



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Asignatura:

Desarrollo de sistemas distribuidos

*Grupo:*4CM3

Tarea 8. Desarrollo de un cliente para un servicio
web REST

Alumno: Garcia Garcia Rafael

Profesor: Carlos Pineda Guerrero



Lo primero que se tiene que hacer es ejecutar servicio REST que hicimos en la práctica 7, una vez el servicio arriba podemos proceder a realizar nuestra práctica.

Lo primero que tenemos que hacer es agregar una función en nuestro programa **Servicio.java** esta función se llamará “limpiar” la cuál será una petición POST que se encargará de eliminar a todos los usuarios y las fotos subidas al servidor

La siguiente sentencia de código es lo que se agrega a nuestro archivo Servicio.java

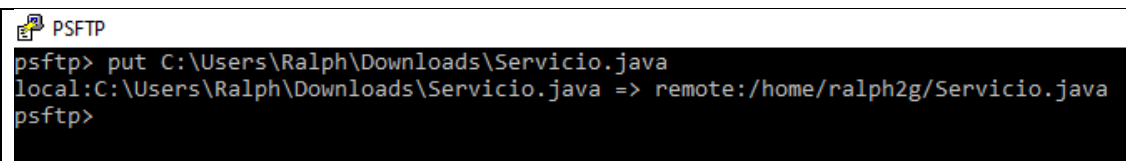
```
1. //Limpiamos los valores de nuestra base de nuestras tablas en la base de datos
   para borrar a todos los usuarios
2. @POST
3. @Path("limpiar")
4. @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
5. @Produces(MediaType.APPLICATION_JSON)
6. public Response borra_todos() throws Exception
7. {
8.     Connection conexion= pool.getConnection();
9.
10.    try
11.    {
12.        PreparedStatement stmt_1 = conexion.prepareStatement("SELECT 1 FROM usuarios");
13.        try
14.        {
15.            ResultSet rs = stmt_1.executeQuery();
16.            try
17.            {
18.                if (!rs.next())
19.                    return Response.status(400).entity(j.toJson(new Error("Pos no pude"))).build();
20.            }
21.            finally
22.            {
23.                rs.close();
24.            }
25.        }
26.        finally
27.        {
28.            stmt_1.close();
29.        }
30.        PreparedStatement stmt_2 = conexion.prepareStatement("DELETE FROM fotos_usuarios");
31.        try
32.        {
33.            stmt_2.executeUpdate();
34.        }
35.        finally
36.        {
37.            stmt_2.close();
38.        }
39.
40.        PreparedStatement stmt_3 = conexion.prepareStatement("DELETE FROM usuarios");
41.        try
42.        {
43.            stmt_3.executeUpdate();
44.        }
45.        finally
46.        {
47.            stmt_3.close();
```

```

48.     }
49. }
50. catch (Exception e)
51. {
52.     return Response.status(400).entity(j.toJson(new Error(e.getMessage()))).build();
53. }
54. finally
55. {
56.     conexion.close();
57. }
58. return Response.ok().build();
59. }

```

Una vez adicionada esta sentencia de código a la API procedemos a transferir nuestro archivo mediante el uso de un programa de FTP como se muestra en la **figura 1** con la siguiente en la terminal:



```

PSFTP
psftp> put C:\Users\Ralph\Downloads\Servicio.java
local:C:\Users\Ralph\Downloads\Servicio.java => remote:/home/ralph2g/Servicio.java
psftp>

```

Figura 1. Transferencia de **Servicio.java** a nuestra máquina en Azure

Posteriormente lo que tenemos que hacer es conectarnos a nuestra máquina mediante terminal para poder compilar y ejecutar de nuevo el servicio REST que se encuentra en nuestra máquina virtual

Para esto ejecutaremos los siguientes comandos en nuestra terminal, estos comandos lo que harán es copiar el nuevo programa, compilarlo, crear un nuevo paquete .war de nuestro servicio y ejecutar el servidor apache para que se puede acceder a esta nueva petición en nuestro servicio:

