Laboratorio

¡Hola, iOS!

Versión: 1.0.0
Agosto de 2017

Miguel Muñoz Serafín
@msmdotnet

CONTENIDO

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 iOS.
- Diseñar una interfaz de usuario sencilla de una aplicación iOS.
- Agregar archivos de código a un proyecto iOS.
- Personalizar el título e icono de una aplicación iOS.
- Desplegar la aplicación hacia un emulador iOS.

Requisitos

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

- Un equipo de desarrollo con Visual Studio 2017. Los pasos descritos en este laboratorio fueron diseñados con Visual Studio Enterprise 2017 sobre una máquina con Windows 10 Pro.
- Xamarin.iOS para Visual Studio 2017.
- Un equipo Mac con sistema operativo Sierra 10.12 o posteriores.
- La última versión de Xcode. Al momento de elaborar este documento la versión actual de Xcode es la versión 8.3.3.
- La última versión de Xamarin.iOS para Mac.

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

Ejercicio 1: Creando una aplicación iOS

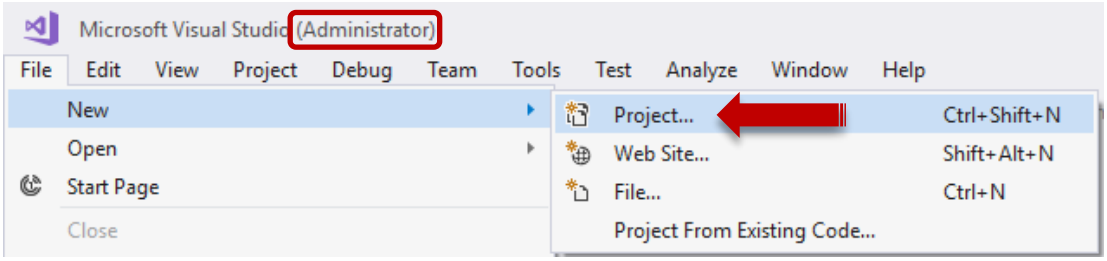
En este ejercicio crearás una aplicación iOS que te permitirá introducirte en las herramientas, conceptos y pasos para crear y desplegar aplicaciones Xamarin.iOS.

Tarea 1. Crear un proyecto iOS.

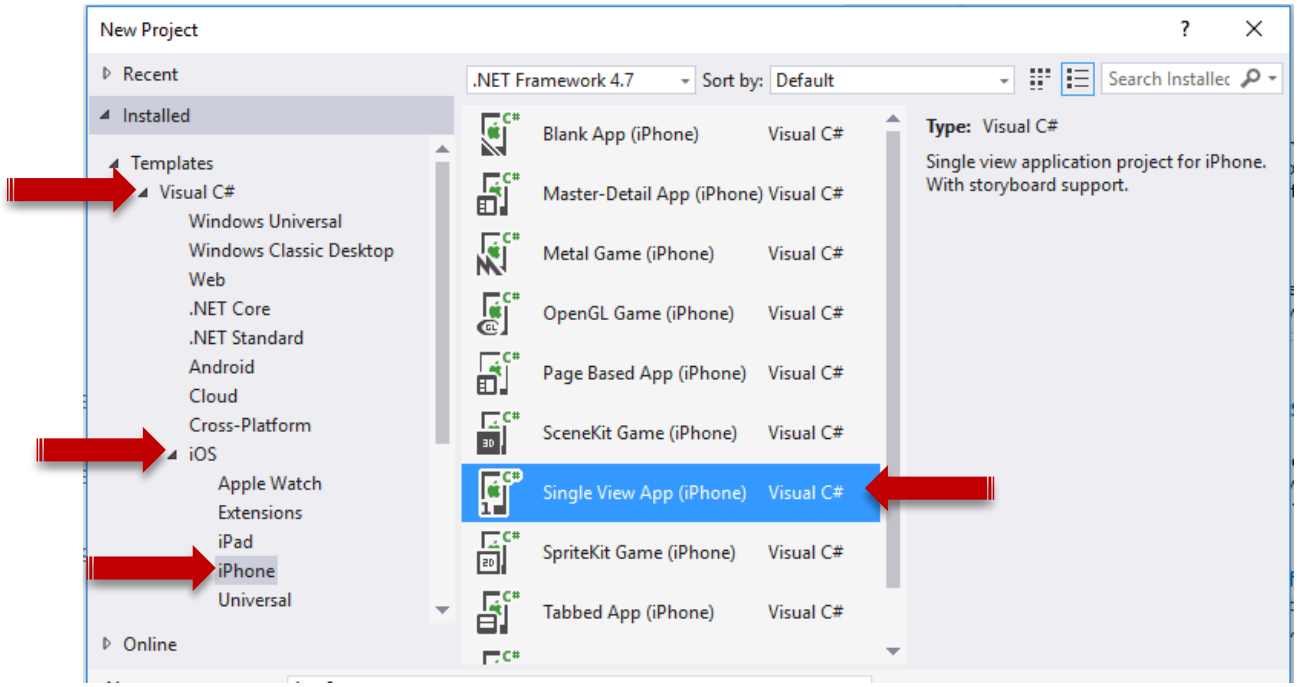
Realiza los siguientes pasos para crear un proyecto Xamarin.iOS.

1. Abre Visual Studio en el contexto del Administrador.

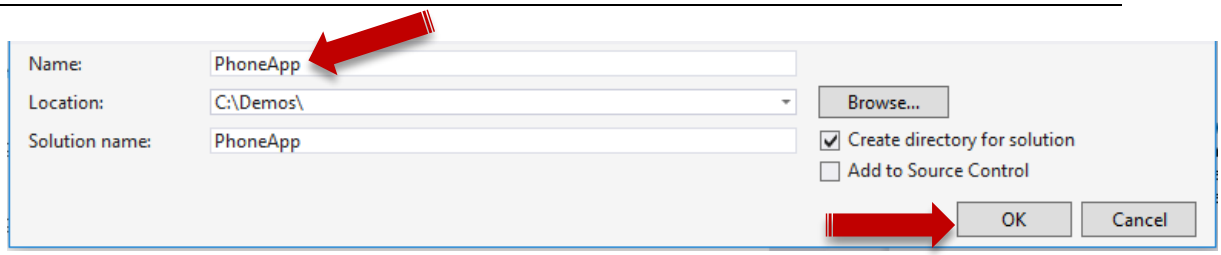
2. Selecciona la opción ***File*** > ***New*** > ***Project***.



3. En la ventana ***New Project*** selecciona la plantilla ***Single View App (iPhone)*** para crear una nueva solución Xamarin.iOS.



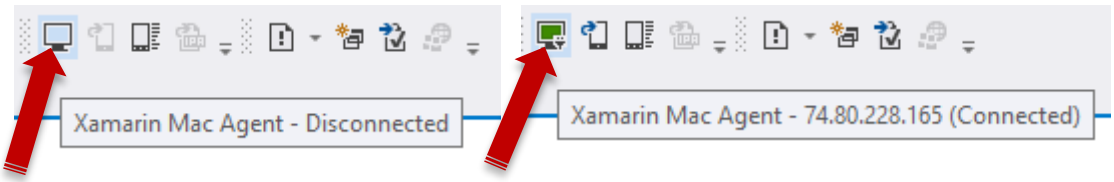
4. Asigna un nombre al proyecto iOS y haz clic en ***OK*** para crear la solución.



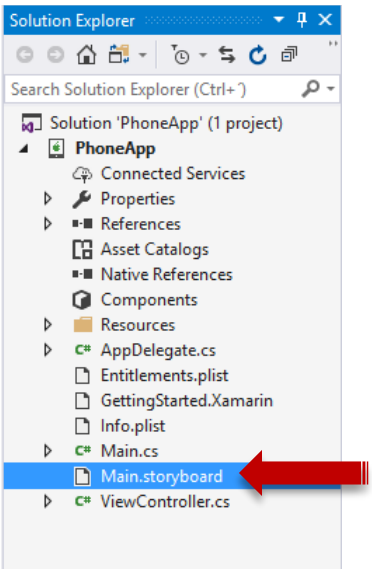
Tarea 2. Diseñar la interfaz de usuario de la aplicación.

