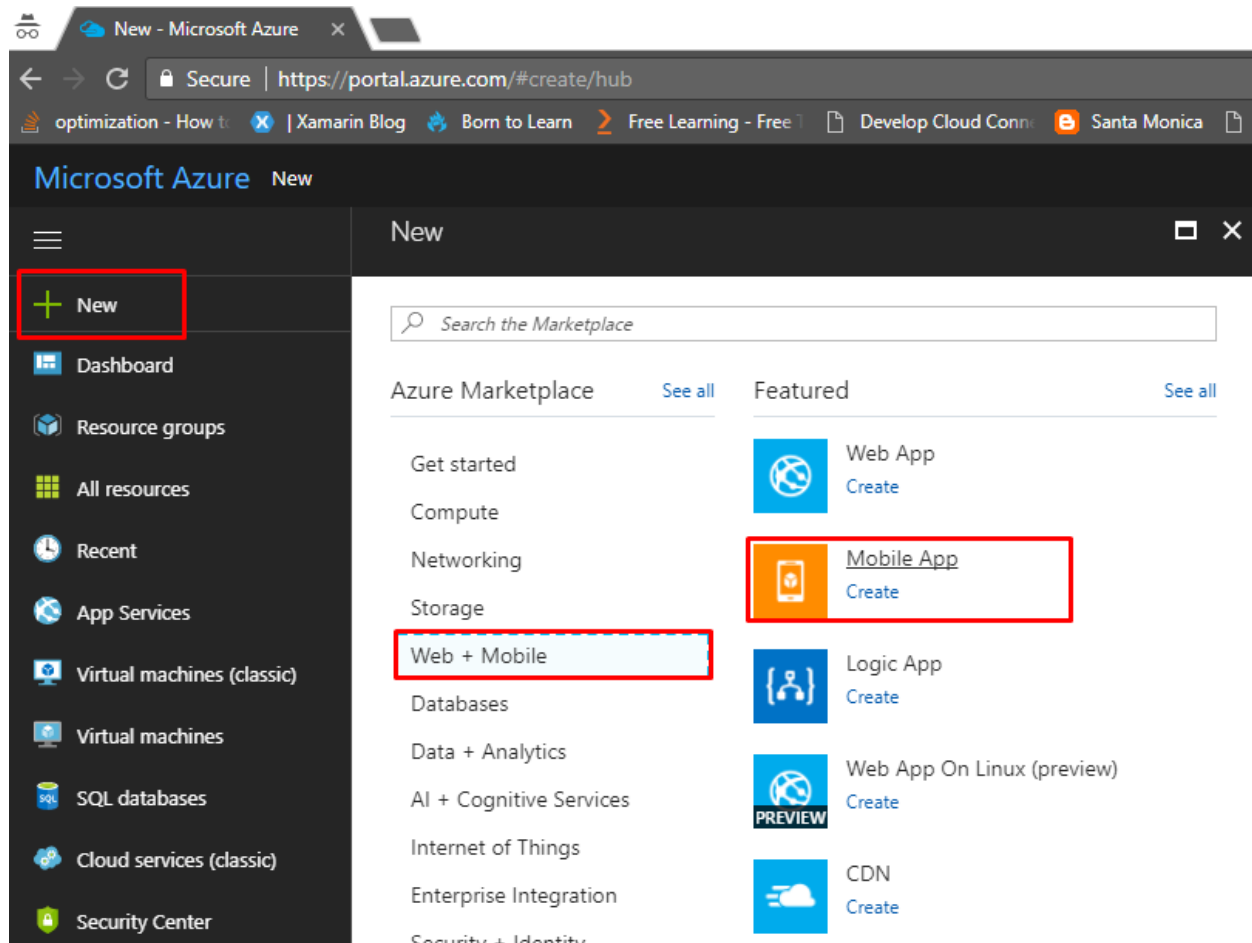



Práctica: Easy Tables con Azure y Xamarin – Autor: Luis Beltrán

En esta sesión vamos a crear una aplicación móvil multiplataforma con Xamarin que consume una base de datos creada en Azure a través del servicio EasyTables de Azure. **Para esta sesión se requiere tener una cuenta activa de Azure.**

Paso 1. Da clic en **Nuevo** → **Web y Móvil** → **Mobile App**






Paso 2. En el asistente selecciona **Crear**









Mobile App

Microsoft



Accelerate your mobile app development with this turnkey way to structure storage, authenticate users, and send push notifications. With native and cross-platform SDKs for iOS, Android, Windows, and HTML, as well as a powerful and flexible REST API, Mobile Apps empowers you to build connected applications for any platform and deliver a consistent experience across devices.

- Integrate with SQL, Oracle, SAP, MongoDB, and more.
- Make your app work offline and sync.
- Connect to on-premises data.
- Leverage enterprise single sign-on with Active Directory.
- Integrate with social providers like Facebook, Twitter, and Google.
- Broadcast push notifications across platforms, with customer segmentation.
- Gain insights with mobile analytics.
- Auto-scale to millions of devices.



PUBLISHER	Microsoft
USEFUL LINKS	Documentation Service Overview Pricing Details

Create

Paso 3. Ingresas los datos del servicio móvil: El **nombre de la app** debe ser único y se recomienda crear un **grupo de recursos** y un **plan de app service** específico para la app; una vez establecidos estos elementos, da clic en **Crear**.

The image shows two side-by-side windows from the Azure portal. The left window is titled 'Mobile App' and the right is 'App Service plan'.

Mobile App window:

- * App name** 1: Input field with 'futbolapp-luisb' and a green checkmark.
- * Subscription**: Dropdown menu showing 'Visual Studio Enterprise'.
- * Resource Group** 2: Radio buttons for 'Create new' (selected) and 'Use existing'. Below is an input field with 'futbolapp-rg' and a green checkmark.
- * App Service plan/Location** 3: A blue button with a right arrow and the text 'ServicePlanb9e3f7e9-bdef(South ...)'.
- Application Insights**: Toggle switch set to 'Off'.
- Pin to dashboard** 5: A checkbox that is currently unchecked.
- Create** 6: A blue button at the bottom left.
- Automation options**: A link at the bottom right.

