2020

Elias Ruiz González

Osiptel

12-8-2020

Trabajo 01



Contenido

[1. Introducción 3](#_Toc48530186)

[2. Requerimientos 3](#_Toc48530187)

[3. Análisis y Diseño 4](#_Toc48530188)

[3.1. Casos de uso 4](#_Toc48530189)

[3.1.1. Conversión 4](#_Toc48530190)

[3.1.2. Selecciona monedas 4](#_Toc48530191)

[3.1.3. Actualiza tipo de cambio 4](#_Toc48530192)

[3.2. Arquitectura 5](#_Toc48530193)

[3.3. Diseño 5](#_Toc48530194)

[3.3.1. Aplicación 5](#_Toc48530195)

[3.3.2. Servicio web 5](#_Toc48530196)

[3.3.3. Base de datos 5](#_Toc48530197)

[4. Implementación 6](#_Toc48530198)

[4.1. Diagrama de Flujo de aplicación 6](#_Toc48530199)

[4.1.1. Diagrama de flujo de actividad principal 6](#_Toc48530200)

[4.1.2. Diagrama de flujo de botones de actividad principal 7](#_Toc48530201)

[4.1.3. Diagrama de flujo de actividad de configuración del usuario (Configurar) 7](#_Toc48530202)

[4.1.4. Diagrama de flujo de botones de actividad de configuración del usuario (Configurar) 8](#_Toc48530203)

[4.2. Inicialización 8](#_Toc48530204)

[4.2.1. Creación de proyecto vacío 8](#_Toc48530205)

[4.2.2. Adición de librería de materiales 8](#_Toc48530206)

[4.2.3. Ajuste de teclado virtual 9](#_Toc48530207)

[4.2.4. Métodos de log no funcionan correctamente 9](#_Toc48530208)

[4.2.5. Errores de librería de compatibilidad 9](#_Toc48530209)

[4.3. Clases auxiliares 10](#_Toc48530210)

[4.3.1. Estructura general de una clase 10](#_Toc48530211)

[4.3.2. Clase de lógica de negocio 11](#_Toc48530212)

[4.3.3. Clase de tipo de cambio 11](#_Toc48530213)

[4.4. Conversión 11](#_Toc48530214)

[4.5. Control de versiones 11](#_Toc48530215)

[4.5.1. Selección de plataforma de control de versiones 11](#_Toc48530216)

[4.5.2. Configuración del control de versiones 12](#_Toc48530217)

[4.6. Conexión a servicio web 12](#_Toc48530218)

[4.6.1. Librería de acceso a servicio web 12](#_Toc48530219)

[4.6.2. Configuración de permisos 12](#_Toc48530220)

[4.6.3. Configuración de llamado a API de servicio web 12](#_Toc48530221)

[4.6.4. Procesamiento de web service 13](#_Toc48530222)

[4.6.5. Uso de objeto JSON resultante del webservice 13](#_Toc48530223)

[4.6.6. Corrección de problema de sincronización 13](#_Toc48530224)

[4.7. Conexión a base de datos 14](#_Toc48530225)

[4.7.1. Selección de librería 14](#_Toc48530226)

[4.7.2. Inicialización de base de datos 14](#_Toc48530227)

[4.7.3. Uso de clase auxiliar 14](#_Toc48530228)

[4.8. Consideraciones sobre ListView 14](#_Toc48530229)

[4.8.1. Reemplazo de ListView por ReciclerView 14](#_Toc48530230)

[4.8.2. Código de referencia para manejo de ListView 15](#_Toc48530231)

[4.9. Imágenes de aplicación en funcionamiento 16](#_Toc48530232)

Trabajo 01

# Introducción

El presente documenta los pasos seguidos desde la identificación de requerimientos hasta la implementación de una aplicación que permite demostrar los conocimientos adquiridos en las primeras lecciones del curso “Introducción al desarrollo de aplicaciones móviles para Android”.

