JSON

paco@portadaalta.es

Índice

¿Qué es JSON?

JSON (**J**ava**S**cript **O**bject **N**otation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

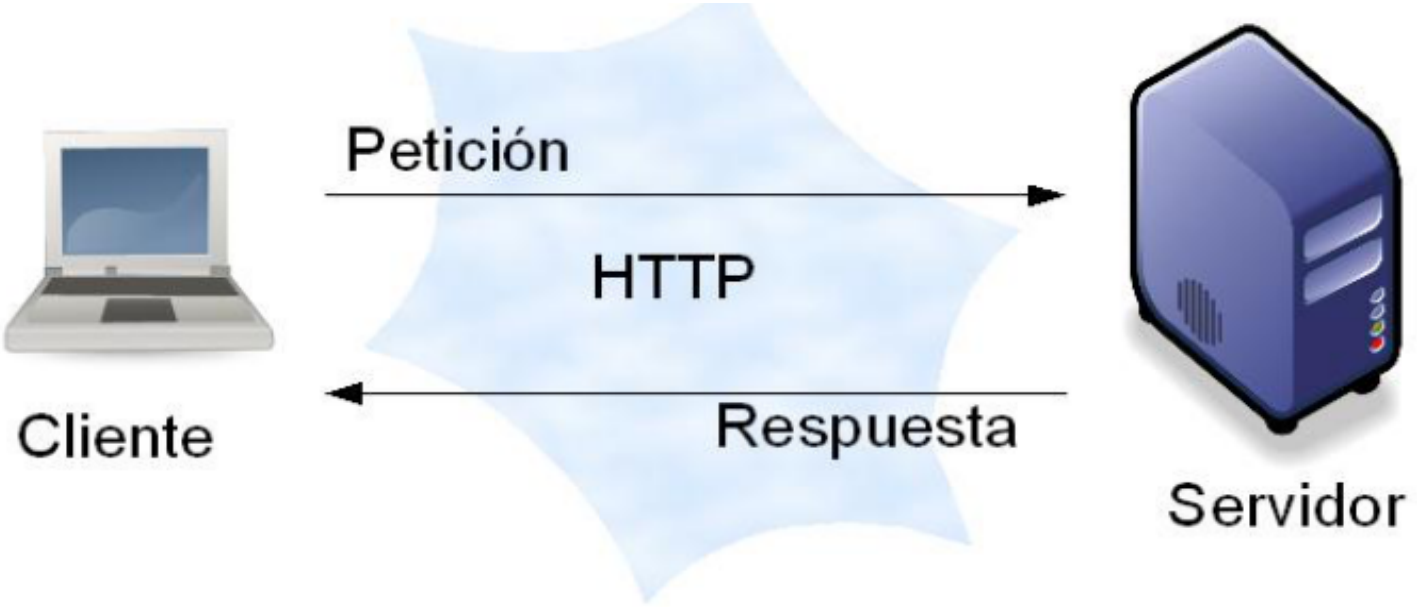
Está basado en un subconjunto del lenguaje de **programación** JavaScript

Características

- JSON es un formato independiente del lenguaje, pero utiliza convenciones y conceptos usados en muchos lenguajes: C, C++, C#, Java, etc.
- JSON usa texto plano, fácil de leer, escribir y generar.
- Es ligero.
- Por defecto su codificación es UTF-8.
- Su tipo MIME es application/json.

¿Para qué sirve?

Las características de JSON hacen que sea adecuado para el intercambio de datos.



Estructuras

JSON está constituido por dos estructuras:

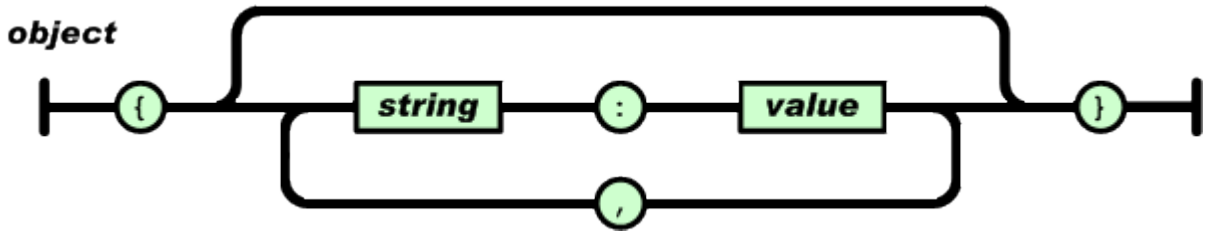
- **Una colección de pares de nombre/valor**. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash o lista de claves.

- **Una lista ordenada de valores**. En la mayoría de los lenguajes, esto se implementa como arrays, vectores, listas o secuencias.

Estas son estructuras universales; casi todos los lenguajes de programación las soportan de una forma u otra.

Objetos

- Un objeto es un conjunto desordenado de pares nombre/valor.
- Un objeto comienza con **{** (llave de apertura) y termine con **}** (llave de cierre).
- Cada nombre es seguido por **:** (dos puntos) y los pares nombre/valor están separados por **,** (coma).



Ejemplo de objeto

```
{
  "Libro": {
                "Titulo": "Sopa de pollo para el alma",
                "Autor": "Jack Canfield, Mark Victor Hansen",
                "Editorial": "Alba",
                "Precio": 16.22,
                "FechaPublicacion": "20-junio-2013"
  }
}
```

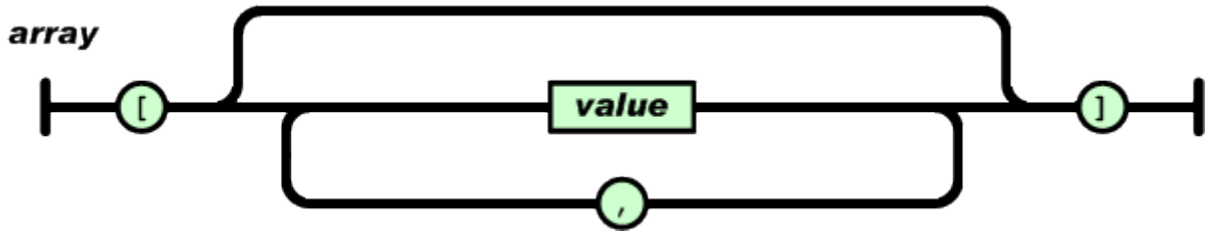
Ejercicio:
¿Objeto con los datos de un contacto (nombre, dirección, ciudad, email y teléfono)?

En el teléfono se almacenarán el de casa, el móvil y el del trabajo.

Arrays

- Un array es una colección de valores.

- Un array comienza con **[** (corchete izquierdo) y termina con **]** (corchete derecho). Los valores se separan por , (coma).



