Laboratorio

¡Hola, Android Multipantalla!

Versión: 1.0.0
Mayo de 2017

Miguel Muñoz Serafín
@msmdotnet

CONTENIDO

Introducción

En este laboratorio, agregarás una segunda pantalla a la aplicación *PhoneApp* para realizar un seguimiento del historial de números llamados mediante la aplicación. La aplicación final tendrá una segunda pantalla que mostrará el historial de llamadas.

Objetivos

Al finalizar este laboratorio, los participantes serán capaces de:

- Crear aplicaciones Xamarin.Android con múltiple pantalla.
- Agregar código para navegar a una segunda pantalla en una aplicación Xamarin.Android.

Requisitos

Para la realización de este laboratorio es necesario contar con lo siguiente:

- Un equipo de desarrollo con Visual Studio. Los pasos descritos en este laboratorio fueron realizados con Visual Studio 2017 y Windows 10 Professional, sin embargo, los participantes pueden utilizar la versión de Visual Studio 2015 que ya tengan instalada.
- Xamarin para Visual Studio.
- La aplicación *PhoneApp* realizada en el laboratorio anterior.

Tiempo estimado para completar este laboratorio: **45 minutos**.

Ejercicio 1: Extendiendo la aplicación PhoneApp

En este ejercicio, extenderás la aplicación *Phone App* para agregar nueva funcionalidad que permita mostrar una segunda pantalla en la aplicación.

Tarea 1. Modificar la interfaz de usuario actual.

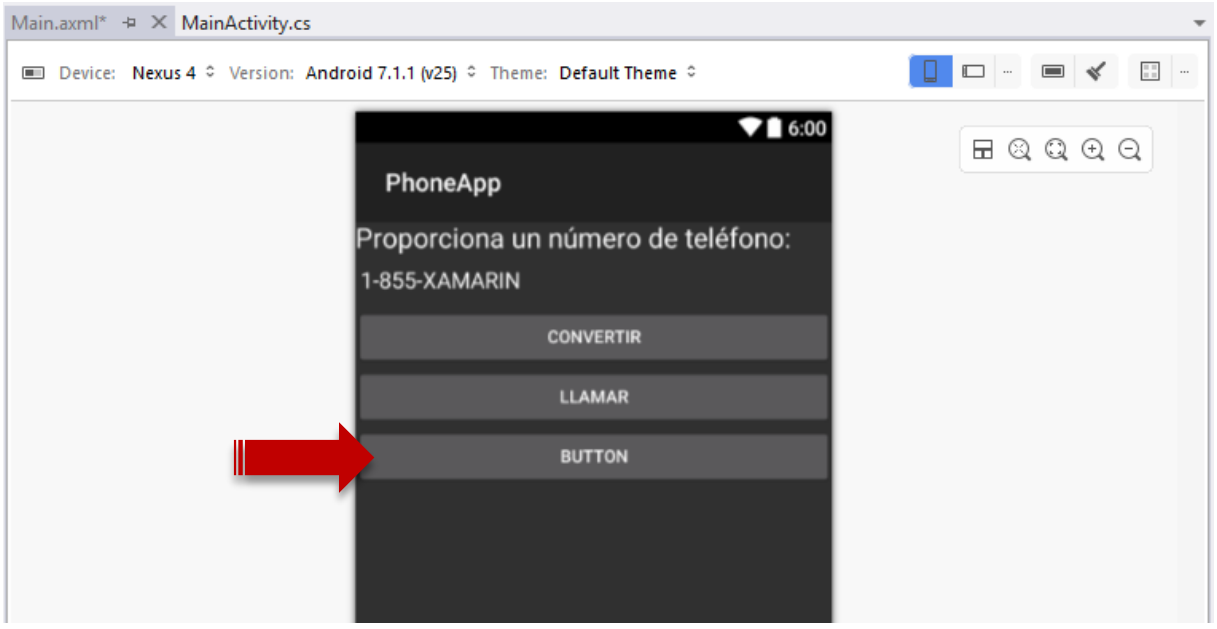
En esta tarea modificarás la interfaz de usuario definida en el archivo de diseño *Main.axml* para agregar un nuevo widget *Button* que permitirá mostrar la pantalla de Historial de llamadas.

