Laboratorio

¡Hola, Android!

Versión: 1.0.0
Mayo de 2017

Miguel Muñoz Serafín
@msmdotnet

CONTENIDO

INTRODUCCIÓN

EJERCICIO 1: CREANDO UNA APLICACIÓN ANDROID.

Tarea 1. Crear un proyecto Android.

Tarea 2. Diseñar la interfaz de usuario de la aplicación.

Tarea 3. Agregar la lógica de conversión.

Tarea 4. Agregar código para mostrar la interfaz de usuario.

Tarea 5. Asignar el permiso de realizar llamadas telefónicas.

Tarea 6. Agregar el toque final a la aplicación.

Tarea 7. Probar la aplicación.

EJERCICIO 2: VALIDANDO TU ACTIVIDAD

Tarea 1. Agregar los componentes de la Capa de acceso a Servicio.

RESUMEN

Introducción

En este laboratorio crearás una aplicación que permitirá al usuario proporcionar un número telefónico que incluya letras y números para posteriormente convertirlo a un número telefónico que incluya únicamente números, por ejemplo, si el número proporcionado es *1-855-XAMARIN*, la aplicación lo convertirá a su equivalente numérico *18559262746*. Después de realizar la conversión, la aplicación dará la opción de realizar una llamada a ese número telefónico.

Objetivos

Al finalizar este laboratorio, los participantes serán capaces de:

- Crear un proyecto Android.
- Diseñar una interfaz de usuario sencilla de una aplicación Android.
- Agregar archivos de código a un proyecto Android.
- Agregar código para mostrar la interfaz de usuario de una aplicación Android.
- Asignar permisos de acceso a recursos en una aplicación Android.
- Personalizar el título e icono de una aplicación Android.
- Desplegar la aplicación hacia un emulador 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.

Tiempo estimado para completar este laboratorio: **60 minutos**.

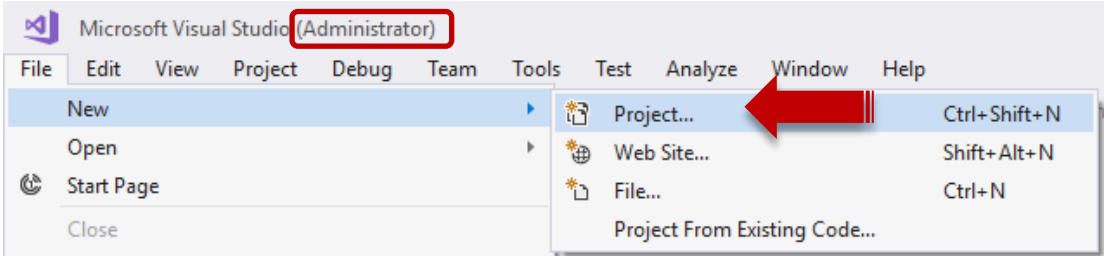
Ejercicio 1: Creando una aplicación Android

En este ejercicio crearás una aplicación Android que te permitirá introducirte en las herramientas, conceptos y pasos para crear y desplegar aplicaciones Xamarin.Android.

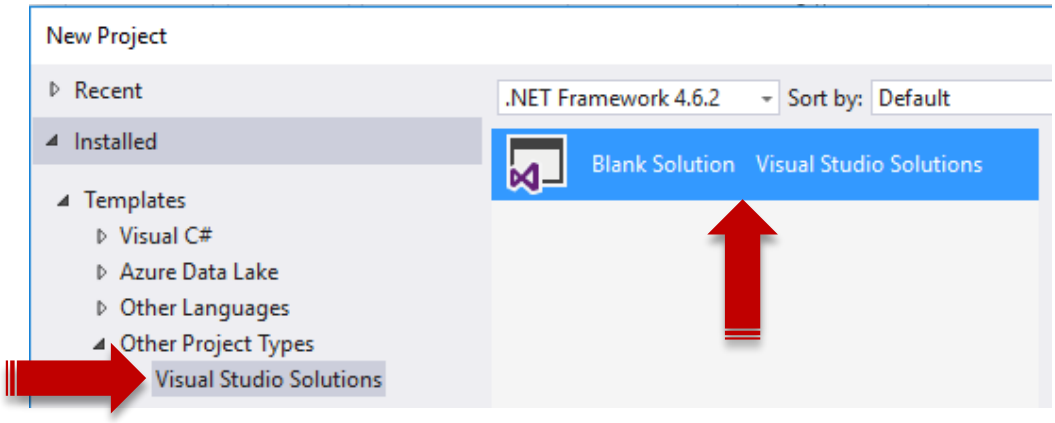
Tarea 1. Crear un proyecto Android.

Realiza los siguientes pasos para crear una solución vacía y agregarle un proyecto Android.

1. Abre Visual Studio en el contexto del Administrador.
2. Selecciona la opción **File** > **New** > **Project**.

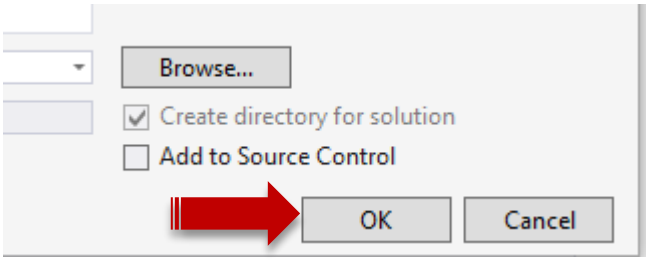


3. En la ventana **New Project** selecciona la plantilla **Blank Solution**.



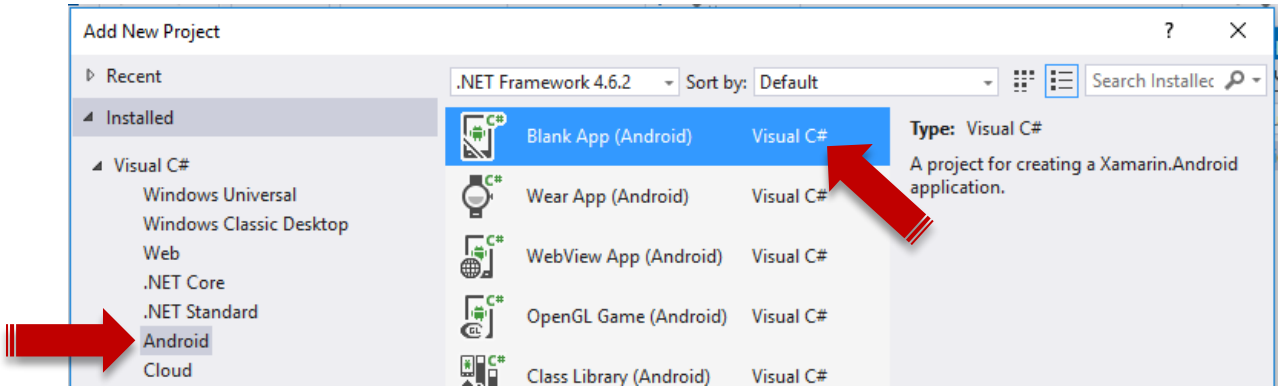
4. Proporciona el nombre y ubicación de la solución.

5. Haz clic en **OK** para crear la solución en blanco.

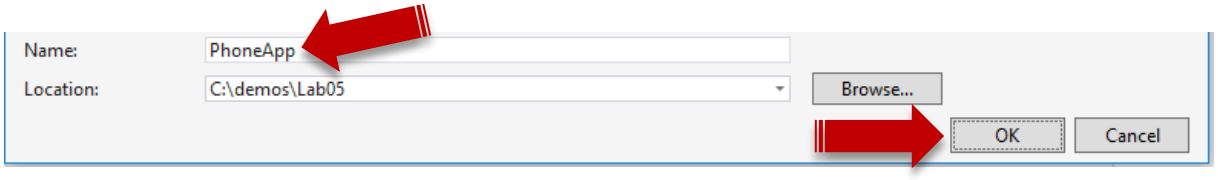


6. Para agregar un nuevo proyecto Android a la solución, selecciona la opción **File** > **Add** > **New Project…** de la barra de menús de Visual Studio.

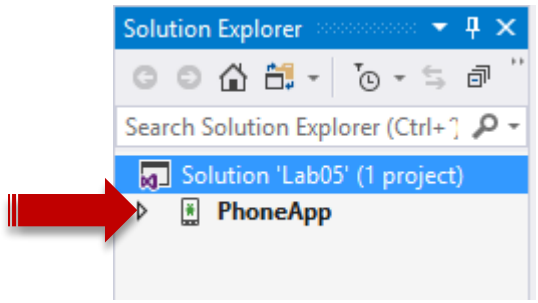
7. En la ventana **Add New Project** selecciona la plantilla **Blank App (Android)**.



8. Asigna un nombre al proyecto Android y haz clic en **OK** para agregarlo a la solución.



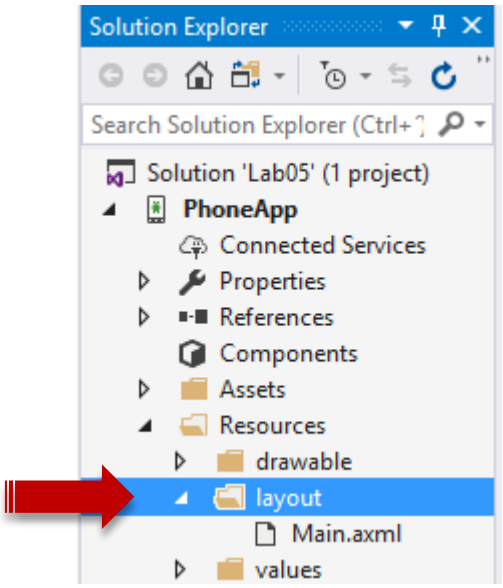
El explorador de soluciones mostrará la solución con el proyecto Android agregado.