Ejemplo de Array

```
{
  "Frutas": [
              { "Nombre":"Manzana" , "cantidad":10 },
              { "Nombre":"Pera" , "cantidad":20 },
              { "Nombre":"Naranja" , "cantidad":30 }
            ]
}
```

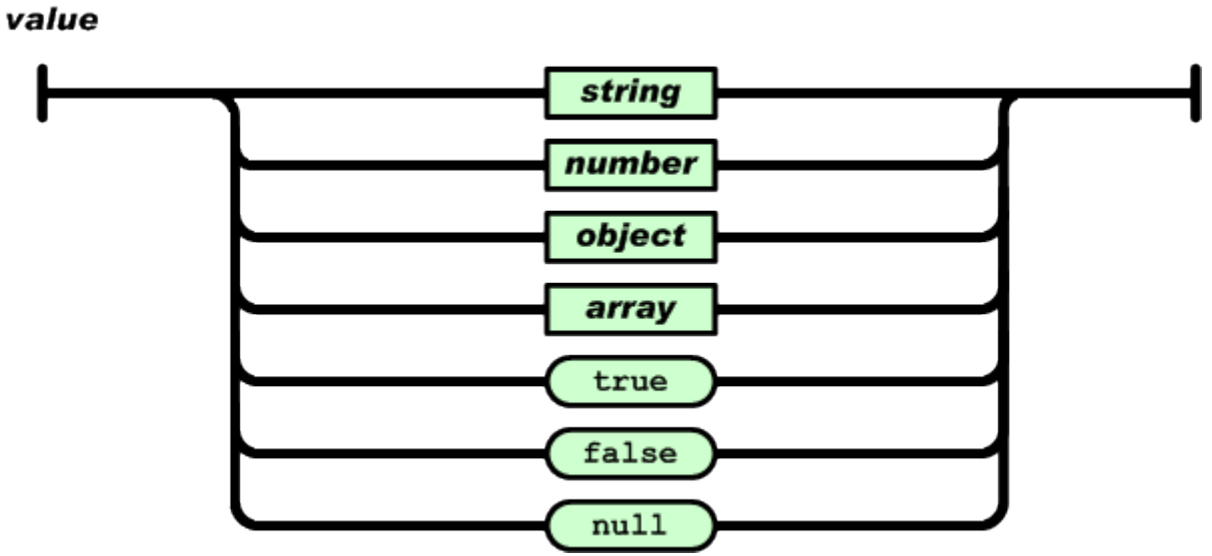
Ejercicio:
¿Objeto con los datos de varios contactos (nombre, dirección, ciudad, email y teléfono)?

En el teléfono se almacenarán el de casa, el móvil y el del trabajo.

Valores

- Un valor puede ser una cadena de caracteres con comillas dobles, un número, true o false, null, un objeto o un array.

- Estas estructuras pueden anidarse.



Herramientas

Extensión para Firefox

Extensión para Chrome

JSON Viewer

Code Beautify

JSON Lint

JSON versus XML
Similitudes

- Ambos son legibles por los humanos
- Tienen una sintaxis simple
- Son independientes del lenguaje de programación

JSON: The Fat-Free Alternative to XML

JSON versus XML
Diferencias

- Sintaxis diferente
- JSON es más compacto
- JSON puede incluir Arrays
- XML tiene espacios de nombres (Namespaces)
- XML tiene una estructura fácilmente comprobable (DTD, XML Schema)
- XML puede visualizarse (CSS)
- XML puede procesarse (XSL)

JSON versus XML
Ejemplo

XML

```
<listado>
  <persona>
     <nombre>Juan</nombre>
     <apellidos>Palomo</apellidos>
     <fecha>20/10/1980</fecha>
  </persona>
  <persona>
    <nombre>Pepe</nombre>
    <apellidos>Gotera</apellidos>
    <fecha>7/8/1990</fecha>
  </persona>
</listado>
```

JSON

```
{"listado": [
                {
                "nombre":"Juan",
                "apellidos":"Palomo",
                "fecha": "20/10/1980"
                },
                {
                "nombre": "Pepe",
                "apellidos": "Gotera",
                "fecha": "7/8/1990"
                }
              ]
}
```

JSON y Android

JSON	Android
string	String
number	Number
true | false	Boolean
null	null
object	JSONObject
array	JSONArray

JSON y Android

Clase JSONObject

JSONObject(String json)	Creates a new JSONObject with name/value mappings from the JSON string.
getInt(String name)	Returns the value mapped by name if it exists and is an int or can be coerced to an int, or throws otherwise.
getString(String name)	Returns the value mapped by name if it exists, coercing it if necessary, or throws if no such mapping exists.
getJSONObject(String name)	Returns the value mapped by name if it exists and is a JSONObject, or throws otherwise.
getJSONArray(String name)	Returns the value mapped by name if it exists and is a JSONArray, or throws otherwise.
put(String name, int value)	Maps name to value, clobbering any existing name/value mapping with the same name.
toString()	Encodes this object as a compact JSON string.

Clase JSONArray

JSONArray(String json)	Creates a new JSONArray with values from the JSON string.
toString(int indentSpaces)	Encodes this array as a human readable JSON string for debugging

Ejercicios

Primitiva

Lista de contactos

Lista de contactos (con Gson)

Creación de JSON

Retrofit

Ejercicio: Primitiva

Almacenar en un fichero en formato JSON los resultados de los últimos sorteos de la lotería primitiva y colocarlo en un servidor web (local o en Internet).

Crear una aplicación que descargue (usando Volley) los datos del fichero *primitiva.json* y muestre los resultados en pantalla.

Ejercicio: Primitiva

Ejercicio: Primitiva


```
{
  "info": "Primitiva",
  "sorteo": [
    {
      "id": "1",
      "numero1": 1,
      "numero2": 4,
      "numero3": 15,
      "numero4": 24,
      "numero5": 43,
      "numero6": 48,
      "complementario": 28,
      "reintegro": 9,
      "fecha": "06-12-2012",
      "numeroSorteo": 53
    },
    {

    }
  ]
}
```

Ejercicio: Primitiva

primitiva.json

Ejercicio: Primitiva

build.gradle:

```
useLibrary 'org.apache.http.legacy'

    compile 'cz.msebera.android:httpclient:4.4.1.1'
    compile 'com.squareup.okhttp3:okhttp:3.4.1'
    compile 'com.android.volley:volley:1.0.0'
```

Permisos en el manifiesto:

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

Añadir las clases:

MySingleton
OkHttp3Stack

Ejercicio: Primitiva

Request Json with Volley

```
TextView mTxtDisplay;
ImageView mImageView;
mTxtDisplay = (TextView) findViewById(R.id.txtDisplay);
String url = "http://my-json-feed";

JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {

    @Override
    public void onResponse(JSONObject response) {
        mTxtDisplay.setText("Response: " + response.toString());
    }
}, new Response.ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
        // TODO Auto-generated method stub

    }
});

// Access the RequestQueue through your singleton class.
MySingleton.getInstance(this).addToRequestQueue(jsObjRequest);
```

Ejercicio: Primitiva

```
public static String analizarPrimitiva(JSONObject texto) throws JSONException {
    JSONArray jsonContenido;
    String tipo;
    StringBuilder cadena = new StringBuilder();

    tipo = texto.getString("info");
    jsonContenido = new JSONArray(texto.getString("sorteo"));
    cadena.append("Sorteos de la Primitiva:" + "\n");

    for (int i = 0; i < jsonContenido.length(); i++) {



    }
    return cadena.toString();
}
```

Ejercicio: Primitiva

```
public class Primitiva extends Activity implements View.OnClickListener{
    public static final String TAG = "MyTag";
    public static final String WEB = "http://192.168.1.20/acceso/primitiva.json";
    //public static final String WEB = "https://portadaalta.mobi/acceso/primitiva.json";

    Button mButton;
    TextView mTextView;
    RequestQueue mRequestQueue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_primitiva);

        mButton = (Button) findViewById(R.id.button);
        mButton.setOnClickListener(this);
        mTextView = (TextView) findViewById(R.id.textView);

        mRequestQueue = MySingleton.getInstance(this.getApplicationContext()).getRequestQueue();
    }

    @Override
    public void onClick(View view) {
        if (view == mButton)
            descarga();
    }
```

