Laboratorio

Manejando Recursos Alternativos

Versión: 1.0.0
Mayo de 2017

Miguel Muñoz Serafín
@msmdotnet

CONTENIDO

Introducción

Los recursos alternativos son aquellos recursos orientados a un dispositivo o configuración especifica en tiempo de ejecución, por ejemplo, el idioma actual, el tamaño de una pantalla en particular o la densidad de píxeles. Si Android puede detectar un recurso que sea más específico que el recurso predeterminado para un dispositivo o configuración particular, entonces ese recurso alternativo será utilizado en su lugar. Si Android no encuentra un recurso alternativo que coincida con la configuración actual, se cargarán entonces los recursos predeterminados.

En este laboratorio conocerás la forma en que Android elije los recursos alternativos para que puedas desarrollar aplicaciones que soporten diferentes resoluciones y densidades de pantalla, así como como el soporte de las diferentes regiones geográficas en que una aplicación podría ser utilizada.

Objetivos

Al finalizar este laboratorio, los participantes serán capaces de:

- Describir el propósito de los recursos alternativos.
- Describir la forma en que Android selecciona los recursos alternativos.

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: Trabajando con Recursos Alternativos

De la misma forma en que los recursos predeterminados son organizados, los recursos alternativos también se organizan según el tipo de recurso, en subdirectorios dentro de la carpeta *Resources*. El nombre del subdirectorio de recursos alternativos tiene la siguiente sintaxis:

<TipoDeRecurso>-<Calificador>

Calificador es un nombre que identifica una configuración de dispositivo específica. Puede haber más de un calificador en un nombre, cada uno de ellos separados por un guion.

Las siguientes reglas se aplican al agregar calificadores a un tipo de recurso:

1. Puede haber más de un calificador, con cada calificador separado por un guion.
2. Los calificadores pueden especificarse sólo una vez.
3. Los calificadores deben estar en el orden en que aparecen en la tabla de calificadores.

Calificador	Descripción
MCC y **MNC**	El código de país móvil (Mobile Country Code) y opcionalmente el código de red móvil (Mobile Network Code). La tarjeta SIM proporciona el MCC mientras que la red a la que está conectado el dispositivo proporciona el MNC. Aunque es posible segmentar ubicaciones utilizando el código de país móvil, el enfoque recomendado es utilizar el calificador de *Lenguaje*. Por ejemplo, para destinar recursos a Alemania, el calificador sería **mcc262**. Para orientar los recursos para T-Mobile en los Estados Unidos, el calificador es **mcc310-mnc026**. Para ver la lista completa de códigos de países móviles y códigos de redes móviles, puedes consultar el siguiente enlace: http://mcc-mnc.com/.
Lenguaje	El código de lenguaje ISO 639-1 formado por dos letras y opcionalmente seguido por el código de región ISO-3166-alpha-2 también de dos letras. Si ambos calificadores son proporcionados, entonces deben estar separados por **-r**. Por ejemplo, para destinar la aplicación a las regiones donde se habla francés utilizamos el calificador **fr**. Para destinar la aplicación a las regiones Francesas-Canadienses utilizamos ***fr-rCA***.
Ancho más pequeño	Especifica el tamaño de ancho más pequeño de la pantalla del dispositivo en el que se podrá ejecutar la aplicación. Disponible en API nivel 13 (Android 3.2) y superior. Por ejemplo, el calificador *sw320dp* se utiliza para destinar a dispositivos cuya altura y ancho es de al menos 320dp.
Ancho disponible	El ancho mínimo de la pantalla en el formato wNdp, donde N es el ancho en densidad de píxeles independientes. Este valor puede cambiar a medida que el usuario gira el dispositivo. Disponible en el nivel API 13 (Android 3.2) y superior.

