Laboratorio

Localización de Aplicaciones

Versión: 1.0.0
Mayo de 2017

Miguel Muñoz Serafín
@msmdotnet

CONTENIDO

Introducción

La *Localización de una Aplicación* es la acción de proporcionar recursos alternativos destinados a una región o entorno local específico. Por ejemplo, podríamos proporcionar valores cadena localizados con el lenguaje de varios países o incluso cambiar colores o diseños para que coincidan con culturas particulares. Android carga y utiliza en tiempo de ejecución los recursos apropiados para la configuración regional del dispositivo sin la necesidad de realizar algún cambio en el código fuente.

Este laboratorio presenta una introducción a las principales características de localización de Android y la forma de acceder a ellas desde Xamarin.

Objetivos

Al finalizar este laboratorio, los participantes serán capaces de:

- Utilizar recursos alternativos para localizar aplicaciones Android.
- Obtener cadenas localizadas mediante código.
- Colocar cadenas localizadas en los diseños de interfaz de usuario.
- Utilizar Assets en aplicaciones 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: Localizando aplicaciones Android

La estrategia de localización de Android tiene los siguientes puntos clave:

- El directorio de recursos que contiene cadenas, imágenes y otros recursos localizados.
- El Método ***GetText*** que es utilizado para obtener cadenas localizadas mediante código.
- ***@string/id*** en archivos AXML para colocar automáticamente cadenas localizadas en los diseños de interfaz de usuario.

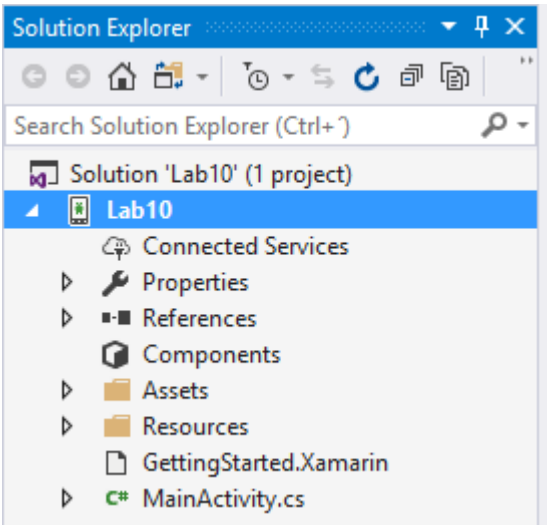
En este ejercicio explorarás los elementos fundamentales de Android para la internacionalización de aplicaciones.

Tarea 1. Agregar recursos para localización.

En esta tarea crearás una nueva aplicación Android y agregarás recursos para que la interfaz de usuario se muestre apropiadamente en dispositivos configurados en inglés, español o portugués.

1. Abre Visual Studio bajo el contexto del Administrador.
2. Utiliza la plantilla ***Blank App (Android)*** para crear una solución con un proyecto *Xamarin.Android* llamado **Lab10**.

   El explorador de soluciones deberá mostrar algo similar a lo siguiente.

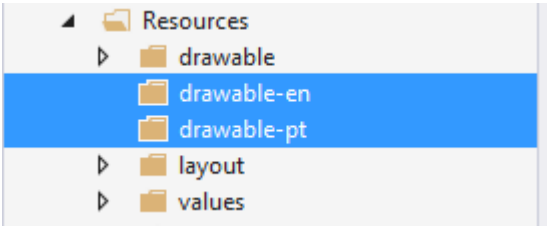


   Las aplicaciones Android administran la mayor parte del contenido en directorios de recursos tales como:

- **layout**. Contiene archivos de diseño AXML.
- **drawable**. Contiene imágenes y otros recursos gráficos.
- **values**. Contiene cadenas de texto.
- **raw**. Contiene archivos de datos.

De la misma forma en que utilizamos sufijos **dpi** en el directorio *drawable* para proporcionar múltiples versiones de una imagen y dejar que Android seleccione la versión correcta para cada dispositivo, este mismo mecanismo es utilizado para proporcionar múltiples traducciones de lenguaje agregando sufijos a los directorios de recursos con los identificadores de lenguaje y cultura.