Una vez creado el proyecto iOS, empezaremos por diseñar la interfaz de usuario de la aplicación.

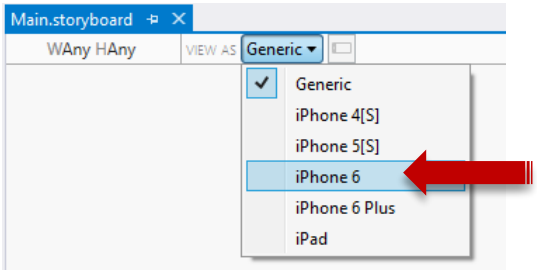
1. Confirma que el icono ***Xamarin Mac Agent*** se encuentre en color verde (conectado). De no ser así, haz clic en él para realizar la conexión a la Mac.



2. Haz doble clic sobre el archivo ***Main.storyboard*** para abrirlo en el diseñador de iOS.

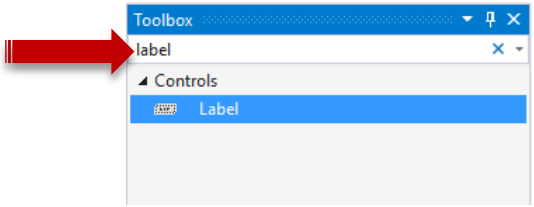


3. En la barra de herramientas del diseñador haz clic en ***View As*** y selecciona la opción ***iPhone 6***.

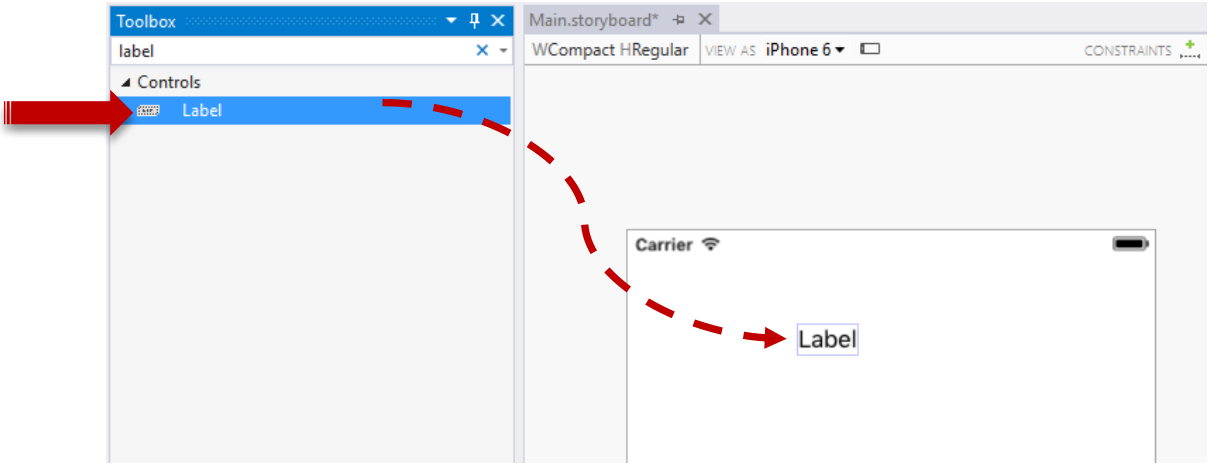


4. Selecciona la opción ***View*** > ***Toolbox*** de la barra de menús de Visual Studio para abrir la caja de herramientas.

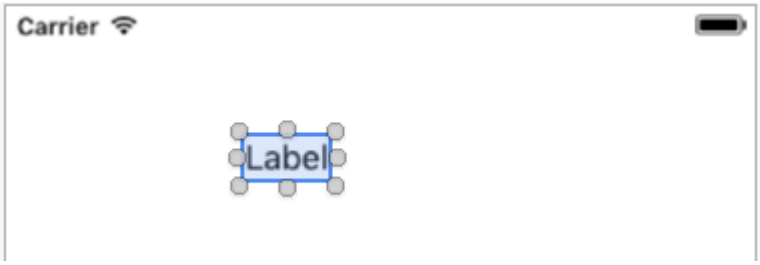
5. En el campo de búsqueda de la caja de herramientas, escribe ***label***.



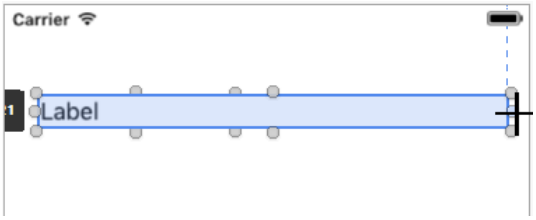
6. Arrastra el elemento ***Label*** y suéltalo sobre cualquier parte de la superficie de diseño.



7. Haz clic sobre el elemento ***Label*** para mostrar sus ***Dragging Controls*** (los círculos alrededor del control).

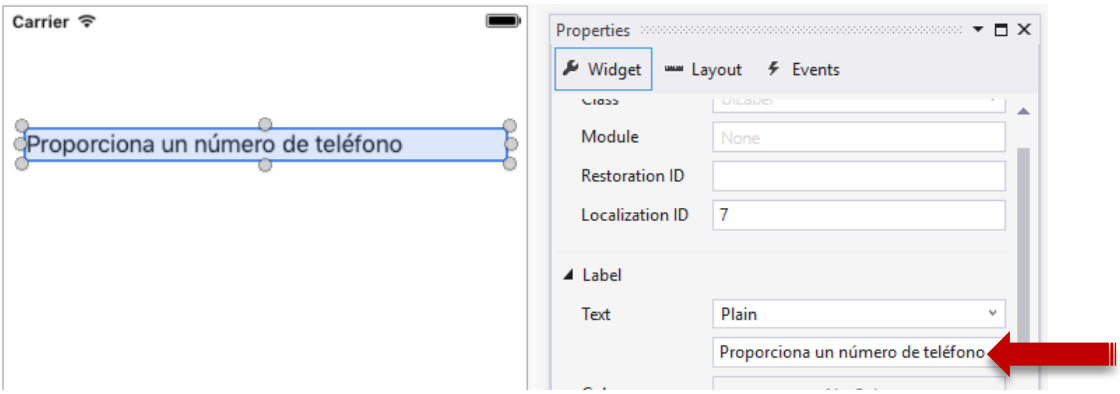


8. Utiliza los ***Dragging Controls*** para arrastrarlos y hacer más ancho el control ***Label***.

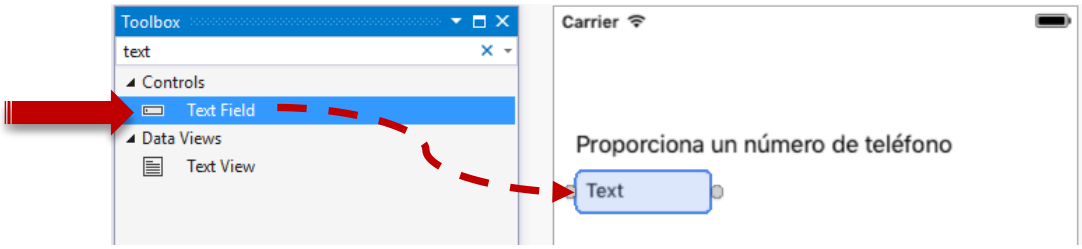


9. Con el ***Label*** seleccionado, presiona ***F4*** para abrir la ventana de propiedades.

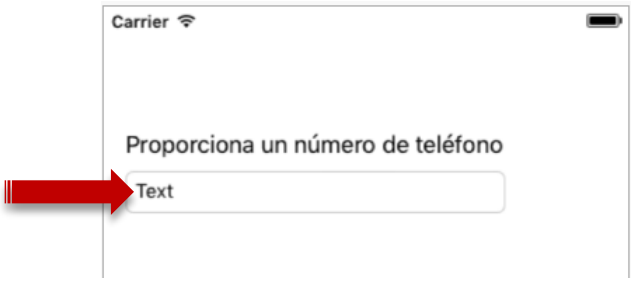
10. En la ventana de propiedades del ***Label***, modifica el valor de la propiedad ***Text*** por lo siguiente: ***Proporciona un número de teléfono***.