	Por ejemplo, el calificador *w720dp* se utiliza para destinar a dispositivos que tienen un ancho de al menos 720dp.
Altura disponible	La altura mínima de la pantalla en el formato hNdp, donde N es la altura en dp. Este valor puede cambiar a medida que el usuario gira el dispositivo. Disponible en el nivel API 13 (Android 3.2) y superior. Por ejemplo, el calificador *h720dp* se utiliza para destinar a dispositivos que tienen una altura de al menos 720dp.
Tamaño de Pantalla	Este calificador es una generalización del tamaño de pantalla para la que están destinados los recursos. Los valores posibles son *small*, *normal*, *large* y *xlarge*. Disponible desde el nivel de API 9 (Android 2.3 / Android 2.3.1 / Android 2.3.2).
Aspecto de pantalla	Esto se basa en la relación de aspecto, no en la orientación de la pantalla. Una pantalla larga es más ancha. Añadido en API nivel 4 (Android 1.6). Los valores posibles son *long* y *notlong*.
Orientación de la pantalla	Orientación de pantalla Portrait (vertical) o Landscape (horizontal). Esto puede cambiar durante el tiempo de vida de una aplicación. Los posibles valores son *port* y *land*.
Modo Dock	Para dispositivos en un *Car Dock* o en un *Desk Dock*. Adicionado en el nivel API 8 (Android 2.2.x). Los posibles valores son *car* y *desk.*
Modo Nocturno	Para cuando la aplicación este ejecutándose en la noche o en el día. Esto puede cambiar durante la vida de una aplicación y tiene la intención de dar a los desarrolladores la oportunidad de utilizar versiones más oscuras de una interfaz de usuario por la noche. Disponible desde el nivel API 8 (Android 2.2.x). Los valores posibles son *night* y *notnight*.
Densidad de pixeles en pantalla (dpi)	El número de pixeles en un área física de la pantalla. Típicamente es expresado en puntos por pulgada (dpi). Algunos de los posibles valores son: • **ldpi**. Para pantallas de baja densidad. • **mdpi**. Para pantallas de mediana densidad. • **hdpi**. Para pantallas de alta densidad. • **xhdpi**. Para pantallas de extra alta densidad. • **nodpi**. Recursos que no deben ser escalados. • **tvdpi**. Introducido en nivel api 13 (android 3.2) para pantallas entre mdpi y hdpi.
Tipo de pantalla táctil	Especifica el tipo de pantalla táctil que un dispositivo puede tener. Los posibles valores son *notouch* (pantallas no táctiles), *stylus* (pantallas táctiles adecuadas para lápiz óptico) y *finger* (pantallas touch).
Disponibilidad de teclado	Especifica qué tipo de teclado está disponible. Esto puede cambiar durante la vida de una aplicación, por ejemplo, cuando un usuario abre un teclado de hardware. Los valores posibles son:

	• **keysexposed**. El dispositivo tiene un teclado disponible. Si no hay teclado de software habilitado, entonces esto sólo se utiliza cuando se abre el teclado de hardware. • **keyshidden**. El dispositivo tiene un teclado de hardware, pero está oculto y no está habilitado ningún teclado de software. • **keyssoft**. El dispositivo tiene un teclado de software habilitado
Método principal de entrada de texto	Se utiliza para especificar qué tipos de teclas de hardware están disponibles para la entrada de datos. Los valores posibles son: • **nokeys**. No hay teclas de hardware para la entrada. • **qwerty**. Hay un teclado *qwerty* disponible. • **12key**. Hay un teclado de hardware 12-key.
Disponibilidad de teclas de navegación	Para cuando la navegación **5-way** o **d-pad** (directional-pad) está disponible. Esto puede cambiar durante el tiempo de vida de la aplicación. Los valores posibles son: • **navexposed**. Las teclas de navegación están disponibles para el usuario. • **navhidden**. Las teclas de navegación no están disponibles.
Método principal de navegación no táctil	El tipo de navegación disponible en el dispositivo. Los valores posibles son: • **nonav**. El único método de navegación es la pantalla táctil. • **dpad**. Un *d-pad* está disponible para la navegación. • **trackball**. El dispositivo tiene *TrackBall* para navegación. • **wheel**. El escenario poco común donde hay una o más *Wheels* direccionales disponibles.
Versión de Plataforma (Nivel de API)	El nivel de API admitido por el dispositivo en el formato *vN*, donde N es el nivel de API al que se está orientando. Por ejemplo, ***v11*** se orientará a un dispositivo de nivel 11 (Android 3.0).

Para una información más completa acerca de los calificadores de recursos, puedes consultar el siguiente enlace:

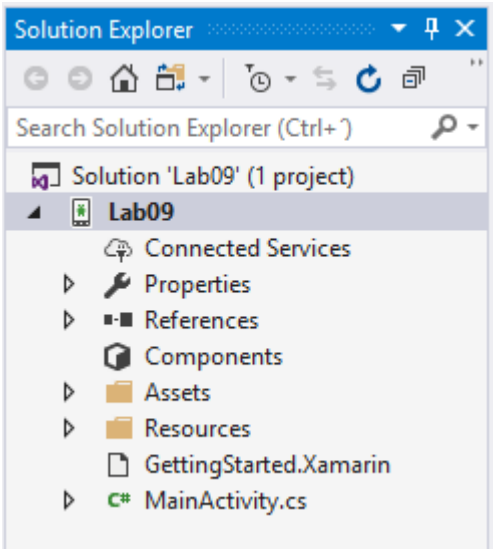
Provisión de recursos
https://developer.android.com/guide/topics/resources/providing-resources.html

Tarea 1. Verificar la forma en que Android determina los recursos a utilizar.

En esta tarea verificarás la forma en que Android determina los recursos que debe utilizar cuando existen recursos alternativos.