3. Dentro del directorio ***Resources*** agrega los siguientes 2 directorios ***drawable*** que nos permitirán almacenar imágenes destinadas a dispositivos configurados en inglés o portugués. El directorio ***Resources*** será similar al siguiente.



El directorio de recursos predeterminado (***drawable***) tendrá las imágenes que serán utilizadas cuando el dispositivo no esté configurado en inglés o portugués. En nuestro laboratorio, el lenguaje predeterminado de la aplicación será español y pondremos las imágenes en español en el directorio predeterminado *drawable*.

4. Agrega al directorio *Resources\drawable* el archivo *drawable\XamarinDiplomado.jpg* que se encuentra adjunto a este documento. Esta es la versión en español de la imagen.

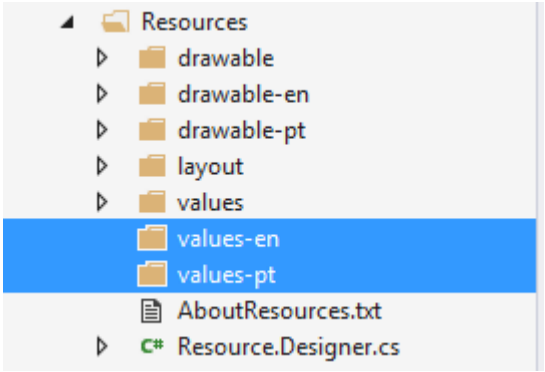
5. Agrega al directorio *Resources\drawable-en* el archivo *drawable-en\XamarinDiplomado.jpg* que se encuentra adjunto a este documento. Esta es la versión en inglés de la imagen.

6. Agrega al directorio *Resources\drawable-pt* el archivo *drawable-pt\XamarinDiplomado.jpg* que se encuentra adjunto a este documento. Esta es la versión en portugués de la imagen.

De la misma forma en que podemos *Localizar* imágenes en directorios *drawable*, también podemos localizar cadenas de texto en directorios *values*. Un directorio *values* localizado debe contener un archivo llamado *Strings.xml* que contendrá el texto traducido para una región.

7. Dentro del directorio *Resources* agrega un directorio llamado ***values-en*** y un directorio llamado ***values-pt***. Estos directorios almacenarán el archivo *Strings.xml* con cadenas de texto en inglés y portugués respectivamente.

   La estructura del directorio *Resources* será similar a la siguiente.



8. Abre el archivo *values\Strings.xml*. Este archivo contendrá los mensajes para regiones que no sean en inglés o en portugués.

9. Modifica el archivo *Strings.xml* para que el contenido sea similar al siguiente.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="ApplicationName">Localización en Android</string>
    <string name="ButtonText">Hola Mundo, hazme Clic!</string>
    <string name="ContentHeader">Contenido del curso (español)</string>
</resources>
```

   Puedes notar que cada cadena traducida es un elemento XML con un ID de recurso especificado con el atributo ***name*** y la cadena traducida con el atributo ***valor***.

10. Dentro del directorio ***values-en***, utiliza la plantilla ***XML File*** para agregar un nuevo elemento llamado *Strings.xml*.

11. Tomando en cuenta los recursos de texto en español, agrega ahora las cadenas traducidas al inglés. El contenido será similar al siguiente.

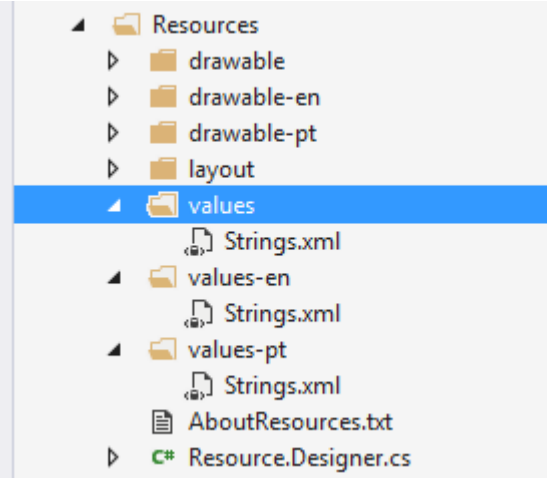
```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
  <string name="ApplicationName">Android Localization</string>
  <string name="ButtonText">Hello World, Click Me!</string>
  <string name="ContentHeader">Course content (spanish)</string>
</resources>
```

12. Dentro del directorio ***values-pt***, utiliza la plantilla ***XML File*** para agregar un nuevo elemento llamado *Strings.xml*.

13. Tomando en cuenta los recursos de texto en español, agrega ahora las cadenas traducidas al portugués. El contenido será similar al siguiente.

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
  <string name="ApplicationName">Localização no Android</string>
  <string name="ButtonText">Olá Mundo, Clique-me!</string>
  <string name="ContentHeader">Conteúdo do curso (espanhol)</string>
</resources>
```

