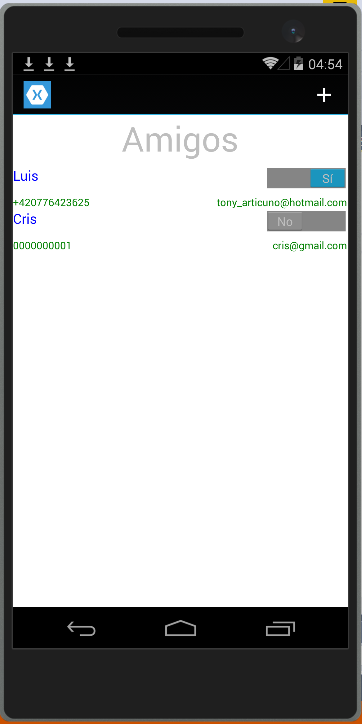
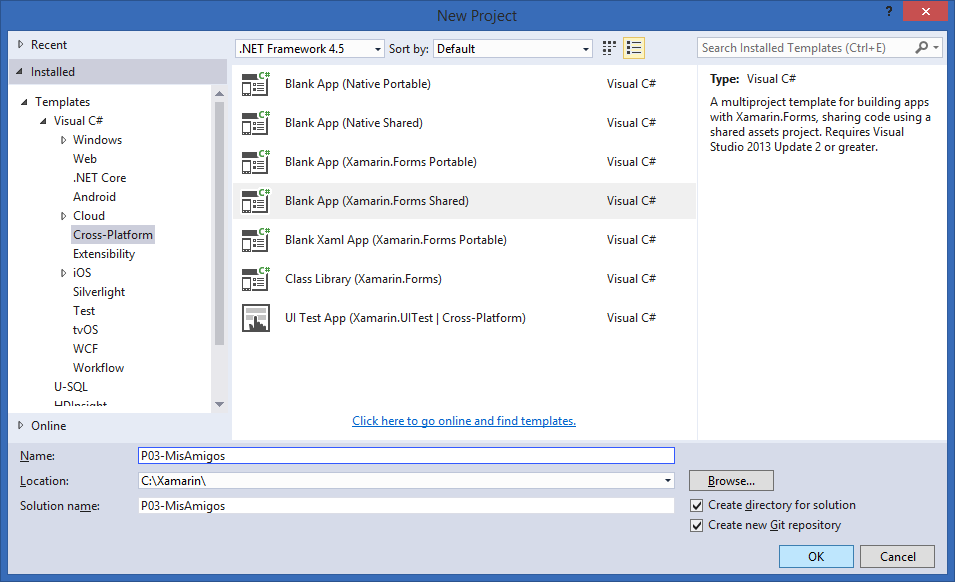
**Práctica 03 – Lista de Amigos**

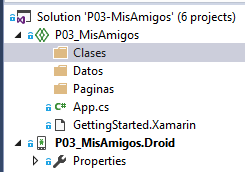
El objetivo de esta práctica es crear una aplicación que nos permita almacenar información de nuestros amigos localmente en nuestra app utilizando **SQLite**.

**Paso 1:** Crea el proyecto **P03-MisAmigos** que es de tipo Xamarin.Forms Shared (localizado bajo la categoría Cross-Platform):

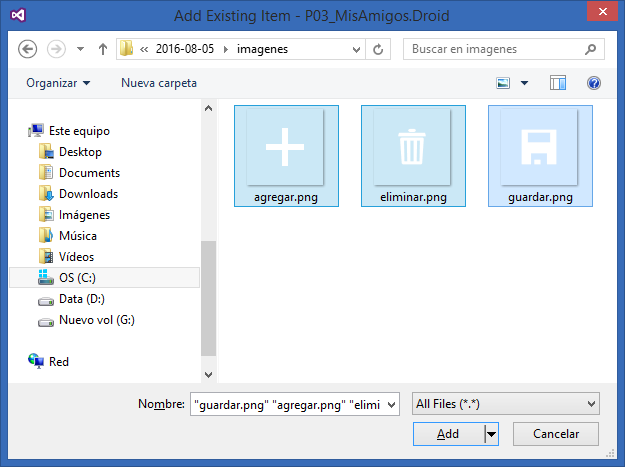


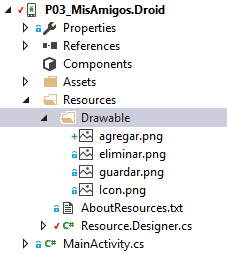
**Paso 2:** Agrega carpetas al proyecto compartido. Da clic derecho sobre el nombre del proyecto compartido (**P03\_MisAmigos**) en el Explorador de Soluciones y selecciona la opción Agregar 🡪 Nueva carpeta. Las carpetas que agregaremos serán: **Clases**, **Paginas** y **Datos**.



