



Laboratorio

Proyectos Compartidos *(Shared Projects)*

Versión: 1.0.0
Abril de 2017



[Miguel Muñoz Serafín](#)
@msmdotnet





CONTENIDO

INTRODUCCIÓN

EJERCICIO 1: UTILIZANDO PROYECTOS COMPARTIDOS PARA COMPARTIR CÓDIGO

Tarea 1. Crear un Proyecto Compartido.

Tarea 2. Agregar proyectos de aplicación a la solución.

EJERCICIO 2: VALIDANDO TU ACTIVIDAD

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

Tarea 2. Agregar la funcionalidad para validar la actividad.

Tarea 3. Ejecutar la aplicación.

RESUMEN



Introducción

Los Proyectos Compartidos (también conocidos como **Shared Asset Projects o SAP**), nos permiten escribir código que es compartido entre varios tipos de proyectos destino, incluidas las aplicaciones Xamarin. El código común que es referenciado por varios proyectos de diferentes aplicaciones es compilado como parte de cada proyecto que le hace referencia y puede incluir directivas de compilación para ayudar a incorporar funcionalidad específica de la plataforma dentro del código base compartido.

El código en un Proyecto Compartido puede contener directivas de compilación que habiliten o deshabiliten secciones de código dependiendo del proyecto de aplicación esté utilizando ese código.

Un Proyecto Compartido no es compilado por sí mismo, existe simplemente para agrupar archivos de código fuente que pueden ser incluidos en otros proyectos. Cuando es referenciado por otro proyecto, el código es efectivamente compilado como parte de ese proyecto. Los Proyectos Compartidos no pueden referenciar a otro tipo de proyecto, incluyendo a otros Proyectos Compartidos. Si anteriormente has utilizado la vinculación de archivos para compartir código entre proyectos, los Proyectos Compartidos trabajan de una manera similar, pero con mucho mayor soporte del IDE.

El soporte a Proyectos Compartidos fue agregado en Xamarin Studio 5 y Visual Studio 2013 Update 2.

En este laboratorio examinarás la forma de crear y utilizar Proyectos Compartidos utilizando Visual Studio.

Objetivos

Al finalizar este laboratorio, los participantes serán capaces de:

- Crear un Proyecto Compartido con Visual Studio.
- Utilizar un Proyecto Compartido desde un proyecto de aplicación.

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 diseñados con Visual Studio Enterprise 2017 y Windows 10 Professional.
- Xamarin para Visual Studio.

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



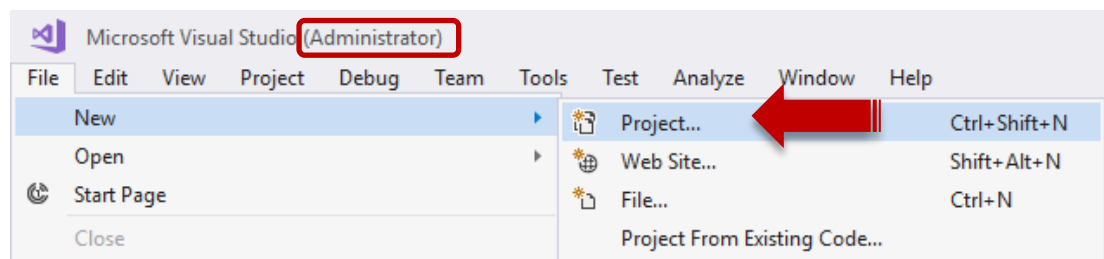
Ejercicio 1: Utilizando Proyectos Compartidos para compartir código

En este ejercicio crearás un Proyecto Compartido que será referenciado por otros proyectos de aplicación.

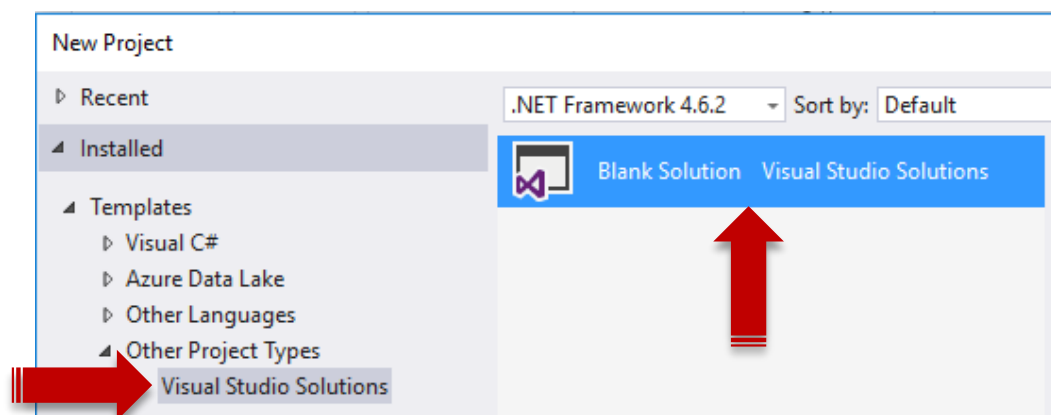
Tarea 1. Crear un Proyecto Compartido.

Realiza los siguientes pasos para crear una solución con un proyecto Compartido.

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**.

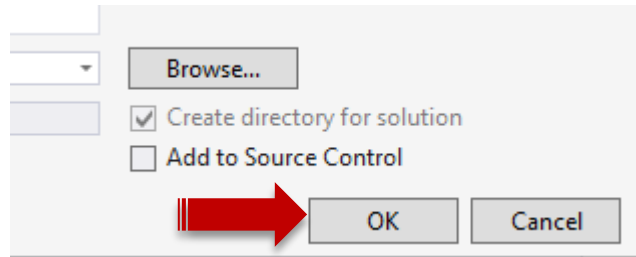


4. Proporciona el nombre y ubicación de la solución.

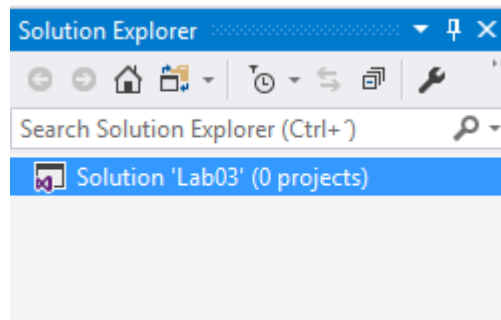
Name:	Lab03
Location:	C:\demos\
Solution name:	Lab03