Ejercicio: Primitiva

```
    private void descarga() {









        // Set the tag on the request.
        jsObjRequest.setTag(TAG);
        // Set retry policy
        jsObjRequest.setRetryPolicy(new DefaultRetryPolicy(3000, 1, 1));
        // Add the request to the RequestQueue.
        mRequestQueue.add(jsObjRequest);

    }

    @Override
    protected void onStop() {
        super.onStop();

        if (mRequestQueue != null) {
            mRequestQueue.cancelAll(TAG);
        }
    }
}
```

Ejercicio: Primitiva

```
    @Override
    public void onErrorResponse(VolleyError error) {
        // TODO Auto-generated method stub
        StringBuilder message = new StringBuilder();
        //Toast.makeText(getApplicationContext(), "Error: " + error.getMessage(),
    Toast.LENGTH_SHORT).show();
        NetworkResponse response = error.networkResponse;

        if(response != null && response.data != null)
            message.append("Error: " + response.statusCode);
        else {
            String errorMessage = error.getClass().getSimpleName();
            if(!TextUtils.isEmpty(errorMessage))
                message.append("Error: " + errorMessage);

            else
                message.append("Error de conexión con Volley");
        }
        showMessage(message.toString());
    }


private void showMessage(String s) {
    Toast.makeText(this, s, Toast.LENGTH_SHORT).show();
}
```

Ejercicio: Lista de contactos

Crear una aplicación que muestre una lista con los nombres de los contactos almacenados en formato JSON en un fichero situado en el servidor web local (o en Internet).

Cada contacto contendrá los campos: nombre, dirección, email y teléfono. En el teléfono se podrán guardar tres valores (casa, móvil y trabajo).

Cuando se pulse en un nombre de la lista, se mostrará su móvil en una notificación.

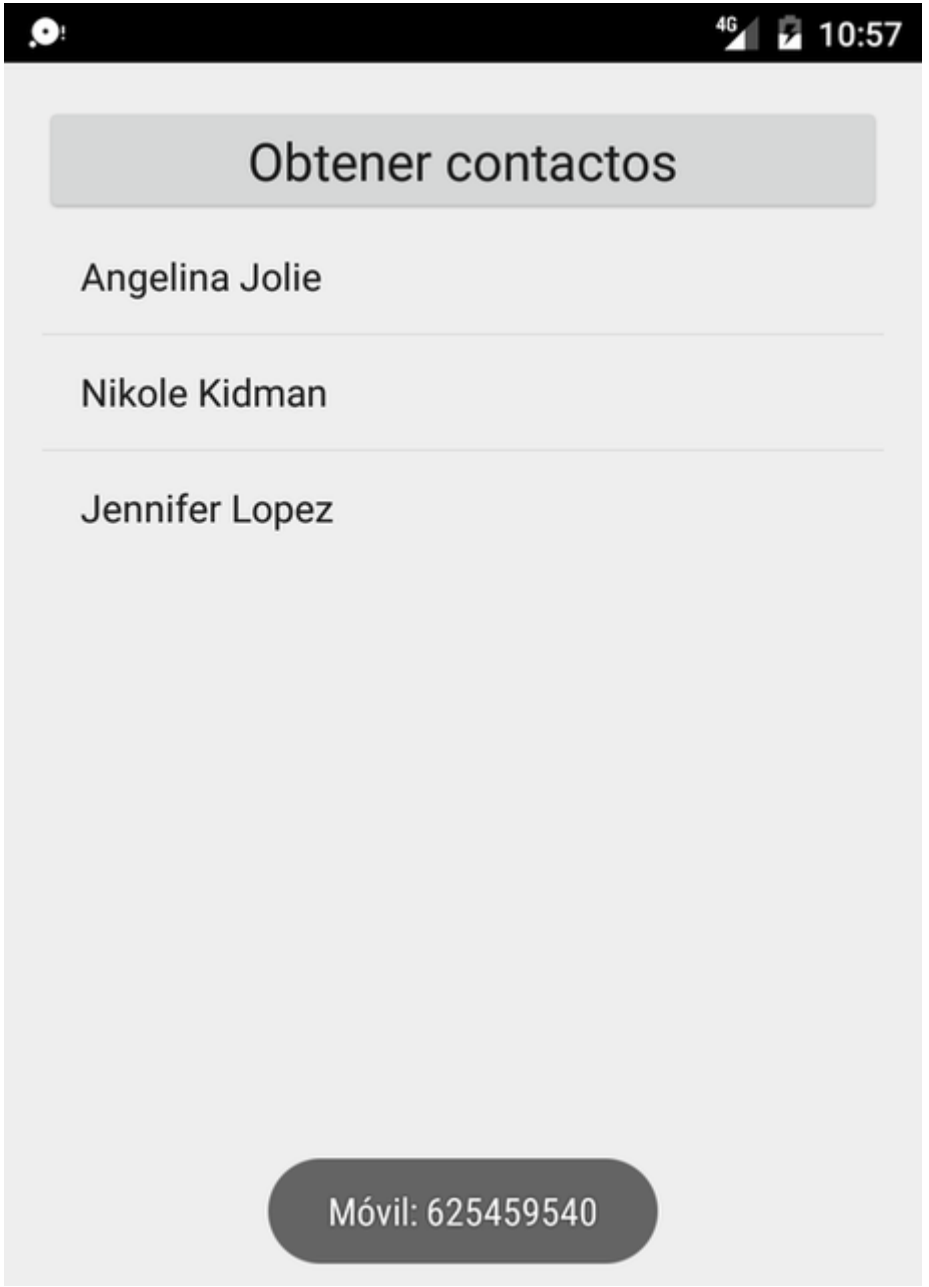
Ejercicio: Lista de contactos

Ejercicio: Lista de contactos

contactos.json

```
{
"contactos":[
        {
        "nombre": "Angelina Jolie",
        "direccion": "calle los remedios, 29007, malaga",
        "email": "angie@gmail.com",
        "telefono": {
                "casa": "952245407",
                "movil": "603459530",
                "trabajo": "951230246"
                }
        }
        ,
        {
        "nombre": "Nikole Kidman",
        "direccion": "calle hilera, 29010, malaga",
        "email": "nikole@gmail.com",
        "telefono": {
                "casa": "952233407",
                "movil": "625459540",
                "trabajo": "951230246"
                }
        }
        ,
        {
        "nombre": "Jennifer Lopez",
        "direccion": "calle larios, 29002, malaga",
        "email": "jenny@gmail.com",
        "telefono": {
                "casa": "951065407",
                "movil": "628659550",
                "trabajo": "952002146"
                }
        }
        ]
}
```

Ejercicio: Lista de contactos



Ejercicio: Lista de contactos

Mapas mentales

Ejercicio: Lista de contactos

Dependencias en Gradle:

```
dependencies {

    compile 'cz.msebera.android:httpclient:4.4.1.1'
    compile 'com.loopj.android:android-async-http:1.4.9'
    compile 'com.google.code.gson:gson:2.8.2'
}
```

Permisos en el manifiesto:

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

Ejercicio: Lista de contactos

```
public class Telefono {
	private String casa;
	private String movil;
	private String trabajo;

	public String getCasa() {
		return casa;
	}
	public void setCasa(String casa) {
		this.casa = casa;
	}
	public String getMovil() {
		return movil;
	}
	public void setMovil(String movil) {
		this.movil = movil;
	}
	public String getTrabajo() {
		return trabajo;
	}
	public void setTrabajo(String trabajo) {
		this.trabajo = trabajo;
	}
}
```

Ejercicio: Lista de contactos

```
public class Contacto {
	private String nombre;
	private String direccion;
	private String email;
	private Telefono telefono;

	public String getNombre() { return nombre; }
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getDireccion() { return direccion; }
	public void setDireccion(String direccion) {
		this.direccion = direccion;
	}
	public String getEmail() { return email; }
	public void setEmail(String email) {
		this.email = email;
	}
	public Telefono getTelefono() { return telefono; }
	public void setTelefono(Telefono t) {
		this.telefono = t;
	}
	public String toString() {
		return nombre;
	}
}
```

Ejercicio: Lista de contactos

```
public static ArrayList<Contacto> analizarContactos(JSONObject respuesta) throws JSONException {
        JSONArray jAcontactos;
        JSONObject jOcontacto, jOtelefono;
        Contacto contacto;
        Telefono telefono;
        ArrayList<Contacto> personas;

        // añadir contactos (en JSON) a personas









    return personas;
}
```

Ejercicio: Lista de contactos

```
public class ListaContactos extends Activity implements OnClickListener, OnItemClickListener {
    public static final String WEB = "http://192.168.1.20/acceso/contactos.json";
    //public static final String WEB = "https://portadaalta.mobi/acceso/contactos.json";
    Button boton;
    ListView lista;
    ArrayList<Contacto> contactos;
    ArrayAdapter<Contacto> adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lista_contactos);

        boton = (Button) findViewById(R.id.button);
        boton.setOnClickListener(this);
        lista = (ListView) findViewById(R.id.listView);
        lista.setOnItemClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (v == boton)
            descarga(WEB);
    }
```

Ejercicio: Lista de contactos

```
    //usar JsonHttpResponseHandler()
    private void descarga(String web) {
        final ProgressDialog progreso = new ProgressDialog(this);

        RestClient.get(web, new JsonHttpResponseHandler() {
                @Override
                public void onStart() {
                    super.onStart();
                    progreso.setProgressStyle(ProgressDialog.STYLE_SPINNER);
                    progreso.setMessage("Conectando . . .");
                    progreso.setCancelable(true);
                    progreso.show();
                }


            } );
    }
    private void mostrar() {
        if (contactos != null)
            if (adapter == null) {
                adapter = new ArrayAdapter<Contacto>(this, android.R.layout.simple_list_item_1, contactos);
                lista.setAdapter(adapter);
            } else {
                adapter.clear();
                adapter.addAll(contactos);
            }
        else
            Toast.makeText(getApplicationContext(), "Error al crear la lista", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(this, "Móvil: " + contactos.get(position).getTelefono().getMovil(), Toast.LENGTH_SHORT).show();
    }
}
```

Gson

Gson (también conocido como Google Gson) es una biblioteca de código abierto para el lenguaje de programación Java y para Android que permite la conversión entre objetos Java y su representación en notación JSON de una manera sencilla, simplemente invocando los métodos **toJson()** o **fromJson()**.

Gson

Ejemplo de uso: Gson en Wikipedia

Proyecto en GitHub: google-gson

Documentación de Gson: Javadoc

Tutorial: Gson – Getting started

Dependencias en Gradle:

```
compile 'com.google.code.gson:gson:2.8.2'
```

Ejercicio: Lista de contactos con Gson

Ejercicio: Lista de contactos con Gson

contacts.json

```
{
"contacts":[
        {
        "name": "Halle Berry",
        "address": "calle los remedios, 29007, malaga",
        "email": "halle@gmail.com",
        "phone":{
                "home": "952245407",
                "mobile": "603459530",
                "work": "951230246"
                }
        }
        ,
        {
        "name": "Julia Roberts",
        "address": "calle hilera, 29010, malaga",
        "email": "julia@gmail.com",
        "phone":{
                "home": "952233407",
                "mobile": "625459540",
                "work": "951230246"
                }
        }
        ,
        {
        "name": "Scarlett Johansson",
        "address": "calle larios, 29002, malaga",
        "email": "Scarlett@gmail.com",
        "phone":{
                "home": "951065407",
                "mobile": "628659550",
                "work": "952002146"
                }
        }
        ]
}
```

Ejercicio: Lista de contactos con Gson

¿Clases necesarias para usar Gson?

Ejercicio: Lista de contactos con Gson

```
public class ListaContactosGson extends Activity implements OnClickListener, OnItemClickListener {
    public static final String WEB = "http://192.168.1.20/acceso/contacts.json";
    //public static final String WEB = "https://portadaalta.mobi/acceso/contacts.json";
    Button boton;
    ListView lista;
    ArrayAdapter<Contact> adapter;
    Person person;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lista_contactos);

        boton = (Button) findViewById(R.id.button);
        boton.setOnClickListener(this);
        lista = (ListView) findViewById(R.id.listView);
        lista.setOnItemClickListener(this);
        contacts = new ArrayList<Contact>();
    }

    @Override
    public void onClick(View v) {
        if (v == boton)
            descarga(WEB);
    }
```

Ejercicio: Lista de contactos con Gson

```
private void descarga(String web) {
      final ProgressDialog progreso = new ProgressDialog(this);

      RestClient.get(WEB, new JsonHttpResponseHandler() {
            @Override
            public void onStart() {
               super.onStart();
               progreso.setProgressStyle(ProgressDialog.STYLE_SPINNER);
               progreso.setMessage("Conectando . . .");
               progreso.setCancelable(true);
               progreso.show();
            }



         });
   }
   private void mostrar() {
      if (person != null) {
         if (adapter == null) {
            adapter = new ArrayAdapter<Contact>(this, android.R.layout.simple_list_item_1, person.getContacts());
            lista.setAdapter(adapter);
         }
         else {
            adapter.clear();
            adapter.addAll(person.getContacts());
         }
      } else
         Toast.makeText(getApplicationContext(), "Error al crear la lista", Toast.LENGTH_SHORT).show();
   }
   @Override
   public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
      Toast.makeText(this, "Móvil: " + person.getContacts().get(position).getPhone().getMobile(), Toast.LENGTH_SHORT).show();
   }
}
```

Ejercicio: Lista de contactos con Gson

Posibles errores:

- Está caído el servidor web

- No se encuentra el fichero (error 404)

- El documento Json no es correcto

- El documento Json no tiene la sintaxis esperada

Ejercicio: Crear JSON

Crear un documento, el cual se guardará en memoria externa, en formato JSON que contenga los titulares, enlaces, descripciones y fechas de creación de las noticias obtenidas en el feed del blog de Alejandro Suárez.

