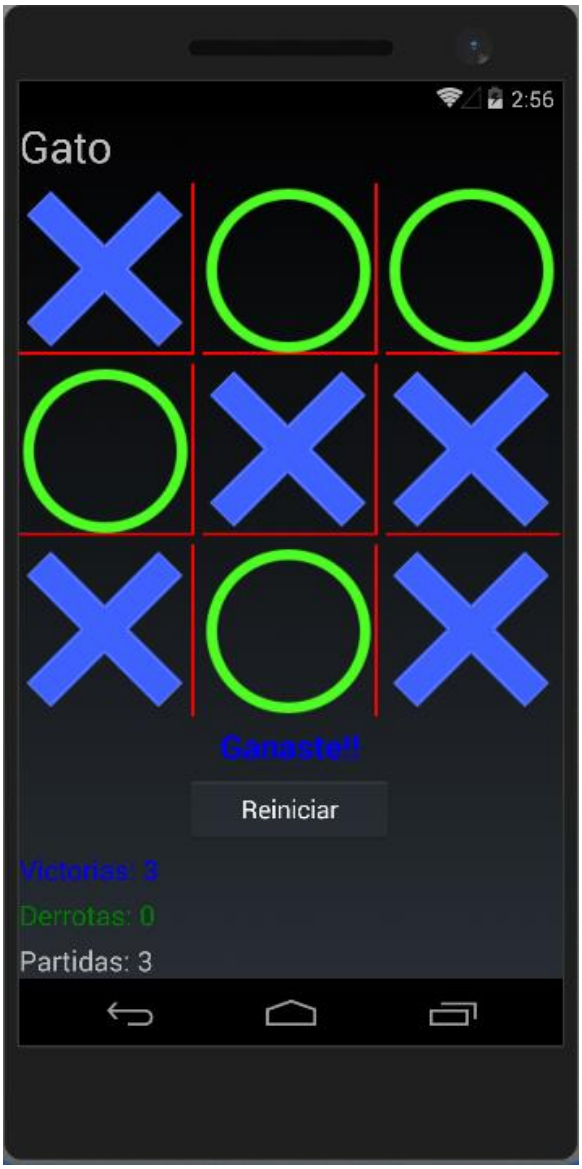
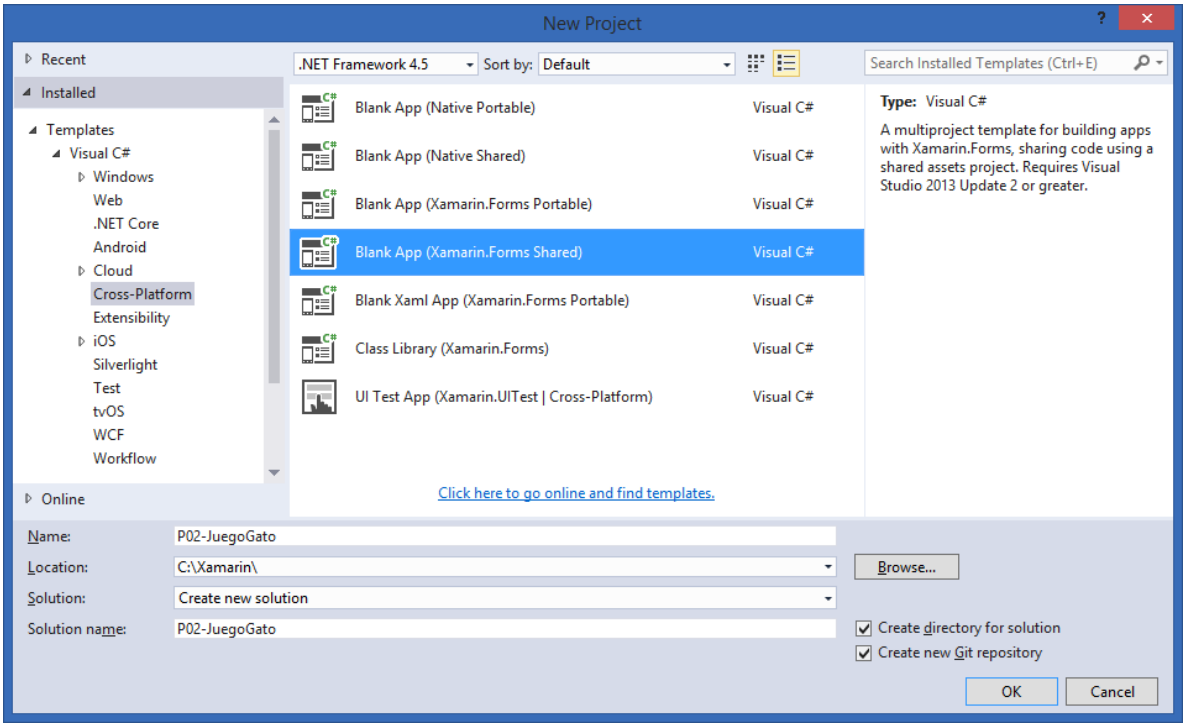
Práctica 02 – El juego del Gato

