

SISTEMAS ORIENTADOS A SERVICIOS 2021

Práctica: Diseño e Implementación de un Servicio Web. RESTful.

AUTOR: Jason Felipe Vaz

UPM Informática

ÍNDICE

1. Resumen del Diseño de la Base de Datos.....	2
i. Diseño Del servicio RESTful.....	3
ii. 1. Añadir usuario a la red de lectura.....	4
iii. 2. Ver los datos básicos de un usuario:.....	5
iv. 3. Cambiar datos básicos del usuario (excepto el nombre de usuario):	5
v. 4. Obtener un listado con los usuarios de red, se permite el filtrado por nombre.....	6
vi. 5. Borrar Nuestro perfil de la red	7
vii. 6. Añadir la lectura de un libro por un usuario con una calificación.....	7
viii. 7. Eliminar la lectura de un libro por un usuario.	8
ix. 8. Editar un libro de la red.....	9
x. 9. Consultar los últimos libros leídos por un usuario, donde se podrá filtrar por fecha	10
xi. 10. Obtener información del libro de cierto usuario.....	11
xii. 11. Añadir amigo dentro de la red de lectura	11
xiii. 12. Eliminar amigo.....	12
xiv. 13. Obtener una lista de todos Nuestros Amigos, donde se puede limitar el número de amigos.....	12
xv. 14. Buscar en libros recomendados por nuestros amigos	13
xvi. 15. Aplicación Móvil	14
xvii. Estructura de objetos en memoria.....	16
2. Capturas de la ejecución de las operaciones anteriores	17
3. Capturas de la ejecución de las operaciones anteriores en el cliente.....	32
4. Datos de la entrega	33
5. Comentario final	33

1. Resumen del Diseño de la Base de Datos

Se ha construido una base de datos que constará de 3 tablas fundamentales: Usuarios, Lecturas y Amigos.

A continuación, mostraremos el diagrama de entidad-relación que se ha utilizado en esta práctica:

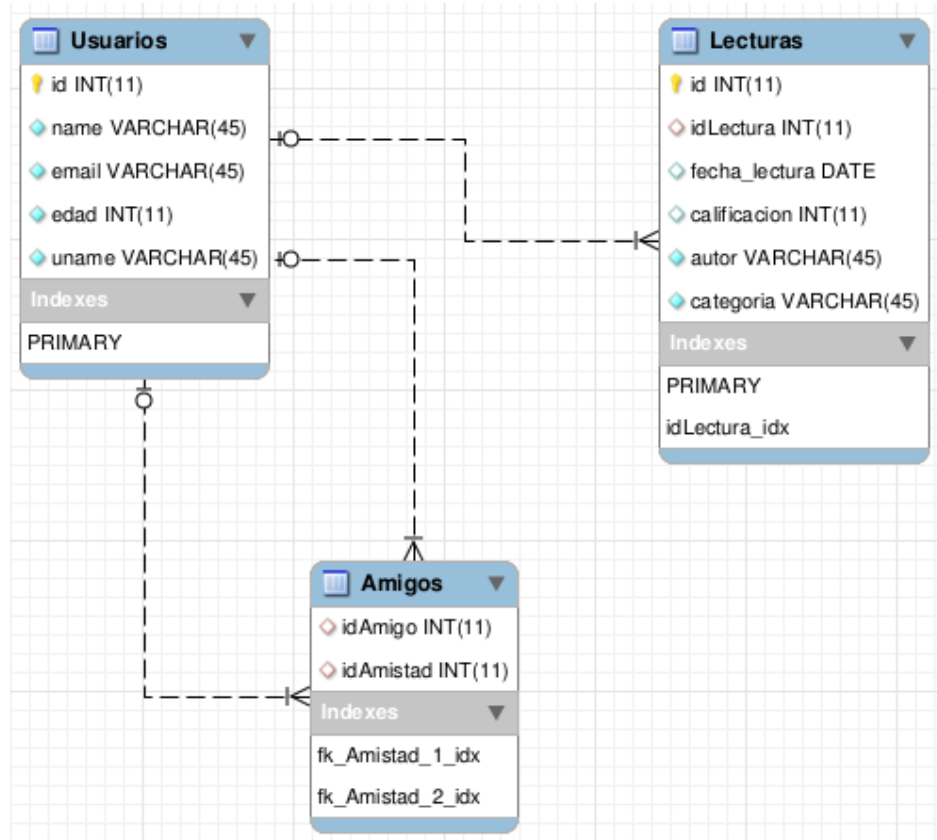


IMAGEN 1: DIAGRAMA DE ENTIDAD RELACIÓN (1-N)

- En la primera tabla de **Usuarios** la clave primaria será el **id**.
- En **Lecturas** la clave primaria será también **id**, y donde **idLectura** será la clave foránea que apunta al **id** de la tabla de Usuarios.
- En tabla de **Amigos**, habrás 2 claves indexadas que harás que claves foráneas, **idAmigo** e **idAmistad** que apuntarán al **id** de la tabla de **Usuarios**, de esta manera se podrán realizar las relaciones de amistad con mayor eficacia.

Además, como para la prueba del servicio se ha optado por utilizar una BBDD, se incluirá un fichero sql.txt, con el código SQL de creación de la BBDD, y también con las sentencias INSERT que se han considerado necesarias para la ejecución del cliente realizado. También se adjuntará el usuario y password requerida para la BBDD.