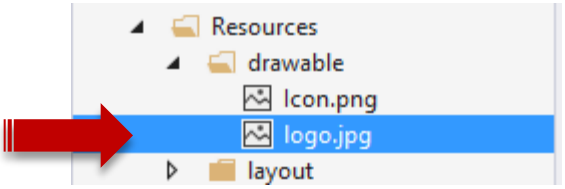
1. Abre Visual Studio bajo el contexto del Administrador.
2. Utiliza la plantilla ***Blank App (Android)*** para crear una solución con un proyecto Xamarin.Android llamado **Lab09**.

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



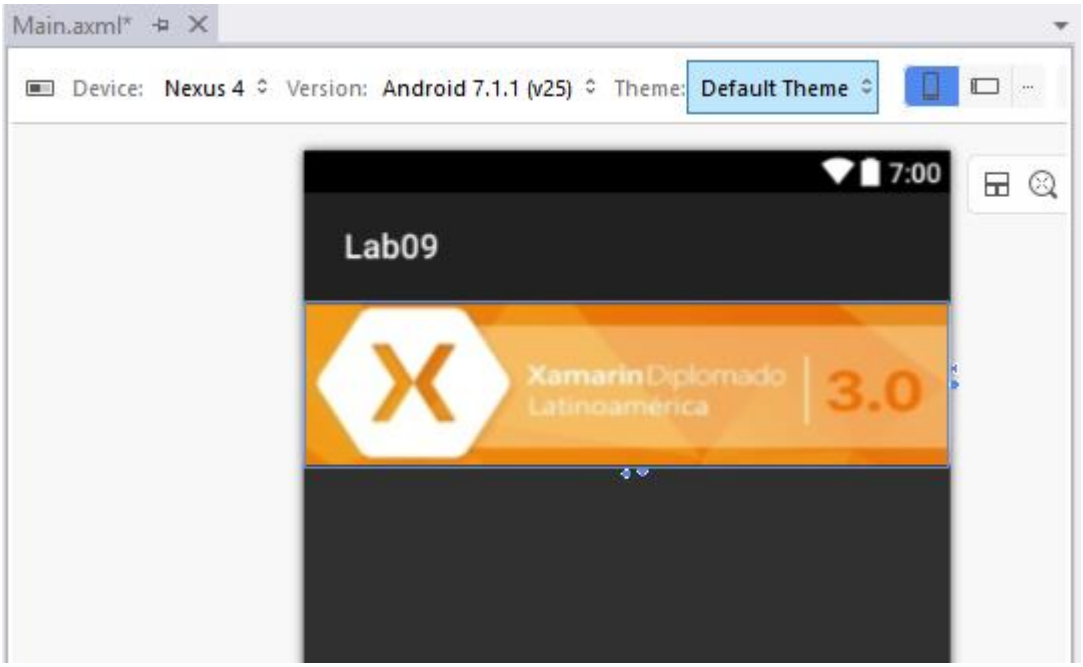
3. Dentro de la carpeta **Resources/drawable** agrega el archivo **logo.jpg** adjunto a este documento.

   El explorador de soluciones debe mostrar la carpeta ***Resources*** de la siguiente forma.



4. Abre el archivo ***Main.axml*** en el Diseñador de Android.
5. Agrega un elemento ***ImageView*** a la superficie de diseño.
6. Asigna el recurso ***drawable/logo*** a la propiedad ***src*** del elemento ***ImageView***. Recuerda que el valor que debes establecer es ***@drawable/logo***.

   El Diseñador de Android se verá de la siguiente manera.



7. Guarda los cambios realizados.
8. Abre el archivo ***MainActivity.cs***.
9. Quita el comentario a la línea que establece el recurso ***Main*** como el contenido de la *Activity*.

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

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

10. Guarda los cambios y ejecuta la aplicación.

    El emulador te mostrará una pantalla similar a la siguiente.

11. Regresa a Visual Studio y detén la ejecución.

Tarea 2. Agregar Recursos Alternativos.

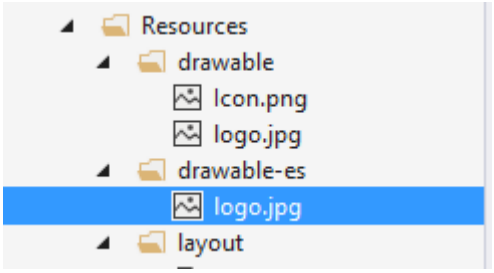
Es muy probable que una aplicación Android contenga muchos recursos, por lo tanto, es importante entender la forma en que Android selecciona los recursos para una aplicación cuando se ejecuta en un dispositivo.

1. Agrega una carpeta dentro del directorio ***Resources*** para almacenar recursos imagen que se utilicen en dispositivos que tengan una configuración regional en español. Recuerda que la sintaxis de los directorios que contienen recursos alternativos es la siguiente:

   <TipoDeRecurso>-<Calificador>

   El tipo de recurso en este caso es ***drawable*** y el Calificador es el código del lenguaje, en este caso "**es"**. Por lo tanto, el directorio que deberás agregar es ***drawable-es***.