En particular se centra en los siguientes conocimientos:

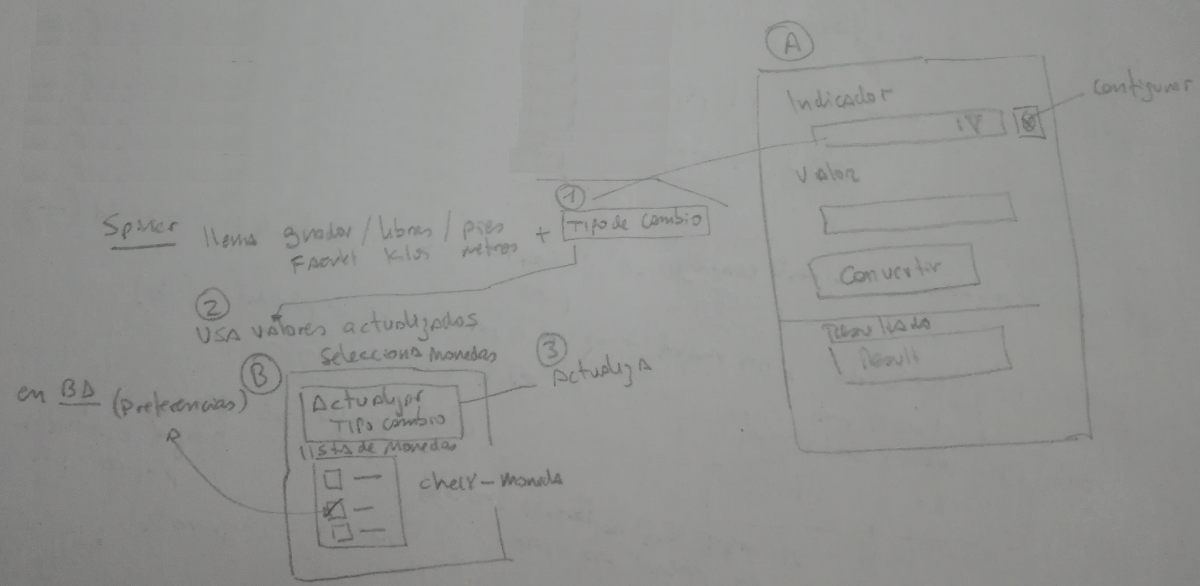
* Almacenamiento de información en base de datos local
* Conexión a un servicio web
* Llenado dinámico de un control “spinner“

De manera adicional se incluirán los siguientes conocimientos:

* Manejo de más de una actividad (“activity”)
* Incremento de complejidad de la lógica básica de aplicación
* Control de versiones/publicación de contenido usando GitHub

# Requerimientos

1. Se desea elaborar una aplicación Android que permita al usuario realizar la conversión de un conjunto de indicadores a elección del usuario, escribiendo números en un cuadro de texto, presionando un botón de cálculo y obteniendo el resultado en un cuadro de texto.
2. Parte de dichos indicadores son elementos fijos: temperatura (grados Celsius/Fahrenheit), Peso (kilogramos/libras), Distancia (metros/pies).
3. Parte de dichos indicadores son elementos variables: Tipos de Cambio (diversas monedas).
4. La lista de monedas y tipos de cambio disponibles se obtiene de un servicio web externo.
5. El usuario puede seleccionar las monedas que serán plausibles de utilizarse para conversión, presionando un botón en la página principal que muestre una lista de monedas, mediante el empleo de un checkbox.
6. La aplicación Android debe incorporar control de versiones.



# Análisis y Diseño

## Casos de uso

Los casos de uso identificados se muestran en el gráfico siguiente, y su descripción se realiza en los acápites subsiguientes.

Actualiza Tipo de Cambio

Conversión

Selecciona monedas

### Conversión

El sistema muestra el “Formulario Principal” que contiene un selector, un cuadro de texto donde ingresa el valor, un botón, y un cuadro de texto donde se mostrará el resultado.

Cuando se carga el formulario la primera vez, el sistema llama a “Actualiza Tipo de Cambio” y obtiene las equivalencias de todas las monedas. Almacena/actualiza los pares de moneda/equivalencia en su base de datos.

Cuando el formulario se activa el sistema lee de su base de datos las monedas que el usuario ha seleccionado para visualizar. Esta lista corresponde a los indicadores variables del selector. A continuación actualiza el selector con los indicadores fijos y los indicadores variables.

El sistema valida que el valor del cuadro de texto correspondiente sea un numero decimal.

El resultado se obtiene multiplicando el valor por su equivalencia.