En esta práctica trabajaremos con los controles **Label, Button** y **StackLayout**, además de un **Grid** construido con código combinado de C# y XAML.

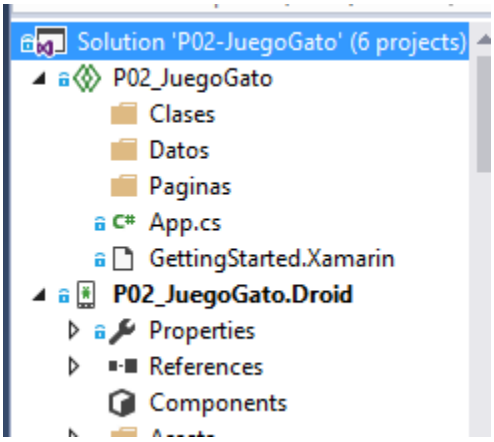
El objetivo es crear una aplicación que simule el juego del gato entre dos jugadores. En prácticas posteriores, añadiremos más funcionalidad a la app.



Paso 1: Crea el proyecto **P02-JuegoGato** que es de tipo Xamarin.Forms Shared (localizado bajo la categoría Cross-Platform):

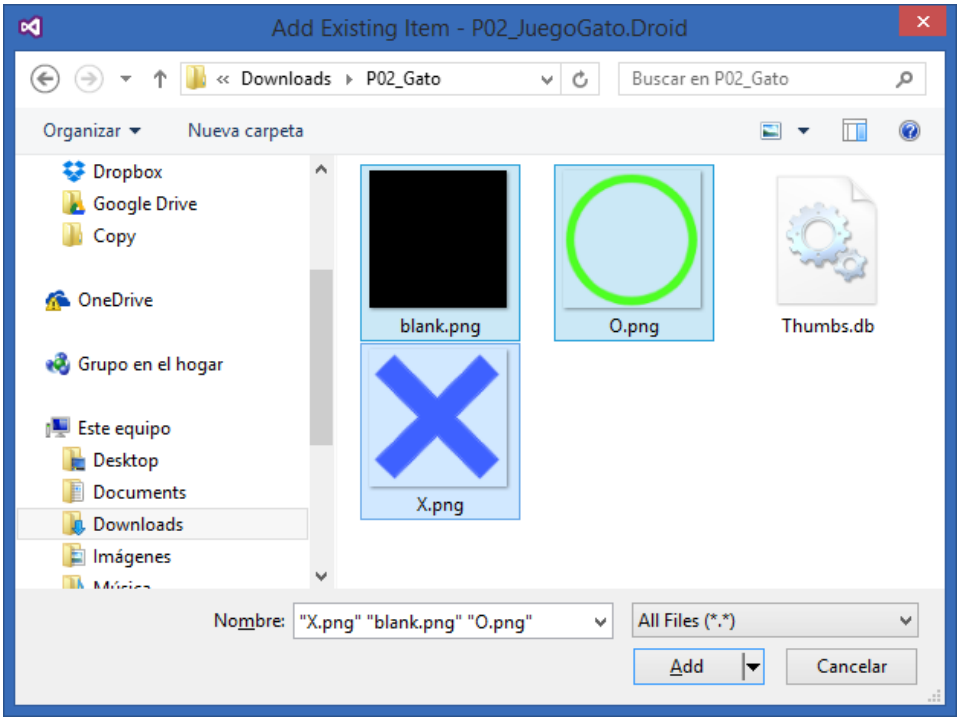


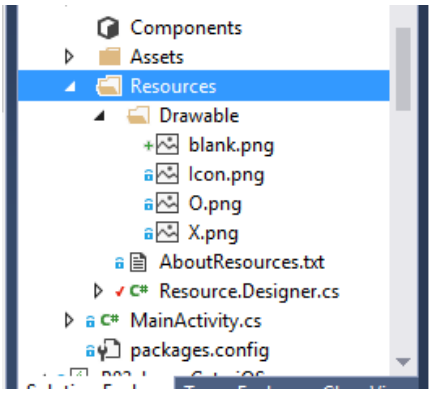
Paso 2: Agrega carpetas al proyecto compartido. Da clic derecho sobre el nombre del proyecto compartido (P02_JuegoGato) en el Explorador de Soluciones y selecciona la opción Agregar → Nueva carpeta. Las carpetas que agregaremos serán: **Clases**, **Paginas** y **Datos**.



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

a) Para **Android**: En el proyecto **P02_JuegoGato.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 a una casilla en blanco, una O y una X. Agrégalas.