Diseño Del servicio RESTful (XML/JSON)

OPERACIONES(USUARIO)

- **GET** */usuarios*
 - Devuelve la lista de usuarios en la red de lectura
 - **Parámetros: name**
 - Filtra por nombre
- **GET** */usuarios*
 - Devuelve la información básica del usuario de la red de lectura
- **POST** */usuarios/{idUser}*
 - Introduce un usuario a la red de lecturas
- **PUT** */usuarios/{idUser}*
 - Actualiza la información básica del usuario
- **DELETE** */usuarios/{idUser}*
 - Borra el perfil cierto usuario de la red de lectura

OPERACIONES(LECTURA)

- **GET** */usuarios/{idUser}/lectura*
 - Devuelve las lecturas del usuario anteriores a cierta fecha.
 - **Parámetros: fecha**
 - Filtra por fecha, devolviendo los datos anteriores a esta fecha.
- **GET** */usuarios/{idUser}/lectura/{idLectura}*
 - Devuelve la información de la lectura
- **DELETE** */usuarios/{idUser}/{idLibro}*
 - Elimina una lectura determinada de la lista de lecturas del usuario.
- **PUT** */usuarios/{idUser}/{idLectura}*
 - Actualiza la lectura de un libro del usuario
- **POST** */usuarios/{idUser}/{idLibro}*
 - Añade una lectura a un usuario con una calificación.

OPERACIONES(AMIGOS)

- **GET** */usuarios/{idUser}/amigos*
 - Devuelve la lista de amigos del usuario.
 - **Parámetros: limite**
 - Limita la información a devolver, por defecto devuelve todo.
- **DELETE** */usuarios/{idUser}/amigos/{id_amigo}*
 - Elimina un amigo de la lista de amistad.
- **POST** */usuarios/amigos/recomendaciones*
 - Añade a un amigo dentro de la red de lectura

OPERACIONES(RECOMENDACIONES)

- **GET** */usuarios/amigos/recomendaciones*
 - Devuelve los libros recomendados por sus amigos
 - **Parámetros:**
 - **calificación:** devuelve recomendaciones mayores a dicha calificación.
 - **autores:** filtra por el nombre de autores
 - **categoría:** filtra por la categoría del libro.

OPERACIONES(MÓVIL)

- **GET /usuarios/{idUser}/movil**
 - Devuelve la aplicación móvil, con los datos del usuario, nº de amigos, Uris hacia los últimos libros leídos por sus amigos, y el último libro leído por el usuario.

1. Añadir usuario a la red de lectura:

POST: <http://localhost:8080/RESTFUL/api/usuarios>

Cabecera a usar= <http://localhost:8080/RESTFUL/api/usuarios>

- **Cuerpo de la petición (application/xml):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user>
    <correo>Jason100ry@gmail.com</correo>
    <edad>20</edad>
    <name>Jason </name>
    <userName>Jason99</userName>
</user>
```

- **Cuerpo de la petición(application/JSON):**

```
{
    "correo": " Jason100ry@gmail.com ",
    "edad": 20,
    "name": " Jason ",
    "userName": " Jason99"
}
```

- **Código de Respuesta: 201**
Cabeceras: <http://localhost:8080/RESTFUL/api/usuarios/4>
- **Código de Respuesta: 400**
Error al crear el usuario.

2. Ver los datos básicos de un usuario:

GET: <http://localhost:8080/RESTFUL/api/usuarios/{idUser}>

Cabecera a utilizar: <http://localhost:8080/RESTFUL/api/usuarios/4>

- **Código de respuesta:** 200
 - **Cabecera:** <http://localhost:8080/RESTFUL/api/usuarios/4>
 - **Representación(application/XML)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<user id="4">
```

```
  <correo>Jason100ry@gmail.com</correo>
```

```
  <edad>20</edad>
```

```
  <name>Jason </name>
```

```
  <userName>Jason99</userName>
```

```
</user>
```

- **Representación(application/JSON)**

```
{
  "ID": 4,
  "correo": "Jason@gmail.com",
  "edad": 20,
  "name": "Jason",
  "userName": "Jason99"
}
```
- **Código de Respuesta:** 404
 - **Cuerpo:** Elemento no encontrado.

3. Cambiar datos básicos del usuario (excepto el nombre de usuario):

PUT: <http://localhost:8080/RESTFUL/api/usuarios/{idUser}>

Cabecera a utilizar: <http://localhost:8080/RESTFUL/api/usuarios/2>

- **Cuerpo de la petición:**
 - **Representación(application/XML)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<user >
```

```
  <correo>Jason@gmail.com</correo>
```

```
  <edad>29</edad>
```

```
  <name>Jason</name>
```

```
</user>
```

- **Representación(application/JSON)**

```
{
  "correo": "Jason@gmail.com",
  "edad": 29,
  "name": "Jason"
}
```

- **Código de respuesta: 200**

- **Cabeceras:** Content-location: <http://localhost:8080/RESTFUL/api/usuarios/2>

- **Representación(application/XML)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user id="2">
  <correo>Jason@gmail.com</correo>
  <edad>29</edad>
  <name>Jason</name>
  <userName>jason</userName>
</user>
```

- **Representación(application/JSON)**

```
{
  "ID": 2,
  "correo": "Jason@gmail.com",
  "edad": 29,
  "name": "Jason",
  "userName": "jason"
}
```

- **Código de Respuesta: 401**

- **Cuerpo:** No se ha encontrado el ID del usuario

- **Código de Respuesta: 400**

- **Cuerpo:** No se pudieron convertir los índices a números

4. [Obtener un listado con los usuarios de red, se permite el filtrado por nombre:](#)