### Selecciona monedas

El “Formulario Principal” incluye un botón que abre el “Formulario Selección”. Este formulario contiene un botón y una lista.

Cuando se muestra el formulario, el sistema lee de la base de datos las monedas y actualiza la lista. Dicha lista muestra parejas de checkbox y nombres de monedas.

Cuando el usuario presiona el botón, el sistema llama a “Actualizar tipo de cambio”, y adiciona las monedas que no existen en la lista asumiendo que el usuario no deseará visualizarlas, y elimina las monedas que ya no existen en la lista. En todos los casos actualiza la equivalencia con el tipo de cambio respectivo. Finalmente actualiza la lista.

Cuando el usuario presiona algún checkbox, el sistema interpreta si el usuario desea o no visualizar la moneda asociada y actualiza el registro asociado en la base de datos.

### Actualiza tipo de cambio

El sistema llama a un “web service”, obtiene una lista de monedas y tipo de cambio disponible, y finalmente devuelve el resultado.

## Arquitectura

Aplicación

Dispositivo móvil

Servicio web

Internet

El sistema es una aplicación que se ejecuta en un dispositivo móvil (programable) que dispone de una conexión a internet y puede conectarse a un sitio web.

## Diseño

### Aplicación

La aplicación se ejecutará sobre el sistema operativo Android, y se desarrollará utilizando el entorno Android Studio.

El nombre de la aplicación es “Convertisador” y su correspondiente paquetes es “pe.pucp.convertisador”.

Se ha elegido la mínima versión de los dispositivos Android con los que se cuentan en casa, la cual corresponde a Android 4.4.2 (Api 19 KitKat) del dispositivo Samsung Galaxy Tab de 7 pulgadas y modelo SM-t230nt.

El software Android Studio se utiliza en su versión 4.0.1 (build 24/jun/2020).

El lenguaje de programación es Java.

Debido a dificultades laborales, el empleo del emulador se descarta (no se cuentan con privilegios para instalar el software HAXM) por lo que la depuración se realizará utilizando el teléfono del programador, el cual corresponde a Android 6 (Api 23 Marshmallow) del dispositivo Huawei modelo CAM-L03.

### Servicio web

La conexión al web service se realizará mediante la librería Volley.

### Base de datos

La base de datos se ejecutará localmente al dispositivo móvil y se utilizará Sqlite, con su respectiva librería de software.

La base de datos requerirá de una tabla denominada Equivalencias, con los campos que se detallan en la tabla siguiente.

Tabla 1: Estructura de Tabla Equivalencias

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Tamaño | Comentario |
| Moneda | Texto | 20 caracteres | Sigla de la moneda |
| Equivalencia | Numero | Doble | Tipo de cambio con respecto al Nuevo Sol |
| Ver | Numero | Entero | 1: visualizar  0: ocultar |

# Implementación

## Diagrama de Flujo de aplicación

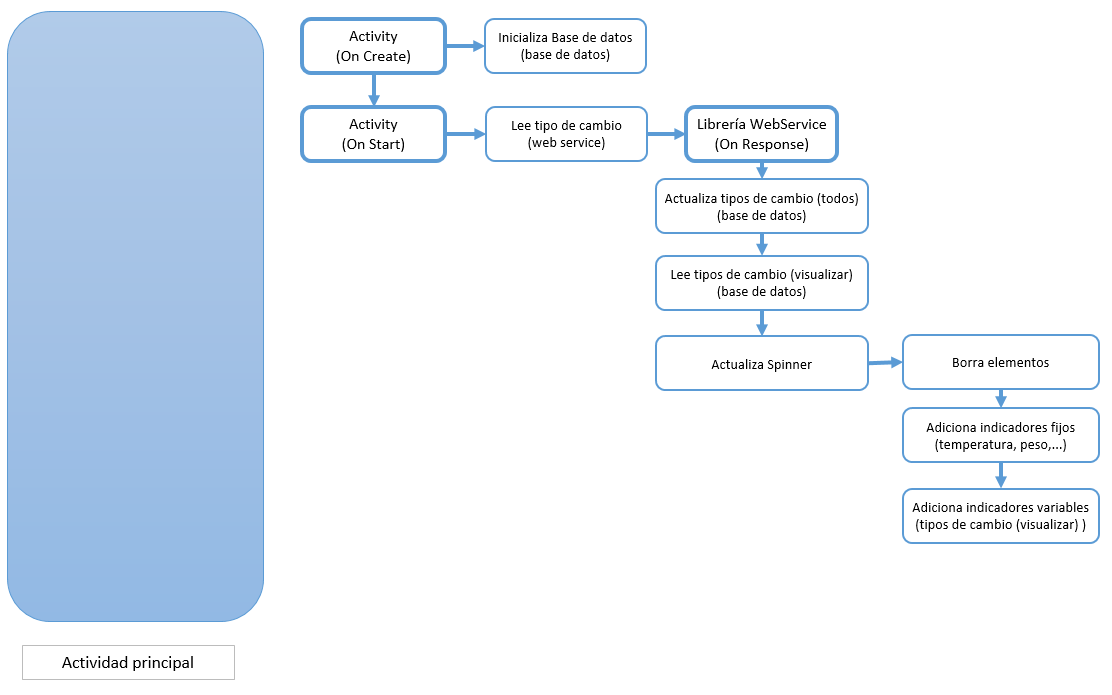
### Diagrama de flujo de actividad principal