**Paso 3:** Agrega imágenes a cada Proyecto que deseas implementar:

a) Para **Android**: En el proyecto **P03\_MisAmigos.Droid**, expande **Resources** y da clic derecho en la carpeta **Drawable**. Selecciona la opción **Agregar** 🡪 **Elemento existente**. Selecciona las 3 imágenes que se incluyen en la práctica, correspondientes los íconos **Agregar, Modificar** y **Eliminar**. Agrégalas.



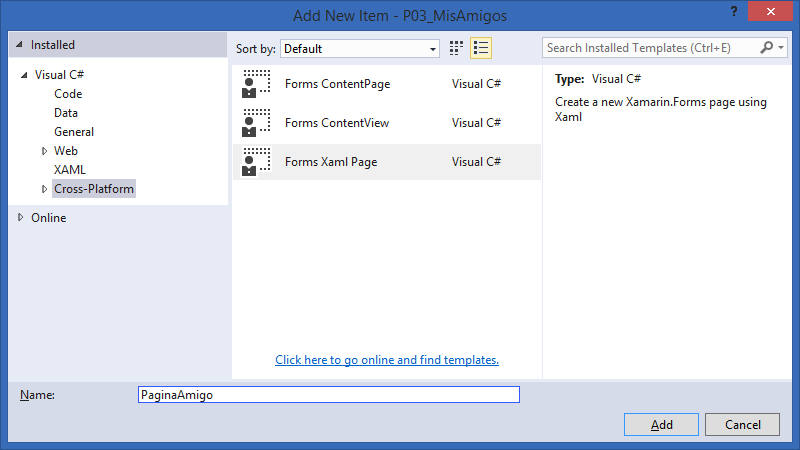


b) Para **iOS**: En el proyecto **P03\_MisAmigos.iOS**, da clic derecho en la carpeta **Resources** y selecciona la opción **Agregar 🡪 Elemento existente**. Selecciona las 3 imágenes que se incluyen en la práctica, correspondientes los íconos **Agregar, Modificar** y **Eliminar**. Agrégalas.

c) Para **Windows Phone, Windows y UWP**: Da clic derecho en el nombre del proyecto respectivo y selecciona la opción **Agregar 🡪 Elemento existente**. Selecciona las 3 imágenes que se incluyen en la práctica, correspondientes los íconos **Agregar, Modificar** y **Eliminar**. Agrégalas.

**Paso 4:** Regresando al proyecto compartido **P03\_MisAmigos**, agrega una nueva página dando clic derecho sobre la carpeta **Paginas** y selecciona **Agregar 🡪 Nuevo elemento**.

Selecciona **Forms Xaml Page** (en la categoría **Cross-Platform**) para crear una nueva página que incluya diseño (**XAML**) y lógica (**C#**), escribe el nombre **PaginaAmigo**:



**Paso 5:** En esta página vamos a mostrar al usuario la información que queremos almacenar de un amigo, para lo cual construiremos la siguiente interfaz:



* Para el **nombre** utilizaremos un View de tipo **Entry**, que es básicamente una caja de texto con la sugerencia (*hint* o *placeholder*) “Nombre”
* Para el **sexo** (Hombre/Mujer) utilizamos un **Picker**, que permite elegir entre varias opciones
* Para el **cumpleaños** utilizamos un **DatePicker** para que el usuario seleccione una fecha y la muestre en el formato [día] de [mes]
* Para el **correo** y **teléfono**, se utilizan 2 **Entry**.
* Finalmente, se puede indicar si es o no un **mejor amigo** mediante un control **Switch**.

Observa también que en la parte superior aparecen 2 íconos. En realidad son ToolbarItems que se agregan a la Toolbar de la página. La posición de la barra de herramientas es una característica dependiente de plataforma (por ejemplo, en Android aparecen en la parte superior, mientras que en Windows Phone aparecen en la parte inferior).

El código **XAML** respectivo se muestra a continuación.

