service mis-amigos-app, ahora selecciona **Easy tables** y la tabla **Amigos**. Da clic en el botón **Edit script**.

****

**Paso 6:** ¡En este paso vamos a escribir el código que lanzará una notificación push a los diferentes dispositivos conectados a nuestra app! Verifica que se ha abierto el archivo Amigos.js y que una vez que escribas el código aparezca el mensaje **Saved**. El código lo tienes debajo de la siguiente imagen y responde al evento **insert**, es decir, cada vez que se agregue un nuevo amigo a la tabla, los usuarios de la app serán notificados.

****

El código es el siguiente:

var azureMobileApps = require('azure-mobile-apps'),

promises = require('azure-mobile-apps/src/utilities/promises'),

logger = require('azure-mobile-apps/src/logger');

var table = azureMobileApps.table();

table.insert(function (context) {

logger.info('Agregando nuevo amigo');

var payload = '{"messageParam": "' + context.item.Nombre + ' ha sido agregado" }';

return context.execute()

.then(function (results) {

if (context.push) {

context.push.send(null, payload, function (error) {

if (error) {

logger.error('Error al enviar la notification push: ', error);

} else {

logger.info('¡Notification push enviada con éxito!');

}

});

}

return results;

})

.catch(function (error) {

logger.error('Error al ejecutar context.execute: ', error);

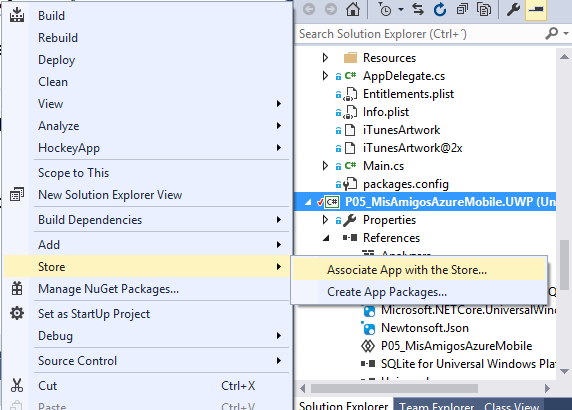
});

});

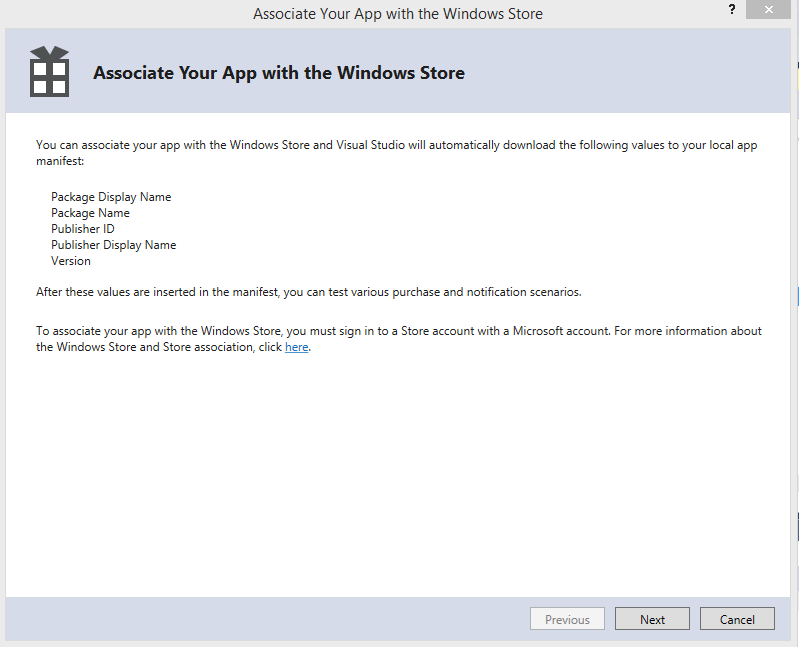
module.exports = table;

**Configurando la aplicación de UWP para que reciba notificaciones (también funciona para los proyectos de Windows y Windows Phone)**

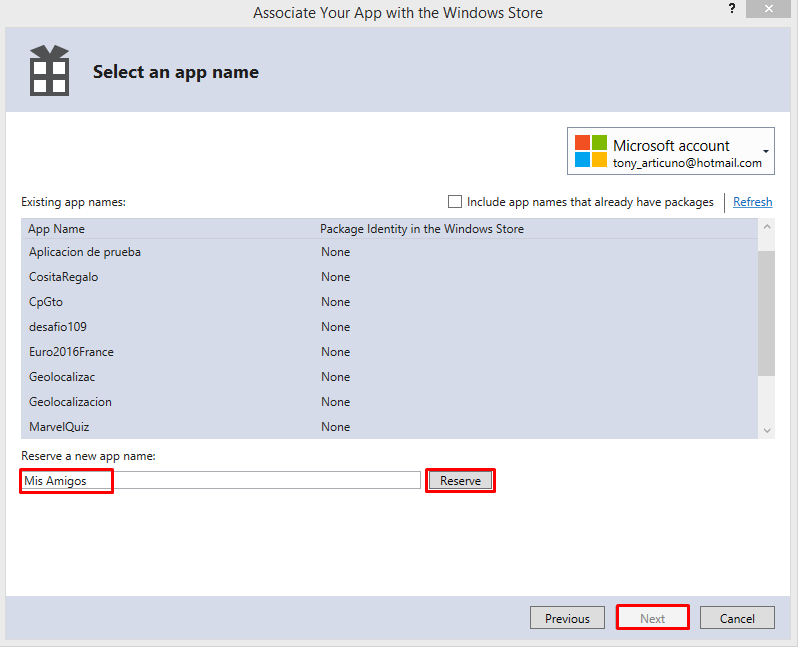
**Paso 7:** Da clic derecho en el nombre del proyecto **UWP** y selecciona **Store 🡪 Associate App with the Store…**