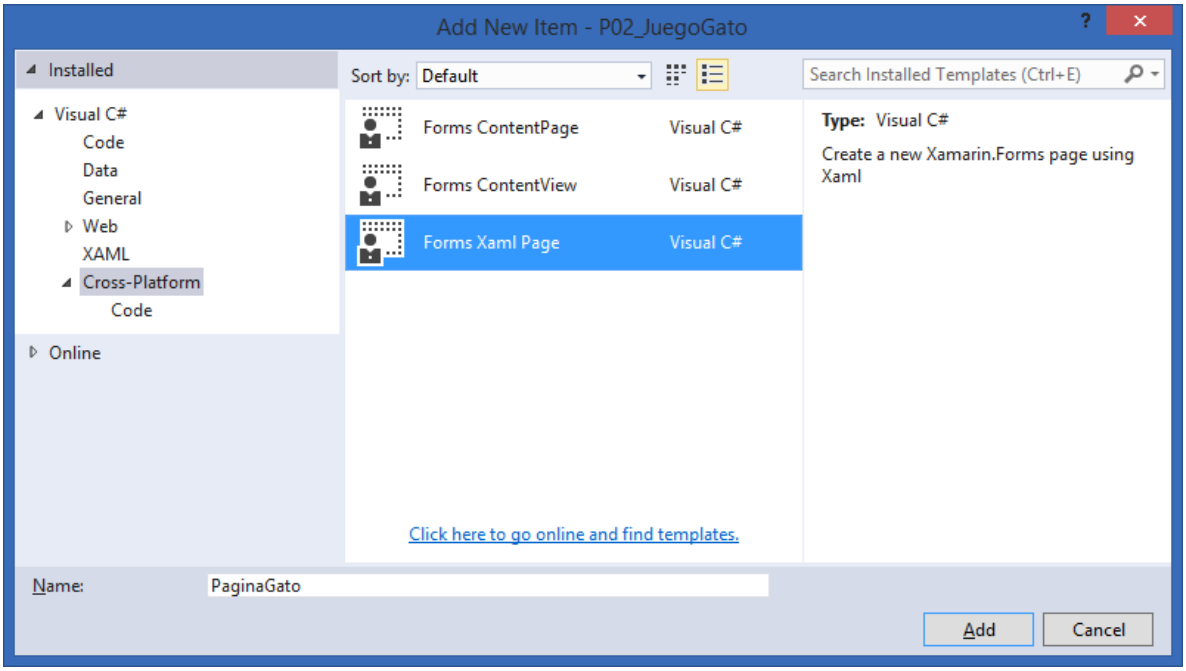


b) Para **iOS**: En el proyecto **P02_JuegoGato.iOS**, da clic derecho en la carpeta **Resources** y selecciona la opción **Agregar → Elemento existente**. Selecciona las 3 imágenes que se incluyen en la práctica, correspondientes a una casilla en blanco, una O y una X. Agrégalas.

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 a una casilla en blanco, una O y una X. Agrégalas.

Paso 4: Regresando al proyecto compartido Add P02_JuegoGato, 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 **PaginaGato**:



Paso 5: A continuación se muestra el código **XAML** de dicha página. Se incluye un **StackLayout** como elemento principal. Los controles contenidos dentro de él son un **Label** (etiqueta, a manera de título), un **Grid** (llamado tablero, donde se visualizará la partida), otro **Label** para mostrar mensajes al usuario y un botón que reiniciará el juego a su estado inicial. Se incluyen al final 3 controles **Label** más que muestran estadísticas de la sesión (victorias, derrotas y partidas).

Cada elemento contiene propiedades para modificar su aspecto; por ejemplo **FontSize** representa el tamaño del texto. Para el caso del **Grid** llamado **tablero**, se utilizan sus propiedades **WidthRequest** y **HeightRequest** para indicar el tamaño que ocupará el control, mientras que **HorizontalOptions** y **VerticalOptions** permiten ubicar al control de manera centrada. Con respecto al botón **btnReiniciar**, el manejador de evento **Clicked** indica que en el código de C# hay un método llamado **btnReiniciar_Clicked** que implementará la funcionalidad.

```
<?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="P02_JuegoGato.Paginas.PaginaGato">