Ejercicio: Crear JSON


```
{
  ▾ rss: {
      ▾ titulares: [
          ▾ {
              fecha: "Thu, 19 Nov 2015 18:33:14 +0000",
              titular: "Entrevista en Generació digital de Catalunya Radio",
              "descripción": "",
              enlace: http://www.alejandrosuarez.es/2015/11/entrevista-en-generacio-digital-de-catalunya-radio/
            },
          ▾ {
              fecha: "Thu, 19 Nov 2015 18:31:35 +0000",
              titular: "Entrevista en RAC1",
              descripcion: "",
              enlace: http://www.alejandrosuarez.es/2015/11/entrevista-en-rac1/
            },
          ▾ {
              fecha: "Thu, 19 Nov 2015 18:28:24 +0000",
              titular: "Entrevista en 8 tv, para el programa 8 al día.",
              descripcion: https://vimeo.com/146221192,
              enlace: http://www.alejandrosuarez.es/2015/11/entrevista-en-8-tv-para-el-programa-8-al-dia/
            }
        ],
        web: http://www.alejandrosuarez.es/,
        link: http://www.alejandrosuarez.es/feed/
    }
}
```

Ejercicio: Crear JSON

```
public static void escribirJSON(ArrayList<Noticia> noticias, String fichero) throws IOException, JSONException
{
   OutputStreamWriter out;
   File miFichero;
   JSONObject objeto, rss;
   JSONArray lista;

   miFichero = new File(Environment.getExternalStorageDirectory().getAbsolutePath(), fichero);
   out = new FileWriter(miFichero);

   //crear objeto JSON
   objeto = new JSONObject();
   objeto.put("web", "http://www.alejandrosuarez.es/");
   objeto.put("link", "http://www.alejandrosuarez.es/feed/");
   lista = new JSONArray();









   out.write(rss.toString(4)); //tabulación de 4 caracteres
   out.flush();
   out.close();
   Log.i("info", objeto.toString());
}
```

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Ejercicio: Crear JSON

```
public class CreacionJson extends Activity implements OnClickListener {
    public static final String WEB = "http://www.alejandrosuarez.es/feed/";
    //public static final String WEB = "http://192.168.2.110/acceso/alejandro.rss";
    public static final String RESULTADO_JSON = "resultado.json";
    public static final String RESULTADO_GSON = "resultado_gson.json";
    public static final String TEMPORAL = "alejandro.xml";
    ArrayList<Noticia> noticias;
    Button boton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_escribir_json);

        boton = (Button) findViewById(R.id.button);
        boton.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        if (v == boton)
            descarga(WEB, TEMPORAL);
    }
    //obtener el rss y escribir los ficheros
    private void descarga(String web, String temporal) {



    }
}
```

Ejercicio: Crear JSON con Gson

Crear un documento, el cual se guardará en memoria externa, en formato JSON que contenga los titulares, enlaces, descripciones y fechas de creación de las noticias obtenidas en el feed del blog de Alejandro Suárez.

Se utilizará la biblioteca Gson.

Ejercicio: Crear JSON con Gson


```
{
  "feed": "http://www.alejandrosuarez.es/feed/",
  "titles": [
    {
      "description": "",
      "link": "http://www.alejandrosuarez.es/2015/11/entrevista-en-generacio-digital-de-catalunya-radio/",
      "pubDate": "Thu, 19 Nov 2015 18:33:14 +0000",
      "title": "Entrevista en GeneraciÃ³ digital de Catalunya Radio"
    },
    {
      "description": "",
      "link": "http://www.alejandrosuarez.es/2015/11/entrevista-en-rac1/",
      "pubDate": "Thu, 19 Nov 2015 18:31:35 +0000",
      "title": "Entrevista en RAC1"
    },
    {
      "description": "https://vimeo.com/146221192",
      "link": "http://www.alejandrosuarez.es/2015/11/entrevista-en-8-tv-para-el-programa-8-al-dia/",
      "pubDate": "Thu, 19 Nov 2015 18:28:24 +0000",
      "title": "Entrevista en 8 tv, para el programa 8 al dÃ•a."
    }
  ],
  "web": "http://www.alejandrosuarez.es/"
}
```

Ejercicio: Crear JSON con Gson

```
public class Titular {

	private String web;
	private String feed;
	@SerializedName("titles")
	private List<Noticia> titulares;

	public String getWeb() {
		return web;
	}
	public void setWeb(String web) {
		this.web = web;
	}
	public String getFeed() {
		return feed;
	}
	public void setFeed(String feed) {
		this.feed = feed;
	}
	public List<Noticia> getTitulares() {
		return titulares;
	}
	public void setTitulares(List<Noticia> titulares) {
		this.titulares = titulares;
	}
}
```

Ejercicio: Crear JSON con Gson

```
public static void escribirGSON(ArrayList<Noticia> noticias, String fichero) throws IOException {
    OutputStreamWriter out;
    File miFichero;
    Titular titulares;
    String texto;

    GsonBuilder gsonBuilder = new GsonBuilder();
    gsonBuilder.setDateFormat("dd-MM-yyyy");
    gsonBuilder.setPrettyPrinting();
    Gson gson = gsonBuilder.create();

    titulares = new Titular();
    titulares.setWeb("http://www.alejandrosuarez.es/");
    titulares.setFeed("http://www.alejandrosuarez.es/feed/");
    titulares.setTitulares(noticias);
    texto = gson.toJson(titulares);

    miFichero = new File(Environment.getExternalStorageDirectory().getAbsolutePath(), fichero);
    out = new FileWriter(miFichero);
    out.write(texto);
    out.close();
}
```

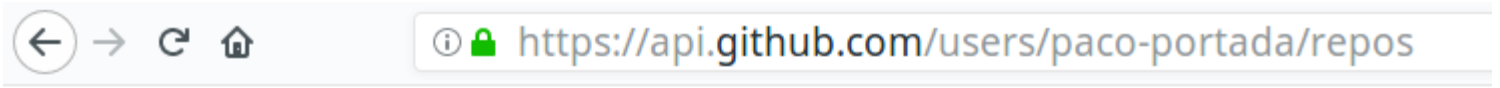
Ejercicio: Retrofit

Crear una aplicación que pida un usuario y obtenga todos sus repositorios en Github.

Se mostrará una lista (usando un RecyclerView) con los nombres, descripciones y fechas de actualización de todos sus repositorios.