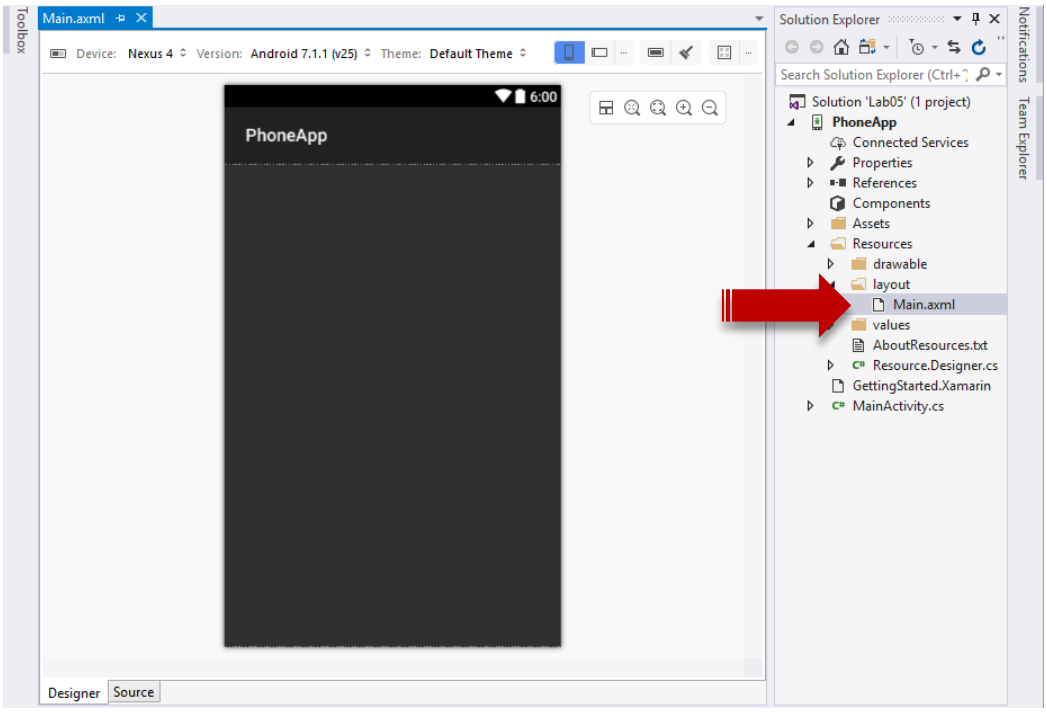
Tarea 2. Diseñar la interfaz de usuario de la aplicación.

Una vez agregado el proyecto Android, empezaremos por diseñar la interfaz de usuario de la aplicación.

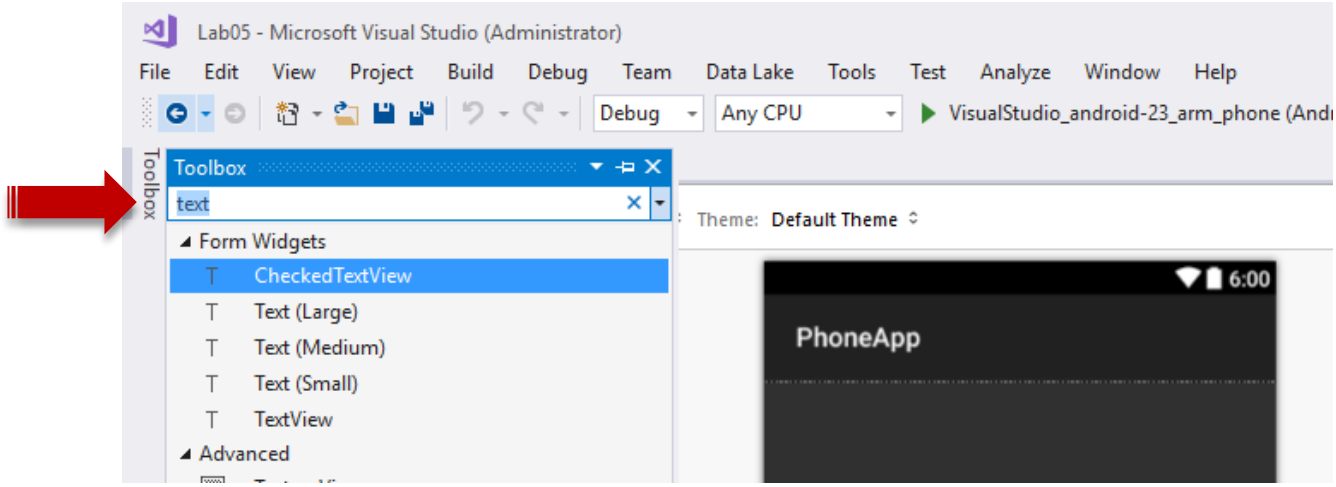
1. Expande el folder **Resources\Layout**.



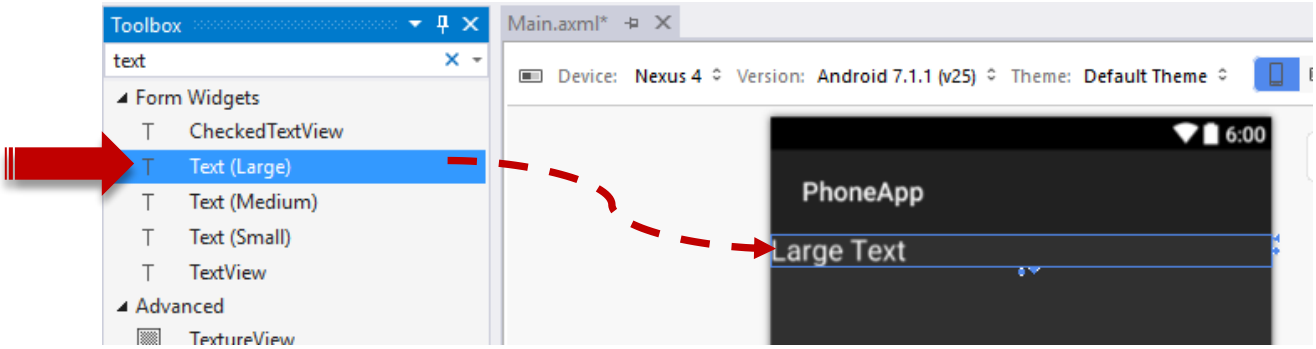
2. Haz doble clic sobre el archivo **Main.axml** para abrirlo en el ***Diseñador de Android (Android Designer)***. Este es el archivo del diseño de la interfaz de usuario de la aplicación.



3. En el campo de búsqueda de la caja de herramientas, escribe ***text***.

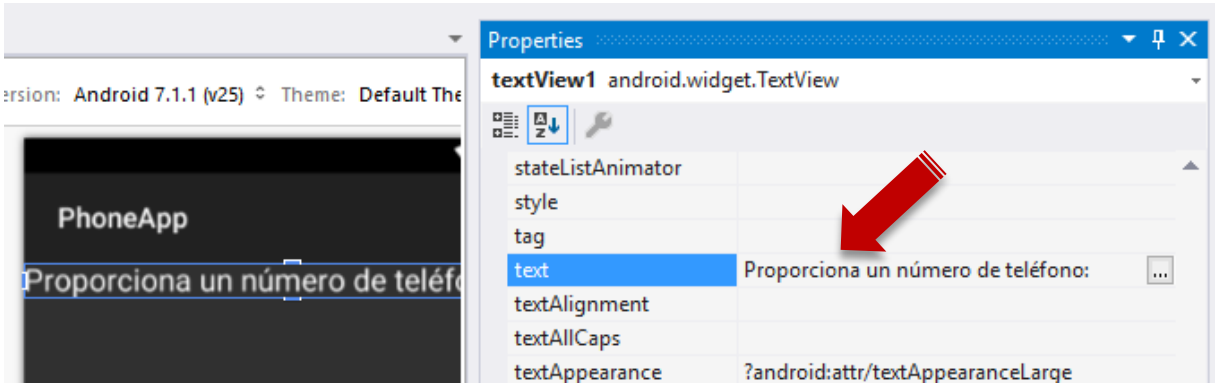


4. Arrastra el widget **Text (Large)** y suéltalo sobre cualquier parte de la superficie de diseño. El widget será posicionado en la parte superior.

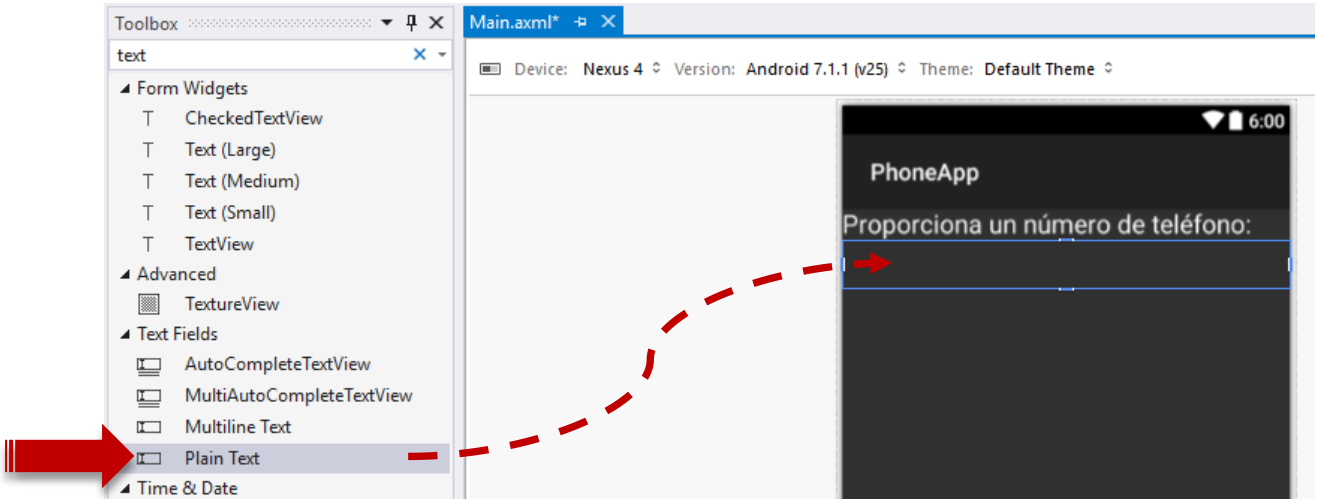


5. Haz clic sobre el widget agregado para seleccionarlo y presiona **F4** para abrir la ventana de propiedades del widget.

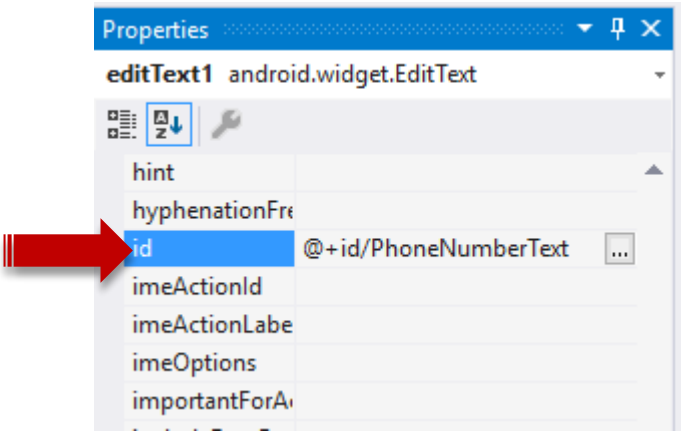
6. En la ventana de propiedades del widget, modifica el valor de la propiedad **text** por lo siguiente: ***Proporciona un número de teléfono:***



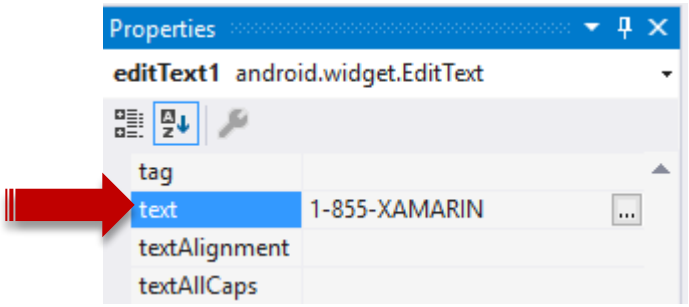
7. Desde la caja de herramientas, arrastra un widget **Plain Text** y suéltalo sobre la superficie de diseño para colocarlo debajo del widget **Text (Large)**.



