

AE-2. Servicios REST

Grupo 13

Javier Franz Calle
Roberto Fernández Peña
David Gómez

Primero hemos hecho el servidor REST y hemos creado un array de videojuegos que se cargan en memoria cuando arranca el servicio:

```
@Component
public class DaoVideojuego {

    public List<Videojuego> listaVideojuegos;
    public int contador;

    public DaoVideojuego() {
        System.out.println("DaoVideojuego -> Creando la lista de videojuegos!");
        listaVideojuegos = new ArrayList<Videojuego>();
        Videojuego v1 = new Videojuego(contador++, "PACMAN", "NAMCO", 90); //ID: 0
        Videojuego v2 = new Videojuego(contador++, "SONIC", "SEGA", 95); //ID: 1
        Videojuego v3 = new Videojuego(contador++, "PANG", "CAPCOM", 98); //ID: 2
        Videojuego v4 = new Videojuego(contador++, "ARKANOID", "TAITO", 85); //ID: 3
        Videojuego v5 = new Videojuego(contador++, "CONTRA", "KONAMI", 87); //ID: 4
        listaVideojuegos.add(v1);
        listaVideojuegos.add(v2);
        listaVideojuegos.add(v3);
        listaVideojuegos.add(v4);
        listaVideojuegos.add(v5);
    }
}
```

En la misma clase DaoVideojuego, hemos escrito los métodos que proporcionarán el servicio CRUD (se pueden leer en el código).

Para comprobar el correcto funcionamiento del servidor y sus métodos, vamos a utilizar POSTMAN. Primero, probaremos el método GET:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/videojuegos/1`. The response body is a JSON object:

```
1  {
2      "id": 1,
3      "nombre": "SONIC",
4      "compania": "SEGA",
5      "nota": 95
6  }
```

También comprobamos que nos devuelva la lista completa de videojuegos:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/videojuegos/`. The response body is a JSON array containing three video game objects:

```
1 [ { "id": 0, "nombre": "PACMAN", "compania": "NAMCO", "nota": 90 }, { "id": 1, "nombre": "SONIC", "compania": "SEGA", "nota": 95 }, { "id": 2, "nombre": "PANG", "compania": "CAPCOM", "nota": 98 } ]
```

Vamos a probar el método POST, dando de alta un nuevo videojuego.

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/videojuegos/`. The request body is a JSON object representing a new video game:

```
1 { "nombre": "QBERT", "compania": "HASBRO", "nota": 80 }
```

The response status is 201 Created, indicating the game was successfully added. The response body shows the newly created game object:

```
1 [ { "id": 5, "nombre": "QBERT", "compania": "HASBRO", "nota": 80 } ]
```

Comprobamos que se ha creado correctamente'.

Comprobamos después el método PUT, sustituyendo el videojuego con id 5 (Qbert) por otro diferente (Loom):

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** http://localhost:8080/videojuegos/5
- Body:** JSON (selected)
- JSON Body:**

```

1
2   "nombre": "LOOM",
3   "compania": "LUCASFILM",
4   "nota": 75
5

```
- Response Headers:** 200 OK, 13 ms, 123 B
- Buttons:** Send, Save Response

Por último comprobamos el método DELETE:

The screenshot shows a POSTMAN interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8080/videojuegos/5
- Body:** JSON (selected)
- JSON Body:**

```

1
2   "id": 5,
3   "nombre": "LOOM",
4   "compania": "LUCASFILM",
5   "nota": 75
6

```
- Response Headers:** 200 OK, 8 ms, 221 B
- Buttons:** Send, Save Response

Como se puede ver, el servicio CRUD funciona al completo.

Requerimiento 2:

Lo que hacemos es antes de dar la orden de dar de alta es comparar el ID del nuevo con los ya existentes. Si no existe creamos el videojuego, y en caso contrario devolvemos un error 409.