La estructura de directorios se verá de la siguiente forma.



14. Guarda todos los cambios realizados.

Tarea 2. Acceder a los recursos localizados.

Una vez que los recursos han sido agregados, es posible hacer referencia a ellos desde los archivos de diseño AXML o mediante código.

1. Abre el archivo *Main.axml*.

2. Agrega un elemento ***ImageView*** al diseño.

3. En los archivos de diseño, para referenciar una imagen localizada utilizamos la sintaxis *@drawable/id*. Por lo tanto, para mostrar una imagen en el elemento ***ImageView***, agrega el siguiente valor a su propiedad ***src****: @drawable/xamarindiplomado*

4. Agrega un elemento ***Button*** al diseño.

5. Agrega el siguiente valor a la propiedad **text** del botón*: @string/ButtonText*. En los archivos de diseño, para referenciar una cadena localizada utilizamos la sintaxis *@string/id*.

6. Agrega el siguiente valor a la propiedad **id** del botón: *@+id/ClickMe*

15. Agrega un elemento ***Text (Large)*** al diseño y establece su propiedad ***id*** con el valor ***@+id/ClickCounter.*** En este elemento estableceremos el número de veces que el usuario haya dado clic en el botón.

16. Agrega un elemento ***Text (Large)*** al diseño y establece su propiedad ***id*** con el valor ***@+id/ContentHeader.*** Estableceremos el valor de este elemento mediante código.

17. Guarda los cambios realizados.

18. Abre el archivo *MainActivity.cs*.

19. Modifica el valor del parámetro *Label* para que muestre el valor del recurso *ApplicationName* localizado. El código será similar al siguiente.

```
[Activity(Label = "@string/ApplicationName", MainLauncher = true, Icon =
"@drawable/icon")]
```

20. Dentro del método **OnCreate**, quita el comentario a la línea de código que establece la vista de la actividad.

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

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

21. Para obtener las cadenas traducidas desde código, podemos utilizar el método ***GetText*** o el método ***GetString*** proporcionando el ***id*** del recurso cadena. A diferencia de *GetString*, el método *GetText* nos permite trabajar con texto enriquecido (con formato).

    Agrega el siguiente código para asignar al elemento ***ContentHeader*** el valor del recurso *string/CounterHeader*.

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

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

    var ContentHeader = FindViewById<TextView>(Resource.Id.ContentHeader);
    ContentHeader.Text = GetText(Resource.String.ContentHeader);
}
```

22. Guarda los cambios.

23. Configura tu emulador o dispositivo para una región en español.

24. Ejecuta la aplicación. Te deberá mostrar la versión en español de la aplicación.

25. Configura tu emulador o dispositivo para un lenguaje inglés. Al regresar a la aplicación te será mostrada la versión en inglés.

26. Configura tu emulador o dispositivo para un lenguaje portugués. Al regresar a la aplicación te será mostrada la versión en portugués.

Los recursos cadena de Android también nos permiten crear cadenas de cantidad (Quantity strings) para poder proporcionar traducciones para diferentes cantidades como:

Inglés	Español	Portugués
There is 1 Click	Hay 1 Clic	Há 1 Clique
There are 2 Clicks	Hay 2 Clicks	Há 2 Cliques

Esto evita mostrar algo genérico como “There are n Clic(s)”

27. Agrega el siguiente recurso cadena de cantidad al archivo *values\Strings.xml*.