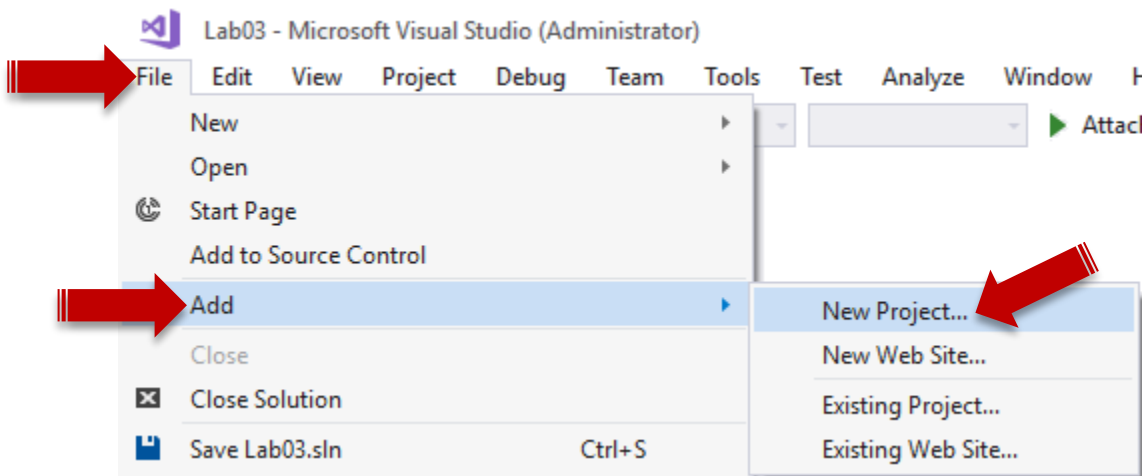
5. Haz clic en **OK** para crear la solución en blanco.



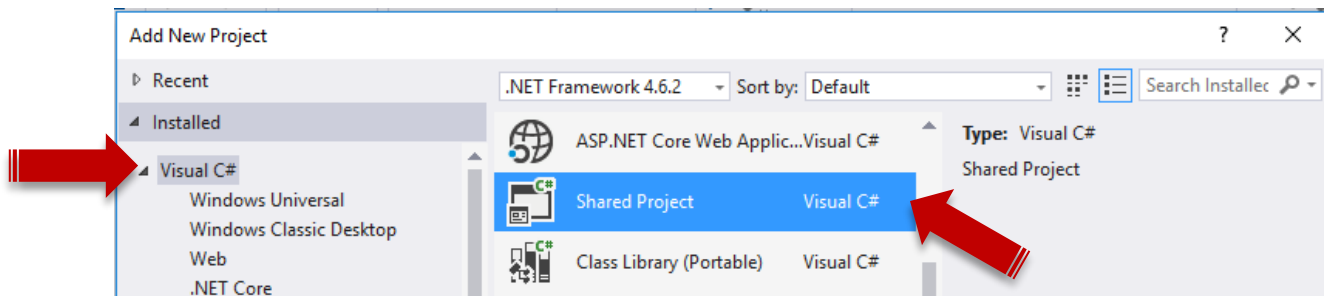
El explorador de soluciones de Visual Studio mostrará la solución vacía.



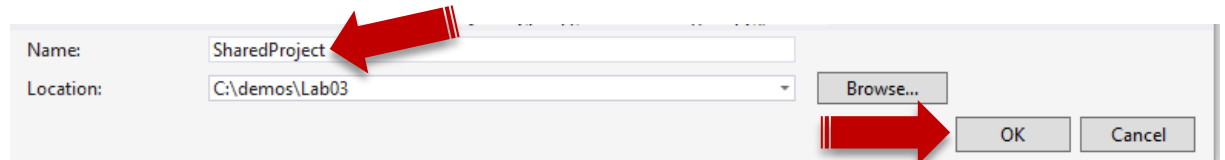
6. Para agregar un nuevo Proyecto Compartido a la solución, selecciona la opción **File > Add > New Project...**



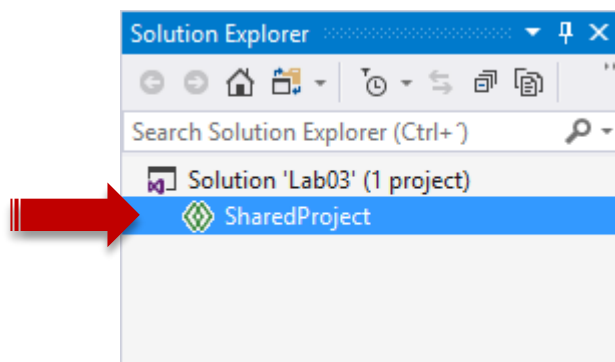
7. En la ventana **New Project** selecciona la plantilla **Shared Project**.



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

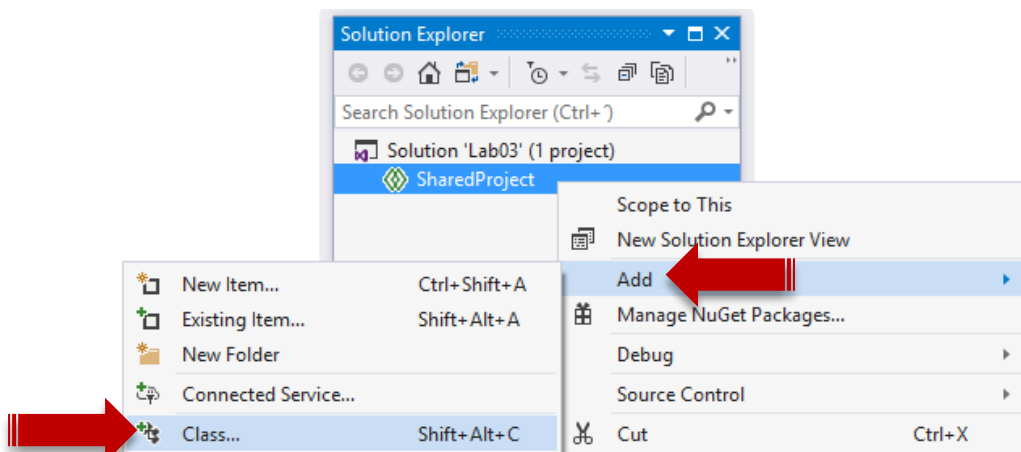


El explorador de soluciones te mostrará el nuevo proyecto agregado.



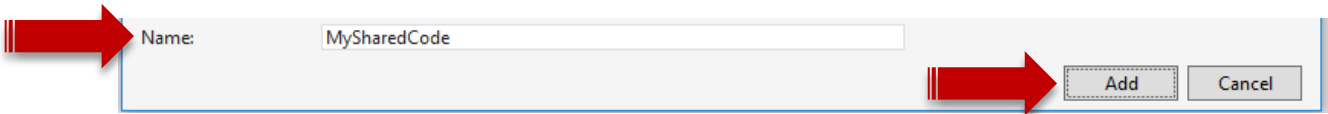
Puedes observar que no hay referencias o nodos de componentes ya que estos no son soportados en los Proyectos Compartidos.

9. Selecciona la opción **Add > Class...** del menú contextual del Proyecto Compartido para agregar un archivo **.cs** que contendrá el código a compartir.

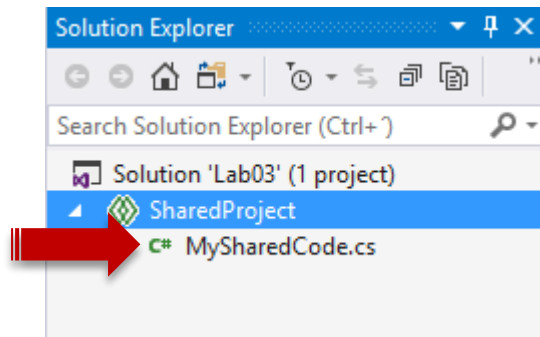




10. Asigna un nombre a la clase y haz clic en **Add** para agregar el archivo al Proyecto Compartido.



El explorador de soluciones mostrará el nuevo archivo de código agregado.