****

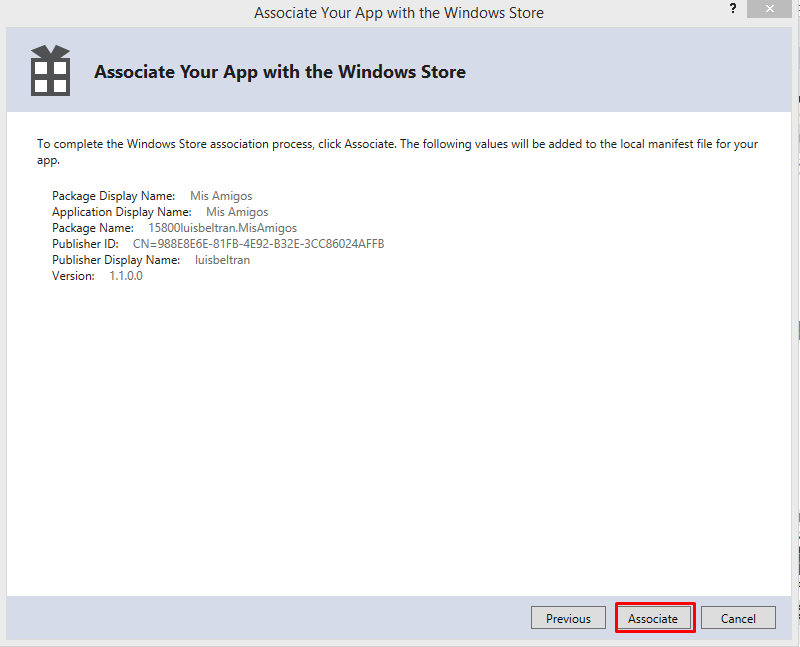
**Paso 8:** Inicia sesión con tu usuario de la cuenta de **Windows Dev Center** y da clic en **Next**.

****

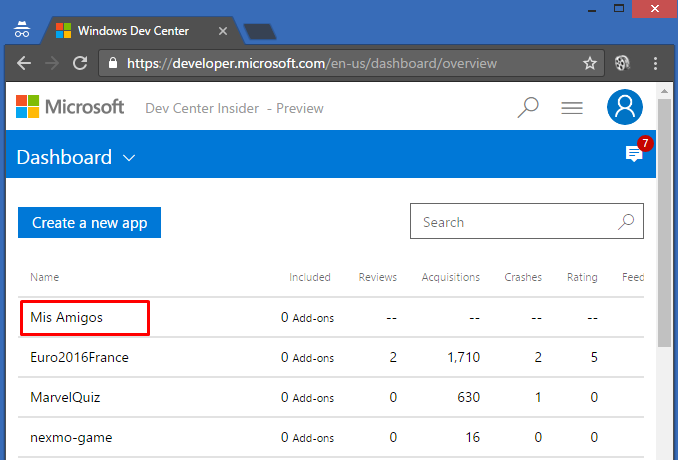
**Paso 9:** Escribe un **nuevo nombre** de app (tiene que ser único) y da clic en **Reserve**. Luego, **Next**.

****

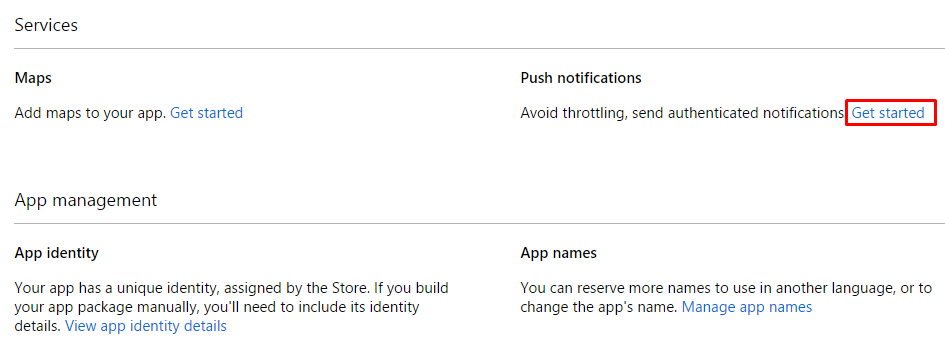
**Paso 10:** Da clic en **Associate** si los datos son correctos.

****

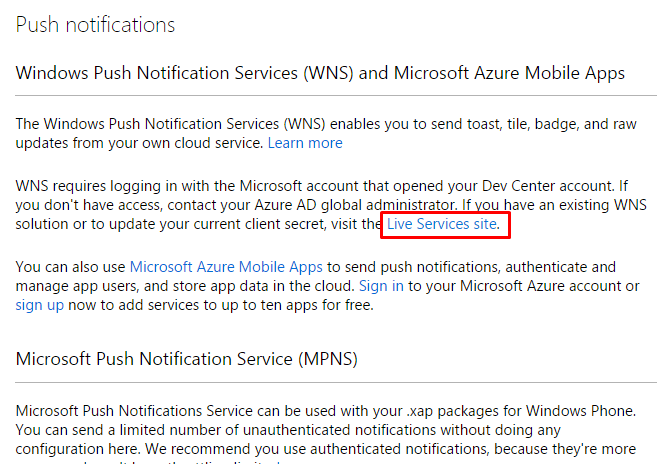
**Paso 11:** Ahora accede a <https://developer.microsoft.com/en-us/dashboard/apps/overview> para ingresar al **Dashboard** en el **Windows Dev Center.** Localiza tu aplicación y **selecciónala**.

****

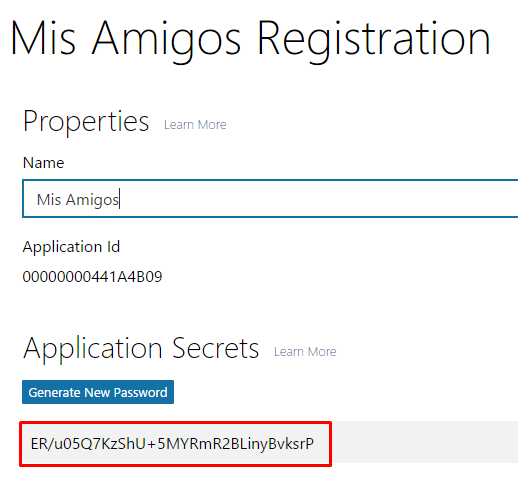
**Paso 12:** En la sección **Services**, da clic en el enlace **Get Started** de la categoría **Push notifications**.