- cp Servicio.java negocio/
- javac -cp \$CATALINA_HOME/lib/javax.ws.rs-api-2.0.1.jar:\$CATALINA_HOME/lib/gson-2.3.1.jar:. negocio/Servicio.java

Ejecución del servicio y servidor APACHE

- rm WEB-INF/classes/negocio/*
- cp negocio/*.class WEB-INF/classes/negocio/.
- jar cvf Servicio.war WEB-INF META-INF
- cp Servicio.war apache-tomcat-8.5.60/webapps/
- sh \$CATALINA_HOME/bin/catalina.sh start

```
ralph2g@Rest: ~  
ralph2g@Rest:~$ cp Servicio.java negocio/  
ralph2g@Rest:~$ javac -cp $CATALINA_HOME/lib/javax.ws.rs-api-2.0.1.jar:$CATALINA_HOME/lib/gson-2.3.1.jar:. negocio/Servicio.java  
ralph2g@Rest:~$ rm WEB-INF/classes/negocio/*  
ralph2g@Rest:~$ cp negocio/*.class WEB-INF/classes/negocio/.  
ralph2g@Rest:~$ jar cvf Servicio.war WEB-INF META-INF  
added manifest  
adding: WEB-INF/(in = 0) (out= 0) (stored 0%)  
adding: WEB-INF/classes/(in = 0) (out= 0) (stored 0%)  
adding: WEB-INF/classes/negocio/(in = 0) (out= 0) (stored 0%)  
adding: WEB-INF/classes/negocio/AdaptadorGsonBase64.class(in = 1799) (out= 737) (deflated 59%)  
adding: WEB-INF/classes/negocio/Error.class(in = 278) (out= 214) (deflated 23%)  
adding: WEB-INF/classes/negocio/Servicio.class(in = 8425) (out= 3823) (deflated 54%)  
adding: WEB-INF/classes/negocio/Usuario.class(in = 899) (out= 518) (deflated 42%)  
adding: WEB-INF/web.xml(in = 672) (out= 296) (deflated 55%)  
ignoring entry META-INF/  
adding: META-INF/context.xml(in = 304) (out= 213) (deflated 29%)  
ralph2g@Rest:~$ cp Servicio.war apache-tomcat-8.5.60/webapps/  
ralph2g@Rest:~$ sh $CATALINA_HOME/bin/catalina.sh start  
Using CATALINA_BASE:   /home/ralph2g/apache-tomcat-8.5.60  
Using CATALINA_HOME:   /home/ralph2g/apache-tomcat-8.5.60  
Using CATALINA_TMPDIR: /home/ralph2g/apache-tomcat-8.5.60/temp  
Using JRE_HOME:        /usr/lib/jvm/java-8-openjdk-amd64  
Using CLASSPATH:        /home/ralph2g/apache-tomcat-8.5.60/bin/bootstrap.jar:/home/ralph2g/apache-tomcat-8.5.60/bin/tomcat-juli.jar  
Tomcat started.  
ralph2g@Rest:~$
```

Figura 2. Ejecución de nuestro Servicio REST

Una vez montado nuestro servicio actualizado con la nueva funcionalidad procedemos a crear un cliente para poder hacer uso de las solicitudes del servidor