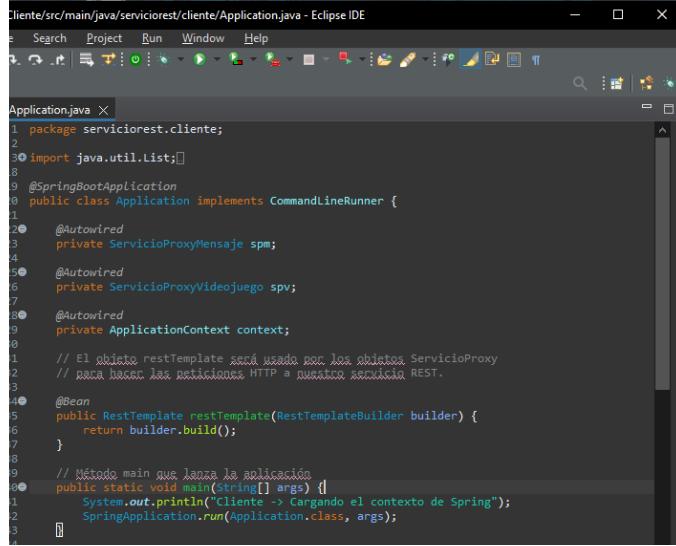
```

//POST
@PostMapping(path="videojuegos",consumes=MediaType.APPLICATION_JSON_VALUE,
produces=MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Videojuego> altaVideojuego(@RequestBody Videojuego v) {
    for (int i = 0; i < daoVideojuego.altaVideojuego.contador; i++) {
        if (v.getId() == daoVideojuego.list().get(i).getId()) {
            return new ResponseEntity<Videojuego>(v,HttpStatus.CONFLICT); //409 CONFLICT , ID ya registrado
        }
    }
    System.out.println("altaVideojuego: objeto videojuego: " + v);
    daoVideojuego.add(v);
    return new ResponseEntity<Videojuego>(v,HttpStatus.CREATED); //201 CREATED
}

```

CLIENTE REST

Para crear el cliente empezamos por la aplicación principal (Application) donde vamos primero a insertar las dependencias y el Bean.

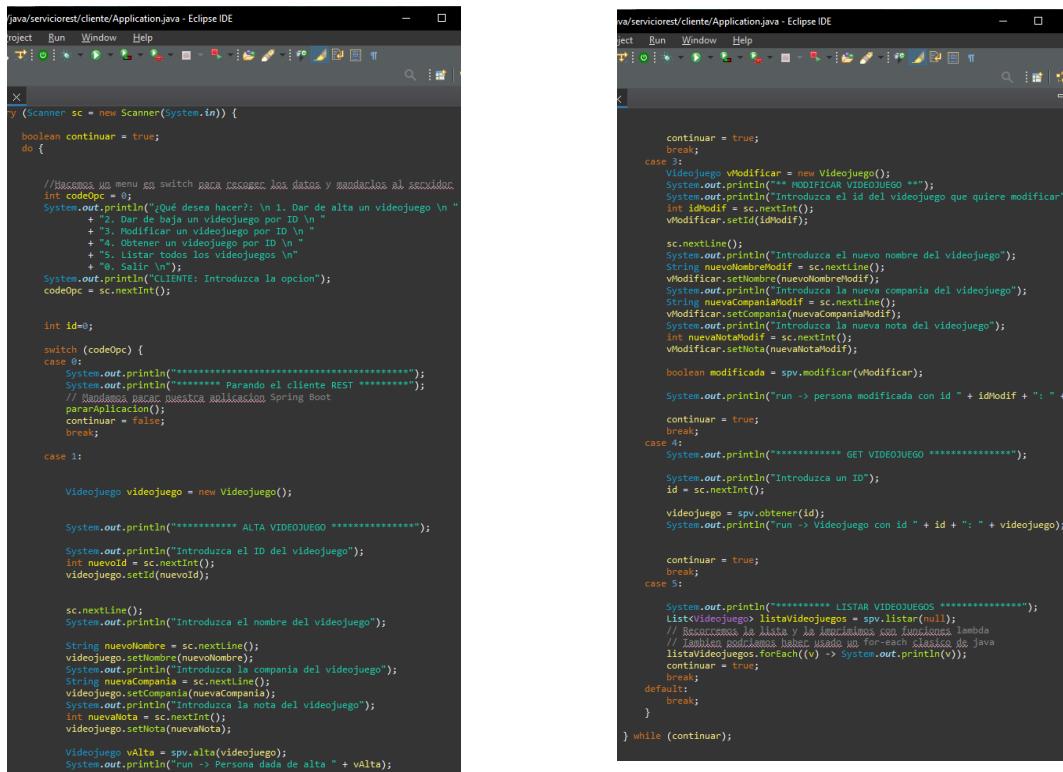


```
Cliente/src/main/java/serviciorest/cliente/Application.java - Eclipse IDE
Search Project Run Window Help
Application.java X
1 package serviciorest.cliente;
2
3 import java.util.List;
4
5 @SpringBootApplication
6 public class Application implements CommandLineRunner {
7
8     @Autowired
9     private ServicioProxyMensaje spm;
10
11     @Autowired
12     private ServicioProxyVideojuego spv;
13
14     @Autowired
15     private ApplicationContext context;
16
17     // El objeto restTemplate será usado por los objetos ServicioProxy
18     // para hacer las peticiones HTTP a nuestro servicio REST.
19
20     @Bean
21     public RestTemplate restTemplate(RestTemplateBuilder builder) {
22         return builder.build();
23     }
24
25     // Método main que lanza la aplicación
26     public static void main(String[] args) {
27         System.out.println("Cliente -> Cargando el contexto de Spring");
28         SpringApplication.run(Application.class, args);
29     }
}
```