****

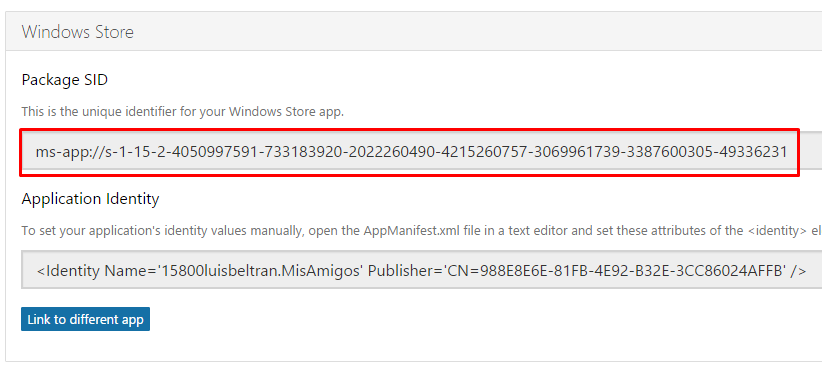
**Paso 13:** Da clic en el enlace **Live Services site**

.****

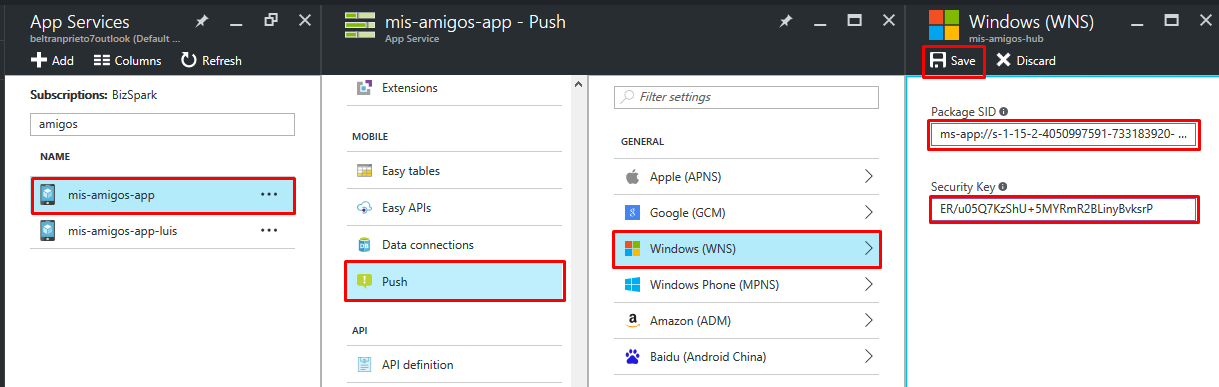
**Paso 14:** Copia el **Application Secret** y pégalo en el bloc de notas. Lo utilizarás en el Paso 16.

****

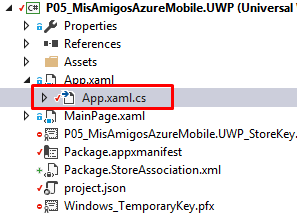
**Paso 15:** También copia el **SID del paquete** y pégalo en el bloc de notas.

****

**Paso 16:** En el portal de Azure, selecciona tu App Service (mis-amigos-app). Dentro de la categoría **Mobile**, selecciona **Push** y da clic en el proveedor **Windows (WNS)**. Pega los valores de los pasos anteriores en los campos respectivos: **Package SID** (Paso 15) y **Security Key** (Paso 14). Da clic en **Save** una vez incorporados estos valores.

****

**Paso 17:** Ahora vamos al código de la aplicación. Abre la clase **App.xaml.cs** del proyecto UWP.

****

**Paso 18:** Agrega los siguientes espacios de nombres a dicha clase.

using Newtonsoft.Json.Linq;

using Microsoft.WindowsAzure.MobileServices;

using System.Threading.Tasks;

using Windows.Networking.PushNotifications;

using P05\_MisAmigosAzureMobile;

**Paso 19:** Agrega el método asíncrono **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}

};

await P05\_MisAmigosAzureMobile.App.AzureService.MobileService.GetPush().RegisterAsync(channel.Uri, templates);

}

**Paso 20:** En el evento OnLaunched, agrega el modificador async y la llamada al método InitNotificationsAsync:

protected async override void OnLaunched(LaunchActivatedEventArgs e)

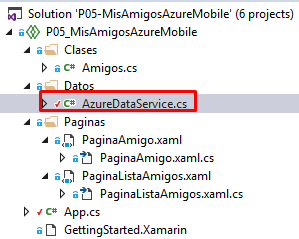
{

await InitNotificationsAsync();

//...

}

**Paso 21:** Ahora abre la clase **AzureDataService** que se localiza en el **SharedProject**.

****

**Paso 22:** Crea un constructor que llame al método **Initialize**.

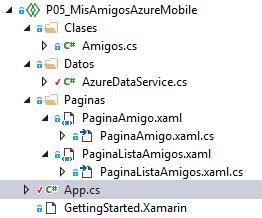
public AzureDataService()

{

Initialize();

}

**Paso 23:** Abre la clase **App.cs** del mismo **SharedProject**.

****

**Paso 24:** Modifica **el constructor** de App y la inicialización del objeto **AzureService**. En el constructor creaba la instancia de AzureService. Ahora se hará a nivel de la clase, es decir, el código quedará de la siguiente manera:

public static AzureDataService AzureService = new AzureDataService();