GET: <http://localhost:8080/RESTFUL/api/usuarios>

Cabecera a utilizar: <http://localhost:8080/RESTFUL/api/usuarios?name=J>

- **Código de respuesta: 200**

- **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios?name=J>

- **Representación (application/XML)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<usuarios>
```

```
  <usuario name="Jason" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/2"/>
```

```
  <usuario name="Jason" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/4"/>
```

```
</usuarios>
```

Representación(application/JSON)

```
{
  "users": [
    {
      "name": "Jason",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/2"
    },
    {
      "name": "Jason",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/4"
    }
  ]
}
```

5. [Borrar Nuestro perfil de la red](#)

DELETE: <http://localhost:8080/RESTFUL/api/usuarios/{idUser}>

Cabecera a utilizar: http://localhost:8080/RESTFUL/api/usuarios/4

- **Código de respuesta:** 204, Borrado correcto
- **Código de respuesta:** 404
 - **Cuerpo:** No se pudo eliminar el Usuario

6. [Añadir la lectura de un libro por un usuario con una calificación](#)

POST: <http://localhost:8080/RESTFUL/api/usuarios/{id}/lectura>

Cabecera a utilizar: http://localhost:8080/RESTFUL/api/usuarios/1/lectura

- **Cuerpo de la petición(application/xml)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lectura>
  <fechaLectura>2002-03-03</fechaLectura>
  <autor>Manuel</autor>
  <calificacion>10</calificacion>
  <categoria>Musical</categoria>
  <idLect>1</idLect>
</lectura>
```


- **Cuerpo de la petición(application/JSON)**

```
{
  "autor": " Manuel ",
  "calificacion": 10,
  "categoria": " Musical",
  "fechaLectura": "2002-03-03",
  "idLect": 1
}
```

- **Código de respuesta: 200**

- **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios/1/lectura>
- **Representación(application/xml):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<lectura idLectura="10">
```

```
    <autor>Manuel</autor>
```

```
    <calificacion>10</calificacion>
```

```
    <categoria>Musical</categoria>
```

```
    <idLect>1</idLect>
```

```
</lectura>
```

- **Representación(application/json):**

```
{
  "autor": "Manuel",
  "calificacion": 10,
  "categoria": "Musical",
  "idLect": 1,
}
```

- **Código de respuesta: 400**

- **Cuerpo:** No se puede añadir lectura.

7. [Eliminar la lectura de un libro por un usuario.](#)

DELETE: <http://localhost:8080/RESTFUL/api/usuarios/{id}/lectura/{idLectura}>

Cabecera a utilizar: <http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2>

- **Código de respuesta: 204**

- **Código de respuesta: 404**

- **Cuerpo:** No se puede eliminar la lectura.

8. [Editar un libro de la red](#)

PUT: `http://localhost:8080/RESTFUL/api/usuarios/{id}/lectura/{idLectura}`

Cabecera a usar: `http://localhost:8080/RESTFUL/api/usuarios/1/lectura/8`

- **Cuerpo de la petición(application/xml):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lectura>
    <autor>Federico Garcia</autor>
    <calificacion>10</calificacion>
    <categoria>Novela</categoria>
    <fechaLectura>2004-04-04</fechaLectura>
</lectura>
```

- **Cuerpo de la petición(application/Json):**

```
{
  "autor": "Federico Garcia",
  "calificacion": 10,
  "categoria": "Novela",
  "fechaLectura": 2002-04-04,
}
```

- **Código de respuesta: 200**

- **Cabeceras:** `http://localhost:8080/RESTFUL/api/usuarios/1/lectura/8`

En XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lectura fechaLectura="2021-04-23T22:20:19.962+02:00" idLectura="1">
  <autor>Federico Garcia</autor>
  <calificacion>10</calificacion>
  <categoria>Novela</categoria>
  <idLect>1</idLect>
</lectura>
```

En JSON:

```
{
  "autor": "Federico Garcia",
  "calificacion": 10,
  "categoria": "Novela",
  "fechaLectura": "2021-04-23T20:20:37.71Z[UTC]",
  "idLect": 1,
  "idLectura": 1
}
```

- **Código de respuesta: 400**

- **Cuerpo:** No se pudo actualizar la lectura del usuario.

- **Código de respuesta: 404**

- **Cuerpo:** No se ha encontrado el Id de lectura.

9. [Consultar los últimos libros leídos por un usuario, donde se podrá filtrar por fecha](#)

GET: <http://localhost:8080/RESTFUL/api/usuarios/{id}/lectura>

Cabecera a usar: <http://localhost:8080/RESTFUL/api/usuarios/1/lectura?fecha=2021-04-21>

- **Código de respuesta:** 200
 - **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios/1/lectura?fecha=2021-04-21>
 - **Representación(application/xml):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lectura>
  <lecturas date="2004-02-05" rel="self"
href="http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1"/>
  <lecturas date="2004-04-05" rel="self"
href="http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"/>
</lectura>
```

- **Representación(application/JSON):**

```
{
  "lecturas": [
    {
      "date": "2004-02-05",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1"
    },
    {
      "date": "2004-04-05",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"
    }
  ]
}
```

- **Código de respuesta:** 404
 - **Cuerpo:** No se ha podido parsear el id del usuario.

