Tarea 8

Cliente REST

Sistemas distribuidos

Martes, 1 de diciembre de 2020

Joel Harim Hernández Javier

Descripción

Cada alumno deberá desarrollar un programa Java que consuma el servicio web que creamos en la tarea 7.

- ➔ El programa deberá desplegar el siguiente menú:
  MENU
  a. Alta usuario
  b. Consulta usuario
  c. Borra usuario
  d. Borra todos los usuarios
  e. Salir
  Opción: _
- ➔ La opción "Alta usuario" leerá del teclado el email, el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género ("M" o "F"). Entonces se invocará el método "alta" del servicio web. Se deberá desplegar "OK" si se pudo dar de alta el usuario, o bien, el mensaje de error que regresa el servicio web.
- ➔ La opción "Consulta usuario" leerá del teclado el email de un usuario previamente dado de alta. Entonces se invocará el método "consulta" del servicio web. Si el usuario existe se desplegará en pantalla el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género. La foto del usuario se ignorará. Notar que el método web "consulta" regresa una string JSON la cual representa un objeto de tipo Usuario (ver el archivo Servicio.java), por tanto será necesario utilizar GSON para convertir la string JSON a un objeto de tipo Usuario y posteriormente desplegar los campos del objeto (excepto el campo "foto"). Si hubo error, se desplegará el mensaje que regresa el servicio web.
- ➔ La opción "Borra usuario" leerá del teclado el email de un usuarios previamente dado de alta. Entonces se invocará el método "borra" del servicio web. Se deberá desplegar "OK" si se pudo borrar el usuario, o bien, el mensaje de error que regresa el servicio web.
- ➔ La opción "Borra todos los usuarios" invocará el método "borrar_usuarios" creado en la actividad individual de la clase pasada. Se deberá desplegar "OK" si se pudo borrar el usuario, o bien, el mensaje de error que regresa el servicio web.
- ➔ La opción "Salir" terminará el programa.

Desarrollo

Modificación de la aplicación servidor

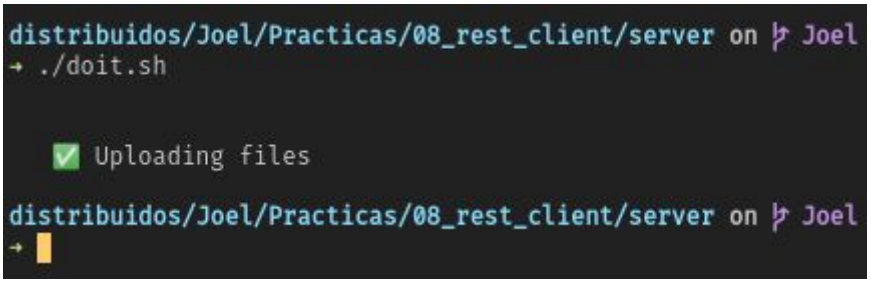
Se agregó el método `borra_todos()` a `Servicio.java`, el cual es el encargado de borrar toda la información de la tabla de usuarios, y las fotos asociadas:

```
   @POST
   @Path("borrar_usuarios")
   @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
   @Produces(MediaType.APPLICATION_JSON)
   public Response borra_todos() throws Exception {
       Connection conn = pool.getConnection();
       PreparedStatement order;

       try {
           order = conn.prepareStatement("DELETE FROM fotos_usuarios");
           order.executeUpdate();
           order.close();

           order = conn.prepareStatement("DELETE FROM usuarios");
           order.executeUpdate();
           order.close();

       } catch (Exception e) {
           return Response.status(400).entity(j.toJson(new
Error(e.getMessage()))).build();
       } finally {
           conn.close();
       }
       return Response.ok().build();
   }
```

Creación de la máquina virtual y despliegue del servidor

Se creó una máquina virtual en Azure con el puerto 8080 abierto para el servidor. Se utilizaron los scripts escritos en bash de la práctica pasada.