    <StackLayout>
        <Label Text="Gato" FontSize="32"/>

        <Grid x:Name="tablero" WidthRequest="390"
            HeightRequest="390" HorizontalOptions="Center"
            VerticalOptions="Center" />

        <Label x:Name="lblMensaje" Text="" FontSize="24"
            FontAttributes="Bold" HorizontalOptions="Center"/>

        <Button x:Name="btnReiniciar" Clicked="btnReiniciar_Clicked"
            Text="Reiniciar" WidthRequest="150"
            HorizontalOptions="Center"/>

        <Label x:Name="lblVictorias" Text="Victorias: 0"
            FontSize="20" TextColor="Blue"/>

        <Label x:Name="lblDerrotas" Text="Derrotas: 0"
            FontSize="20" TextColor="Green"/>

        <Label x:Name="lblPartidas" Text="0" FontSize="20"/>
    </StackLayout>
</ContentPage>
```

Paso 6. Accede al código de PaginaGato.xaml.cs. Este archivo representa el "code-behind" de la página. Para este caso, se explicará paso a paso el código que se agrega:

a) En los espacios de nombre, asegúrate de tener al menos los siguientes:

```
using System;
using System.Linq;
using Xamarin.Forms;
```

- **System** es un espacio de nombre muy común. Se utilizará para manipular cadenas de texto.
- **System.Linq** nos permite trabajar con colecciones y métodos en un estilo similar a como lo haríamos con una base de datos.
- **Xamarin.Forms** es el espacio de nombres más importante porque permite utilizer toda la funcionalidad proveída por Xamarin en proyectos compartidos, reconocer los controles y más.

b) Copia y pega las siguientes variables **dentro** de la clase **PaginaJuego**:

```
int[,] movs;
Random aleatorio;
bool finDelJuego;
int jugador = 1;
int cpu = -1;
int partidas = 0, victorias = 0, derrotas = 0;
```

