Desarrollo de aplicaciones móviles I

Tema Nº7:Componentes de Android (Actividades)

Indicador de logro Nº7:Utiliza componentes en una Actividad en Android (Activity) mediante la implementación de una aplicación móvil que integren los elementos más comunes.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº7:**

Componentes de Android (Actividades).

**Subtema 7.1:**

¿Qué es un Activity?

La clase Activity es un componente clave de una app para Android, y la forma en que se inician y se crean las actividades es una parte fundamental del modelo de aplicación de la plataforma. A diferencia de los paradigmas de programación en los que las apps se inician con un método main(), el sistema Android inicia el código en una instancia de Activity invocando métodos de devolución de llamada específicos que corresponden a etapas específicas de su ciclo de vida.

Cómo configurar el manifiesto

Para que tu app pueda usar actividades, debes declararlas, y también declarar algunos de sus atributos, en el manifiesto.

Cómo declarar actividades

Para declarar tu actividad, abre tu archivo de manifiesto y agrega el elemento <activity> como objeto secundario del elemento <application>. Por ejemplo:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

**Cómo administrar el ciclo de vida de la actividad**

A lo largo de su vida útil, una actividad pasa por varios estados. Para administrar las transiciones entre estados, debes usar una serie de devoluciones de llamadas. En las siguientes secciones, se presentan estas devoluciones de llamadas.

**onCreate()**

Debes implementar esta devolución de llamada, que se activa cuando el sistema crea tu actividad. Tu implementación debe inicializar los componentes esenciales de tu actividad. Por ejemplo, tu app debería crear vistas y vincular datos a listas aquí. Principalmente, este es el momento en el que debes llamar a setContentView() para definir el diseño de la interfaz de usuario de tu actividad.

Cuando onCreate() finaliza, la siguiente devolución de llamada siempre es onStart().

**onStart()**

Cuando se cierra onCreate(), la actividad pasa al estado Iniciada y se vuelve visible para el usuario. Esta devolución de llamada contiene los preparativos finales de la actividad para pasar al primer plano y convertirse en interactiva.

**onResume()**

El sistema invoca esta devolución de llamada justo antes de que la actividad comience a interactuar con el usuario. En este punto, la actividad es la primera de la pila de actividades y captura todo lo que el usuario ingresa. La mayor parte de la funcionalidad principal de una app se implementa en el método onResume().

La devolución de llamada onPause() siempre va después de onResume().

**onPause()**

El sistema llama a onPause() cuando la actividad pierde el foco y pasa al estado Detenida. Este estado se produce, por ejemplo, cuando el usuario presiona el botón Atrás o Recientes. Cuando el sistema llama a onPause() para tu actividad, significa que esta aún está parcialmente visible, pero, a menudo, indica que el usuario está saliendo de la actividad y que esta pronto pasará al estado Detenida o Reanudada.

Una actividad en estado Detenida puede continuar con la actualización de la IU si el usuario espera que esta acción ocurra. Entre los ejemplos de tal actividad, se incluye mostrar una pantalla de mapa de navegación o la reproducción de un reproductor multimedia. Incluso si estas actividades pierden el foco, el usuario espera que su IU continúe actualizándose.

No debes usar onPause() para guardar datos de la aplicación o del usuario, realizar llamadas de red o ejecutar transacciones de base de datos. Para obtener más información sobre cómo guardar datos, consulta Cómo guardar y restablecer el estado de la actividad.

Una vez que finalice la ejecución de onPause(), la siguiente devolución de llamada será onStop() o onResume(), dependiendo de lo que ocurra cuando la actividad pase al estado Detenida.

**onStop()**

El sistema llama a onStop() cuando la actividad ya no es visible para el usuario, lo cual puede ocurrir porque se está eliminando la actividad, porque se inicia una nueva o porque una ya existente pasa al estado Reanudada y cubre la actividad que se detuvo. En todos estos casos, la actividad detenida ya no está visible en absoluto.

La siguiente devolución de llamada que el sistema invoca puede ser onRestart() (si la actividad vuelve a interactuar con el usuario) o onDestroy() (si esta actividad finaliza por completo).

**onRestart()**

El sistema invoca esta devolución de llamada cuando una actividad en estado Detenida está por volver a iniciarse. onRestart() restaura el estado de la actividad desde el momento en que esta se detuvo.

Luego de esta devolución de llamada, siempre sigue onStart().

**onDestroy()**

El sistema invoca esta devolución de llamada antes de que se elimine una actividad.

Esta devolución de llamada es la última que recibe la actividad. onDestroy() suele implementarse a fin de garantizar que todos los recursos de una actividad se liberen cuando esta, o el proceso que la contiene, se elimina.