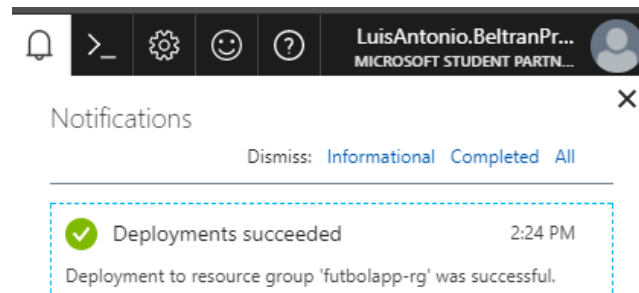
App Service plan window:

- Create New** 4: A blue button with a plus icon and the text 'Create New'.
- ServicePlanb9e3f7e9-bdef(S1) (New)**: The name of the selected plan.
- South Central US**: The location of the plan.
- New Plan**: A link to create a new plan.

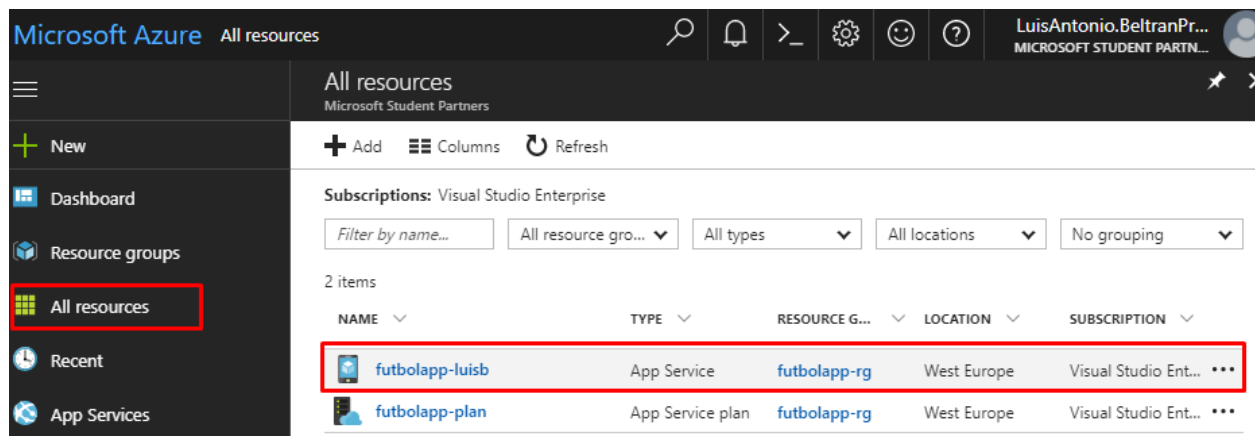
The image shows the 'New App Service Plan' window in the Azure portal.

- * App Service plan** 1: Input field with 'futbolapp-plan' and a green checkmark.
- * Location** 2: Dropdown menu showing 'West Europe'.
- * Pricing tier** 3: Input field with 'S1 Standard' and a right arrow.
- OK** 4: A blue button at the bottom.

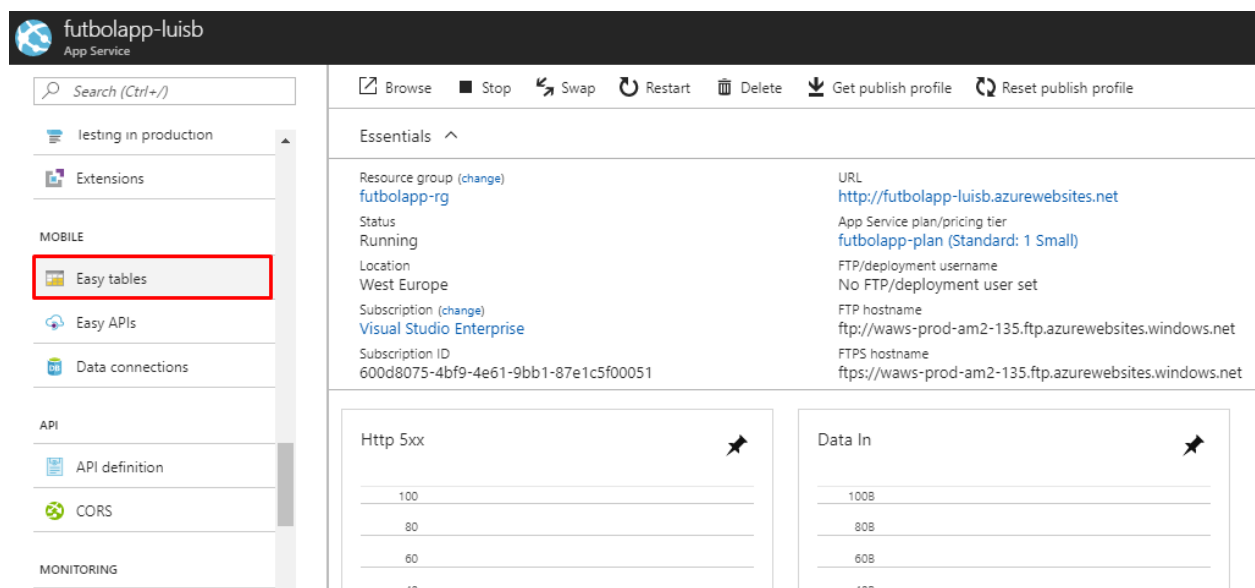
Paso 4. Cuando la app sea creada, recibirás una notificación



Paso 5. De regreso al dashboard, da clic en **Todos los recursos** y selecciona el **App Service** que acaba de ser creado



Paso 6. En la pantalla principal localiza la categoría **Móvil** y selecciona **Tablas Fáciles** de la lista



Paso 7. Al ser la primera vez que se ingresa a este servicio en esta app, deberás crearlo y configurarlo. Da clic en el mensaje **Click here to continue**.

+ Add **↗ Add from CSV**

i Need to configure Easy Tables/Easy APIs - Click here to continue →

NAME

No results.


Paso 8. Da clic en el **enlace** del paso 1 para **crear una base de datos**.

Easy Tables

Easy Tables is not supported on your current App Service app. Please initialize your App Service app for Easy Tables support.

1

Connect a database



You need a database to use Easy Tables. [Click here to create one.](#)

Paso 9. Da clic en Add para agregar una conexión de datos.