La aplicación inicia en la actividad principal, y utiliza el evento “OnCreate” para inicializar la base de datos.

Al incorporar una actividad adicional, se utiliza el evento “OnStart” para continuar con el flujo. De esta forma se utiliza una librería para leer los tipos de cambio de un webservice determinado.

Es importante notar que las librerías de lectura de webservices utilizan hilos de ejecución asíncronos, es decir siguen un flujo de ejecución alterno al de la actividad principal. Si bien este es el procedimiento establecido para las aplicaciones Android y así evitar que un proceso pueda detener a la aplicación e incluso todo el sistema, es necesario comprender su funcionamiento para evaluar la forma adecuada de hacer uso de este mecanismo. De esta forma el resto de acciones del programa deben esperar que la librería reciba los datos del servicio web y genere el evento “OnStart”. Es dentro de este evento que el flujo del programa continúa.

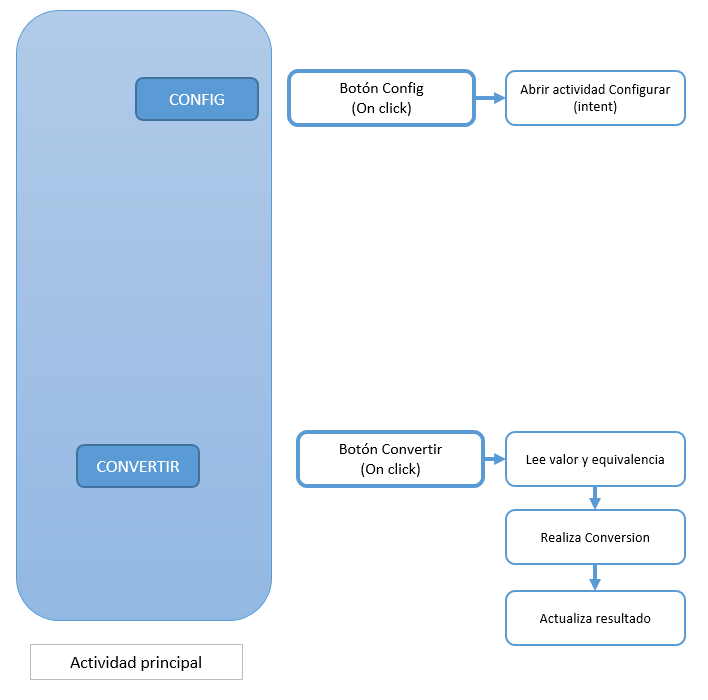
A continuación, utilizando todos los tipos de cambio devueltos por el webservice, se actualizan en la base de datos. Posteriormente se leen sólo los tipos de cambio seleccionados por el usuario de la base de datos y se actualiza el spinner. Considerando que los elementos a mostrar en el spinner pueden variar según el usuario, la actualización del spinner requiere borrarlo, para luego adicionar los indicadores fijos (temperatura, peso, distancia, etc.) y, finalmente, los indicadores variables (tipos de cambio seleccionados para visualizar).