- **movs** es un arreglo bidimensional que contiene el estado del tablero (i.e., qué celdas contienen una X, cuáles una O, o un espacio en blanco).
- **aleatorio** es usado para determinar un movimiento aleatorio hecho por el teléfono.
- **finDelJuego** indicia que el juego ha terminado (ya sea porque alguien –humano o teléfono– ha ganado o porque no hay movimientos restantes posibles y es empate).
- **jugador** es un valor (1) usado para identificar al jugador humano en el arreglo de movimientos.
- **cpu** es un valor (-11) usado para identificar al jugador teléfono en el arreglo de movimientos.
- **partidas, victorias,** y **derrotas** son estadísticas del juego.

f) Dentro de la clase, copia y pega los siguientes métodos.

```
void IniciarJuego()
{

}
```

```
        void DibujarTablero()
        {

        }

        void JugarCPU()
        {

        }

        int VerificarGanador()
        {
            return 0;
        }

        int ContarMovimientosRestantes()
        {
            return 0;
        }

        void MostrarMensaje(int resultado)
        {

        }

    void btnReiniciar_Clicked(object sender, System.EventArgs e)
    {

    }
```

¿Para qué sirve cada método? Veamos…

- **IniciarJuego** inicializa variables para el comienzo de juego y llama al método DibujarTablero.
- **DibujarTablero** dibuja líneas y agrega 9 imágenes de casillas en blanco para que el usuario pueda hacer Tap sobre ellas posteriormente y colocar una X en el tablero.
- **JugarCPU** coloca un movimiento aleatorio válido (es la jugada del teléfono) y dibuja una O en el tablero.
- **VerificarGanador** determina si las X o las O ganaron el juego.
- **ContarMovimientosRestantes** devuelve el número de casillas en blanco que aún se pueden jugar.
- **MostrarMensaje** despliega un mensaje indicando el resultado del juego (victoria, derrota o empate).

- **btnReiniciar_Clicked** es el Manejador de Evento para el botón creado en el código XAML. Permite que el usuario reinicie el juego (pero incrementa el contador de derrotas en 1 si aún no ha terminada la partida actual).

g) Llama al método **IniciarJuego** después de **InitializeComponent** en el constructor de la página:

```
public PaginaJuego ()
{
    InitializeComponent ();
    IniciarJuego();
}
```

h) Código para el método **IniciarJuego**:

```
void IniciarJuego()
{
   movs = new int[3, 3] { { 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 } };
   aleatorio = new Random();
   finDelJuego = false;
   lblMensaje.Text = "";
   lblPartidas.Text = String.Format("Partidas: {0}", ++partidas);

   DibujarTablero();
}
```

- El arreglo bidimensional es creado y llenado con 9 ceros (significa que no hay X u O en esa casilla).
- Un generador de números aleatorios es instanciado.
- finDelJuego se coloca en falso porque el juego acaba de comenzar.
- No hay resultado del juego aún, por lo que lblMensaje.Text es una cadena de texto vacía
- lblJPartidas muestra y actualiza el contador de partidas.
- Una llamada al método DibujarTablero indica que la UI será actualizada con los elementos necesarios para que el jugador pueda iniciar la partida.