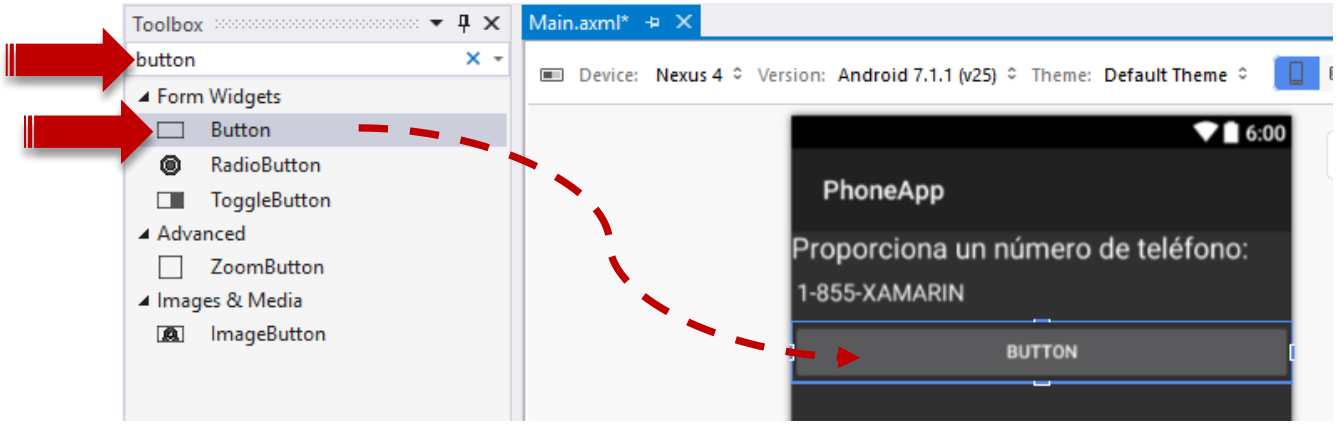
8. En la ventana de propiedades del widget **Plain Text** localiza la propiedad **id** y modifica su valor por: **@+id/PhoneNumberText**.



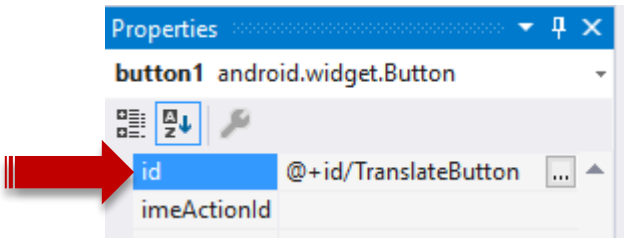
9. En la ventana de propiedades del widget **Plain Text** localiza la propiedad **text** y modifica su valor por: **1-855-XAMARIN**.



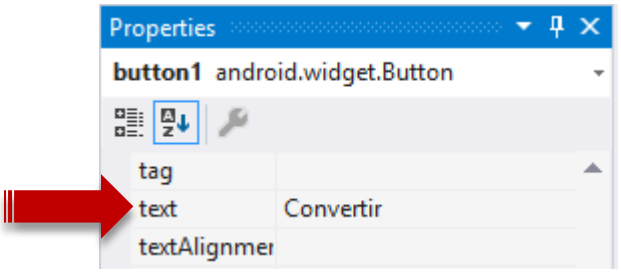
10. Desde la caja de herramientas, arrastra un widget **Button** y suéltalo sobre la superficie de diseño para colocarlo debajo del widget **Plain Text**.



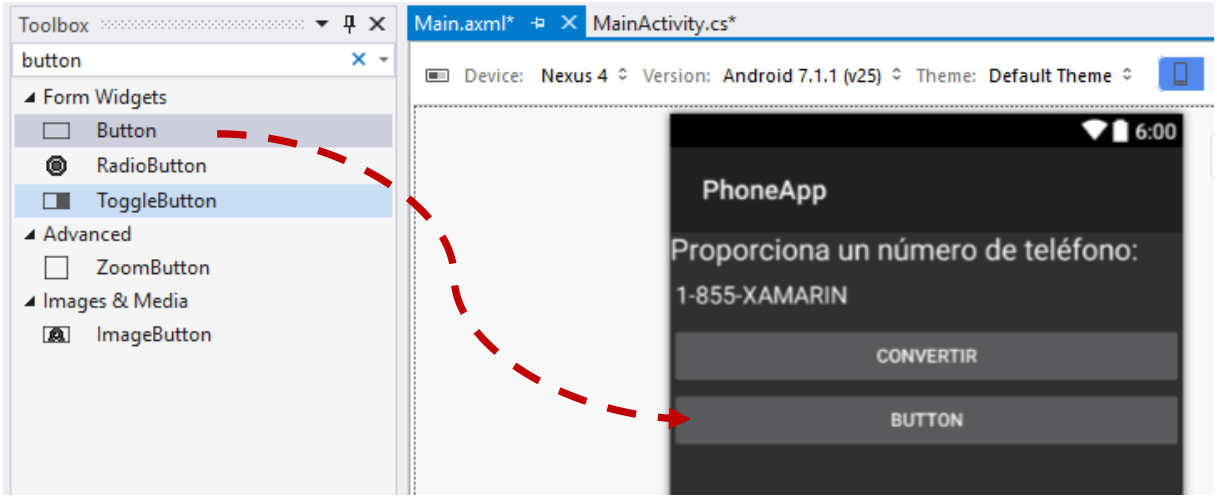
11. En la ventana de propiedades del widget **Button** localiza la propiedad **id** y modifica su valor por: **@+id/TranslateButton**.



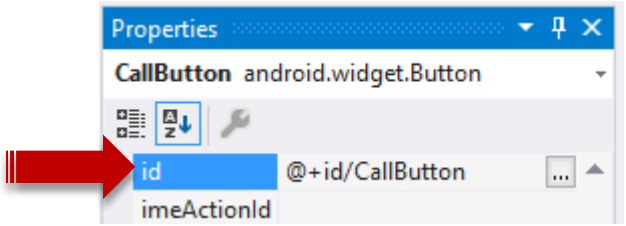
12. En la ventana de propiedades del widget **Button** localiza la propiedad **text** y modifica su valor por: **Convertir**.



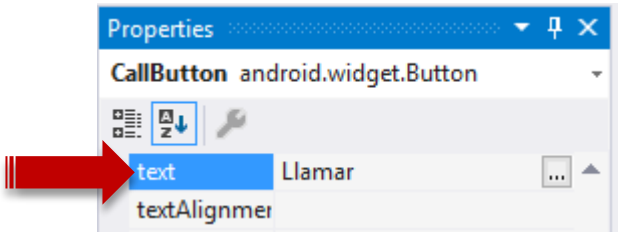
13. Desde la caja de herramientas, arrastra un segundo widget **Button** y suéltalo sobre la superficie de diseño para colocarlo debajo del widget **Button** anterior.



14. En la ventana de propiedades del widget **Button** que acabas de agregar localiza la propiedad **id** y modifica su valor por: **@+id/CallButton**.



15. En la ventana de propiedades del widget **Button** que acabas de agregar localiza la propiedad **text** y modifica su valor por: **Llamar**.