### Diagrama de flujo de botones de actividad principal

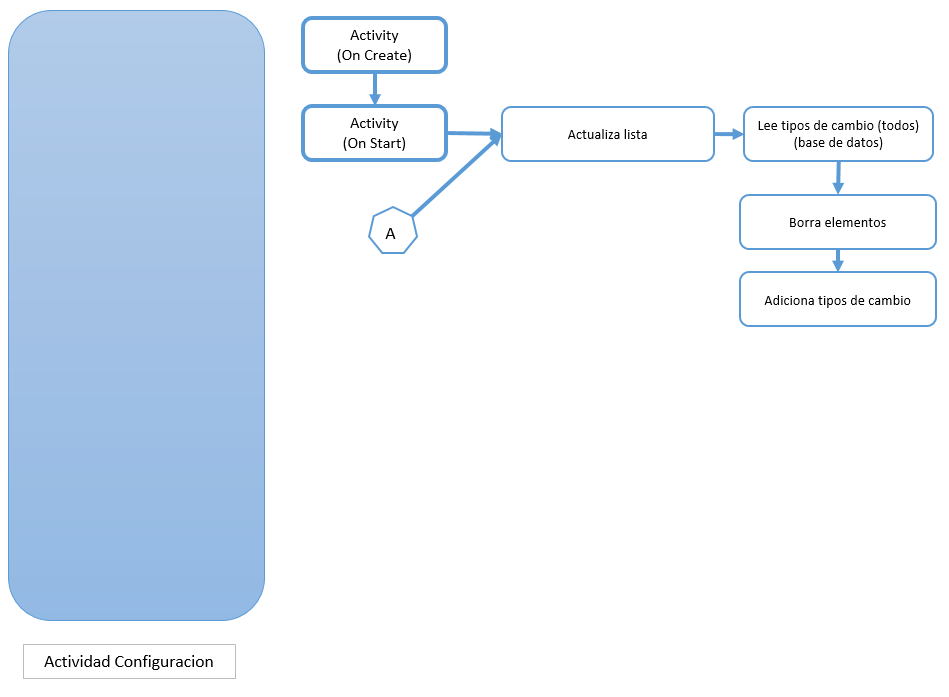
El usuario presiona el botón “Config” y el sistema abre la actividad configurar.

El usuario presiona el botón “Convertir” y el sistema lee el valor a convertir y su equivalencia, realiza la conversión y finalmente actualiza el resultado.



### Diagrama de flujo de actividad de configuración del usuario (Configurar)

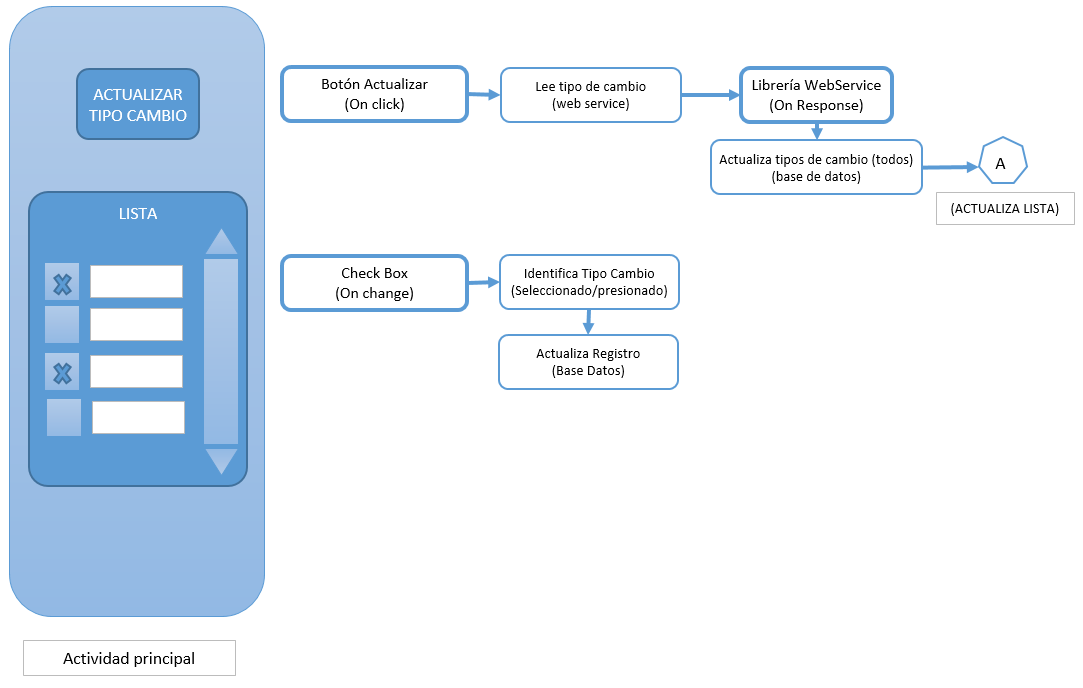
La aplicación inicia en el evento OnStart y procede a actualizar la lista, es decir, lee la lista de tipos de cambio de la base de datos, borra los elementos de la lista y luego los adiciona con los nuevos datos.