Subida de archivos:



Ejecución del deploy en el servidor:

```
joelcito@ubunturm:~/server$ ./deploy.sh

   🚀 Updating system and installing utilities
   💺 Installing tomcat
   📦 Configuring MySQL:
      Please enter 'root' as root's password and 'hugo' as hugo password

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.
```

```
All done!
      Enter root's password: (root)
Enter password:
      Enter hugo's password: (hugo)
Enter password:
   📦 Configured MySQL
   🎉 Deploying app

   Tomcat server is running at port 8080. Stop with /hom
a.sh stop
joelcito@ubunturm:~/server$
```

Client.java

Esta clase es el cliente REST del servidor, su ejecución consta de un menú de elección para el usuario, y cada uno de ellos manda a llamar al método `Helper.fetch()`, el cuál es el encargado de realizar la construcción de la URL y realizar la petición al servidor.

Fragmento de la función `main` de `Client`:

```
      while (true) {
          System.out.println("a. Alta de usuario");
          System.out.println("b. Consulta de usuario");
          System.out.println("c. Borrar usuario");
          System.out.println("d. Borrar todos los usuarios");
          System.out.println("e. Salir");
          System.out.println("Ingresa una opción: ");

          BufferedReader keyboard = new BufferedReader(new
InputStreamReader(System.in));
          String option = keyboard.readLine();

          if (option.equals("a")) {
              userRegistration(keyboard);
          } else if (option.equals("b")) {
              queryUser(keyboard);
          } else if (option.equals("c")) {
              deleteUser(keyboard);
```

```
            } else if (option.equals("d")) {
                deleteAllUsers();
            } else if (option.equals("e")) {
                break;
            } else {
                System.out.println("\"" + option + "\" no es una opción válida.");
            }
        }

        System.out.println("Bye!");
```

`Helper.fetch()`:

```
    public static String fetch(String operation, String paramName, String args,
String ip)
            throws MalformedURLException, IOException, RuntimeException {

        URL url = new URL("http://" + ip + ":8080/Servicio/rest/ws/" + operation);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();

        conn.setDoOutput(true);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");

        String params = paramName + "=" + URLEncoder.encode(args, "UTF-8");

        OutputStream sender = conn.getOutputStream();

        sender.write(params.getBytes());
        sender.flush();

        if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
            BufferedReader error = new BufferedReader(new
InputStreamReader(conn.getErrorStream()));
            return recoverResponse(error);
        }

        BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        String response = recoverResponse(reader);

        return response.equals("") ? "OK" : response;
    }
```

Ejecución

Compilación y ejecución:

```
distribuidos/Joel/Practicas/08_rest_client/client on  Joel
→ make run
javac Client.java -cp gson-2.8.6.jar:.
java -cp gson-2.8.6.jar:. Client 52.188.174.9
a. Alta de usuario
b. Consulta de usuario
c. Borrar usuario
d. Borrar todos los usuarios
e. Salir
Ingresa una opción:
```

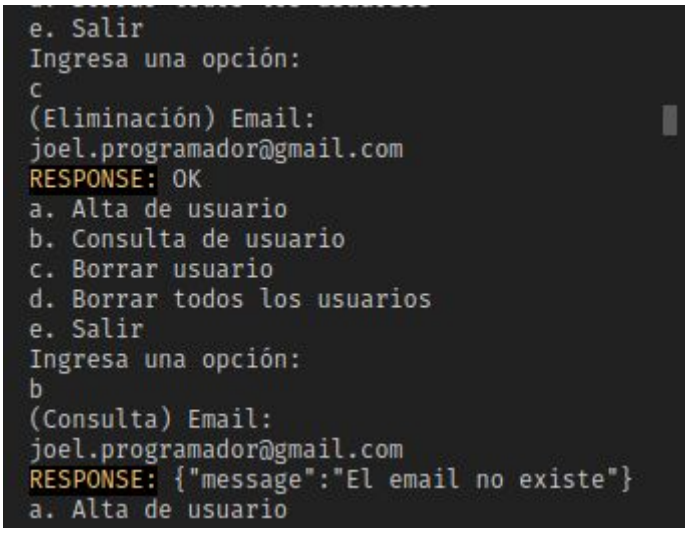
Creación de usuario:

```
Ingresa una opción:
a
(Creación)
Email:
joel.programador@gmail.com
Nombre:
Joelcito
Apellido paterno:
Hernandez
Apellido materno:
Javier
Fecha de nacimiento (YYYY-MM-DD):
1999-07-02
Teléfono:
123
Genero (M / F):
M
RESPONSE: OK
```