11. Abre el archivo **.cs** que agregaste.

12. Agrega la siguiente definición de método a la clase.

```
class MySharedCode
{
    public string GetFilePath(string fileName)
    {
    }
}
```

El método **GetFilePath** implementará la funcionalidad para determinar la ruta de un archivo dependiendo de la plataforma en la que se esté ejecutando la aplicación. Cada plataforma tiene una manera distinta de acceder a su sistema de archivos.

13. Agrega el siguiente código para devolver la ruta del archivo en plataformas UWP. Nota el uso de la directiva **#if**.


```
#if WINDOWS_UWP
var FilePath = Path.Combine(
    Windows.Storage.ApplicationData.Current.LocalFolder.Path,
    filename);
```

Cuando el proyecto que haga referencia al proyecto compartido defina el símbolo **WINDOWS_UWP** (como en el caso de las aplicaciones UWP), la directiva **#if** se evaluará en **true** y el código será compilado.

14. Agrega la siguiente directiva **#else** para indicar otra condición a evaluar cuando la directiva **#if** no sea evaluada en **true**.



```
#if WINDOWS_UWP
var FilePath = Path.Combine(
    Windows.Storage.ApplicationData.Current.LocalFolder.Path,
    fileName);
#else
```



15. Agrega el siguiente código para devolver la ruta del archivo en plataformas Android y finalizar las ramificaciones **#if**.

```
#if __ANDROID__
string LibraryPath =
    Environment.GetFolderPath(
        Environment.SpecialFolder.Personal); ;
var FilePath = Path.Combine(LibraryPath, fileName);
#endif
#endif
```

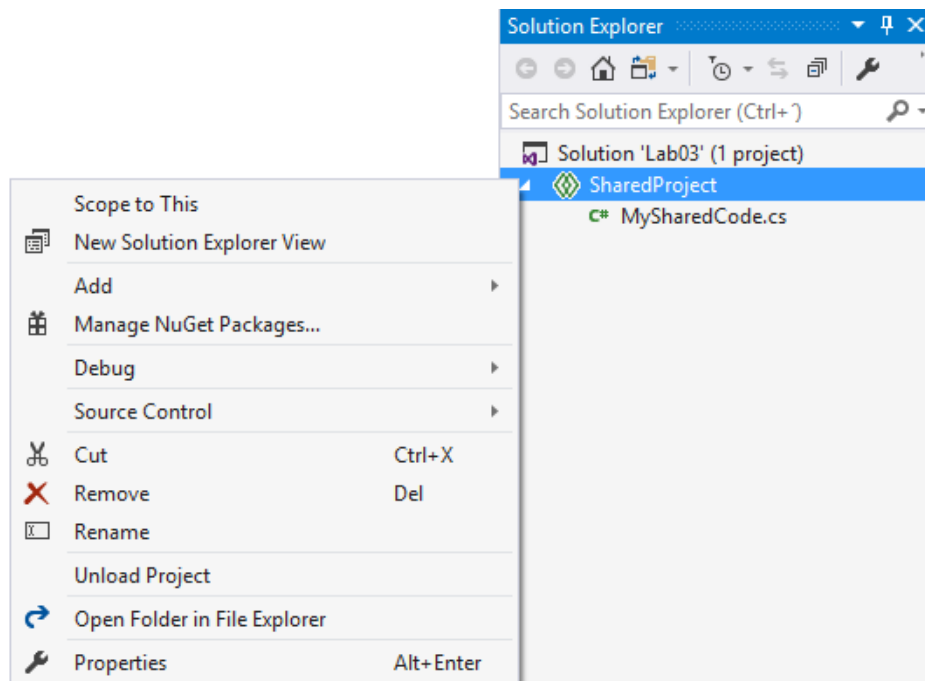
16. Escribe el siguiente código para devolver el valor del método.

```
return FilePath;
```

17. Guarda los cambios realizados.

Un Proyecto Compartido no es compilado cuando no exista algo que lo referencie, por eso los errores de sintaxis (o cualquier otro error) no serán señalados hasta que haya sido referenciado por alguien más.

Puedes notar que el menú contextual del Proyecto Compartido no tiene las opciones de compilación.

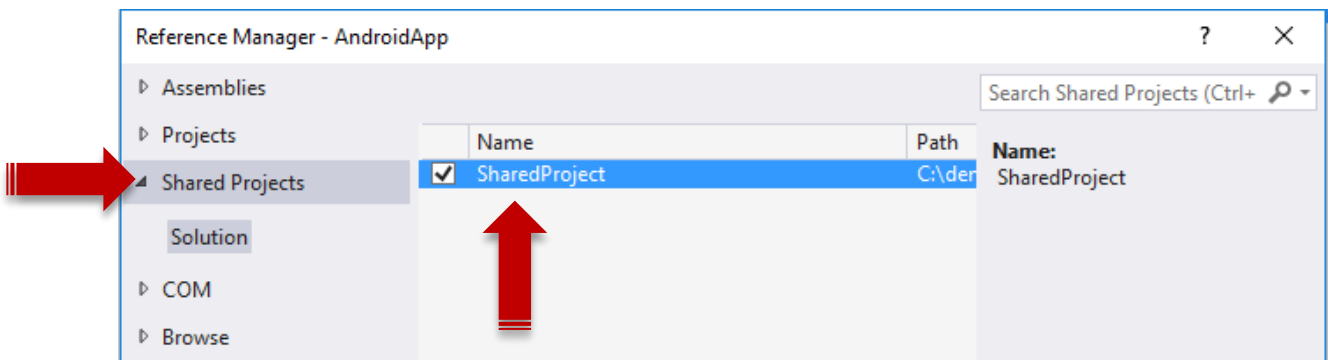




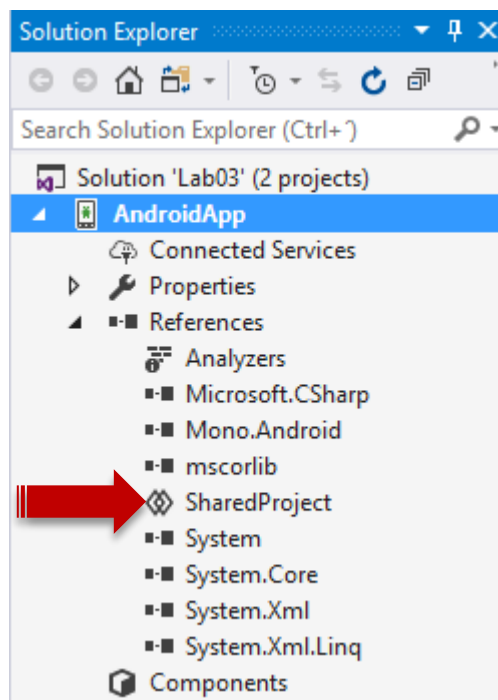
Tarea 2. Agregar proyectos de aplicación a la solución.