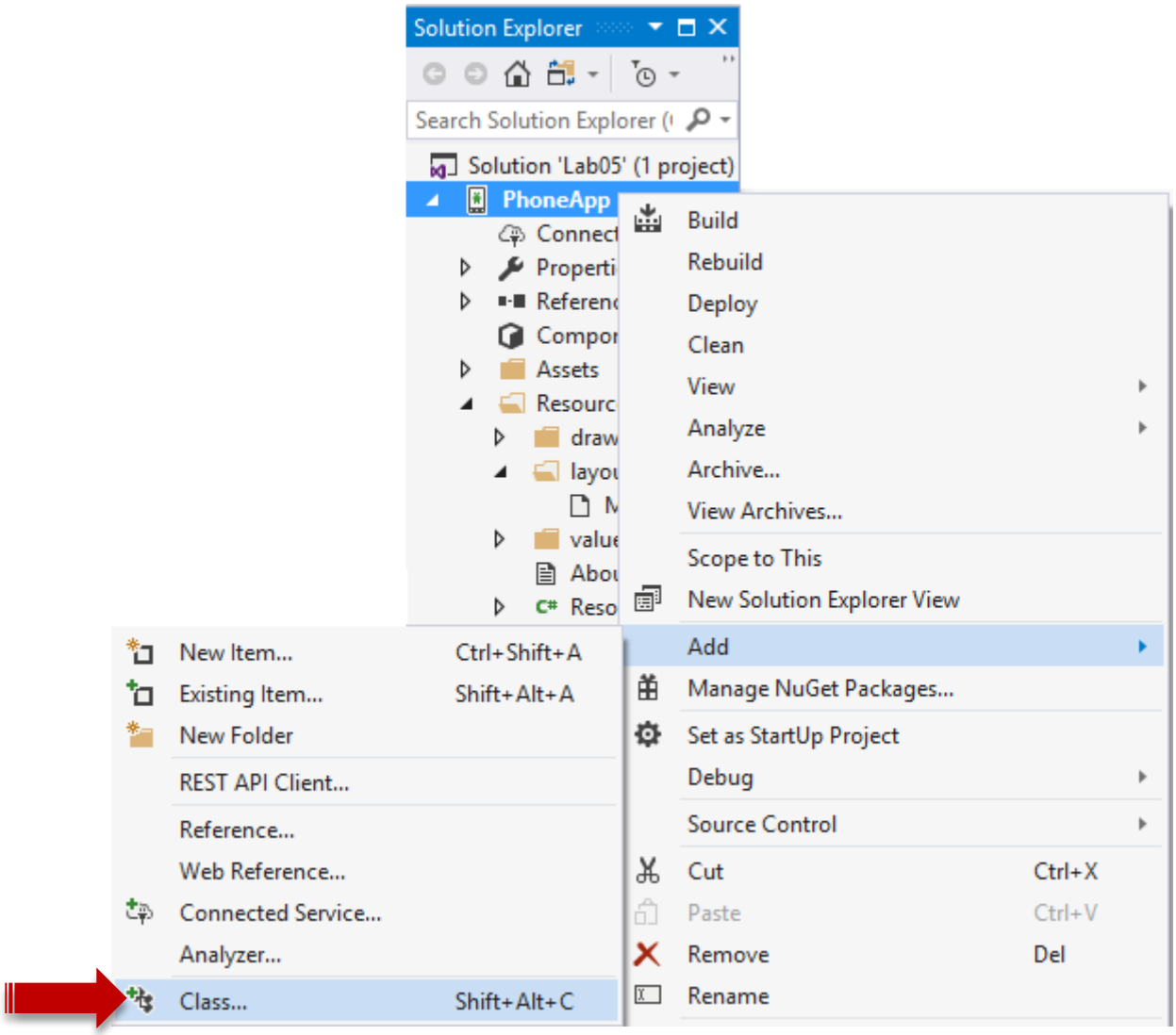


16. Presiona **CTRL-S** para guardar los cambios.

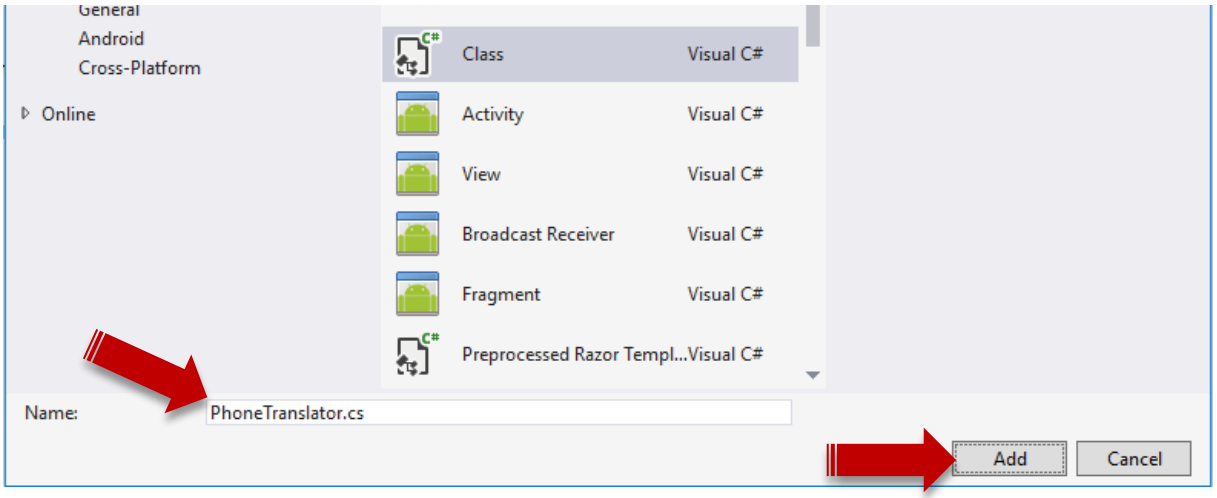
Tarea 3. Agregar la lógica de conversión.

En este momento la interfaz de usuario ha sido creada, el siguiente paso será agregar algo de código para traducir el número telefónico alfanumérico a su equivalente numérico.

1. Selecciona la opción **Add** > **New Class** del menú contextual del proyecto Android.



2. Asigna el nombre **PhoneTranslator.cs** al archivo y haz clic en **Add** para agregarlo al proyecto.



3. Modifica la clase **PhoneTranslator** para que sea una clase pública.

```
public class PhoneTranslator
{
}
```

4. Dentro de la clase **PhoneTranslator** agrega el siguiente código que declara e inicializa 2 variables de tipo **string**. Estas variables facilitarán la conversión de las letras de un número telefónico hacia su equivalente numérico.

```
string Letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
string Numbers = "22233344455566677778889999";
```

La declaración anterior representa la relación entre las letras de un teclado telefónico y el número que representan, por ejemplo, las letras A, B y C equivalen al número 2 mientras que las letras P, Q, R y S representan al número 7.

5. Agrega el siguiente código a la clase. Este código define un método que implementará la lógica para realizar la conversión. Tómate tu tiempo para entender la lógica de conversión.

```
public string ToNumber(string alfanumericPhoneNumber)
{
    var NumericPhoneNumber = new StringBuilder();
    if (!string.IsNullOrWhiteSpace(alfanumericPhoneNumber))
    {
        alfanumericPhoneNumber = alfanumericPhoneNumber.ToUpper();
        foreach (var c in alfanumericPhoneNumber)
        {
            if ("0123456789".Contains(c))
            {
                NumericPhoneNumber.Append(c);
            }
            else
            {
                var Index = Letters.IndexOf(c);
                if (Index >= 0)
                {
```

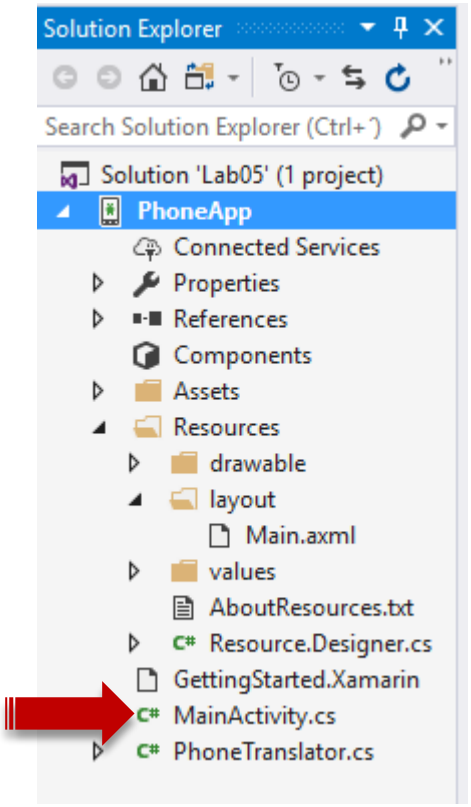
```
                    NumericPhoneNumber.Append(Numbers[Index]);
                }
            }
        }
    }
    return NumericPhoneNumber.ToString();
}
```

6. Guarda los cambios realizados.

Tarea 4. Agregar código para mostrar la interfaz de usuario.

El siguiente paso es agregar código para mostrar la interfaz de usuario. El código será agregado dentro de la clase **MainActivity**.

1. Abre el archivo **MainActivity.cs**.



2. Quita el comentario a la línea de código que establece a **Resource.Layout.Main** como el recurso que define la interfaz de usuario.

   El código del método **OnCreate** será similar al siguiente.

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    // Set our view from the "main" layout resource
    SetContentView (Resource.Layout.Main);
}
```

3. Agrega el siguiente código justo después de la llamada al método **SetContentView**. Este código obtiene una referencia a los controles que fueron creados en el archivo del diseño de la interfaz de usuario **Main.axml**.

```
var PhoneNumberText = FindViewById<EditText>(Resource.Id.PhoneNumberText);
var TranslateButton = FindViewById<Button>(Resource.Id.TranslateButton);
var CallButton = FindViewById<Button>(Resource.Id.CallButton);
```

4. Agrega el siguiente código para deshabilitar el botón "**Llamar**".

```
CallButton.Enabled = false;
```

5. Agrega el siguiente código para declarar una variable que almacenará el número telefónico convertido a solo números.

```
var TranslatedNumber = string.Empty;
```

6. Agrega el siguiente código que será ejecutado cuando el usuario presione el botón "**Convertir**".

```
TranslateButton.Click += (object sender, System.EventArgs e) =>
  {
      var Translator = new PhoneTranslator();
      TranslatedNumber = Translator.ToNumber(PhoneNumberText.Text);
      if (string.IsNullOrWhiteSpace(TranslatedNumber))
      {
          // No hay número a llamar
          CallButton.Text = "Llamar";
          CallButton.Enabled = false;
      }
      else
      {
          // Hay un posible número telefónico a llamar
          CallButton.Text = $"Llamar al {TranslatedNumber}";
          CallButton.Enabled = true;
      }
  };
```

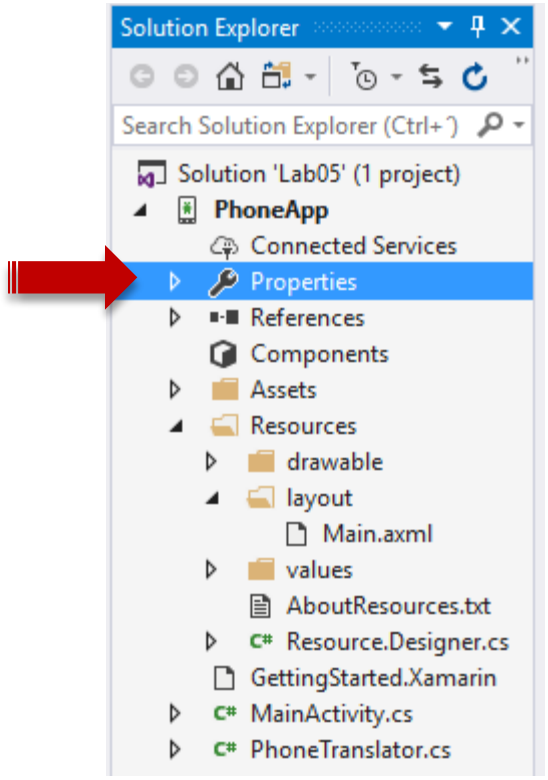
7. Agrega ahora el siguiente código que será ejecutado cuando el usuario presione el botón "**Llamar**".

