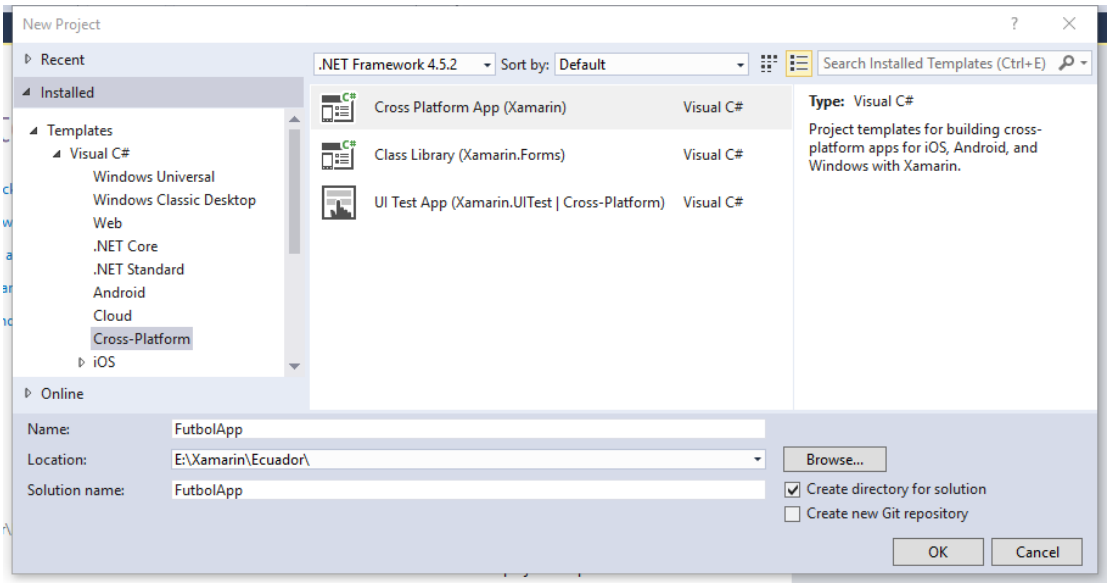
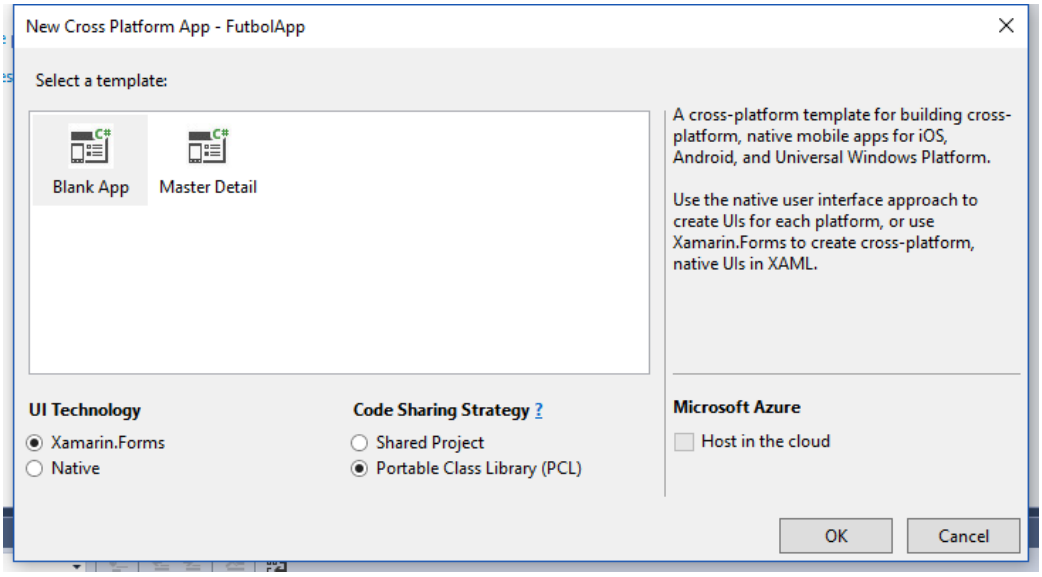
Práctica: Almacenamiento local con SQLite en Xamarin – Autor: Luis Beltrán

En esta sesión vamos a crear una aplicación móvil multiplataforma con Xamarin que almacena información en el dispositivo mediante SQLite.

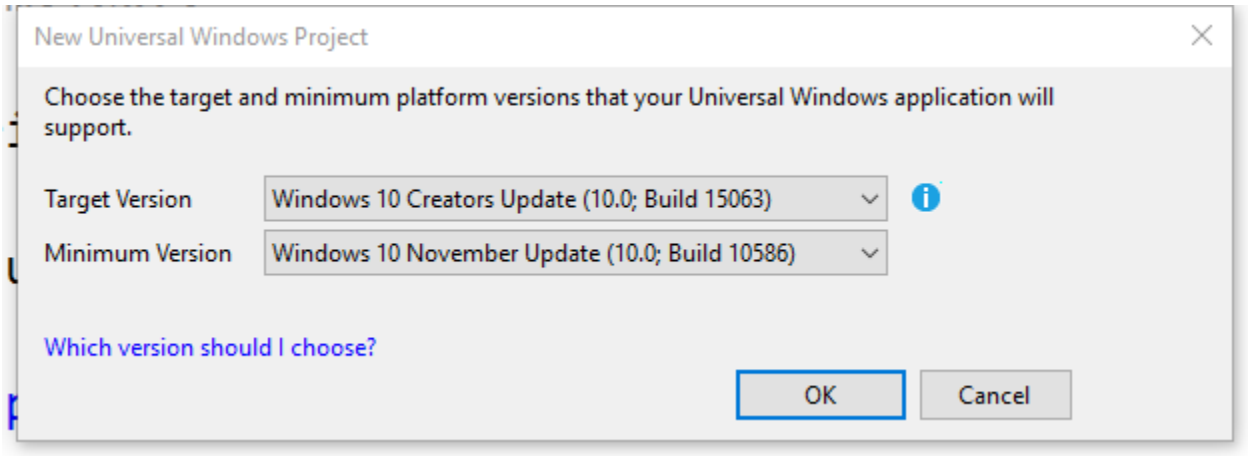
Paso 1. Crea un nuevo proyecto de la categoría **Cross-Platform** selecciona **Aplicación multiplataforma (Xamarin.Forms o nativa)** y coloca el nombre de proyecto **FutbolApp**. Además, la ruta del proyecto debe ser una ubicación corta para evitar problemas de ruta larga.



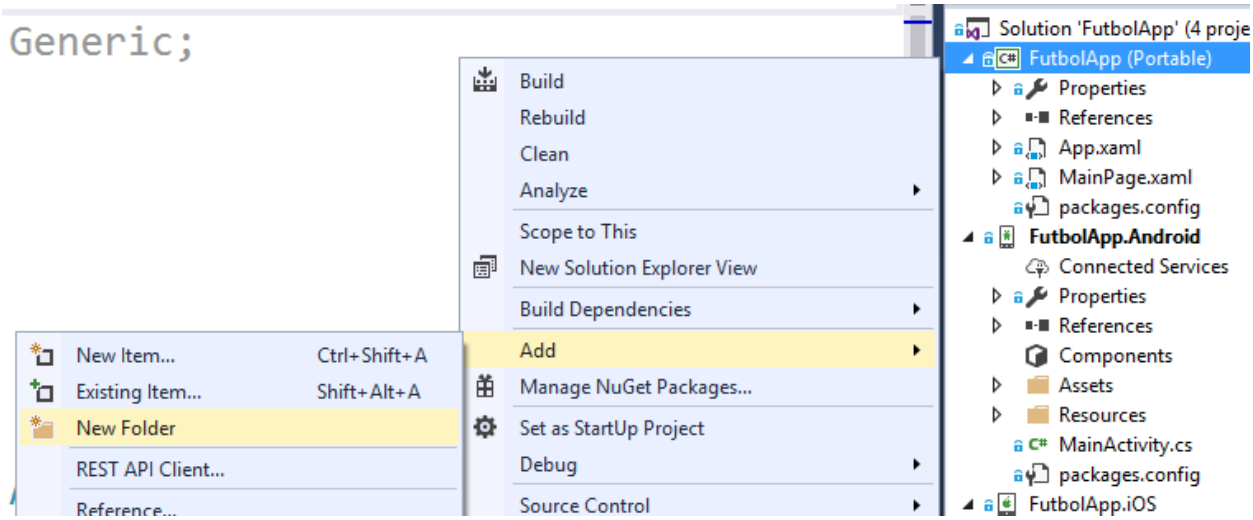
Paso 2. Selecciona la plantilla **Aplicación en blanco**, la tecnología de IU **Xamarin.Forms** y la estrategia de uso compartido de código **Biblioteca de clases portátil (PCL)**. Da clic en **OK**.



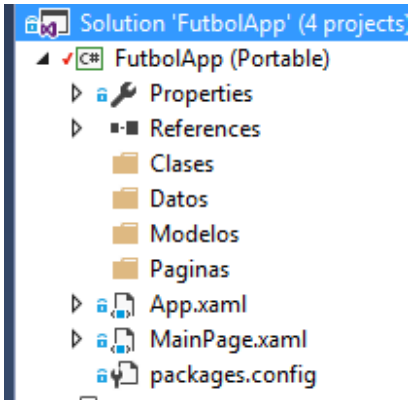
Paso 3. Si tienes instalado el SDK de Windows 10, aparecerá la ventana de selección del Target y Minimum Version. Selecciónalas a conveniencia, según la versión que tengas instalada.



