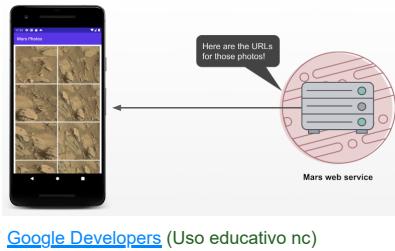


Caso práctico



Una de las aplicaciones que BK Programación ha desarrollado es para un cliente con presencia en distintos países del mundo. Esto implica que además del factor multidioma hay que tener en cuenta otra cuestión importante como la monedas -divisas- utilizadas en su tienda.

- Tenemos un cliente con tienda en distintos países del mundo -explica Ada a su equipo-, y necesita que se pueda comprar en su tienda en la divisa que corresponda a cada país.
- Eso es un problema -razona Juan-, puesto que el valor de la divisa fluctúa cada día. Para saber cuánto cuesta un producto de nuestro cliente en distintas monedas, necesitaríamos dedicar varios minutos cada día (sin excepción) para calcular, por ejemplo, cuántos dólares (USD) cuestan los productos fabricados en euros (EUR). Pero seguro que ya tienes una solución al problema y nos la vas a mostrar.
- ¡Por supuesto! -exclama Ada-. No se puede dedicar tiempo cada día a realizar la conversión, debe ser algo automático. Por ello es necesario que nuestra aplicación se conecte con algún servicio que ofrezca esta información publicada en Internet. El mecanismo normal más extendido actualmente es publicar esta información siguiendo unos estándares de llamadas que usan el protocolo HTTP y responden la información usando formatos como el XML o el JSON: estamos hablando de REST.
- ¿Y de este modo se puede acceder sólo a texto publicado? -pregunta María.
- No -responde Ada-. Además de formato texto, es posible también publicar la URL de distintas imágenes.

En la figura se ve cómo la NASA, por ejemplo, publica imágenes capturadas por sus robots en Marte.

El equipo se dispone a trabajar con distintas librerías como Retrofit que, con conocimientos de programación orientada a objetos (Clases, POJO, etc.), facilitarán la labor de hacer aplicaciones Android más y mejor conectadas.



[Ministerio de Educación y Formación Profesional.](#) (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

[Aviso Legal](#) 

1.- HttpURLConnection.

HttpURLConnection es una clase específica que a través del protocolo HTTP permite a los sistemas clientes (habitualmente navegadores, aunque en nuestro caso, el propio dispositivo Android será el cliente sin tener que mediar un navegador para acceder al contenido) acceder al contenido de páginas web.

Los navegadores hacen uso de la primitiva o método **GET** de HTTP para descargarse una página web del servidor. Es decir, cuando en un navegador web cargamos la URL: `https://dominio.com/dir/pagina.html`, internamente se está haciendo un `GET dominio.com/dir/pagina.html`. El servidor recibe esta petición y envía al cliente la página en formato HTML.

De la misma forma, nuestro dispositivo Android, con las librerías adecuadas, será capaz de reproducir directamente el proceso anterior. Dichas librerías principalmente son de dos grupos, las librerías **java.net.*** y las librerías **org.apache.commons.httpclient.***.

Veremos un ejemplo de uso de HttpURLConnection, que pertenece al primer grupo de las librerías anteriores. Concretamente, usaremos las clases:

- ✓ `java.net.HttpURLConnection`
- ✓ `java.net.URL`
- ✓ `java.net.URLEncoder`

1.1.- Ejercicio resuelto 1 (HttpURLConnection).

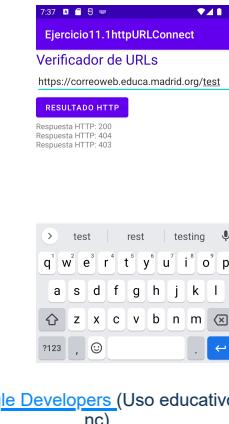
Ejercicio Resuelto

Realizaremos una aplicación que solicitará al usuario una URL (incluyendo el protocolo) y se conectará a ella para devolver el resultado HTTP de la misma. Te mostramos en la imagen cómo quedaría la aplicación con estos ejemplos:

- ✓ `https://www.educa2.madrid.org/educamadrid/"` OK 200
- ✓ `https://www.educa2.madrid.org/educamadridNO/"` No encontrado 404
- ✓ `https://correoweb.educa.madrid.org/test` Forbidden 403

Es importante que conozcas los distintos [códigos de respuesta del protocolo HTTP](#).

NOTA: Debes tener en cuenta, que en ocasiones, dependiendo de la calidad del certificado del servidor al que te conectes puedes recibir códigos de estado no esperados.



Google Developers (Uso educativo nc)

[Mostrar retroalimentación](#)

AndroidManifest.xml

Lo primero será dar permiso de uso de Internet a través del **AndroidManifest.xml** mediante la directiva:

```
1 | <uses-permission android:name="android.permission.INTERNET"/>
```

activity_main.xml

Además crearemos una interfaz básica a través de **activity_main.xml**:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical"
8     tools:context="com.cidead.pmdm.ejercicio111httpurlconnect.MainActivity">
9
10    <TextView
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="Verificador de URLs"
14        android:textSize="25dp"
15        android:textColor="@color/purple_700"
16        android:textAlignment="center"/>
17
18    <EditText android:id="@+id/et1"
19        android:layout_height="wrap_content"
20        android:layout_width="match_parent"
21        android:hint="URL a verificar"
22        android:text="https://www.educa2.madrid.org/educamadrid/"/>
23
24    <Button android:id="@+id/btn1"
25        android:layout_height="wrap_content"
26        android:layout_width="wrap_content"
27        android:onClick="buscar"
28        android:text="Resultado HTTP"/>
```

```
29\n30    <TextView android:id="@+id/tv1"\n31        android:layout_height="match_parent"\n32        android:layout_width="match_parent"/>\n33\n34 </LinearLayout>
```

MainActivity.java

Por último, crearemos el **MainActivity.java**:

```
1 package com.cidead.pmdm.ejercicio111httpurlconnect;\n2\n3 import androidx.appcompat.app.AppCompatActivity;\n4\n5 import android.os.Bundle;\n6 import android.os.StrictMode;\n7 import android.util.Log;\n8 import android.view.View;\n9 import android.widget.EditText;\n10 import android.widget.TextView;\n11 import java.net.HttpURLConnection;\n12 import java.net.URL;\n13\n14 public class MainActivity extends AppCompatActivity {\n15     private EditText et1;\n16     private TextView tv1;\n17\n18     @Override\n19\n20     protected void onCreate(Bundle savedInstanceState) {\n21         super.onCreate(savedInstanceState);
```