El cliente lo que hará es generar una clase usuario el cual instanciara un objeto que se serializará con Gson para poder registrar y consultar usuarios. Tendrá los métodos alta, consulta, borrar y borrarTodos los cuáles se encargarán de hacer las solicitudes al servicio Web. El código del servicio WEB es el siguiente:

```
1. import com.google.gson.Gson;  
2. import com.google.gson.GsonBuilder;  
3. import java.net.URLEncoder;  
4. import java.io.*;  
5. import java.net.HttpURLConnection;  
6. import java.net.URL;  
7. import java.util.Scanner;  
8.  
9. public class Cliente {  
10.     static Scanner teclado = new Scanner(System.in);  
11.  
12.     static class Usuario {  
13.         String email;  
14.         String nombre;  
15.         String apellido_paterno;  
16.         String apellido_materno;  
17.         String fecha_nacimiento;  
18.         String telefono;  
19.         String genero;  
20.         byte[] foto;  
21.         Usuario(String email,String nombre,String apellido_paterno,String apelli  
do_materno,String fecha_nacimiento,String telefono,String genero,byte[] foto){  
22.             this.email = email;  
23.             this.nombre = nombre;  
24.             this.apellido_paterno = apellido_paterno;  
25.             this.apellido_materno = apellido_materno;  
26.             this.fecha_nacimiento = fecha_nacimiento;  
27.             this.telefono = telefono;  
28.             this.genero = genero;  
29.             this.foto = foto;  
30.         }  
31.     }  
32.     public static void borrarTodos() throws Exception {
```

```

33.         // Se conecta con el servicio REST
34.         URL url = new URL("http://52.171.215.82:8080/Servicio/rest/ws/limpiar"
35.         );
36.         HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
37.         conexion.setDoOutput(true);
38.         conexion.setRequestMethod("POST");
39.         conexion.setRequestProperty("Content-Type", "application/x-www-form-
40.         urlencoded");
41.         // se debe verificar si hubo error
42.         if (conexion.getResponseCode() != HttpURLConnection.HTTP_OK)
43.             // throw new RuntimeException("Codigo de error HTTP: " + conexion.
44.             getResponseCode());
45.             System.out.print("\n**Error eliminando usuarios**\n");
46.         else
47.             System.out.print("\n**Los usuario se han eliminado con exito**\n")
48.         ;
49.         conexion.disconnect();
50.     }
51.
52.     public static void alta() throws Exception {
53.         System.out.print("Email: ");
54.         String email = teclado.nextLine();
55.         System.out.print("Nombre: ");
56.         String nombre = teclado.nextLine();
57.         System.out.print("ApellidoPaterno: ");
58.         String apellido_paterno = teclado.nextLine();
59.         System.out.print("ApellidoMaterno: ");
60.         String apellido_materno = teclado.nextLine();
61.         System.out.print("FechaNacimiento: ");
62.         String fecha_nacimiento = teclado.nextLine();
63.         System.out.print("Telefono: ");
64.         String telefono = teclado.nextLine();
65.         System.out.print("Genero: ");
66.         String genero = teclado.nextLine();
67.         Usuario user = new Usuario(email,nombre,apellido_paterno,apellido_mate
68.         rno,fecha_nacimiento,telefono,genero,null);
69.         Gson j = new GsonBuilder().setDateFormat("yyyy-MM-
70.         dd'T'HH:mm:ss.SSS").create();
71.         String jsonData = j.toJson(user);
72.         // Se conecta con el servicio REST
73.         URL url = new URL("http://52.171.215.82:8080/Servicio/rest/ws/alta");
74.
75.         HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
76.
77.         conexion.setDoOutput(true);
78.         conexion.setRequestMethod("POST");
79.         conexion.setRequestProperty("Content-Type", "application/x-www-form-
80.         urlencoded");
81.         String parametros = "usuario=" + URLEncoder.encode(jsonData,"UTF-
82.         8");
83.         OutputStream os = conexion.getOutputStream();
84.         os.write(parametros.getBytes());
85.         os.flush();
86.         // se debe verificar si hubo error
87.         if (conexion.getResponseCode() != HttpURLConnection.HTTP_OK)
88.             throw new RuntimeException("Codigo de error HTTP: " + conexion.get
89.             ResponseCode());
90.         else
91.             System.out.print("Ok\n");
92.         conexion.disconnect();
93.     }
94.
95.     public static void consulta() throws Exception {
96.         System.out.print("Email: ");
97.         String email = teclado.nextLine();