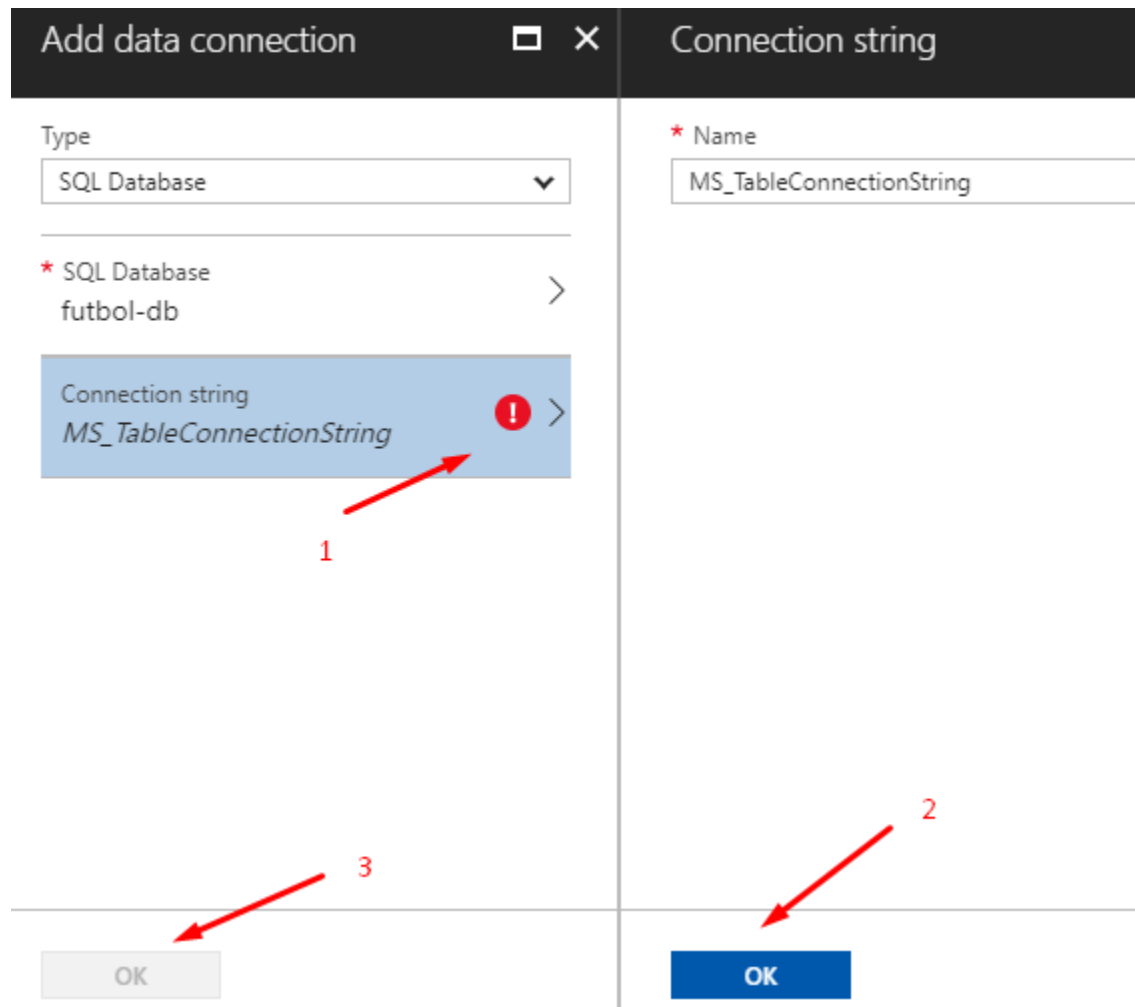
Data Connections	
+ Add	
NAME	TYPE
No Data Connections	

Paso 10. Da clic en **SQL Database** para configurar elementos requeridos, luego **crea una base de datos**, coloca un **nombre** y selecciona **servidor de destino** para crear uno. Escribe el **nombre del servidor** y configura un **usuario** y **contraseña** para acceder; también elige una **ubicación** lo más cercana posible a donde estarán los clientes que consumirán la app.

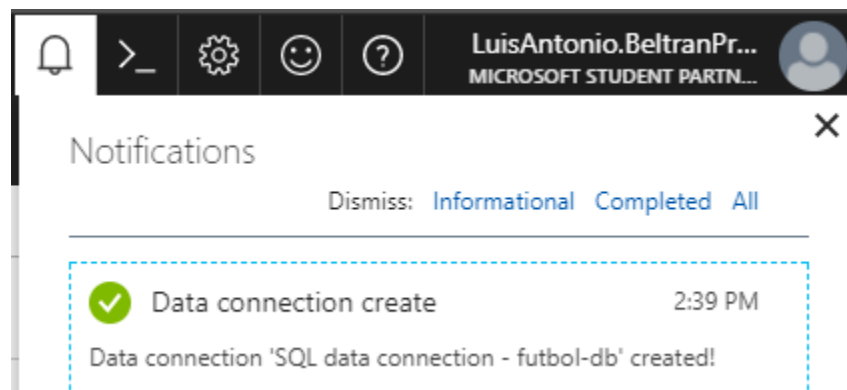
The screenshot shows the 'Add data connection' dialog with the 'SQL Database' tab selected. The breadcrumb navigation at the top reads: 'ables > Data Connections > Add data connection > Database > SQL Database > Server > New server'. A search bar on the right says 'Search resources'. The dialog is divided into three main sections: 'Add data connection', 'Database', and 'SQL Database'. In the 'Add data connection' section, a dropdown menu shows 'SQL Database' (arrow 1), and below it are links for 'SQL Database' and 'Connection string', both labeled 'Configure required settings'. The 'Database' section shows a '+ Create a new database' button (arrow 2) and 'No databases found' (arrow 3). The 'SQL Database' section contains fields for 'Name' (filled with 'futbol-db', arrow 4), 'Target server' (labeled 'Configure required settings', arrow 5), 'Pricing tier' (labeled 'Configure required settings', arrow 6), and 'Collation' (filled with 'SQL_Latin1_General_CP1_CI_AS', arrow 7). At the bottom right is a 'Select' button (arrow 11). An 'OK' button is at the bottom left.

The screenshot shows the 'New server' dialog. The breadcrumb navigation at the top reads: 'Server > New server'. The dialog has two main sections: 'Server' and 'New server'. The 'Server' section shows a '+ Create a new server' button (arrow 5) and 'No servers found'. The 'New server' section contains fields for 'Server name' (filled with 'futbol-server-luisb', arrow 6), 'Server admin login' (filled with 'icebeam', arrow 7), 'Password' (masked with dots, arrow 8), 'Confirm password' (masked with dots, arrow 9), 'Location' (dropdown menu showing 'West Europe', arrow 10), and a checkbox 'Allow azure services to access server' which is checked. A 'Select' button is at the bottom right.