10. Obtener información del libro de cierto usuario

GET: <http://localhost:8080/RESTFUL/api/usuarios/{id}/lectura/{idLectura}>

- **Cabecera a usar:** <http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1>
- **Código de respuesta:** 200
 - **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1>
 - **Representación(application/xml)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lectura fechaLectura="2021-04-25T09:56:25.754+02:00" idLectura="1">
  <autor>Cervantez</autor>
  <calificacion>5</calificacion>
  <categoria>Novela</categoria>
  <idLect>1</idLect>
</lectura>
```
 - **Representación(application/JSON):**

```
{
  "autor": "Cervantez",
  "calificacion": 5,
  "categoria": "Novela",
  "fechaLectura": "2021-04-25T07:55:29.679Z[UTC]",
  "idLect": 1,
  "idLectura": 1
}
```
- **Código de respuesta:** 404
 - **Cuerpo:** No se ha podido parsear el id de lectura.

11. Añadir amigo dentro de la red de lectura

POST: <http://localhost:8080/RESTFUL/api/usuarios/{id}/amigos>

Cabecera a usar: <http://localhost:8080/RESTFUL/api/usuarios/2/amigos>

- **Cabecera a usar:** <http://localhost:8080/RESTFUL/api/usuarios/2/amigos>
- **Cuerpo de la petición(application/xml)**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<enviaSolicitud>
  <idSolicitante>Henry99</idSolicitante>
  <msgSolicitud>Mensaje</msgSolicitud>
</enviaSolicitud>
```
- **Cuerpo de la petición(application/JSON)**

```
{
  "idSolicitante": "Henry99",
  "msgSolicitud": "mensaje"
}
```

- **Código de Respuesta:** 201
 - **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios/2/amigos>
 - **Representación(xml):**

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<usuarios>

<usuario name="Henry" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/3"/>

</usuarios>

- **Representación (JSON):**

```
{
  "name": "Henry",
  "rel": "self",
  "url": "http://localhost:8080/RESTFUL/api/usuarios/3"
}
```

- **Código de respuesta:** 400
 - **Cuerpo:** no se pudo enviar la solicitud

12. [Eliminar amigo](#)

- **DELETE:** http://localhost:8080/RESTFUL/api/usuarios/{id}/amigos/{id_amigo}
- **Cabecera a usar:** <http://localhost:8080/RESTFUL/api/usuarios/2/amigos/3>
- **Código de respuesta:** 204, borrado correctamente
- **Código de respuesta:**
 - **Cuerpo:** No se ha encontrado el id del amigo

13. [Obtener una lista de todos Nuestros Amigos, donde se puede limitar el número de amigos](#)

GET: <http://localhost:8080/RESTFUL/api/usuarios/{id}/amigos>

Cabecera a usar: <http://localhost:8080/RESTFUL/api/usuarios/1/amigos?limite=2>

- **Código de respuesta:** 200
 - **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios/1/amigos?limite=2>
 - **Representación(application/xml):**

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<usuarios>

<usuario rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/amigos/2"/>

<usuario rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/amigos/3"/>

</usuarios>

- **Representación(application/json):**

```
{
  "users": [
    {
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/amigos/2"
    },
    {
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/amigos/3"
    }
  ]
}
```

14. Buscar en libros recomendados por nuestros amigos.

GET: <http://localhost:8080/RESTFUL/api/usuarios/{id}/amigos/recomendaciones>

Cabecera a usar:

<http://localhost:8080/RESTFUL/api/usuarios/1/amigos/recomendaciones?calificacion=5&categoria=Novela&autor=Charles>

- **Código de respuesta: 200**

- **Cabeceras:**

<http://localhost:8080/RESTFUL/api/usuarios/1/amigos/recomendaciones?calificacion=5&categoria=Novela&autor=Charles>

- **Representación(application/xml):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<recomendaciones>
```

```
  <recomendacion autor="Charles" calificaciones="10" categoria="Novela" rel="self"
href="http://localhost:8080/RESTFUL/api/usuarios/1/amigos/recomendaciones/5"/>
```

```
</recomendaciones>
```

- **Representación(application/json):**

```
{
  "recomendacion": [
    {
      "autor": "Charles",
      "calificaciones": 10,
      "categoria": "Novela",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/amigos/recomendaciones/5"
    }
  ]
}
```

15. Aplicación Móvil

GET: <http://localhost:8080/RESTFUL/api/usuarios/{id}/movil>

Cabecera a usar: <http://localhost:8080/RESTFUL/api/usuarios/1/movil>

- **Código de respuesta:** 200
 - **Cabeceras:** <http://localhost:8080/RESTFUL/api/usuarios/1/movil>
 - **Representación(application/xml):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<movil>
  <amigos>2</amigos>
  <LibrosAmigos idAmigo="2" rel="self"
href="http://localhost:8080/RESTFUL/api/usuarios/2/lectura/4"/>
  <LibrosAmigos idAmigo="3" rel="self"
href="http://localhost:8080/RESTFUL/api/usuarios/3/lectura/6"/>
  <ultimoLibro rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"/>
  <user id="1">
    <correo>pepe@gmail.com</correo>
    <edad>21</edad>
    <name>Pepe</name>
    <userName>Pepe99</userName>
  </user>
</movil>
```

- **Representación(application/json):**

```
{
  "amigos": 2,
  "listaLectura": [
    {
      "idAmigo": 2,
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/2/lectura/4"
    },
    {
      "idAmigo": 3,
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/3/lectura/6"
    }
  ],
  "ultimoLibro": [
    {
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"
    }
  ],
  "user": {
    "ID": 1,
    "correo": "pepe@gmail.com",
    "edad": 21,
    "name": "Pepe",
    "userName": "Pepe99"
  }
}
```


Estructura de objetos en memoria

La Estructuración del código ha sido la siguiente:

- El fichero de **clase.BDD**: tendrá la clase principal `API.java`, con todas las operaciones implementadas para esta práctica.
- La carpeta de **clase.datos**: Está formado por:
 - **Amigos.java**: clase de amigos, donde se define el id del amigo y el id de amistad.
 - **enviaSolicitud.java**: Clase donde se define la estructura del mensaje de solicitud de amistad.
 - **Lectura.java**: Clase donde se define la estructura de lecturas del usuario.
 - **movil.java**: Clase donde se define la estructura de la aplicación móvil.
 - **User.java**: Clase donde se define la estructura del usuario.
- La carpeta de **clase.memoria**: Está formado por:
 - **Link.java**: Estructura de datos que guardará la uri del usuario.
 - **LinkFriend.java**: Estructura de datos que guardará la uri de la lista de amigos.
 - **LinkMovil.java**: Estructura de datos que guardará la uri de los amigos del usuario.
 - **LinkRecomendaciones.java**: Estructura de datos que guardará la uri de la lista de recomendaciones de los usuarios.
 - **Serv.java**: Estructura que guardará en Array de listas la estructura de Link.java.
 - **ServfriendMovil.java**: Estructura que guardará en Array de listas la estructura de Link.java.
 - **ServfriendMovil2.java**: Estructura que guardará en Array de listas la estructura de LinkMovil.java.
 - **servLectura.java**: Estructura que guardará en Array de listas la estructura de LinkFriend.java.
 - **servRecomendaciones.java**: Estructura que guardará en Array de listas la estructura de LinkRecomendaciones.java.
- La carpeta de **cliente**: contendrá el código del cliente implementado tanto para XML como para JSON.

2. Capturas de la ejecución de las operaciones anteriores

The first screenshot shows a REST client interface with a POST request to `http://localhost:8080/RESTFUL/api/usuarios`. The 'Body' tab is selected, and the 'raw' radio button is chosen. The payload is XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user>
  <correo>Jason100ry@gmail.com</correo>
  <edad>20</edad>
  <name>Jason </name>
  <userName>Jason99</userName>
</user>
```

 The status bar shows 'Status: 201 Created', 'Time: 588 ms', and 'Size: 407 B'. The second screenshot shows the same request but with the 'JSON (application/json)' radio button selected. The payload is JSON:

```
{
  "correo": " Jason100ry@gmail.com ",
  "edad": 20,
  "name": " Jason ",
  "userName": " Jason99"
}
```

 The status bar shows 'Status: 201 Created', 'Time: 61 ms', and 'Size: 324 B'.

Figura 1: Añade un usuario a la red de lectura rellenando los datos básicos. Donde se le asigna el id=5 al añadirlo en nuestra base de datos de usuario. **(Añadir usuario a la red de lectura)**

Figura 2: Es la misma representación, pero usando JSON. **(Añadir usuario a la red de lectura)**

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/4 Params Send Save ▼

Authorization Headers (3) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets ▼
<input type="checkbox"/> Content-Type	application/json				
<input type="checkbox"/> Accept	application/json				
<input checked="" type="checkbox"/> Accept	application/xml				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 29 ms Size: 286 B

Pretty Raw Preview XML ▼ ≡

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <user id="4">
3   <correo> Jason100ry@gmail.com </correo>
4   <edad>20</edad>
5   <name> Jason </name>
6   <userName> Jason99</userName>
7 </user>

```

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/4 Params Send Save ▼

Authorization Headers (3) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets ▼
<input type="checkbox"/> Content-Type	application/json				
<input checked="" type="checkbox"/> Accept	application/json				
<input type="checkbox"/> Accept	application/xml				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 69 ms Size: 199 B

Pretty Raw Preview JSON ▼ ≡

```

1 {
2   "ID": 4,
3   "correo": " Jason100ry@gmail.com ",
4   "edad": 20,
5   "name": " Jason ",
6   "userName": " Jason99"
7 }

```

Figura 1: Muestra los datos básicos del usuario con id=4. (Ver los datos básicos de un usuario)

Figura 2: Misma representación, pero en JSON (Ver los datos básicos de un usuario)