```
22     setContentView(R.layout.activity_main);
23     et1 = (EditText) findViewById(R.id.et1);
24     tv1 = (TextView) findViewById(R.id.tv1);
25     //Con la siguiente línea permitiremos que se pueda acceder a la red desde el hilo principal (ya que n
26     StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder().permitNetwork().build());
27 }
28
29 public void buscar(View view) {
30     try {
31         String urlBusqueda = et1.getText().toString();
32         String resultado = resultadosHTTP(urlBusqueda);
33         tv1.append("Respuesta HTTP: " + resultado + "\n");
34     } catch (Exception e) {
35         tv1.append("Error al conectar\n");
36         Log.e("HTTP", e.getMessage(), e);
37     }
38 }
39
40
41 String resultadosHTTP(String urlABuscar) throws Exception {
42     String resultado = "No conectado";
43     try {
44         //URL url = new URL("https://google.es", "UTF-8") + "\n";
45         URL url = new URL(urlABuscar);
46         //Preparamos la conexión con la clase HttpURLConnection
47         HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
48         //Establecemos cabeceras HTTP. Podemos asignar a la clave "User-Agent" parámetros de un navegador
49         //"Mozilla/5.0 (Windows NT 6.1)", pero si lo dejamos vacío (""), también funciona.
50         conexion.setRequestProperty("User-Agent", "");
51         //Con el método getResposeCode() estableceremos la conexión y devuelve el resultado
52         resultado = String.valueOf(conexion.getResponseCode());
53     } catch (Exception e) {
54         tv1.append("Error en la URL\n");
55         Log.e("HTTP", e.getMessage(), e);
56     }
57     return resultado;
58 }
59 }
```





2.- REST.

De la Wikipedia:

La **transferencia de estado representacional** (en inglés *representational state transfer*) o **ReST** es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP, y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

REST describe cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos en cualquier formato (XML, JSON, etc.), sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- ✓ Un protocolo **cliente/servidor sin estado**: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
- ✓ Un conjunto de **operaciones bien definidas** que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son **POST**, **GET**, **PUT** y **DELETE**. Con frecuencia estas operaciones se equiparan a las operaciones **CRUD** en bases de datos (CLAB en castellano: crear, leer, actualizar, borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
- ✓ Una **sintaxis universal** para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- ✓ El uso de **hipermedios**, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente **HTML** o **XML**. Como resultado de esto, es posible navegar de un recurso REST a muchos otros simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional

Debes conocer

Existen diversos mecanismos para consumir un servicio web. Los más popularizados son SOAP y REST. Por simplificación, en esta unidad veremos sólamente REST, aunque es importante que conozcas las diferencias entre ambos mecanismos, puede que en algún momento tengas que utilizarlas. Las diferencias principales entre SOAP y REST son:

- ✓ SOAP significa Protocolo simple de acceso a objetos, mientras que REST significa Transferencia de estado representacional.
- ✓ SOAP es un protocolo mientras que REST es un patrón arquitectónico.
- ✓ SOAP usa interfaces de servicio para exponer su funcionalidad a las aplicaciones cliente, mientras que REST usa localizadores de servicios uniformes para acceder a los componentes en el dispositivo de hardware.
- ✓ SOAP necesita más ancho de banda para su uso, mientras que REST no necesita mucho ancho de banda.
- ✓ Comparando SOAP vs REST API, SOAP solo funciona con formatos XML, mientras que REST funciona con texto sin formato, XML, HTML y JSON. SOAP no puede hacer uso de REST mientras que REST puede hacer uso de SOAP.

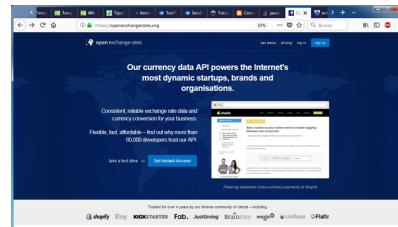
2.1.- Ejercicio resuelto 2 (Fuentes de datos REST).

Existen múltiples fuentes de datos que proporcionan servicio web mediante REST.

Una de ellas es: <https://openexchangerates.org> 

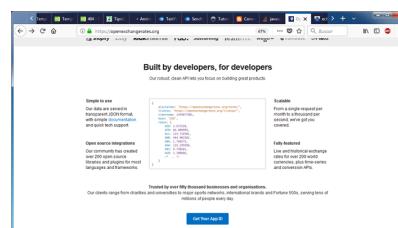
Si nos conectamos a dicha URL (figura 1), bajando en esa ventana vemos que hay un botón para obtener un ID de uso, ya que es habitual que para evitar abusos de utilización haya que registrarse (algunos otros servicios no requieren identificación ni API key) (figura 2). Utilizaremos el plan gratuito (figura 3).

Figura 1. Acceso Open Exchange Rates



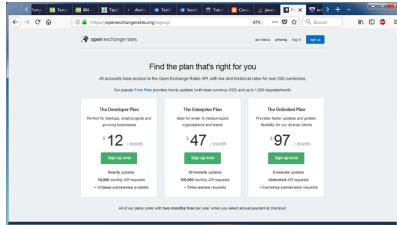
[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

Figura 2. Obtención de ID.



[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

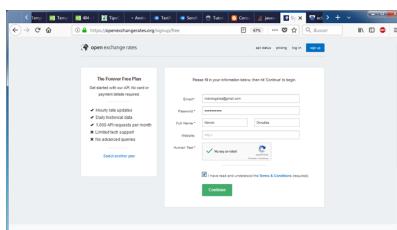
Figura 3. Plan gratuito.



[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

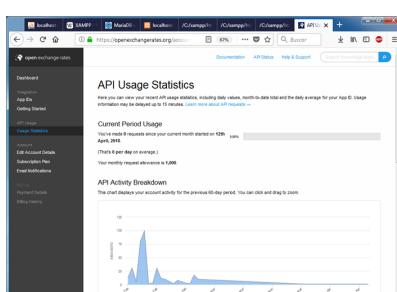
Nos registramos (figura 4). Una vez validados, podemos entrar en nuestro escritorio (figura 5) y buscar la App ID que tenemos (figura 6).

Figura 4. Registro.



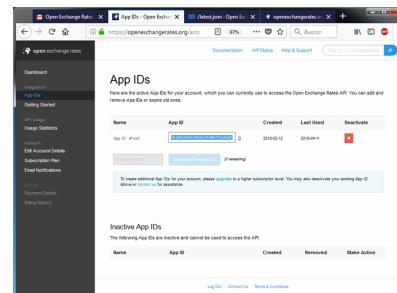
[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

Figura 5. Acceso a escritorio.



[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

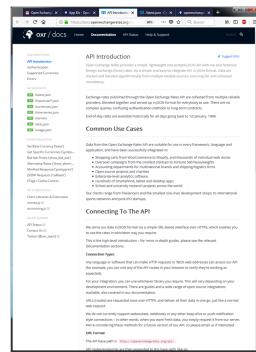
Figura 6. Búsqueda de la App ID



[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

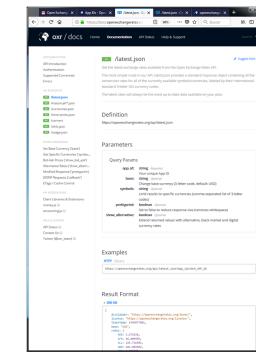
Ahora tendremos que buscar API *Endpoints*. Para ello, vamos al enlace "Getting Started" de la ventana anterior, abriéndose la ventana mostrada en la figura 7. De entre las API EndPoints cogemos, por ejemplo, la primera (/latest.json) (figura 8). Dicha página describe como se utiliza (definición y parámetros que admite) dicha API EndPoint.

Figura 7. Enlace "Getting Started"



[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

Figura 8. Seleccionamos /latest.json

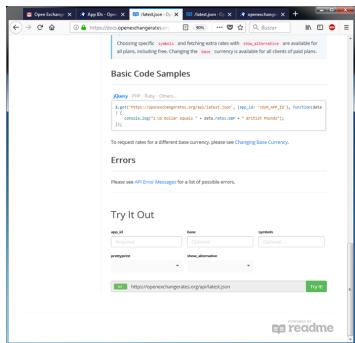


[Openexchangerates.org](https://openexchangerates.org) (Uso educativo nc)

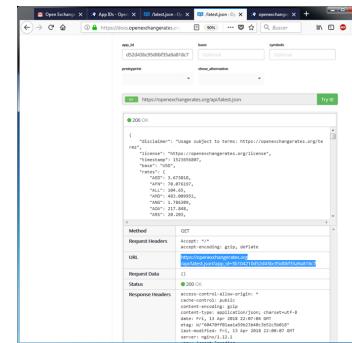
En la parte de abajo podemos probarla, para lo cual debemos suministrar obligatoriamente la App ID anteriormente mencionada (figura 9) y darle a "Try It!" (figura 10):

Figura 9. Suministrar la App ID

Figura 10. Try it!



[Openexchangerates.org](https://openexchangerates.org/) (Uso educativo nc)



[Openexchangerates.org](https://openexchangerates.org/) (Uso educativo nc)

Podemos ver que se genera la última valoración (ver que la fecha coincide con la del día actual en el que hacemos la petición) de cambio del dólar (base) con respecto a las demás monedas, usando formato **JSON**.

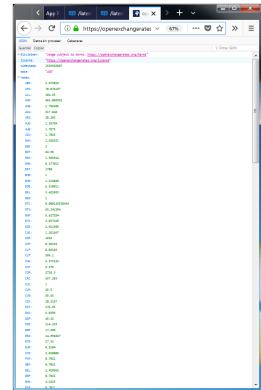