A continuación empezamos a comunicarnos con el usuario a través de una interfaz sencilla

y recogeremos la respuesta en una variable para posteriormente usarla en un switch.

Este switch será el que realiza la función definida por el usuario mientras recoge más datos y los manda al servidor, tales como Id o nombre, compañía y precio si la opción lo requiere.



```
java/serviciorest/cliente/Application.java - Eclipse IDE
project Run Window Help
X
try (Scanner sc = new Scanner(System.in)) {
    boolean continuar = true;
    do {

        // Hacemos un switch para recoger los datos y mandarlos al servidor
        int codeOpc = 0;
        System.out.println("¿Qué deseas hacer?: \n 1. Dar de alta un videojuego \n"
                           + " 2. Dar de baja un videojuego por ID \n"
                           + " 3. Modificar un videojuego por ID \n"
                           + " 4. Listar todos los videojuegos \n"
                           + " 5. Salir \n");
        System.out.print("CLIENTE: Introduzca la opción");
        codeOpc = sc.nextInt();

        int id;
        switch (codeOpc) {
        case 0:
            System.out.println("***** Parando el cliente REST *****");
            // Mandamos una petición al servidor Spring Boot
            paraAplicacion();
            continuar = false;
            break;
        case 1:

            Videojuego videojuego = new Videojuego();

            System.out.println("***** ALTA VIDEOJUEGO *****");
            System.out.print("Introduzca el ID del videojuego");
            int nuevodId = sc.nextInt();
            videojuego.setId(nuevodId);

            sc.nextLine();
            System.out.print("Introduzca el nombre del videojuego");
            String nuevoNombre = sc.nextLine();
            videojuego.setNombre(nuevoNombre);
            System.out.print("Introduzca la compañía del videojuego");
            String nuevaCompania = sc.nextLine();
            videojuego.setcompania(nuevaCompania);
            System.out.print("Introduzca la nota del videojuego");
            int nuevalNota = sc.nextInt();
            videojuego.setNota(nuevalNota);

            Videojuego vAlta = spv.alta(videojuego);
            System.out.println("run -> Persona dada de alta " + vAlta);
        }
    }
}
```



```
java/serviciorest/cliente/Application.java - Eclipse IDE
project Run Window Help
X
continuar = true;
break;
case 3:
    Videojuego vModificar = new Videojuego();
    System.out.println("***** MODIFICAR VIDEOJUEGO ***");
    System.out.print("Introduzca el id del videojuego que quiere modificar");
    int idModif = sc.nextInt();
    vModificar.setId(idModif);

    sc.nextLine();
    System.out.println("Introduzca el nuevo nombre del videojuego");
    String nuevoNombreModif = sc.nextLine();
    vModificar.setNombre(nuevoNombreModif);
    System.out.println("Introduzca la nueva compañía del videojuego");
    String nuevaCompaniaModif = sc.nextLine();
    vModificar.setcompania(nuevaCompaniaModif);
    System.out.println("Introduzca la nueva nota del videojuego");
    int nuevalNotaModif = sc.nextInt();
    vModificar.setNota(nuevalNotaModif);

    boolean modificada = spv.modificar(vModificar);
    System.out.println("run -> persona modificada con id " + idModif + ": " + modificada);
    continuar = true;
    break;
case 4:
    System.out.println("***** GET VIDEOJUEGO *****");
    System.out.print("Introduzca un ID");
    id = sc.nextInt();
    Videojuego g = spv.obtener(id);
    System.out.println("run -> Videojuego con id " + id + ": " + g);

    continuar = true;
    break;
case 5:
    System.out.println("***** LISTAR VIDEOJUEGOS *****");
    List<Videojuego> listaVideojuegos = spv.listar(null);
    // Recorremos la lista y la imprimimos con funciones lambda
    // Usando mapas para hacer uso de un for-each clásico de java
    listaVideojuegos.forEach(v -> System.out.println(v));
    continuar = true;
    break;
default:
    break;
}
} while (continuar);
}
```