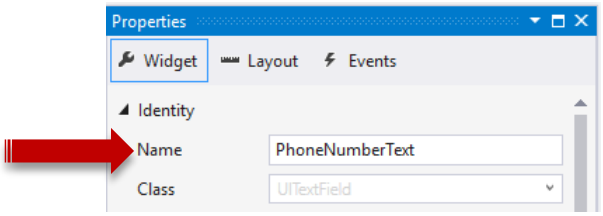
11. Desde la caja de herramientas, arrastra un elemento ***Text Field*** y suéltalo sobre la superficie de diseño para colocarlo debajo del ***Label***.



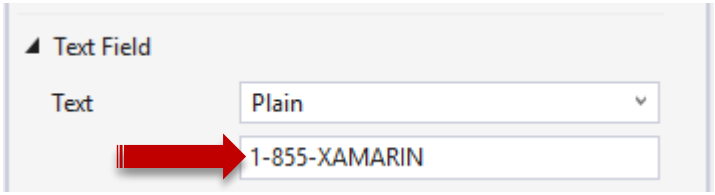
12. Ajusta el ancho del ***Text Field*** para que sea igual al ancho del ***Label***.



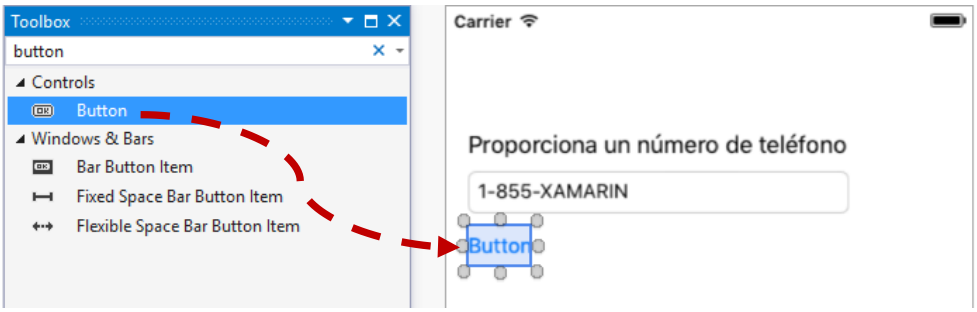
13. En la ventana de propiedades del ***Text Field*** localiza la propiedad ***Name*** y modifica su valor por ***PhoneNumberText***.



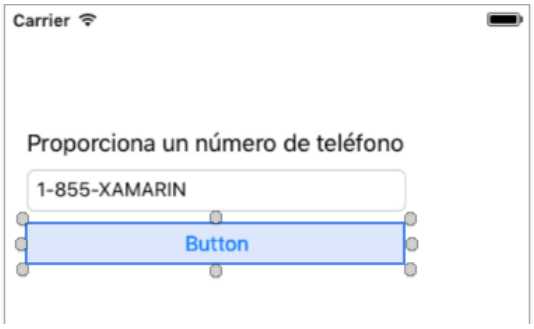
14. En la ventana de propiedades del ***Text Field*** localiza la propiedad ***Text*** y modifica su valor por ***1-855-XAMARIN***.



15. Desde la caja de herramientas, arrastra un elemento ***Button*** y suéltalo sobre la superficie de diseño para colocarlo debajo del ***Text Field***.

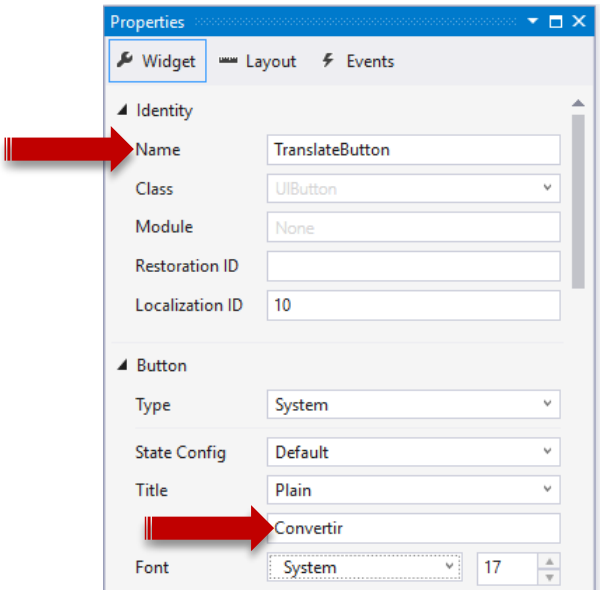


16. Ajusta el ancho del ***Button*** para que sea igual al ancho del ***Text Field***.

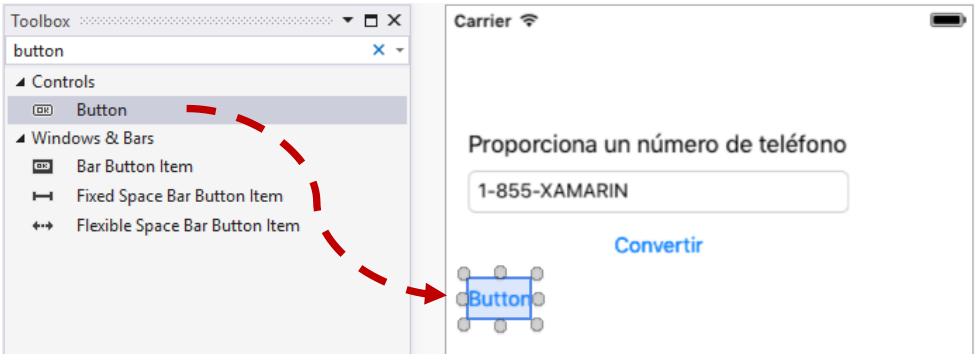


17. En la ventana de propiedades del ***Button*** localiza la propiedad ***Name*** y modifica su valor por ***TranslateButton***

18. En la ventana de propiedades del ***Button*** localiza la propiedad ***Title*** y modifica su valor por ***Convertir.***



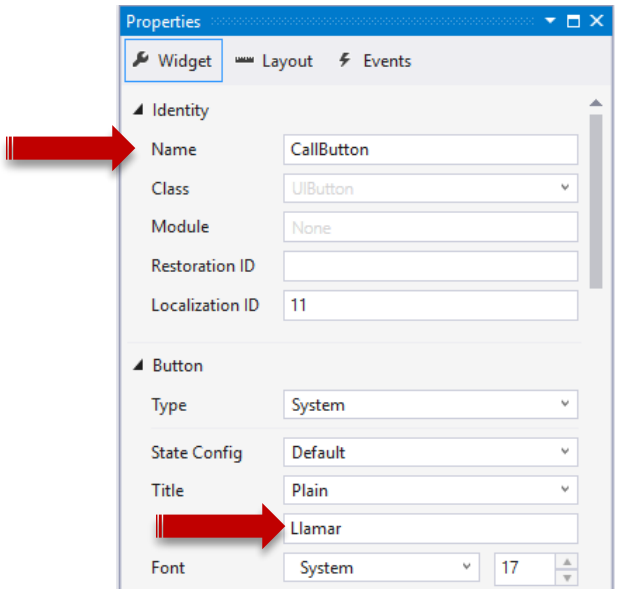
19. Desde la caja de herramientas, arrastra un segundo ***Button*** y suéltalo sobre la superficie de diseño para colocarlo debajo del ***Button*** anterior.