Además podemos poner la URL en un navegador e igualmente aparecerán los resultados en formato **JSON** (figura 11). Dicho fichero será descargable, pero en todo caso será preferible mantener como fuente de datos al servidor web de openexchagerates.org para asegurarnos la puntual actualización de los datos.

Si ahora intentamos cambiar la base a euros (EUR), vemos que desafortunadamente nos da un mensaje de error diciendo que el cambio de base está reservado a planes de pago, por lo cual el único cambio de base que podemos hacer es el de dólar americano (USD) a todo lo demás (figura 12).

En todo caso, este problema no es insalvable ya que será posible hacer el cambio a través de un simple algoritmo matemático, partiendo del cambio de dólar americano a todo lo demás, para poder hacer cualquier conversión posible.

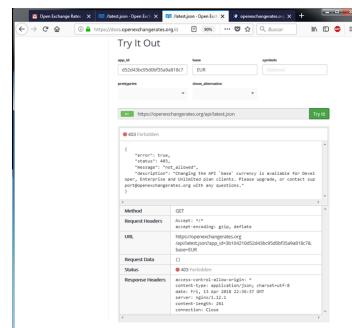
Podría igualmente llevar la URL anterior a aplicaciones como SoapUI para probar el acceso al WebService anterior (figura 13).

Figura 11. Resultados en formato JSON.



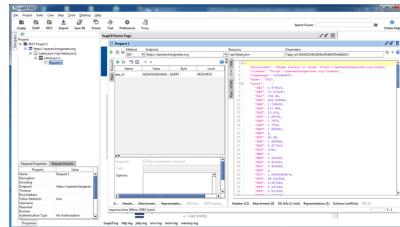
[Openexchangerates.org](https://openexchangerates.org/) (Uso educativo nc)

Figura 12. Cambio de la base a euros.



[Openexchangerates.org](https://openexchangerates.org/) (Uso educativo nc)

Figura 13. Uso de SoapUI.



Openexchangerates.org (Uso educativo nc)

Como vemos, este es un ejemplo de como somos capaces de acceder a servicios REST provistos por web services externos. Hay muchos más casos, solo es cuestión de investigar en Internet. En todo caso, el consumo de estos web services es algo complicado, pero gracias a ciertas librerías que veremos más adelante, el acceso y manipulación de datos será más sencillo. Concretamente veremos Retrofit.

3.- Consumo de un Web Service.

Un web service por tanto no es más que una forma de estructurar un conjunto de acciones que permiten a las aplicaciones interactuar con un servidor para consultar datos o modificarlos mediante los métodos del protocolo HTTP.

Para desarrollar la parte servidor (el web service en sí) puede usarse cualquier tecnología capaz de publicar información mediante HTTP (PHP, ASP.NET, Java + TOMAT, etc.). Para "usar" o "consumir" este web service desarrollado con REST es necesario establecer conexiones HTTP mediante las aplicaciones. Es decir, las aplicaciones se conectan a Internet (sin usar un navegador) y son capaces de descargar e incluso subir datos al servidor.

Web service en el servidor

Para desarrollar la parte de servidor existen herramientas web sobradamente preparadas que proporcionan un servicio web completamente fiable. Estará especialmente indicado la instalación del paquete **XAMPP**, **WAMP**, o **LAMP**, dependiendo de la distribución y del sistema operativo. **XAMPP**, por ejemplo, es un paquete de software libre que proporciona una serie de aplicaciones de alto nivel, entre las cuales destaca el servidor web Apache, el sistema de gestión de bases de datos MySQL y los intérpretes para lenguajes de script PHP y Perl. El nombre proviene del acrónimo de las aplicaciones anteriormente indicadas para cualquier sistema operativo: X (se refiere a cualquiera de los diferentes sistemas operativos más utilizados), Apache, MariaDB/MySQL, PHP, Perl.

A partir de la versión 5.6.15, **XAMPP** cambió la base de datos **MySQL** por **MariaDB**, la cual es una versión actualizada de MySQL con licencia **GPL**, y que surgió con el propósito de garantizar que los usuarios finales pudieran seguir haciendo uso de un sistema gestor de bases de datos avanzado, cosa que quedaba en entredicho por la amenaza de que **Oracle**, propietaria de MySQL pudiera comenzar a distribuir este SGBD bajo licencias no libres.

El programa proporciona un servidor web libre de altas prestaciones y muy fiable, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente, XAMPP está disponible para Windows, Linux, y OS X. El XAMPP podemos instalarlo tanto en un equipo local, como en un sistema remoto.

Los certificados de servidor

Uno de los aspectos en los que más está evolucionando Android, según avanza las APIs, es la seguridad. Esto ha hecho que desde hace tiempo las aplicaciones sólo tienen disponible conexiones a servidores HTTPS, es decir, a servidores que cuentan con un certificado que garantiza la encriptación de la comunicación. Conseguir esto en un servidor Apache es algo sencillo porque existen algunas formas de instalar lo que se denominan certificados autofirmados que permiten este cifrado en las comunicaciones. Sin embargo, las restricciones de

Android desde la API 28 han aumentado, y ahora las aplicaciones no tienen habilitado el acceso a ningún servicio web que no ofrezca HTTPS con un certificado avalado por una CA (autoridad certificadora) que asegure que el certificado pertenece a la empresa o persona que dice ser. Esto es necesario incluso cuando accedemos a direcciones IP de nuestra propia red o incluso a nuestra máquina anfitriona desde nuestro Android Studio.

Estas nuevas medidas complican la posibilidades de realizar aplicaciones que consuman servicios web desarrollados por nosotros mismos ya que es necesario instalar certificados **reales** en el servidor o crear entidades certificadoras, lo cual excede de los objetivos del módulo.

Autoevaluación

Sobre REST, indica las afirmaciones correctas...

- Puede usarse para conseguir datos de un servicio web e incluso subir datos al servidor con los métodos HTTP.

- No guarda estado.

- Devuelve siempre la información en formato JSON.

- En un sistema REST, cada recurso es direccionable mediante su URI.

[Mostrar retroalimentación](#)

Solución

1. Correcto
2. Correcto
3. Incorrecto
4. Correcto

3.1.- Ejercicio resuelto 3 (Consumo de un Web Service).

Ejercicio Resuelto

A partir del web service público con el que trabajamos en la unidad anterior (<https://openexchangerates.org/api/latest.json>) que devuelve un conjunto de divisas y los valores de cambio de las mismas con respecto al dólar (USD) vamos a realizar una aplicación capaz de hablar con este servicio para que calcule el número de divisas disponibles.

Para realizar el ejercicio, la aplicación solicitará al usuario el ID de API que debes conseguir tal como se explicó en el ejercicio anterior. En caso de que este no sea correcto, la aplicación devolverá un mensaje de error a través de TOAST. En caso de que pueda conectar, la aplicación simplemente contará las divisas que tiene disponibles y mostrará el JSON devuelto por el servicio web.

Este es el aspecto que tendrá la aplicación antes y después de establecer la conexión con el servicio.

Antes de establecer la conexión con el servicio.



Después de establecer la conexión con el servicio.



Nota: Puedes encontrar cualquier imagen relacionada con las divisas en Internet para usarla como recurso "drawable".

[Mostrar retroalimentación](#)

AndroidManifest.xml