1. Abre la aplicación *Phone App* en Visual Studio bajo el contexto del Administrador.

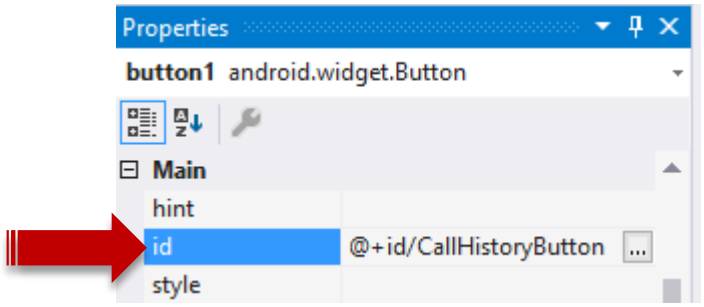
   Empezaremos esta tarea modificando la interfaz de usuario actual.

2. Abre el archivo de diseño de la interfaz de usuario **Main.axml**.

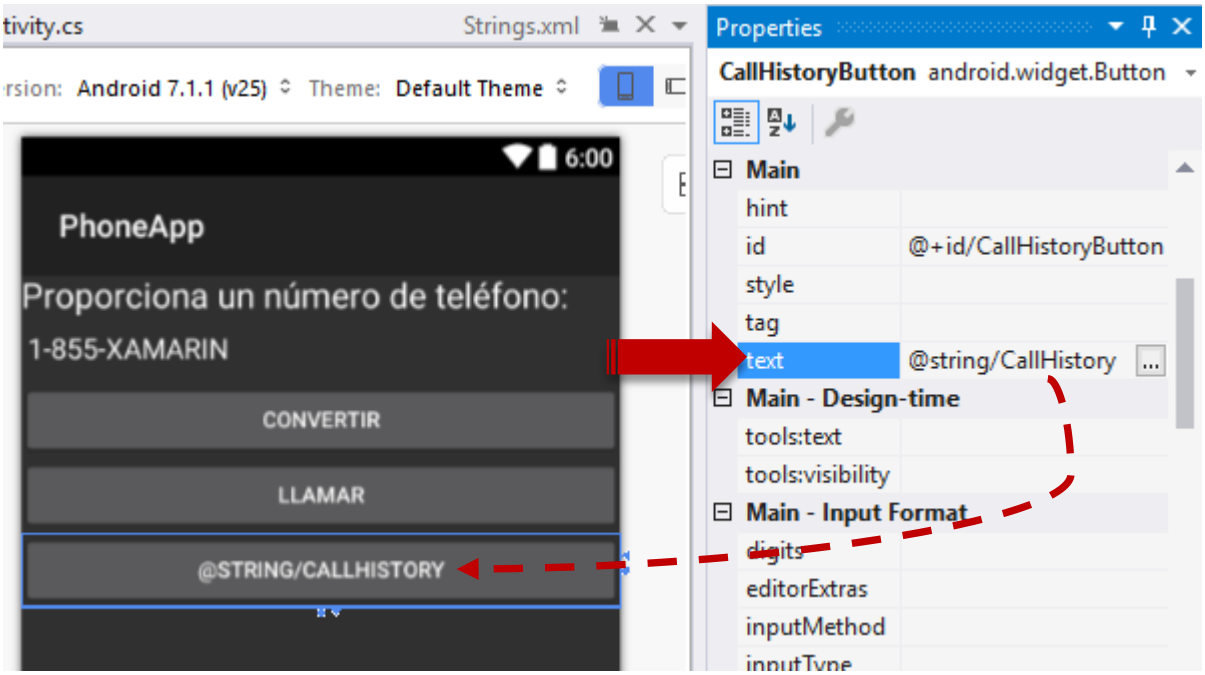
3. Desde la caja de herramientas agrega un widget *Button* a la superficie de diseño y colócalo debajo del botón *Llamar*.



