Anexo 1. Peticiones REST

Conceptos previos

JSON (JavaScript Object Notation)

JSON es un formato estándar de transmitir datos basado en pares clave-valor. Investiga este formato, dos buenos puntos de inicio son la Wikipedia y su página web oficial:

- http://en.wikipedia.org/wiki/JSON
- http://www.json.org

Si has entendido correctamente, ahora no debería suponerte ningún problema leer estos JSON:

```
{"name":"Julián","address":"Campus de Arrosadía s/n"}
```

```
{"players":[{"nick":"Julio","hasPermission":true},{"nick":"Andrés","hasPermission":false}],"games":4}
```

Si no has sido capaz de entender las dos líneas anteriores, debes investigar un poco más. No avances hasta que lo hayas hecho.

Peticiones REST (REpresentational State Transfer)

Para comunicarnos con el servidor, vamos a usar peticiones REST que en esencia son peticiones HTTP, iguales a las que se hacen a un servidor web.

En una petición REST podemos distinguir 3 parámetros característicos:

1. Dirección URL
2. Tipo de petición: GET, POST, PUT y DELETE (entre otros)
3. Cabeceras
4. Contenido (para peticiones tipo POST, PUT y DELETE)

Dirección URL

URL significa *Uniform Resource Locator* y se compone de un protocolo y el nombre del recurso. Así por ejemplo nos encontramos que `http://www.unavarra.es/conocerlauniversidad/campus` usa el protocolo http y el recurso se llama www.unavarra.es/conocerlauniversidad/campus. En el nombre del recurso se suele distinguir entre el dominio (www.unavarra.es) y la ruta (/conocerlauniversidad/campus).

Tipo de petición

Las peticiones son normalmente de tipo GET, es decir, traen información desde la localización remota. Un navegador web por ejemplo, cuando recupera una página lo hace habitualmente usando este tipo de

peticiones. Las peticiones GET no tienen contenido.

Pero no es el único tipo, existen otros entre los que cabe destacar POST, PUT y DELETE. De estos tres, el más habitual es POST ya que se suele usar para enviar información.

Cabeceras

Las cabeceras es cierta información que se envía en una petición pero que no forman parte del contenido. En concreto, el tipo de petición también es una cabecera.

A diferencia del contenido, la cabecera contiene información que permite tratar la petición de una forma u otra, con independencia del contenido. Por ejemplo, algunas de las cabeceras más utilizadas son las que permiten indicar el tipo de contenido, el idioma que se espera en la respuesta o credenciales de autentificación.

Contenido

El contenido es toda aquella información que se quiere enviar. La única petición que no lleva contenido son las de tipo GET.

Por ejemplo, en una petición en la que se quiere crear un nuevo usuario, el contenido de la petición tiene aquellos datos que nos hacen falta: email, nombre, contraseña, etc.

Es posible tener peticiones con la misma URL pero que sean de tipos distintos, lo que nos permite dar a entender el significado de una petición de forma sencilla. Por ejemplo, si pensamos en acciones a realizar sobre un hipotético usuario "antonio":

- Obtener datos de su perfil: `GET /user/antonio`
- Modificar sus datos: `POST /user/antonio`
- Borrar el usuario: `DELETE /user/antonio`

Las tres peticiones usan el mismo recurso (se ha representado sólo la ruta en el ejemplo anterior), pero son de tipos distintos y por lo tanto, esperamos acciones distintas.

Trabajar con JSON

Una de las formas de trabajar con JSON es usar objetos de Java que nos permitan dar mayor significado a su contenido a la vez de permitirnos trabajar con elementos propios de Java. Y para ello se utilizan ciertas librerías para convertir a y de JSON.

Algunas de las librerías más conocidas para android son Jackson (http://wiki.fasterxml.com/JacksonHome) y Gson (https://github.com/google/gson). Trabajaremos con Gson, busca cómo añadirla usando gradle.

El objetivo que queremos conseguir con esta librería es doble, por un lado

poder generar un JSON cuando hay que enviar información y por otro convertir un texto JSON que recibimos (por ejemplo desde el servidor) en un objeto Java con el que poder trabajar.