```
1 | <uses-permission android:name="android.permission.INTERNET"/>
```

activity_main.xml

```
1 | <?xml version="1.0" encoding="utf-8"?>
2 | <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3 |   xmlns:app="http://schemas.android.com/apk/res-auto"
4 |   xmlns:tools="http://schemas.android.com/tools"
5 |   android:layout_width="match_parent"
6 |   android:layout_height="match_parent"
7 |   android:orientation="vertical"
8 |   tools:context="com.cidead.pmdm.ejercicio113.MainActivity">
9 |
10|   <TextView
11|     android:id="@+id/tvTitulo"
```

```
11     android:layout_width="match_parent"
12     android:layout_height="wrap_content"
13     android:gravity="center"
14     android:text="Acceso a openexchangerates.org"
15     android:textSize="20dp"
16     tools:text="API ID"
17     android:layout_margin="15dp"/>/
18
19 <ImageView
20     android:id="@+id/imgvLogin"
21     android:layout_width="match_parent"
22     android:layout_height="80dp"
23     app:srcCompat="@drawable/divisas"
24     android:layout_margin="20dp"/>/
25
26 <EditText
27     android:id="@+id/etAPIID"
28     android:layout_width="match_parent"
29     android:layout_height="wrap_content"
30     android:hint="Introduce tu API ID"/>/
31
32 <Button
33     android:id="@+id/btnLogin"
34     android:layout_width="match_parent"
35     android:layout_height="wrap_content"
36     android:text="Calcular divisas disponibles"
37     android:layout_margin="15dp"/>/
38
39 <TextView
40     android:id="@+id/tvDivisasEncontradas"
41     android:layout_width="match_parent"
42     android:layout_height="wrap_content"
43     android:gravity="center"
44     android:textSize="20dp"
45     android:layout_margin="15dp"/>/
46
47 <TextView
48     android:id="@+id/tvJsonSent"
49     android:layout_width="match_parent"
```

```
50     android:layout_height="wrap_content"
51     android:gravity="center"
52     android:textSize="12dp"
53     android:layout_margin="5dp"/>
54
55 </LinearLayout>
56
```

MainActivity.java

```
1 package com.cidead.pmdm.ejercicio113;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.util.Log;
8 import android.view.View;
9 import android.widget.Button;
10 import android.widget.EditText;
11 import android.widget.TextView;
12 import android.widget.Toast;
13
14 import org.json.JSONArray;
15 import org.json.JSONObject;
16
17 import java.io.DataOutputStream;
18 import java.net.HttpURLConnection;
19 import java.net.URL;
20 import java.util.Scanner;
21
22 public class MainActivity extends AppCompatActivity {
23     EditText etAPIID;
```

```
24     TextView tvDivisasEncontradas, tvJsonSent;
25     Button btnLogin;
26
27     @Override
28     protected void onCreate(Bundle savedInstanceState) {
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.activity_main);
31         etAPIID=(EditText)findViewById(R.id.etAPIID);
32         tvDivisasEncontradas =(TextView)findViewById(R.id.tvDivisasEncontradas);
33         tvJsonSent = (TextView)findViewById(R.id.tvJsonSent);
34         btnLogin=(Button)findViewById(R.id.btnLogin);
35         btnLogin.setOnClickListener(new View.OnClickListener() {
36
37             @Override
38             public void onClick(View view) {
39                 //Creamos un hilo para conectar probando la APIID
40                 Thread hilo=new Thread(){
41                     @Override
42                     public void run() {
43                         super.run();
44                         final String respuesta=enviarPost(etAPIID.getText().toString());
45                         runOnUiThread(new Runnable() { //El método runOnUiThread me permite trabajar con la i
46                             @Override
47                             public void run() {
48                                 //Limpiamos por si se repite la acción
49                                 tvDivisasEncontradas.setText("");
50                                 tvJsonSent.setText("");
51                                 //Toast.makeText(getApplicationContext(), respuesta, Toast.LENGTH_LONG).show();
52                                 int numElementosEncontrados = objetoJSON(respuesta);
53                                 if (numElementosEncontrados>0){
54                                     tvDivisasEncontradas.setText("Divisas disponibles:" + Integer.toString(nu
55                                         "Json de Respuesta:");
56                                     tvJsonSent.setText(respuesta);
57                                 }
58                                 else{
59                                     Toast.makeText(getApplicationContext(), "La consulta no devolvió divisas.
60                                 }
61                             }
62                         });
63                     });
64                 });
65             }
66         });
67     }
68 }
```

```
63         }
64     };
65     hilo.start(); //Con esto arrancamos el hilo
66 }
67 });
68 }
69
70 public String enviarPost(String app_id){
71     String parametros="app_id="+app_id; //Los literales corresponden a los parámetros que se pasan en la
72     HttpURLConnection conexion=null;
73     String respuesta="";
74     try{
75         //Tenemos que consultar la página enviándole los datos de mi aplicación (correo y password) como
76         URL url=new URL("https://openexchangerates.org/api/latest.json" + "?" + parametros);
77         conexion=(HttpURLConnection)url.openConnection();
78         conexion.setRequestMethod("POST");
79         conexion.setRequestProperty("Content-Length", "");
80         conexion.setDoOutput(true);
81         DataOutputStream dos=new DataOutputStream((conexion.getOutputStream()));
82         dos.close();
83         //Recogeremos la respuesta que nos devuelve el WebService y la recorremos para cargarla en la cad
84         Scanner lectura=new Scanner(conexion.getInputStream());
85         while (lectura.hasNextLine())
86             respuesta += lectura.nextLine();
87     }catch (Exception e){
88         Log.e("Error",e.toString());
89     }
90     return respuesta.toString();
91 }
92
93 //Creamos un método para contar cuantas divisas devuelve la consulta
94 //Pasaremos como parámetro la respuesta que da el métooo enviarPost() anterior
95 public int objetoJSON(String respuesta){
96     int numDivisas = 0;
97     try{
98         JSONObject jobject = new JSONObject(respuesta);
99         JSONObject divisas = jobject.getJSONObject("rates");
100        numDivisas = divisas.length();
101        //Log.d("Divisas", Integer.toString(numDivisas));
```