```
CallButton.Click += (object sender, System.EventArgs e) =>
 {
     // Intentar marcar el número telefónico
     var CallDialog = new AlertDialog.Builder(this);
     CallDialog.SetMessage($"Llamar al número {TranslatedNumber}?");
     CallDialog.SetNeutralButton("Llamar", delegate
     {
         // Crear un intento para marcar el número telefónico
         var CallIntent =
            new Android.Content.Intent(Android.Content.Intent.ActionCall);
         CallIntent.SetData(
             Android.Net.Uri.Parse($"tel:{TranslatedNumber}"));
         StartActivity(CallIntent);
     });
     CallDialog.SetNegativeButton("Cancelar", delegate { });
     // Mostrar el cuadro de diálogo al usuario y esperar una respuesta.
     CallDialog.Show();
 };
```

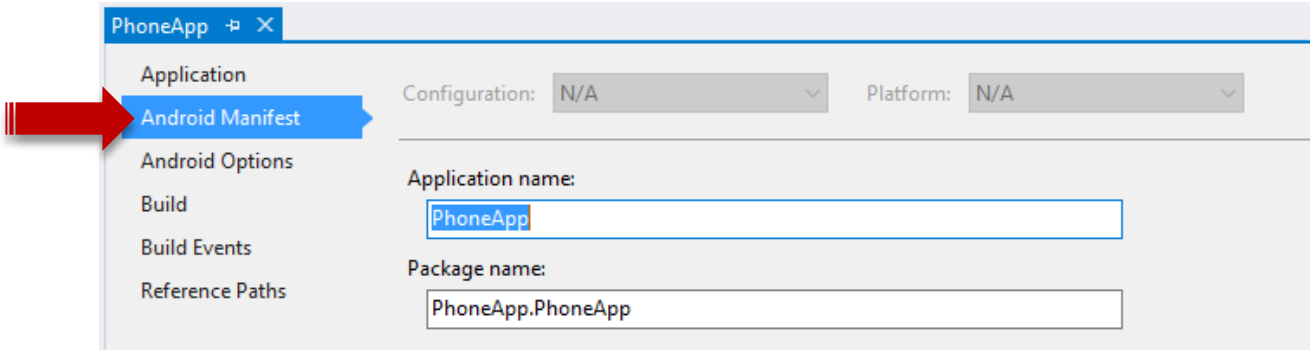
Tarea 5. Asignar el permiso de realizar llamadas telefónicas.

Finalmente, es tiempo de asignar a la aplicación el permiso para realizar llamadas telefónicas. Los permisos de la aplicación pueden ser editados en el manifiesto Android.

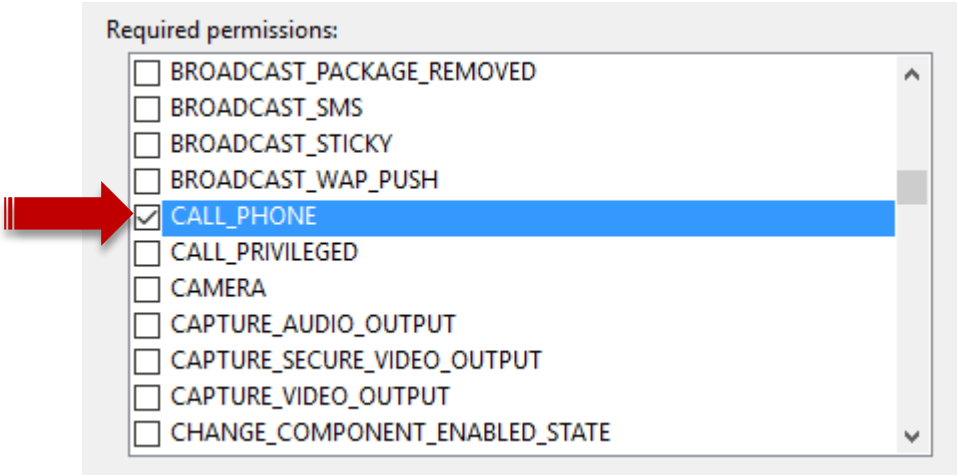
1. Haz doble clic en el nodo **Properties** del proyecto Android para abrir el panel de propiedades del proyecto.



2. En la ventana de propiedades del proyecto Android selecciona **Android Manifest**.

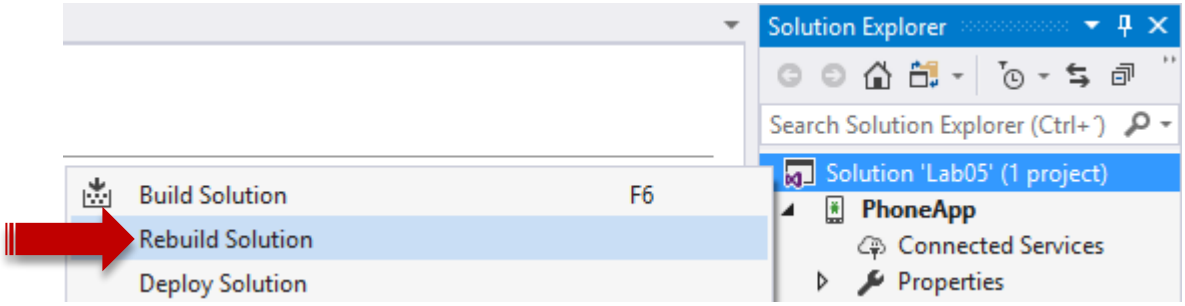


3. En la sección **Required Permissions** habilita el permiso **CALL_PHONE**.

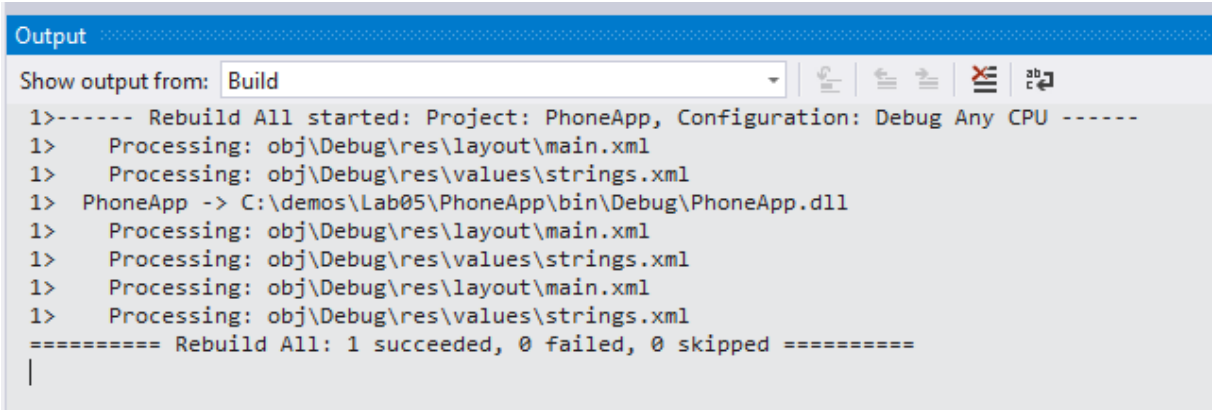


4. Guarda todo tu trabajo presionando **CTRL-SHIFT-S** o seleccionando la opción **File** > **Save All** de la barra de menús de Visual Studio.

5. Selecciona la opción **Rebuild Solution** del menú contextual de la solución.



6. Verifica que todo compile correctamente.



Si hay errores, sigue los pasos anteriores y corrige cualquier error hasta que la aplicación sea generada correctamente. Si obtienes un error de generación como, "***Resource does not exist in the current context***", verifica que el espacio de nombres de **MainActivity.cs** coincida con el nombre de proyecto y vuelve a generar completamente la solución. Si sigues teniendo errores de compilación, comprueba que hayas instalado las últimas actualizaciones de Xamarin.Android.