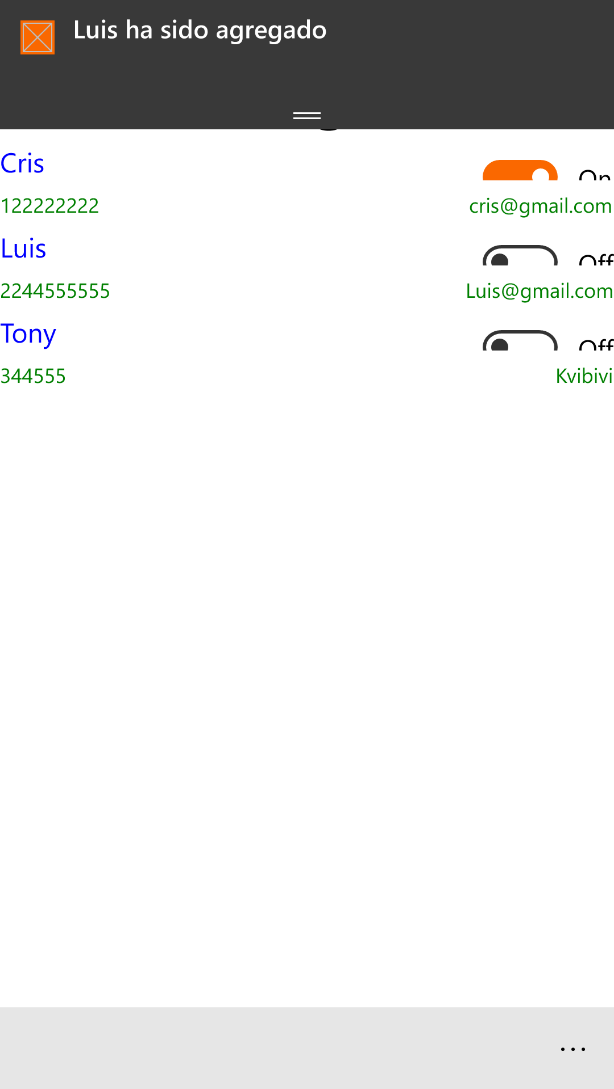
public App ()

{

MainPage = new NavigationPage(new PaginaListaAmigos());

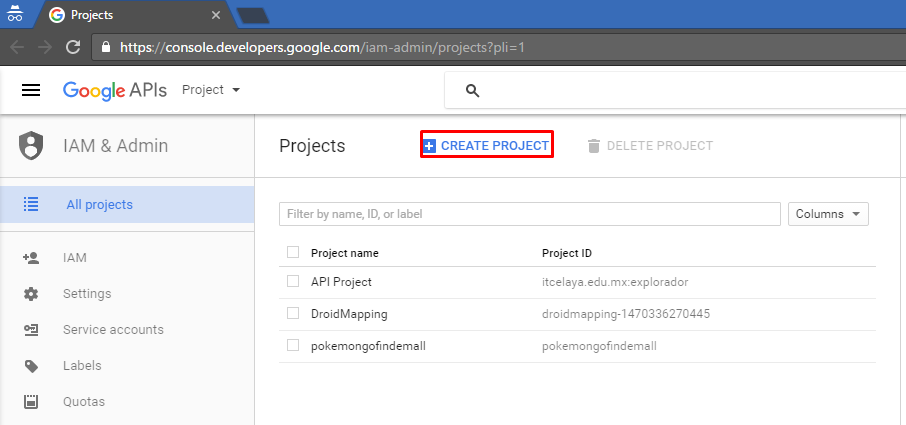
}

**Paso 25:** Compila y ejecuta la aplicación. Agrega un nuevo amigo. En el momento en que sea insertado en la aplicación, verás que aparece una notificación indicando el éxito de la operación. ¡Genial!

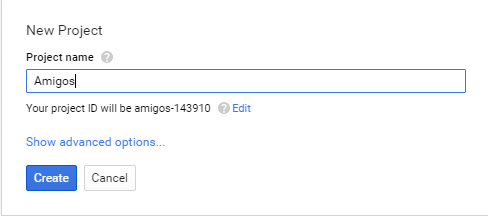


**Configurando la aplicación de Android para que reciba notificaciones**

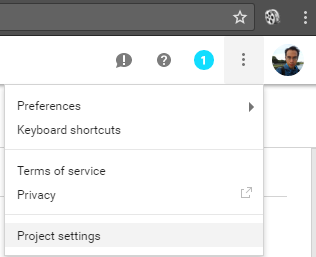
**Paso 26:** Accede a la consola de desarrolladores de Google <http://console.developers.google.com> e inicia sesión. **Crea un nuevo proyecto**.

****

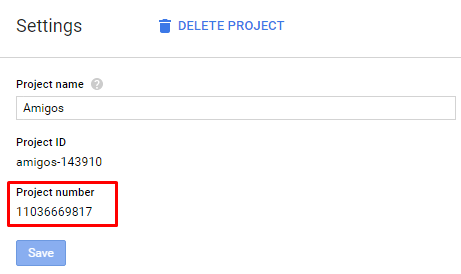
**Paso 27:** Escribe **un nombre** para el proyecto (por ejemplo, **Amigos**) y da clic en **Create**.