20. Ajusta el ancho del nuevo ***Button*** para que sea igual al ancho del ***Button*** anterior.

21. En la ventana de propiedades del ***Button*** que acabas de agregar localiza la propiedad ***Name*** y modifica su valor por ***CallButton***.

22. En la ventana de propiedades del ***CallButton*** localiza la propiedad ***Title*** y modifica su valor por ***Llamar***.



23. 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 iOS.

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.

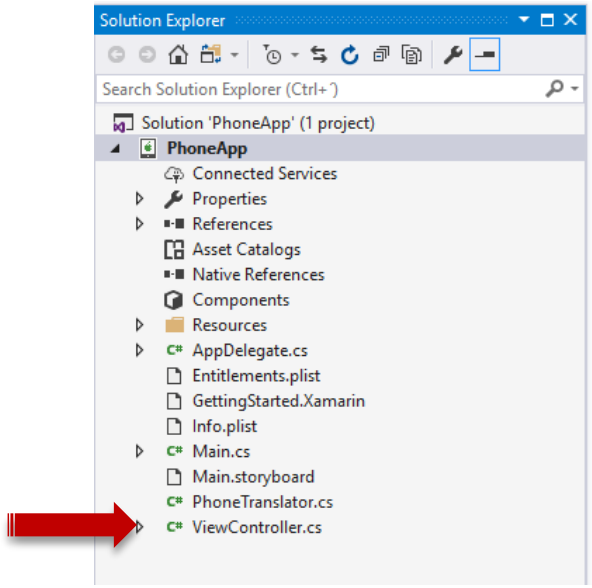
```
public string ToNumber(string alfanumericPhoneNumber)
{
    var NumericPhoneNumber = new StringBuilder();
    if (!string.IsNullOrWhiteSpace(alfanumericPhoneNumber))
    {
        alfanumericPhoneNumber = alfanumericPhoneNumber.ToUpper();
        foreach (var c in alfanumericPhoneNumber)
        {
            if ("0123456789".IndexOf(c) >= 0)
            {
                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 ***ViewController***.

1. Haz doble clic sobre el archivo ***ViewController.cs*** para abrirlo en el editor de código**.**



2. Dentro de la clase ***ViewController***, localiza el método ***ViewDidLoad*** y agrega el siguiente código debajo de la instrucción ***base.ViewDisLoad()***.

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

Este código declara una variable que almacenará el número telefónico convertido a solo números.

3. Debajo de la instrucción anterior, agrega el siguiente código que será ejecutado cuando el usuario toque el botón "***Convertir***".

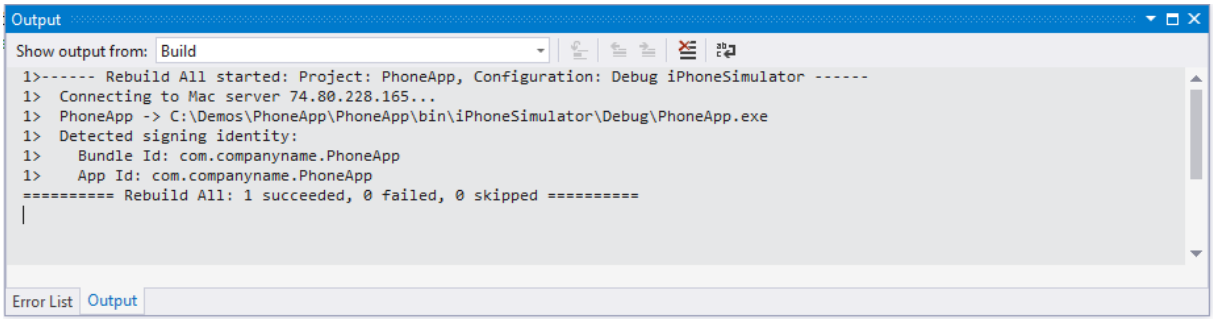
```
TranslateButton.TouchUpInside += (object sender, EventArgs e) =>
{
    var Translator = new PhoneTranslator();
    TranslatedNumber = Translator.ToNumber(PhoneNumberText.Text);
    if(string.IsNullOrWhiteSpace(TranslatedNumber))
    {
        // No hay número a llamar
        CallButton.SetTitle("Llamar",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

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

```
CallButton.TouchUpInside += (object sender, EventArgs e) =>
{
    var URL = new Foundation.NSUrl($"tel:{TranslatedNumber}");

    // Utilizar el manejador de URL con el prefijo tel: para invocar a la
    // aplicación Phone de Apple, de lo contrario mostrar un diálogo de alerta.
    if(!UIApplication.SharedApplication.OpenUrl(URL))
    {
        var Alert = UIAlertController.Create("No soportado",
            "El esquema 'tel:' no es soportado en este dispositivo",
            UIAlertControllerStyle.Alert);
        Alert.AddAction(UIAlertAction.Create("Ok",
            UIAlertActionStyle.Default, null));
        PresentViewController(Alert, true, null);
    }
};
```

5. Guarda los cambios.

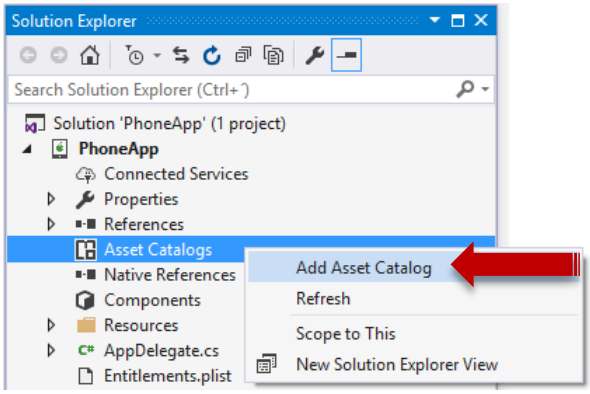
6. Compila la aplicación y verifica que no haya errores.



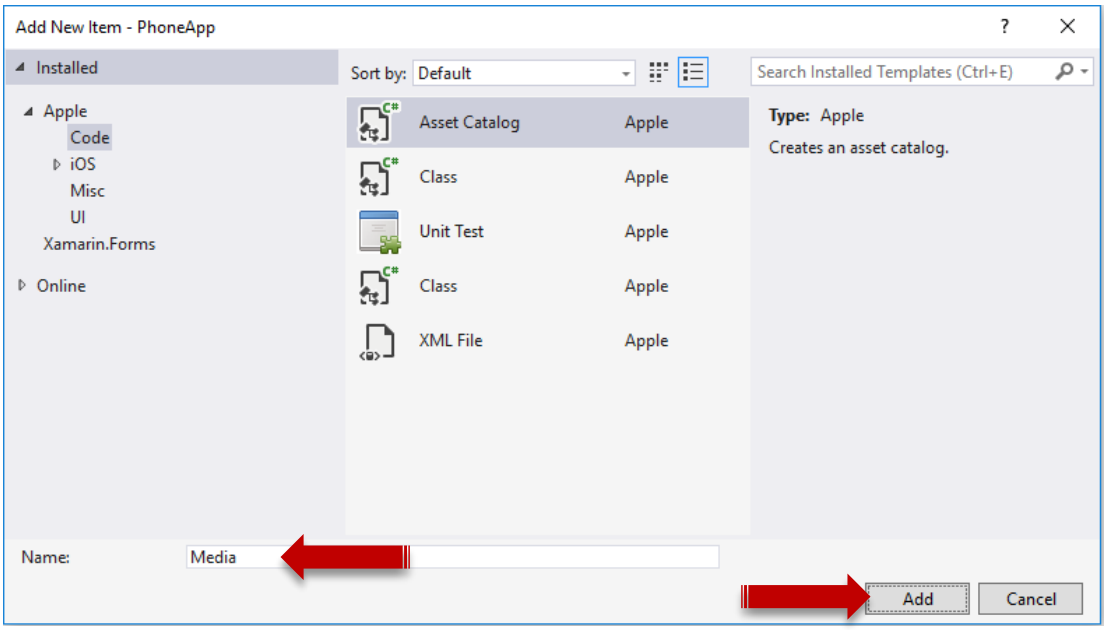
Tarea 5. Agregar el toque final a la aplicación.