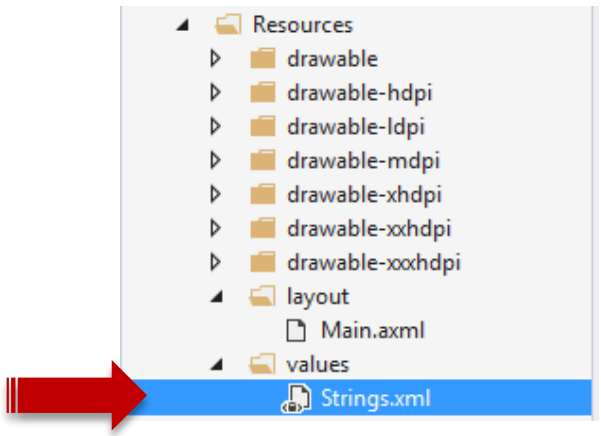
4. En la ventana de propiedades cambia el **id** del botón por ***@+id/CallHistoryButton***.



5. Establece el valor ***@string/CallHistory*** a la propiedad **text** del botón. El diseñador de Android lo interpretará literalmente como un texto, pero haremos algunos cambios para que el texto del botón se muestre correctamente.



6. Abre el archivo de recursos de cadenas **Strings.xml**.

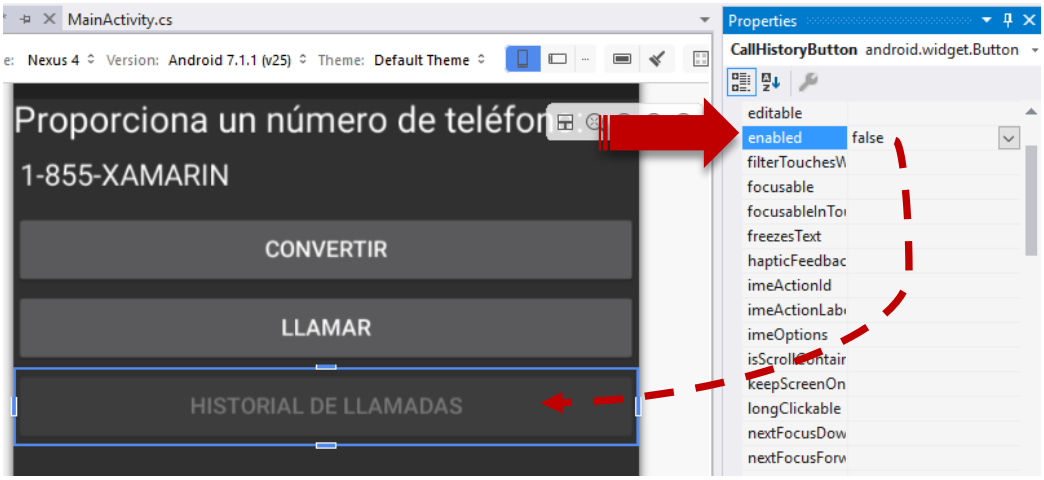


7. Agrega el nombre de cadena *CallHistory* y valor *Historial de llamadas* como se muestra a continuación.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="Hello">Hello World, Click Me!</string>
    <string name="ApplicationName">PhoneApp</string>
    <string name="CallHistory">Historial de llamadas</string>
</resources>
```

8. Guarda el archivo *Strings.xml*. Puedes notar que el texto del botón *CallHistory* se ha actualizado y refleja el nuevo valor de cadena que agregaste.