### Diagrama de flujo de botones de actividad de configuración del usuario (Configurar)

El usuario presiona el botón actualizar tipo de Cambio, y el sistema consulta al webservice usando la librería seleccionada. En el evento OnResponse de dicha librería el sistema actualiza los tipos de cambio (usando los métodos ya utilizados en la actividad principal) y continua actualizando la lista (usando los métodos ya utilizados en la actividad de configuración).

Cuando el usuario presiona el checkbox del elemento de lista, el Sistema identifica el elemento seleccionado y actualiza el registro respectivo en la base de datos.



## Inicialización

### Creación de proyecto vacío

Crear el proyecto con actividad en blanco, estableciendo las opciones nombre de aplicación, proyecto y API.

### Adición de librería de materiales

Se modifica el archivo build.gradle del proyecto

dependencies {…

implementation **'com.google.android.material:material:1.1.0'**

…}

Se modifica archivo styles.xml

<**style name="AppTheme" parent="Theme.MaterialComponents.Light"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
</**style**>

### Ajuste de teclado virtual

Se modifica el archivo AndroidManifest.xml

<**activity android:name=".MainActivity" android:windowSoftInputMode="adjustResize"**>

### Métodos de log no funcionan correctamente

En algunos casos los métodos de depuración (Log.d) no muestran información en el Android Studio.

Se ha solucionado utilizando la función “Log.wtf”

Log.*wtf*(**"url"**,url);

### Errores de librería de compatibilidad

Ejecutando el programa, sin incorporar aun el código subsiguiente, se obtiene un error de compatibilidad, tal como se aprecia a continuación:

…

2020-08-15 14:33:42.846 4808-4808/pe.pucp.convertisador I/art: Rejecting re-init on previously-failed class java.lang.Class<androidx.core.view.ViewCompat$2>: java.lang.NoClassDefFoundError: Failed resolution of: Landroid/view/View$OnUnhandledKeyEventListener;

2020-08-15 14:33:42.846 4808-4808/pe.pucp.convertisador I/art: at void androidx.core.view.ViewCompat.setBackground(android.view.View, android.graphics.drawable.Drawable) (ViewCompat.java:2559)

…

2020-08-15 14:33:42.847 4808-4808/pe.pucp.convertisador I/art: at void androidx.appcompat.app.AppCompatActivity.setContentView(int) (AppCompatActivity.java:161)

2020-08-15 14:33:42.847 4808-4808/pe.pucp.convertisador I/art: at void pe.pucp.convertisador.MainActivity.onCreate(android.os.Bundle) (MainActivity.java:47)

…

Donde la línea 47 del archivo MainActivity.java, coloreada en verde, se muestra a continuación:

…

*//oncreate se encarga de inicializar la base de datos*@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
…

Este problema se debe a la librería de compatibilidad de Android, la cual se debe actualizar a una versión más reciente. En particular en “[https://stackoverflow.com/questions/51782548/androidxappcompat-iart-error-android-view-viewonunhandledkeyeventlistener/52954286#52954286](https://stackoverflow.com/questions/51782548/androidxappcompat-iart-error-android-view-viewonunhandledkeyeventlistener/52954286%2352954286)” se recomienda lo siguiente:

Reemplazar:

implementation **'androidx.appcompat:appcompat:1.1.0'**

por:

implementation **'androidx.appcompat:appcompat:1.3.0-alpha01'**

Se observa en la ejecución que ya no aparece el error encontrado.