En este momento la aplicación ya debe trabajar correctamente. Es tiempo de agregar los toques finales para editar el nombre de la aplicación y establecer sus iconos.

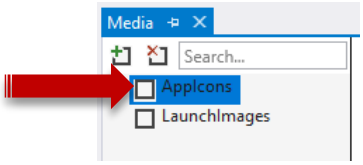
1. Para establecer los iconos de la aplicación se requiere un catálogo de recursos (***Asset Catalog***) que contenga todas las imágenes. Haz clic en la opción ***Add Asset Catalog*** del menú contextual del elemento ***Asset Catalogs***.



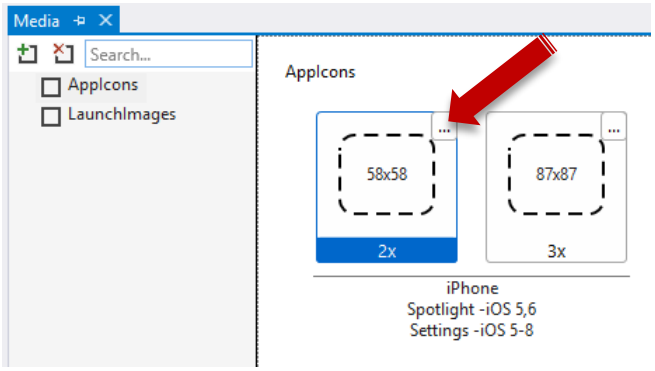
2. Asigna el nombre ***Media*** al catálogo y presiona ***Add***.



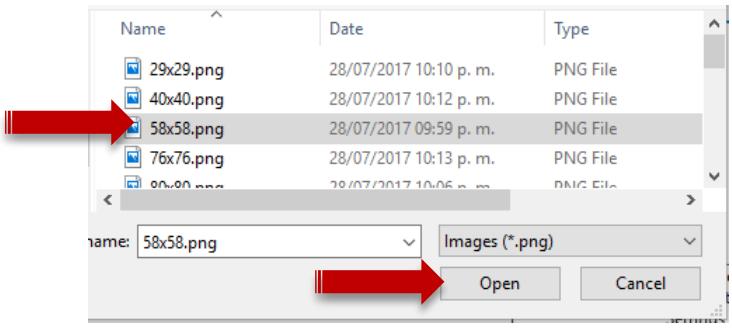
3. Haz clic en la opción ***AppIcons***.



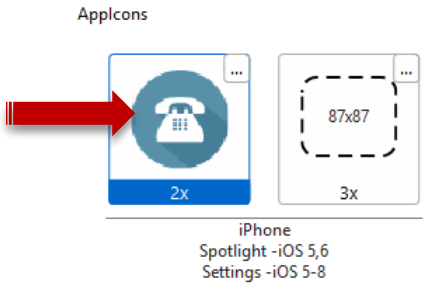
4. Haz clic en el selector de archivos  del elemento ***58x58***.



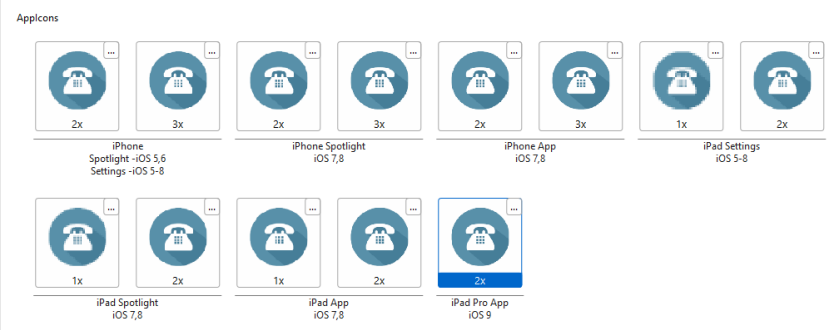
5. Selecciona el archivo ***58x58.png*** que se encuentra en el folder ***Media*** adjunto a este documento y haz clic en ***Open*** para agregarlo.



Visual Studio colocará la imagen en el elemento del catálogo.

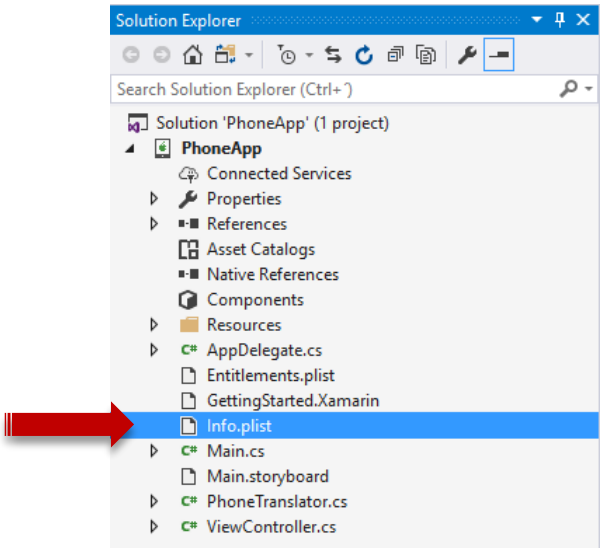


6. Repite el proceso anterior para agregar los demás iconos de la aplicación.

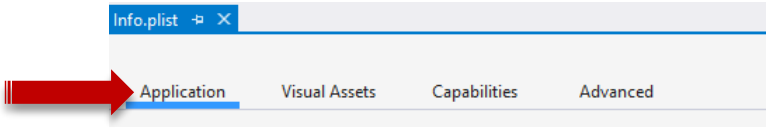


El siguiente paso será especificar el nombre y los iconos de la aplicación.

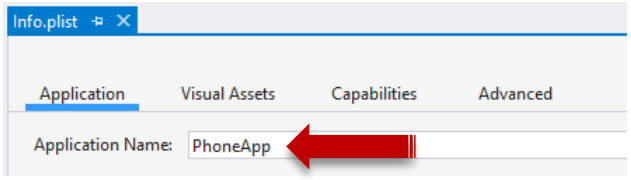
7. Haz doble clic sobre el archivo del manifiesto de la aplicación ***Info.plist***. Este archivo contiene información de configuración esencial de la aplicación.



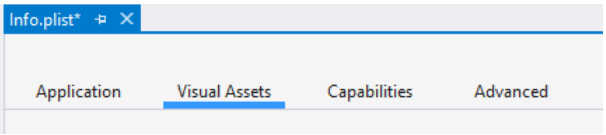
8. En la hoja de propiedades del archivo ***Info.plist*** selecciona la pestaña ***Application***.



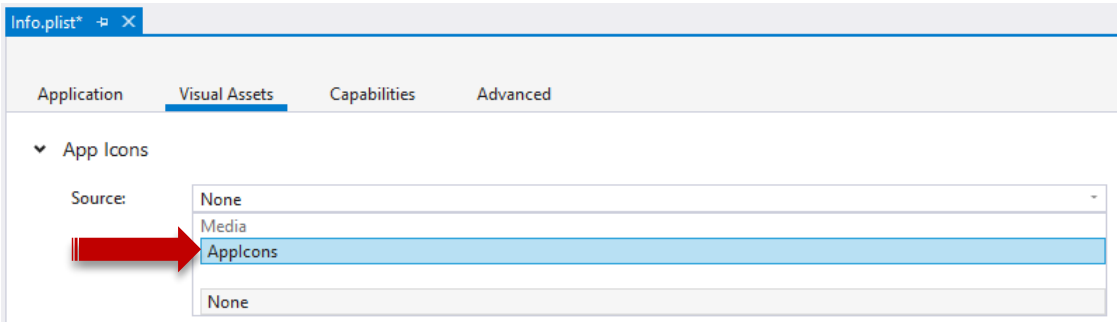
9. Verifica que el valor de ***Application Name*** sea ***PhoneApp***.