Tarea 6. Agregar el toque final a la aplicación.

En este momento la aplicación ya debe trabajar correctamente. Es tiempo de agregar los toques finales.

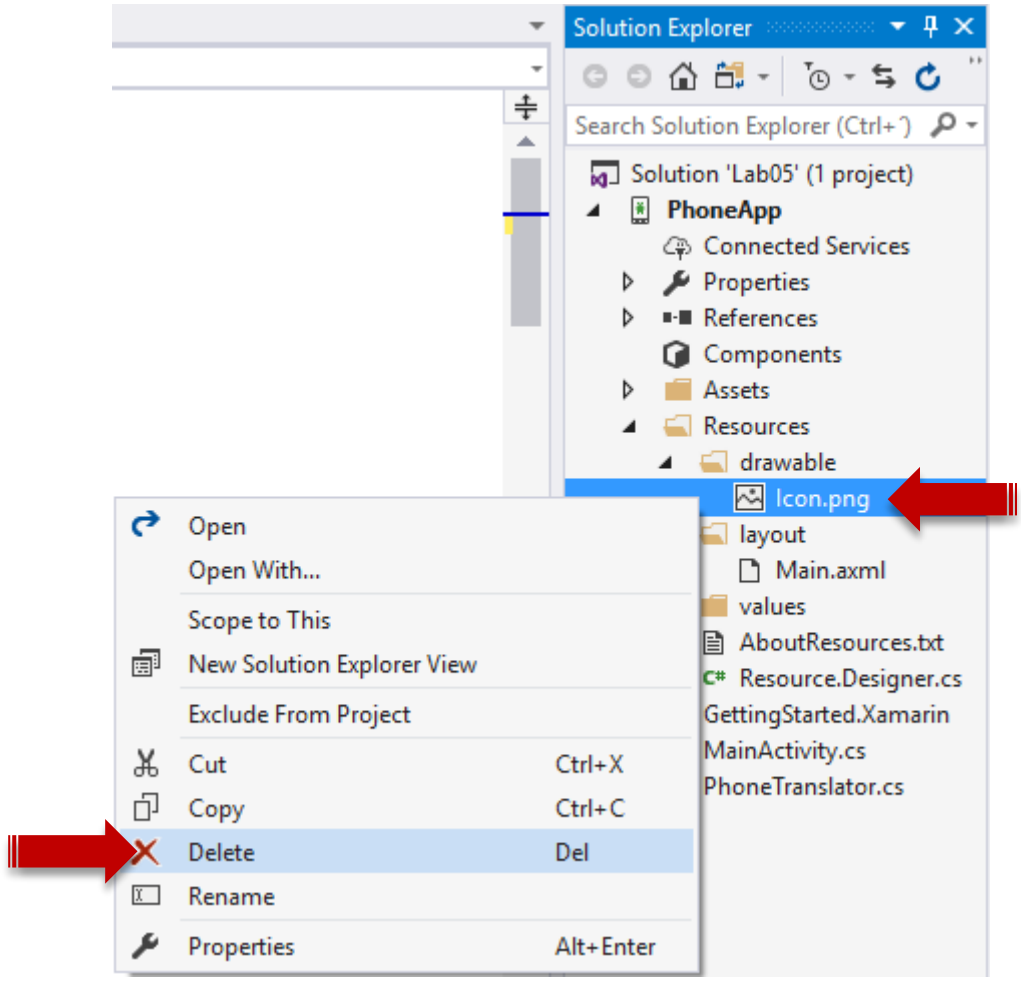
1. Abre el archivo **MainActivity.cs**.

2. Edita el valor del parámetro **Label** para asignarle el valor "**Phone App**". El valor del parámetro **Label** es lo que Android muestra en la parte superior de la pantalla para permitir a los usuarios conocer la parte de la aplicación en la que se encuentran.

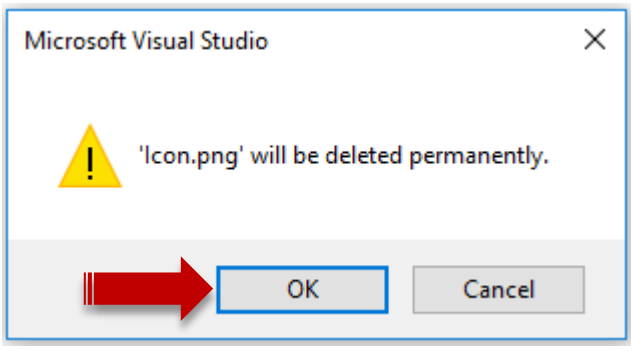
```
[Activity(Label = "Phone App", MainLauncher = true, Icon = "@drawable/icon")]
public class MainActivity : Activity
```

   El siguiente paso será establecer el icono de la aplicación.

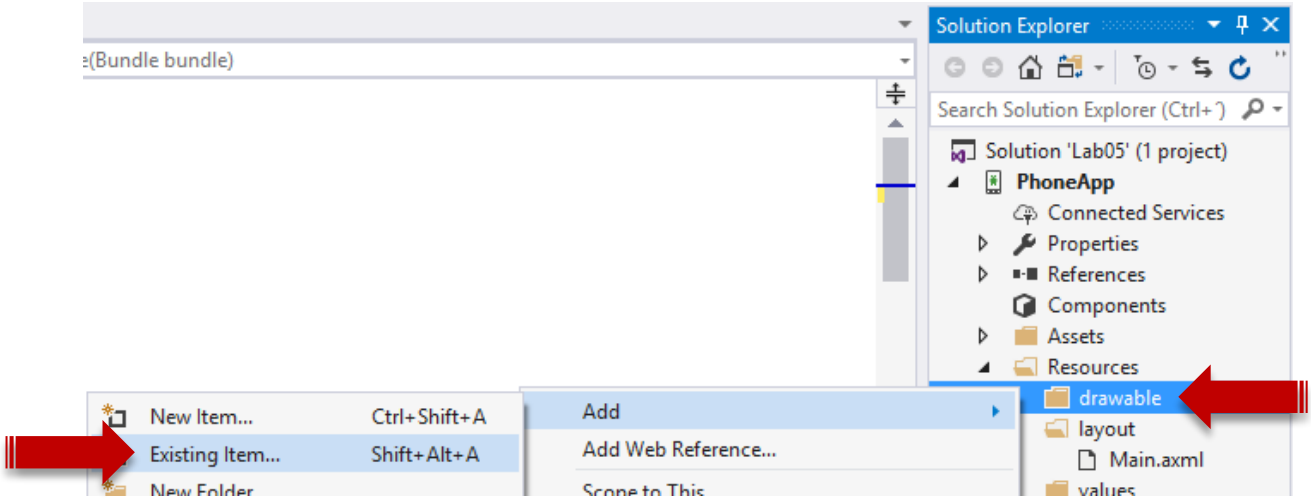
3. Elimina el archivo **Icon.png** actual ubicado en el folder **Resources\drawable**.



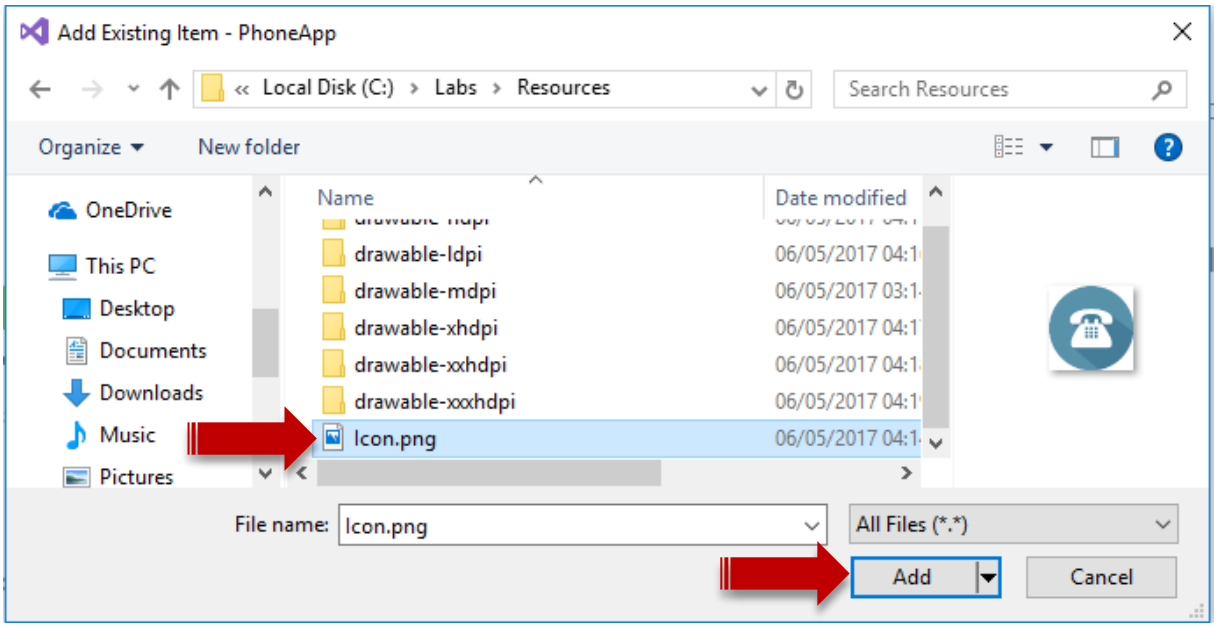
4. Acepta la eliminación del archivo.



5. Selecciona la opción **Add** > **Existing Item…** del menú contextual del folder **drawable**.



6. En la ventana de dialogo **Add Existing Item…** selecciona el archivo **Icon.png** que acompaña a este documento y haz clic en **Add** para agregarlo al proyecto.

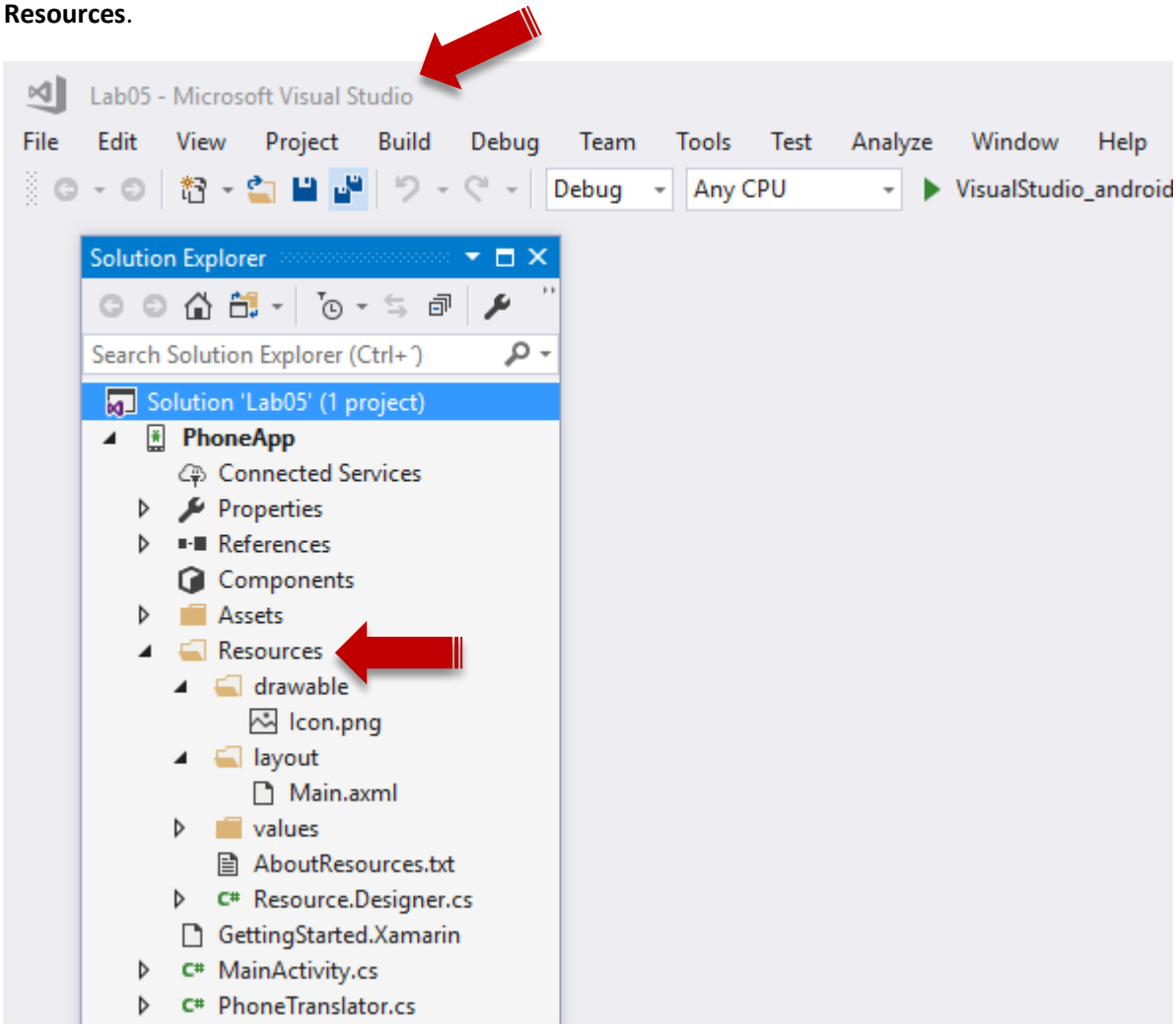


El siguiente paso es agregar todos los folders **drawable-*** que acompañan a este documento hacia el folder **Resources** del proyecto. Estos folders proporcionan diferentes resoluciones del icono para que se pueda mostrar correctamente en diferentes dispositivos con diferentes densidades de pantalla.

7. Cierra Visual Studio.

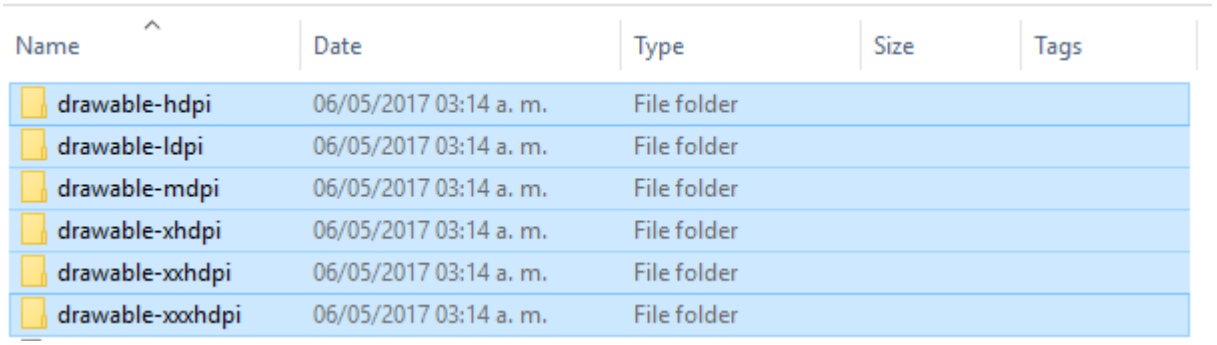
8. Abre Visual Studio en modo usuario (no bajo el contexto de Administrador). Eso te permitirá arrastrar las carpetas desde el Explorador de archivos de Windows hacia la ventana **Solution Explorer** de Visual Studio.

9. Abre la solución y el explorador de soluciones de Visual Studio para mostrar el folder **Resources**.

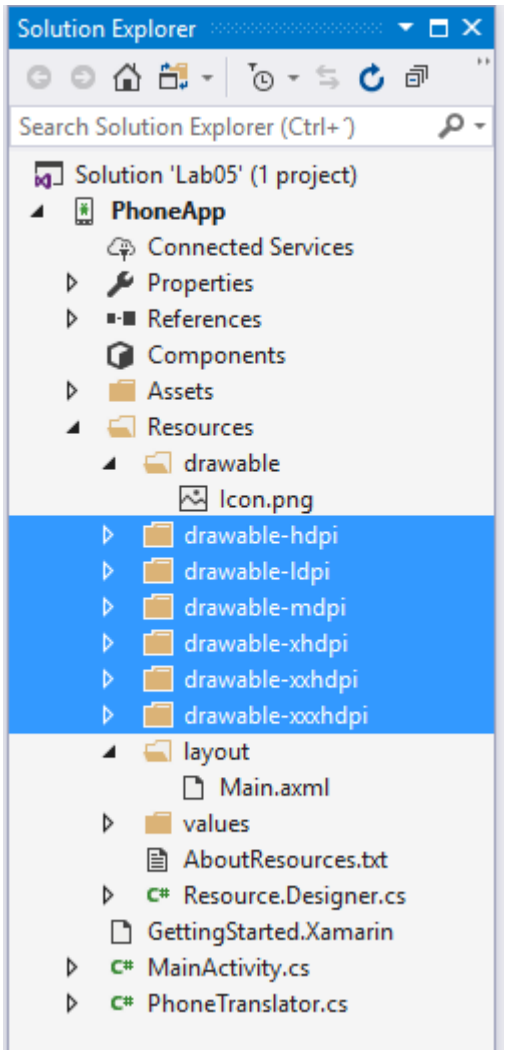


10. Abre una instancia del Explorador de archivos de Windows.

11. Navega a la carpeta donde se encuentran los folders **drawable-*** y selecciónalos.