```
102     }catch (Exception e){
103         Log.e("Error", e.getMessage());
104     }
105     return numDivisas;
106 }
107 }
```



Este es el texto JSON devuelto por el servicio

```
1 {
2     "disclaimer": "Usage subject to terms: https://openexchangerates.org/terms",
3     "license": "https://openexchangerates.org/license",
4     "timestamp": 1654365613,
5     "base": "USD",
6     "rates": {
7         "AED": 3.6731,
8         "AFN": 89.162943,
9         "ALL": 112.223267,
10        "AMD": 447.269866,
11        "ANG": 1.805741,
12        "AOA": 426.1758,
13        "ARS": 120.67334,
14        "AUD": 1.387465,
15        "AWG": 1.8005,
16        "AZN": 1.7,
17        "BAM": 1.824519,
18        "BBD": 2,
19        "BDT": 89.20903,
20        "BGN": 1.821413,
21        "BHD": 0.37697,
```

```
22 "BIF": 2060.263278,  
23 "BMD": 1,  
24 "BND": 1.376531,  
25 "BOB": 6.898421,  
26 "BRL": 4.775,  
27 "BSD": 1,  
28 "BTC": 0.000033644322,  
29 "BTN": 77.768691,  
30 "BWP": 11.999496,  
31 "BYN": 3.38321,  
32 "BZD": 2.019646,  
33 "CAD": 1.25957,  
34 "CDF": 2004.038697,  
35 "CHF": 0.961757,  
36 "CLF": 0.029469,  
37 "CLP": 811.81,  
38 "CNH": 6.657479,  
39 "CNY": 6.6603,  
40 "COP": 3781.3558,  
41 "CRC": 684.124098,  
42 "CUC": 1,  
43 "CUP": 25.75,  
44 "CVE": 103.3,  
45 "CZK": 23.0091,  
46 "DJF": 178.372548,  
47 "DKK": 6.9387,  
48 "DOP": 55.207702,  
49 "DZD": 145.41,  
50 "EGP": 18.630186,  
51 "ERN": 15.000001,  
52 "ETB": 52.074408,  
53 "EUR": 0.932923,  
54 "FJD": 2.1368,  
55 "FKP": 0.800737,  
56 "GBP": 0.800737,  
57 "GEL": 2.965,  
58 "GGP": 0.800737,  
59 "GHS": 7.840332,  
60 "GIP": 0.800737,
```

```
61 "GMD": 54.15,
62 "GNF": 8870.767996,
63 "GTQ": 7.710011,
64 "GYD": 209.623421,
65 "HKD": 7.8448,
66 "HNL": 24.622892,
67 "HRK": 7.0142,
68 "HTG": 114.038788,
69 "HUF": 365.63,
70 "IDR": 14433.5,
71 "ILS": 3.335,
72 "IMP": 0.800737,
73 "INR": 77.6934,
74 "IQD": 1462.353565,
75 "IRR": 42350,
76 "ISK": 129.03,
77 "JEP": 0.800737,
78 "JMD": 154.180666,
79 "JOD": 0.7102,
80 "JPY": 130.845,
81 "KES": 116.978189,
82 "KGS": 79.525751,
83 "KHR": 4068.935217,
84 "KMF": 458.749854,
85 "KPW": 900,
86 "KRW": 1251.53,
87 "KWD": 0.306277,
88 "KYD": 0.834911,
89 "KZT": 435.599831,
90 "LAK": 13390.901035,
91 "LBP": 1514.968721,
92 "LKR": 361.204622,
93 "LRD": 151.999978,
94 "LSL": 15.510228,
95 "LYD": 4.786048,
96 "MAD": 9.878728,
97 "MDL": 19.067231,
98 "MGA": 4055.425659,
99 "MKD": 57.492275,
```

```
100 "MMK": 1855.118735,  
101 "MNT": 3105.3554,  
102 "MOP": 8.095189,  
103 "MRU": 36.541135,  
104 "MUR": 43.150001,  
105 "MVR": 15.435,  
106 "MWK": 816.25,  
107 "MXN": 19.54515,  
108 "MYR": 4.391,  
109 "MZN": 63.850001,  
110 "NAD": 15.49,  
111 "NGN": 415.814006,  
112 "NIO": 35.919846,  
113 "NOK": 9.43645,  
114 "NPR": 124.429799,  
115 "NZD": 1.535745,  
116 "OMR": 0.385029,  
117 "PAB": 1,  
118 "PEN": 3.713443,  
119 "PGK": 3.530509,  
120 "PHP": 52.909995,  
121 "PKR": 198.136548,  
122 "PLN": 4.28225,  
123 "PYG": 6861.311273,  
124 "QAR": 3.641,  
125 "RON": 4.611,  
126 "RSD": 109.539103,  
127 "RUB": 63.360006,  
128 "RWF": 1022.445353,  
129 "SAR": 3.750665,  
130 "SBD": 8.116969,  
131 "SCR": 14.330625,  
132 "SDG": 455.5,  
133 "SEK": 9.7671,  
134 "SGD": 1.3761,  
135 "SHP": 0.800737,  
136 "SLL": 12887.2,  
137 "SOS": 579.587715,  
138 "SRD": 21.22,
```