Un ejemplo sería tener un objeto Java como:

```
public class Usuario {
     private String name;
     private int birthyear;
}
```

Y que se pueda convertir de forma sencilla a y desde:

```
{“name”:”Juanjo Rodre”,”birthyear”,1995}
```

Busca en el User Guide de Gson algunos ejemplos como este.

Android Asynchronous Http Client

En android vamos a usar esta librería que nos va a ayudar mucho a trabajar con peticiones REST.

En la web oficial de esta librería http://loopj.com/android-async-http, puedes encontrar cómo añadirla usando Gradle.

Estructura de una petición

Debemos recordar que el paradigma de programación en android es que trabajamos en base a eventos. Con esto podemos simplificar el proceso a mostrar una pantalla y a esperar que ocurra algo, bien porque por ejemplo el usuario realiza alguna acción (toca un botón, una imagen, mueve elementos, etc) o porque se recibe el resultado de alguna acción.

Con las peticiones al servidor vamos a tener al menos dos acciones. La primera es cuando se realiza la petición, tendremos que configurar los parámetros necesarios y enviarla. El envío se hace en segundo plano, en un hilo independiente y por lo tanto cuando se obtenga el resultado o el error, se generará otro evento al que responderemos.

Las peticiones se hacen siempre de forma asíncrona.

Respuesta de una petición

Cuando vayamos a realizar una petición, un punto a tener en cuenta es qué vamos a hacer cuando llegue la respuesta.

Para cada respuesta vamos a generar una clase que extienda de `AsyncHttpResponseHandler`. Debes tener en cuenta que, al ser la respuesta a una petición asíncrona, será esta clase la que realice los cambios necesarios en la aplicación.

Esta clase debe implementar los métodos `onSuccess` que se llamará

cuando la respuesta es válida y `onFailure` que se llamará en caso contrario.

Debemos tener en cuenta que una respuesta "válida" es aquella cuya petición se ejecuta en servidor y este responde con un código de estado HTTP (http://en.wikipedia.org/wiki/List_of_HTTP_status_codes) de tipo 2xx o 3xx. Sin embargo, será "no válida" cuando su estado sea 4xx ó 5xx.

Primera petición

Vamos a probar a realizar una petición al módulo "welcome". El funcionamiento es sencillo, se trata de realizar una petición POST en la que enviaremos un nombre y el servidor nos responderá con un mensaje de bienvenida. Una vez que recibamos ese mensaje, mostraremos un `Toast` con él.

Necesitamos dos elementos: un `Activity` que lance la petición y una clase que muestre el `Toast` cuando llegue la petición.

Vamos a empezar por el segundo elemento y vamos a crear una clase que se llame `WelcomeResponseHandler`. Analizando este nombre que le damos a la clase, "*ResponseHandler*" nos indica que es una respuesta y "*Welcome*" hace referencia a la petición. De esta forma podremos identificar muy rápidamente a qué corresponde. Usa como base este código:
https://gist.github.com/tatai/99821db2014448c2263dbc7b17857c0b

Como puedes ver, esta clase recibe `Context` (es decir, el `Activity`) en el constructor para usarlo posteriormente en el `Toast`. Una vez que llega la petición, si todo ha ido bien, ejecutará onSuccess y mostrará el Toast.

Estamos en disposición de hacer la petición. El proceso es muy sencillo ahora que tenemos preparada la respuesta, debemos únicamente preparar la información a enviar al servidor. Vamos a usar un JSON directamente, pero lo ideal es que usases el mapper tal y como hemos visto anteriormente:

```
AsyncHttpClient client = new AsyncHttpClient();
client.post(this,        "http://api.messenger.tatai.es/welcome",        new
StringEntity("{\"name\":\"Fran\"}"),        "application/json",        new
WelcomeResponseHandler(this));
```

Permisos

Debes tener en cuenta que en android es necesario declarar explícitamente los servicios a los que quiere acceder tu aplicación en el `AndroidManifest.xml`. En concreto, para poder acceder a Internet debes usar esto:

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

Investiga cómo y dónde debes añadirlo en el `AndroidManifest.xml`.

API del proyecto

En la web de la asignatura tienes un enlace a una página con la especificación de la API, es decir, los distintos módulos que lo componen, la dirección de la petición, el método, parámetro de entrada y de salida. Dispones en esta misma página de la opción de probar las peticiones.