10. Selecciona la pestaña ***Visual Assets***.



11. En el cuadro de lista desplegable ***Source*** de la sección ***App Icons*** selecciona ***Media*** > ***AppIcons***.

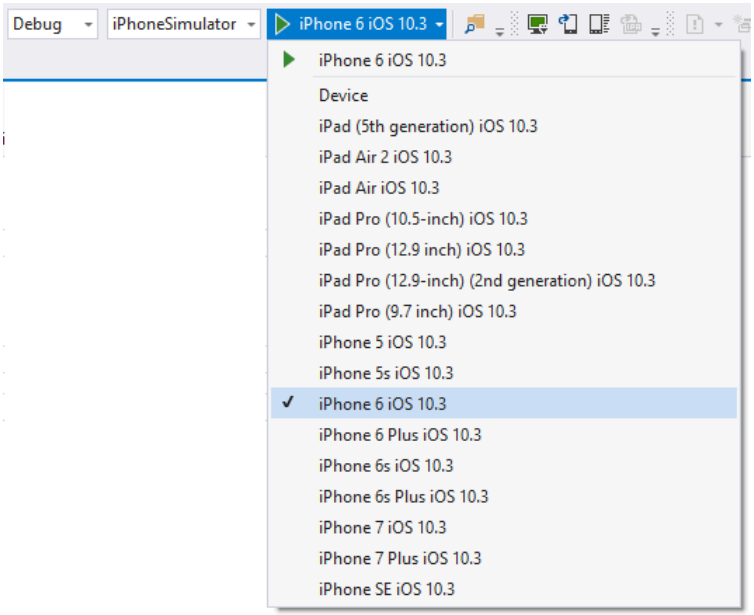


12. Guarda los cambios.

Tarea 6. Probar la aplicación.

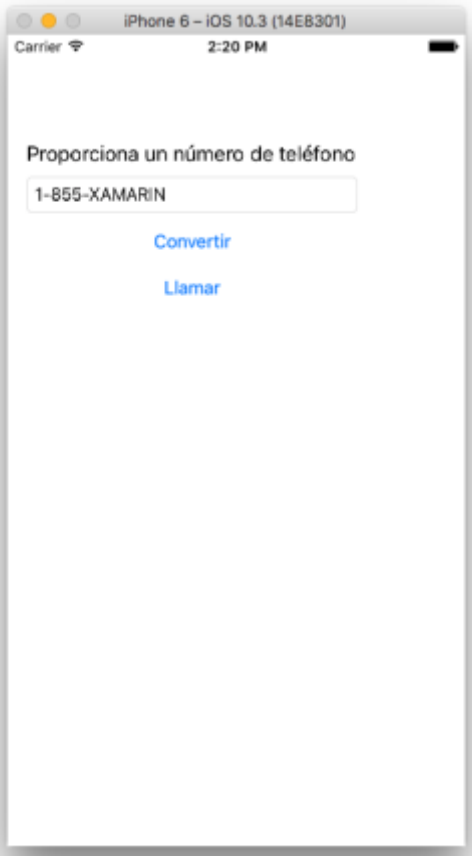
En este momento ya puedes probar la aplicación desplegándola hacia un simulador de iOS.

1. Selecciona el dispositivo en el cual deseas probar tu aplicación.



2. Haz clic en ***Start*** para desplegar y ejecutar la aplicación en el simulador seleccionado.

La aplicación será mostrada en el simulador.



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

4. Haz clic en el botón "***Llamar***". Las llamadas por teléfono no son soportadas en el simulador de iOS por lo que un diálogo de alerta será mostrado.

5. Haz clic en ***Ok*** para cerrar el diálogo de alerta.

6. Regresa a Visual Studio y detén la aplicación.

7. Si te es posible, intenta ejecutar la aplicación en un dispositivo iOS físico.

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.

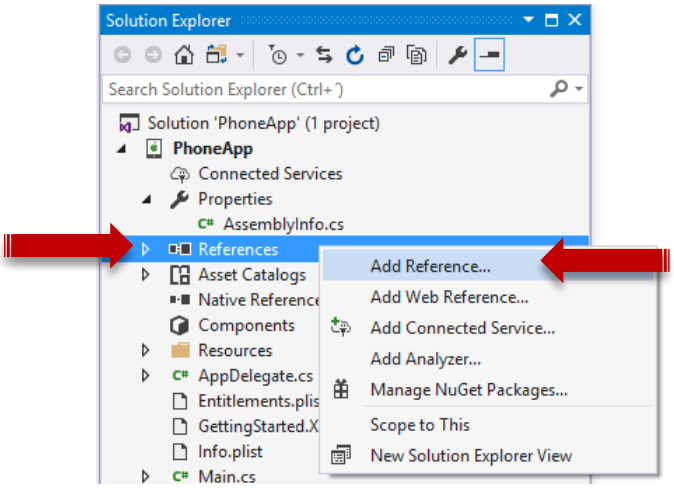
La funcionalidad que agregarás consumirá un ensamblado que representa una Capa de acceso a servicio (SAL) que será consumida por tu aplicación iOS.

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

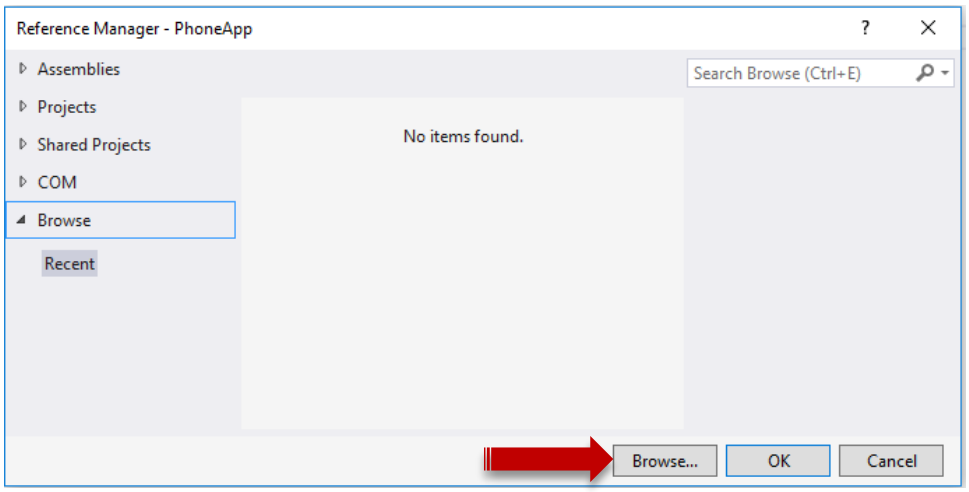
Tarea 1. Agregar los componentes de la Capa de acceso a Servicio.

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.

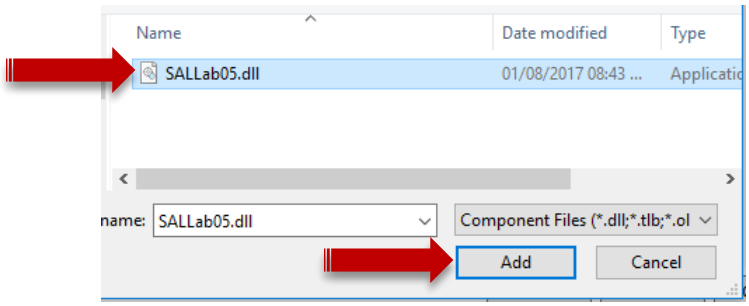
1. Accede a la opción ***Add Reference…*** del menú contextual del nodo ***References*** del proyecto iOS.