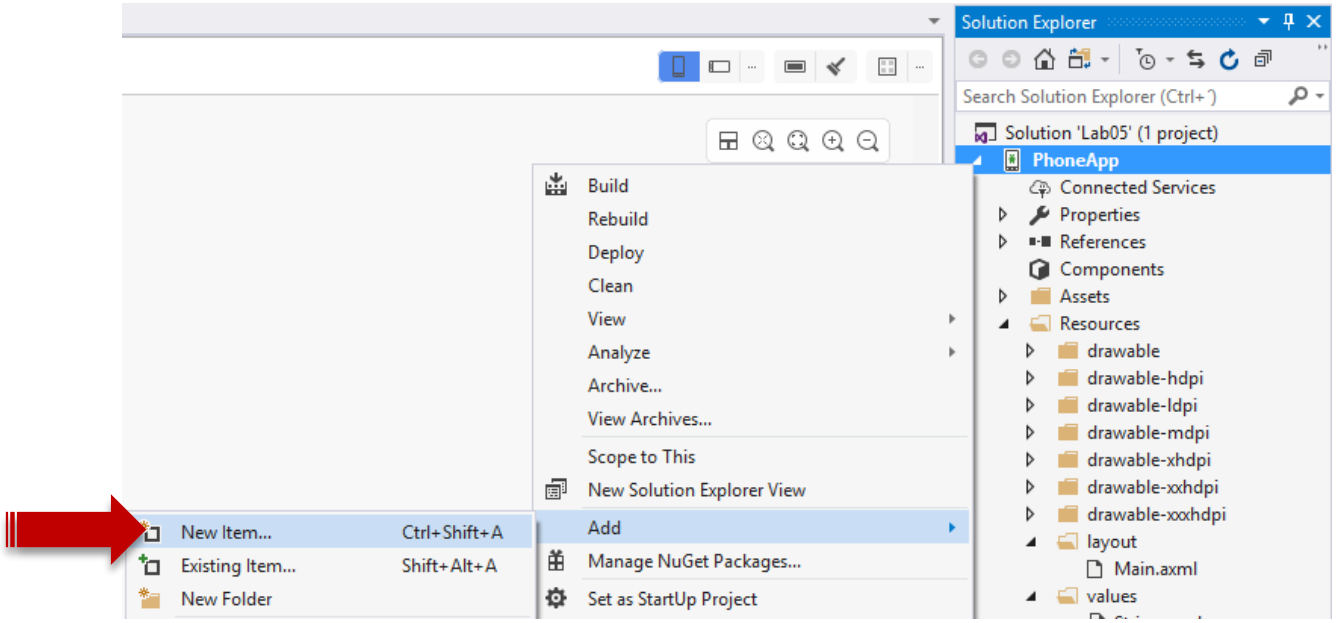
9. Regresa al diseñador de Android y establece el valor ***false*** a la propiedad ***enabled*** del botón *CallHistoryButton* para deshabilitarlo. Esto hará que el botón se muestre más oscuro en la superficie de diseño.



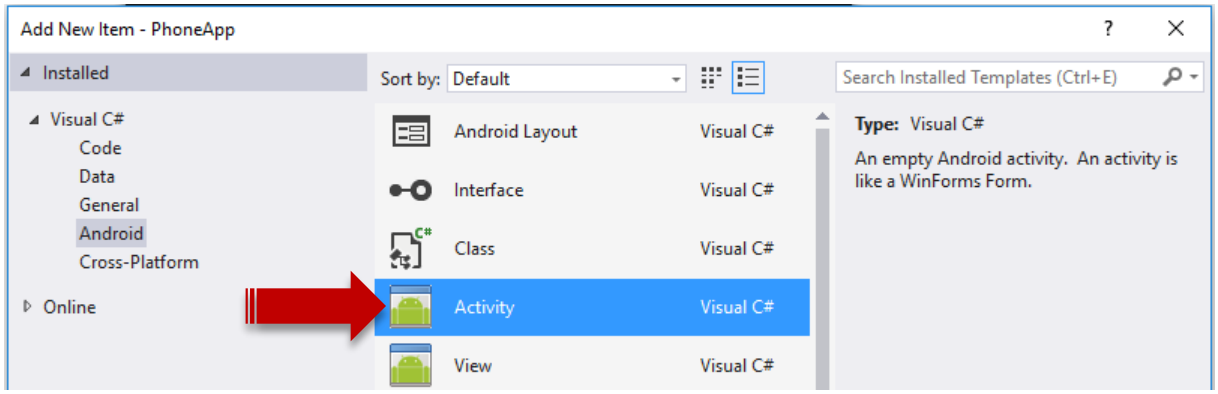
Tarea 2. Agregar una nueva Activity.

En esta tarea agregarás una nueva *Activiy* que dará vida a la segunda pantalla de la aplicación.

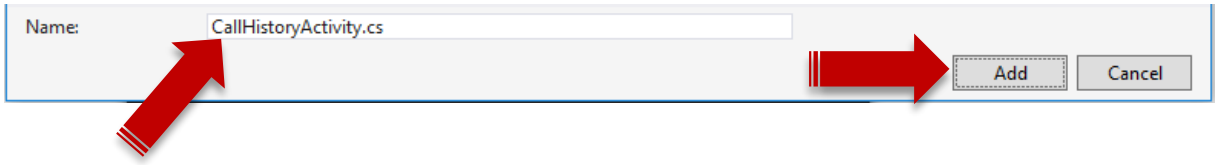
1. En el explorador de soluciones, selecciona la opción **Add** > **New Item** del menú contextual del proyecto Android.



2. En el cuadro de diálogo **Add New Item**, elige la plantilla **Activity**.



3. Escribe **CallHistoryActivity.cs** para el nombre del archivo de la *Activity* y haz clic en **Add** para agregarla.



4. Modifica el valor del parámetro *Label* del atributo *Activity* para que pueda mostrar el valor de la nueva cadena agregada en la tarea anterior.