****

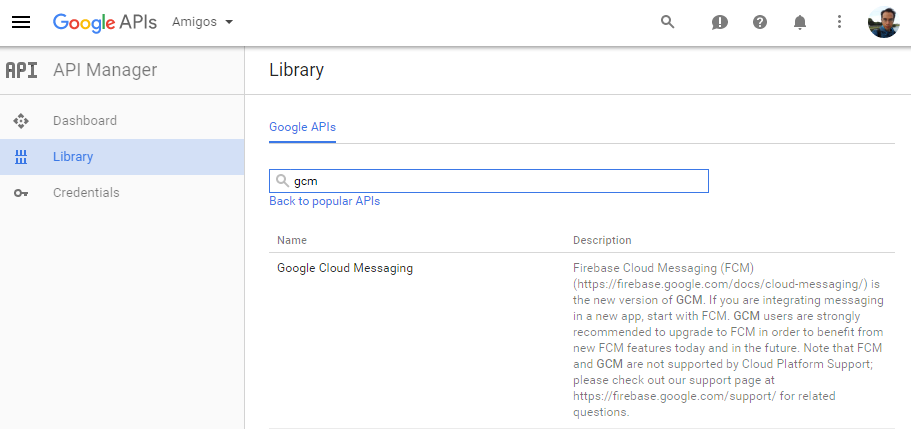
**Paso 28:** Una vez creado el proyecto, accede a la configuración dando clic en el ícono **Utilities and more** y seleccionando **Project settings**:

****

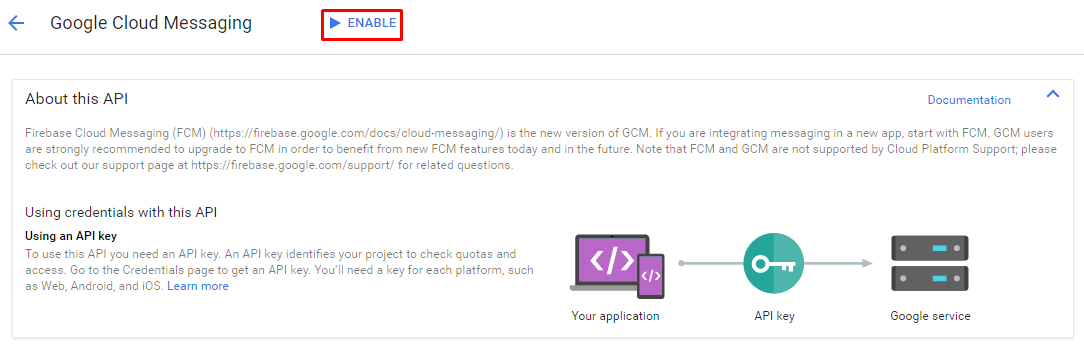
**Paso 29:** Copia y pega en un bloc de notas el **Project Number**. Lo utilizarás más adelante.



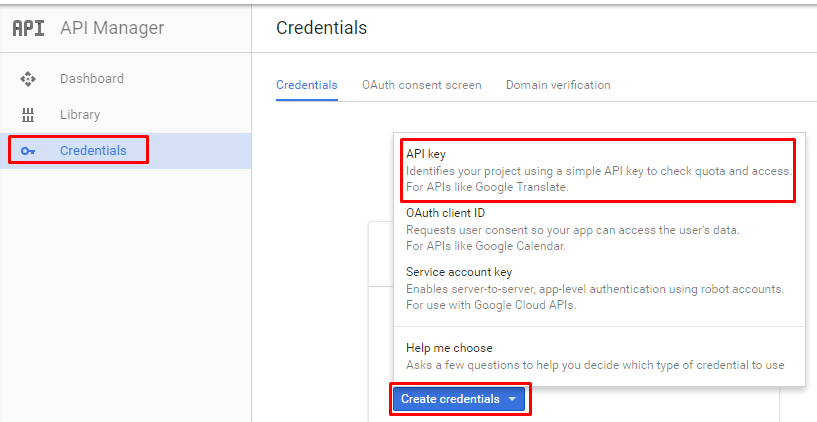
**Paso 30:** Regresa al dashboard de tu aplicación. Selecciona **Library** en el menú izquierdo, escribe **gcm** en el cuadro de búsqueda y da clic en **Google Cloud Messaging**.

****

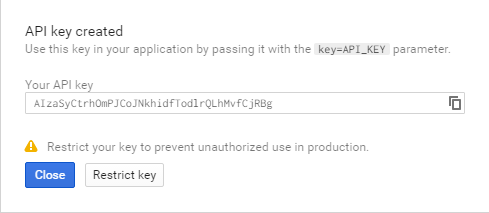
**Paso 31:** Da clic en **Enable** para habilitar el servicio de Google Cloud Messaging, el cual hará posible el envío de notificaciones push a tu aplicación de Android.

****

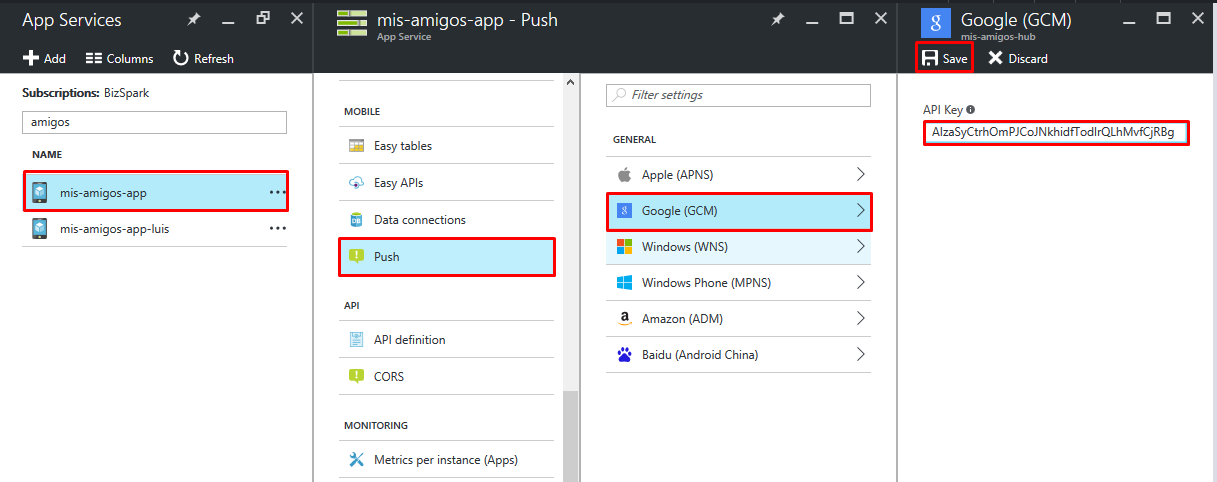
**Paso 32:** Del menú izquierdo, selecciona **Credentials** y da clic en el botón **Create credentials**. Selecciona **API key**. Si aparece un tipo de credencial, selecciona **Server**.