Para que un Proyecto Compartido pueda ser útil, necesita ser referenciado por al menos un proyecto que pueda ser compilado (tal como una aplicación Android, iOS, Windows o un proyecto PCL).

1. Agrega un nuevo proyecto Android a la solución utilizando la plantilla **Blank App (Android)**.
2. En el proyecto Android, agrega la referencia del Proyecto Compartido. La forma de agregar una referencia del Proyecto Compartido es similar a la forma en que se agrega una referencia a cualquier otro proyecto.



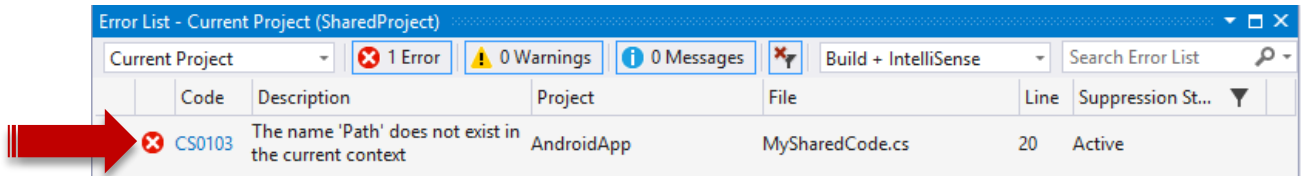
El explorador de soluciones mostrará la referencia del proyecto agregado.



Una vez que el Proyecto Compartido es referenciado por otro proyecto, podemos compilar la solución y ver cualquier error en el código.



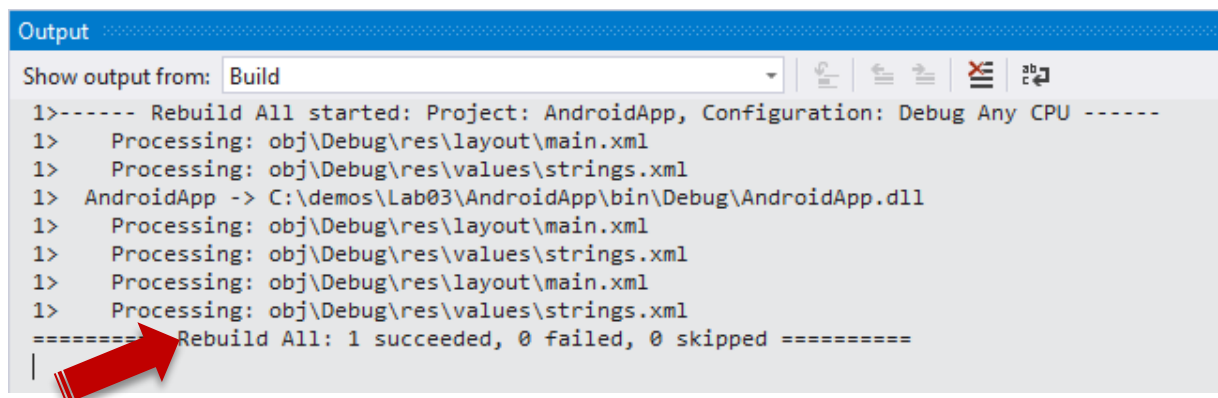
3. Compila la solución. Puedes notar que un error ha sido encontrado.



4. Agrega el siguiente código a la lista de sentencias using del archivo .cs del Proyecto Compartido.

```
using System.IO;
```

5. Compila nuevamente la solución y asegúrate de que haya sido compilada con éxito.



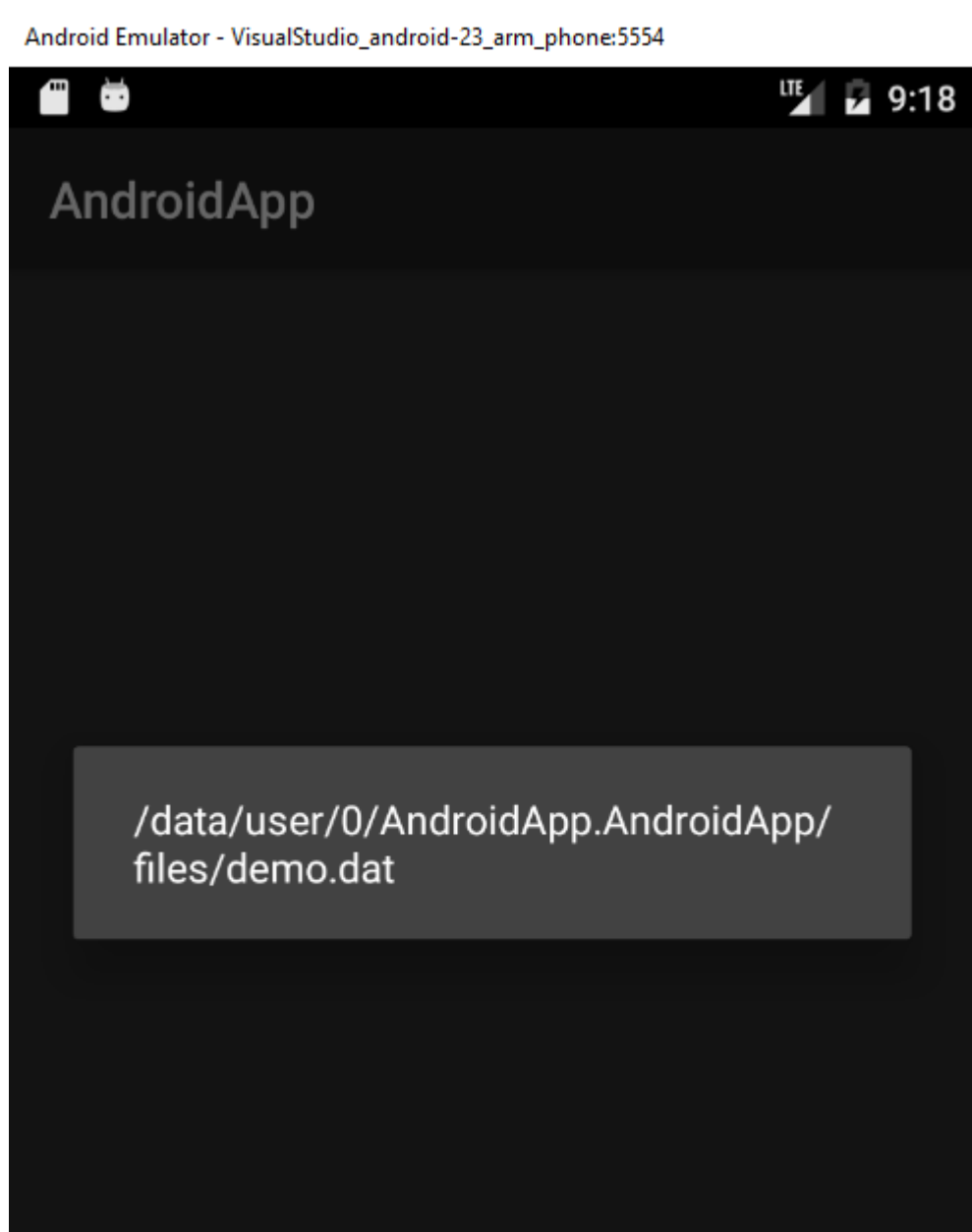
6. Abre el archivo **MainActivity.cs** del proyecto Android y agrega el siguiente código para invocar al código compartido y mostrar el resultado devuelto.