Paso 11. De regreso en la pantalla de conexión de datos, da clic en la **cadena de conexión** y acepta el nombre que aparece por defecto.



Paso 12. Recibirás una **notificación** cuando la base de datos sea creada



Paso 13. Ingresa de nuevo en el servicio, localiza **Tablas Fáciles** y da clic nuevamente en el mensaje **Click here to continue**.

AddAdd from CSV

Need to configure Easy Tables/Easy APIs - Click here to continue →

NAME

No results.

Paso 14. La conexión de datos ha sido creada. Ahora solo marca el **checkbox** del paso 2 y da clic en **Iniciar App**.

Easy Tables

Easy Tables is not supported on your current App Service app. Please initialize your App Service app for Easy Tables support.

1

Connect a database

SQL

You already have a data connection

2

Initialize your App Service app to use Easy Tables. Note that this will overwrite your existing site contents.

☒

I acknowledge that this will overwrite all site contents.

Initialize App

Paso 15. Recibirás una notificación cuando el servicio haya sido inicializado

LuisAntonio.BeltranPr...
MICROSOFT STUDENT PARTN...

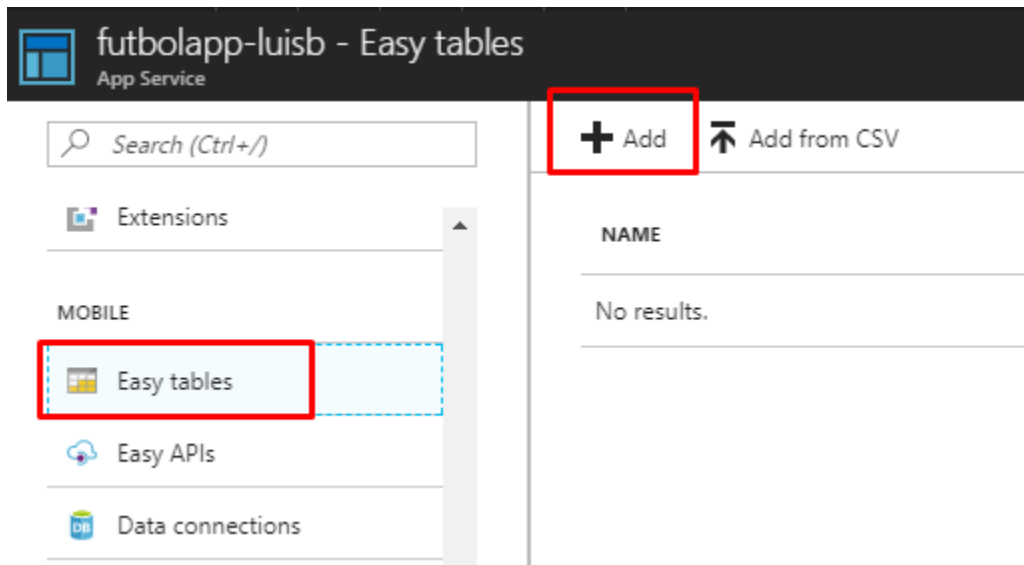
Notifications

Dismiss: Informational Completed All

App Service app backend initialize2:45 PM

App Service app backend for 'futbolapp-luisb' initialized!

Paso 16. Da clic en **Agregar** después de seleccionar **Tablas Fáciles**.



Paso 17. Agrega la tabla Equipo

Add a table

*

Name

Equipo

✓

Insert permission

Allow anonymous access

▼

Update permission

Allow anonymous access

▼

Delete permission

Allow anonymous access

▼

Read permission

Allow anonymous access

▼

Undelete permission

Allow anonymous access

▼

OK

Paso 18. En la página principal del servicio, **localiza la URL** y cópiala. La utilizaremos más adelante.

Microsoft Azure futbolapp-luisb

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

DEPLOYMENT

Quickstart

Deployment credentials

Deployment slots

Deployment options

Continuous Delivery (Preview)

SETTINGS

Essentials

Resource group (change) [futbolapp-rg](#)

Status: Running

Location: West Europe

Subscription (change) [Visual Studio Enterprise](#)

Subscription ID: 600d8075-4bf9-4e61-9bb1-87e1c5f00051

URL: <http://futbolapp-luisb.azurewebsites.net>

App Service plan/pricing tier: [futbolapp-plan \(Standard: 1 Small\)](#)