2. Dentro del directorio agregado en el paso anterior, agrega el archivo ***1_drawable-es\logo.jpg*** adjunto a este documento. La carpeta se verá similar a lo siguiente.

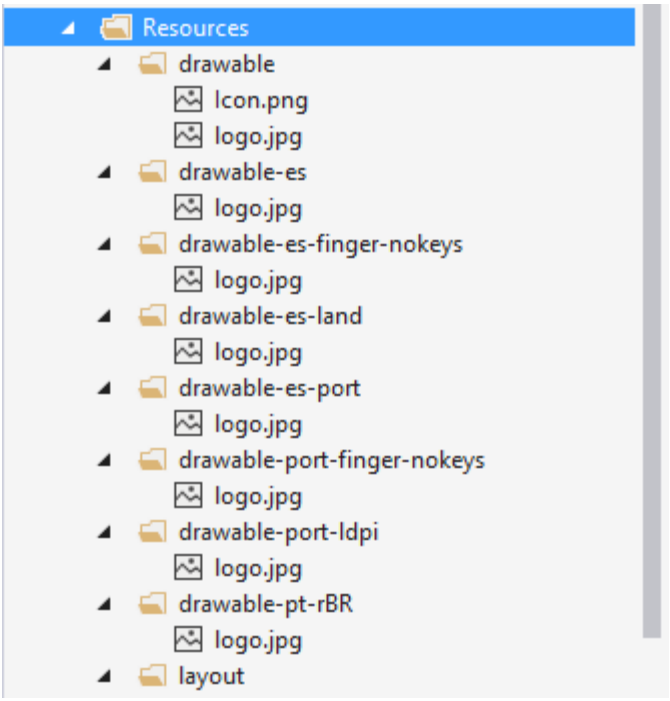


3. Repite los pasos anteriores para agregar los siguientes archivos *logo.jpg* como recursos alternativos de tipo ***drawable*** que Android utilizará dependiendo de la configuración del dispositivo donde se esté ejecutando la aplicación.

Calificadores	Directorio de recursos a crear	Directorio que contiene el archivo *logo.jpg* a agregar	Descripción
pt, rBR	drawable-pt-rBR	2_drawable-pt-rBR	Para dispositivos configurados en portugués de Brasil.
es, land	drawable-es-land	3_drawable-es-land	Para dispositivos configurados en español y con orientación horizontal.
es, port	drawable-es-port	4_drawable-es-port	Para dispositivos configurados en español y con orientación vertical.
es, finger, nokeys	drawable-es-finger-nokeys	5_drawable-es-finger-nokeys	Para dispositivos configurados en español con tipo de pantalla táctil y sin teclas de hardware para entrada de datos.

port, finger, nokeys	drawable-port-finger-nokeys	6_drawable-port-finger-nokeys	Para dispositivos con orientación vertical con pantalla táctil y sin teclas de hardware para entrada de datos.
port, ldpi	drawable-port-ldpi	7_drawable-port-ldpi	Para dispositivos con orientación vertical y pantalla de baja densidad.

Después de agregar los recursos alternativos, el directorio **Resources** será similar a lo siguiente.



4. Tomando como base los recursos que acabas de agregar, para verificar la forma en que Android determina los recursos a utilizar debes realizar los siguientes pasos de esta tarea en un dispositivo o emulador Android con las siguientes características:

    a. **Densidad de pixeles en pantalla:** HDPI. Si lo deseas, puedes utilizar cualquier dispositivo con densidad diferente a LDPI. Las imágenes de este documento corresponden al emulador de Visual Studio para Android 4.5 KitKat (4.4) HDPI Phone API Level 19 (KitKat, 4.4).

b. **Orientación de pantalla:** Vertical. Cuando abras el emulador, asegúrate que esté en posición vertical.

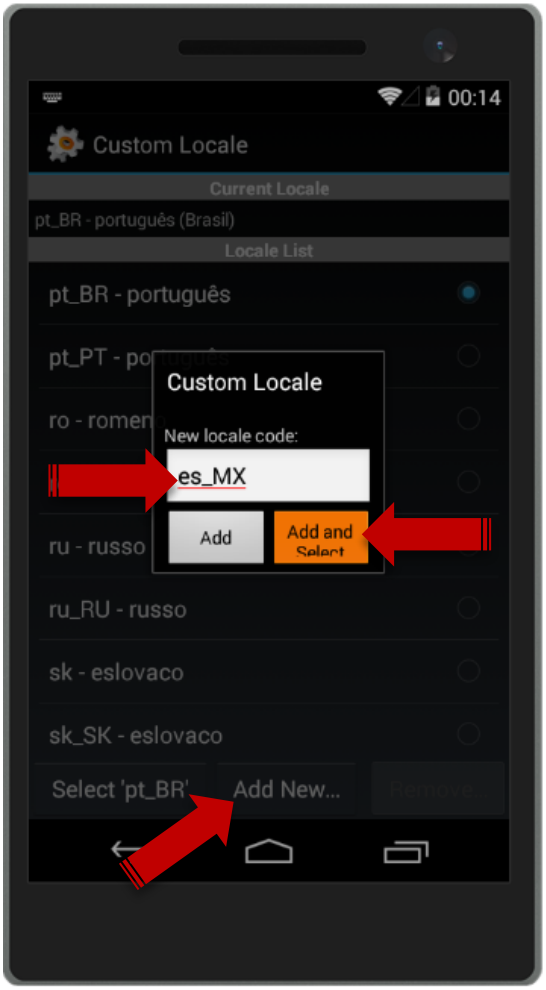
c. **Tipo de pantalla:** Táctil. De manera predeterminada los emuladores funcionan con un tipo de pantalla táctil.