```
[Activity(Label = "@string/CallHistory")]
public class CallHistoryActivity : Activity
```

5. Modifica la definición de la clase para que herede de la clase **ListActivity**. La clase *ListActivity* es un tipo de *Activity* que muestra una lista de elementos mediante un enlace a una fuente de datos tal como un arreglo.

```
public class CallHistoryActivity : ListActivity
```

6. Agrega el siguiente código al método *OnCreate* para crear el contenido de la nueva *Activity*.

```
// Create your application here
var PhoneNumbers =
    Intent.Extras.GetStringArrayList("phone_numbers") ?? new string[0];
this.ListAdapter =
    new ArrayAdapter<string>(this,
    Android.Resource.Layout.SimpleListItem1, PhoneNumbers);
```

En esta clase, estás creando una *ListActivity* y la estás llenando mediante código, por lo tanto, no necesitas crear un nuevo archivo de diseño .axml para esta *Activity*.

Tarea 3. Modificar la clase MainActivity.

En esta tarea agregarás código a la clase *MainActivity* para implementar la funcionalidad que permita mostrar la segunda pantalla de la aplicación.

1. Abre el archivo *MainActivity.cs*.

   La aplicación recopila los números de teléfono que el usuario ha marcado en la primera pantalla y los pasa a la segunda pantalla. Los números de teléfono se almacenan como una lista de cadenas.

2. Agrega el siguiente código a la clase *MainActivity* para crear una lista vacía que pueda ser llenada con los números telefónicos.

```
[Activity(Label = "Phone App", MainLauncher = true, Icon = "@drawable/icon")]
public class MainActivity : Activity
{
    static readonly System.Collections.Generic.List<string> PhoneNumbers =
        new System.Collections.Generic.List<string>();
```

3. Después de la declaración del botón *CallButton*, agrega el siguiente código para registrar el botón de historial de llamadas.