FTP/deployment username: No FTP/deployment user set

FTP hostname: ftp://waws-prod-am2-135.ftp.azurewebsites.windows.net

FTPS hostname: ftps://waws-prod-am2-135.ftp.azurewebsites.windows.net

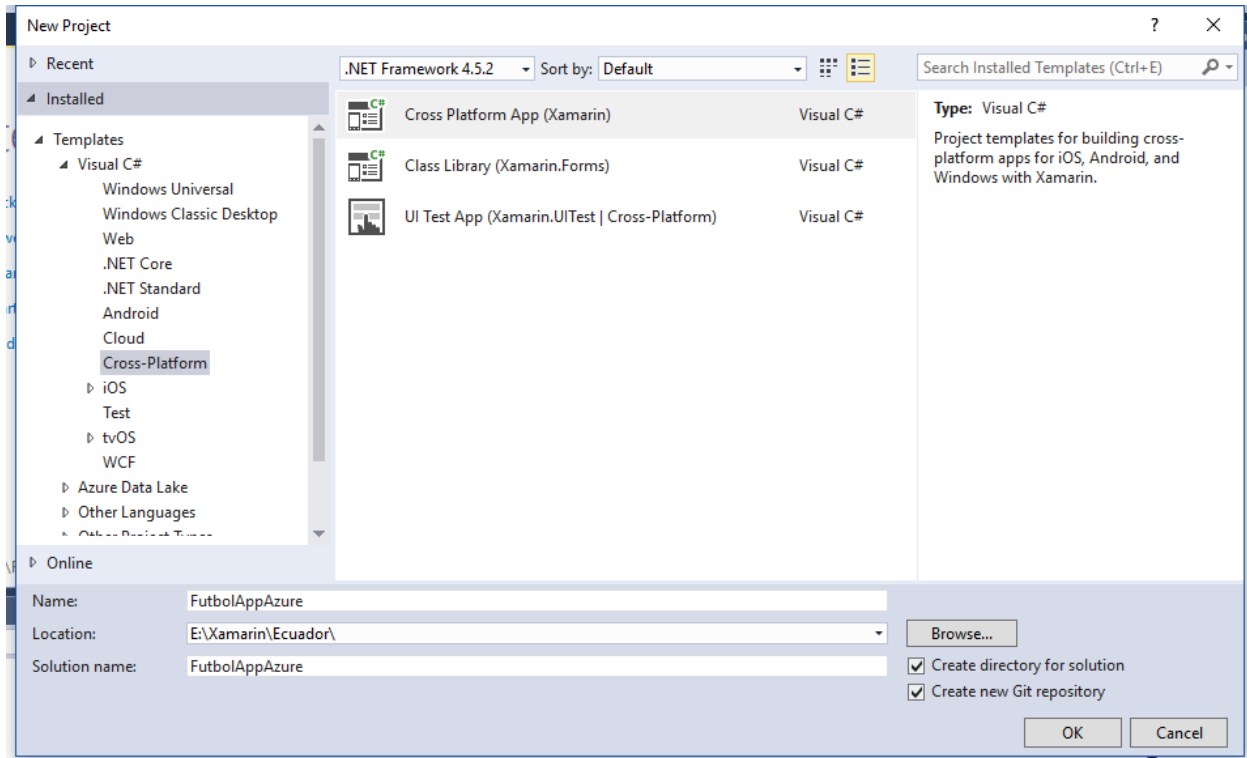
Http 5xx

Data In

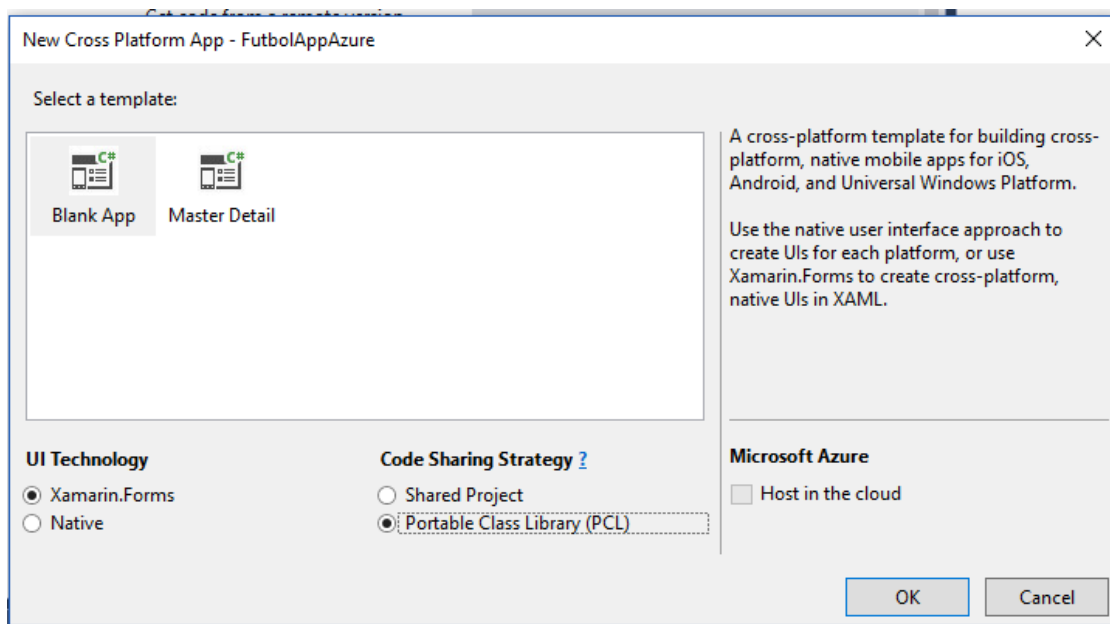
HTTP SERVER ERRORS

DATA IN

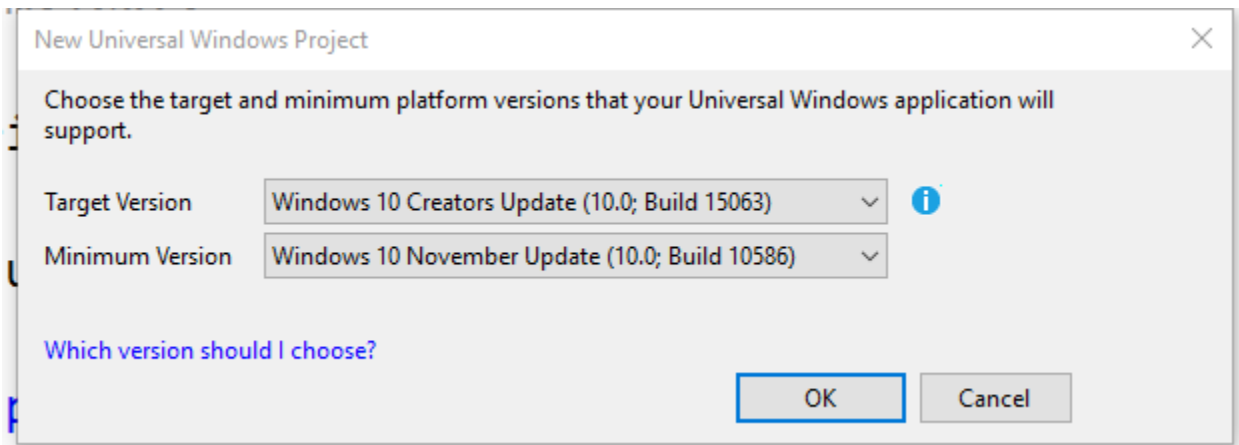
Paso 19. Crea un nuevo proyecto de la categoría **Cross-Platform** selecciona **Aplicación multiplataforma (Xamarin.Forms o nativa)** y coloca el nombre de proyecto **FutbolAppAzure**. Además, la ruta del proyecto debe ser una ubicación corta para evitar problemas de ruta larga.



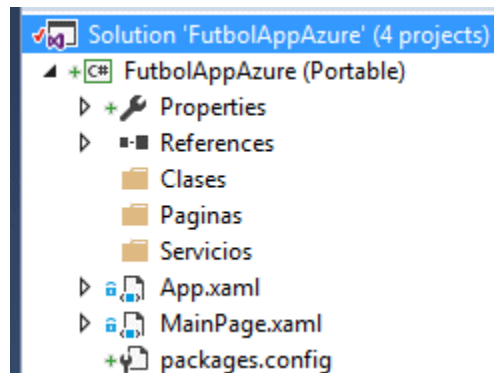
Paso 20. Selecciona la plantilla **Aplicación en blanco**, la tecnología de IU **Xamarin.Forms** y la estrategia de uso compartido de código **Biblioteca de clases portátil (PCL)**. Da clic en **OK**.