i) Código para **DibujarTablero**:

```
void DibujarTablero()
{
    tablero.Children.Clear();
    tablero.RowDefinitions.Clear();
    tablero.ColumnDefinitions.Clear();

    for (int i = 0; i < 3; i++)
      tablero.RowDefinitions.Add(new RowDefinition());
```

```
for (int i = 0; i < 3; i++)
  tablero.ColumnDefinitions.Add(new ColumnDefinition());

for (int i = 0; i < 6; i++)
{
  BoxView LineaH = new BoxView()
  {
    Color = Color.Red,
    WidthRequest = 130,
    HeightRequest = 2,
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.End
  };

  Grid.SetRow(LineaH, i / 3);
  Grid.SetColumn(LineaH, i % 3);
  tablero.Children.Add(LineaH);

  BoxView LineaV = new BoxView()
  {
    Color = Color.Red,
    HeightRequest = 130,
    WidthRequest = 2,
    HorizontalOptions = LayoutOptions.End,
    VerticalOptions = LayoutOptions.Center
  };

  Grid.SetRow(LineaV, i / 2);
  Grid.SetColumn(LineaV, i % 2);
  tablero.Children.Add(LineaV);
}

for (int i = 0; i < 9; i++)
{
  Image img = new Image
  {
    Source = ImageSource.FromFile("blank.png"),
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.Center,
    StyleId = String.Format("casilla{0}", i)
  };

  Grid.SetRow(img, i / 3);
  Grid.SetColumn(img, i % 3);
  tablero.Children.Add(img);
```

```
      var tgr = new TapGestureRecognizer
      {
        Command = new Command(() =>
        {
          if (!finDelJuego)
          {
            int x = Grid.GetRow(img);
            int y = Grid.GetColumn(img);

            if (movs[x, y] == 0)
            {
              movs[x, y] = jugador;
              img.Source = ImageSource.FromFile("X.png");

              if (VerificarGanador() == jugador)
                MostrarMensaje(jugador);
              else
                if (ContarMovimientosRestantes() > 0)
                  JugarCPU();
                else
                  MostrarMensaje(0);
            }
          }
        }),
        NumberOfTapsRequired = 1
      };

      img.GestureRecognizers.Add(tgr);
    }
}
```

El método DibujarTablero crea la UI pero también incluye código importante en relación a la funcionalidad de la aplicación, pues añade imágenes que pueden ser presionadas por el usuario y el código para manejar el evento Tap. Una pequeña explicación del código es:

- El grid **tablero** es el contenedor principal.
- El tablero es limpiado de cualquier componente que pueda tener (esto es útil para las partidas posteriores a la primera).
- **Tres renglones** y **3 columnas** son creadas a fin de tener **un grid de 9 celdas** (es el código de los 2 primeros ciclos for).
- **Seis líneas rojas horizontales y seis líneas rojas verticales** son dibujadas para crear un tablero similar al del juego del Gato (tercer ciclo for).
- Cuarto ciclo for:

- Una **imagen** mostrando un **cuadro vacío** es dibujada en cada una de las 9 celdas del grid.
- **TapGestureRecognizer** nos permite colocar un evento **Tap** a cada una de las imágenes.
- **Cuando el jugador hace Tap sobre una celda**, debemos asegurar que:
  - La partida es válida (**finDelJuego no es true**).
  - La celda está vacía (**movimiento válido**)
- **Si no se cumple alguna de las 2 condiciones**, el jugador debe hacer Tap en otra celda o reiniciar la partida (porque el juego ya terminó).
- **Si el movimiento es válido**, entonces una **X es dibujada**, el arreglo **movs** se **actualiza** y el método **VerificarGanador** es llamado para determinar si el jugador ha ganado:
  - En caso afirmativo, el método **MostrarMensaje** es llamado.
  - En caso contrario, el método **ContarMovimientosestantes** es llamado para determinar si el teléfono (cpu) puede colocar una O (**JugarCPU**) o se finaliza el juego con un empate (**MostrarMensaje**).

j) Código para **JugarCPU**:

```
void JugarCPU()
{
    int p;

    do
    {
      p = aleatorio.Next(0, 9);
    } while (movs[p / 3, p % 3] != 0);

    movs[p / 3, p % 3] = cpu;

    var casilla = (tablero.Children
                  .Where(x => x.StyleId ==
                         String.Format("casilla{0}", p))
                  .First()) as Image;
    casilla.Source = ImageSource.FromFile("O.png");

    if (VerificarGanador() == cpu)
      MostrarMensaje(cpu);
    else
      if (ContarMovimientosRestantes() == 0)
        MostrarMensaje(0);
}
```

Este método primero obtiene una celda aleatoria válida (vacía) en la que el teléfono (cpu) pueda colocar una O. Entonces, la imagen O.png es mostrada y el método **VerificarGanador** es llamado

para revisar si esta nueva jugada es ganadora (gana el CPU). Los métodos **MostrarMensaje** y **ContarMovimientosRestantes** son llamados en caso de empate/derrota o si hay al menos una celda vacía válida disponible para jugar, respectivamente.