```
var CallButton = FindViewById<Button>(Resource.Id.CallButton);
var CallHistoryButton = FindViewById<Button>(Resource.Id.CallHistoryButton);
```

4. Agrega el siguiente código al final del método *OnCreate* para definir el manejador del evento *Click* del botón de Historial de llamadas.

```
CallHistoryButton.Click += (sender, e) =>
{
    var Intent = new Android.Content.Intent(this,
        typeof(CallHistoryActivity));
    Intent.PutStringArrayListExtra("phone_numbers",
        PhoneNumbers);
    StartActivity(Intent);
};
```

El siguiente paso es extender la funcionalidad del botón *CallButton* para agregar un número de teléfono a la lista de números y habilitar el botón *CallHistoryButton* cada vez que el usuario marque un nuevo número.

5. Modifica el código del botón neutral del dialogo de alerta. El código deberá ser similar al siguiente.

```
CallButton.Click += (object sender, System.EventArgs e) =>
{
    // Intentar marcar el número telefónico
    var CallDialog = new AlertDialog.Builder(this);
    CallDialog.SetMessage($"Llamar al número {TranslatedNumber}?");
    CallDialog.SetNeutralButton("Llamar", delegate
    {
        // Agregar el número marcado a la lista de números marcados
        PhoneNumbers.Add(TranslatedNumber);
        // Habilitar el botón CallHistotyButton
        CallHistoryButton.Enabled = true;