d. **Método principal de entrada:** Sin teclas de hardware. De manera predeterminada los emuladores no tienen habilitada la opción de emular teclas de hardware.

e. **Configuración regional:** Español de México. Si lo deseas, puedes utilizar la configuración regional de cualquier país con lenguaje español.

   En el caso del emulador utilizado en este documento, la aplicación **Custom Locale** del dispositivo Android te permite establecer la configuración regional.

   Por ejemplo, puedes abrir la aplicación **Custom Locale**, hacer clic en **Add New…**, escribir **es_MX** (o el código de tu país con lenguaje español) y hacer clic en **Add and Select.**



Podrás notar que el dispositivo se configurará a español como se muestra en la siguiente imagen.

Ahora la configuración de tu dispositivo cumple con varias especificaciones que podrían coincidir con más de uno de los directorios de recursos *drawable* que agregaste en los pasos anteriores de esta tarea.

Antes de ejecutar la aplicación en el dispositivo, responde a las siguientes preguntas: ***¿De qué directorio de recursos seleccionará Android la imagen que mostrará en el dispositivo? ¿Por qué?***

5. Ejecuta la aplicación. La imagen del directorio de recursos ***drawable-es-port*** será mostrada. ¿Es la imagen que suponías?

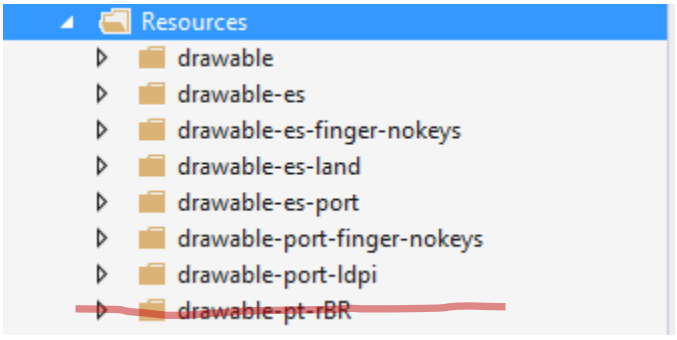
¿Cómo determina Android qué recursos utilizar?

Android determina el recurso a utilizar mediante una iteración y evaluación de las siguientes reglas:

1. **Eliminar Calificadores contradictorios**. Por ejemplo, si la orientación del dispositivo es vertical, todos los recursos del directorio *land* serán ignorados.

   No todos los calificadores están disponibles para todos los niveles de API. Si un directorio de recursos contiene un calificador que no es soportado por el dispositivo, entonces ese directorio de recurso será ignorado.

   En nuestro ejercicio, el directorio **drawable-pt-rBR** es eliminado porque se contradice con la configuración regional actual del dispositivo ***es-MX***.

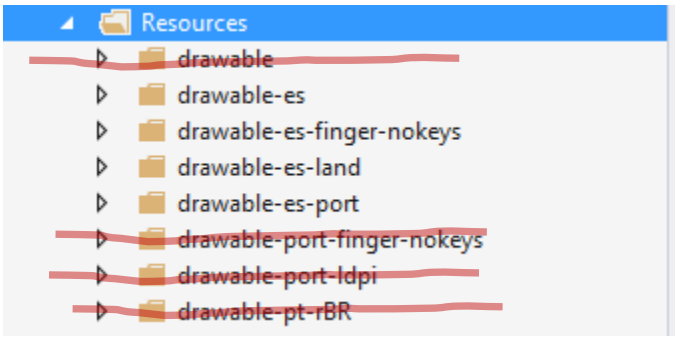


   Excepción: La densidad de píxeles de la pantalla es el único calificador que no se elimina, aunque exista una contradicción de calificadores. Aunque la densidad de la pantalla del dispositivo es **hdpi**, **drawable-port-ldpi** no se elimina porque todas las densidades de pantalla se consideran como una coincidencia en este punto.