```
139     "SSP": 130.26,  
140     "STD": 22994,  
141     "STN": 23.15,  
142     "SVC": 8.767037,  
143     "SYP": 2512.53,  
144     "SZL": 15.512281,  
145     "THB": 34.29387,  
146     "TJS": 11.306259,  
147     "TMT": 3.51,  
148     "TND": 3.02,  
149     "TOP": 2.299263,  
150     "TRY": 16.4176,  
151     "TTD": 6.791608,  
152     "TWD": 29.375,  
153     "TZS": 2328,  
154     "UAH": 29.457448,  
155     "UGX": 3745.800949,  
156     "USD": 1,  
157     "UYU": 40.022165,  
158     "UZS": 11015.948404,  
159     "VES": 5.11435,  
160     "VND": 23192,  
161     "VUV": 114.629832,  
162     "WST": 2.612952,  
163     "XAF": 611.957277,  
164     "XAG": 0.04559972,  
165     "XAU": 0.00054022,  
166     "XCD": 2.70255,  
167     "XDR": 0.727495,  
168     "XOF": 611.957277,  
169     "XPD": 0.00050545,  
170     "XPF": 111.32731,  
171     "XPT": 0.00098188,  
172     "YER": 250.249937,  
173     "ZAR": 15.543525,  
174     "ZMW": 17.033219,  
175     "ZWL": 322  
176 }  
177 }
```



4.- Retrofit.

Retrofit es una tecnología basada en REST con la cual podremos desarrollar aplicaciones en Android que accedan a servicios web de una forma relativamente sencilla.

Para usar esta tecnología debemos incorporar dos librerías:

- ✓ La librería Retrofit para Android y Java (compatible con Kotlin) para hacer llamadas de red y obtener el resultado.
- ✓ Una librería que nos permita convertir el resultado devuelto “... parseando” de forma automática a su objeto; esto facilita mucho realizar peticiones a un API y procesar la respuesta.

Dispone de métodos que manejan API REST con contenidos en formato XML o en formato JSON. Retrofit puede manejar las clásicas primitivas de HTTP (GET, POST, PUT, HEAD...).

El resultado de la consulta al servicio web se incorpora en nuestra aplicación en una estructura de clases que representen los objetos de la respuesta esperada en formato POJO. Estas clases las debemos crear nosotros mismos con la misma estructura que tenga el resultado de la respuesta. Por ejemplo, si la respuesta es en formato JSON, deberemos construir una estructura de clases que represente los campos que nos interesan en la consulta de este JSON.

En este enlace puedes aprender más sobre [el cliente HTTP de Retrofit](#)  .

En este otro enlace puedes ver un ejemplo en Kotlin y algunas explicaciones de [cómo obtener datos de Internet desde un cliente Retrofit en una aplicación de Android](#)  .

4.1.- Ejercicio resuelto 4 (Retrofit).

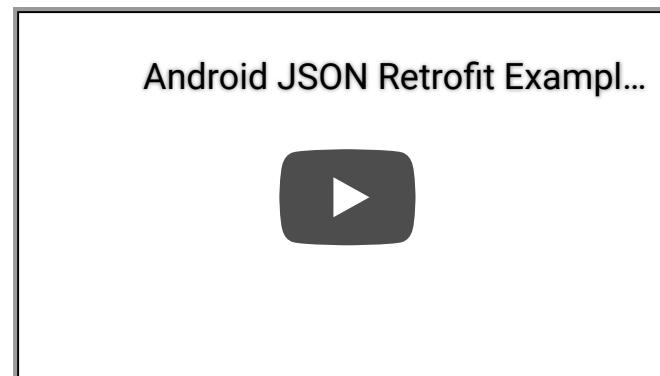
Ejercicio Resuelto



Para facilitar la tarea, vamos a tratar de realizar con la tecnología Retrofit un acceso al servicio web de openexchangerates.org del ejercicio visto en el apartado de "Consumo de un Web Service". El ejercicio devolverá el cambio de seis divisas en el momento actual (puedes elegir las que tú quieras) con respecto al dólar, ya que es la información que devuelve el Web Service.

Concretamente, el aspecto que buscaremos para la salida será similar al mostrado en la captura de pantalla de móvil.

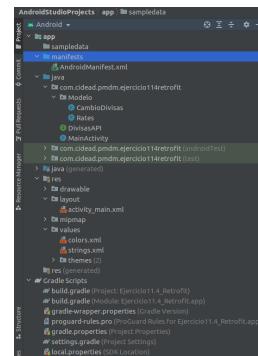
En el siguiente vídeo puedes encontrar un ejemplo de cómo construir este tipo de aplicaciones paso a paso con Android Studio.



[Descripción textual del video Ejemplo de Retrofit](#)

[Mostrar retroalimentación](#)

La estructura final de archivos implicados en el proyecto podría ser la siguiente:



[Google Developers](#). (Uso educativo nc)

build.gradle (Module app)

En primer lugar, incorporaremos en **build.gradle (Module app)** las dependencias:

```
1 dependencies {  
2     ...  
3     // Retrofit  
4     implementation "com.squareup.retrofit2:retrofit:2.9.0"  
5     // Retrofit with Converters gson  
6     implementation "com.squareup.retrofit2:converter-gson:2.9.0"  
7 }
```

AndroidManifest.xml

No debemos olvidar configurar el permiso de acceso a Internet en el fichero **AndroidManifest.xml**:

```
1 | <uses-permission android:name="android.permission.INTERNET"/>
```

activity_main.xml

La interfaz es sencilla y quedará definida en el archivo **activity_main.xml**:

```
1 | <?xml version="1.0" encoding="utf-8"?>
2 | <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3 |   xmlns:app="http://schemas.android.com/apk/res-auto"
4 |   xmlns:tools="http://schemas.android.com/tools"
5 |   android:layout_width="match_parent"
6 |   android:layout_height="match_parent"
7 |   android:orientation="vertical"
8 |   tools:context="com.cidead.pmdm.ejercicio114retrofit.MainActivity">
9 |
10|   <TextView
11|     android:id="@+id/tvTitulo"
12|     android:layout_width="match_parent"
13|     android:layout_height="wrap_content"
14|     android:gravity="center"
15|     android:text="Acceso a openexchangerates.org"
16|     android:textSize="20sp"
17|     tools:text="API ID"
18|     android:layout_margin="15dp"/>
19|
20|   <ImageView
21|     android:id="@+id/imgvLogin"
22|     android:layout_width="match_parent"
23|     android:layout_height="80dp"
```

```
24     app:srcCompat="@drawable/divisas"
25     android:layout_margin="20sp"/>"
26
27 <EditText
28     android:id="@+id/etAPIID"
29     android:layout_width="match_parent"
30     android:layout_height="wrap_content"
31     android:hint="Introduce tu API ID"/>
32
33 <Button
34     android:id="@+id/btnGetData"
35     android:layout_width="match_parent"
36     android:layout_height="wrap_content"
37     android:text="Obtener cambio divisas"
38     android:layout_margin="15sp"/>
39
40 <TextView
41     android:id="@+id/tvBase"
42     android:layout_width="match_parent"
43     android:layout_height="wrap_content"
44     android:gravity="center"
45     android:textSize="20sp"
46     android:layout_margin="15dp"
47     />
48
49 <TextView
50     android:id="@+id/tvConvert"
51     android:layout_width="match_parent"
52     android:layout_height="wrap_content"
53     android:inputType="textMultiLine"
54     android:gravity="center"
55     android:textSize="20sp"
56     android:layout_margin="15dp"
57     />
58
59 </LinearLayout>
```

Por otro lado, dentro de la carpeta modelos hemos creado una serie de clases auxiliares que representan el formato de los objetos que vamos a recibir del web service en formato JSON. El formato esperado en el momento de redactar este ejercicio es el mostrado a continuación. Si hubiera cambios tendrás que adaptar los nombres de los atributos de las clases usadas e incluso la estructura de los archivos para que sea similar a la estructura del JSON recibido.

Formato esperado

```
1  {
2      "disclaimer": "Usage subject to terms: https://openexchangerates.org/terms",
3      "license": "https://openexchangerates.org/license",
4      "timestamp": 1655024402,
5      "base": "USD",
6      "rates": {
7          "AED": 3.6731,
8          "AFN": 89.399125,
9          "ALL": 114.229019,
10         "AMD": 429.910968,
11         "ANG": 1.815174,
12         "AOA": 435.8264,
13         "ARS": 122.446573,
14         "AUD": 1.411943,
15         "AWG": 1.8005,
16         "AZN": 1.7,
17         "BAM": 1.859508,
18         "BBD": 2,
19         "BDT": 93.651536,
20         "BGN": 1.85946,
21         "BHD": 0.376878,
22         "BIF": 2071.295103,
23         "BMD": 1,
24         "BND": 1.391139,
25         ...
26         "ZWL": 322
27     }
```

```
28 |     }
    }
```

CambioDivisas.java

Clase CambioDivisas.java:

```
1 package com.cidead.pmdm.ejercicio114retrofit.Modelo;
2
3 import com.google.gson.annotations.Expose;
4 import com.google.gson.annotations.SerializedName;
5
6 public class CambioDivisas {
7     @SerializedName("timestamp")
8     @Expose
9     private String timestamp;
10    @SerializedName("base")
11    @Expose
12    private String base;
13    @SerializedName("rates")
14    @Expose
15    private Rates rates;
16
17    public String getTimestamp() {
18        return timestamp;
19    }
20
21    public void setTimestamp(String timestamp) {
22        this.timestamp = timestamp;
23    }
24
25    public String getBase() {
```