Paso 21. Si tienes instalado el SDK de Windows 10, aparecerá la ventana de selección del Target y Minimum Version. Selecciónalas a conveniencia, según la versión que tengas instalada.



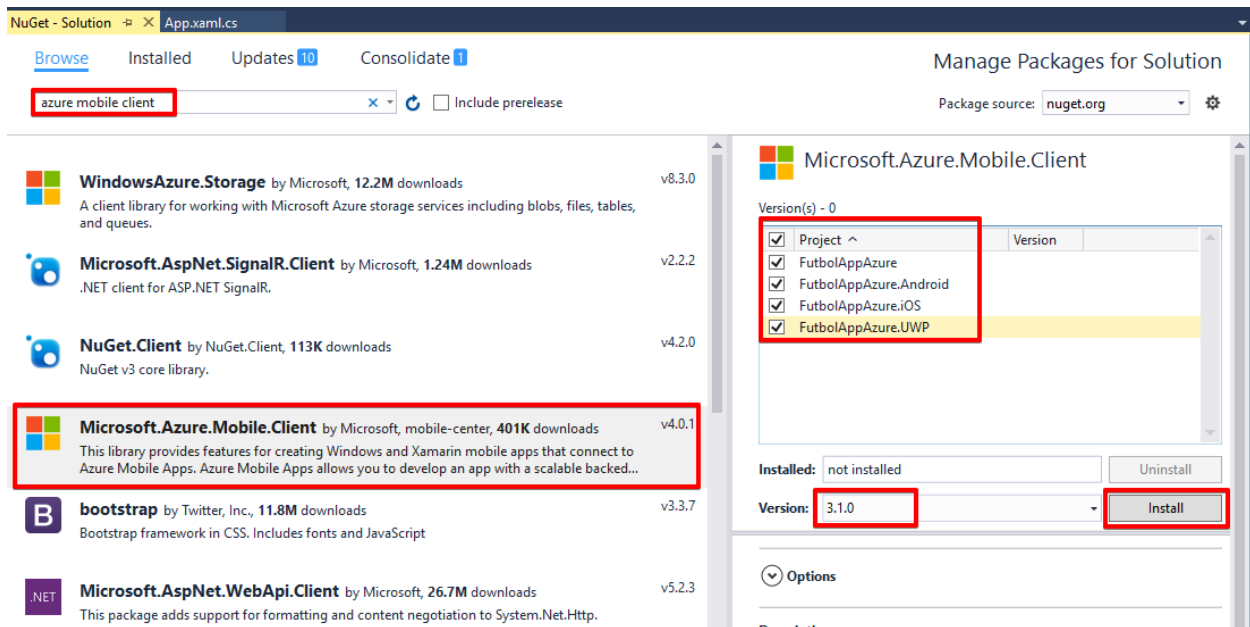
Paso 22. Da clic derecho en el **proyecto portable** y selecciona **Agregar → Nueva carpeta**. Agrega las carpetas **Clases**, **Paginas** y **Servicios**.



Paso 23. Da clic derecho en la solución y selecciona la opción **Administrar paquetes NuGet para la solución...**

Paso 24. Agrega los siguientes paquetes Nuget al proyecto.

a) Microsoft.Azure.Mobile.Client versión 3.1.0



Paso 25. En la carpeta **Clases** agrega una nueva clase llamada **Equipo**, con el siguiente código:

```
using System;
using Microsoft.WindowsAzure.MobileServices;

namespace FutbolAppAzure.Clases
{
    [DataTable("Equipo")]
    public class Equipo
    {
        public string Id { get; set; }
        public string Nombre { get; set; }
        public string Pais { get; set; }

        [Version]
        public string Version { get; set; }
    }
}
```

Paso 26. En la carpeta **Servicios** agrega una nueva clase llamada **ServicioAzure**. El código es el siguiente:

```
using Microsoft.WindowsAzure.MobileServices;
using FutbolAppAzure.Clases;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace FutbolAppAzure.Servicios
{
    public class ServicioAzure
    {
        public MobileServiceClient Cliente;
        private IMobileServiceTable<Equipo> TablaEquipo;

        public ServicioAzure()
        {
            Cliente = new MobileServiceClient("TU-URL-DEL-PASO-18");
            TablaEquipo = Cliente.GetTable<Equipo>();
        }

        public async Task AgregarEquipo(Equipo equipo)
        {
            try { await TablaEquipo.InsertAsync(equipo); }
            catch (Exception ex) { }
        }

        public async Task ModificarEquipo(Equipo equipo)
        {
            try { await TablaEquipo.UpdateAsync(equipo); }
            catch (Exception ex) { }
        }

        public async Task EliminarEquipo(Equipo equipo)
        {
            try { await TablaEquipo.DeleteAsync(equipo); }
            catch (Exception ex) { }
        }

        public async Task<List<Equipo>> ObtenerEquipos()
        {
            return await TablaEquipo.OrderBy(x => x.Nombre).ToListAsync();
        }

        public async Task<Equipo> ObtenerEquipo(string id)
        {
            var equipo = TablaEquipo.Where(x => x.Id == id);

            if (equipo != null)
            {
                var list = await equipo.ToListAsync();
                return list.Count > 0 ? list[0] : null; ;
            }
            else return null;
        }
    }
}
```

Paso 27. Modifica App.xaml.cs del proyecto portable:

```
using FutbolAppAzure.Servicios;
using Xamarin.Forms;

namespace FutbolAppAzure
{
    public partial class App : Application
    {
        public static ServicioAzure ServicioAzure = new ServicioAzure();

        public App()
        {
            InitializeComponent();
            MainPage = new NavigationPage(new Paginas.PaginaListaEquipos());
        }
    }
}
```

Paso 28. Crea dos páginas de tipo **ContentPage** en la carpeta **Paginas**:

a) PaginaListaEquipos: Esta página contiene la lista de equipos registrados en la base de datos mediante un **ListView**. Usando **bindings** en el **DataTemplate** indicamos las propiedades de Equipo que vamos a mostrar, por ejemplo, **Nombre** y **Pais**. En el código de C# se muestra la forma de obtener los equipos consultando la base de datos y asignándolos al **ItemSource** del **ListView**. Además, al **seleccionar** un equipo se navega a **PaginaEquipo** para mostrar la información de dicho elemento elegido por el usuario para editarlo, eliminarlo o consultarlo simplemente. Si el usuario da clic en **Agregar**, se navega a la misma **PaginaEquipo**, con la diferencia de que el usuario podrá dar de alta un nuevo registro.