```
<plurals name="numberOfClicks">
  <item quantity="one">Hay %d Clic.</item>
  <item quantity="other">Hay %d Clics.</item>
</plurals>
```

Las opciones de cadenas de cantidad son: *zero*, *one*, *two*, *few*, *many*, *other*. Como desarrolladores, siempre deberíamos proporcionar cadenas “*one*” y “*other*”.

28. Agrega el siguiente recurso cadena de cantidad al archivo *values-en\Strings.xml*.

```
<plurals name="numberOfClicks">
  <item quantity="one">There is %d Clic.</item>
  <item quantity="other">There are %d Clics.</item>
</plurals>
```

29. Agrega el siguiente recurso cadena de cantidad al archivo *values-pt\Strings.xml*.

```
<plurals name="numberOfClicks">
  <item quantity="one">Há %d Clique.</item>
  <item quantity="other">Há %d Cliques.</item>
</plurals>
```

30. Agrega el siguiente código a la clase *MainActivity* para declarar una variable que permita llevar un contador de los clics que el usuario realice.

```
public class MainActivity : Activity
{
    int Counter = 0;

    protected override void OnCreate(Bundle bundle)
```

31. Agrega el siguiente código al método *OnCreate* para incrementar el contador de clics y mostrar la información al usuario.

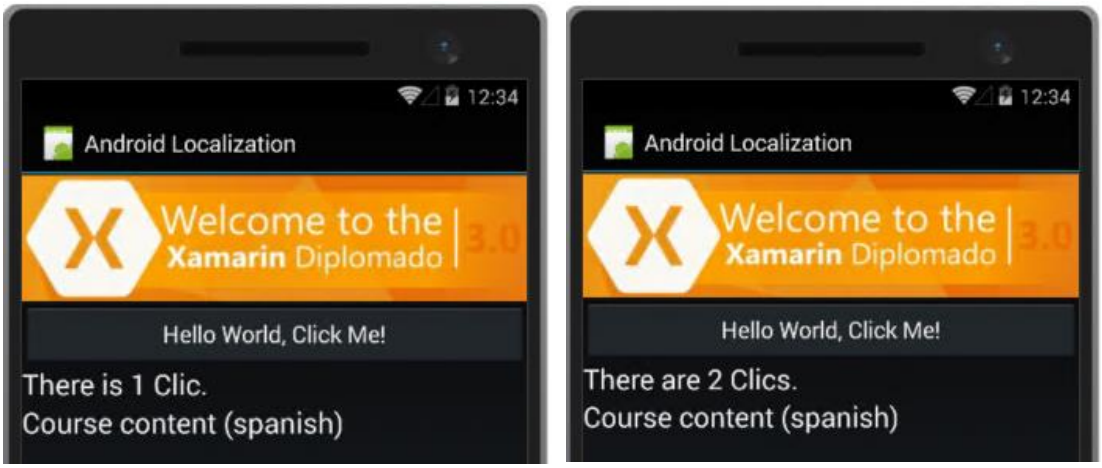
```
var ClickMe = FindViewById<Button>(Resource.Id.ClickMe);
var ClickCounter = FindViewById<TextView>(Resource.Id.ClickCounter);
ClickMe.Click += (s, e) =>
  {
    Counter++;
    ClickCounter.Text = Resources.GetQuantityString(
          Resource.Plurals.numberOfClicks, Counter, Counter);
  };
```

Observa que para obtener el valor de la cadena de cantidad utilizamos el método *GetQuantityString* pasándole el ID de recurso, el valor a mostrar y el valor que le permitirá a Android determinar cuál cadena de cantidad utilizar. En este ejemplo, el valor a mostrar y el valor para determinar la cadena a mostrar son el mismo (Counter).