k) Código para **VerificarGanador**:

```
int VerificarGanador()
{
    int m00 = movs[0, 0];

    if (m00 != 0)
    {
      if (m00 == movs[0, 1] && m00 == movs[0, 2])
        return m00;

      if (m00 == movs[1, 0] && m00 == movs[2, 0])
        return m00;

      if (m00 == movs[1, 1] && m00 == movs[2, 2])
        return m00;
    }

    int m11 = movs[1, 1];

    if (m11 != 0)
    {
      if (m11 == movs[0, 1] && m11 == movs[2, 1])
        return m11;

      if (m11 == movs[1, 0] && m11 == movs[1, 2])
        return m11;

      if (m11 == movs[2, 0] && m11 == movs[0, 2])
        return m11;
    }

    int m22 = movs[2, 2];

    if (m22 != 0)
    {
      if (m22 == movs[0, 2] && m22 == movs[1, 2])
        return m22;

      if (m22 == movs[2, 0] && m22 == movs[2, 1])
        return m22;
```

```
    }

    return 0;
}
```

El método VerificarGanador determina si una de las 8 combinaciones ganadoras en el juego del Gato se cumple. Si la celda contiene un 0, significa que está vacía. En caso contrario, significa que el usuario (1) o el teléfono (-1) ha colocado su marca (X, O) ahí.

l) Código para **ContarMovimientosRestantes**:

```
int ContarMovimientosRestantes()
{
    var ml = from int item in movs
             where item == 0
             select item;

    int x = ml.Count();
    return x;
}
```

Uilizando **LINQ**, el arreglo bidimensional es evaluado y se devuelve el número de celdas vacías.

m) Código para **MostrarMensaje**:

```
void MostrarMensaje(int resultado)
{
    string mensaje = "";
    Color color = Color.Transparent;

    switch (resultado)
    {
      case 1:
        mensaje = "Ganaste!!";
        color = Color.Blue;
        victorias++;
        lblVictorias.Text = String.Format("Victorias: {0}",
victorias);
      break;
      case -1:
        mensaje = "Perdiste!!";
        color = Color.Green;
        derrotas++;
```

```
        lblDerrotas.Text = String.Format("Derrotas: {0}",
derrotas);
      break;
      case 0:
        mensaje = "Empate!!";
        color = Color.Yellow;
      break;
    }

    lblMensaje.TextColor = color;
    lblMensaje.Text = mensaje;

    finDelJuego = true;
}
```

El método anterior muestra uno de tres posibles mensajes: **Ganaste** (en color **Azul**), **Perdiste** (en color **Verde**), o **Empate** (en color **Amarillo**). Además, asigna **true** a la variable booleana **finDelJuego**.

n) Código para btnReiniciar_Clicked:

```
void btnReiniciar_Clicked(object sender, System.EventArgs e)
{
    if (!finDelJuego)
    {
      derrotas++;
      lblDerrotas.Text = String.Format("Derrotas: {0}", derrotas);
    }

    IniciarJuego();
}
```

Finalmente, este método llama al método IniciarJuego para comenzar una nueva partida. Si no se ha decidido un ganador en la partida actual, se incrementa el contador de derrotas y se actualiza el mensaje.

Paso 7. Finalmente, modifica la página de inicio de este proyecto. En la clase **App** del proyecto compartido (**P02_JuegoGato**) asigna una nueva instancia de **PaginaGato** al objeto **MainPage** (agrega el espacio de nombres **P02_JuegoGato.Paginas**):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Xamarin.Forms;
using P02_JuegoGato.Paginas;

namespace P02_JuegoGato
{
	public class App : Application
	{
		public App ()
		{
			// The root page of your application
			MainPage = new PaginaGato();
		}

		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 8. Compila y ejecuta la aplicación. Se debe mostrar el juego y debería funcionar.