****

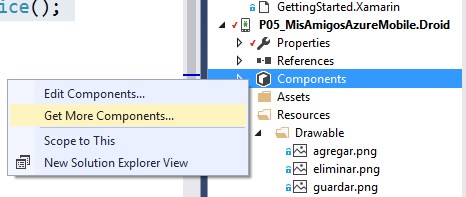
**Paso 33:** Copia el **API key** que aparece, lo utilizarás en el paso siguiente. Da clic en Close.

****

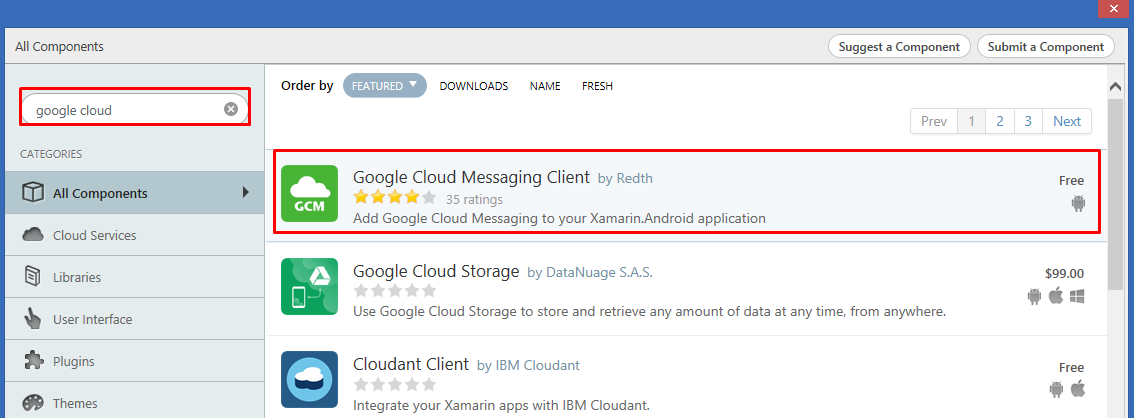
**Paso 34:** En el portal de Azure, ahora selecciona el proveedor **Google (GCM)** y pega el **API key** que obtuviste en el paso anterior. Da clic en **Save**.

****

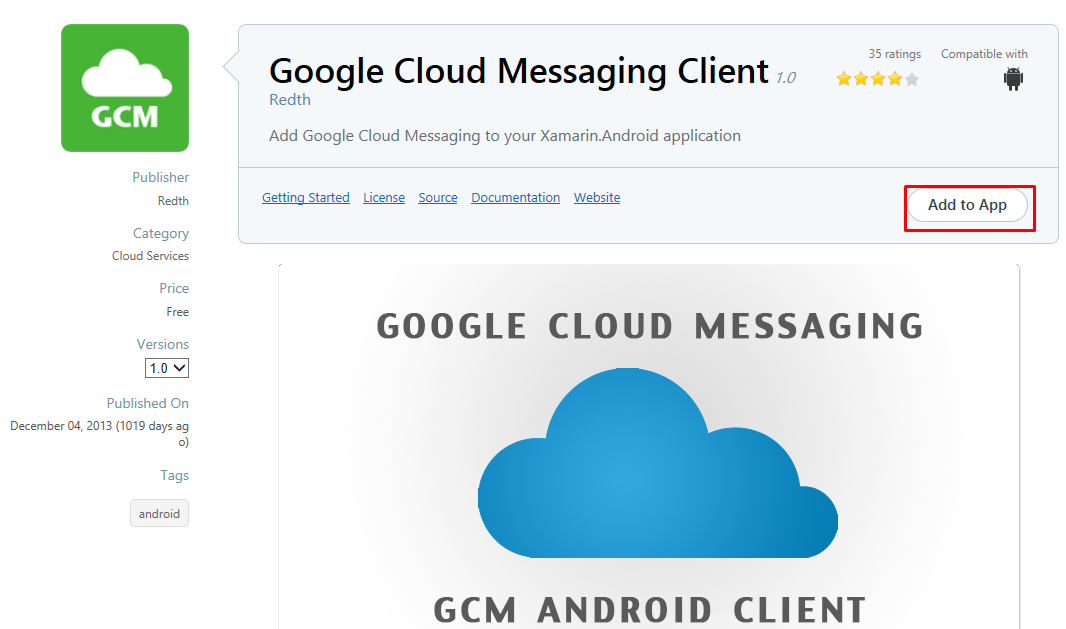
**Paso 35:** Regresa al proyecto de Xamarin y da clic derecho en **Components**, dentro del proyecto de Android. Selecciona **Get More Components…**



**Paso 36.** Busca el componente **Google Cloud Messaging** y **selecciona** el resultado que aparece.

****

**Paso 37:** Instálalo dando clic en **Add to App**.

****

**Paso 38:** Abre el archivo MainActivity del proyecto de Android y realiza los siguientes cambios:

1. Agrega el namespace Gcm.Client:

using Gcm.Client;

1. En el método **OnCreate**, coloca el siguiente código después de la línea de **LoadApplication**

try

{

GcmClient.CheckDevice(this);

GcmClient.CheckManifest(this);

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");

}

1. A nivel de clase, fuera de cualquier método, agrega el siguiente código:

private void CreateAndShowDialog(String message, String title)

{

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.SetMessage(message);

builder.SetTitle(title);

builder.Create().Show();