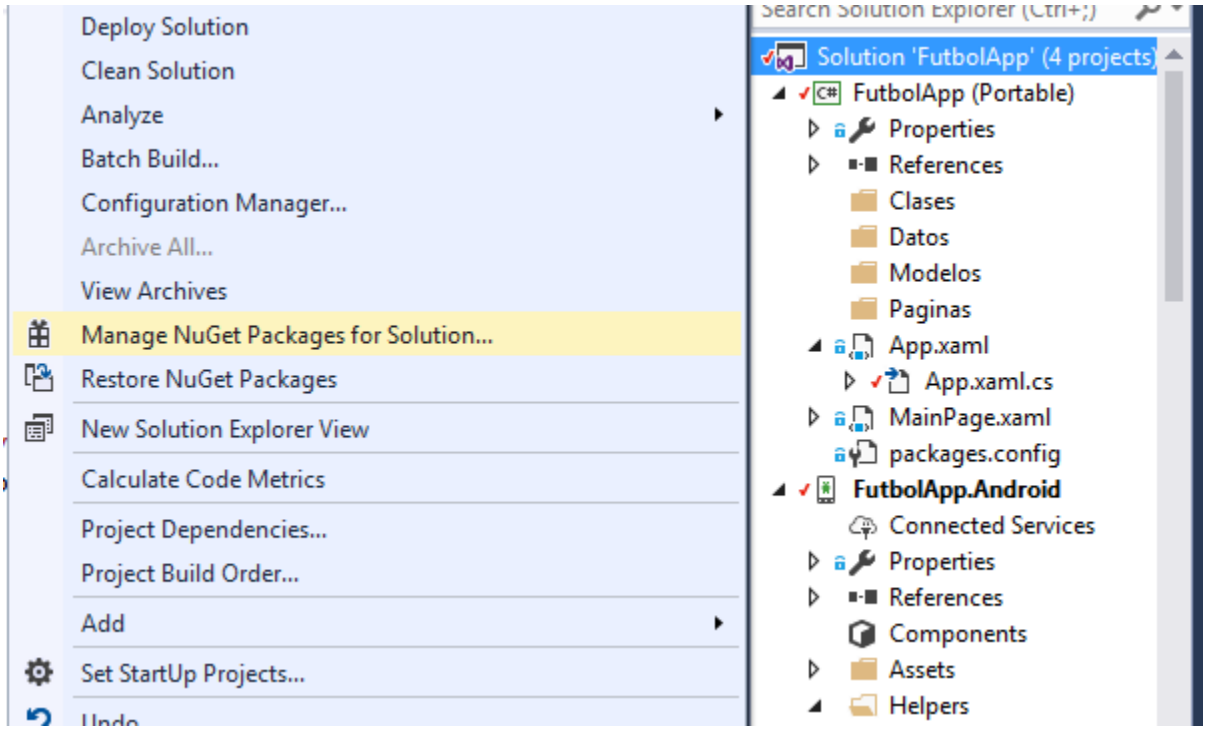
Paso 4. Da **clic derecho** en el **proyecto portable** y selecciona **Agregar → Nueva carpeta**.



Paso 5. Agrega las carpetas Clases, Datos, Modelos y Paginas.

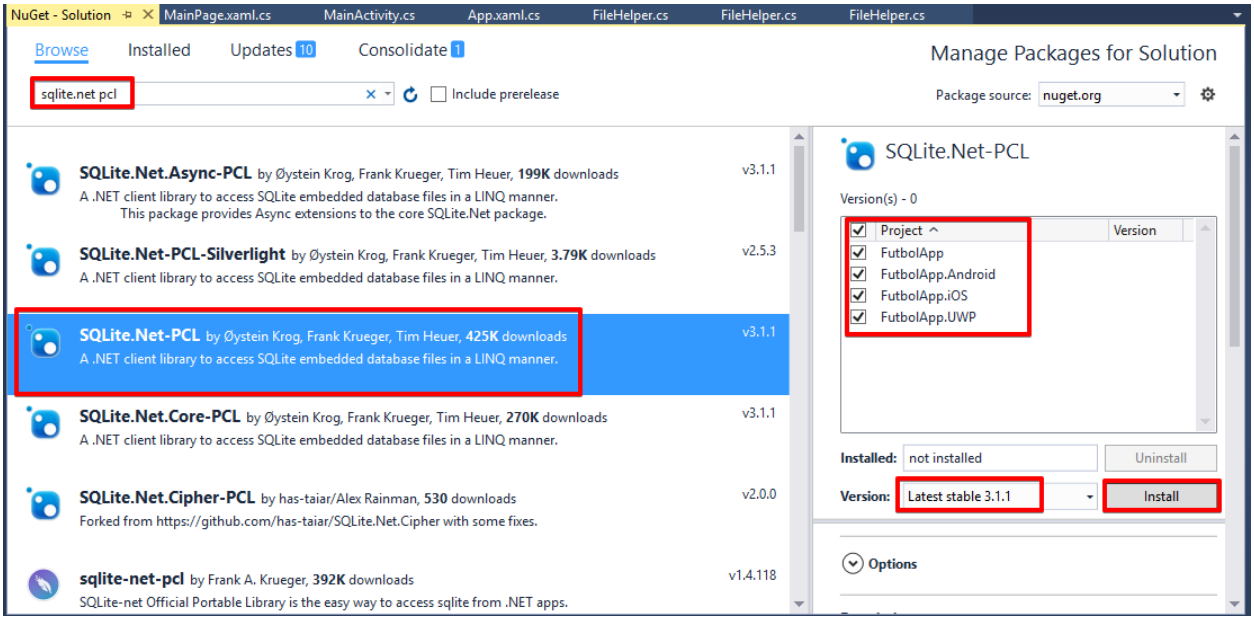


Paso 6. Da clic derecho en la solución y selecciona la opción **Administrar paquetes NuGet para la solución…**

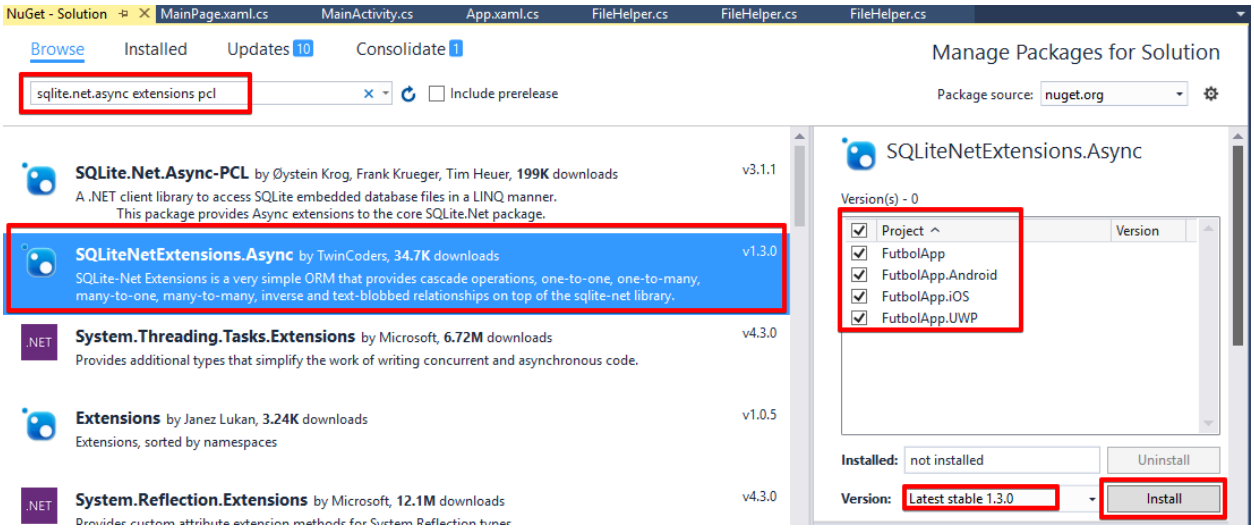


Paso 7. Agrega los siguientes paquetes Nuget:

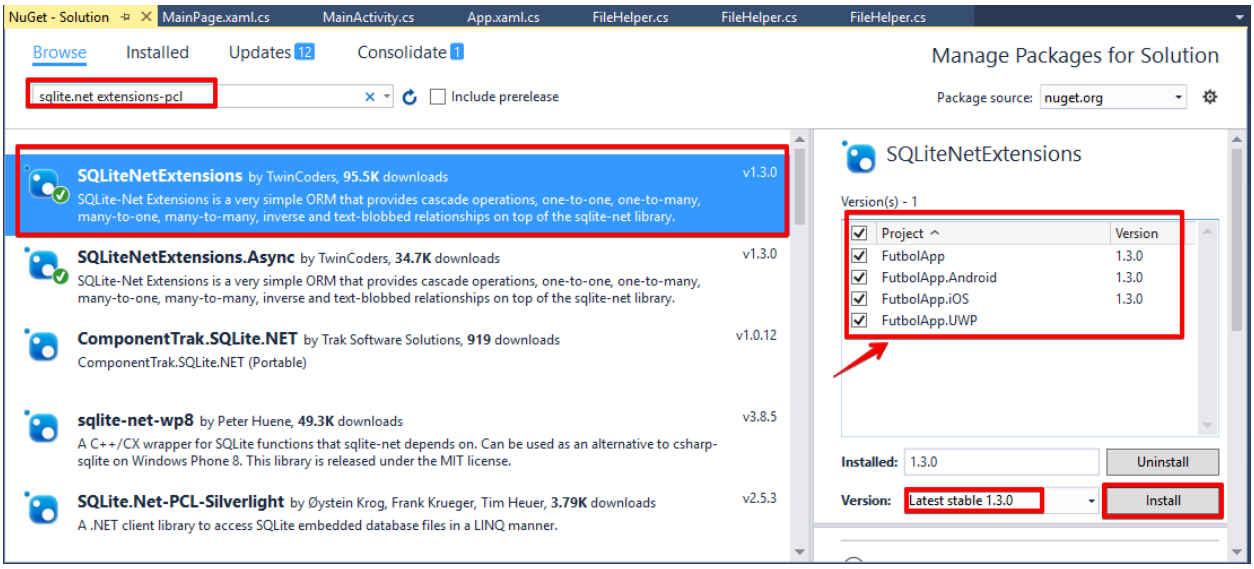
a) **SQLite.Net-PCL** (para el soporte de bases de datos locales con SQLite en proyectos PCL)