```

```

87.         // Se conecta con el servicio REST
88.         URL url = new URL("http://52.171.215.82:8080/Servicio/rest/ws/consulta
");
89.         HttpURLConnection conexion = (HttpURLConnection) url.openConnection();

90.         conexion.setDoOutput(true);
91.         conexion.setRequestMethod("POST");
92.         conexion.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
93.         String parametros = "email=" + URLEncoder.encode(email, "UTF-8");
94.         OutputStream os = conexion.getOutputStream();
95.         os.write(parametros.getBytes());
96.         os.flush();
97.         // se debe verificar si hubo error
98.         if (conexion.getResponseCode() != HttpURLConnection.HTTP_OK)
99.             // throw new RuntimeException("Codigo de error HTTP: " + conexion.
getResponseCode());
100.            System.out.println("\n**Usuario NO encontrado**\n");
101.        else {
102.            BufferedReader br = new BufferedReader(new InputStreamReader
r((conexion.getInputStream())));
103.            String jsonData = br.readLine();
104.            Gson j = new GsonBuilder().create();
105.            Usuario user = (Usuario)j.fromJson(jsonData, Usuario.class);

106.            System.out.println("\nUSUARIO ENCONTRADO: \nEmail: " + user
.email + "\nNombre: " + user.nombre + "\nApellido Paterno: " + user.apellido_p
aterno + "\nApellido Materno: " + user.apellido_materno + "\nFecha de nacimien
to: " + user.fecha_nacimiento + "\nTelefono: " + user.telefono + "\nGenero: "
+ user.genero + "\n");
107.        }
108.        conexion.disconnect();
109.    }

110.
111.    public static void borrar() throws Exception {
112.        System.out.print("Email: ");
113.        String email = teclado.nextLine();
114.        // Se conecta con el servicio REST
115.        URL url = new URL("http://52.171.215.82:8080/Servicio/rest/ws/b
orra");
116.        HttpURLConnection conexion = (HttpURLConnection) url.openConnec
tion();
117.        conexion.setDoOutput(true);
118.        conexion.setRequestMethod("POST");
119.        conexion.setRequestProperty("Content-Type", "application/x-www-
form-urlencoded");
120.        String parametros = "email=" + URLEncoder.encode(email, "UTF-
8");
121.        OutputStream os = conexion.getOutputStream();
122.        os.write(parametros.getBytes());
123.        os.flush();
124.        // se debe verificar si hubo error
125.        if (conexion.getResponseCode() != HttpURLConnection.HTTP_OK)
126.            // throw new RuntimeException("Codigo de error HTTP: " + co
nexion.getResponseCode());
127.        System.out.print("\n**El usuario NO se ha podido eliminar**
\n");
128.        else
129.            System.out.print("\n**El usuario se ha eliminado con exito*
*\n");
130.        conexion.disconnect();
131.    }

132.
133.
134.    public static void main(String args[]) throws Exception {
135.        for(;;) {

```

```

136.         System.out.println("1-Dar de alta un usuario");
137.         System.out.println("2-Consultar un usuario");
138.         System.out.println("3-Borrar un usuario");
139.         System.out.println("4-Borrar los usuarios");
140.         System.out.println("otro. Salir");
141.         System.out.print("Opcion: ");
142.         String opcion = teclado.nextLine();
143.         switch (opcion) {
144.             case "1":
145.                 alta();
146.                 break;
147.             case "2":
148.                 consulta();
149.                 break;
150.             case "3":
151.                 borrar();
152.                 break;
153.             case "4":
154.                 borrarTodos();
155.                 break;
156.             default:
157.                 System.exit(0);
158.                 break;
159.         }
160.     }
161. }

```

Compilamos nuestro archivo:

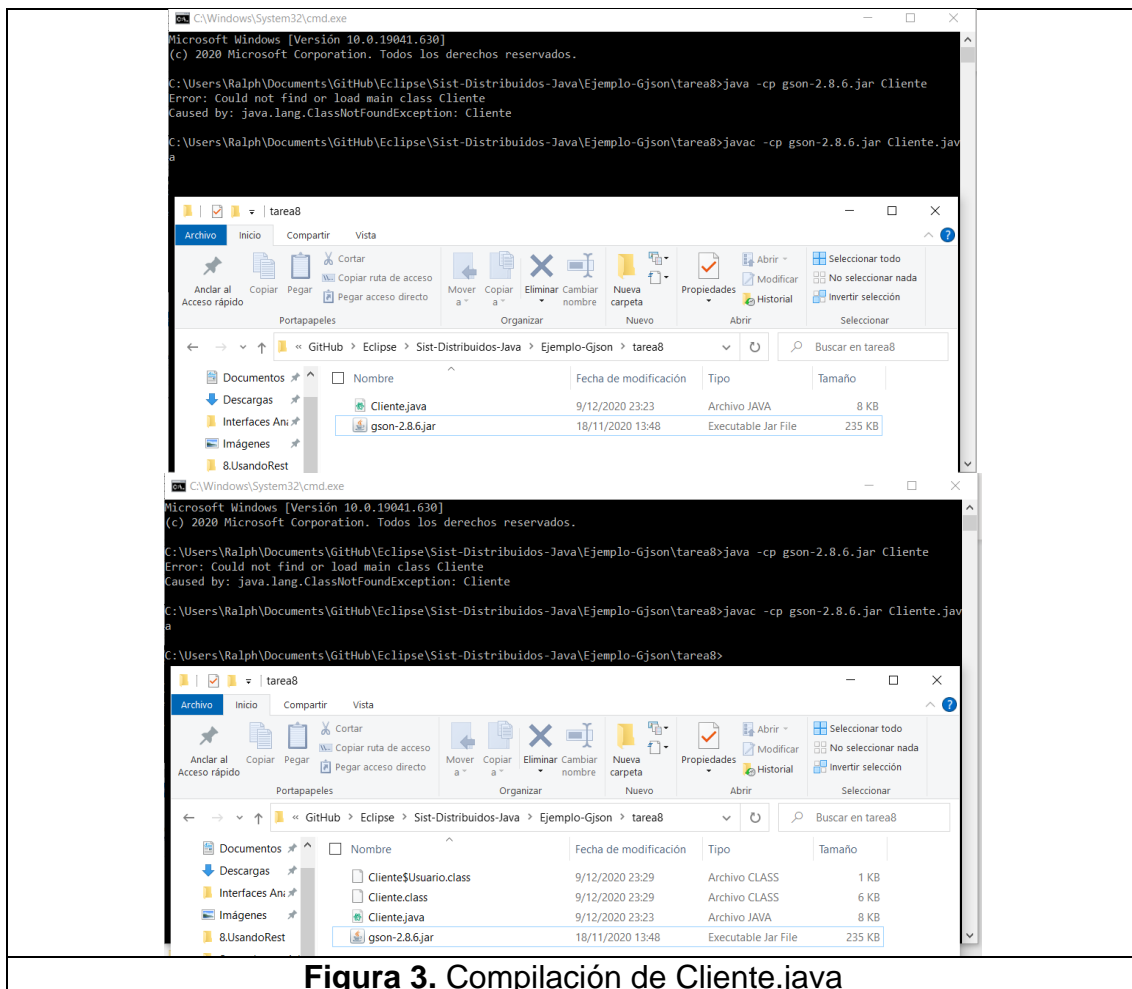
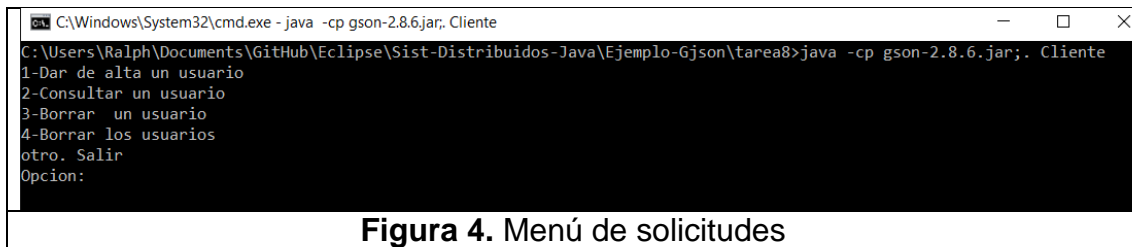


Figura 3. Compilación de Cliente.java

Finalmente nos desplegará un menú en donde haremos las siguientes solicitudes:

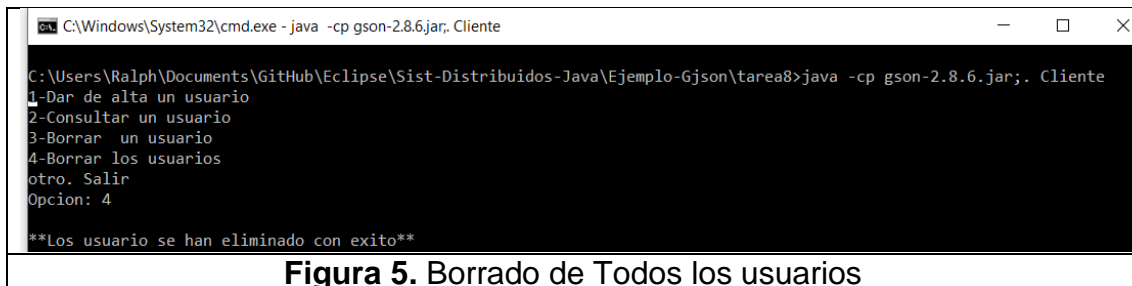
A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar; Cliente". The window shows the output of a Java application. The application displays a menu with five options: "1-Dar de alta un usuario", "2-Consultar un usuario", "3-Borrar un usuario", "4-Borrar los usuarios", and "otro. Salir". Below the menu, it prompts the user with "Opcion:". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar; Cliente
C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\Ejemplo-Gjson\tarea8>java -cp gson-2.8.6.jar;. Cliente
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion:
```

Figura 4. Menú de solicitudes

Borrar usuarios:

Borramos los usuarios que tengamos de pruebas anteriores

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar; Cliente". The window shows the output of a Java application. The application displays the same menu as in Figure 4. Below the menu, it prompts the user with "Opcion:". The user has entered "4". The application then outputs "**Los usuario se han eliminado con exito**". The window has standard Windows window controls in the top right corner.

```
C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar; Cliente
C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\Ejemplo-Gjson\tarea8>java -cp gson-2.8.6.jar;. Cliente
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion: 4
**Los usuario se han eliminado con exito**
```

Figura 5. Borrado de Todos los usuarios

Alta de usuario:

Damos de alta a un usuario y vemos lo que responde el servidor

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar; Cliente". The window shows the output of a Java application. The application displays the same menu as in Figure 4. Below the menu, it prompts the user with "Opcion:". The user has entered "1". The application then prompts for user details: "Email: rgarciag1305@alumno.ipn.mx", "Nombre: Rafael", "ApellidoPaterno: Garcia", "ApellidoMaterno: Garcia", "FechaNacimiento: 1997-12-12", "Telefono: 12345678", and "Genero: M". After the user enters "Ok", the application displays the same menu again. The window has standard Windows window controls in the top right corner.

```
C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar; Cliente
C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\Ejemplo-Gjson\tarea8>java -cp gson-2.8.6.jar;. Cliente
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion: 1
Email: rgarciag1305@alumno.ipn.mx
Nombre: Rafael
ApellidoPaterno: Garcia
ApellidoMaterno: Garcia
FechaNacimiento: 1997-12-12
Telefono: 12345678
Genero: M
Ok
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion:
```

Figura 6. Alta de usuario

Consulta de usuario:

Consultamos al usuario que dimos de alta


```
C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar, Cliente
Nombre: Rafael
ApellidoPaterno: Garcia
ApellidoMaterno: Garcia
FechaNacimiento: 1997-12-12
Telefono: 12345678
Genero: M
Ok
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion: 2
Email: rgarciag1305@alumno.ipn.mx