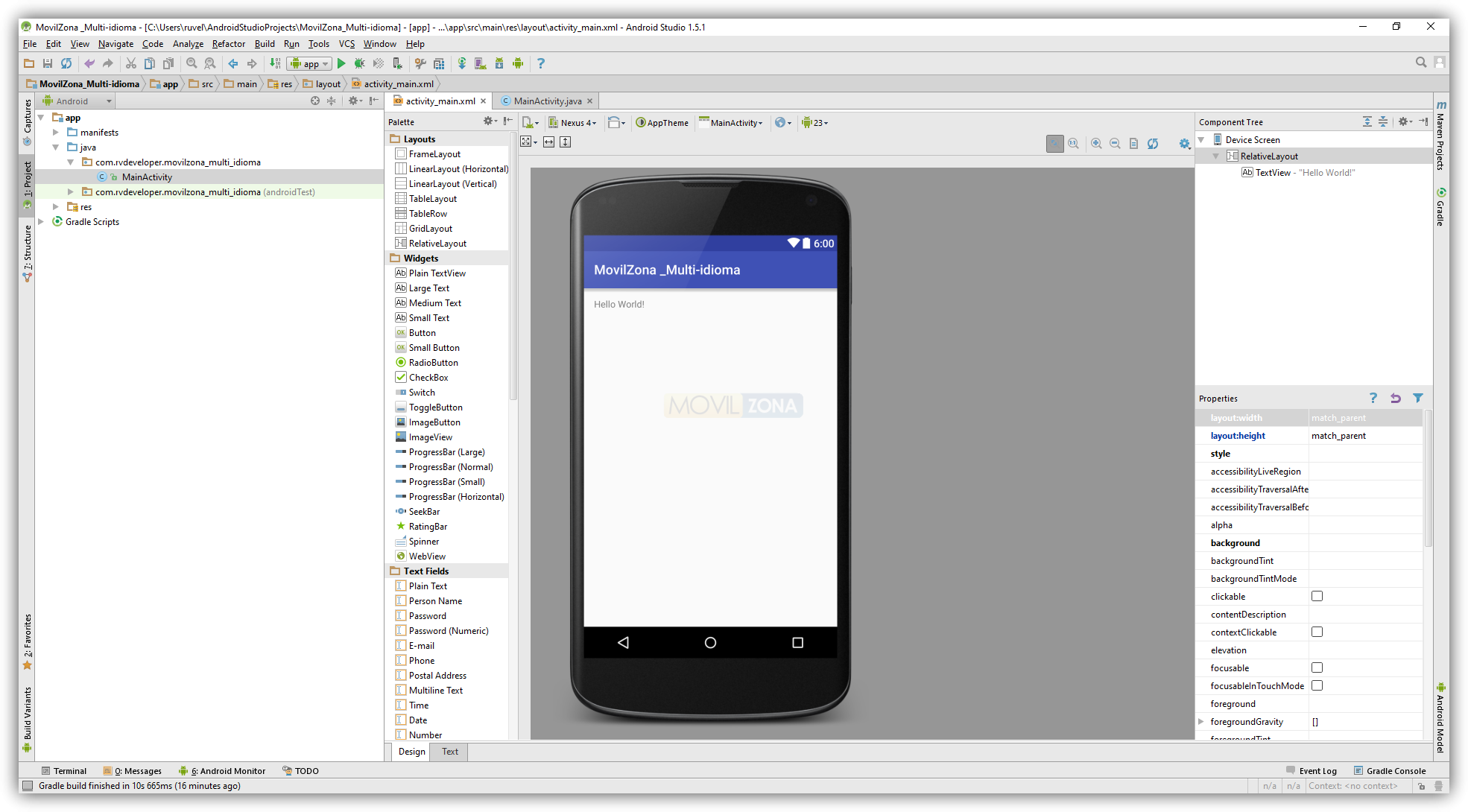
**Subtema 7.2:**

Maquetando nuestros Activities

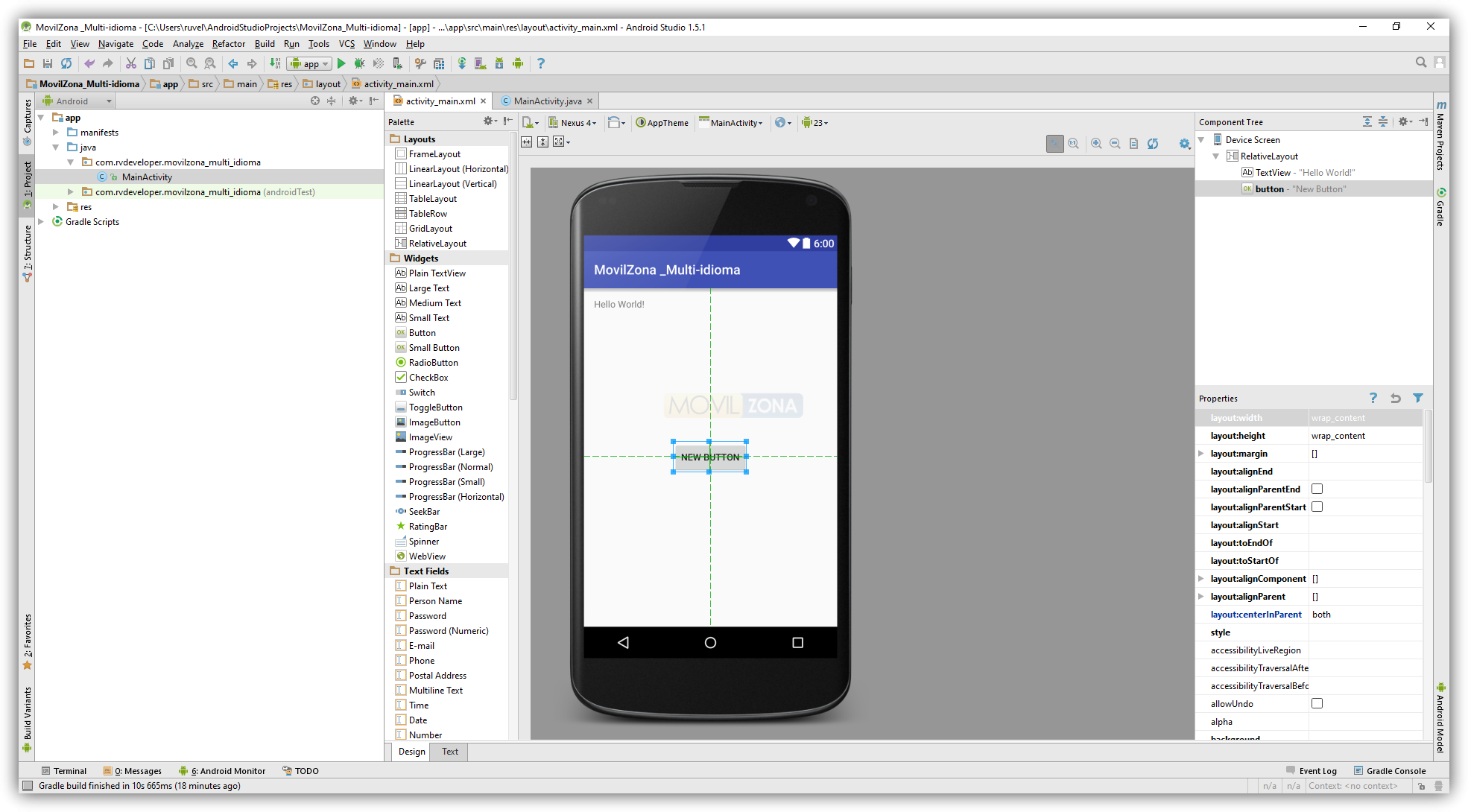
**Ejemplos:**

Lo primero que debemos hacer es crear un proyecto nuevo para este nuevo tema de desarrollo Android. No vamos a profundizar en esto ya que a estas alturas del curso debemos saber cómo crearlo. Este proyecto no tendrá nada de especial, ya que solo lo vamos a usar para ver las funciones de traducción.

Una vez tengamos nuestro proyecto creado (nosotros lo vamos a llamar «MovilZona\_Multi-idioma») abriremos el fichero «**Activity\_main.xml**» y veremos el típico programa inicial «Hola Mundo!».