        // Crear un intento para marcar el número telefónico
        var CallIntent =
            new Android.Content.Intent(Android.Content.Intent.ActionCall);
        CallIntent.SetData(
            Android.Net.Uri.Parse($"tel:{TranslatedNumber}"));
        StartActivity(CallIntent);
    });
```

6. Guarda todos los cambios.

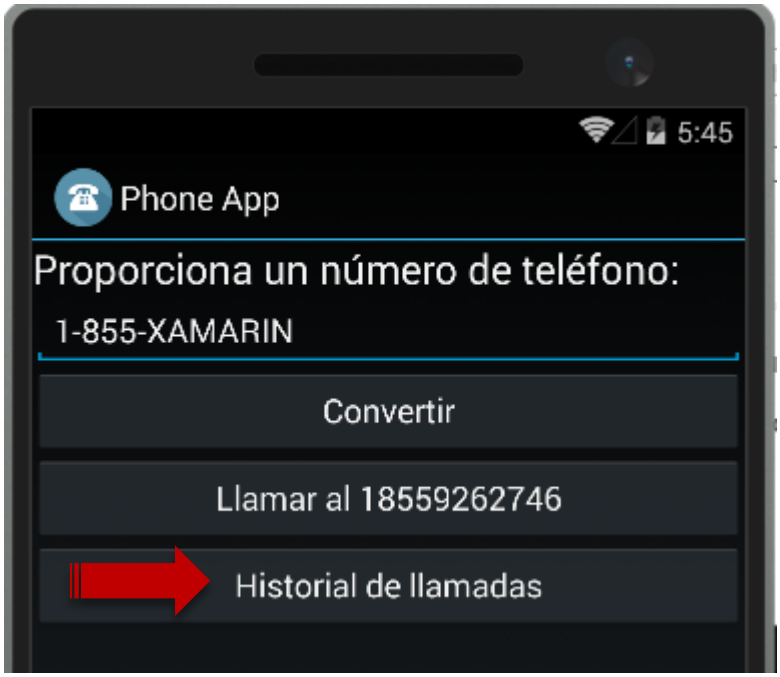
7. Ejecuta la aplicación en un emulador o dispositivo Android. Se mostrará una pantalla similar a la siguiente.

8. Presiona el botón **Convertir**.

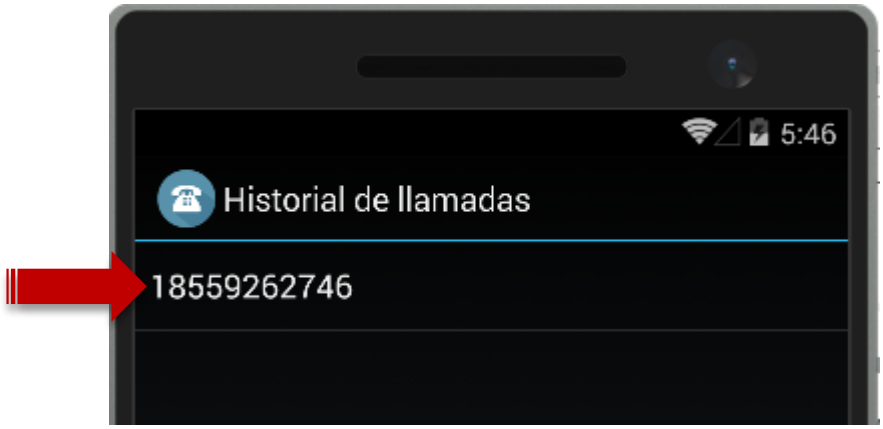
9. Presiona el botón **Llamar al 18559262746**.

10. Presiona el botón **Llamar** del diálogo de alerta.

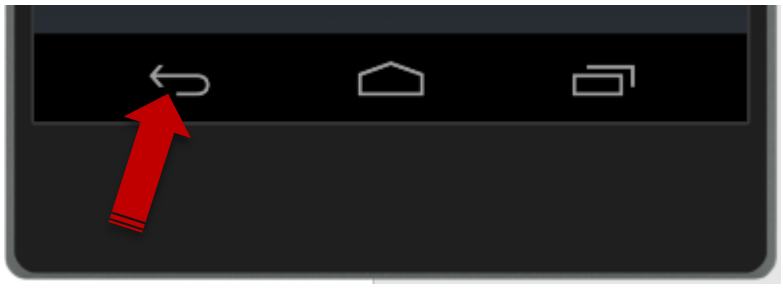
11. Regresa a la aplicación y presiona el botón de Historial de llamadas.



12. La segunda pantalla será mostrada con la lista de números telefónicos marcados (en este caso uno).

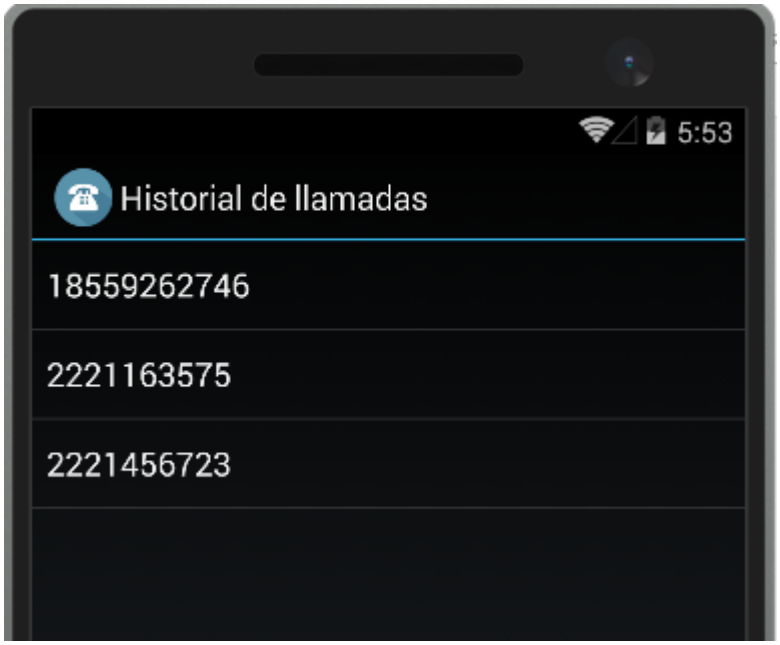


13. Haz clic en el botón regresar para ir a la primera pantalla.



14. Intenta convertir más números y realizar llamadas.

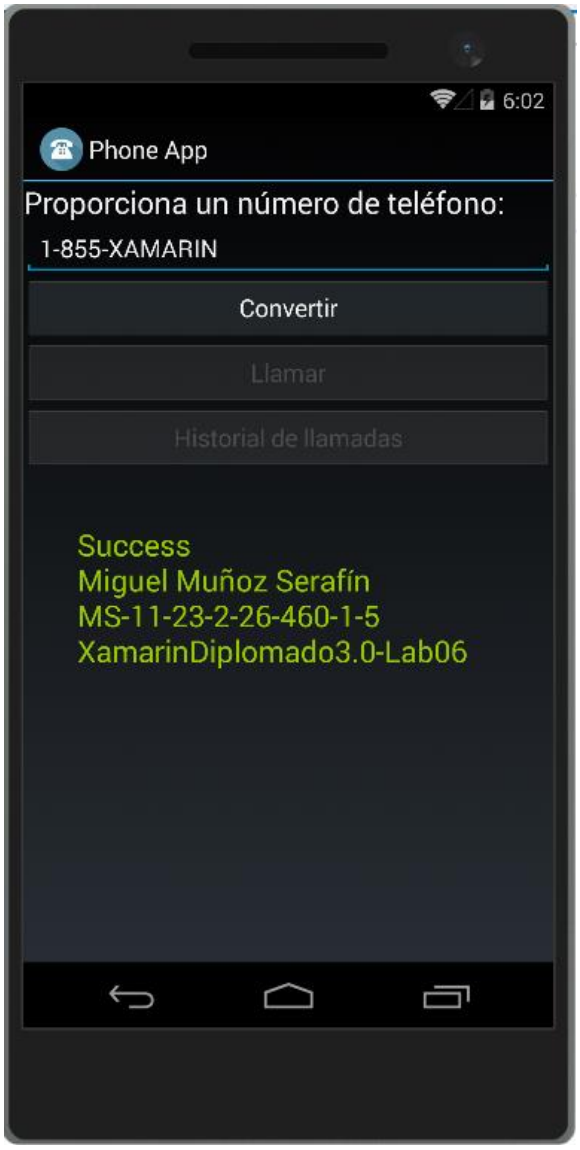
15. Accede nuevamente a la pantalla del historial de llamadas. Deberá mostrarse la lista de números llamados.



Ejercicio 2: Validando tu actividad