Despues creamos el servicio proxy que incluye las funciones de dar de alta, dar de baja, modificar, buscar, y listar todos (podriamos decir que son Post, Delete, Update y 2 Get (uno con filtro y otro sin ello)).

```
/*
 * XXXXX
 */
public Videojuego obtener(int id){
    try {
        ResponseEntity<Videojuego> re = restTemplate.getForEntity(URL + id, Videojuego.class);
        HttpStatus hs = re.getStatusCode();
        if(hs == HttpStatus.OK) {
            return re.getBody();
        }else {
            System.out.println("obtener -> Respuesta no contemplada");
            return null;
        }
    }catch (HttpClientErrorException e) { //XXXXX 4XX
        System.out.println("obtener -> El videojuego NO se ha encontrado, id: " + id);
        System.out.println("obtener -> Codigo de respuesta: " + e.getStatusCode());
        return null;
    }
}
```

```
/*
 */
public Videojuego alta(Videojuego v){
    try {
        ResponseEntity<Videojuego> re = restTemplate.postForEntity(URL, v, Videojuego.class);
        System.out.println("alta -> Codigo de respuesta " + re.getStatusCode());
        return re.getBody();
    } catch (HttpClientErrorException e) { //XXXXX 4XX
        System.out.println("alta -> El videojuego NO se ha dado de alta, id: " + v.getId());
        System.out.println("alta -> Codigo de respuesta: " + e.getStatusCode());
        return null;
    }
}
```

```
/*
 */
public boolean modificar(Videojuego v){
    try {
        restTemplate.put(URL + v.getId(), v, Videojuego.class);
        return true;
    } catch (HttpClientErrorException e) {
        System.out.println("modificar -> El videojuego NO se ha modificado, id: " + v.getId());
        System.out.println("modificar -> Codigo de respuesta: " + e.getStatusCode());
        return false;
    }
}
```

```
/*
 */
public boolean borrar(int id){
    try {
        restTemplate.delete(URL + id);
        return true;
    } catch (HttpClientErrorException e) {
        System.out.println("borrar -> El videojuego NO se ha borrado, id: " + id);
        System.out.println("borrar -> Codigo de respuesta: " + e.getStatusCode());
        return false;
    }
}
```

```
/*
 */
public List<Videojuego> listar(String nombre){
    String queryParams = "";
    if(nombre != null) {
        queryParams += "?nombre=" + nombre;
    }

    try {
        //http://localhost:8080/personas?nombre=harry GET
        ResponseEntity<Videojuego[]> response =
            restTemplate.getForEntity(URL + queryParams, Videojuego[].class);
        Videojuego[] arrayVideojuegos = response.getBody();
        return Arrays.asList(arrayVideojuegos); //convertimos el array en un ArrayList
    } catch (HttpClientErrorException e) {
        System.out.println("listar -> Error al obtener la lista de videojuegos");
        System.out.println("listar -> Codigo de respuesta: " + e.getStatusCode());
        return null;
    }
}
```