## Clases auxiliares

### Estructura general de una clase

Las clases siguen la convención de nombre-archivo.

Asimismo la estructura de las clases incluirá secciones para propiedades, constructores (Java no utiliza destructores) y métodos, según la plantilla siguiente:

public Class Nombre {

//propiedades

…

//constructor

--constructor nativo

--constructores auxiliares

…

//métodos

…

}

### Clase de lógica de negocio

El Sistema utilizara una clase auxiliar “Cambiador” que encapsulará la lógica de conexión a la base de datos.

### Clase de tipo de cambio

El Sistema utilizará una clase genérica “TipoCambio” para almacenar el nombre y equivalencia de las monedas.

Asimismo adicionará una variable “Ver” que permitirá conocer si la moneda se visualizará u ocultará.

Para simplificar el llenado del spinner, esta clase se utilizará como fuente de sus opciones. Para ello realizará un override de la función ToString, devolviendo el campo “TextoSpinner”, el cual es llenado en forma externa.

## Conversión

La estrategia utilizada en la implementación es utilizar objetos de tipo “TipoCambio” para todas las conversiones. Dichos objetos ya han sido llenados por las funciones respectivas.

En general existen dos formas de conversión, la primera se refiere a cálculos mediante expresiones complejas, como la conversión de temperatura de grados Celsius a Fahrenheit, y la segunda que se refiere a expresiones simples (lineales: y= a.x ) como la conversión de temperaturas, distancias y monedas.

De esta forma, el evento “onclick” del botón de conversión recupera el objeto “TipoCambio” seleccionado del spinner (*variable =* *(cast de tipo)spiner. getSelectedItem()*) y evalúa si se trata de expresiones complejas o simples (*switch – case-break-default*). El evento calcula en forma específica las primeras, en tanto que utiliza una forma común para las demás.

## Control de versiones

### Selección de plataforma de control de versiones

El proyecto utilizará el control de versiones utilizando herramientas libres. En particular se utiliza GitHub, considerando que el programador ya dispone de una cuenta en dicha plataforma.

### Configuración del control de versiones

Los pasos a seguir para utilizar el control de versión se basan en las indicaciones contenidas en <https://academiaandroid.com/uso-de-github-desde-android-studio/>, y se resumen a continuación:

1. Disponer una cuenta GitHub.
   1. Crear una cuenta gratuita o utilizar una ya existente.
   2. En este proyecto se ha utilizado la cuenta existente del programador.
2. Activar la cuenta de GitHub en Android Studio.
3. Activar el control de versiones usando GitHub en el proyecto específico en Android Studio.
4. Utilizar el control de versiones.

El proyecto se encuentra disponible en la siguiente url:

<https://github.com/ergio2000/convertisador>

Finalmente, recordar que, al momento de termnar de contener parte del código operativo, realizar una acción commit (VCS/Commit), y luego el push respectivo (VCS/Git/Push).

## Conexión a servicio web

### Librería de acceso a servicio web

Se utilizará la librería Volley

Se adiciona referencia en archivo build.gradle de la aplicación

implementation **'com.android.volley:volley:1.1.1'**

### Configuración de permisos

Habilitar el permiso de acceso a internet en el archivo AndroidManifest.xml

<**uses-permission android:name="android.permission.INTERNET"**/>

### Configuración de llamado a API de servicio web

El código de ejemplo incluye una lista acotada de monedas, al utilizar el comando “symbol”.

String url = **"https://openexchangerates.org/api/latest.json?app\_id=HEXADECIMAL&symbols=PEN,EUR,COP,CLP"**;

En el Sistema se omitirá para tener disponibles todas las monedas. En otro procedimiento o función se verificará si el usuario desea incluirla en el spinner, o si se almacenará en la base de datos local.

### Procesamiento de web service

El resultado del web service es un documento en formato JSON. Según la documentación disponible en <https://docs.openexchangerates.org/docs/latest-json> su estructura tiene la forma “{disclaimer, license, timestamp, base, rates}”

Para fines de implementación se debe considerar dos aspectos importantes. El primero referido a la moneda base de comparación, por defecto es el dólar americano (USD). El segundo es referido a la lista de monedas disponibles, las cuales se encuentran en la variable “rates”, la cual a su vez contiene las parejas “{sigla de moneda: tipo de cambio respecto a moneda base}”.