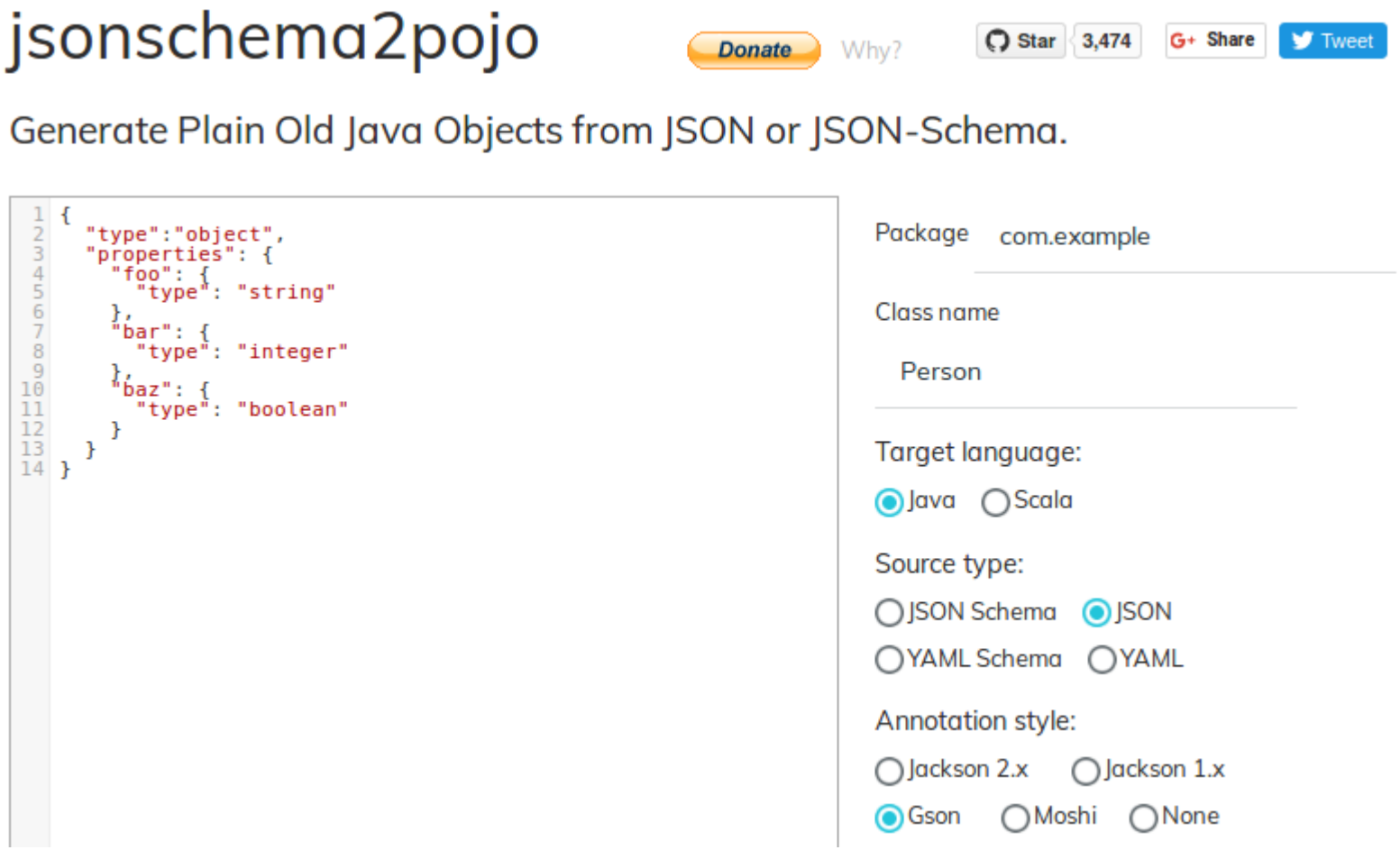
Cuando se pulse en un elemento de la lista, se abrirá el navegador para mostrar el repositorio elegido.

Ejercicio: Retrofit




```
[
  {
    "id": 117340887,
    "name": "android_guides",
    "full_name": "paco-portada/android_guides",
    "owner": {
      "login": "paco-portada",
      "id": 14198211,
      "avatar_url": "https://avatars1.githubusercontent.com/u/14198211?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/paco-portada",
      "html_url": "https://github.com/paco-portada",
      "followers_url": "https://api.github.com/users/paco-portada/followers",
      "following_url": "https://api.github.com/users/paco-portada/following{/other_user}",
      "gists_url": "https://api.github.com/users/paco-portada/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/paco-portada/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/paco-portada/subscriptions",
      "organizations_url": "https://api.github.com/users/paco-portada/orgs",
      "repos_url": "https://api.github.com/users/paco-portada/repos",
      "events_url": "https://api.github.com/users/paco-portada/events{/privacy}",
      "received_events_url": "https://api.github.com/users/paco-portada/received_events",
      "type": "User",
      "site_admin": false
    },
    "private": false,
    "html_url": "https://github.com/paco-portada/android_guides",
    "description": "Extensive Open-Source Guides for Android Developers",
    "fork": true,
    "url": "https://api.github.com/repos/paco-portada/android_guides",
```

Ejercicio: Retrofit



Ejercicio: Retrofit

```
dependencies {

    compile 'com.google.code.gson:gson:2.8.2'
    compile 'com.squareup.okhttp3:okhttp:3.4.1'
    compile 'com.squareup.retrofit2:retrofit:2.2.0'
    compile 'com.squareup.retrofit2:converter-gson:2.2.0'
    compile 'com.squareup.okhttp3:logging-interceptor:3.6.0'
    compile 'com.android.support:cardview-v7:26.1.0'
    compile 'com.android.support:recyclerview-v7:26.1.0'
}
```

Permisos:

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

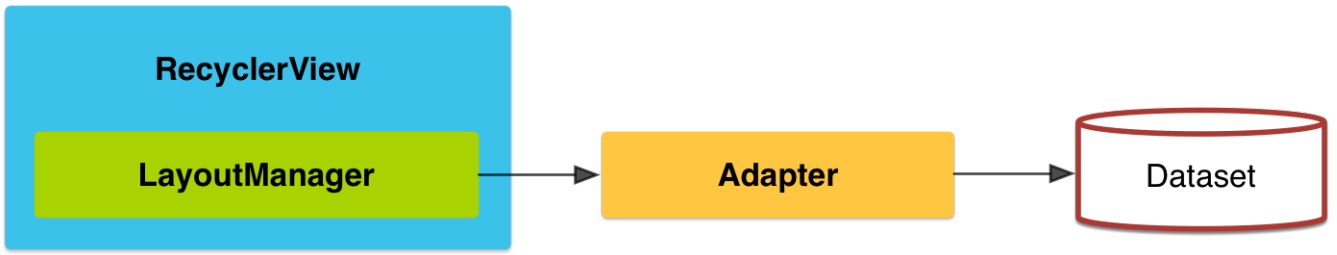
Ejercicio: Retrofit

Using a RecyclerView has the following key steps:

1. Add RecyclerView support library to the gradle build file
2. Define a model class to use as the data source
3. Add a RecyclerView to your activity to display the items
4. Create a custom row layout XML file to visualize the item
5. Create a RecyclerView.Adapter and ViewHolder to render the item
6. Bind the adapter to the data source to populate the RecyclerView