2. **Elegir el siguiente Calificador de mayor precedencia.** Haciendo referencia a la tabla anterior, Android selecciona el siguiente calificador de más alta prioridad (Comienza con MCC y continua en forma descendente).
3. **¿Alguno de los directorios de recursos incluye este Calificador?**
    - Si la respuesta es no, volver al paso 2 y examinar el siguiente calificador.
    - Si la respuesta es sí, continuar con el paso 4.

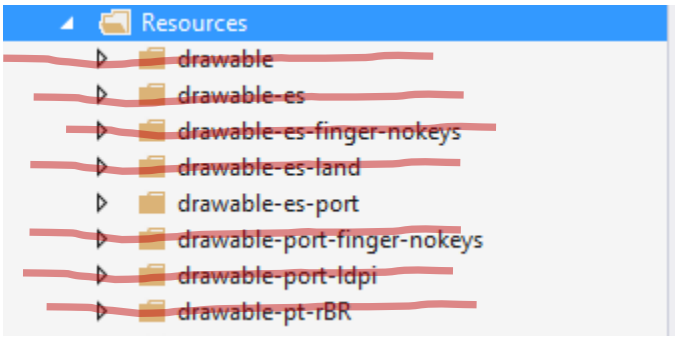
    En nuestro ejercicio, el primer calificador encontrado es el calificador de lenguaje: "**es**".
4. **Eliminar los directorios de recursos que no incluyen este calificador**. Android elimina todos los directorios que no incluyen el Calificador en cuestión.

    En nuestro ejercicio, el sistema elimina todos los directorios que no incluyen un Calificador de idioma.



    Excepción: Si el calificador en cuestión es la densidad de píxeles de la pantalla, Android selecciona la opción que más coincida con la densidad de la pantalla del dispositivo. En general, Android prefiere reducir una imagen original más grande que ampliar una imagen original más pequeña.
5. **Ir al paso 2**. Android vuelve a repetir los pasos 2, 3 y 4 hasta que solo quede un directorio.

    En nuestro ejercicio, según la precedencia de los Calificadores, la orientación de la pantalla Vertical es el próximo Calificador para el cual existen coincidencias. Por lo tanto, se eliminan los recursos que no especifican una orientación de pantalla vertical.



    El directorio que queda es **drawable-es-port**.

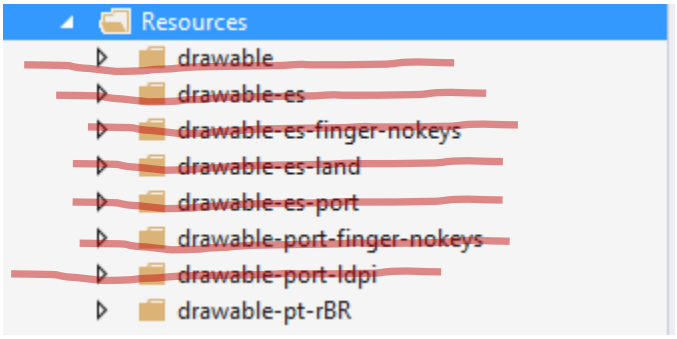
Si el dispositivo se encontrara en orientación horizontal (landscape) entonces el directorio que quedaría sería **drawable-*es-land***.

6. Gira el emulador para que se muestre en forma horizontal. Podrás notar que se muestra la imagen del directorio ***drawable-es-land***.

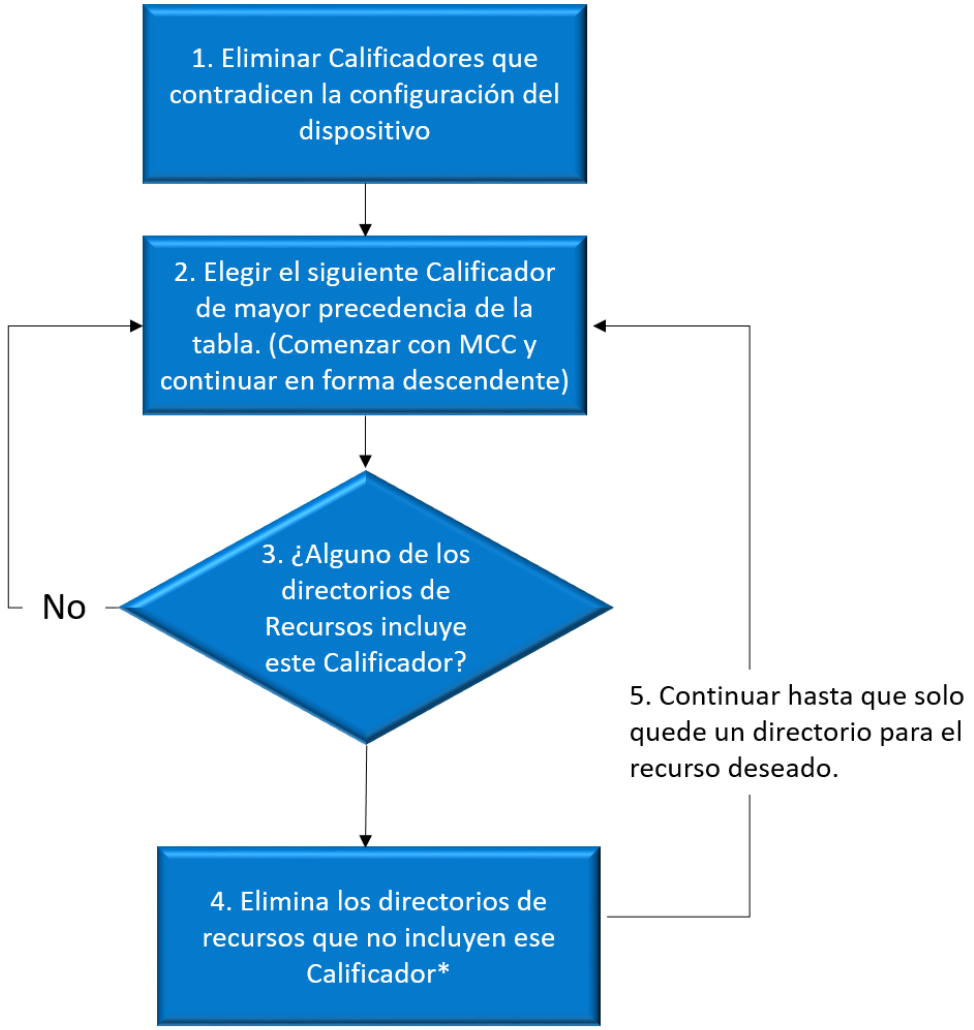
7. Modifica la configuración regional del dispositivo para portugués de Brasil. (pt_BR).