Probando el cliente REST

Ejecutamos prierio el erver y luego el cliente. Podemos ver como se crea el entorno spring.

The screenshot shows two terminal windows. The left window displays the logs for the Spring Boot application (serviciorest) running on port 8080. It includes logs for the main application class and various Tomcat and Spring components. The right window shows the logs for the REST client (serviciorest.cliente.Application) interacting with the server. It includes a menu of options for managing video games, such as alta (create), baja (delete), modificar (update), obtener (get), and listar (list). The client also sends a message to the server at `http://localhost:8080/mensajeHTML`.

```
2022-11-21 21:31:25.819 INFO 12508 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : 
2022-11-21 21:31:25.629 INFO 12508 --- [main] o.apache.catalina.core.StandardService : 
2022-11-21 21:31:25.630 INFO 12508 --- [main] org.apache.catalina.core.StandardEngine : 
2022-11-21 21:31:25.766 INFO 12508 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : 
2022-11-21 21:31:25.767 INFO 12508 --- [main] w.s.c.ServletWebServerApplicationContext : 
2022-11-21 21:31:26.095 INFO 12508 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : 
2022-11-21 21:31:26.103 INFO 12508 --- [main] serviciorest.cliente.Application : 
***** Arrancando el cliente REST ***** 
***** MENSAJE ***** 
obtener -> Ruta final: http://localhost:8080/mensaje 
run -> Mensaje: Esto es un mensaje para confirmar la conexion 
***** MENSAGE HTML ***** 
obtener -> Ruta final: http://localhost:8080/mensajeHTML 
run -> Mensaje: <!DOCTYPE html><html><head><title>Prueba html</title></head><body><h1>Hol</h1></body> 
¿Qué desea hacer?: 
1. Dar de alta un videojuego 
2. Dar de baja un videojuego por ID 
3. Modificar un videojuego por ID 
4. Obtener un videojuego por ID 
5. Listar todos los videojuegos 
0. Salir 
CLIENTE: Introduzca la opcion
```

Se conecta el cliente al servidor

The screenshot shows the logs for the client (serviciorest.cliente.Application) connecting to the server. The client sends a message to the server at `http://localhost:8080/mensaje`, and the server responds with the message "Servicio Rest -> Contexto de Spring cargado!".

```
2022-11-21 21:31:12.233 INFO 10532 --- [main] serviciorest.Application : 
Servicio Rest -> Contexto de Spring cargado! 
2022-11-21 21:31:26.180 INFO 10532 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : 
2022-11-21 21:31:26.180 INFO 10532 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : 
2022-11-21 21:31:26.181 INFO 10532 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : 
```

Probamos la primera opcion

The screenshot shows the client interacting with the server. The client sends a message to the server at `http://localhost:8080/borrarVideojuego`. The server responds with the message "Servicio Rest -> Contexto de Spring cargado!", indicating that the new video game object has been created successfully.

```
CLIENTE: Introduzca la opcion 
1 ***** ALTA VIDEOJUEGO ***** 
Introduzca el ID del videojuego 
5 
Introduzca el nombre del videojuego 
Call of Duty 
Introduzca la compania del videojuego 
Activision 
Introduzca la nota del videojuego 
80 
alta -> Codigo de respuesta 201 CREATED 
run -> Persona dada de alta Videojuego [id=5, nombre=Call of Duty, compania=Activision, nota=80] 
¿Qué desea hacer?: 
1. Dar de alta un videojuego 
2. Dar de baja un videojuego por ID 
3. Modificar un videojuego por ID 
4. Obtener un videojuego por ID 
5. Listar todos los videojuegos 
0. Salir 
CLIENTE: Introduzca la opcion
```