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

    var Helper = new SharedProject.MySharedCode();
    new AlertDialog.Builder(this)
        .SetMessage(Helper.GetFilePath("demo.dat"))
        .Show();

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

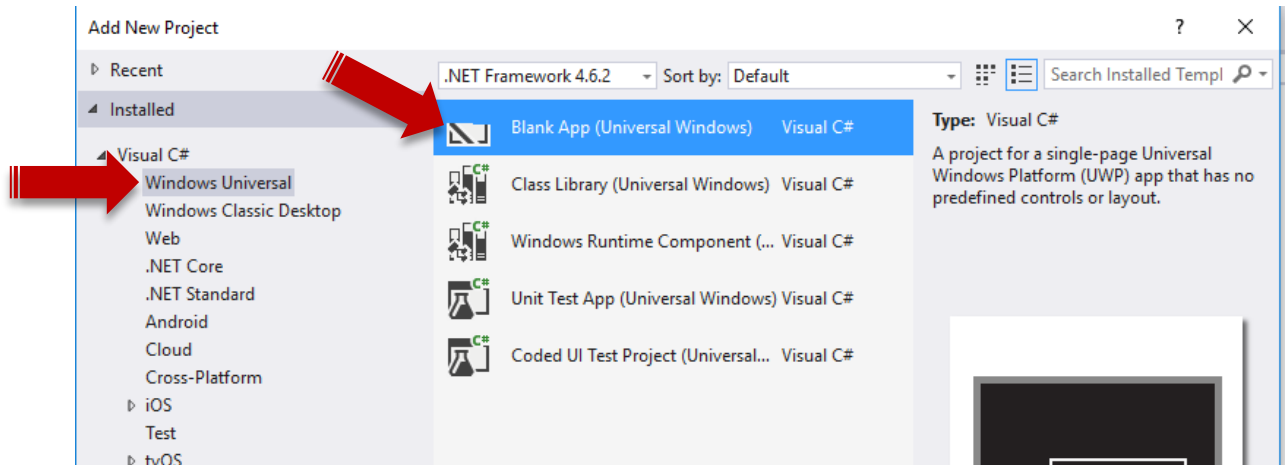
7. Ejecuta la aplicación. Un mensaje será mostrado con el resultado devuelto por el método *GetFilePath*.



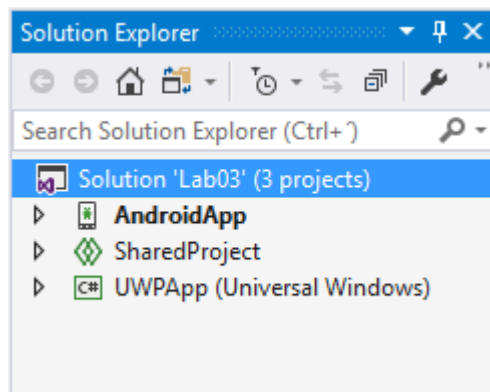
Podrás notar que el código que se está ejecutando corresponde al contenido dentro de la directiva `#if __Android__` del proyecto compartido.

Los proyectos Android definen de manera predeterminada el identificador `__ANDROID__` y varios identificadores `__ANDROID_nn__` (donde nn representa cada nivel de API Android soportada).

8. Agrega a la solución un nuevo proyecto para una aplicación UWP utilizando la plantilla **Blank App (Universal Windows)**.

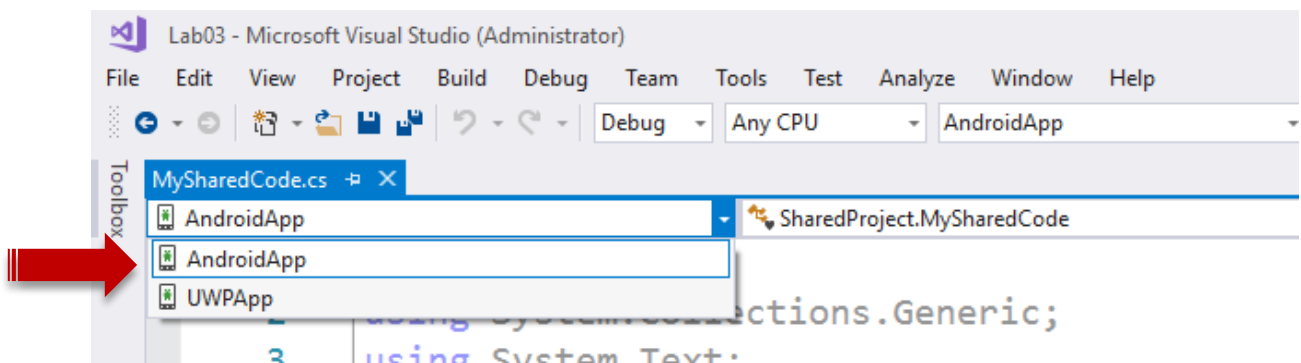


9. El explorador de soluciones mostrará ahora los 3 proyectos.

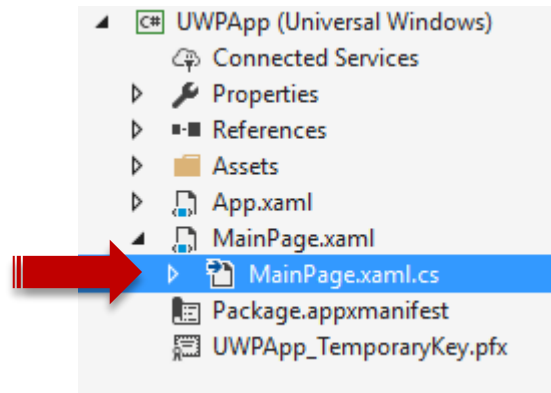


10. En el proyecto UWP, agrega la referencia del Proyecto Compartido.

Puedes notar que la lista de proyectos que hacen referencia al Proyecto Compartido es mostrada en la parte superior izquierda del editor de código.