### Uso de objeto JSON resultante del webservice

El sistema requiere conocer las parejas de “moneda-tipo de cambio”, y encapsularlas en un objeto interno (TipoCambio) para lo cual debe leerse el nombre de las monedas (nombre de propiedad JSON), y luego el valor del tipo de cambio (valor de la propiedad JSON).

Para tal efecto se hace uso de la función “keys()” que devuelve un iterador con los nombres de las propiedades JSON del objeto resultante del webservice, y por medio de éste, se accede a los diferentes tipos de cambio disponibles.

…

mr= **new** ArrayList<TipoCambio>();

…

TipoCambio mtc;

…

JSONObject tasas = response.getJSONObject(**"rates"**);  
Iterator<String> iter = tasas.keys();  
**while** (iter.hasNext()) {  
 *//obtiene moneda y tipo de cambio* key = iter.next();  
 valor = tasas.getDouble(key);  
 *//crea objeto de respuesta* mtc = **new** TipoCambio(key,valor);  
 *//adiciona a respuesta* mr.add(mtc);  
 *//Log.d("keys", key);*}

…

### Corrección de problema de sincronización

La librería Volley es asíncrona por naturaleza, en tal sentido, la llamada se realiza en un hilo diferente al de la actividad principal.

El tratamiento respectivo se detalla en el acápite 4.1.1

## Conexión a base de datos

### Selección de librería

La base de datos seleccionada será la versión de Sqlite3 para Android.

Para la implementación se utilizará una clase adicional denominada DbHandler.

### Inicialización de base de datos

Se realiza en el evento “onCreate” de la actividad principal

Se verifica que la base de datos y tabla de equivalencias exista (se crean en caso contrario)

El nombre del archivo de base de datos será “monedasdb”.

Por su parte la tabla de denominará “equivalencias” y tiene la estructura de la tabla Equivalencias descrita en la sección 3 (Análisis y Diseño)

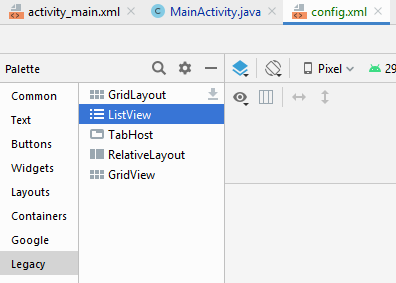
### Uso de clase auxiliar

La implementación del acceso real a la base de datos se encapsula en la clase “DbHandler”, la cual contiene el detalle nombre del archivo físico, nombre de la tabla donde se almacenan los tipos de cambio (“Equivalencias”), instrucciones de creación de dicha tabla.

## Consideraciones sobre ListView

### Reemplazo de ListView por ReciclerView

El clásico control listView es reemplazado desde el 2016/2017 por el control ReciclerView (puede observarse que ListView aparece dentro del rubro Legacy).



El sustento utilizado fue la eficiencia: Listview crea una vista por cada elemento de la vista la cual resulta sumamente ineficiente cuando la lista contiene cientos de elementos. Para tal efecto el nuevo control ReciclerView crea un única vista para todos los elementos. (Ver <https://medium.com/@relferreira/goodbye-listview-recyclerview-f83dc1133850#:~:text=RecyclerView%20is%20a%20somewhat%20new%20view%20that%20came%20to%20substitute,and%20better%20arrangements%20of%20elements.>, o <https://thelurkingvariable.com/2017/02/18/android-recyclerview-listview-replacement/>)

No obstante, para fines de la presente implementación se mantendrá el empleo de ListView.

### Código de referencia para manejo de ListView

Se utiliza la referencia desarrollada en <https://www.tutorialspoint.com/how-to-bind-data-from-a-database-to-an-android-checkbox-in-a-listview>.

Asimismo, para resolver el problema que el checkbox superpone el foco de la fila, se considera las modificaciones en el elemento checkbox indicadas en <https://stackoverrun.com/es/q/4950828>.

…

<**CheckBox  
 android:id="@+id/checkBox"  
…**

**android:focusable="false"  
 android:clickable="false"** />

…

## Imágenes de aplicación en funcionamiento

A continuación se muestran ejemplos de la aplicación en funcionamiento.