```
26     return base;
27 }
28
29 public void setBase(String base) {
30     this.base = base;
31 }
32
33 public Rates getRates() {
34     return rates;
35 }
36
37 public void setRates(Rates rates) {
38     this.rates = rates;
39 }
40
41 @Override
42 public String toString() {
43     return "CambioDivisas{" +
44         "timestamp='" + timestamp + '\'' +
45         ", base='" + base + '\'' +
46         ", rate='" + rates +
47         '}';
48 }
49 }
```

Rates.java

Clase Rates.java:

```
1 package com.cidead.pmdm.ejercicio114retrofit.Modelo;
2
3 import com.google.gson.annotations.Expose;
```

```
4 import com.google.gson.annotations.SerializedName;
5
6 public class Rates {
7     @SerializedName("EUR")
8     @Expose
9     private String EUR;
10
11    @SerializedName("AUD")
12    @Expose
13    private String AUD;
14
15    @SerializedName("CNY")
16    @Expose
17    private String CNY;
18
19    @SerializedName("GBP")
20    @Expose
21    private String GBP;
22
23    @SerializedName("ISK")
24    @Expose
25    private String ISK;
26
27    @SerializedName("MXN")
28    @Expose
29    private String MXN;
30
31    @SerializedName("PLN")
32    @Expose
33    private String PLN;
34
35    public String getEUR() {
36        return EUR;
37    }
38
39    public void setEUR(String EUR) {
40        this.EUR = EUR;
41    }
42
```

```
43     public String getAUD() {
44         return AUD;
45     }
46
47     public void setAUD(String AUD) {
48         this.AUD = AUD;
49     }
50
51     public String getCNY() {
52         return CNY;
53     }
54
55     public void setCNY(String CNY) {
56         this.CNY = CNY;
57     }
58
59     public String getGBP() {
60         return GBP;
61     }
62
63     public void setGBP(String GBP) {
64         this.GBP = GBP;
65     }
66
67     public String getISK() {
68         return ISK;
69     }
70
71     public void setISK(String ISK) {
72         this.ISK = ISK;
73     }
74
75     public String getMXN() {
76         return MXN;
77     }
78
79     public void setMXN(String MXN) {
80         this.MXN = MXN;
81     }
```

```
82
83     public String getPLN() {
84         return PLN;
85     }
86
87     public void setPLN(String PLN) {
88         this.PLN = PLN;
89     }
90
91     @Override
92     public String toString() {
93         return "Rates{"
94             + "EUR=" + EUR + '\n' +
95             ", AUD=" + AUD + '\n' +
96             ", CNY=" + CNY + '\n' +
97             ", GBP=" + GBP + '\n' +
98             ", ISK=" + ISK + '\n' +
99             ", MXN=" + MXN + '\n' +
100            ", PLN=" + PLN + '\n' +
101        '}';
102    }
103}
```

Configuración de la conexión al servidor web

Además, para configurar la conexión al servidor web, tenemos la interfaz que conecta con el servicio web que mediante el método GET solicita la aplicación llamada latest.json pasando el parámetro app_id. Si quisiéramos consumir este servicio desde un navegador, tendríamos que usar esta URL con tu ID de API https://openexchangerates.org/api/latest.json?app_id=8b2c865cad6xxxxxxxx..., pero con Retrofit debemos construirla de la siguiente manera:

```
1 package com.cidead.pmdm.ejercicio114retrofit;
2
3 import com.cidead.pmdm.ejercicio114retrofit.Modelo.CambioDivisas;
4
5 import retrofit2.Call;
6 import retrofit2.http.GET;
7 import retrofit2.http.Headers;
8 import retrofit2.http.Query;
9
10 public interface DivisasAPI {
11     String BASE_URL = "https://openexchangerates.org/api/";
12     @Headers("Content-type: application/json")
13     @GET("latest.json")
14     Call<CambioDivisas> getStuff(@Query("app_id") String key);
15 }
```

MainActivity.java

Por último la clase MainActivity.java:

```
1 package com.cidead.pmdm.ejercicio114retrofit;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.EditText;
10 import android.widget.TextView;
11 import android.widget.Toast;
```

```
12 import com.cidead.pmdm.ejercicio114retrofit.Modelo.CambioDivisas;
13 import com.cidead.pmdm.ejercicio114retrofit.Modelo.Rates;
14
15 import retrofit2.Call;
16 import retrofit2.Callback;
17 import retrofit2.Response;
18 import retrofit2.Retrofit;
19 import retrofit2.converter.gson.GsonConverterFactory;
20
21 public class MainActivity extends AppCompatActivity {
22     private static final String TAG = "MainActivity";
23     private static String BASE_URL = "https://openexchangerates.org/api/";
24
25     @Override
26     protected void onCreate(Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_main);
29         Button btnGetData = (Button) findViewById(R.id.btnGetData);
30         EditText etAPIID = (EditText) findViewById(R.id.etAPIID);
31         TextView tvBase = (TextView) findViewById(R.id.tvBase);
32         TextView tvConvert = (TextView) findViewById(R.id.tvConvert);
33
34         btnGetData.setOnClickListener(new View.OnClickListener() {
35
36             @Override
37             public void onClick(View view) {
38                 Retrofit retrofit = new Retrofit.Builder()
39                     .baseUrl(DivisasAPI.BASE_URL)
40                     .addConverterFactory(GsonConverterFactory.create())
41                     .build();
42
43                 DivisasAPI divisasAPI = retrofit.create(DivisasAPI.class);
44                 Call<CambioDivisas> call = divisasAPI.getStuff(etAPIID.getText().toString());
45
46                 call.enqueue(new Callback<CambioDivisas>() {
47
48                     @Override
49                     public void onResponse
50                         (Call<CambioDivisas> call, Response<CambioDivisas> response) {
```