<?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="P03\_MisAmigos.Paginas.PaginaAmigo">

<ContentPage.Content>

<StackLayout Spacing="20" Padding="20">

<Entry x:Name="txtNombre" Placeholder="Nombre" />

<Picker x:Name="pckSexo" Title="Sexo">

<Picker.Items>

<x:String>Hombre</x:String>

<x:String>Mujer</x:String>

</Picker.Items>

</Picker>

<StackLayout Orientation="Horizontal">

<Label Text="Cumpleaños" VerticalOptions="CenterAndExpand"/>

<DatePicker x:Name="dppCumple" Format="dd \de MMMM" VerticalOptions="CenterAndExpand" HeightRequest="50"/>

</StackLayout>

<Entry x:Name="txtCorreo" Placeholder="Correo" />

<Entry x:Name="txtTelefono" Placeholder="Telefono" />

<StackLayout Orientation="Horizontal">

<Label Text="¿Es mejor amigo?" VerticalOptions="CenterAndExpand"/>

<Switch x:Name="swtMejorAmigo" VerticalOptions="CenterAndExpand"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="btnGuardar" Text="Guardar" Priority="0" Clicked="btnGuardar\_Click" Order="Primary" Icon="guardar.png"/>

<ToolbarItem x:Name="btnEliminar" Text="Eliminar" Priority="1" Clicked="btnEliminar\_Click" Order="Primary" Icon="eliminar.png"/>

</ContentPage.ToolbarItems>

</ContentPage>

**Paso 6.** Accede al código de **PaginaAmigo.xaml.cs**. Este archivo representa el “code-behind” de la página. De momento solo crea 2 métodos, que son los manejadores de evento de los botones de la barra de herramientas:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

namespace P03\_MisAmigos.Paginas

{

public partial class PaginaAmigo : ContentPage

{

public PaginaAmigo ()

{

InitializeComponent ();

}

void btnGuardar\_Click(object sender, EventArgs a)

{

}

void btnEliminar\_Click(object sender, EventArgs a)

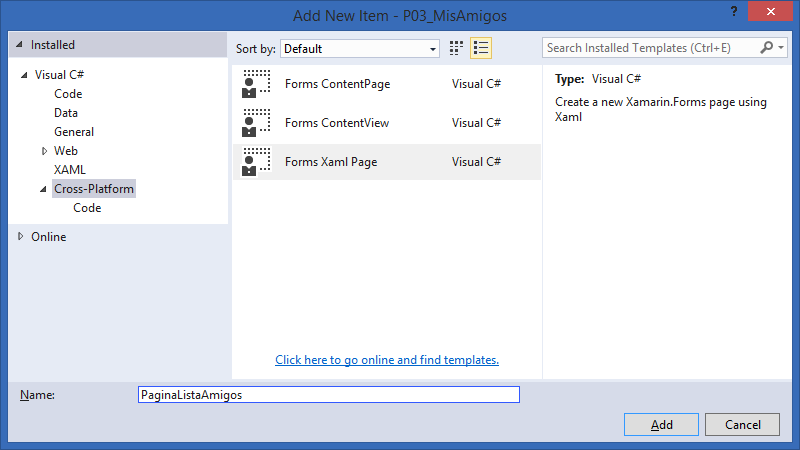
{

}

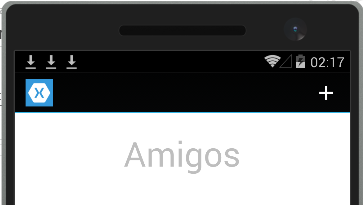
}

}

**Paso 7**. Agrega otra página dando clic derecho sobre la carpeta **Paginas** y selecciona **Agregar 🡪 Nuevo elemento**. Selecciona **Forms Xaml Page** (en la categoría **Cross-Platform**) y escribe el nombre **PaginaListaAmigos**:



**Paso 8.** La interfaz de usuario de esta página es muy sencilla aparentemente, pues consiste en un **Label** (título: Amigos), seguido de un **ListView** que mostrará la **lista de amigos almacenados** en la base de datos y un **ToolbarItem** (botón) en la barra de herramientas que permitirá **agregar** nuevos amigos a la colección llamando a la página anterior. La interfaz se muestra a continuación, seguida del código XAML respectivo:



<ContentPage.Content>

<StackLayout BackgroundColor="White">

<Label Text="Amigos" FontSize="40" HorizontalOptions="Center"/>

<ListView x:Name="lsvAmigos" ItemSelected="lsvAmigos\_Selected" RowHeight="50">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<StackLayout VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">

<StackLayout Orientation="Horizontal">

<Label Text="{Binding Nombre}"