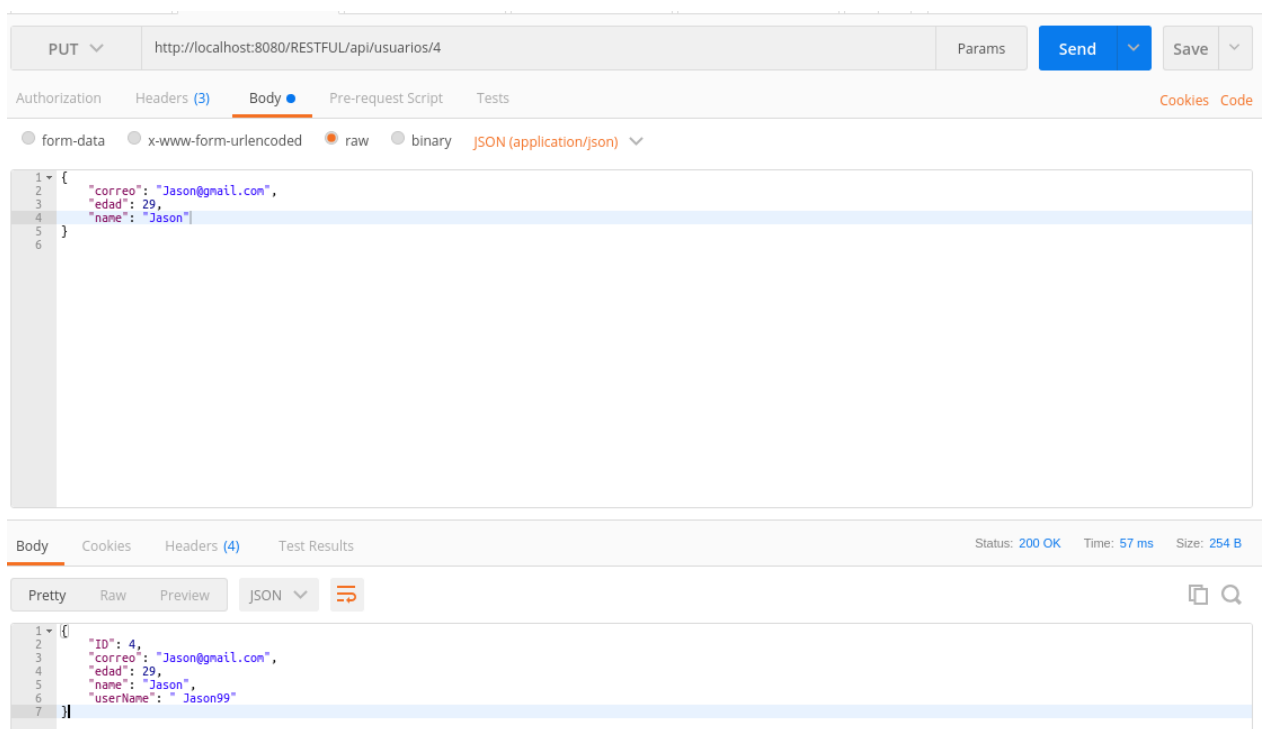
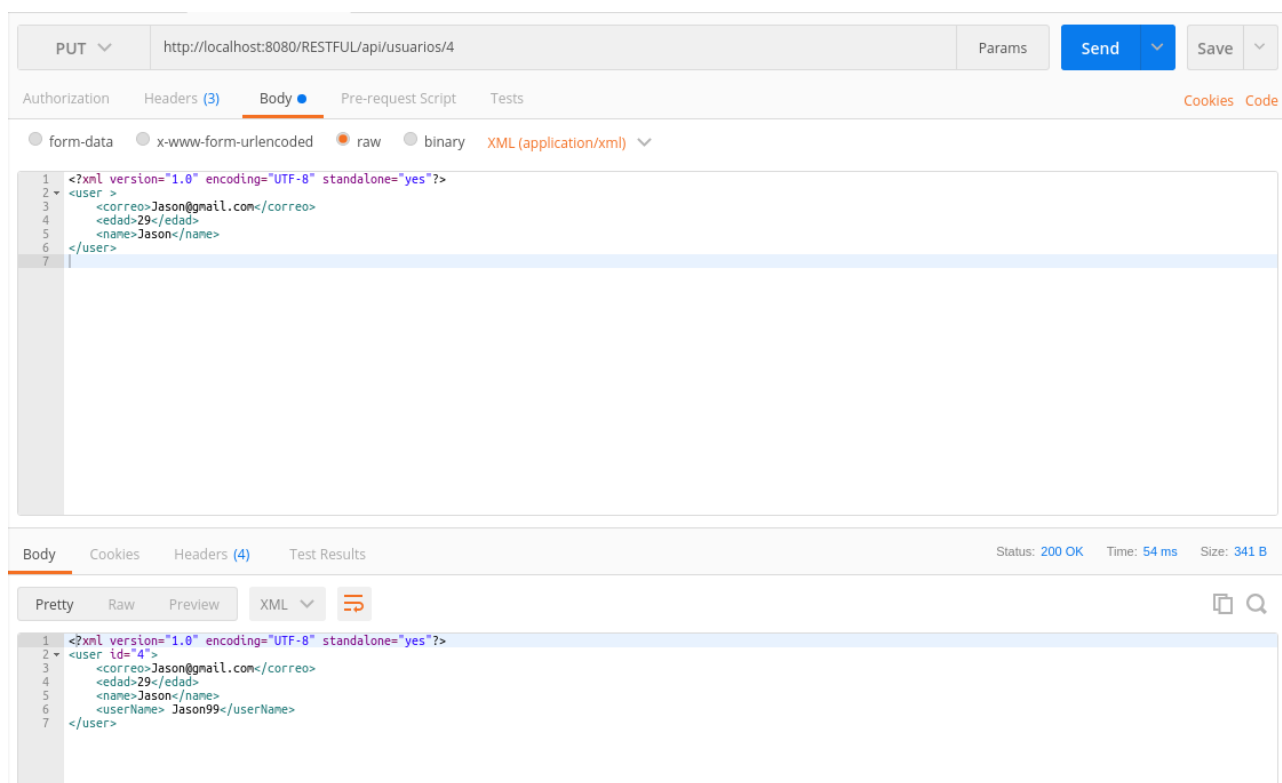


Figura 1: Modifica los datos básicos del usuario con id=4, menos el nombre de usuario que no se podrá modificar. **(Cambiar datos básicos del usuario (excepto el nombre de usuario))**

Figura 2: Misma representación, pero en JSON (Cambiar datos básicos del usuario (excepto el nombre de usuario))

The image displays two screenshots of a REST client interface, likely Postman, showing the results of an API call to `http://localhost:8080/RESTFUL/api/usuarios?name=j`.