   La aplicación mostrará la siguiente imagen. ¿Por qué?

Esto es debido a que cuando se encuentra la coincidencia del calificador lenguaje pt-rBR, Android elimina todos los directorios que no incluyen un Calificador de idioma pt-rBR.



Estas reglas son ilustradas en el siguiente diagrama de flujo.



* Si el Calificador es la densidad de la pantalla, Android selecciona la opción que más coincida con la densidad de la pantalla del dispositivo.

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 **SALLab09.dll** que implementa la capa de acceso a servicio. El archivo **SALLab09.dll** se encuentra disponible junto con este documento.

1. En el proyecto Xamarin.Android, agrega una referencia del ensamblado **SALLab09.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. Configura el emulador o dispositivo para que utilice el lenguaje español y el código de tu país. Por ejemplo, para el caso de México la configuración regional deberá ser **es-MX**. Puedes consultar el código de tu país en el siguiente enlace: https://www.iso.org/obp/ui/#search

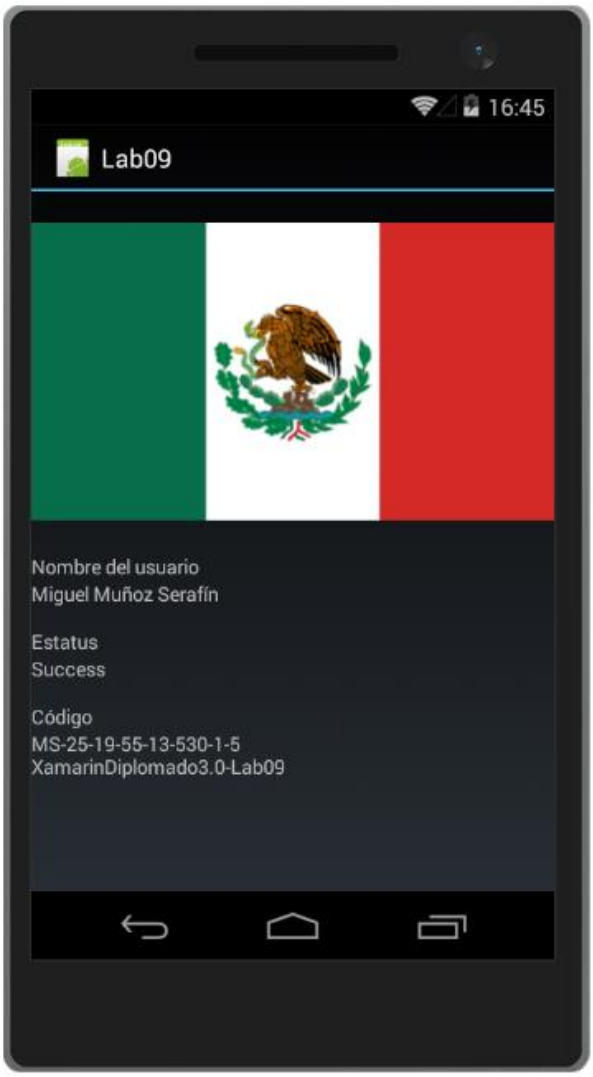
2. Agrega los recursos necesarios para que al ejecutar la aplicación:

   a. En modo vertical se muestre la bandera de tu país y los datos de validación de tu actividad: Nombre, Estatus y Token.

   b. En modo vertical con una configuración regional español distinta de tu país se muestre la imagen predeterminada y los datos de validación de tu actividad: Nombre, Estatus y Token.

   c. En modo horizontal se muestre la imagen predeterminada del laboratorio y los datos de validación de tu actividad: Nombre, Estatus y Token.

