


Open Opened 2 days ago by  **Rubén Montero**

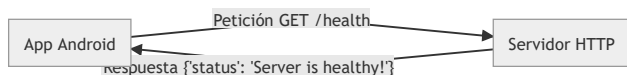
Petición POST de registro

Resumen

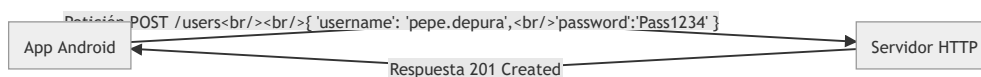
- Cuando el usuario pulse *Registrarse* enviaremos una petición HTTP al API REST para que grabe los datos del nuevo usuario
- La petición contendrá un cuerpo JSON
- Si la respuesta es exitosa, mostraremos un **Toast**

Descripción

Hasta ahora nuestras peticiones eran **GET** que recuperaban datos del servidor. Usábamos el *cuerpo* de la *respuesta*.



Ahora, mandaremos una petición **POST** que *sube* datos al servidor:



Esto, en la práctica, lo conseguiremos *creando* un **JSONObject** y pasándolo a una instancia de **JsonObjectRequest** como tercer parámetro, jén lugar de el **null** que siempre hemos pasado hasta ahora!

La tarea

Sigue los pasos indicados en **Server.java** para configurar la IP del servidor y que nuestro emulador tenga conectividad.

A continuación, en **RegisterActivity.java** **añade** un atributo tipo **RequestQueue** e **inicialízalo** en **onCreate** como se indica a continuación:

```
public class RegisterActivity extends AppCompatActivity {
    /* ... */
    private RequestQueue requestQueue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* ... */

        requestQueue = Volley.newRequestQueue(this);
    }
}
```

Ahora encapsularemos la *lógica* relacionada a *enviar* la petición **POST** en un nuevo método privado **registerNewUser**. Dicho método se *invocará* cuando el usuario haga *click* en el botón. Para ello, **añade** el código como se expone a continuación:

```
public class RegisterActivity extends AppCompatActivity {
    /* ... */

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* ... */

        buttonRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Aquí estará tu Toast de la tarea anterior
                // No lo elimines

                registerNewUser();
            }
        });
    }
}
```

```
private void registerNewUser() {

}

}
```

Luego, en `registerNewUser` instancia un objeto `JSONObject` e invoca `.put` como se indica a continuación:

```
private void registerNewUser() {
    JSONObject requestBody = new JSONObject();
    requestBody.put("username", editTextName.getText().toString());
    requestBody.put("password", editTextPassword.getText().toString());

}
```

(Así, hemos construido un objeto JSON que, predeciblemente, representa un JSON como el siguiente, donde `"pepe.depura"` y `"Pass1234"` serán los valores reales introducidos por el usuario)

```
{
  "username": "pepe.depura",
  "password": "Pass1234"
}
```

También, envuelve dicho código en un `try-catch`:

```
private void registerNewUser() {
    JSONObject requestBody = new JSONObject();
    try {
        requestBody.put("username", editTextName.getText().toString());
        requestBody.put("password", editTextPassword.getText().toString());
    } catch (JSONException e) {
        throw new RuntimeException(e);
    }

}
```

Ahora... ¡vamos a enviar una petición HTTP para *registrar* al usuario en el servidor!

Como primer paso, instancia un objeto `JsonObjectRequest` y como cinco parámetros del constructor, especifica:

- (1) `Request.Method.POST`.
- (2) `Server.name + "/users"`.
- (3) `requestBody`. Solíamos usar `null` en otras peticiones... ¡Aquí no!
- (4) `Response.Listener` anónimo.
- (5) `Response.ErrorListener` anónimo.

Quedará así:

```
JsonObjectRequest request = new JsonObjectRequest(
    Request.Method.POST,
    Server.name + "/users",
    requestBody,
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {

        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {

        }
    }
);
```

A continuación, en `onResponse` (respuesta de éxito), muestra un `Toast` con el texto: `"Cuenta registrada"`. Indica `Toast.LENGTH_LONG`.

Luego, al final del método `registerNewUser()`, añade la petición a `this.requestQueue` para que se envíe.

Configuración general en una *app* para usar Internet

Cuando termines los pasos anteriores, puedes efectuar dos cambios necesarios para que funcione la petición.

Primero, en `AndroidManifest.xml` **declara** el permiso de `INTERNET` . Así:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.afundacion.fp.sessions">

    <!-- Aquí, teclea <uses-permi y verás cómo Android Studio autocompleta -->
    <!-- Queremos añadir "android.permission.INTERNET" -->
    <application
        android:allowBackup="true"
```

(Debemos declarar `INTERNET` siempre que nuestra aplicación lance peticiones de red. De lo contrario, serán denegadas por el propio dispositivo.)

Después, también en `AndroidManifest.xml` , **añade** `android:usesCleartextTraffic="true"` :

```
<application
+     android:usesCleartextTraffic="true"
    android:allowBackup="true"
```

(Nuestro API REST no es seguro. Emplea HTTP en lugar de conexiones cifradas HTTPS. Por ello, debemos añadir este atributo. ¡Ojo! Esto *n** debe ser así para apps que estén publicadas.)

Después de esto, ya puedes probar la *app* y registrarte introduciendo datos y pulsando el botón.

¡Enhorabuena! Has completado con éxito tu primer `POST` a un API REST para registrar usuarios.

Por último

Sube tus cambios al repositorio en un nuevo *commit*.



Rubén Montero @ruben.montero changed milestone to [%Sprint 3](#) 2 days ago