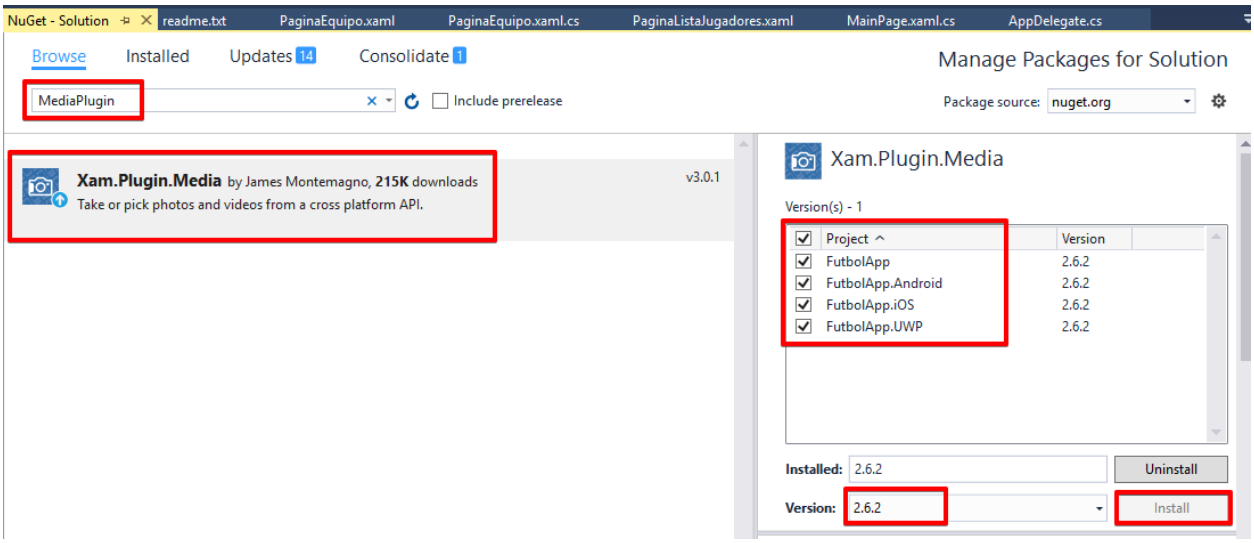
b) **SQLiteNetExtensions.Async** (una extensión de SQLite que permite manejar operaciones asíncronas e incorpora funcionalidad adicional, por ejemplo, relaciones entre tablas)



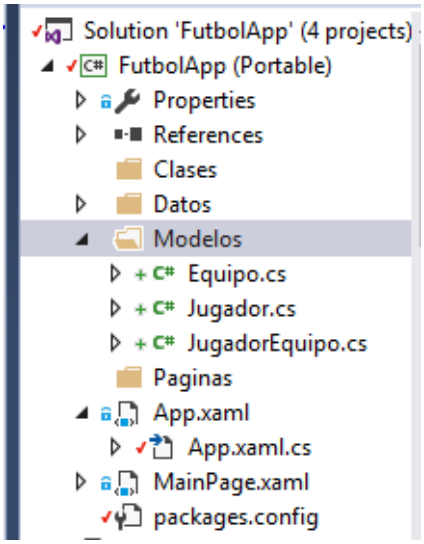
c) **SQLiteNetExtensions** (una extensión de SQLite que incorpora funcionalidad adicional, por ejemplo, relaciones entre tablas)



d) Xam.Plugin.Media (este paquete Nuget permite capturar fotos y video de la cámara o seleccionar elementos multimedia de la galería de imágenes/video del dispositivo)



Paso 8. En la carpeta **Modelos** agrega tres clases: **Equipo, Jugador y JugadorEquipo**, los cuales representan las tablas de la base de datos que vamos a modelar



Paso 9. El código de cada clase es el siguiente:

a) **Jugador:** Datos básicos de un jugador de futbol. **Foto** es una ruta a la imagen del jugador en el dispositivo. **Equipos** es un elemento virtual que representa la navegación a la colección de N equipos a los que pertenece el Jugador en algún momento dado. **ID** es la llave primaria de cada jugador.

```
using SQLite.Net.Attributes;
using SQLiteNetExtensions.Attributes;
using System;
using System.Collections.Generic;

namespace FutbolApp.Modelos
{
    public class Jugador
    {
        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }

        public string Nombre { get; set; }
        public string FechaNacimiento { get; set; } = DateTime.MinValue.ToString();
        public string Foto { get; set; }

        [ManyToMany(typeof(JugadorEquipo))]
        public List<Equipo> Equipos { get; set; }
    }
}
```

b) **Equipo:** Datos básicos de un equipo de futbol. **Escudo** es una ruta a la imagen del escudo de un equipo en el dispositivo. **Jugadores** es un elemento virtual que representa la navegación a la colección de N jugadores que pertenecen al Equipo en algún momento dado. **ID** es la llave primaria de cada jugador.

```
using SQLite.Net.Attributes;
using SQLiteNetExtensions.Attributes;
using System.Collections.Generic;

namespace FutbolApp.Modelos
{
    public class Equipo
    {
        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }

        public string Nombre { get; set; }
        public string Escudo { get; set; }

        [ManyToMany(typeof(JugadorEquipo))]
        public List<Jugador> Jugadores { get; set; }
    }
}
```

c) **JugadorEquipo**: Este modelo representa la tabla asociativa obtenida por la relación de **Muchos a Muchos** entre **Jugador** y **Equipo**. Además del **ID** como llave primaria requerida, se tienen los **identificadores de cada tabla** en forma de **llave foránea**. **Numero** y **Goles** son características particulares de cada jugador en el equipo.

```
using SQLite.Net.Attributes;
using SQLiteNetExtensions.Attributes;

namespace FutbolApp.Modelos
{
    public class JugadorEquipo
    {
        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }

        [ForeignKey(typeof(Equipo))]
        public int IDEquipo { get; set; }

        [ForeignKey(typeof(Jugador))]
        public int IDJugador { get; set; }

        public int Numero { get; set; }
        public int Goles { get; set; }
    }
}
```

Paso 10. En la carpeta **Clases** agrega las clases **DetalleJugadorEquipo** y **ServicioImagenes**

a) DetalleJugadorEquipo: Es una vista de datos que combina información de Jugadores, Equipos y su detalle.

```
namespace FutbolApp.Clases
{
    public class DetalleJugadorEquipo
    {
        public string FotoJugador { get; set; }
        public string NombreJugador { get; set; }
        public string EscudoEquipo { get; set; }
        public string NombreEquipo { get; set; }
        public int Numero { get; set; }
        public int Goles { get; set; }
    }
}
```

b) ServicioImagenes: Es una clase que incorpora funcionalidad del MediaPlugin para obtener una fotografía de la galería, es decir, de la carpeta pública de Imágenes del dispositivo.

```
using Plugin.Media;
using Plugin.Media.Abstractions;
using System.Threading.Tasks;

namespace FutbolApp.Clases
{
    public static class ServicioImagenes
    {
```

```
        public static async Task<MediaFile> SeleccionarImagen()
        {
            await CrossMedia.Current.Initialize();
            var file = await CrossMedia.Current.PickPhotoAsync();
            return file;
        }
    }
}
```

Paso 11. Ahora agrega una clase llamada **BaseDatos** en la carpeta **Datos**. Esta clase sirve para conectarnos al almacenamiento local de SQLite.

- El objeto **locker** servirá para incorporar un bloqueo al realizar alguna operación sobre la base de datos, para manejar la concurrencia entre tablas.
- **_plataforma** y **_rutaBD** son elementos privados específicos de cada plataforma móvil. El primer elemento representa el tipo de conexión particular (Android tiene el suyo, iOS usa un objeto distinto y UWP opera de manera diferente, por lo que se requiere un objeto específico) y el segundo es la ruta física donde estará la base de datos en el dispositivo.
- En el constructor se especifican los elementos particulares por plataforma.
- El método **Conectar** crea una conexión a la base de datos local. Si no existe, se crea el archivo físico, así como las tablas por primera vez; si ya existe, simplemente devuelve una referencia.
- Se definen los métodos específicos por tabla para las operaciones CRUD (Create, Read, Update y Delete) para poder obtener información, agregar, modificar y eliminar datos en cada tabla. Con LINQ se definen algunas consultas particulares, ordenamientos, etc.

```
using System.Collections.Generic;
using System.Linq;
using SQLite.Net;
using SQLite.Net.Interop;
using FutbolApp.Modelos;
using SQLiteNetExtensions.Extensions;
using FutbolApp.Clases;

namespace FutbolApp.Datos
{
    public class BaseDatos
    {
        static object locker = new object();
        readonly ISQLitePlatform _plataforma;
        string _rutaBD;

        public SQLiteConnection Conexion { get; set; }

        public BaseDatos(ISQLitePlatform plataforma, string rutaBD)
        {
            _plataforma = plataforma;
            _rutaBD = rutaBD;
        }

        public void Conectar()
        {
```

```
        Conexion = new SQLiteConnection(_plataforma, _rutaBD,
            SQLiteOpenFlags.ReadWrite | SQLiteOpenFlags.Create
            | SQLiteOpenFlags.FullMutex, true);

        Conexion.CreateTable<Jugador>();
        Conexion.CreateTable<Equipo>();
        Conexion.CreateTable<JugadorEquipo>();
    }

    #region Metodos de la tabla Jugador

    public void AgregarJugador(Jugador jugador, Equipo equipo = null)
    {
        lock(locker)
        {
            Conexion.Insert(jugador);
        }
    }

    public void ActualizarJugador(Jugador jugador, Equipo equipo = null)
    {
        lock (locker)
        {
            Conexion.Update(jugador);

            if (equipo != null)
            {
                if (jugador.Equipos == null)
                {
                    jugador.Equipos = new List<Equipo>();
                }

                if (jugador.Equipos.Where(x => x.ID == equipo.ID).Count() == 0)
                    jugador.Equipos.Add(equipo);

                Conexion.UpdateWithChildren(jugador);
            }
        }
    }

    public void EliminarJugador(Jugador jugador)
    {
        lock (locker)
        {
            Conexion.Delete(jugador);
        }
    }

    public List<Jugador> ObtenerJugadores()
    {
      lock (locker)
      {
       return Conexion.GetAllWithChildren<Jugador>().OrderBy(x => x.Nombre).ToList();
      }
    }

    public Jugador ObtenerJugador(int id)
    {
```