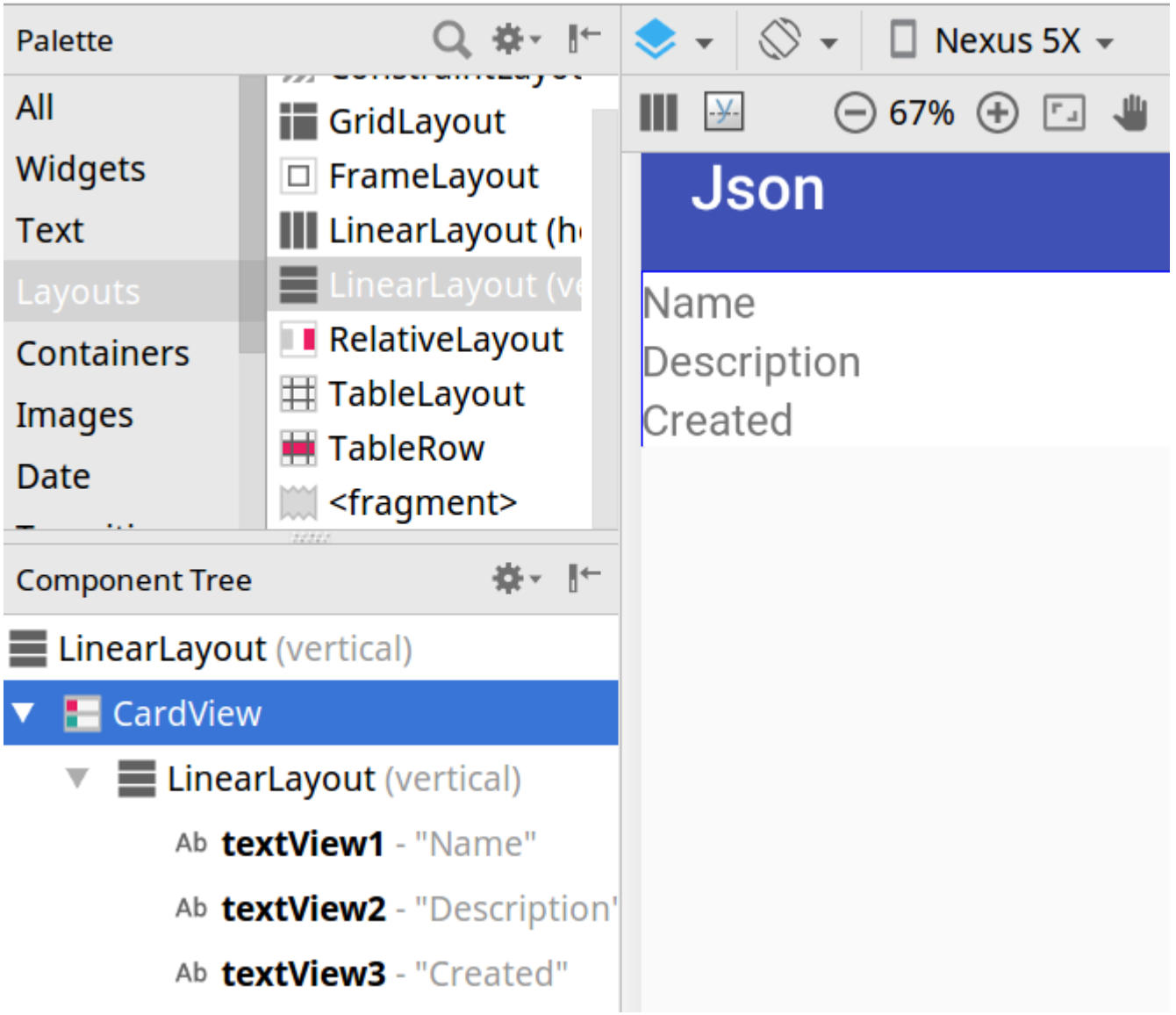
Ejercicio: Retrofit

Using the RecyclerView



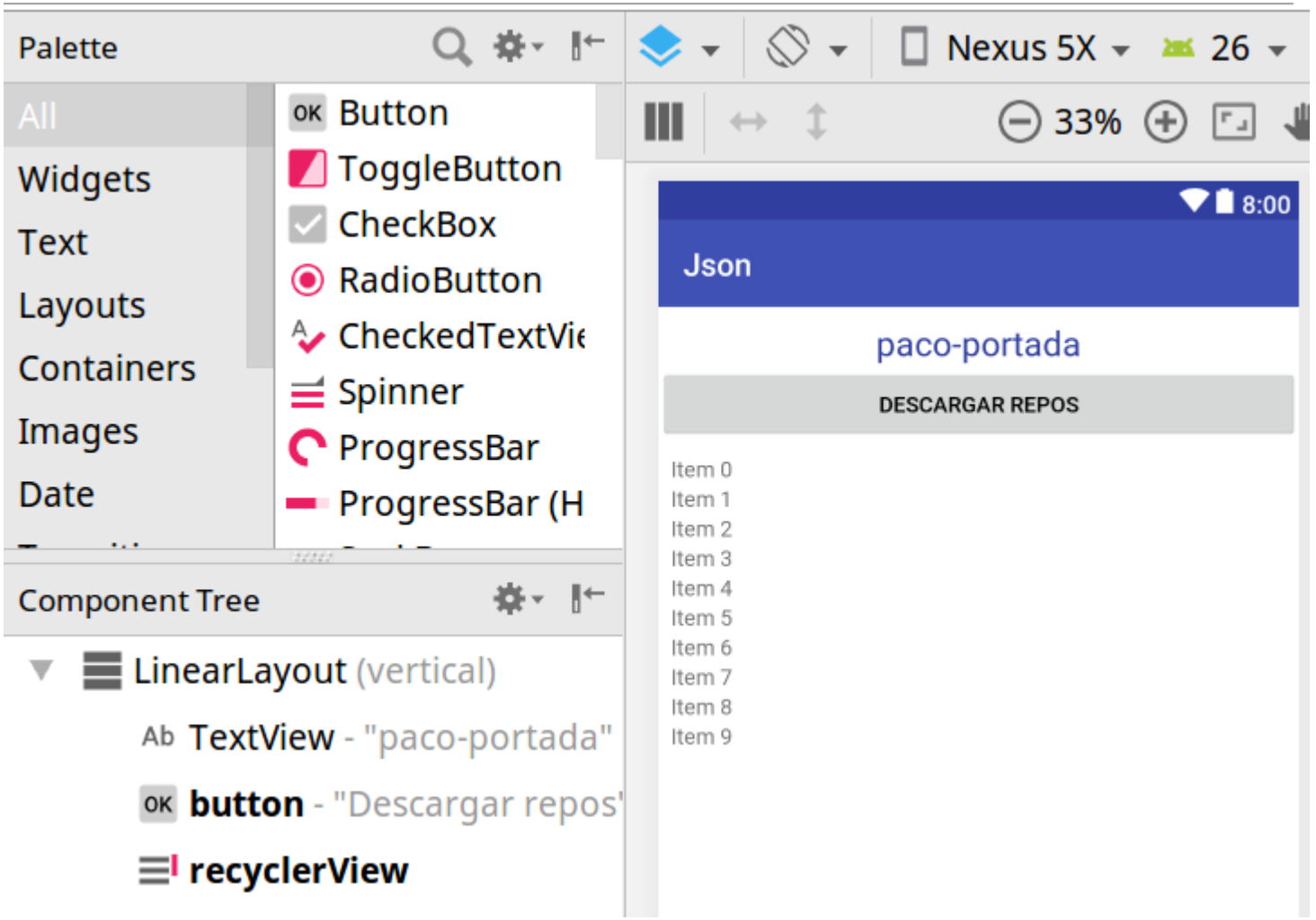
Ejercicio: Retrofit

item_view.xml



Ejercicio: Retrofit

activity_repositorios.xml



Ejercicio: Retrofit

```
public class Repositorios extends AppCompatActivity  implements View.OnClickListener {

    EditText nombreUsuario;
    Button botonDescarga;

    RecyclerView rvRepos;
    private RepositoryAdapter adapter;
    private ArrayList<Repo> repos;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_repositorios);

        nombreUsuario = (EditText) findViewById(R.id.editText);
        botonDescarga = (Button) findViewById(R.id.button);
        botonDescarga.setOnClickListener(this);
        rvRepos = (RecyclerView) findViewById(R.id.recyclerView);

        adapter = new RepositoryAdapter(this);
        rvRepos.setAdapter(adapter);
        rvRepos.setLayoutManager(new LinearLayoutManager(this));

        //manage click

    }

}
```

Ejercicio: Retrofit

Attaching Click Handlers using Listeners

In certain cases, you'd want to setup click handlers for views within the RecyclerView but define the click logic within the containing Activity or Fragment (i.e bubble up events from the adapter). To achieve this, create a custom listener within the adapter and then fire the events upwards to an interface implementation defined within the parent