11. Abre el archivo **MainPage.xaml.cs** del proyecto UWP.



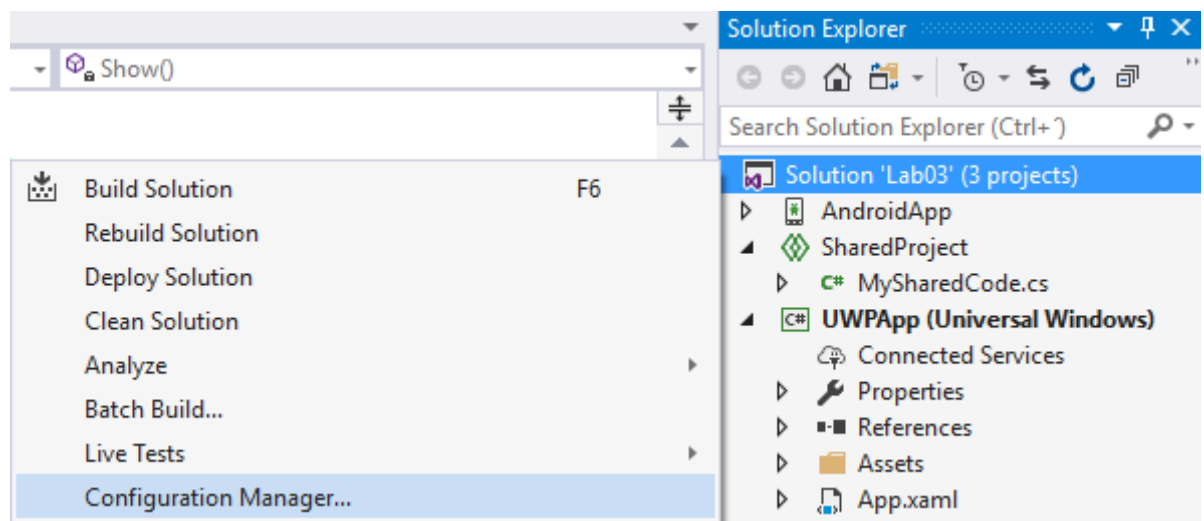
12. Agrega el siguiente código para invocar al código compartido y mostrar el resultado devuelto.

```
public MainPage()
{
    this.InitializeComponent();
    Show();
}

async void Show()
{
    var Helper = new SharedProject.MySharedCode();
    var Dialog =
        new Windows.UI.Popups.MessageDialog(
            Helper.GetFilePath("demo.dat"));
    await Dialog.ShowAsync();
}
```

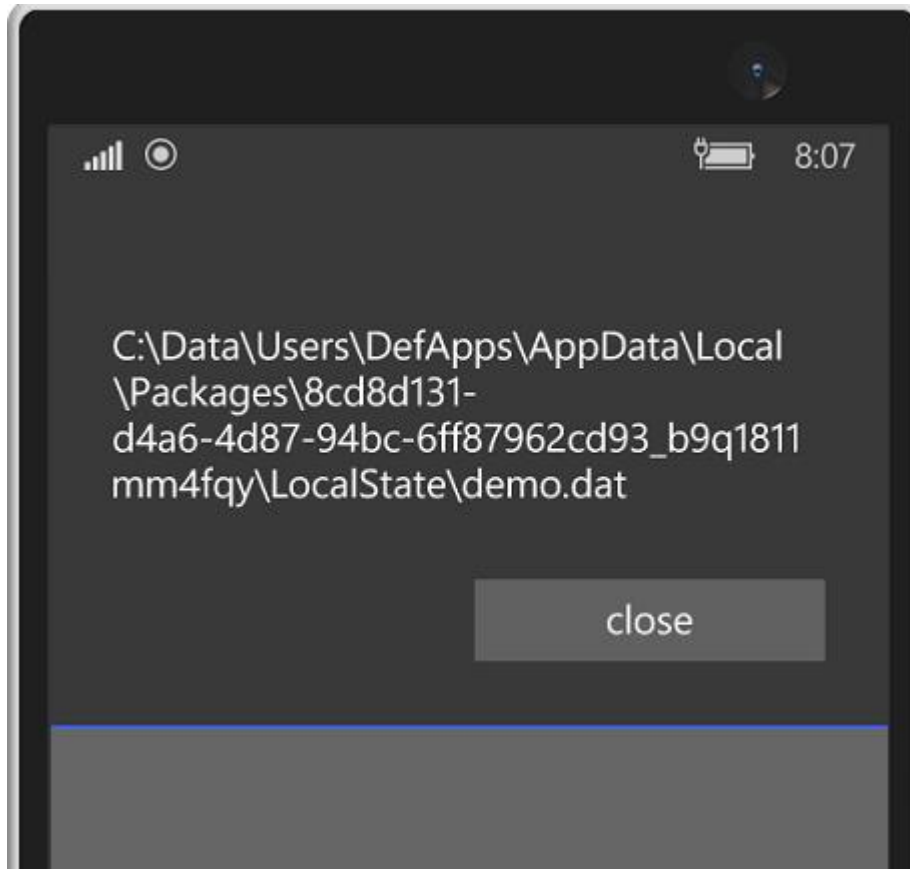
13. Establece el proyecto UWP como proyecto de inicio.

14. Accede a la opción **Configuration Manager...** del menú contextual de la solución.



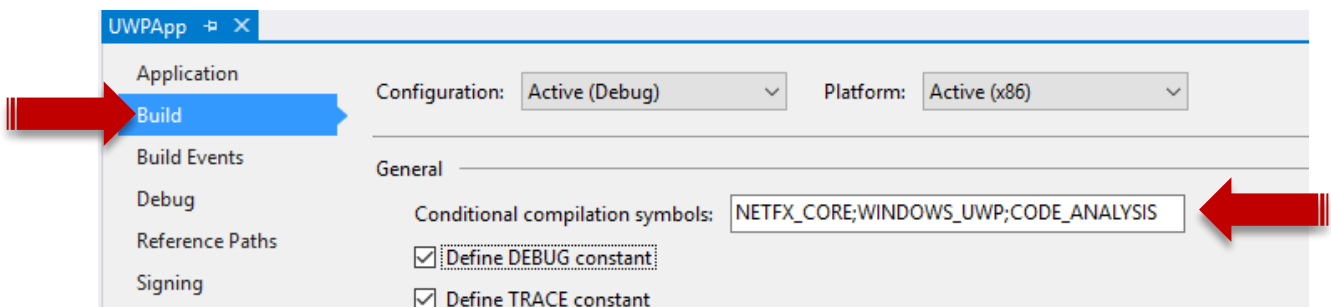


17. Ejecuta la aplicación. Un mensaje será mostrado con el resultado devuelto por el método *GetFilePath*.



Puedes notar que las rutas devueltas por las aplicaciones Android y UWP tienen una sintaxis distinta.

Los proyectos UWP definen el identificador **WINDOWS_UWP** y varios identificadores más (separados por punto y coma) que puedes observar en las propiedades del proyecto. Los identificadores (o símbolos de compilación) son sensibles a mayúsculas y minúsculas.





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.

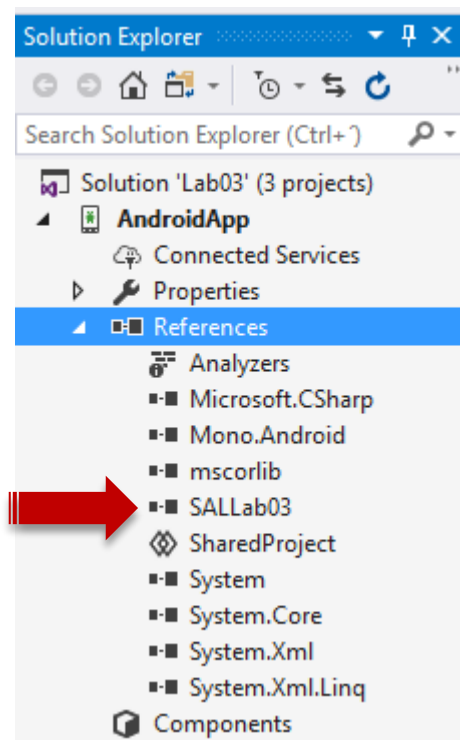
Es importante que realices cada laboratorio del diplomado ya que esto te dará derecho a obtener el diploma final del mismo.