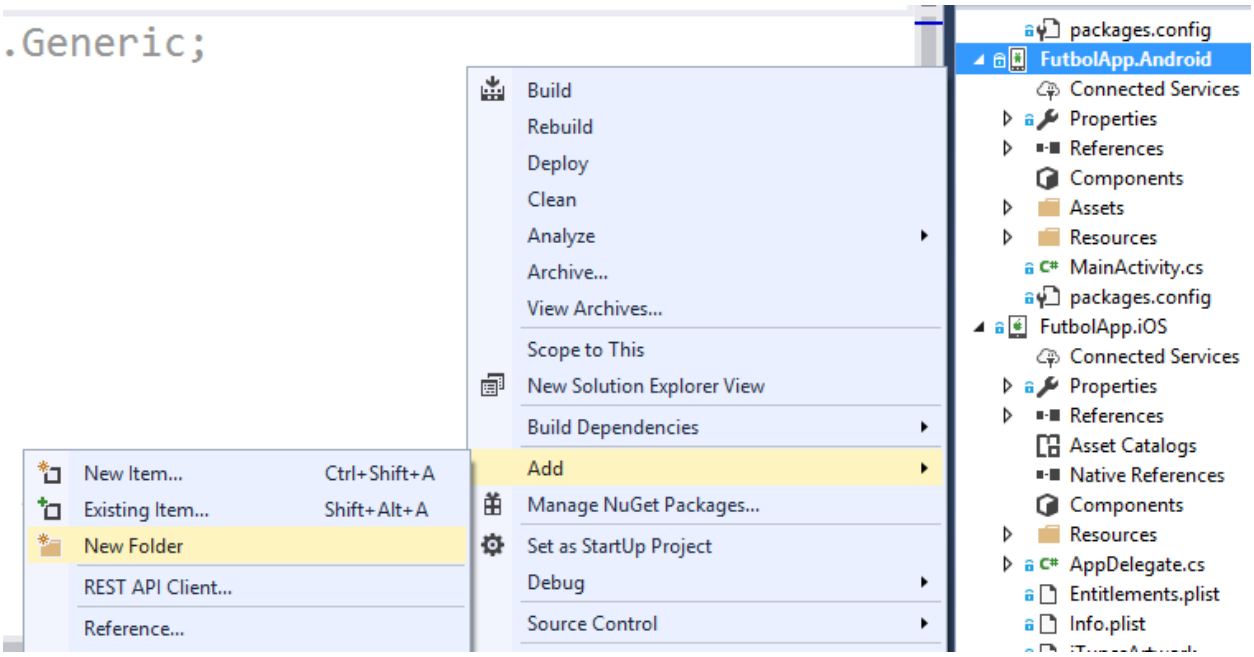
```
    lock (locker)
    {
        return Conexion.GetWithChildren<Jugador>(id);
    }
}

public List<Equipo> ObtenerEquiposJugador(int id)
{
    lock (locker)
    {
        return Conexion.GetWithChildren<Jugador>(id).Equipos;
    }
}

#endregion

#region Metodos de la tabla Equipo
public void AgregarEquipo(Equipo equipo)
{
    lock (locker)
    {
        Conexion.Insert(equipo);
    }
}

public void ActualizarEquipo(Equipo equipo)
{
    lock (locker)
    {
        Conexion.Update(equipo);
    }
}

public void EliminarEquipo(Equipo equipo)
{
    lock (locker)
    {
        Conexion.Delete(equipo);
    }
}

public List<Equipo> ObtenerEquipos()
{
  lock (locker)
  {
    return Conexion.GetAllWithChildren<Equipo>().OrderBy(x => x.Nombre).ToList();
  }
}

public Equipo ObtenerEquipo(int id)
{
    lock (locker)
    {
        return Conexion.GetWithChildren<Equipo>(id);
    }
}

public List<Jugador> ObtenerJugadoresEquipo(int id)
```

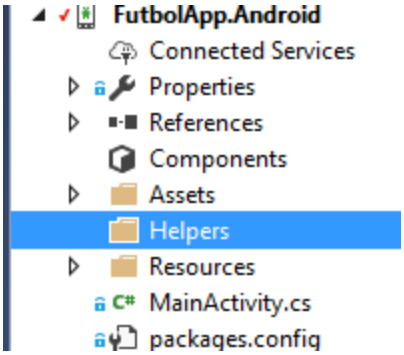
```
        {
            lock (locker)
            {
                return Conexion.GetWithChildren<Equipo>(id).Jugadores;
            }
        }
        #endregion

        #region Metodos de la tabla JugadorEquipo
        public JugadorEquipo ObtenerJugadorEquipo(int idEquipo, int idJugador)
        {
            lock(locker)
            {
                var tabla = Conexion.Table<JugadorEquipo>().ToList();
                var num = tabla.Where(x => x.IDEquipo == idEquipo && x.IDJugador ==
idJugador).Count();

                if (num > 0)
                    return tabla.Where(x => x.IDEquipo == idEquipo && x.IDJugador ==
idJugador).First();
                else
                    return new JugadorEquipo() { IDEquipo = idEquipo, IDJugador =
idJugador, Goles = 0, Numero = 0 };
            }
        }

        public DetalleJugadorEquipo ObtenerDetalleJugadorEquipo(Equipo equipo, Jugador
jugador)
        {
            lock (locker)
            {
                var jugadorEquipo = ObtenerJugadorEquipo(equipo.ID, jugador.ID);

                var detalle = new DetalleJugadorEquipo()
                {
                    NombreJugador = jugador.Nombre,
                    FotoJugador = jugador.Foto,
                    NombreEquipo = equipo.Nombre,
                    EscudoEquipo = equipo.Escudo,
                    Goles = (jugadorEquipo != null) ? jugadorEquipo.Goles : 0,
                    Numero = (jugadorEquipo != null) ? jugadorEquipo.Numero : 0
                };

                return detalle;
            }
        }

        public void ActualizarJugadorEquipo(JugadorEquipo jugadorEquipo)
        {
            lock (locker)
            {
                Conexion.Update(jugadorEquipo);
            }
        }
        #endregion
    }
}
```

Paso 12. Modifica **App.xaml.cs** para inicializar la base de datos, conectarnos a ella y definir la página de inicio:

```
using FutbolApp.Datos;
using SQLite.Net.Interop;
using Xamarin.Forms;

namespace FutbolApp
{
    public partial class App : Application
    {
        public static BaseDatos BaseDatos { get; set; }

        public App(string rutaBD, ISQLitePlatform plataforma)
        {
            BaseDatos = new BaseDatos(plataforma, rutaBD);
            BaseDatos.Conectar();

            InitializeComponent();

            MainPage = new NavigationPage(new FutbolApp.Paginas.PaginaMenu());
        }
    }
}
```

Paso 13. En el proyecto de Android crea una carpeta llamada **Helpers**





Paso 14. Agrega una clase llamada **FileHelper**, con el siguiente código, donde obtenemos la ruta específica donde se almacenará la base de datos.

```
using System;
using System.IO;

namespace FutbolApp.Droid.Helpers
{
    public class FileHelper
    {
        public static string ObtenerRutaLocal(string archivo)
        {
            string ruta = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            return Path.Combine(ruta, archivo);
        }
    }
}
```

Paso 15. Ahora repetimos el proceso anterior pero en el proyecto de **iOS** para crear un método que devuelva la ruta específica de la base de datos en este proyecto; así pues, crea la carpeta **Helpers** y la clase **FileHelper**, con el siguiente código:

```
using System;
using System.IO;

namespace FutbolApp.iOS.Helpers
{
    public class FileHelper
    {
        public static string ObtenerRutaLocal(string archivo)
        {
           string folder = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
           string libreria = Path.Combine(folder, "..", "Library");

           if (!Directory.Exists(libreria))
               Directory.CreateDirectory(libreria);
           return Path.Combine(libreria, archivo);
        }
    }
}
```

Paso 16. Con el mismo propósito, en el proyecto de **UWP** crea la carpeta **Helpers** y la clase **FileHelper**, con el siguiente código:

```
using System.IO;

namespace FutbolApp.UWP.Helpers
{
    public class FileHelper
    {
        public static string ObtenerRutaLocal(string archivo)
        {
            string ruta = Windows.Storage.ApplicationData.Current.LocalFolder.Path;
            return Path.Combine(ruta, archivo);
        }
    }
}
```

Paso 17. Dentro del proyecto de **Android**, modifica **MainActivity** para definir el objeto SQLite específico de plataforma (**SQLitePlatformAndroid**) y pasar la ruta específica de la base de datos:

```
    [Activity(Label = "FutbolApp", Icon = "@drawable/icon", Theme = "@style/MainTheme",
MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle bundle)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate(bundle);

            global::Xamarin.Forms.Forms.Init(this, bundle);

            string rutaBD = Helpers.FileHelper.ObtenerRutaLocal("futbol.db3");
            LoadApplication(new App(rutaBD, new
SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid()));
        }

        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, Permission[] grantResults)
        {

Plugin.Permissions.PermissionsImplementation.Current.OnRequestPermissionsResult(requestCo
de, permissions, grantResults);
        }
    }
```

También se escribió el código requerido por el MediaPlugin para obtener los permisos de uso de la cámara.