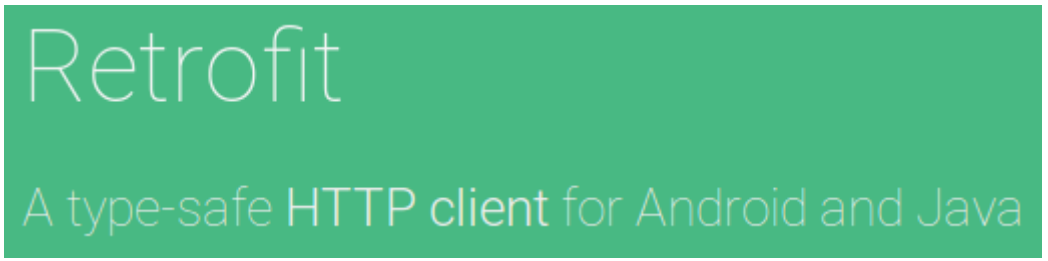
Ejercicio: Retrofit

RecyclerView: Implementing single item click and long press

Ejercicio: Retrofit

```
    //manage click
    rvRepos.addOnItemTouchListener(new RecyclerTouchListener(this, rvRepos,
new ClickListener() {
        @Override
        public void onClick(View view, int position) {
            showMessage("Single Click on position:" + position);
            Uri uri = Uri.parse((String) repos.get(position).getUrl());
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);
            if (intent.resolveActivity(getPackageManager()) != null)
                startActivity(intent);
            else
                showMessage("No hay un navegador");
        }

        @Override
        public void onLongClick(View view, int position) {
            showMessage("Long press on position :" + position);
        }
    }));

    //retrofit

}

private void showMessage(String s) {
    Toast.makeText(this, s, Toast.LENGTH_SHORT).show();
}
```

Ejercicio: Retrofit



Ejercicio: Retrofit

Consuming APIs with Retrofit

Ejercicio: Retrofit

Define the Endpoints

```
public interface MyApiEndpointInterface {
    // Request method and URL specified in the annotation
    // Callback for the parsed response is the last parameter

    @GET("users/{username}")
    Call<User> getUser(@Path("username") String username);

    @GET("group/{id}/users")
    Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);

    @POST("users/new")
    Call<User> createUser(@Body User user);
}
```

Ejercicio: Retrofit

Creating the Retrofit instance

```
// Trailing slash is needed

public static final String BASE_URL = "http://api.myservice.com/";

Gson gson = new GsonBuilder()
        .setDateFormat("yyyy-MM-dd'T'HH:mm:ssZ")
        .create();

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

Ejercicio: Retrofit

Accessing the API

```
MyApiEndpointInterface apiService = retrofit.create(MyApiEndpointInterface.class);

// If we want to consume the API asynchronously, we call the service as follows:

String username = "sarahjean";
Call<User> call = apiService.getUser(username);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {
        int statusCode = response.code();
        User user = response.body();
    }

    @Override
    public void onFailure(Call<User> call, Throwable t) {
        // Log error here since request failed
    }
});
```

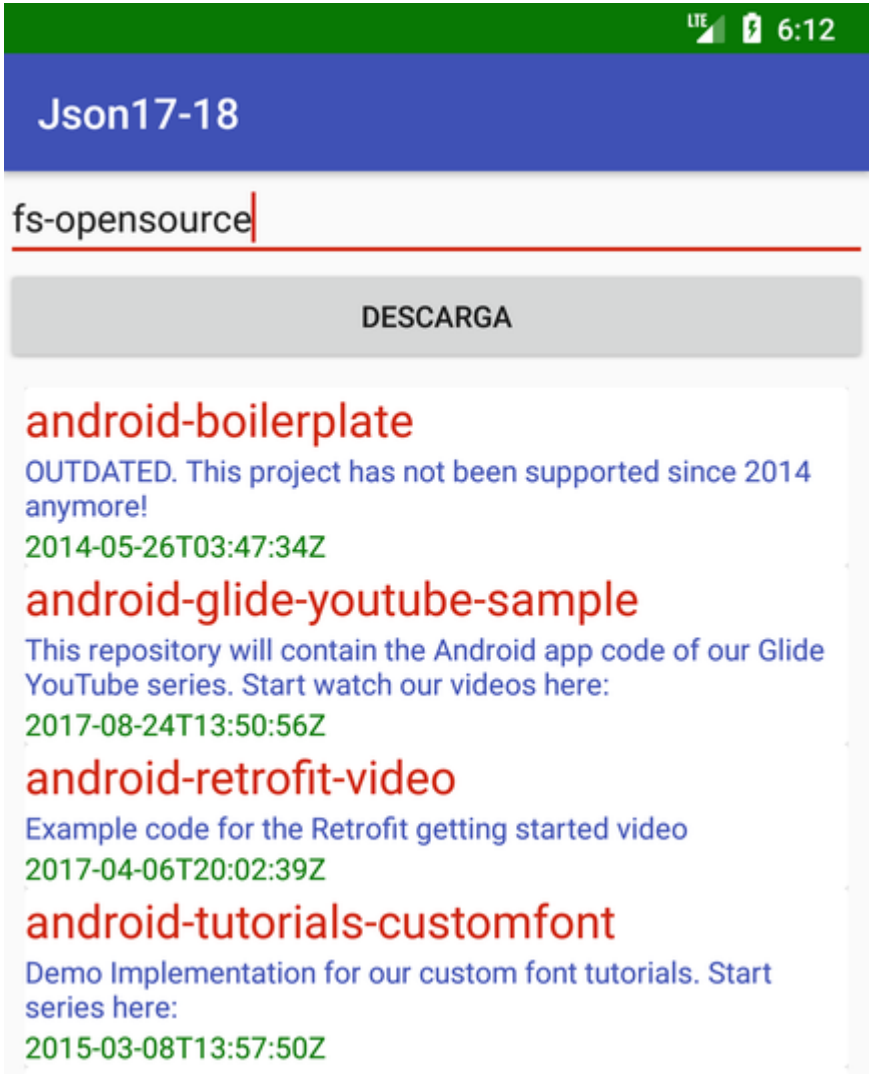
Ejercicio: Retrofit

Retrofit 2 — Customize Network Timeouts

Ejercicio: Retrofit

```
OkHttpClient okHttpClient = new OkHttpClient.Builder()
        .connectTimeout(20, TimeUnit.SECONDS)
        .readTimeout(30, TimeUnit.SECONDS)
        .writeTimeout(15, TimeUnit.SECONDS)
        .build();

Retrofit.Builder builder = new Retrofit.Builder()
        .baseUrl("http://10.0.2.2:3000/")
        .client(okHttpClient)
        .addConverterFactory(GsonConverterFactory.create());
```

Ejercicio: Retrofit



Ejercicio: Retrofit

Cómo usar Retrofit

Ejercicio: Retrofit

¿ ApiAdapter ?

Ejercicio: Retrofit

```
public class Repositorios extends AppCompatActivity  implements View.OnClickListener,
Callback<ArrayList<Repo>> {


    @Override
    public void onClick(View view) {


    }

    @Override
    public void onResponse(Call<ArrayList<Repo>> call, Response<ArrayList<Repo>> response) {


    }

    @Override
    public void onFailure(Call<ArrayList<Repo>> call, Throwable t) {


    }
```

Ejercicio: Retrofit

Posibles errores:

- Está caído el servidor web

- No se encuentra el fichero (error 404)

- El documento Json no es correcto

- El documento Json no tiene la sintaxis esperada

Ejercicio: Retrofit

¿ Comprobación en el entorno local ?

¿Dudas?

¿Sugerencias?

paco@portadaalta.es