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

En esta tarea agregarás una referencia al ensamblado **SALLab03.dll** que implementa la capa de acceso a servicio. El archivo **SALLab03.dll** se encuentra disponible junto con este documento.

1. Accede a la opción **Add > Reference...** del menú contextual de la aplicación Android para agregar la referencia del ensamblado **SALLab03.dll**.

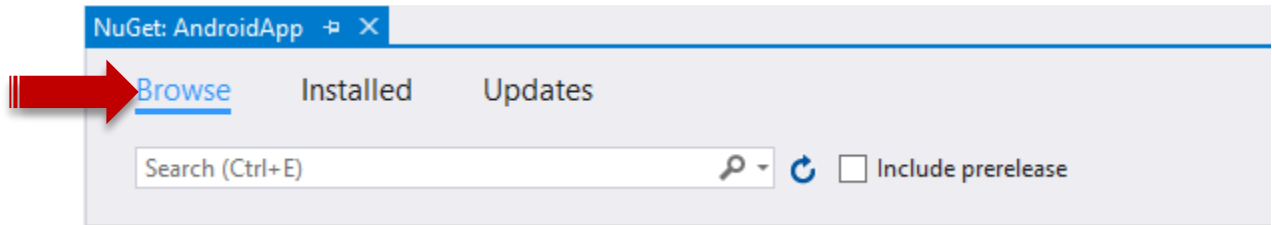
Puedes notar que la referencia ha sido agregada.



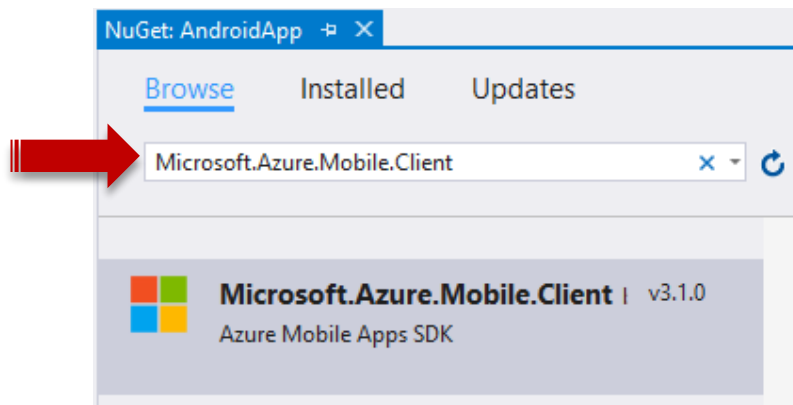
Este componente realiza una conexión a un servicio de Azure Mobile, por lo tanto, será necesario agregar el paquete NuGet **Microsoft.Azure.Mobile.Client**.



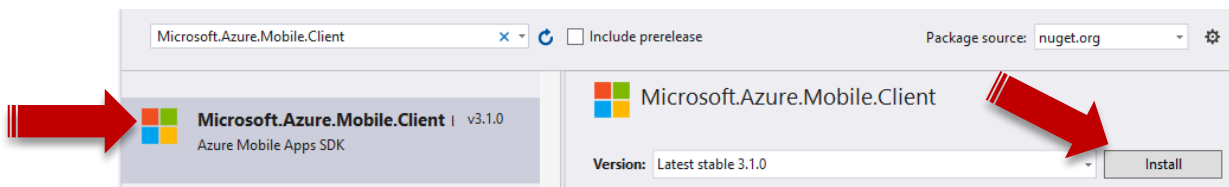
2. Accede a la opción **Manage NuGet Packages...** del menú contextual de la aplicación Android.
3. En la ventana **NuGet** haz clic en **Browse**.



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



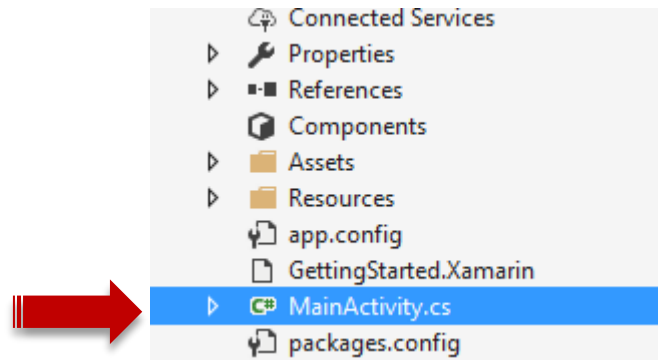
5. En la ventana de resultados haz clic en **Microsoft.Azure.Mobile.Client** y haz clic en **Install** para instalar el paquete. Acepta la instalación de los paquetes adicionales cuando te sea requerido.



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 y Microsoft. El componente se comunica con la plataforma de TI Capacitación para autenticarte y posteriormente envía un registro a la plataforma Microsoft.

1. Abre el archivo **MainActivity.cs** ubicado dentro del proyecto Android.



2. Dentro de la clase **MainActivity** agrega la siguiente definición de método.

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

3. Dentro del método, agrega el siguiente código para crear una instancia del componente de acceso a servicio.

```
var ServiceClient =
    new SALLab03.ServiceClient();
```

4. Agrega el siguiente código con los datos de tus credenciales de acceso a la plataforma del aula virtual de TI Capacitación que utilizas en este entrenamiento.

```
string StudentEmail = "TuCorreoElectrónico";
string Password = "TuContraseña";
```

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.

5. Agrega el siguiente código para obtener el identificador Android que utilizas.

```
string myDevice =
    Android.Provider.Settings.Secure.GetString(
        ContentResolver,
        Android.Provider.Settings.Secure.AndroidId);
```

6. Agrega el siguiente código para invocar al método que validará tu actividad.

```
var Result =
    await ServiceClient.ValidateAsync(
        StudentEmail, Password, myDevice);
```




7. Agrega el siguiente código que permitirá mostrar una alerta con el resultado de la validación.

```
Android.App.AlertDialog.Builder Builder =  
    new AlertDialog.Builder(this);  
AlertDialog Alert = Builder.Create();  
Alert.SetTitle("Resultado de la verificación");  
Alert.SetIcon(Resource.Drawable.Icon);  
Alert.SetMessage(  
    $"{Result.Status}\n{Result.Fullname}\n{Result.Token}");  
Alert.SetButton("Ok", (s, ev) => { });  
Alert.Show();
```

El siguiente paso será agregar el código que invoque a este método.