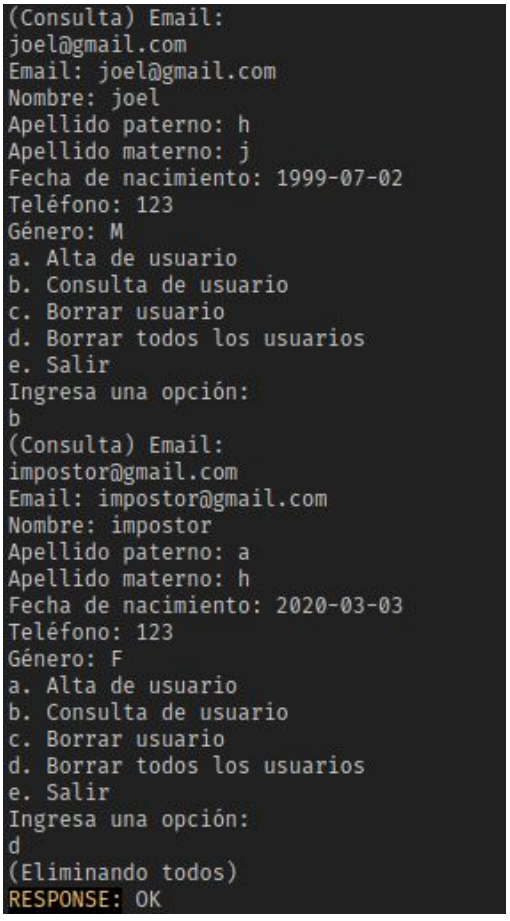
Consulta de usuario:

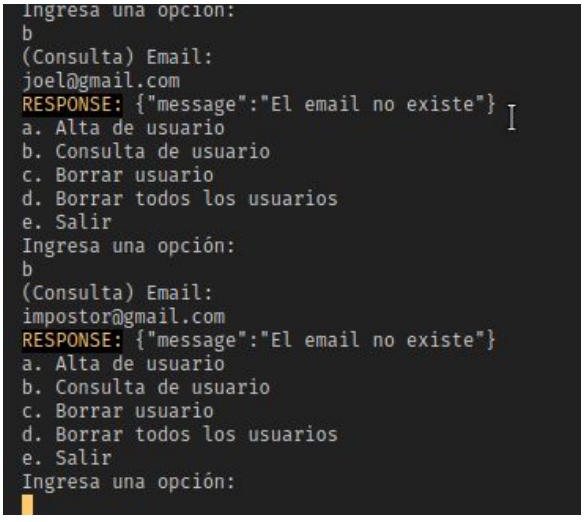
```
a. Alta de usuario
b. Consulta de usuario
c. Borrar usuario
d. Borrar todos los usuarios
e. Salir
Ingresa una opción:
b
(Consulta) Email:
joel.programador@gmail.com
Email: joel.programador@gmail.com
Nombre: Joelcito
Apellido paterno: Hernandez
Apellido materno: Javier
Fecha de nacimiento: 1999-07-02
Teléfono: 123
Género: M
a. Alta de usuario
```

Eliminación de usuario (y consulta posterior):



Eliminación de todos los usuarios (y comprobación):





Conclusión

Podemos concluir que con esta práctica pudimos aprender sobre cómo funciona el paradigma de peticiones-respuesta, lo cual nos permite construir sistemas más confiables y usando JSON como el formato de la información que se comunica.