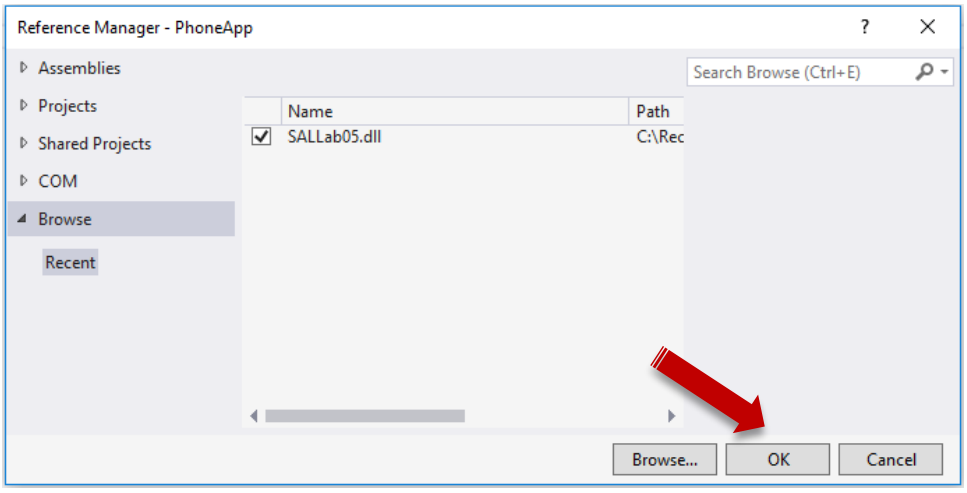
2. En la ventana ***Reference Manager*** haz clic en ***Browse…*** para buscar el ensamblado ***SALLab05.dll***.



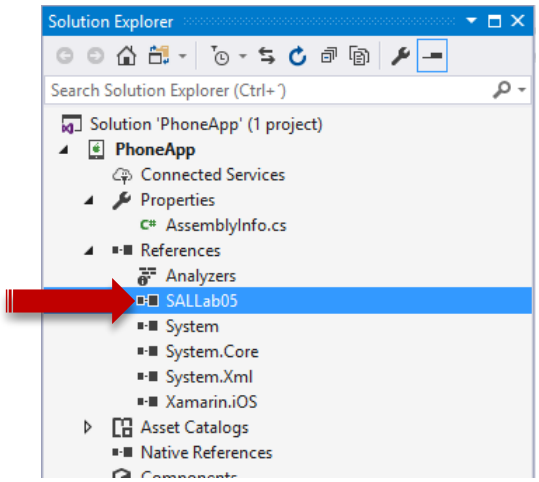
3. Selecciona el ensamblado ***SALLab05.dll*** y haz clic en ***Add***.



4. En la ventana ***Reference Manager*** haz clic en ***OK*** para agregar la referencia.

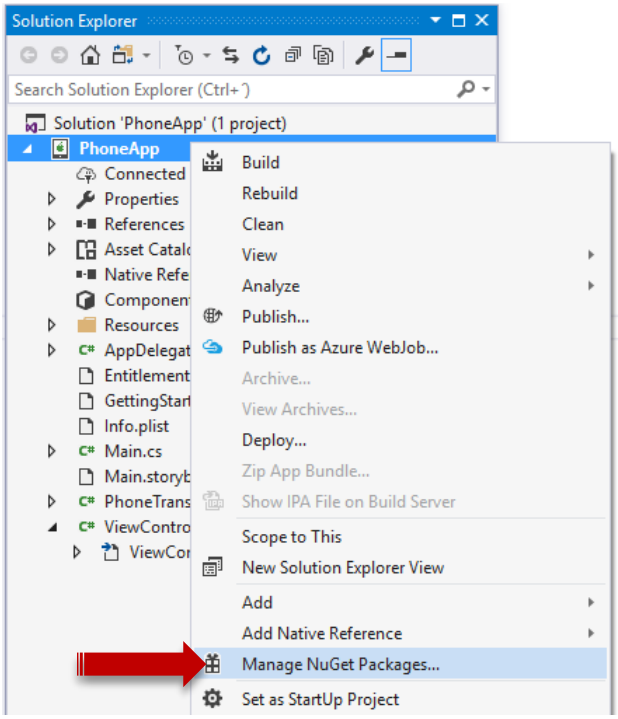


Puedes notar que la referencia ha sido agregada.

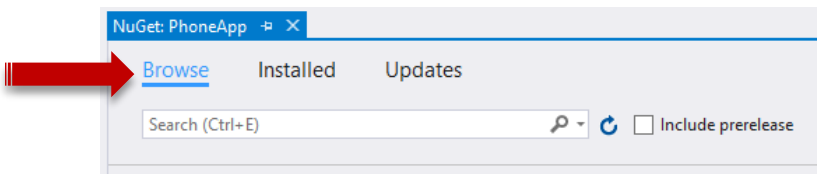


El componente realiza un intercambio de información en formato JSON con el servicio por lo que deberás agregar el paquete NuGet ***Newtonsoft.Json***.

5. Accede a la opción ***Manage NuGet Packages…*** del menú contextual de la aplicación iOS.

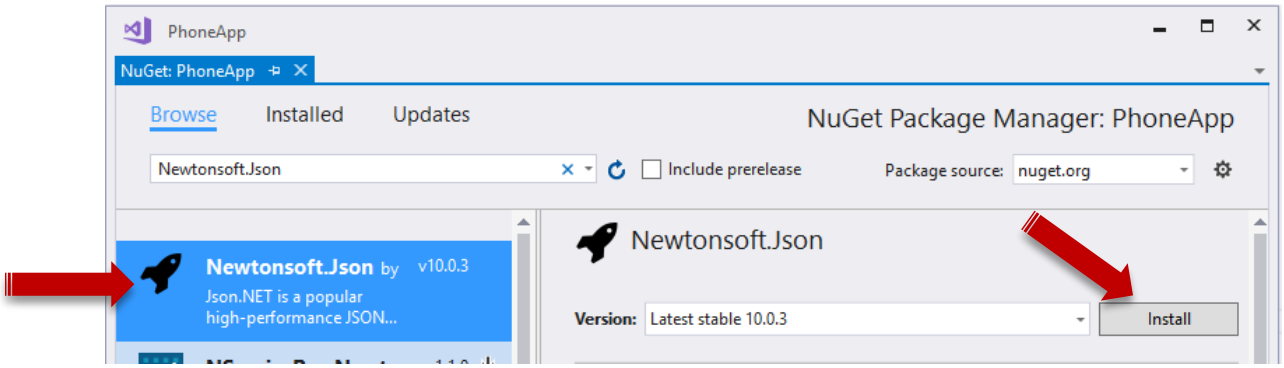


6. En la ventana ***NuGet*** haz clic en ***Browse*.**



7. En el cuadro de búsqueda escribe ***Newtonsoft.Json*.** La lista de resultados se actualizará automaticamente conforme vas escribiendo.

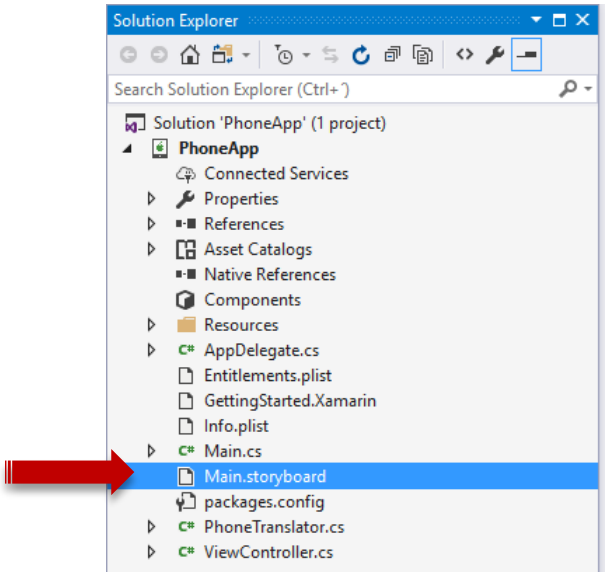
8. En la ventana de resultados haz clic en ***Newtonsoft.Json*** y haz clic en **Install** para instalar el paquete. Acepta el acuerdo de licencia y la instalación de los paquetes adicionales cuando te sea requerido.