****

En esta actividad vamos a mantener nuestro texto de «Hello World!» por defecto y vamos a añadir algún View más, por ejemplo, un botón en el centro.



Rellenando el fichero strings.xml y asignando cada View a una entrada de este fichero

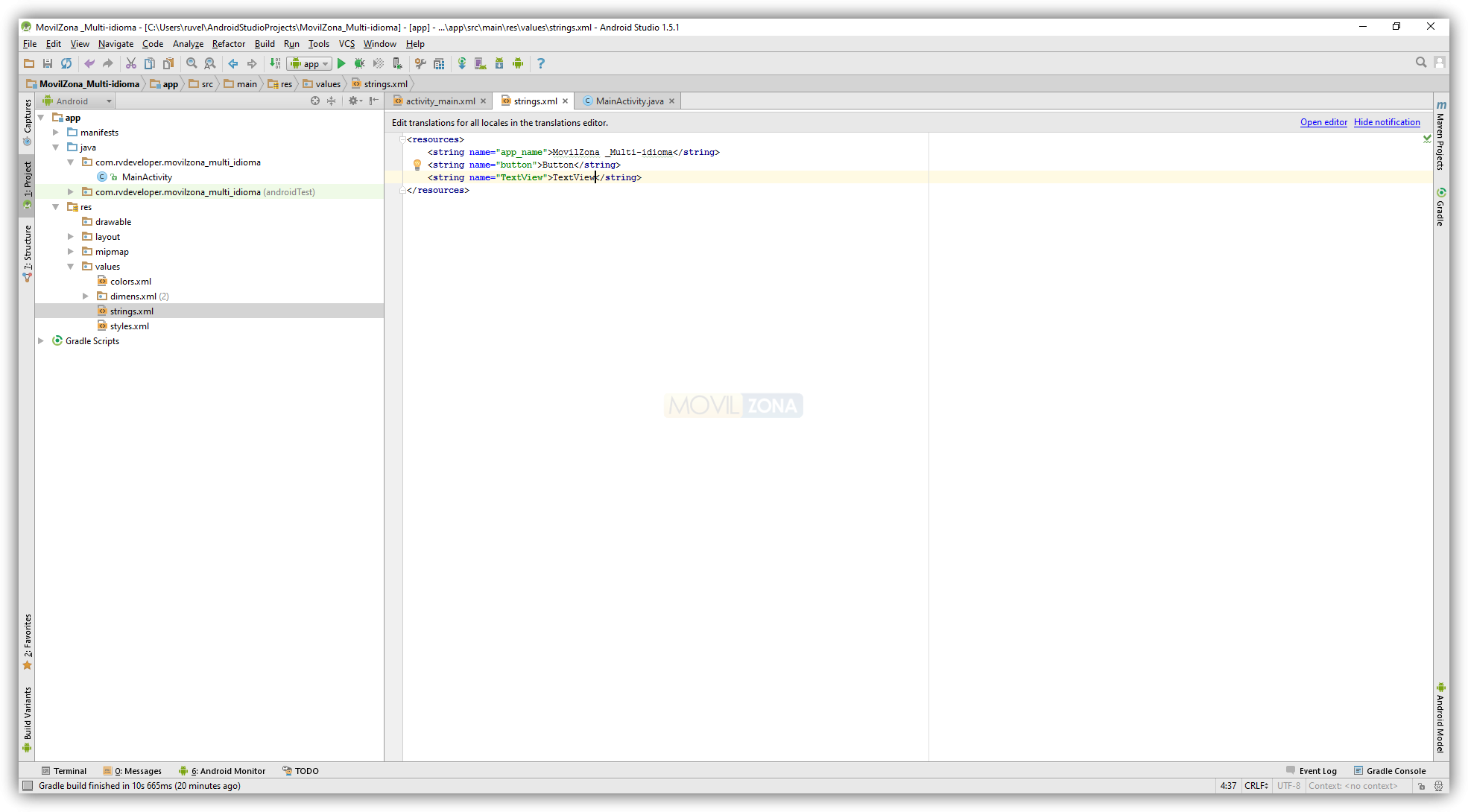
Lo primero que vamos a hacer es **abrir el fichero strings.xml** que, por defecto, estará vacío, solo veremos el nombre de la aplicación. A continuación, agregaremos una línea por cada texto que tengamos en cada View de nuestra aplicación, de la siguiente forma:

<string name=»ID\_DEL\_TEXTO»>TEXTO\_DEL\_VIEW</string>

Por ejemplo:

<string name=»button»>Button</string>

<string name=»TextView»>TextView</string>

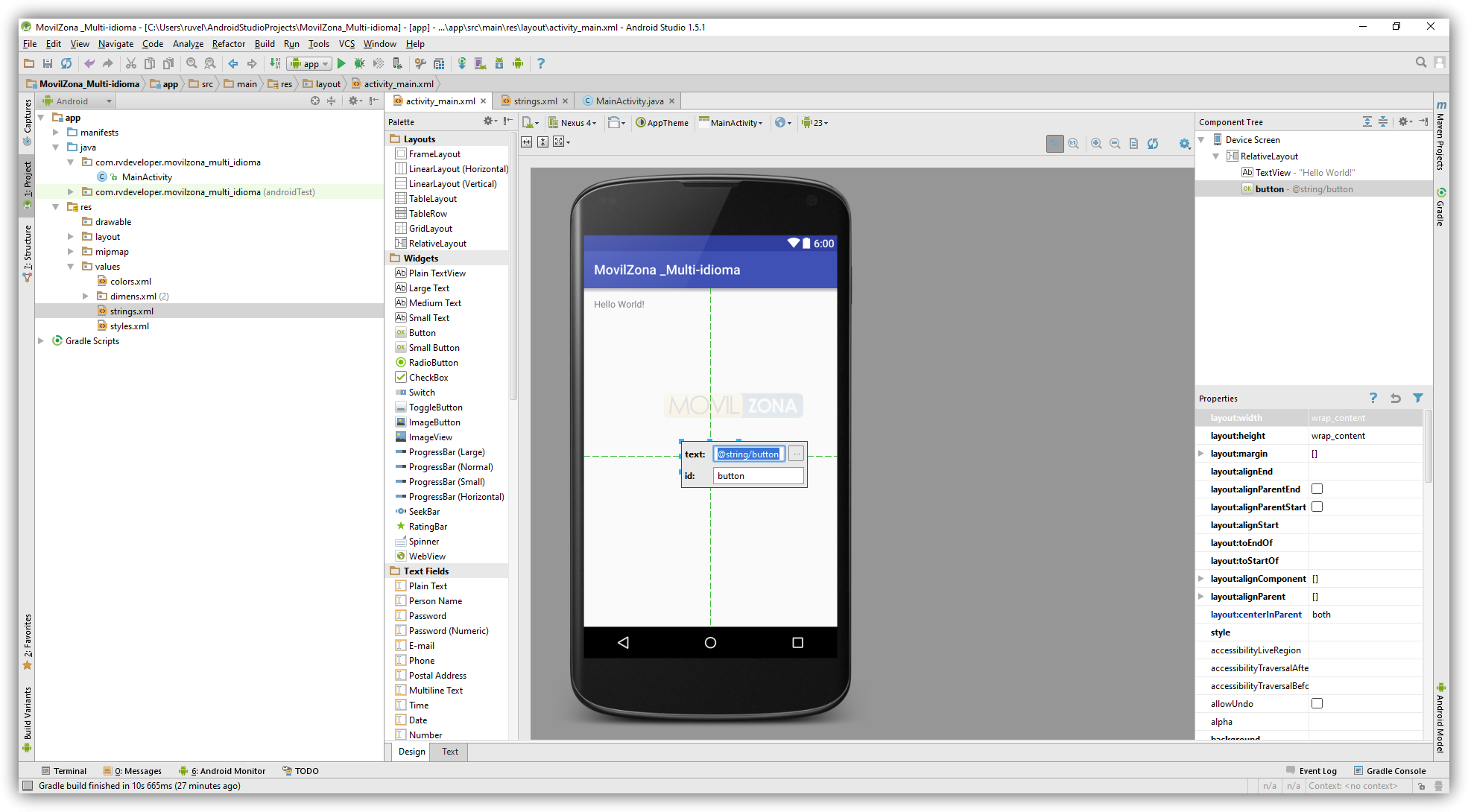


Ahora, volvemos al archivo «Activity\_main.xml» y, en el apartado de texto de cada View, haremos referencia al «String Name» que hayamos añadido en el fichero strings.xml de la siguiente forma:

Text: @string/ID\_DEL\_TEXTO

Por ejemplo:

button -> Text: @string/button



Repetimos lo mismo con el TextView y ya tendremos los dos Views haciendo referencia al contenido del archivo strings.xml. De esta manera, cambiando solo el texto desde el fichero strings.xml se cambiará también el texto de nuestra aplicación.