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



Configuración de tu pais

Español de otro pais

La misma aplicación muestra la imagen predeterminada cuando se encuentra en modo horizontal.

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.

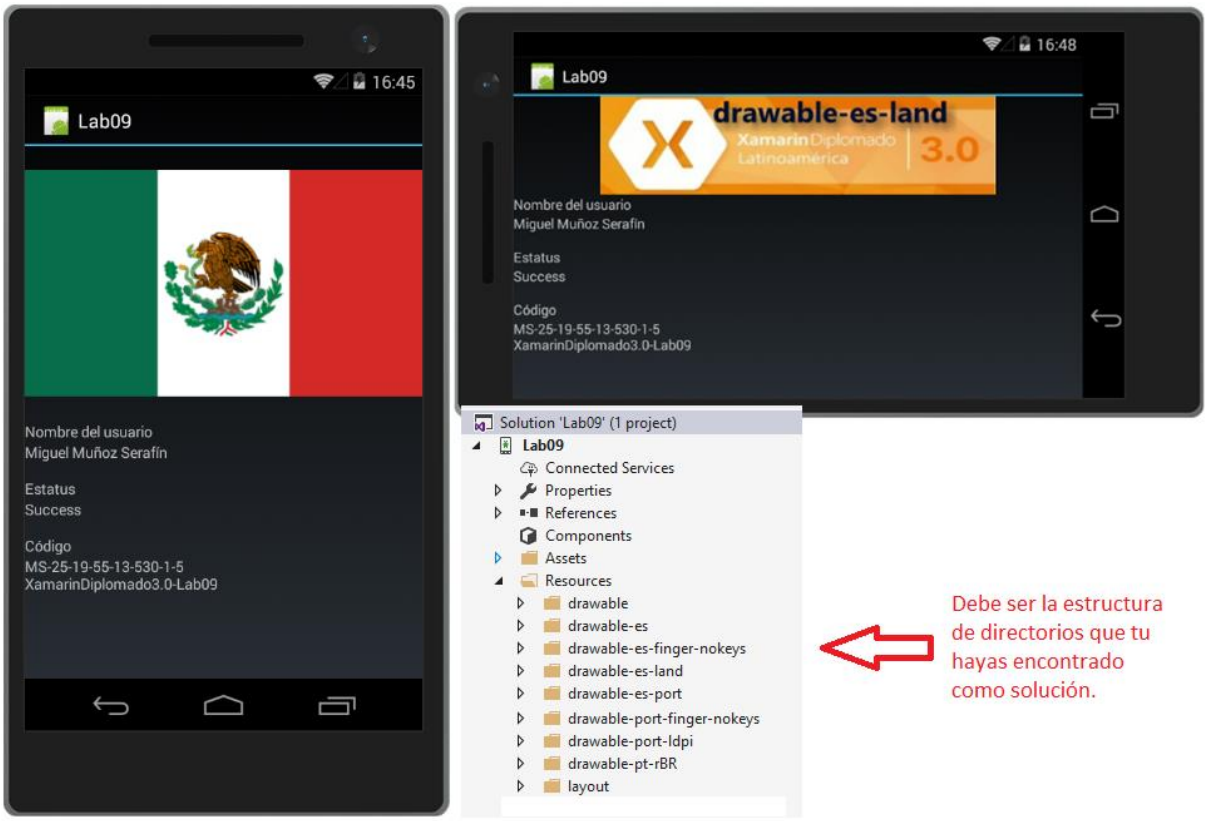
> ***Tu código de lab no es válido, revisa que estés utilizando un código de reto válido. Si tienes preguntas o dudas por favor contacta a dxaudmx@microsoft.com***

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 posición horizontal.
b) La imagen de la pantalla en posición vertical.
c) La estructura de directorios del folder **Resources**.

La imagen deberá ser similar a la siguiente:



La evidencia donde debes subir la imagen es "**Imagen laboratorio 9**".

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

- La imagen debe mostrar la pantalla en posición vertical con la bandera del país.
- La imagen debe mostrar la pantalla en posición horizontal con la imagen predeterminada.
- La imagen debe mostrar la estructura de directorios de la carpeta **Resources** que se haya elegido como solución.

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 exploraste la forma en que Android elije los Recursos Alternativos. Este conocimiento te facilitará la tarea de desarrollar aplicaciones que soporten diferentes resoluciones y densidades de pantalla, así como como el soporte de las diferentes regiones geográficas en que una aplicación podría ser utilizada.

¿Qué te pareció este laboratorio?

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