TextColor="Blue" FontSize="16"/>

<Switch IsToggled="{Binding EsMejorAmigo}" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="{Binding Telefono}" HorizontalOptions="StartAndExpand"

TextColor="Green" FontSize="12"/>

<Label Text="{Binding Correo}" HorizontalOptions="EndAndExpand"

TextColor="Green" FontSize="12"/>

</StackLayout>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage.Content>

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="btnNuevo" Text="Nuevo" Priority="0" Clicked="btnNuevo\_Click" Order="Primary" Icon="agregar.png"/>

</ContentPage.ToolbarItems>

La “complejidad” de la interfaz está en el elemento **DataTemplate** del ListView. Esta propiedad es una plantilla que representa el aspecto que **cada elemento** tendrá, además de la información a mostrar a través del enlace a datos (**Binding**). Por ejemplo, se mostrará el nombre, correo y teléfono en un Label junto con un Switch que indica si es un mejor amigo. Estas propiedades se obtienen de un modelo (clase) que no hemos creado aún.

Los elementos del ListView serán suministrados posteriormente utilizando la propiedad **ItemsSource**.

**Paso 9.** Accede al código de **PaginaListaAmigos.xaml.cs**. Este archivo representa el “code-behind” de la página. De momento solo crea 2 métodos, que son los manejadores de evento del botón de la barra de herramientas y de la selección de items del listview:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

namespace P03\_MisAmigos.Paginas

{

public partial class PaginaListaAmigos : ContentPage

{

public PaginaListaAmigos ()

{

InitializeComponent ();

}

private void lsvAmigos\_Selected(object sender, SelectedItemChangedEventArgs e)

{

}

void btnNuevo\_Click(object sender, EventArgs a)

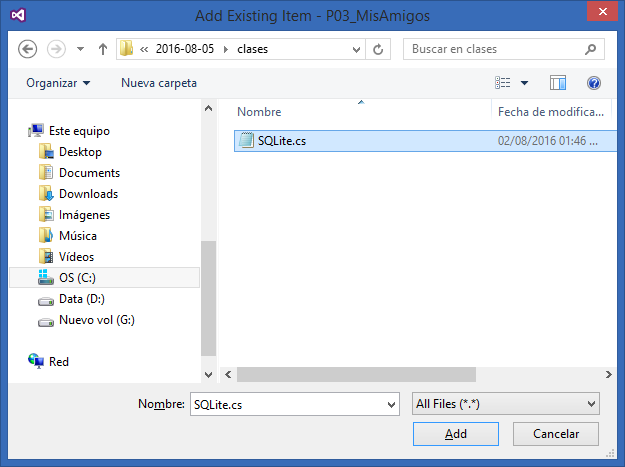
{

}

}

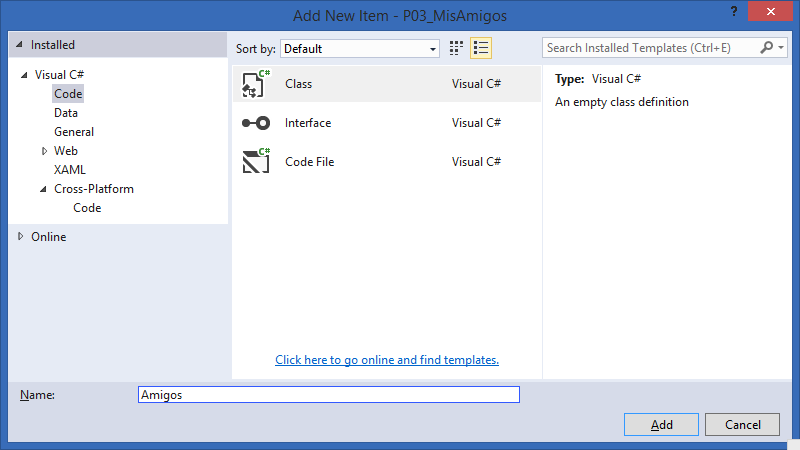
}

**Paso 10.** La librería **SQLite-net** (<https://github.com/praeclarum/sqlite-net>)esde código abierto y nos permite agregar funcionalidad de bases de datos locales en nuestras aplicaciones móviles basadas en .NET y Mono de una manera muy sencilla. Agrega la clase **SQLite.cs** en la carpeta **Datos** dando clic derecho en dicha carpeta y seleccionando **Agregar** 🡪 **Elemento existente**:



La versión más actualizada de **SQLite-Net** siempre se puede descargar del siguiente enlace: <https://github.com/praeclarum/sqlite-net/blob/master/src/SQLite.cs>