Código XAML:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="FutbolAppAzure.Paginas.PaginaListaEquipos"
    BackgroundColor="White">
    <ScrollView>
        <StackLayout>
            <ListView x:Name="lsvEquipos" ItemSelected="lsvEquipos_ItemSelected">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell Text="{Binding Nombre}" Detail="{Binding Pais}"
                            TextColor="Blue" DetailColor="Gray" />
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Agregar" Text="Agregar" Order="Primary" Priority="0"
            Clicked="Agregar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;
using FutbolAppAzure.Clases;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace FutbolAppAzure.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaListaEquipos : ContentPage
    {
        public PaginaListaEquipos()
        {
            InitializeComponent();
        }

        protected async override void OnAppearing()
        {
            base.OnAppearing();

            lsvEquipos.ItemsSource = await App.ServicioAzure.ObtenerEquipos();
        }

        private async void lsvEquipos_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            try
            {
                Equipo equipo = (Equipo)e.SelectedItem;
                await Navigation.PushAsync(new PaginaEquipo(equipo));
            }
            catch (Exception ex)
            {
            }
        }

        private async void Agregar_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginaEquipo(new Equipo()));
        }
    }
}
```


b) PaginaEquipo: En esta página se muestra la información específica de un equipo para editarlo, eliminarlo o dar de alta un nuevo elemento. Se usa **Binding** para asignar los valores de las propiedades a campos específicos. En los botones de la barra de herramientas se llaman a los métodos definidos en la clase **BaseDatos** a fin de actualizar el almacenamiento agregando, eliminando o modificando un registro particular.

Código XAML:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="FutbolAppAzure.Paginas.PaginaEquipo">
    <StackLayout Padding="10" Spacing="10">
        <Label Text="Nombre:" FontSize="Medium" TextColor="Black"
            HorizontalOptions="Start" Margin="10" HorizontalTextAlignment="Start"/>
        <Entry Text="{Binding Nombre}" Placeholder="Nombre" WidthRequest="200"
            FontSize="20" TextColor="Black" BackgroundColor="White" HorizontalOptions="Start"
            Margin="10" HorizontalTextAlignment="Start" FontAttributes="Bold"/>
        <Label Text="Pais:" FontSize="Medium" TextColor="Black" HorizontalOptions="Start"
            Margin="10" HorizontalTextAlignment="Start"/>
        <Entry Text="{Binding Pais}" Placeholder="Nombre" WidthRequest="200"
            FontSize="20" TextColor="Black" BackgroundColor="White" HorizontalOptions="Start"
            Margin="10" HorizontalTextAlignment="Start" FontAttributes="Bold"/>
    </StackLayout>

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Guardar" Text="Guardar" Order="Primary" Priority="0"
            Clicked="Guardar_Clicked"/>
        <ToolbarItem x:Name="Eliminar" Text="Eliminar" Order="Primary" Priority="1"
            Clicked="Eliminar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using FutbolAppAzure.Clases;
using System;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace FutbolAppAzure.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaEquipo : ContentPage
    {
        Equipo Equipo;

        public PaginaEquipo(Equipo equipo)
        {
            InitializeComponent();
            Equipo = equipo;
            this.BindingContext = equipo;
        }

        private async void Guardar_Clicked(object sender, EventArgs e)
        {
            if (!string.IsNullOrEmpty(Equipo.Id))
                await App.ServicioAzure.ModificarEquipo(Equipo);
            else
                await App.ServicioAzure.AgregarEquipo(Equipo);

            await DisplayAlert("FutbolApp", "Equipo registrado con éxito", "OK");
        }

        private async void Eliminar_Clicked(object sender, EventArgs e)
        {
            if (await DisplayAlert("Eliminar", "¿Deseas eliminar el equipo?", "Si",
"No"))
            {
                await App.ServicioAzure.EliminarEquipo(Equipo);
                await DisplayAlert("FutbolApp", "Equipo eliminado con éxito", "OK");
                await Navigation.PopAsync();
            }
        }
    }
}
```

Paso 29. Modifica **MainActivity.cs** en el proyecto de Android agregando la inicialización de los **MobileServices** ANTES de la inicialización de **Forms**:

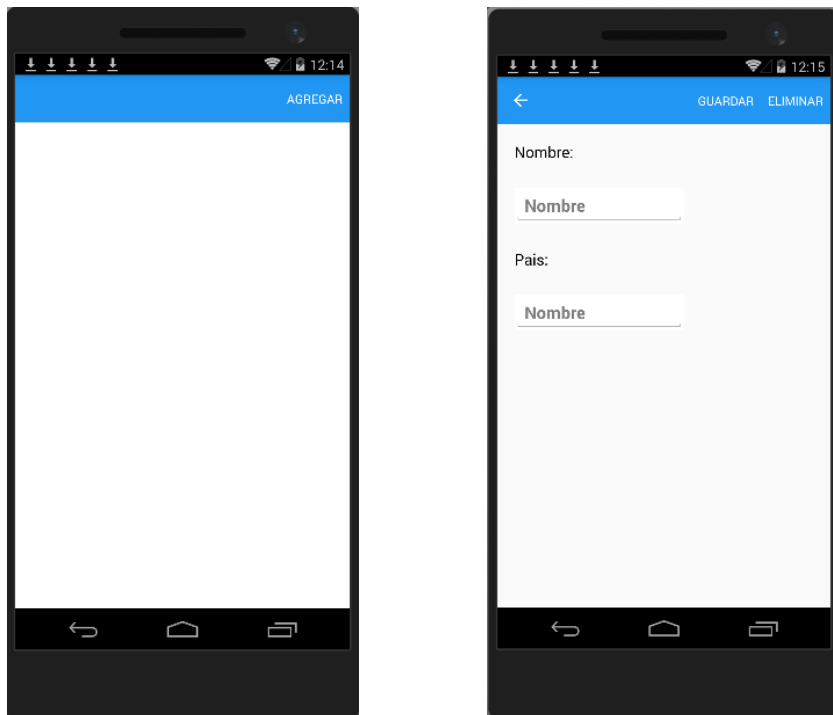
```
protected override void OnCreate(Bundle bundle)
{
    TabLayoutResource = Resource.Layout.Tabbar;
    ToolbarResource = Resource.Layout.Toolbar;

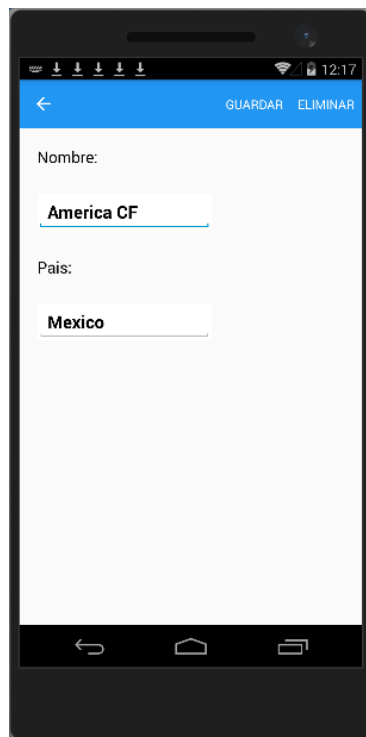
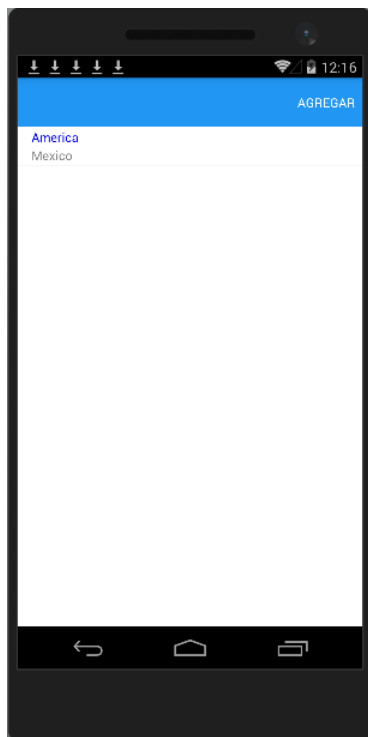
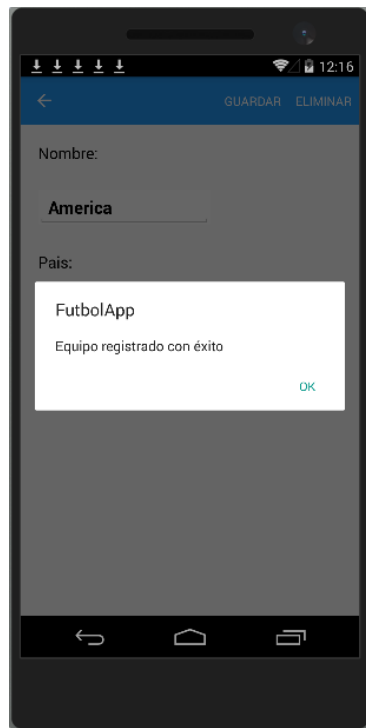
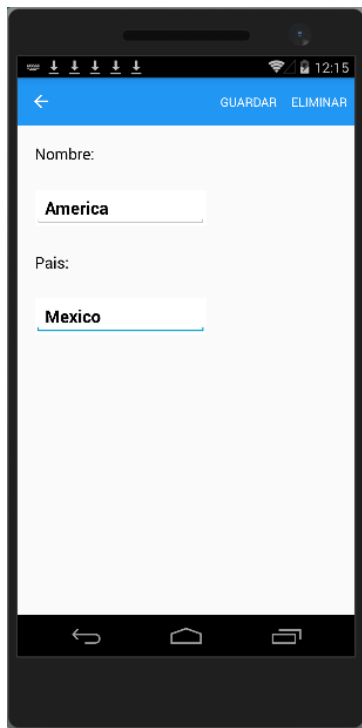
    base.OnCreate(bundle);

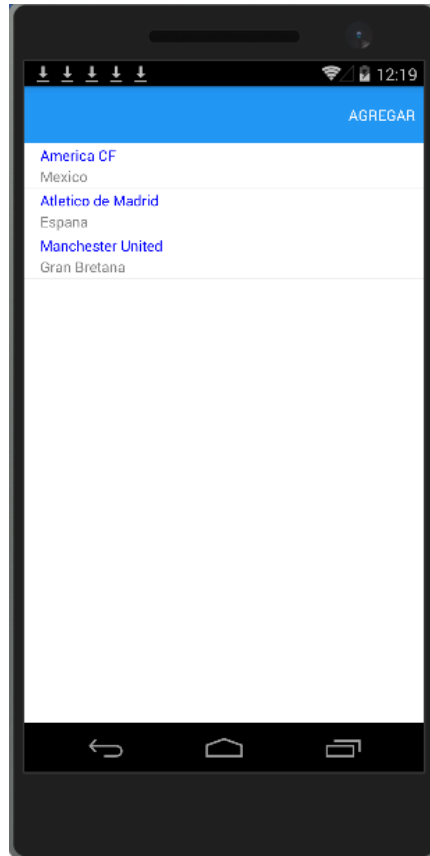
    Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();

    global::Xamarin.Forms.Forms.Init(this, bundle);
    LoadApplication(new App());
}
```

Paso 30. ¡Listo! Compila y ejecuta la aplicación. Observa el resultado







Podemos verificar en el portal de Azure que la tabla Equipo ha sido actualizada con los campos **Nombre** y **Pais**, además de comprobar que se han insertado los nuevos registros

Equipo (3 records)								
✕ Change permissions 🔊 Edit script ✕ Manage schema 🔄 Refresh 🗑 Delete table ✕ Clear table 🗑 Delete rows 📄 Streaming logs								
ID	CREATEDAT	UPDATEDAT	VERSION	DELETE...	NOMBRE	PAIS		
338b1ee1-07a2-4c...	2017-08-16T1...	2017-08-16T10:...	AAAAAAAAA...	false	America CF	Mexico		
7dcca045-862b-48...	2017-08-16T1...	2017-08-16T10:...	AAAAAAAAA...	false	Manchester United	Gran Bretana		
8d85ad1e-495b-40...	2017-08-16T1...	2017-08-16T10:...	AAAAAAAAA...	false	Atletico de Madrid	Espana		