USUARIO ENCONTRADO:
Email: rgarciag1305@alumno.ipn.mx
Nombre: Rafael
Apellido Paterno: Garcia
Apellido Materno: Garcia
Fecha de nacimiento: 1997-12-12
Telefono: 12345678
Genero: M
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion:
```

Figura 7. Consulta de Usuario

Borrar un usuario:

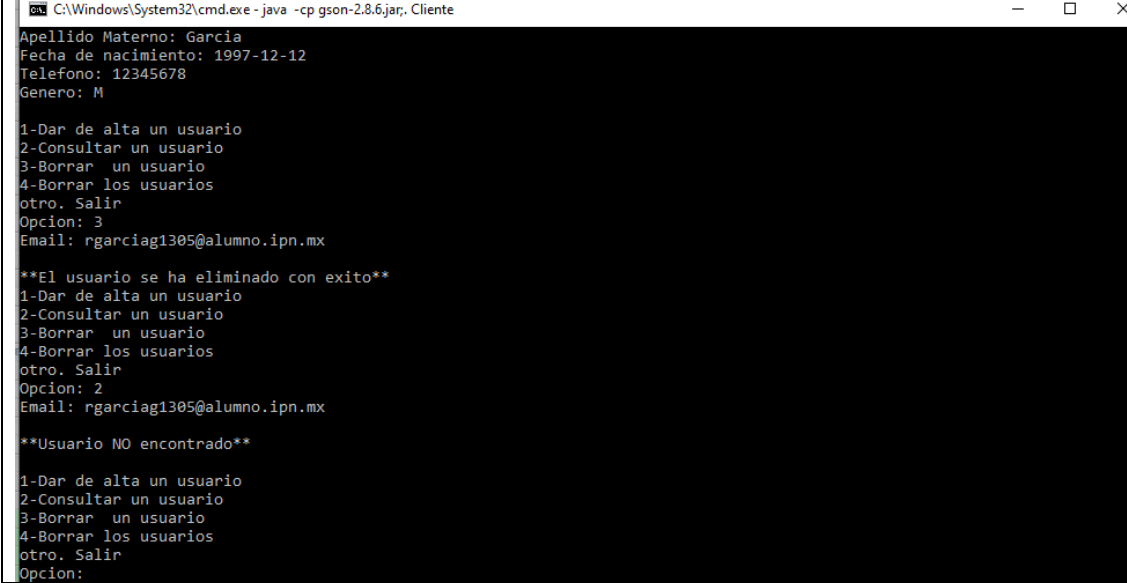
Borramos al usuario que dimos de alta y también lo borramos

```
C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar, Cliente
otro. Salir
Opcion: 2
Email: rgarciag1305@alumno.ipn.mx

USUARIO ENCONTRADO:
Email: rgarciag1305@alumno.ipn.mx
Nombre: Rafael
Apellido Paterno: Garcia
Apellido Materno: Garcia
Fecha de nacimiento: 1997-12-12
Telefono: 12345678
Genero: M
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion: 3
Email: rgarciag1305@alumno.ipn.mx

**El usuario se ha eliminado con éxito**
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion:
```

Figura 8. Borrado de Usuario



```
C:\Windows\System32\cmd.exe - java -cp gson-2.8.6.jar, Cliente
Apellido Materno: Garcia
Fecha de nacimiento: 1997-12-12
Telefono: 12345678
Genero: M

1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion: 3
Email: rgarciag1305@alumno.ipn.mx

**El usuario se ha eliminado con éxito**
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion: 2
Email: rgarciag1305@alumno.ipn.mx

**Usuario NO encontrado**
1-Dar de alta un usuario
2-Consultar un usuario
3-Borrar un usuario
4-Borrar los usuarios
otro. Salir
Opcion:
```

Figura 9. Consulta de Usuario Borrado

Como vimos no nos dio ningún problema ninguna solicitud y por ende el proyecto fue todo un éxito :D

Conclusiones:

Disfrute haber realizado esta práctica ya que desarrollar un cliente que consuma algún servicio WEB realizado es bastante satisfactorio, ya que el internet está construido haciendo uso de estos servicios. Cuando creamos un cliente tenemos que considerar que es lo que el usuario va a hacer para así hacer solicitudes a nuestra API en el servidor, posterior a la solicitud tenemos que esperar la respuesta del servidor para saber si las solicitudes se hicieron de manera correcta y así determinar que se hace con nuestras solicitudes. Del lado del servidor el servicio siempre tiene que actualizarse cada vez que se quiera añadir una función, se tiene que compilar y finalmente volver a crear el paquete del servicio para hacer uso de él.

Por último, puedo decir que pude entender la base del funcionamiento de estos servicios y creo que esto ha aportado valiosos conocimientos a mi desarrollo :D