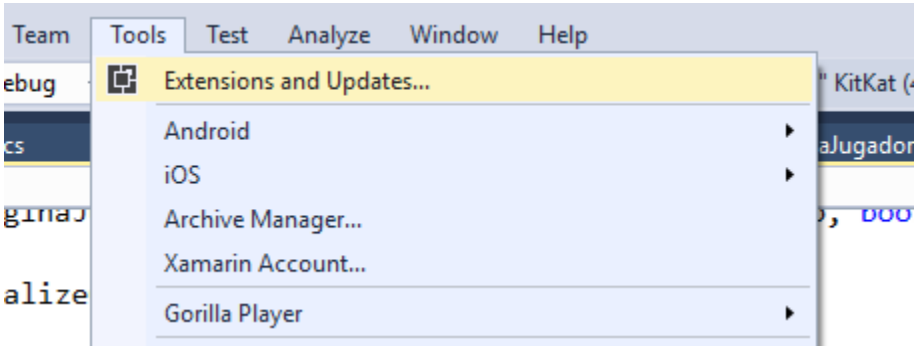
Paso 18. Dentro del proyecto de **iOS**, modifica **AppDelegate** para indicar que debe usar **SQLitePlatformIOS** como referencia para la base de datos local, así como la ruta obtenida por el método **ObtenerRutaLocal** definido en la clase **FileHelper**:

```
        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();

            string rutaBD = Helpers.FileHelper.ObtenerRutaLocal("futbol.db3");
            LoadApplication(new App(rutaBD, new
SQLite.Net.Platform.XamarinIOS.SQLitePlatformIOS()));

            return base.FinishedLaunching(app, options);
        }
```

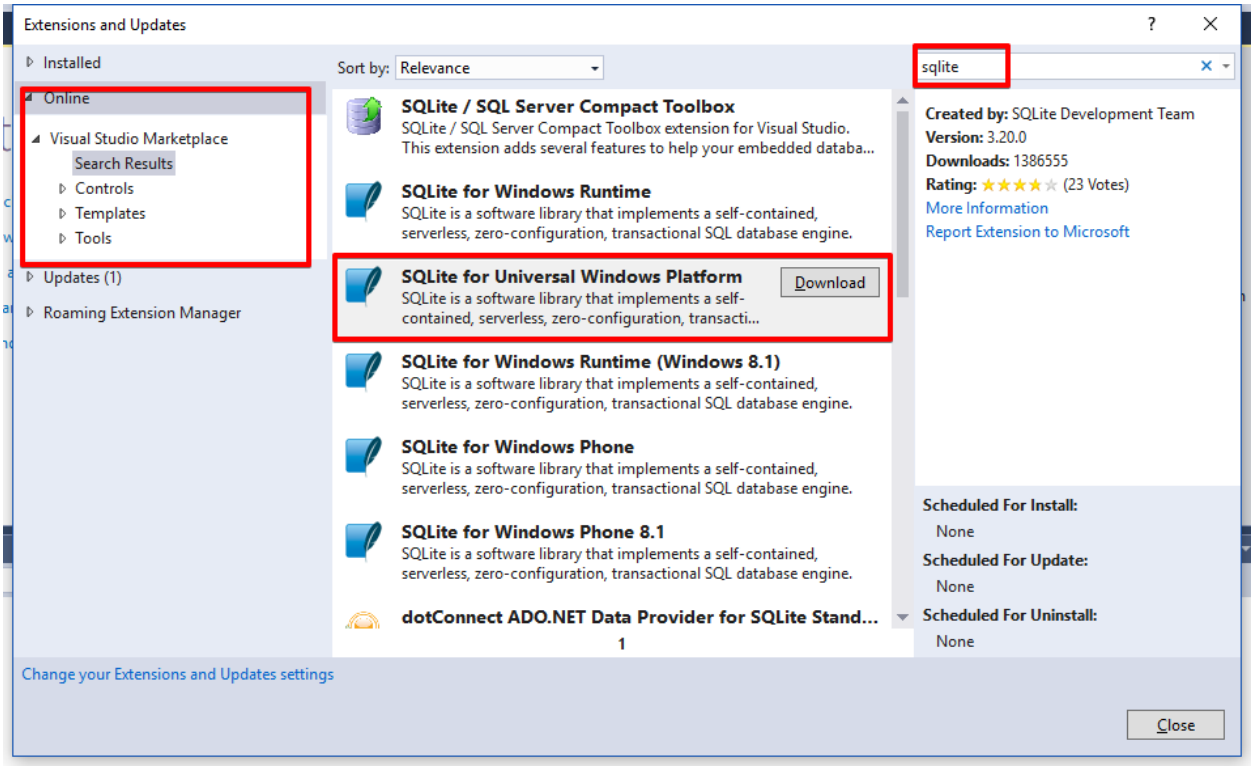
Paso 19. En el caso del proyecto de **UWP** se utiliza un objeto **SQLitePlatformWinRT** para manejar la base de datos local y se indica en el archivo **MainPage**:

```
        public MainPage()
        {
            this.InitializeComponent();

            string rutaBD = Helpers.FileHelper.ObtenerRutaLocal("futbol.db3");
            LoadApplication(new FutbolApp.App(rutaBD, new
SQLite.Net.Platform.WinRT.SQLitePlatformWinRT()));
        }
```

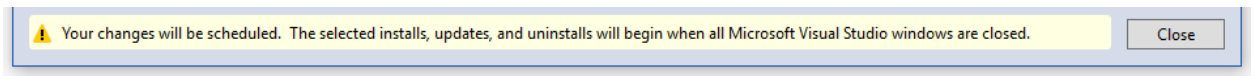
Paso 20: En el proyecto de UWP también debes agregar la extensión de **SQLite para UWP**. Primero, accede al menú **Herramientas** y selecciona **Extensiones y Actualizaciones**:



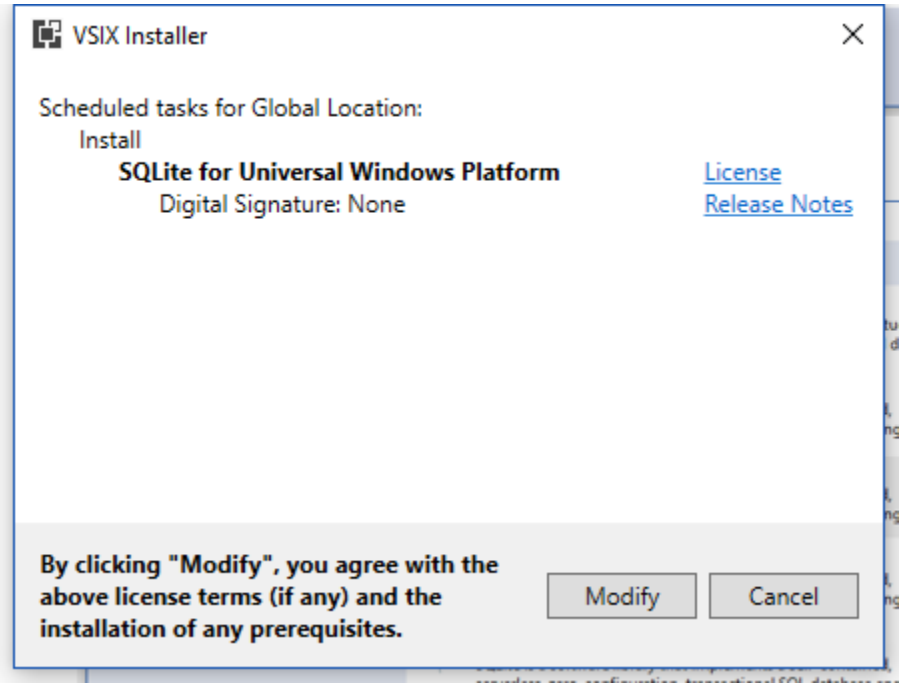
En la categoría **Online** busca **SQLite** y descarga **SQLite for Universal Windows Platform**



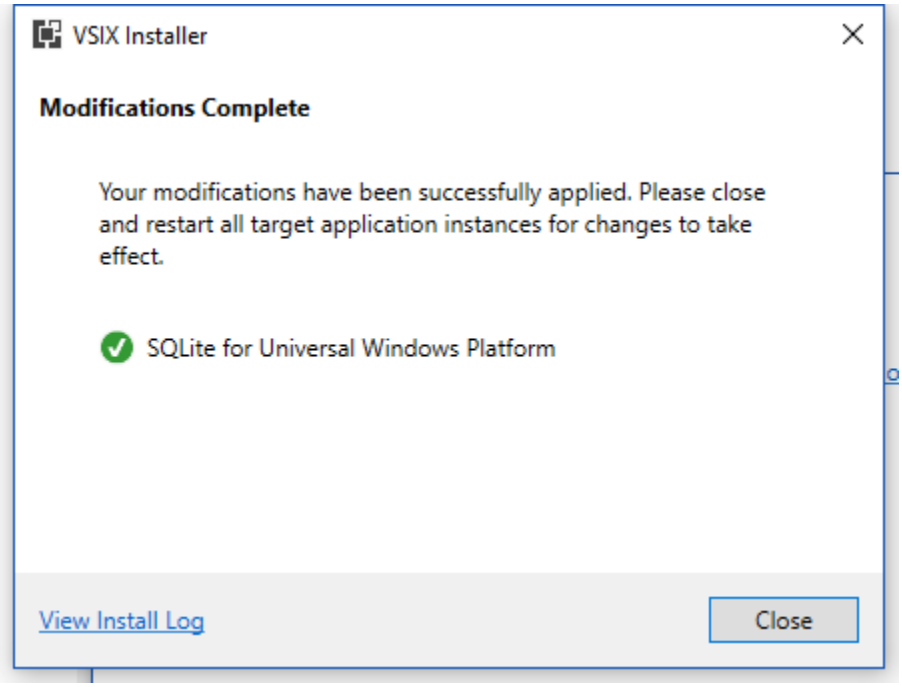
Si te aparece el siguiente mensaje:



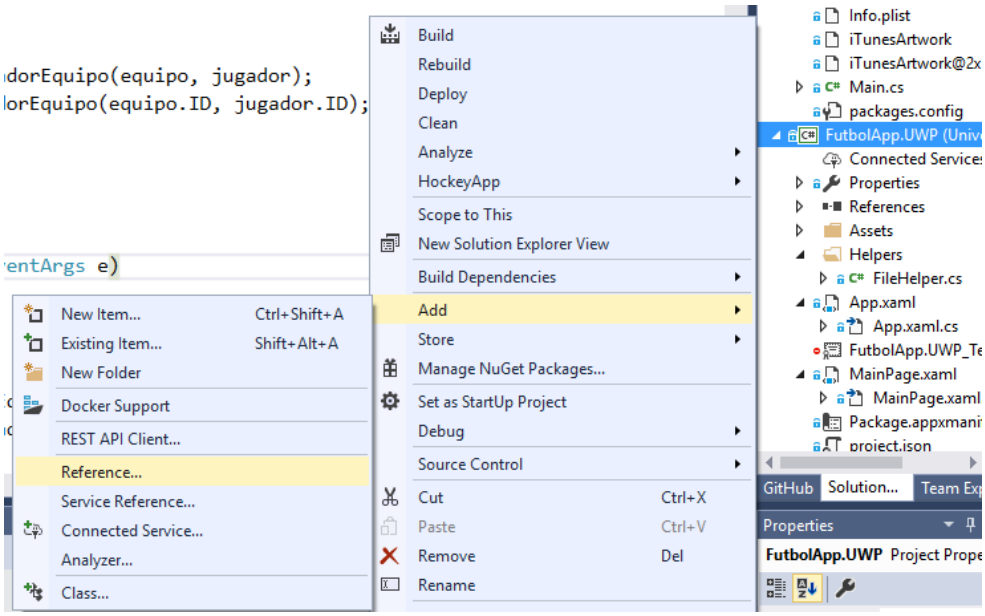
Cierra todas las ventanas de Visual Studio y el proceso de instalación comenzará. Da clic en Instalar o Modificar, según sea el caso:



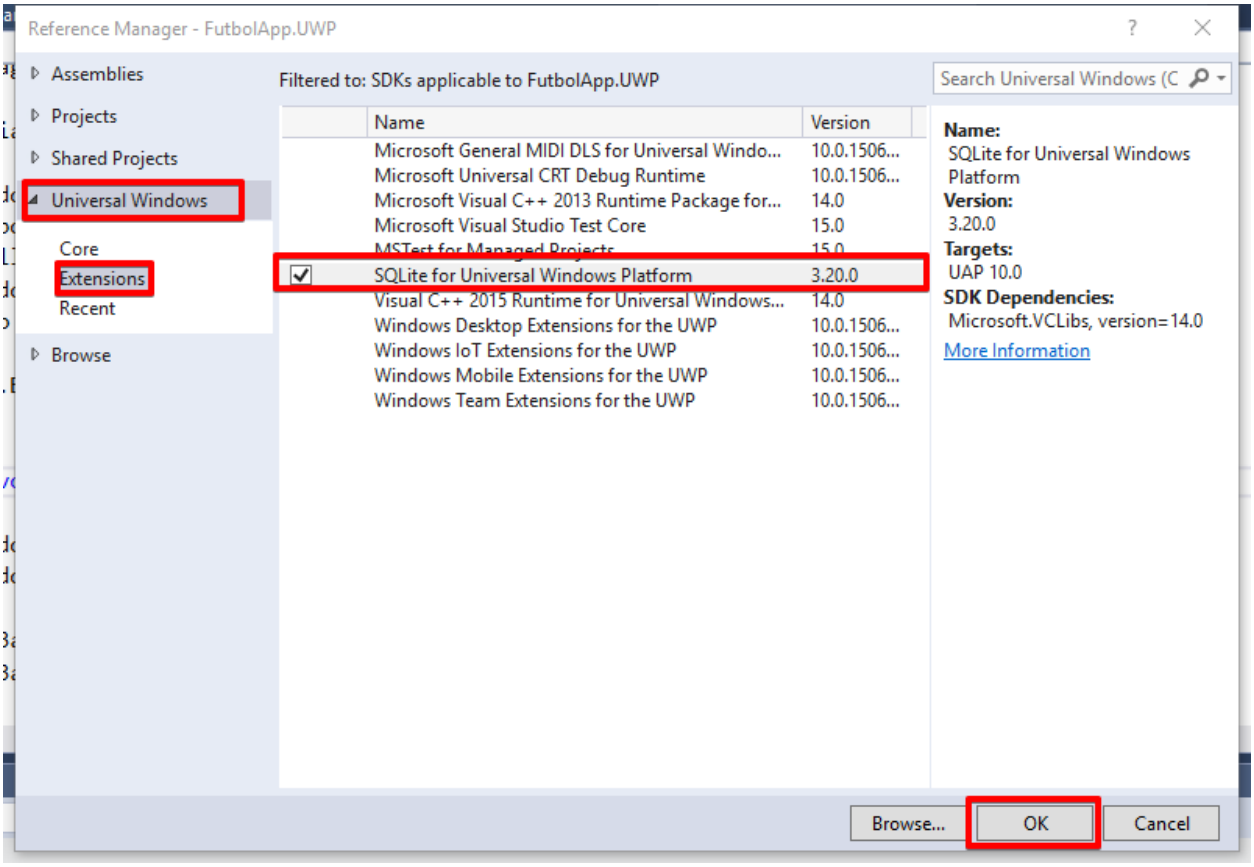
Una vez concluida la instalaciúon, cierra la ventana y abre de nuevo tu proyecto en Visual Studio.



Paso 21: Ahora agrega esta extensión en el proyecto de UWP. Para ello, da **clic derecho** en el proyecto y selecciona **Agregar → Referencias**:



Paso 22: En la categoría **Universal → Extensions** localiza **SQLite for Universal Windows Platform**.



Paso 23. En la carpeta **Paginas** agrega las siguientes páginas de tipo **ContentPage**:

a) PaginaMenu: Esta página representa la vista inicial del proyecto donde el usuario podrá navegar a la vista de **Jugadores** o a la vista de **Equipos**, según su elección; los botones estarán localizados en la **barra de menús** de la app. Primero se muestra el código XAML y posteriormente el código C# donde se muestra la navegación:

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="FutbolApp.Paginas.PaginaMenu" Title="Menu">
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Jugadores" Text="Jugadores" Order="Primary" Priority="0"
Clicked="Jugadores_Clicked"/>
        <ToolbarItem x:Name="Equipos" Text="Equipos" Order="Primary" Priority="1"
Clicked="Equipos_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;

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

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

        private async void Jugadores_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginaListaJugadores());
        }

        private async void Equipos_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginaListaEquipos());
        }
    }
}
```

b) 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, el **Escudo** para en un control **Image** y el **Nombre** del equipo en un **Label**. 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="FutbolApp.Paginas.PaginaListaEquipos" Title="Equipos">
  <ScrollView>
    <StackLayout>
        <ListView x:Name="lsvEquipos" ItemSelected="lsvEquipos_ItemSelected">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout BackgroundColor="White" Spacing="5"
Orientation="Horizontal" Padding="5">
                            <Image Source="{Binding Escudo}" Aspect="AspectFill"
WidthRequest="70" HeightRequest="70"/>
                            <Label Text="{Binding Nombre}" LineBreakMode="WordWrap"
VerticalOptions="Center" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                        </StackLayout>
                    </ViewCell>
                </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 FutbolApp.Modelos;

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

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

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

            lsvEquipos.ItemsSource = App.BaseDatos.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()));
        }
    }
}
```

c) PaginaListaJugadores: Esta página contiene la lista de jugadores 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, la **Foto** para en un control **Image** y el **Nombre** del jugador en un **Label**. En el código de C# se muestra la forma de obtener los jugadores consultando la base de datos y asignándolos al **ItemSource** del **ListView**. Además, al **seleccionar** un jugador se navega a **PaginaJugador** 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 **PaginaJugador**, 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="FutbolApp.Paginas.PaginaListaJugadores" Title="Jugadores">
  <ScrollView>
    <StackLayout>
        <ListView x:Name="lsvJugadores" ItemSelected="lsvJugadores_ItemSelected">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout BackgroundColor="White" Spacing="5"
Orientation="Horizontal" Padding="5">
                            <Image Source="{Binding Foto}" Aspect="AspectFill"
WidthRequest="70" HeightRequest="70"/>
                            <Label Text="{Binding Nombre}" LineBreakMode="WordWrap"
VerticalOptions="Center" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                        </StackLayout>
                    </ViewCell>
                </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 FutbolApp.Modelos;

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

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

        public PaginaListaJugadores(Equipo equipo = null)
        {
            InitializeComponent();

            Equipo = equipo;

            if (Equipo != null)
            {
                if (this.ToolbarItems.Count > 0)
                    this.ToolbarItems.RemoveAt(0);
            }
        }