12. Arrastra las carpetas hacia el folder **Resources** del proyecto Android en Visual Studio.



Ahora los folders forman parte del proyecto.

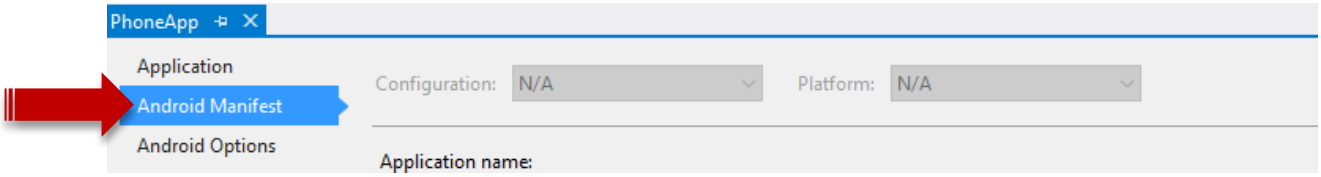
13. Guarda los cambios.

14. Cierra Visual Studio y ábrelo nuevamente bajo el contexto de Administrador.

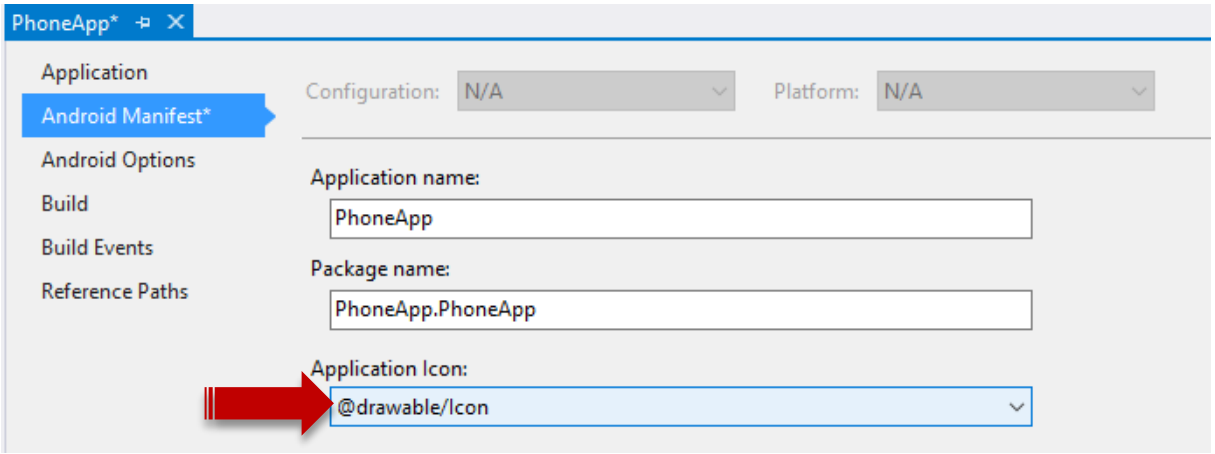
15. Abre la solución.

16. Abre la página de propiedades del proyecto Android.

17. Selecciona la opción **Android Manifest**.



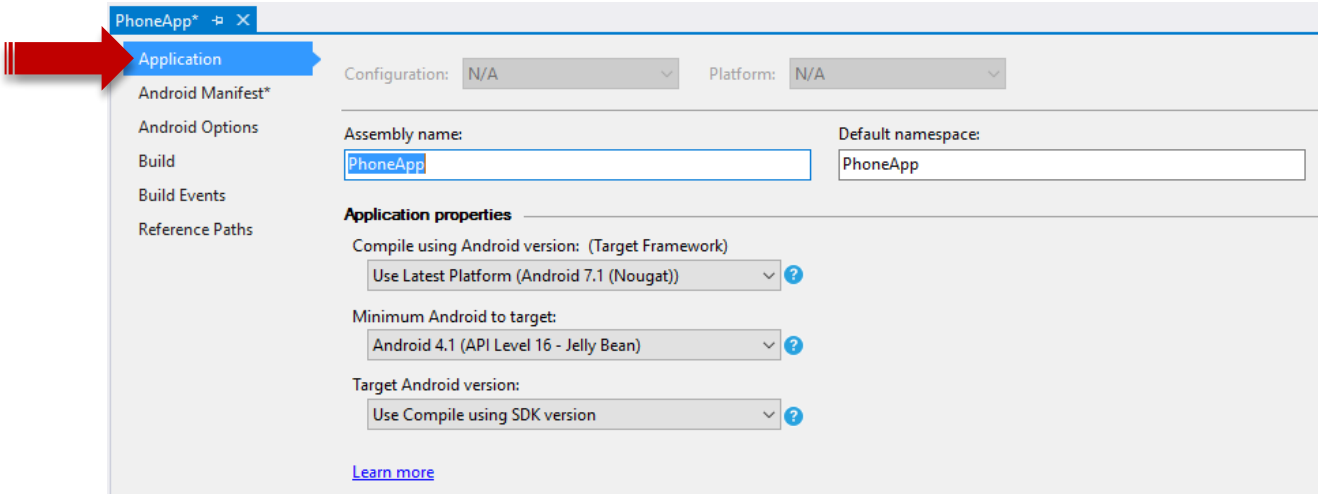
18. En el cuadro de lista desplegable **Application Icon**, selecciona **@drawable/Icon** para especificar el icono de la aplicación.



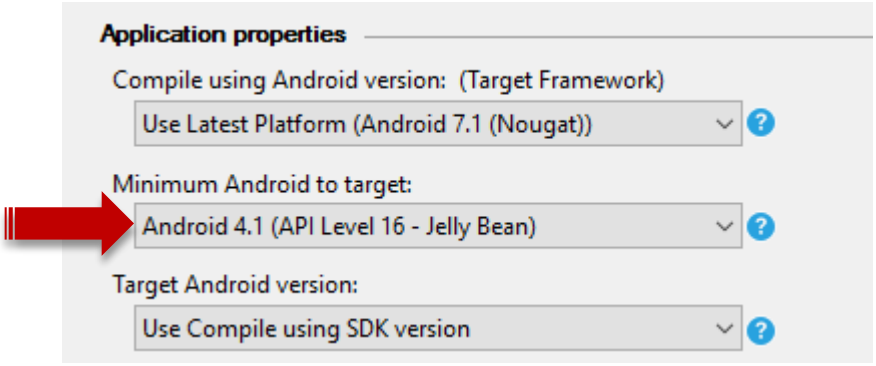
Tarea 7. Probar la aplicación.

En este momento ya puedes probar la aplicación desplegándola hacia un emulador. Sin embargo, antes de que envíes tu aplicación hacia un emulador o dispositivo, debes configurar la versión Android mínima de la aplicación para que coincida con la versión Android del emulador o dispositivo de prueba.

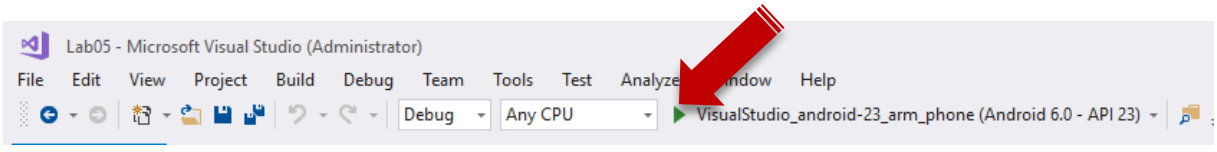
1. Selecciona la opción **Application** de la ventana de propiedades del proyecto Android.



2. Selecciona el nivel de API en el cuadro de lista desplegable ***Minimum Android to target***. En la siguiente imagen el nivel mínimo de API es establecido a **Android 4.1 (API Level 16 – Jelly Bean)**.



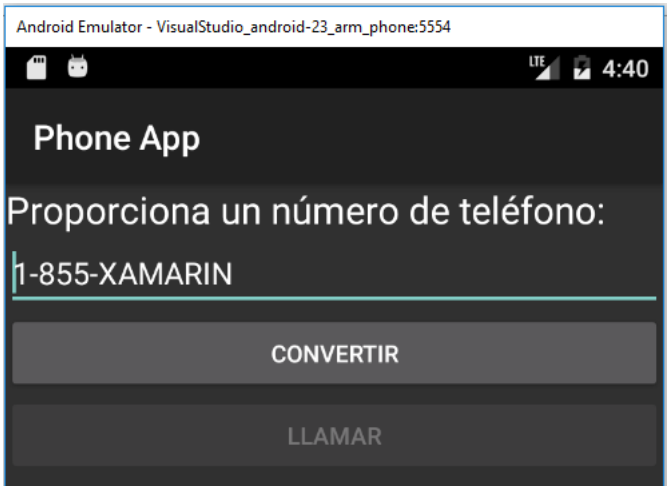
3. Selecciona el emulador a utilizar y haz clic en la flecha verde "**play**" para desplegar y ejecutar la aplicación en el emulador seleccionado.



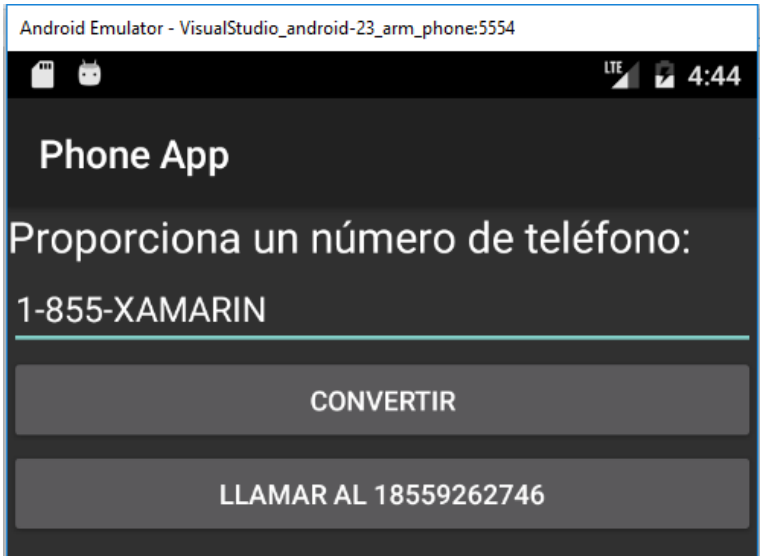
En caso de que no existan emuladores en la lista, debes crear y configurar uno mediante **Android AVD Manager**. Puedes consultar las indicaciones en el siguiente enlace:

Android SDK Emulator
https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/debug-on-emulator/android-sdk-emulator/

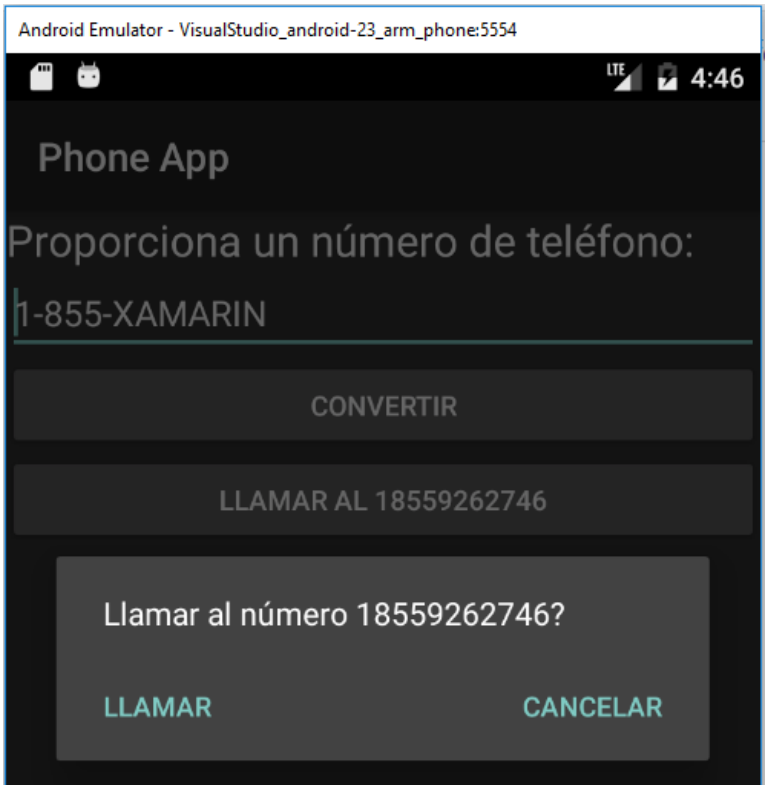
Visual Studio copiará los archivos hacia el emulador antes de instalar y lanzar la aplicación. La siguiente imagen muestra la aplicación ejecutándose en un emulador Android.



4. Haz clic en el botón **Convertir**. El texto del botón "**Llamar**" será actualizado.