32. Ejecuta la aplicación. Haz clic en botón y observa el mensaje que se muestra.

33. Haz clic una segunda vez y observa el mensaje que ahora muestra.

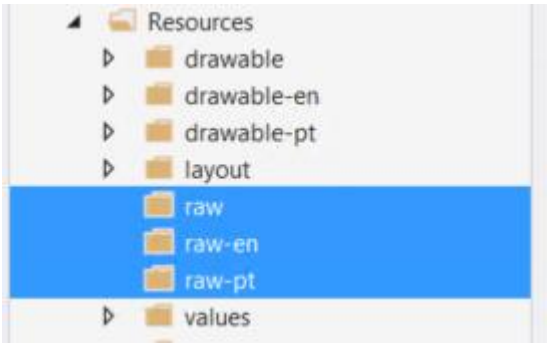
34. Cambia la configuración a inglés y observa el resultado.



35. Cambia la configuración a español y observa el resultado.

También es posible proporcionar otros tipos de recursos alternativos específicos del lenguaje incluyendo archivos de diseño (layouts), animaciones o archivos planos (raw). Esto significa que podríamos proporcionar un diseño de pantalla específico para uno o más de los lenguajes destino o proporcionar archivos de audio específico para uno o más de los lenguajes destino.

36. Dentro del directorio ***Resources*** agrega los siguientes 3 directorios ***raw*** que nos permitirán almacenar archivos mp3 destinados a dispositivos configurados en español, inglés o portugués. El directorio ***Resources*** será similar al siguiente.



El directorio de recursos predeterminado (***raw***) tendrá el archivo de sonido que será utilizado cuando el dispositivo no esté configurado en inglés o portugués. En nuestro laboratorio, el lenguaje predeterminado de la aplicación será español y pondremos el archivo mp3 en español en el directorio predeterminado *raw*.

37. Agrega al directorio *Resources\raw* el archivo *raw\sound.mp3* que se encuentra adjunto a este documento. Esta es la versión en español del sonido.

38. Agrega al directorio *Resources\raw-en* el archivo *raw-en\sound.mp3* que se encuentra adjunto a este documento. Esta es la versión en inglés del sonido.

39. Agrega al directorio *Resources\rawe-pt* el archivo *raw-pt\sound.mp3* que se encuentra adjunto a este documento. Esta es la versión en portugués del sonido.

40. Agrega el siguiente código dentro del manejador del evento *Click* del botón *ClickMe*. Este código permitirá reproducir un archivo mp3 configurado como recurso *raw*.