Como vemos se crea el nuevo objeto y nos devuelve la informacion introducida. Pasamos a la opcion 2 donde daremos de baja ese mismo videojuegos

The screenshot shows the client interacting with the server. The client sends a message to the server at `http://localhost:8080/borrarVideojuego`. The server responds with the message "Servicio Rest -> Contexto de Spring cargado!", indicating that the video game object has been deleted successfully.

```
endencies 0. Salir 
CLIENTE: Introduzca la opcion 
2 ***** BORRAR VIDEOJUEGO ***** 
Introduzca un ID 
5 
run -> Persona con id 5 borrada? true 
¿Qué desea hacer?: 
1. Dar de alta un videojuego 
2. Dar de baja un videojuego por ID 
3. Modificar un videojuego por ID 
4. Obtener un videojuego por ID 
5. Listar todos los videojuegos 
0. Salir 
CLIENTE: Introduzca la opcion
```

El cliente nos comunica la confirmacion de que se ha borrado el objeto que estaba en la posicion 5 (el juego introducido anteriormente).

A continuacion probaremos la opcion 3: modificar un videojuego. Modificaremos el primer juego.

```
CLIENTE: Introduzca la opcion
3
** MODIFICAR VIDEOJUEGO **
Introduzca el id del videojuego que quiere modificar
1
Introduzca el nuevo nombre del videojuego
TETRIS
Introduzca la nueva compania del videojuego
Blue Planet
Introduzca la nueva nota del videojuego
100
run -> persona modificada con id 1: true
¿Qué desea hacer?:
1. Dar de alta un videojuego
```

Aprovecharemos para porbar la opcion 4 buscando el juego que acabamos de modificar.

```
3. Modificar un videojuego por ID
4. Obtener un videojuego por ID
5. Listar todos los videojuegos
0. Salir

CLIENTE: Introduzca la opcion
4
***** GET VIDEOJUEGO *****
Introduzca un ID
1
run -> Videojuego con id 1: Videojuego [id=1, nombre=TETRIS, compania=Blue Planet, nota=100]
¿Qué desea hacer?:
1. Dar de alta un videojuego
2. Dar de baja un videojuego por ID
3. Modificar un videojuego por ID
4. Obtener un videojuego por ID
```

Finalmente probamos la opcion 5 y la opcion 0 que son listar todos los videojuegos y cerrar la aplicacion respectivamente.

```
0. Salir

CLIENTE: Introduzca la opcion
5
***** LISTAR VIDEOJUEGOS *****
Videojuego [id=0, nombre=PACMAN, compania=NAMCO, nota=90]
Videojuego [id=1, nombre=TETRIS, compania=Blue Planet, nota=100]
Videojuego [id=2, nombre=PANG, compania=CAPCOM, nota=98]
Videojuego [id=3, nombre=ARKANOID, compania=TAITO, nota=85]
Videojuego [id=4, nombre=CONTRA, compania=KONAMI, nota=87]
¿Qué desea hacer?:
1. Dar de alta un videojuego
2. Dar de baja un videojuego por ID
3. Modificar un videojuego por ID
4. Obtener un videojuego por ID
5. Listar todos los videojuegos
0. Salir

2. Dar de baja un videojuego por ID
3. Modificar un videojuego por ID
4. Obtener un videojuego por ID
5. Listar todos los videojuegos
0. Salir

CLIENTE: Introduzca la opcion
0
*****
***** Parando el cliente REST *****
```