8. Agrega la siguiente instrucción en el método **OnCreate**.

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



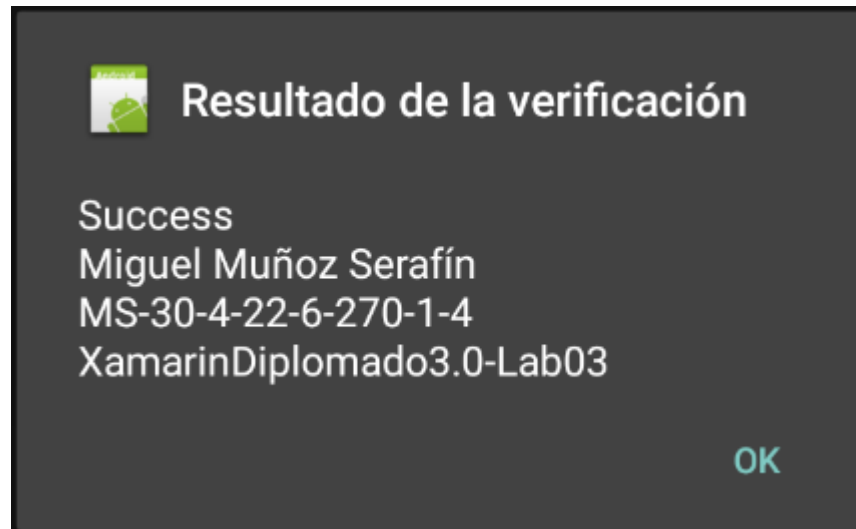
Tarea 3. Ejecutar la aplicación.

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

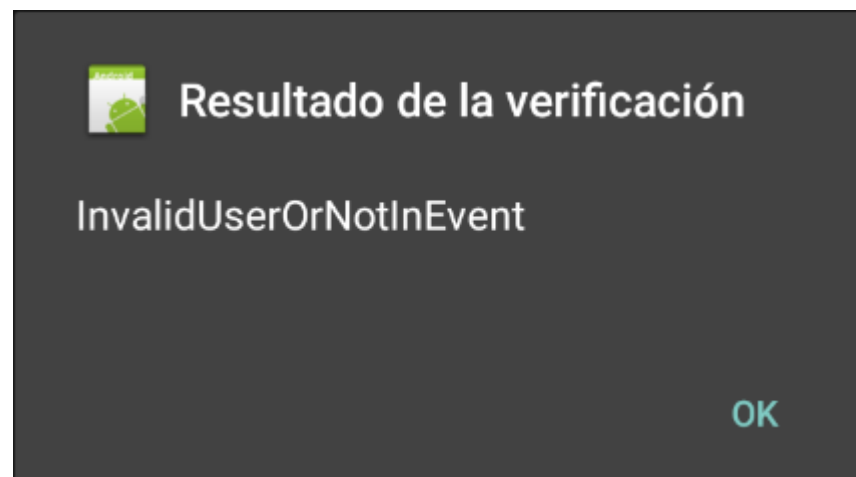
1. Selecciona la opción **Set as StartUp Project** del menú contextual del proyecto Android para establecerlo como proyecto de inicio.
2. Selecciona el emulador Android de tu preferencia y ejecuta la aplicación.

La aplicación será ejecutada en el emulador e iniciará el proceso de validación de la actividad. Al finalizar la validación se mostrará un cuadro de diálogo 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.



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

3. Regresa a Visual Studio y detén la ejecución de la aplicación.
4. **Elimina 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 este laboratorio, puedes solicitar apoyo en los grupos de Facebook siguientes:

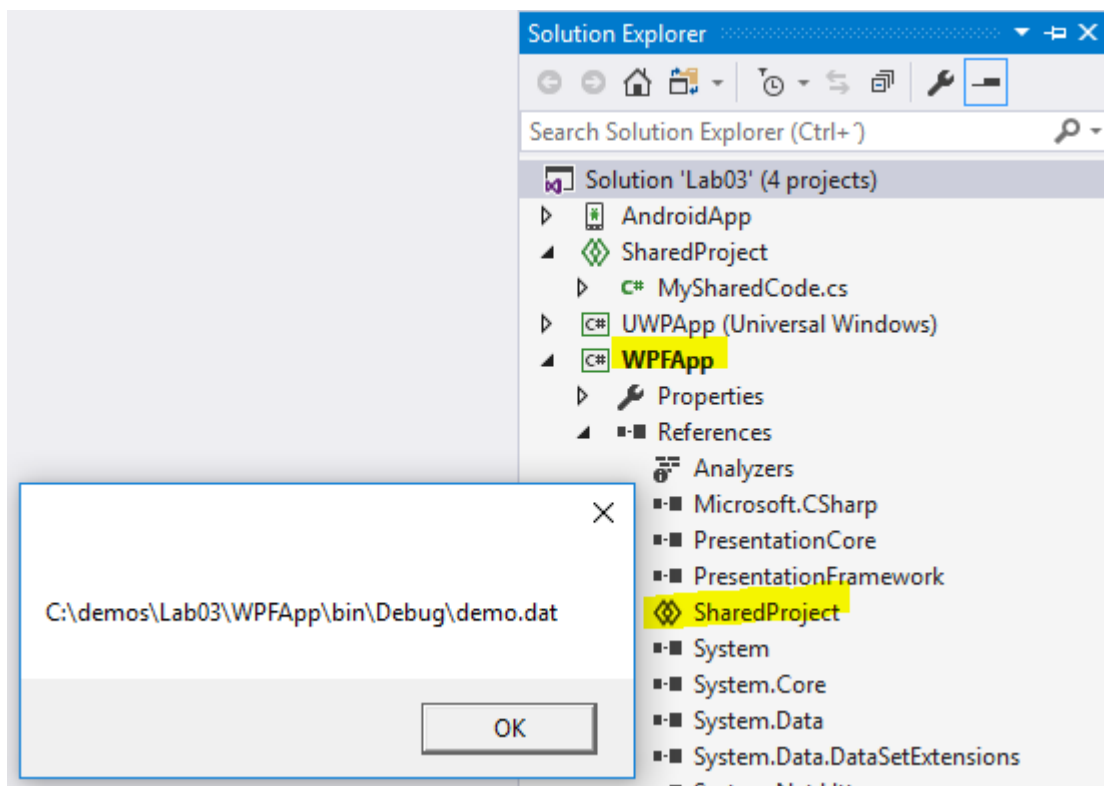
<https://www.facebook.com/groups/iniciandoconxamarin/>

<https://www.facebook.com/groups/xamarindiplomadoitc/>

¿Te gustan los desafíos?

Demuestra lo que aprendiste en este laboratorio realizando la siguiente actividad opcional (seguro aprenderás más al realizar este desafío).

1. Agrega a la solución un nuevo proyecto *Windows Presentation Foundation* utilizando la plantilla **WPF App (.NET Framework)**.
2. Haz los cambios necesarios en las propiedades del proyecto.
3. Agrega funcionalidad en el método **GetFilePath** del archivo **.cs** compartido.
4. Agrega funcionalidad en el archivo **MainWindow.xaml.cs**.
5. Ejecuta la aplicación. Se debe mostrar un resultado similar al siguiente.



Comparte tu imagen en las redes sociales con el hashtag **#XamarinDiplomadoRetoLab03**



Resumen

En este laboratorio exploraste la forma en que trabajan los Proyectos Compartidos comúnmente conocidos como **Shared Asset Projects** o **SAP**.

¿Qué te pareció este laboratorio?

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