**Subtema 7.3:**

Internacionalización de Textos

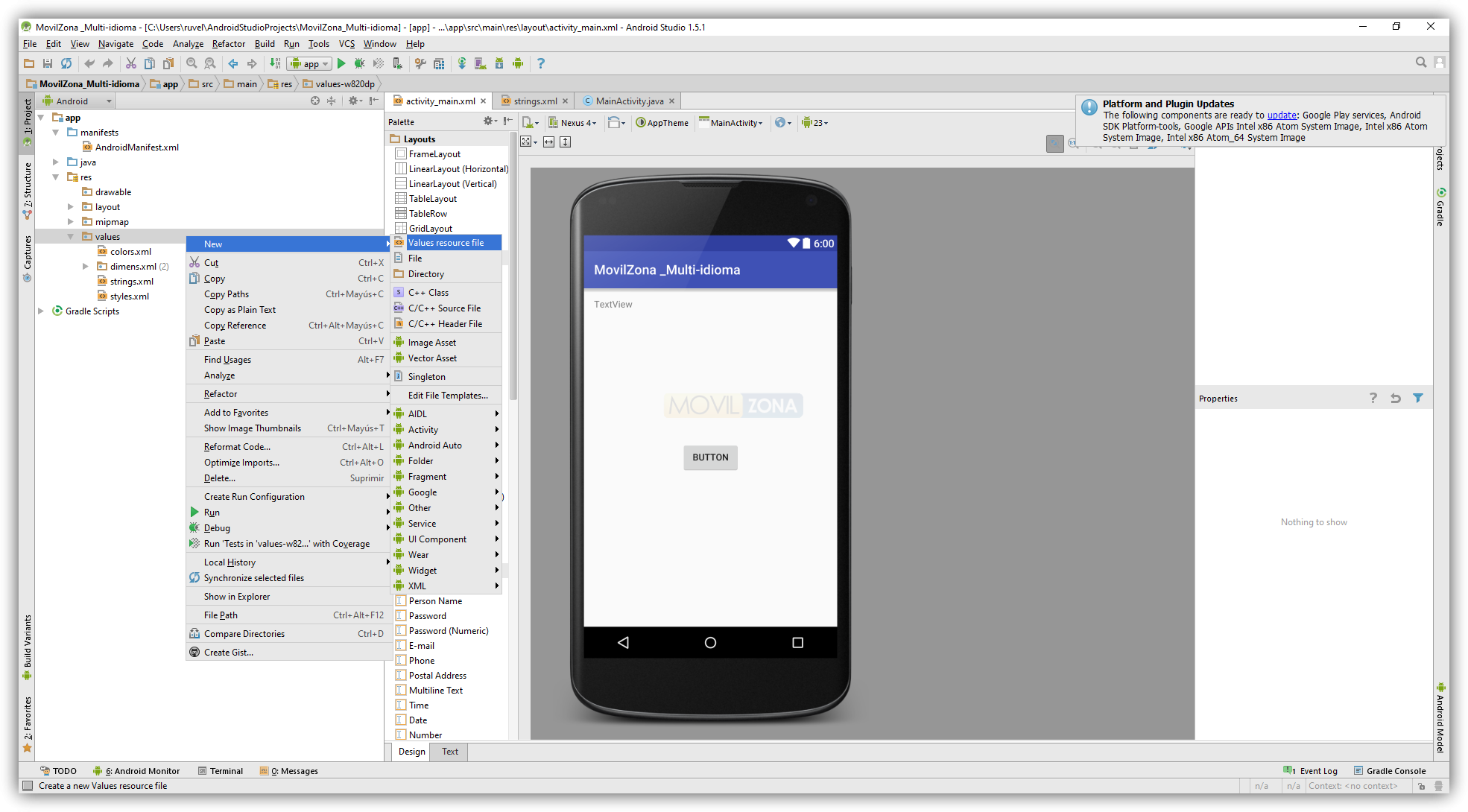
**Ejemplos:**

Cómo traducir nuestra aplicación utilizando varios ficheros string.xml

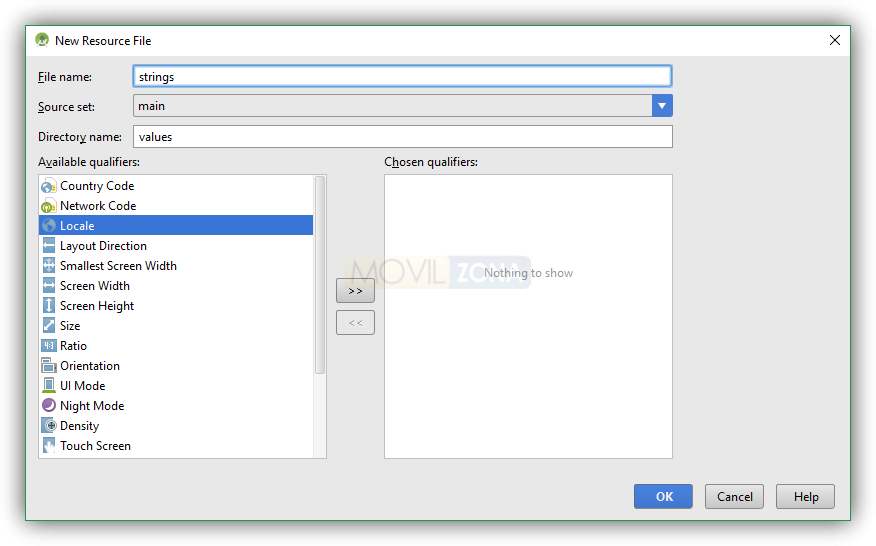
En este momento, todos los textos de nuestra aplicación se almacenan en un fichero strings.xml de manera que todo el código fuente está libre de texto, haciendo siempre referencia a dicho archivo cuando haya que cargar un texto.

Android Studio nos facilita la posibilidad de crear más archivos strings.xml enfocados principalmente a un idioma. De esta manera, cuando Android esté configurado en un idioma cargará su strings correspondiente o, de no haberlo, cargará el strings global. Podemos crear solo las traducciones que necesitemos (incluso crear solo una) ya que, por defecto, si no hay traducción específica cargará el archivo original.

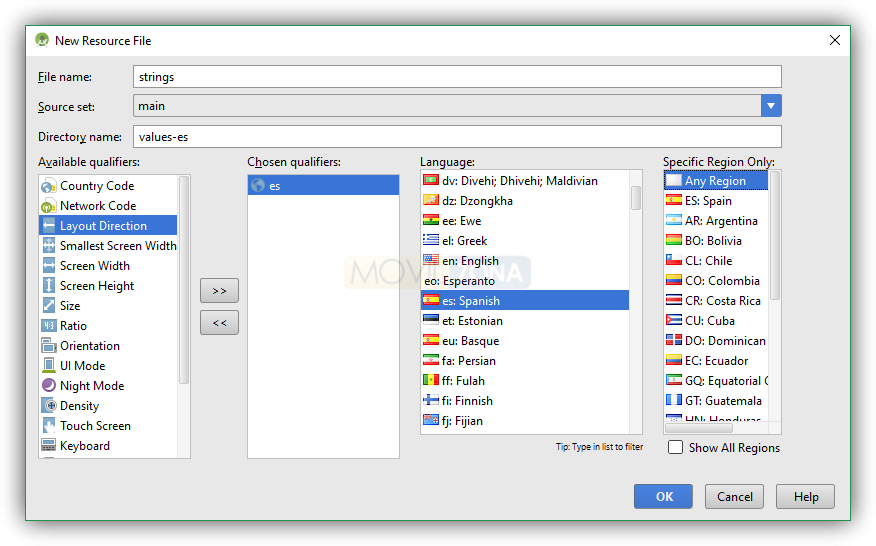
Para crear una traducción debemos hacer clic con el botón derecho en el menú**App > Res > Values** y elegir «**New Values Resource**«.



Se nos abrirá una nueva ventana para crear el recurso. En esta ventana elegiremos el nombre (strings) y el tipo (locale) del menú de la izquierda y pulsaremos sobre la flecha para agregarlo al menú de la derecha.