5. Haz clic en el botón "**Llamar**". El cuadro de diálogo para realizar la llamada será mostrado.



6. Regresa a Visual Studio y detén la aplicación.

Ejercicio 2: Validando tu actividad

En este ejercicio agregarás funcionalidad a tu laboratorio con el único propósito de enviar una evidencia de la realización del mismo.

La funcionalidad que agregarás consumirá un ensamblado que representa una Capa de acceso a servicio (SAL) que será consumida por tu aplicación Android.

Es importante que realices cada laboratorio del diplomado ya que esto te dará derecho a obtener el diploma final del mismo.

Tarea 1. Agregar los componentes de la Capa de acceso a Servicio.

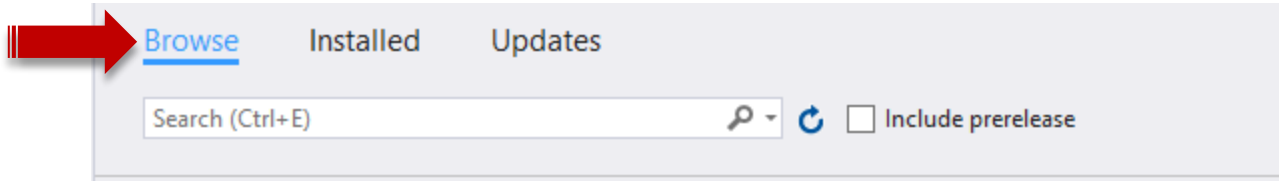
En esta tarea agregarás una referencia al ensamblado **SALLab05.dll** que implementa la capa de acceso a servicio. El archivo **SALLab05.dll** se encuentra disponible junto con este documento.

1. Accede a la opción **Add** > **Reference…** del menú contextual del proyecto Android para agregar la referencia del ensamblado **SALLab05.dll**.

   Este componente realiza una conexión a un servicio de Azure Mobile, por lo tanto, será necesario agregar el paquete NuGet **Microsoft.Azure.Mobile.Client**.

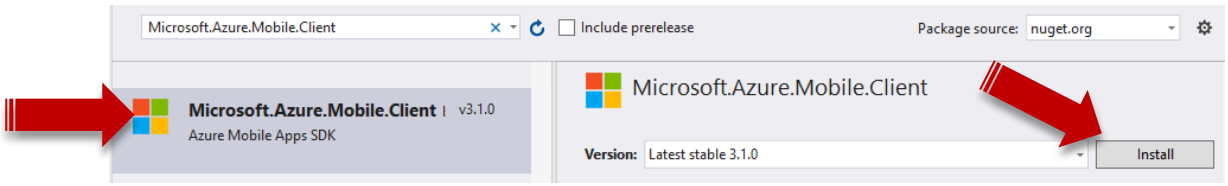
2. Accede a la opción **Manage NuGet Packages…** del menú contextual de la aplicación Android.

3. En la ventana **NuGet** haz clic en **Browse**.

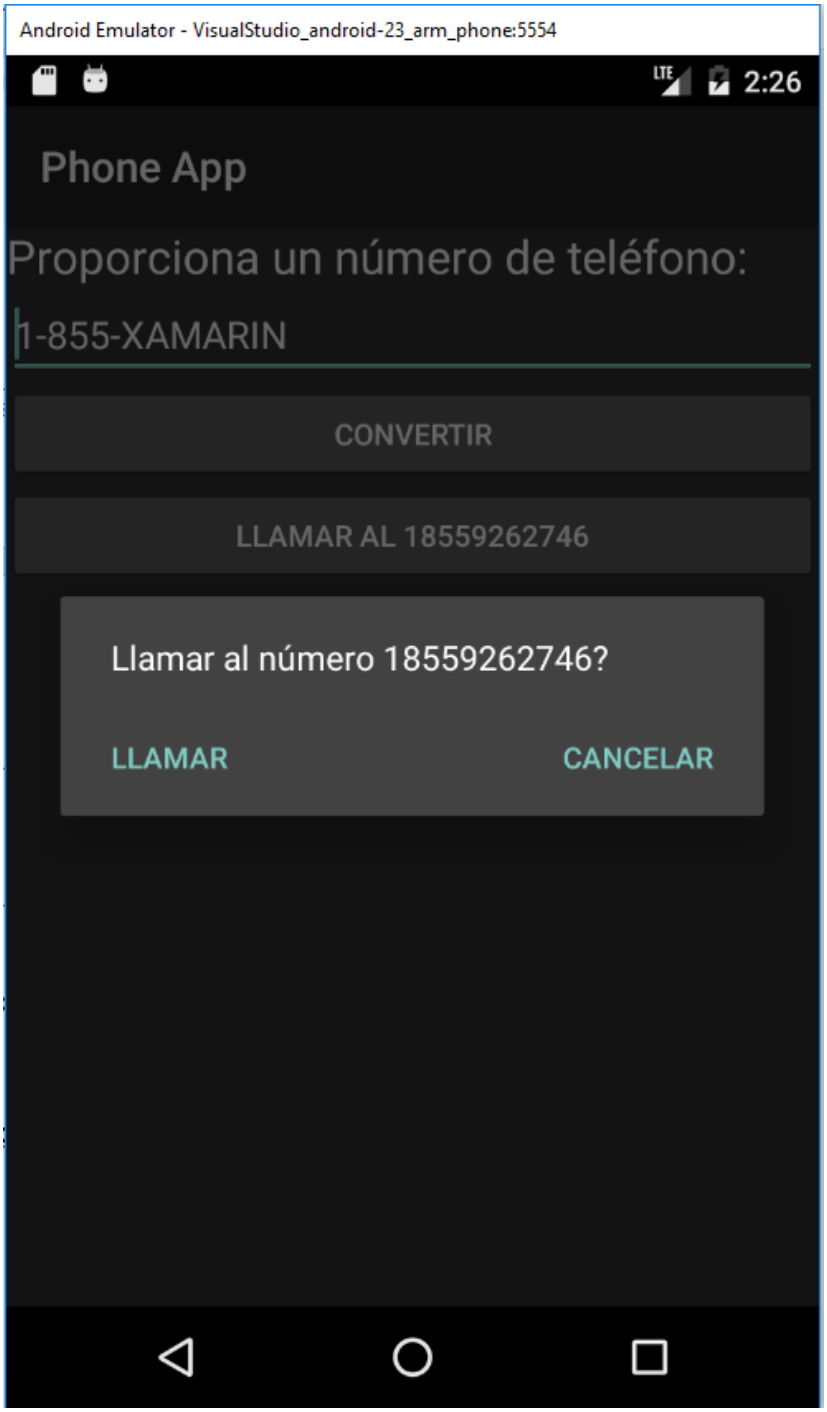


4. En el cuadro de búsqueda escribe **Microsoft.Azure.Mobile.Client.** La lista de resultados se actualizará automaticamente conforme vas escribiendo.

5. En la ventana de resultados haz clic en **Microsoft.Azure.Mobile.Client** y haz clic en **Install** para instalar el paquete. Acepta la instalación de los paquetes adicionales cuando te sea requerido.



6. Verifica que tu aplicación se ejecute correctamente.



7. Regresa a Visual Studio y detén la ejecución.

Por el momento aquí finaliza este laboratorio, en una siguiente actividad agregarás funcionalidad a la aplicación para enviar tu evidencia de realización de este laboratorio.

Resumen

¡Felicidades! Haz completado tu primera aplicación Xamarin.Android. 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.***