**Paso 11.** En la carpeta **Clases** crea una nueva clase llamada **Amigo** (en singular)



El código es muy sencillo, pues contendrá las propiedades que queremos almacenar de un amigo (se mostraron en la interfaz **PaginaAmigos**). Además, utilizando el atributo **Table** de la clase **SQLite**, se indica que será una tabla donde almacenaremos información, mientras que el atributo **PrimaryKey** establece el campo que hará la función de llave primaria (un elemento muy importante de bases de datos).

using System;

using SQLite;

namespace P03\_MisAmigos.Clases

{

[Table("Amigo")]

public class Amigo

{

[PrimaryKey]

public string ID { get; set; }

public string Nombre { get; set; }

public int Sexo { get; set; }

public DateTime Cumple { get; set; }

public string Correo { get; set; }

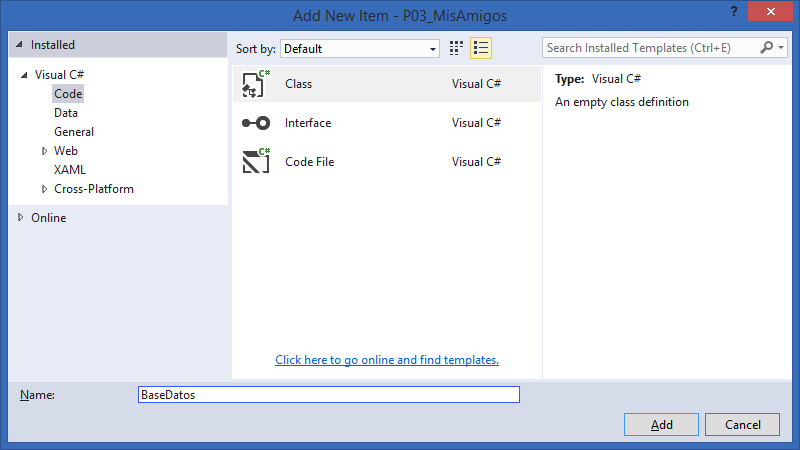
public string Telefono { get; set; }

public bool EsMejorAmigo { get; set; }

}

}

**Paso 12.** En la mismacarpeta **Clases**, crea una clase llamada **BaseDatos**.



Esta clase encapsulará la funcionalidad de crear las tablas y sus operaciones CRUD. El código es el siguiente:

using System;

using System.Linq;

using System.Collections.ObjectModel;

using SQLite;

namespace P03\_MisAmigos.Clases

{

public class BaseDatos : SQLiteConnection

{

public BaseDatos(string path) : base(path)

{

CrearTablas();

}

void CrearTablas()

{

CreateTable<Amigo>();

}

public ObservableCollection<Amigo> ObtenerAmigos()

{

return new ObservableCollection<Amigo>(this.Table<Amigo>().ToList());

}

public Amigo ObtenerAmigo(string id)

{

return (id == "") ? new Amigo() : Table<Amigo>().Where(t => t.ID == id).First();

}

public void GuardarAmigo(string id, string nombre, int sexo, DateTime cumple, string correo, string telefono, bool esMejorAmigo)

{

Amigo amigo = ObtenerAmigo(id);

amigo.Nombre = nombre;

amigo.Sexo = sexo;

amigo.Cumple = cumple;

amigo.Correo = correo;

amigo.Telefono = telefono;

amigo.EsMejorAmigo = esMejorAmigo;

if (id == "")

AgregarAmigo(amigo);

else

ActualizarAmigo(amigo);

}

private void AgregarAmigo(Amigo amigo)

{

amigo.ID = Guid.NewGuid().ToString();

this.Insert(amigo);

}

private void ActualizarAmigo(Amigo amigo)

{

this.Update(amigo);

}

public void EliminarAmigo(string id)

{

Amigo amigo = Table<Amigo>().Where(t => t.ID == id).First();

this.Delete(amigo);

}

}

}

**Paso 13**. Modifica la página de inicio de este proyecto. En la clase **App** del proyecto compartido (**P03\_MisAmigos**) asigna una nueva instancia de **NavigationPage** al objeto **MainPage**. Esto permite que nuestra aplicación maneje más de una página y se comparta información entre páginas utilizando un esquema de Stack, es decir, cada nueva página se muestra encima de la anterior (push). Al presionar el botón de atrás, se puede regresar a la última página visitada (pop). A su vez, el NavigationPage requiere indicar la página específica que se cargará al principio. Utiliza una nueva instancia de **PaginaListaAmigos** (agrega el espacio de nombres **P03\_MisAmigos.Paginas**).