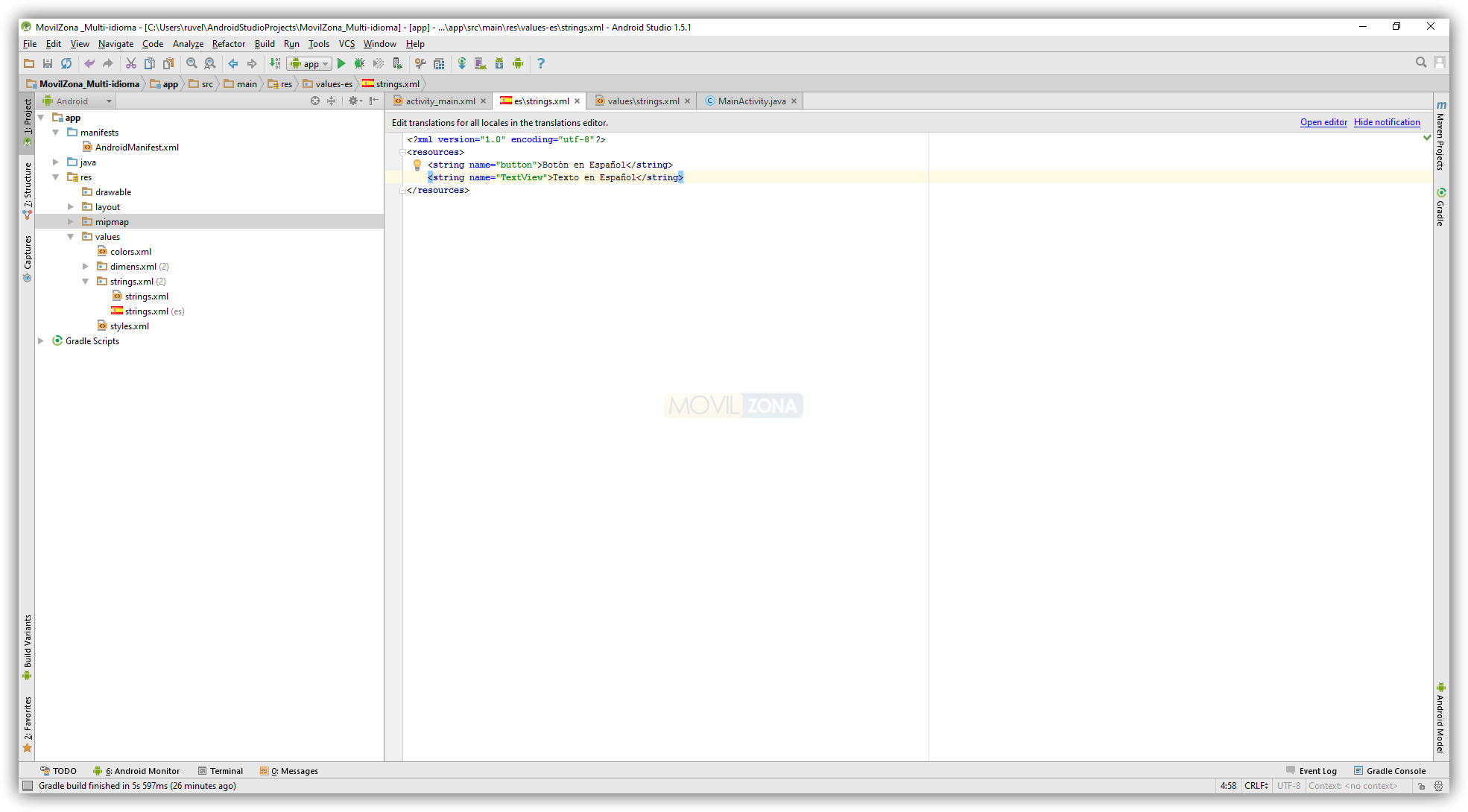


Automáticamente veremos cómo se nos habilitan dos columnas nuevas: Language y Region. Aquí debemos elegir el idioma y la región que correspondan con el idioma al que vamos a traducir, por ejemplo, «es: Spanish» y «Any Region» de manera que esa traducción será «al español» para todas las regiones de habla hispana.