```
var ClickCounter = FindViewById<TextView>(Resource.Id.ClickCounter);
ClickMe.Click += (s, e) =>
  {
    Counter++;
    ClickCounter.Text = Resources.GetQuantityString(
          Resource.Plurals.numberOfClicks, Counter, Counter);

    var Player = Android.Media.MediaPlayer.Create(
        this, Resource.Raw.sound);
    Player.Start();

  };
```

41. Ejecuta la aplicación y haz clic en el botón. Podrás escuchar el sonido en el lenguaje actual de tu dispositivo.

42. Cambia el lenguaje de tu dispositivo para que puedas escuchar el sonido de cada lenguaje.

Ejercicio 2: Utilizando Assets

Los *Assets* proporcionan una forma de incluir archivos arbitrarios como texto, xml, fonts, música y vídeo en una aplicación. Si intentamos incluir esos archivos como "recursos", Android los procesará en su sistema de recursos y no podremos obtener los datos planos (raw). Si deseamos acceder a los datos intactos, los *Assets* son una forma de hacerlo.

Los *Assets* agregados a un proyecto aparecerán como un sistema de archivos que podemos leer desde la aplicación utilizando *AssetManager*.

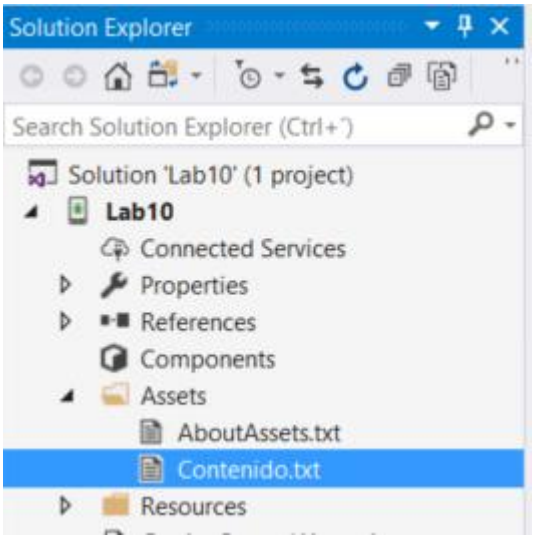
En este ejercicio agregaremos un archivo de texto a nuestro proyecto, lo leeremos utilizando *AssetManager* y lo mostraremos en un *TextView*.

Tarea 1. Agregar un Asset al proyecto.

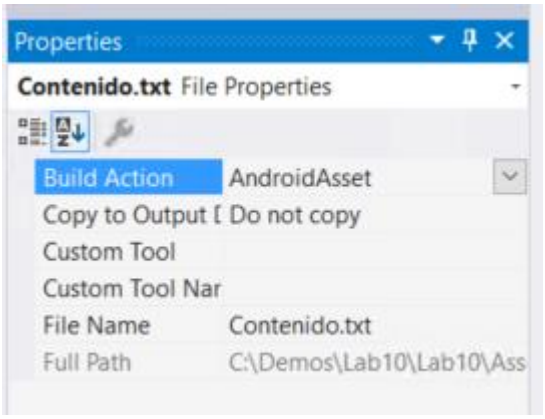
Los *Assets* se almacenan en el directorio *Assets* del proyecto.

En esta tarea agregarás un archivo de texto como un *Asset* de la aplicación Android.

1. Agrega al directorio *Assets* el archivo *Contenido.txt* adjunto a este documento.



2. Presiona **F4** sobre el archivo *Contenido.txt* para mostrar la ventana de propiedades.



Observa que el valor de la propiedad *Build Action* es ***AndroidAsset***. Visual Studio establece ese valor para garantizar que el archivo será empaquetado dentro del APK en tiempo de compilación.

Tarea 2. Leer un Asset.

Los *Assets* se leen utilizando el *AssetManager*. Una instancia de *AssetManager* está disponible mediante el acceso a la propiedad *Assets* de una *Android.App.Context*, tal como una *Activity*. En esta tarea, abriremos el asset *Contenido.txt*, leeremos su contenido y lo mostraremos utilizando un *TextView*.

1. Agrega el siguiente código al final del método *OnCreate*. El código utiliza la clase *Assetmanager* para leer el contenido del asset *Contenido.txt*.

```
Android.Content.Res.AssetManager Manager = this.Assets;