}

static MainActivity instance = null;

public static MainActivity CurrentActivity

{

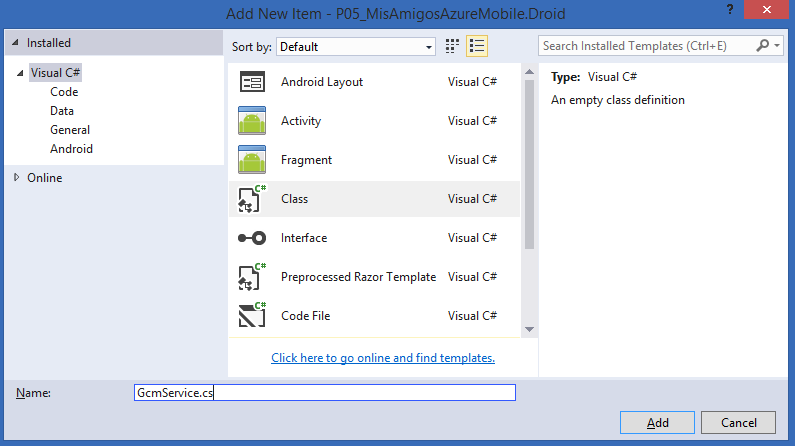
get { return instance; }

}

1. De regreso al método OnCreate, simplemente agrega esta línea justo al inicio (antes de base.OnCreate(this);):

instance = this;

**Paso 39:** Agrega una clase al proyecto de Android (en raíz). Su nombre es **GcmService**:

****

**Paso 40:** A continuación se presenta el código completo de la clase. **NOTA:** El arreglo **SENDER\_IDS** contiene una lista de apps autorizadas para enviar notificaciones push. En este arreglo debes colocar el **Project Number** del paso 29.

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;

[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")]

namespace P05\_MisAmigosAzureMobile.Droid

{

[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[] { " 11036669817" };

}

[Service]

public class GcmService : GcmServiceBase

{

public static string RegistrationID { get; private set; }

public GcmService() : base(PushHandlerBroadcastReceiver.SENDER\_IDS) { }

protected override void OnRegistered(Context context, string registrationId)

{

Log.Verbose("PushHandlerBroadcastReceiver", "GCM Registered: " + registrationId);

RegistrationID = registrationId;

var push = P05\_MisAmigosAzureMobile.App.AzureService.MobileService.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();

}

}

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("Nuevo amigo!", "Amigo: " + 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)

{

var notificationManager = GetSystemService(Context.NotificationService) as NotificationManager;

var uiIntent = new Intent(this, typeof(MainActivity));

NotificationCompat.Builder builder = new NotificationCompat.Builder(this);

var notification = builder.SetContentIntent(PendingIntent.GetActivity(this, 0, uiIntent, 0))

.SetSmallIcon(Android.Resource.Drawable.SymActionEmail)

.SetTicker(title)

.SetContentTitle(title)

.SetContentText(desc)

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

.SetAutoCancel(true).Build();

notificationManager.Notify(1, notification);

}

protected override void OnUnRegistered(Context context, string registrationId)

{

Log.Error("PushHandlerBroadcastReceiver", "Unregistered RegisterationId : " + registrationId);

}

protected override void OnError(Context context, string errorId)

{

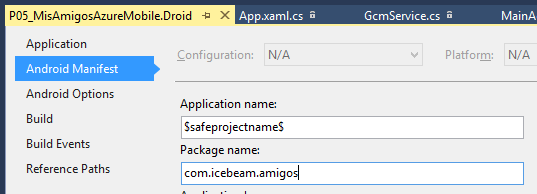
Log.Error("PushHandlerBroadcastReceiver", "GCM Error: " + errorId);