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

            lsvJugadores.ItemsSource = App.BaseDatos.ObtenerJugadores();
        }

    private async void lsvJugadores_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            try
            {
                Jugador jugador = (Jugador)e.SelectedItem;

                if (Equipo == null)
                    await Navigation.PushAsync(new PaginaJugador(jugador));
                else
                    await Navigation.PushAsync(new PaginaJugadorEquipo(jugador, Equipo,
true));
            }
            catch (Exception ex) { }
        }

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

d) PaginaEquipo: En esta página se muestra la información específica de un equipo para editarlo, eliminarlo o dar de alta un nuevo elemento. De igual forma que en los casos anteriores, 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="FutbolApp.Paginas.PaginaEquipo" Title="Equipo">
    <ScrollView>
        <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"/>
            <Button x:Name="btnEscudo" Text="Elegir escudo" BackgroundColor="LightBlue"
Margin="10" TextColor="Black" WidthRequest="200" HorizontalOptions="Center"
Clicked="btnEscudo_Clicked"/>
            <Image x:Name="imgEscudo" Source="{Binding Escudo}"
VerticalOptions="StartAndExpand" HorizontalOptions="Center" Aspect="AspectFill"
HeightRequest="150" WidthRequest="150" Margin="10"/>
            <ListView x:Name="lsvJugadores" ItemSelected="lsvJugadores_ItemSelected"
ItemsSource="{Binding Jugadores}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout BackgroundColor="White" Spacing="5"
Orientation="Horizontal" Padding="5">
                                <Image Source="{Binding Foto}" Aspect="AspectFill"
WidthRequest="70" HeightRequest="70"/>
                                <Label Text="{Binding Nombre}" LineBreakMode="WordWrap"
VerticalOptions="Center" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>

    <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"/>
        <ToolbarItem x:Name="Agregar" Text="Agregar jugador" Order="Primary" Priority="2"
Clicked="Agregar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using FutbolApp.Modelos;
using FutbolApp.Clases;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

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

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

        private async void btnEscudo_Clicked(object sender, EventArgs e)
        {
            var escudo = await ServicioImagenes.SeleccionarImagen();
            Equipo.Escudo = escudo.Path;
            imgEscudo.Source = ImageSource.FromFile(escudo.Path);
        }

        private async void Guardar_Clicked(object sender, EventArgs e)
        {
            if (Equipo.ID > 0) App.BaseDatos.ActualizarEquipo(Equipo);
            else App.BaseDatos.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"))
                App.BaseDatos.EliminarEquipo(Equipo);
            await DisplayAlert("FutbolApp", "Equipo eliminado con éxito", "OK");
            await Navigation.PopAsync();
        }

        private async void lsvJugadores_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            Jugador jugador = (Jugador)e.SelectedItem;
            await Navigation.PushAsync(new PaginaJugadorEquipo(jugador, Equipo, false));
        }

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

e) PaginaJugador: En esta página se muestra la información específica de un jugador para editarlo, eliminarlo o dar de alta un nuevo elemento. De igual forma que en los casos anteriores, 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="FutbolApp.Paginas.PaginaJugador" Title="Jugador">
    <ScrollView>
        <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"/>
            <Button x:Name="btnFoto" Text="Elegir foto" BackgroundColor="LightBlue"
Margin="10" TextColor="Black" WidthRequest="200" HorizontalOptions="Center"
Clicked="btnFoto_Clicked"/>
            <Image x:Name="imgFoto" Source="{Binding Foto}"
VerticalOptions="StartAndExpand" HorizontalOptions="Center" Aspect="AspectFill"
HeightRequest="150" WidthRequest="150" Margin="10"/>
            <Label Text="Fecha de Nacimiento:" FontSize="Medium" TextColor="Black"
HorizontalOptions="Start" Margin="10" HorizontalTextAlignment="Start"/>
            <DatePicker x:Name="dtpFecha" Date="{Binding FechaNacimiento}"
BackgroundColor="White" TextColor="Black"/>
            <ListView x:Name="lsvEquipos" ItemSelected="lsvEquipos_ItemSelected">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout BackgroundColor="White" Spacing="5"
Orientation="Horizontal" Padding="5">
                                <Image Source="{Binding Escudo}" Aspect="AspectFill"
WidthRequest="70" HeightRequest="70"/>
                                <Label Text="{Binding Nombre}" LineBreakMode="WordWrap"
VerticalOptions="Center" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>

    <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 FutbolApp.Modelos;
using FutbolApp.Clases;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace FutbolApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaJugador : ContentPage
    {
        Jugador Jugador;

        public PaginaJugador(Jugador jugador)
        {
            InitializeComponent();
            Jugador = jugador;
            this.BindingContext = jugador;
        }

        private async void btnFoto_Clicked(object sender, EventArgs e)
        {
            var foto = await ServicioImagenes.SeleccionarImagen();
            Jugador.Foto = foto.Path;
            imgFoto.Source = ImageSource.FromFile(foto.Path);
        }

        private async void Guardar_Clicked(object sender, EventArgs e)
        {
            if (Jugador.ID > 0)
                App.BaseDatos.ActualizarJugador(Jugador);
            else
                App.BaseDatos.AgregarJugador(Jugador);

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

        private async void Eliminar_Clicked(object sender, EventArgs e)
        {
            if (await DisplayAlert("Eliminar", "¿Deseas eliminar el jugador?", "Si",
"No"))
            {
                App.BaseDatos.EliminarJugador(Jugador);
                await DisplayAlert("FutbolApp", "Jugador eliminado con éxito", "OK");
                await Navigation.PopAsync();
            }
        }

        private async void lsvEquipos_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            Equipo equipo = (Equipo)e.SelectedItem;
            await Navigation.PushAsync(new PaginaJugadorEquipo(Jugador, equipo, false));
        }
    }
}
```

f) PaginaJugadorEquipo: En esta página se muestra la información específica de un jugador en un equipo particular, es decir, el detalle, la relación de muchos a muchos. Se puede modificar el número de jugador o los goles anotados.

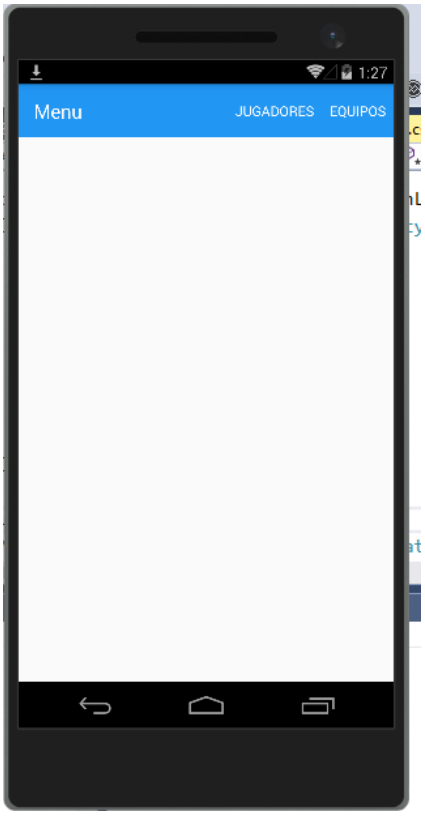
Código XAML:

```
<?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="FutbolApp.Paginas.PaginaJugadorEquipo" Title="Detalle">
    <ScrollView>
        <StackLayout Padding="10" Spacing="10">
            <StackLayout Orientation="Horizontal" Margin="10">
                <Image x:Name="imgFoto" Source="{Binding FotoJugador}"
VerticalOptions="StartAndExpand" HorizontalOptions="Center" Aspect="AspectFill"
HeightRequest="100" WidthRequest="100" Margin="10"/>
                <Label Text="{Binding NombreJugador}" FontSize="20" TextColor="Black"
HorizontalOptions="Start" Margin="10" HorizontalTextAlignment="Start"
FontAttributes="Bold"/>
            </StackLayout>
            <StackLayout Orientation="Horizontal" Margin="10">
                <Image x:Name="imgEscudo" Source="{Binding EscudoEquipo}"
VerticalOptions="StartAndExpand" HorizontalOptions="Center" Aspect="AspectFill"
HeightRequest="100" WidthRequest="100" Margin="10"/>
                <Label Text="{Binding NombreEquipo}" FontSize="20" TextColor="Black"
HorizontalOptions="Start" Margin="10" HorizontalTextAlignment="Start"
FontAttributes="Bold"/>
            </StackLayout>

            <Label Text="Numero" FontSize="Medium" TextColor="Black"
HorizontalOptions="Start" Margin="10" HorizontalTextAlignment="Start"/>
            <Entry Text="{Binding Numero}" Keyboard="Numeric" WidthRequest="100"
FontSize="20" TextColor="Black" BackgroundColor="White" HorizontalOptions="Start"
Margin="10" HorizontalTextAlignment="Start" FontAttributes="Bold"/>

            <Label Text="Goles" FontSize="Medium" TextColor="Black"
HorizontalOptions="Start" Margin="10" HorizontalTextAlignment="Start"/>
            <Entry Text="{Binding Goles}" Keyboard="Numeric" WidthRequest="100"
FontSize="20" TextColor="Black" BackgroundColor="White" HorizontalOptions="Start"
Margin="10" HorizontalTextAlignment="Start" FontAttributes="Bold"/>
        </StackLayout>
    </ScrollView>
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Guardar" Text="Guardar" Order="Primary" Priority="0"
Clicked="Guardar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