Tarea 2. Agregar la funcionalidad para validar la actividad.

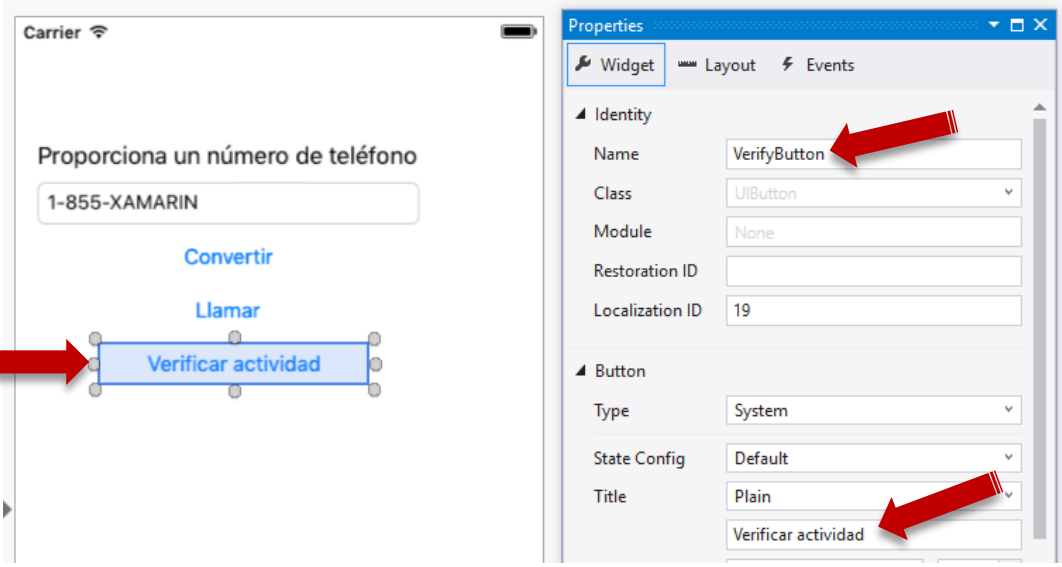
El componente DLL que agregaste te permite registrar tu actividad en la plataforma de TI Capacitación. En esta tarea agregarás la funcionalidad que te permitirá autenticarte en la plataforma de TI Capacitación para verificar la realización de tu actividad.

1. Abre el archivo **Main.storyboard** en el diseñador de iOS.



2. Agrega un elemento ***Button*** con el título ***Verificar actividad***.

3. Asigna el nombre ***VerifyButton*** al ***Button*** agregado. La pantalla se verá similar a la siguiente.



4. Haz doble clic sobre el nuevo botón para crear el manejador del evento ***TouchUpInside*** del botón ***VerifyButton***. El editor de código mostrará la clase ***ViewController*** y el método manejador del evento ***VerifyButton_TouchUpInside***.

```
partial void VerifyButton_TouchUpInside(UIButton sender)
{
    throw new NotImplementedException();
}
```

5. Modifica el código del método ***VerifyButton_TouchUpInside*** para que invoque al método ***Validate*** (que aún no existe).

```
partial void VerifyButton_TouchUpInside(UIButton sender)
{
    Validate();
}
```

6. Agrega en la misma clase ***ViewController*** el siguiente código para definir el método asíncrono ***Validate***.

```
async void Validate()
{
}
```

7. En el cuerpo del método ***Validate*** agrega el siguiente código para crear una instancia del cliente del servicio de validación.

```
var Client = new SALLab05.ServiceClient();
```

8. Agrega ahora el siguiente código para invocar al método ***ValidateAsync*** proporcionándole el correo y clave de acceso que utilizas para iniciar sesión en la plataforma de TI Capacitación. El método también recibe una instancia de la clase ***ViewController***.

```
var Result = await Client.ValidateAsync(
    "TuCorreo", "TuClaveDeAcceso", this);
```

NOTA: Después de terminar el laboratorio es importante que elimines estos datos para que no queden expuestos. En laboratorios posteriores aprenderás a solicitar esos datos en tiempo de ejecución para que no queden expuestos como código duro. La comunicación del componente con la plataforma de TI Capacitación se realiza mediante HTTPS para seguridad de tu información.

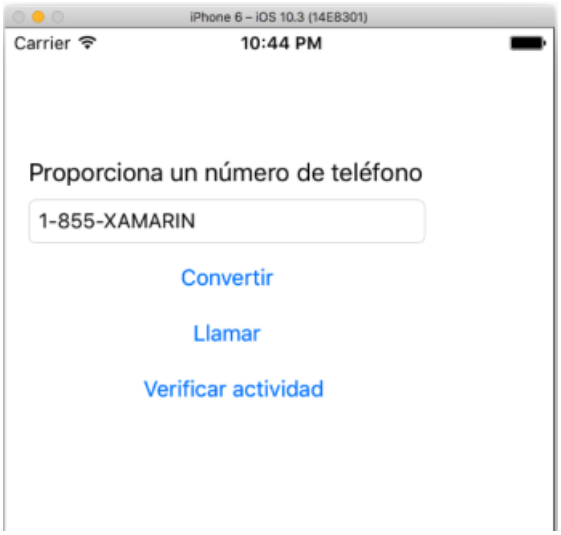
9. Agrega el siguiente código para mostrar un dialogo de alerta con el resultado de la validación.

```
var Alert = UIAlertController.Create("Resultado",
    $"{Result.Status}\n{Result.FullName}\n{Result.Token}",
    UIAlertControllerStyle.Alert);

Alert.AddAction(UIAlertAction.Create("Ok",
    UIAlertActionStyle.Default, null));

PresentViewController(Alert, true, null);
```

Tarea 3. Ejecutar la aplicación.

Finalmente, está todo listo para ejecutar la aplicación y realizar la validación de la actividad.

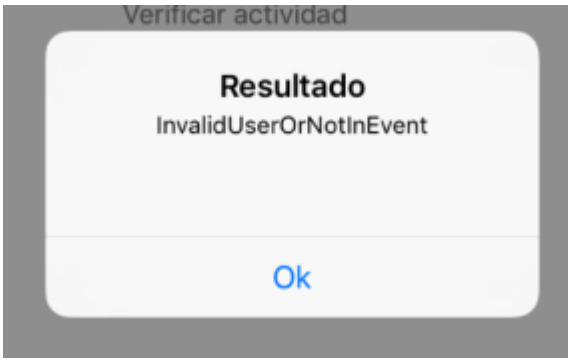
1. Selecciona el simulador iOS de tu preferencia y ejecuta la aplicación. La aplicación se verá similar a la siguiente.



1. Selecciona el botón ***Verificar actividad***. La aplicación iniciará el proceso de validación de la actividad. Al finalizar la validación se mostrará un dialogo de alerta mostrando tus datos y el estatus de la validación.

   Si la validación se realiza con éxito, te será mostrado tu nombre completo y el estatus ***Success*** como en la siguiente imagen.

   Si proporcionaste tus credenciales incorrectas, te mostrará un mensaje similar al siguiente.



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

2. Regresa a Visual Studio y detén la ejecución de la aplicación.

Elimina los datos de tus credenciales de acceso a la plataforma de TI Capacitación.

Resumen

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

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

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