}

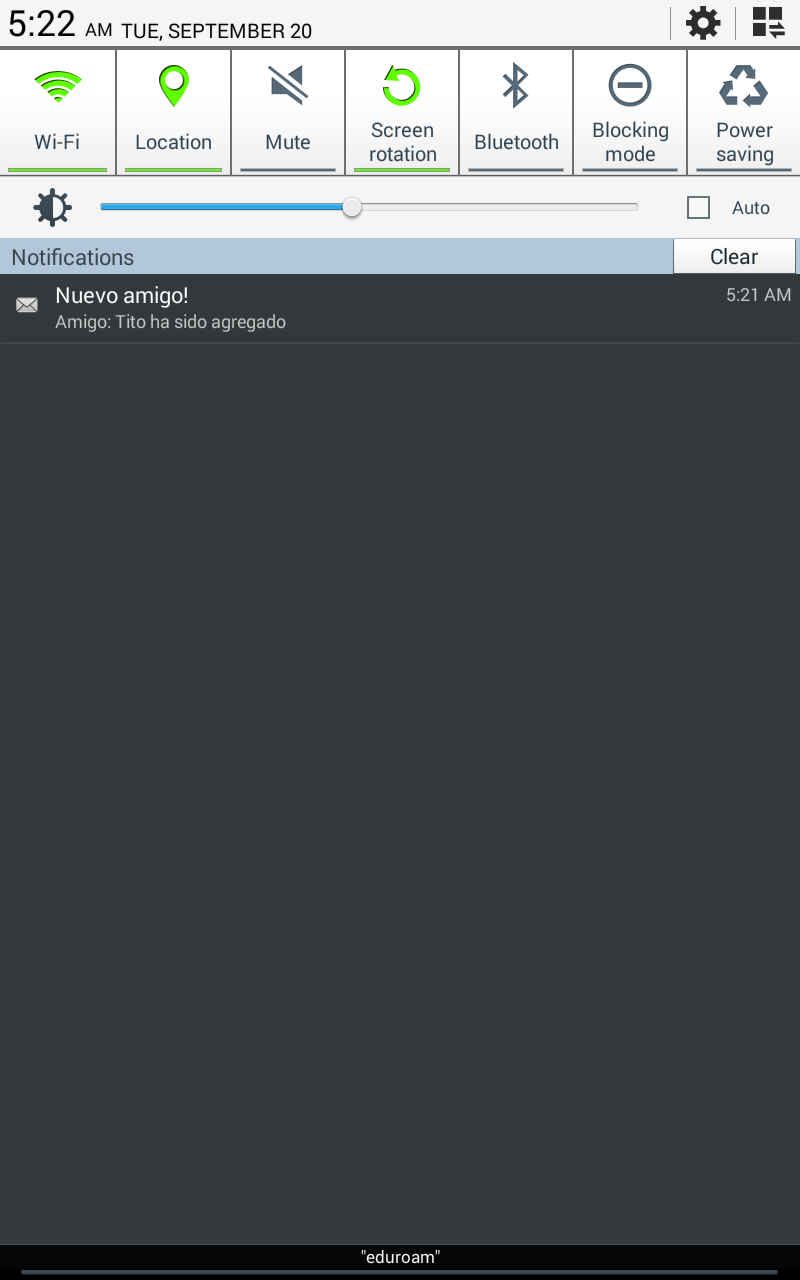
}

}

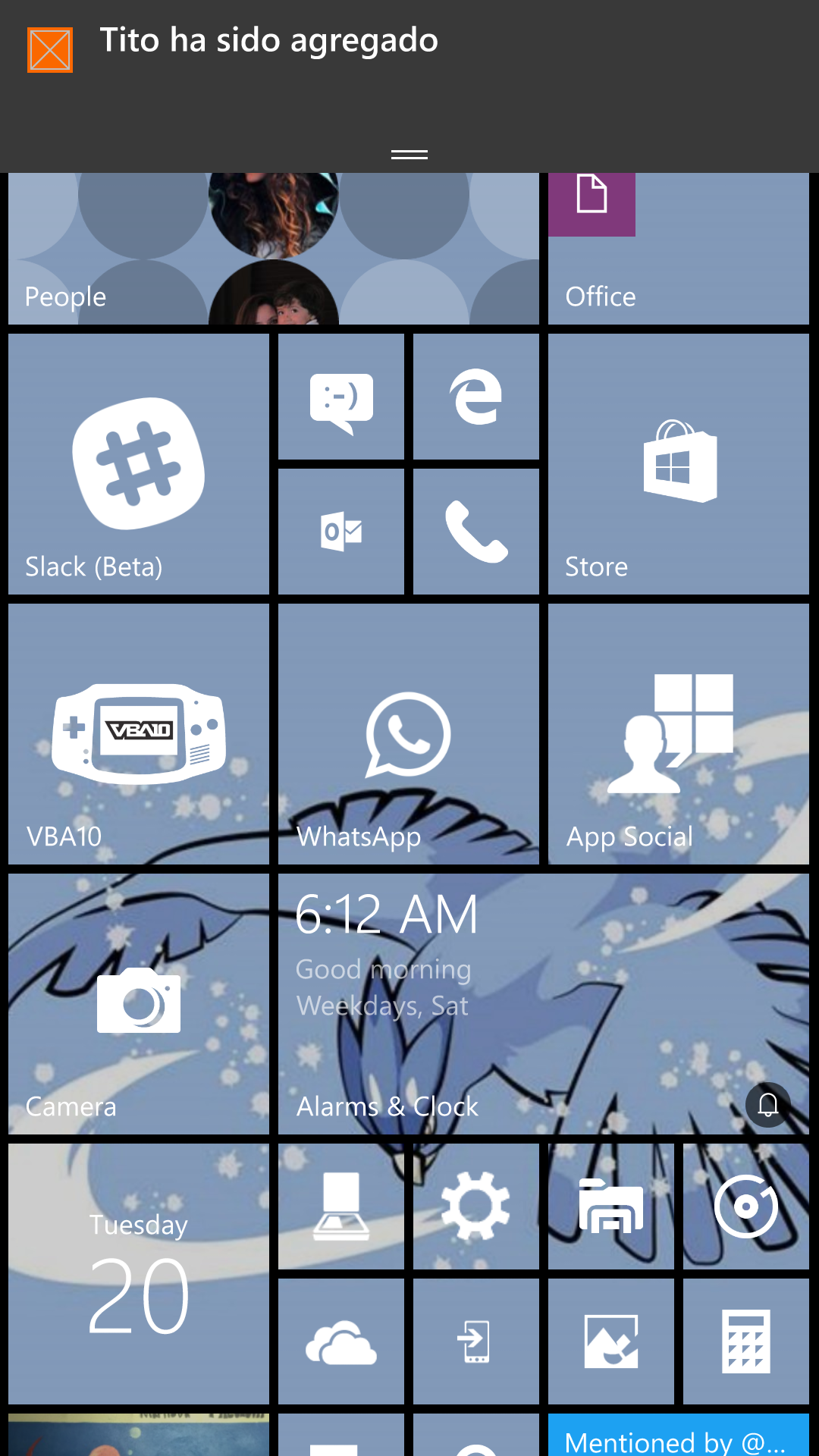
**Paso 41:** Finalmente, revisa si en el **Android Manifest** (dentro de las propiedades del proyecto) se ha eliminado el nombre del paquete (a mí me pasó por alguna razón). Simplemente agrega uno (por ejemplo, **com.icebeam.amigos**).

****

**Paso 42:** Compila y ejecuta la aplicación. Agrega un nuevo amigo. En el momento en que sea insertado en la aplicación, verás que aparece una notificación indicando el éxito de la operación. ¡Genial!



Si configuraste las notificaciones Push para Windows, ¡observa que también recibe la notificación! Es decir, agregar un nuevo amigo desde cualquier app (Android o Windows) enviará la notificación a los usuarios que tengan instalada la app.

****

**¡Felicidades! Has integrado notificaciones Push a tu aplicación móvil multiplataforma con Xamarin, por lo que funciona tanto para Windows como para Android (iOS es mi asignatura pendiente ☺, luego la realizamos).**

Si no puedes esperar, en los siguientes enlaces tienes una guía interesante para implementar Notificaciones Push en iOS y el Apple Notification Push Service (APNS), además de más información sobre las otras plataformas (Android y UWP):

<https://azure.microsoft.com/en-us/documentation/articles/app-service-mobile-xamarin-forms-get-started-push/#optional-configure-and-run-the-ios-project>

<http://stackoverflow.com/questions/29029709/how-to-use-push-notifications-in-xamarin-forms>

<http://xamarinhelp.com/push-notifications/>