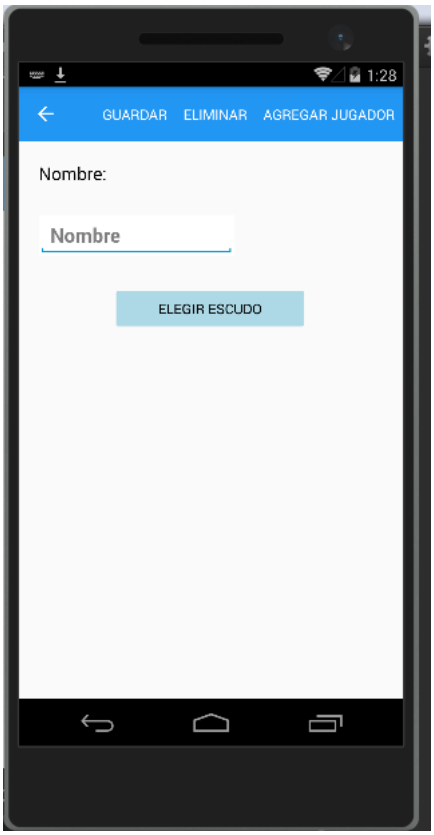
Código C#:

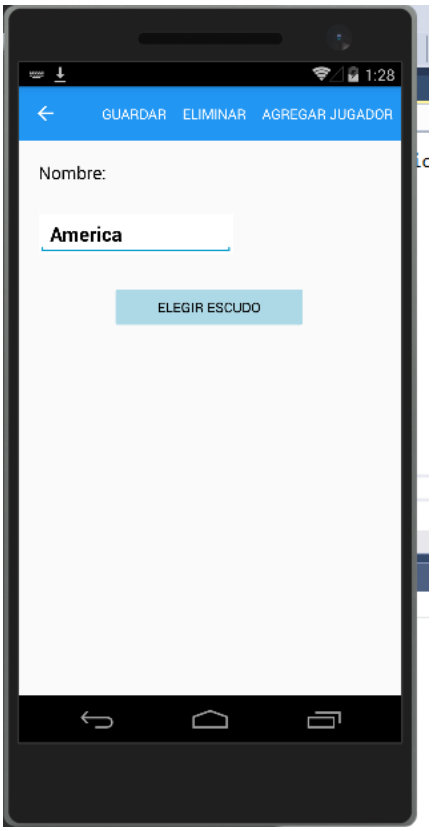
```
using FutbolApp.Clases;
using FutbolApp.Modelos;
using System;

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

namespace FutbolApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaJugadorEquipo : ContentPage
    {
        DetalleJugadorEquipo Detalle;
        JugadorEquipo JugadorEquipo;
        Jugador Jugador;
        Equipo Equipo;
        bool Nuevo;

        public PaginaJugadorEquipo(Jugador jugador, Equipo equipo, bool nuevo)
        {
            InitializeComponent();

            Jugador = jugador;
            Equipo = equipo;
            Detalle = App.BaseDatos.ObtenerDetalleJugadorEquipo(equipo, jugador);
            JugadorEquipo = App.BaseDatos.ObtenerJugadorEquipo(equipo.ID, jugador.ID);
            Nuevo = nuevo;

            this.BindingContext = Detalle;
        }

        private async void Guardar_Clicked(object sender, EventArgs e)
        {
            JugadorEquipo.Goles = Detalle.Goles;
            JugadorEquipo.Numero = Detalle.Numero;

            App.BaseDatos.ActualizarJugador(Jugador, Equipo);
            App.BaseDatos.ActualizarJugadorEquipo(JugadorEquipo);
            await DisplayAlert("FutbolApp", "Datos actualizados con éxito", "OK");
        }
    }
}
```

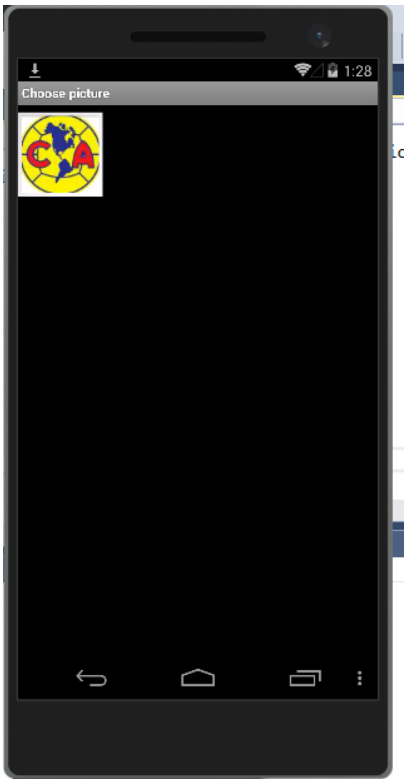
Paso 24. Finalmente, compila y ejecuta la aplicación. Observa el resultado en las diferentes plataformas. Para este ejemplo, se muestran los resultados al ejecutar en un emulador de Android:

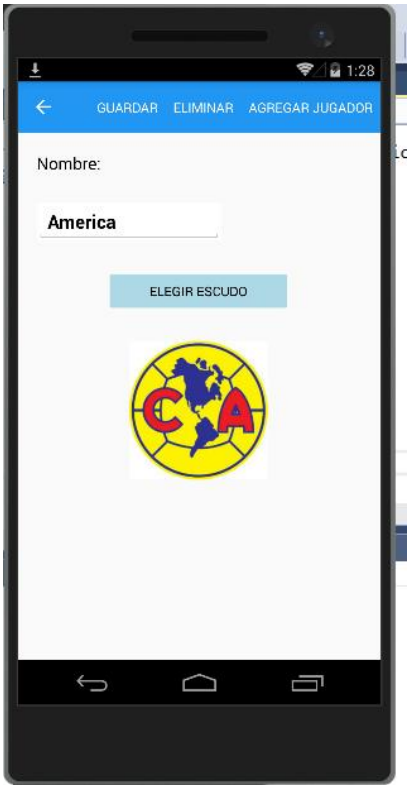

1:27
Menu
JUGADORES
EQUIPOS

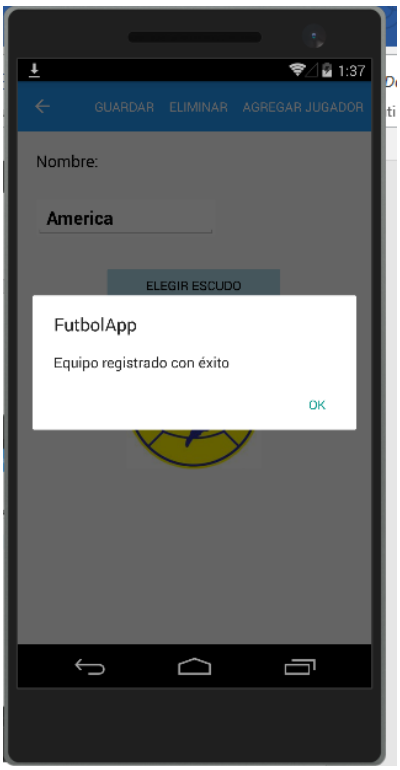

1:27
Equipos
AGREGAR


1:28
GUARDAR
ELIMINAR
AGREGAR JUGADOR
Nombre:
Nombre
ELEGIR ESCUDO

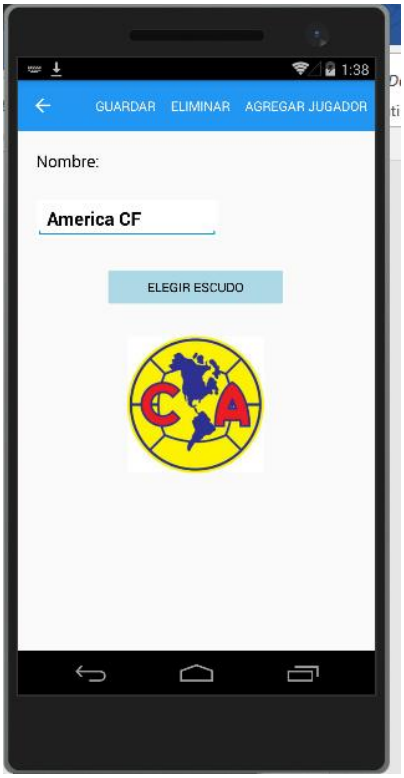

1:28
GUARDAR
ELIMINAR
AGREGAR JUGADOR
Nombre:
America
ELEGIR ESCUDO

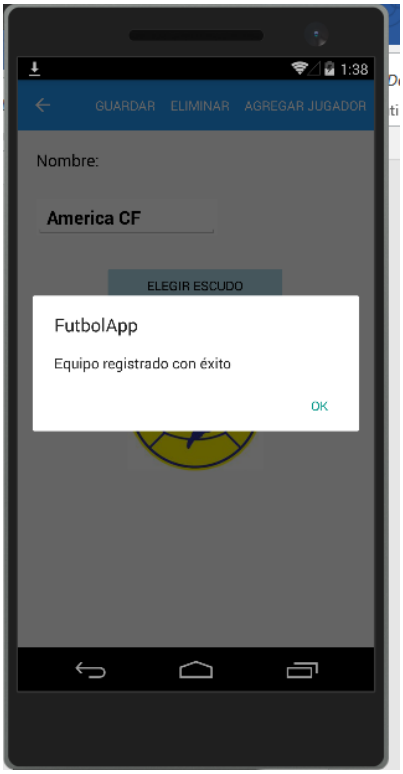

1:28
Choose picture


1:28
GUARDAR ELIMINAR AGREGAR JUGADOR
Nombre:
America
ELEGIR ESCUDO


1:37
GUARDAR ELIMINAR AGREGAR JUGADOR
Nombre:
America
ELEGIR ESCUDO
FutbolApp
Equipo registrado con éxito
OK


1:38
Equipos
AGREGAR
America


1:38
GUARDAR ELIMINAR AGREGAR JUGADOR
Nombre:
America CF
ELEGIR ESCUDO


1:38
GUARDAR ELIMINAR AGREGAR JUGADOR
Nombre:
America CF
ELEGIR ESCUDO
FutbolApp
Equipo registrado con éxito
OK


1:38
Equipos
AGREGAR
America CF