```
51
52         tvBase.setText("Referencia 1" + response.body().getBase());
53         tvConvert.setText("EUR : " + response.body().getRates().getEUR() + "\n" +
54                             "AUD : " + response.body().getRates().getAUD() + "\n" +
55                             "CNY : " + response.body().getRates().getCNY() + "\n" +
56                             "ISK : " + response.body().getRates().getISK() + "\n" +
57                             "MXN : " + response.body().getRates().getMXN() + "\n" +
58                             "PLN : " + response.body().getRates().getPLN() + "\n");
59         //Trazas
60         //Log.d(TAG, "onResponse: Respuesta del servidor; " + response.toString());
61         //Log.d(TAG, "onResponse: Info recibida; " + response.body().toString());
62     }
63
64     @Override
65     public void onFailure(Call<CambioDivisas> call, Throwable t) {
66         //Log.e(TAG, "Error: " + "onFailure: Algo salio mal:" + t.getMessage());
67         Toast.makeText(MainActivity.this, "Algo salio mal:", Toast.LENGTH_SHORT).show();
68     }
69 });
70 }
71 });
72 }
73 }
74 }
```

5.- Obtención de imágenes a través de la web.

Existen varias librerías que permiten descargarnos imágenes procedentes de servidores web. Las más conocidas son Picasso, Glide, Universal Image Loader y Fresco. Además, incluso Retrofit dispone de métodos relacionados con la carga de imágenes como `getImage()`.

La librería Picasso se caracteriza por su sencillez y poco código necesario para poder cargar imágenes con buena calidad (por ejemplo la calidad de imagen que maneja Glide es algo peor, pero a diferencia de Picasso, Glide soporta formato GIF animados). Será necesario incluir la librería a través de la inclusión en el apartado de dependencias del `build.gradle` (Module: app) como hemos hecho con otras librerías, en el apartado de las dependencias.

Para saber más

Aquí puedes encontrar más detalles sobre la [librería Picasso](#) .

5.1.- Ejercicio resuelto 5 (Imágenes con Picasso).

Ejercicio Resuelto

Se pide, usando la librería Picasso, conectar con una API de la NASA que publica imágenes de Marte capturadas por el Rover Curiosity.

NOTA: Para acceder a esta imagen no es necesario usar una API_key aunque puedes conseguirla registrándote con tu correo electrónico.

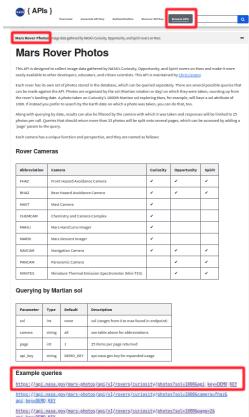
Las APIs están publicadas en esta página web: <https://api.nasa.gov/> .

Como puedes ver en la figura 1, debes ir a la página, pulsar el botón de la parte superior de "Browse APIs" y, a continuación, desplazarnos hasta abajo y encontrar los detalles de la API "Mars Rover Photos".

En la figura 2 vemos que pulsando al primer enlace (https://api.nasa.gov/mars-photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY) de "Example Queries" podemos ver la estructura del JSON devuelto por el servicio.

Desarrolla una App Android que muestre, mediante la librería Picasso, tres de estas imágenes servidas desde estas páginas. Para ello puedes conseguir los enlaces mostrados en alguna de las imágenes publicadas del JSON. El aspecto de la aplicación debe ser el mostrado en la figura 3.

Figura 1



[Google Developers](#) (Uso educativo nc)

Figura 2



[Google Developers](#) (Uso educativo nc)

Figura 3



Mostrar retroalimentación

AndroidManifest.xml

Añadimos el permiso al **AndroidManifest.xml**:

```
1 | <uses-permission android:name="android.permission.INTERNET"/>
```

build.gradle(:app)

En el archivo build.gradle(:app) añadimos la librería Picasso en el apartado de las dependencias:

```
1 dependencies {  
2     implementation 'androidx.appcompat:appcompat:1.4.2'  
3     ...  
4     //Añadimos la librería Picasso  
5     implementation 'com.squareup.picasso:picasso:2.8'  
6 }  
7 }
```

activity_main.xml

La interfaz vendrá dada por **activity_main.xml**:

```
1 <?xml version="1.0" encoding="utf-8"?>  
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
3     xmlns:app="http://schemas.android.com/apk/res-auto"  
4     xmlns:tools="http://schemas.android.com/tools"  
5     android:layout_width="match_parent"  
6     android:layout_height="match_parent"  
7     tools:context=".MainActivity">  
8  
9     <ImageView  
10        android:id="@+id/iv1"  
11        android:layout_width="162dp"  
12        android:layout_height="125dp"  
13        tools:layout_constraintTop_creator="1"  
14        android:layout_marginStart="8dp"  
15        android:layout_marginTop="8dp"
```

```
16     tools:layout_constraintLeft_creator="1"
17     app:layout_constraintLeft_toLeftOf="parent"
18     app:layout_constraintTop_toTopOf="parent"
19     android:layout_marginLeft="8dp" />
20
21 <TextView
22     android:id="@+id/tv1"
23     android:layout_width="wrap_content"
24     android:layout_height="wrap_content"
25     android:text="Nasa - Curiosity - Img0"
26     app:layout_constraintBottom_toBottomOf="parent"
27     app:layout_constraintHorizontal_bias="0.725"
28     app:layout_constraintLeft_toLeftOf="parent"
29     app:layout_constraintRight_toRightOf="parent"
30     app:layout_constraintTop_toTopOf="parent"
31     app:layout_constraintVertical_bias="0.081" />
32
33 <ImageView
34     android:id="@+id/iv2"
35     android:layout_width="162dp"
36     android:layout_height="125dp"
37     android:layout_marginLeft="8dp"
38     android:layout_marginStart="8dp"
39     android:layout_marginTop="180dp"
40     app:layout_constraintLeft_toLeftOf="parent"
41     app:layout_constraintTop_toTopOf="parent"
42     tools:layout_constraintLeft_creator="1"
43     tools:layout_constraintTop_creator="1" />
44
45 <TextView
46     android:id="@+id/tv2"
47     android:layout_width="wrap_content"
48     android:layout_height="wrap_content"
49     android:text="Nasa - Curiosity - Img10"
50     app:layout_constraintBottom_toBottomOf="parent"
51     app:layout_constraintHorizontal_bias="0.748"
52     app:layout_constraintLeft_toLeftOf="parent"
53     app:layout_constraintRight_toRightOf="parent"
54     app:layout_constraintTop_toTopOf="parent"
```

```
55     app:layout_constraintVertical_bias="0.324" />
56
57     <ImageView
58         android:id="@+id/iv3"
59         android:layout_width="162dp"
60         android:layout_height="125dp"
61         android:layout_marginLeft="8dp"
62         android:layout_marginStart="8dp"
63         android:layout_marginTop="337dp"
64         app:layout_constraintLeft_toLeftOf="parent"
65         app:layout_constraintTop_toTopOf="parent"
66         tools:layout_constraintLeft_creator="1"
67         tools:layout_constraintTop_creator="1" />
68
69     <TextView
70         android:id="@+id/tv3"
71         android:layout_width="wrap_content"
72         android:layout_height="wrap_content"
73         android:text="Nasa - Curiosity - Img20"
74         app:layout_constraintBottom_toBottomOf="parent"
75         app:layout_constraintHorizontal_bias="0.748"
76         app:layout_constraintLeft_toLeftOf="parent"
77         app:layout_constraintRight_toRightOf="parent"
78         app:layout_constraintTop_toTopOf="parent"
79         app:layout_constraintVertical_bias="0.561" />
80
81     </androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

Finalmente, el archivo **MainActivity.java**:

```
1 package com.cidead.pmdm.ejercicio115;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.widget.ImageView;
7
8 import com.squareup.picasso.Picasso;
9
10 public class MainActivity extends AppCompatActivity {
11     ImageView iv1;
12     ImageView iv2;
13     ImageView iv3;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19         iv1=(ImageView)findViewById((R.id.iv1));
20         iv2=(ImageView)findViewById((R.id.iv2));
21         iv3=(ImageView)findViewById((R.id.iv3));
22         //URL de la API https://api.nasa.gov/mars-photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO
23
24         Picasso.get()
25             .load("https://mars.nasa.gov/msl-raw-images/proj/msl/redops/ods/surface/sol/01000/opgs/edr/fc")
26             .resize(500, 500)
27             .centerCrop()
28             .into(iv1);
29         Picasso.get()
30             .load("https://mars.nasa.gov/msl-raw-images/msss/01000/mcam/1000MR0044631270503687E03_DXXX.jp")
31             .resize(500, 500)
32             .centerCrop()
33             .into(iv2);
34         Picasso.get()
35             .load("https://mars.nasa.gov/msl-raw-images/msss/01000/mcam/1000MR0044631220503682E01_DXXX.jp")
36             .resize(500, 500)
37             .centerCrop()
38             .into(iv3);
39         /* Otra forma de llamar aun mas corta
```

```
40     Picasso.get().load("https://mars.nasa.gov/msl/raw-images/proj/msl/redops/ods/surface/sol/01000/opgs/e  
41     }  
42  
43 }
```