using (var Reader =
    new System.IO.StreamReader(Manager.Open("Contenido.txt")))
{
    ContentHeader.Text += $"\n\n{Reader.ReadToEnd()}";
}
```

2. Ejecuta la aplicación. Podrás ver una pantalla similar la siguiente.


02:36
Localización en Android
Bienvenidos al
Xamarin Diplomado
3.0
Hola Mundo, hazme Clic!
Large Text
Contenido del curso (español)
Este entrenamiento proporciona a los participantes los conocimientos necesarios para desarrollar aplicaciones multiplataforma utilizando Xamarin.

Ejercicio 3: 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 **SALLab10.dll** que implementa la capa de acceso a servicio. El archivo **SALLab10.dll** se encuentra disponible junto con este documento.

1. En el proyecto Xamarin.Android, agrega una referencia del ensamblado **SALLab10.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. En el proyecto Xamarin.Android, instala el paquete NuGet **Microsoft.Azure.Mobile.Client.**

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. Agrega la funcionalidad necesaria para que, al lanzar tu aplicación, se muestre una pantalla solicitando tus credenciales.

2. Agrega la funcionalidad necesaria para que después de ser autenticado, muestre los datos en la pantalla.

La pantalla deberá mostrar los textos de la interfaz de usuario en español, inglés y portugués dependiendo de la configuración del dispositivo.

Las siguientes imágenes son un ejemplo de lo que tu aplicación debe mostrar.

Configuración Español

Configuración en Inglés

Configuración en Portugués

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

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

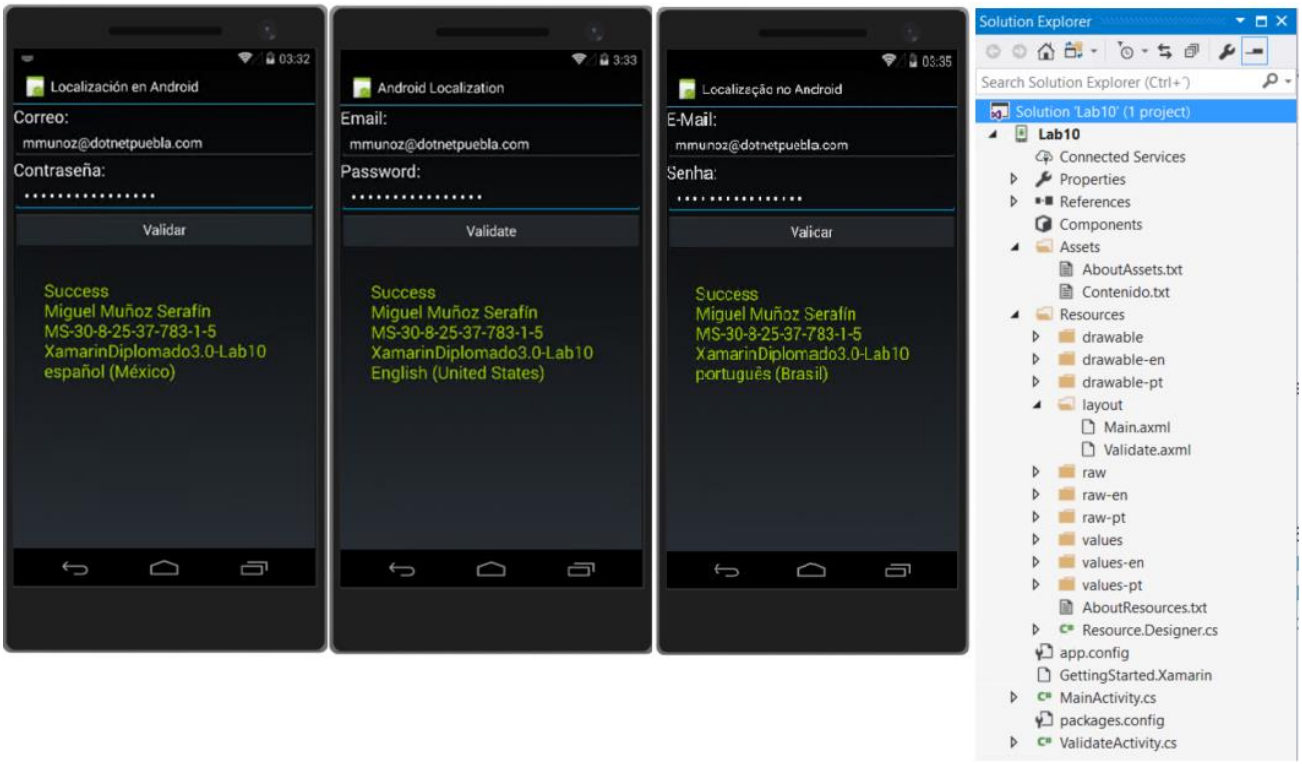
> ***Tu código de lab no es válido, revisa que estés utilizando un código de reto válido. Si tienes preguntas o dudas por favor contacta a dxaudmx@microsoft.com***

Puedes hacer caso omiso al mensaje.

Para verificar que hayas realizado tu actividad, sube una imagen mostrando lo siguiente:

a) La imagen de la pantalla en español.
b) La imagen de la pantalla en inglés.
c) La imagen de la pantalla en portugués.
d) La estructura de directorios del folder **Assets**, **Resources** y **layout**.

La imagen deberá ser similar a la siguiente:



La evidencia donde debes subir la imagen es "**Imagen laboratorio 10**".

Imagen laboratorio 10	No ha enviado evidencia
Fecha Límite: 30 de Junio de 2017	

Los puntos que se evalúan en la evidencia son los siguientes:

- La imagen debe mostrar la pantalla en español.
- La imagen debe mostrar la pantalla en inglés.
- La imagen debe mostrar la pantalla en portugués.
- La imagen debe mostrar la estructura de directorios del folder **Assets**, **Resources** y **layout**.

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/

Resumen

En este laboratorio describimos el uso de recursos Android desde Xamarin.Android. Se presentó una introducción a los recursos predeterminados y alternativos, así como la forma en que pueden ser utilizados para dar soporte a la localización de aplicaciones.

¿Qué te pareció este laboratorio?

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