Para validar tu actividad, realiza los cambios necesarios a tu aplicación para que utilice el ensamblado **SALLab06.dll** en lugar del ensamblado anterior **SALLab05**.

Al ejecutar tu aplicación te deberá mostrar el mensaje **XamarinDiplomado3.0-Lab06**.



Cuando tu actividad se haya validado exitosamente puedes ver el estatus en el siguiente enlace: https://ticapacitacion.com/evidencias/xamarin30.

Recuerda eliminar de tu código los datos de tus credenciales de acceso a la plataforma de TI Capacitación.

Nota: Es probable que recibas un correo similar al siguiente.

Tu código de lab no es válido, revisa que estés utilizando un código de reto válido. Si tienes preguntas o dudas por favor contacta a dxaudmx@microsoft.com

Puedes hacer caso omiso al mensaje.

Si encuentras problemas durante la realización de esta actividad, puedes solicitar apoyo en los grupos de Facebook siguientes:

https://www.facebook.com/groups/iniciandoconxamarin/
https://www.facebook.com/groups/xamarindiplomadoitc/

Resumen

¡Felicidades! Haz completado tu primera aplicación Xamarin.Android con múltiple pantalla. Ahora es tiempo de analizar las herramientas y habilidades que has aprendido.

¿Qué te pareció este laboratorio?

Comparte tus comentarios en twitter y Facebook utilizando el hashtag ***#XamarinDiplomado.***