Por otro lado, también se establece la ruta del archivo de la base de datos de nuestra app. Esto será útil para poder llamar los métodos CRUD de las tablas creadas en cualquier parte de nuestra aplicación.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Xamarin.Forms;

using P03\_MisAmigos.Clases;

using P03\_MisAmigos.Paginas;

namespace P03\_MisAmigos

{

public class App : Application

{

public static BaseDatos BD;

public App ()

{

string db = "amigos.db3";

string ruta = System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal), db);

BD = new BaseDatos(ruta);

// The root page of your application

MainPage = new NavigationPage(new PaginaListaAmigos());

}

protected override void OnStart ()

{

// Handle when your app starts

}

protected override void OnSleep ()

{

// Handle when your app sleeps

}

protected override void OnResume ()

{

// Handle when your app resumes

}

}

}

**Paso 14.** Ahora vamos a llamar los métodos creados en la clase **BaseDatos** en el code-behind de nuestras interfaces de usuario.

Comenzamos con **PaginaListaAmigos.xaml.cs**. Necesitamos:

* **Mostrar la lista de amigos al cargar la página** (se sobreescribe el método **OnAppearing** para asegurar que el control ha sido inicializado).
* **Navegar a PaginaAmigos.xaml cuando se seleccione un amigo de la lista**. Se debe enviar la información de qué amigo fue seleccionado.
* **Navegar a PaginaAmigos.xaml cuando se presione el botón Nuevo**. No se envía otra información.

El código completo es el siguiente:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using P03\_MisAmigos.Clases;

namespace P03\_MisAmigos.Paginas

{

public partial class PaginaListaAmigos : ContentPage

{

public PaginaListaAmigos ()

{

InitializeComponent ();

}

protected override void OnAppearing()

{

base.OnAppearing();

lsvAmigos.ItemsSource = App.BD.ObtenerAmigos();

}

private void lsvAmigos\_Selected(object sender, SelectedItemChangedEventArgs e)

{

if (e.SelectedItem != null)

{

Amigo amigo = e.SelectedItem as Amigo;

PaginaAmigo pagina = new PaginaAmigo();

pagina.ID = amigo.ID;

Navigation.PushAsync(pagina);

}

}

void btnNuevo\_Click(object sender, EventArgs a)

{

Navigation.PushAsync(new PaginaAmigo());

}

}

}

**Paso 14.** A continuación, trabajamos sobre **PaginaAmigo.xaml.cs**. Se necesita:

* **Cargar la información del amigo seleccionado** (se utiliza el método OnAppering para asegurar que los controles han sido dibujados).
* **Guardar el amigo con la información proporcionada** al presionar el botón Guardar.
* **Eliminar al amigo seleccionado** al presionar el botón Eliminar.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using P03\_MisAmigos.Clases;

namespace P03\_MisAmigos.Paginas

{

public partial class PaginaAmigo : ContentPage

{

public string ID = "";

public PaginaAmigo ()

{

InitializeComponent ();

}

protected override void OnAppearing()

{

base.OnAppearing();

if (ID != "")

{

Amigo amigo = App.BD.ObtenerAmigo(ID);

txtNombre.Text = amigo.Nombre;

pckSexo.SelectedIndex = amigo.Sexo;

dppCumple.Date = amigo.Cumple;

txtCorreo.Text = amigo.Correo;

txtTelefono.Text = amigo.Telefono;

swtMejorAmigo.IsToggled = amigo.EsMejorAmigo;

}

}

void btnGuardar\_Click(object sender, EventArgs a)

{

string nombre = txtNombre.Text;

int sexo = pckSexo.SelectedIndex;

DateTime cumple = dppCumple.Date;

string correo = txtCorreo.Text;

string telefono = txtTelefono.Text;

bool esMejorAmigo = swtMejorAmigo.IsToggled;

App.BD.GuardarAmigo(ID, nombre, sexo, cumple, correo, telefono, esMejorAmigo);

Navigation.PopAsync();

}

void btnEliminar\_Click(object sender, EventArgs a)

{

if (ID != "")

{

App.BD.EliminarAmigo(ID);

Navigation.PopAsync();

}

}

}

}

**Paso 15**. ¡Listo! Compila y ejecuta la aplicación. Revisa su correcto funcionamiento.