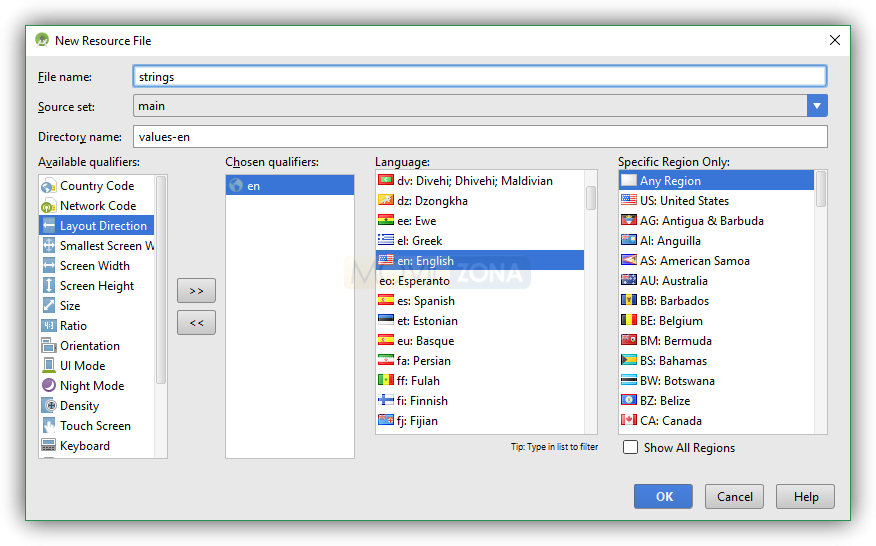


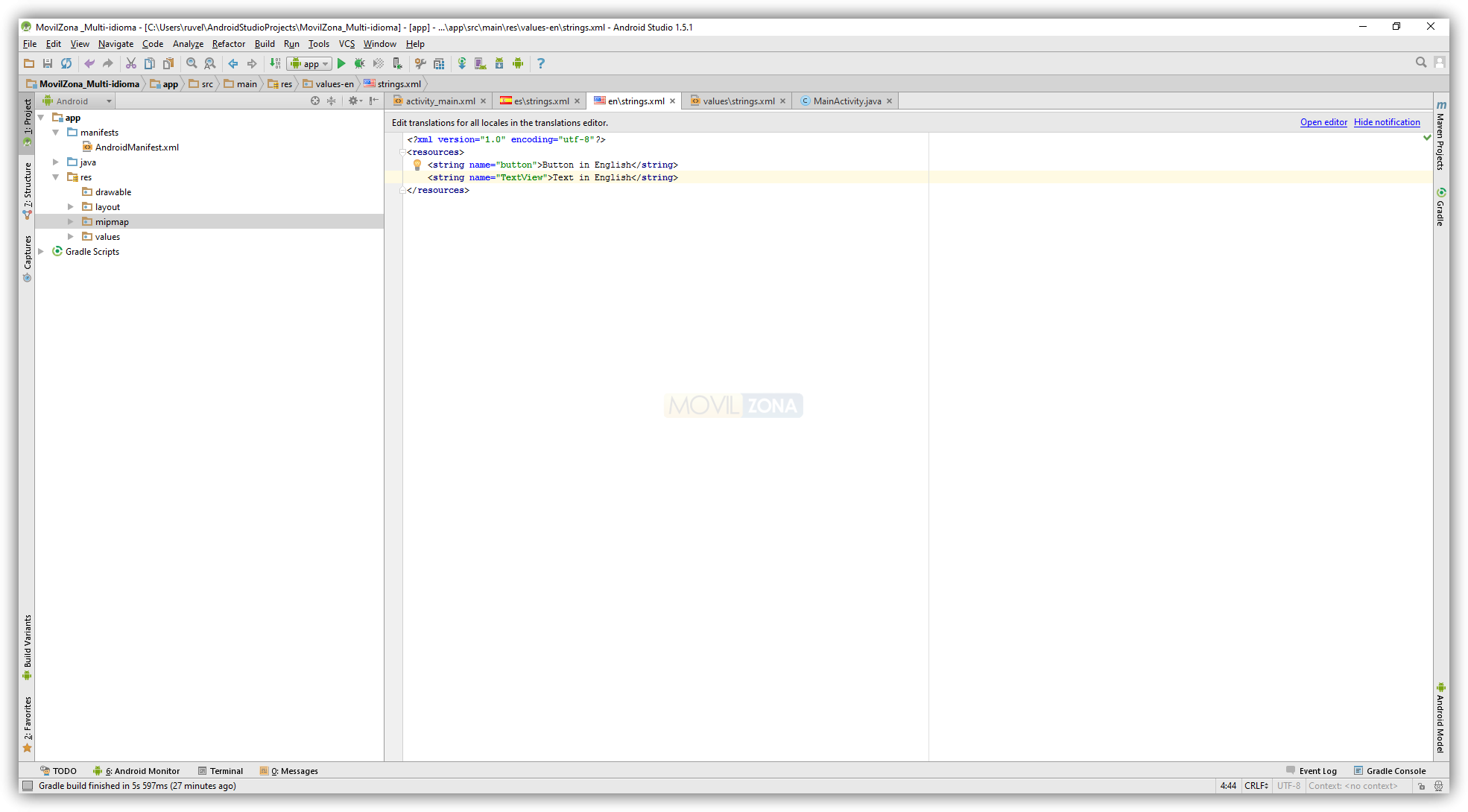
Pulsamos sobre «Ok» y listo, se creará un nuevo strings con el icono de la bandera de España, indicando que dicho fichero almacena la traducción al español.

Abrimos este fichero y lo rellenamos igual que el strings.xml que hemos visto al principio de este tema, aunque cambiando el texto original por el texto traducido.



Si queremos traducir la aplicación al inglés, por ejemplo, el proceso es el mismo tal como vemos en las siguientes imágenes:

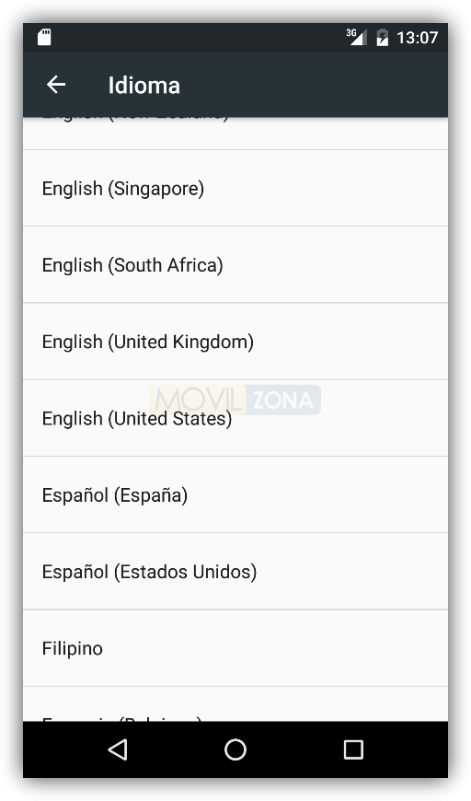




Una vez cambiados los textos, nuestra aplicación ya estará traducida y será multi-idioma, cargando por defecto la traducción acorde al sistema de nuestro Android. Por ejemplo, si nuestro smartphone está en español cargará así:



De lo contrario, si está en inglés, nos cargará nuestra segunda traducción.





Por último, si nuestro móvil está en otro idioma, por ejemplo, en italiano, al no haber traducción el fichero que cargará por defecto será el «strings.xml» original.



Traduciendo nuestra aplicación a varios idiomas, nuestra aplicación tendrá un aspecto más profesional, será más sencilla de utilizar y comprender por todos y llegaremos a un mayor público, lo que a su vez nos permitir obtener un mayor beneficio económico de ella.

**Actividad:**

1. CUESTIONARIO TÉCNICO

* ¿Es adecuado al elegir un idioma en la aplicación la opción Any Region o ser más específico y escoger un país en particular
* ¿Cómo influye en el posicionamiento de una aplicación la implementación de la funcionalidad de internacionalización de textos?
* ¿Qué permite configurar la propiedad atributo fitsSystemWindows?
* ¿Explique por qué no se considera una buena práctica el registrar las cadenas de texto una buena práctica?
* ¿Cuál es su opinión respecto a la importancia de los archivos XML en el desarrollo móvil en Android?

1. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_