Top Screenshot (XML):

- Method:** GET
- URL:** `http://localhost:8080/RESTFUL/api/usuarios?name=j`
- Status:** 200 OK
- Time:** 31 ms
- Size:** 356 B
- Headers:** Content-Type: application/json, Accept: application/json, Accept: application/xml
- Body (XML):**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<usuarios>
  <usuario name="Jason" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/2"/>
  <usuario name="Jason" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/4"/>
</usuarios>
```

Bottom Screenshot (JSON):

- Method:** GET
- URL:** `http://localhost:8080/RESTFUL/api/usuarios?name=j`
- Status:** 200 OK
- Time:** 95 ms
- Size:** 286 B
- Headers:** Content-Type: application/json, Accept: application/json, Accept: application/xml
- Body (JSON):**

```
{
  "users": [
    {
      "name": "Jason",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/2"
    },
    {
      "name": "Jason",
      "rel": "self",
      "url": "http://localhost:8080/RESTFUL/api/usuarios/4"
    }
  ]
}
```

Figura 1: Muestra la lista de usuarios, donde se filtra por nombre, que empiecen por “J”. (Obtener un listado con los usuarios de red, se permite el filtrado por nombre)

Figura 2: Misma representación, pero en JSON. (Obtener un listado con los usuarios de red, se permite el filtrado por nombre)

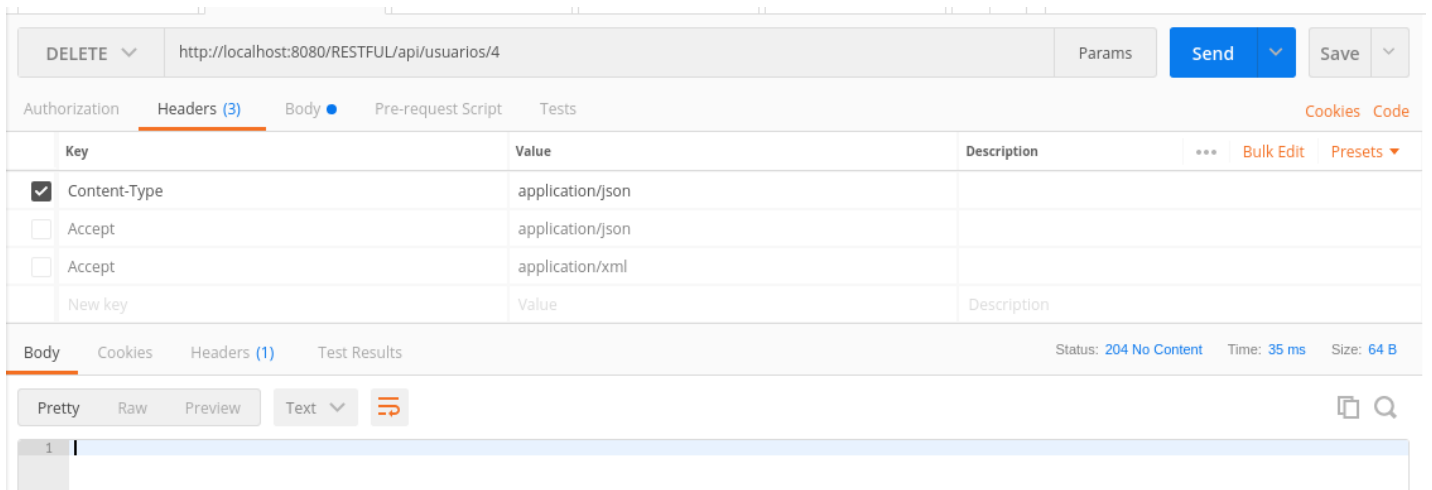


Figura 1: Borrará al usuario con id=4, y solo devolverá su Status correspondiente por ellos no se mostrará otra imagen en JSON ya que es lo mismo.

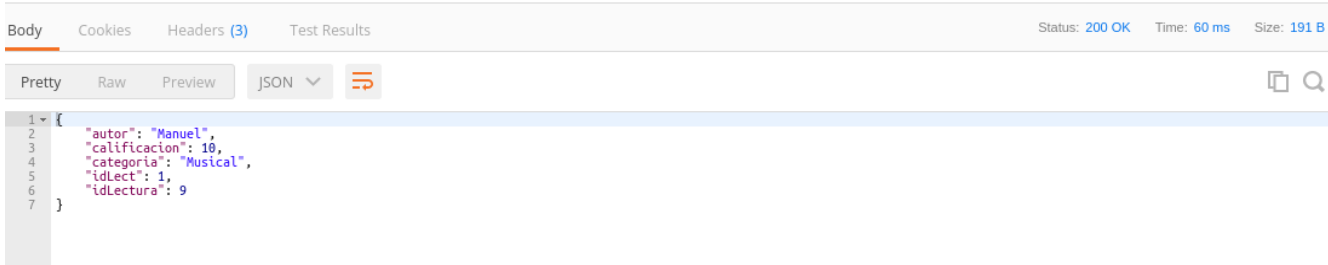
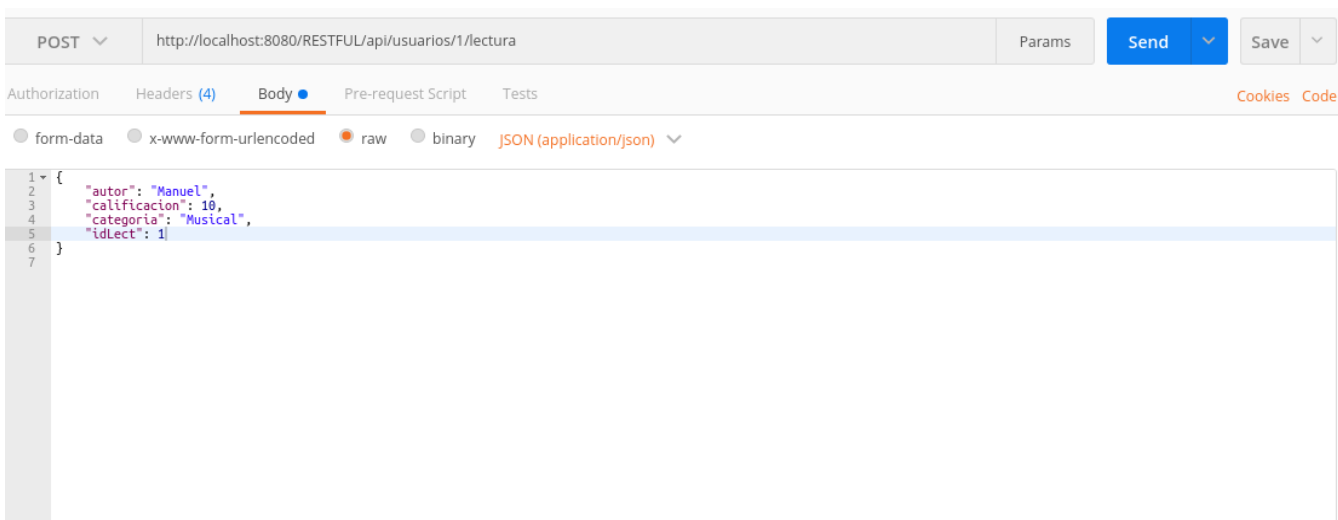
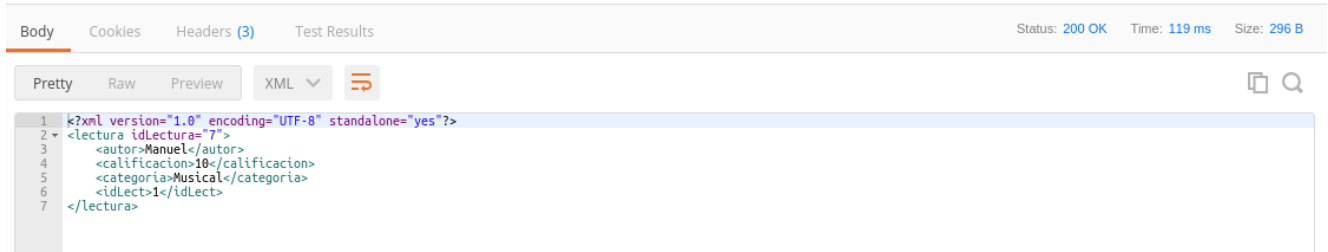
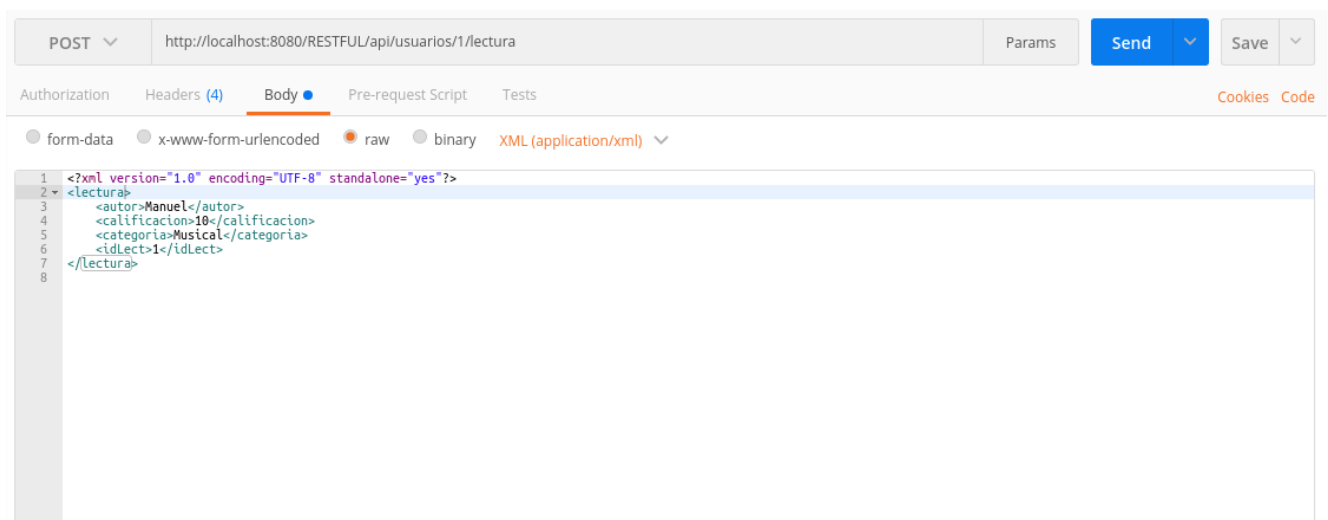


Figura 1: Crea una lectura para el usuario con id=1, y al crearse la lectura, se le asigna a dicha lectura un id de lectura=9. Además de sus correspondientes datos del libro como la calificación. **(Añadir la lectura de un libro por un usuario con una calificación)**

Figura 2: Misma representación, pero en JSON. **(Añadir la lectura de un libro por un usuario con una calificación).** **NOTA:** me he percatado que falta el nombre del libro.

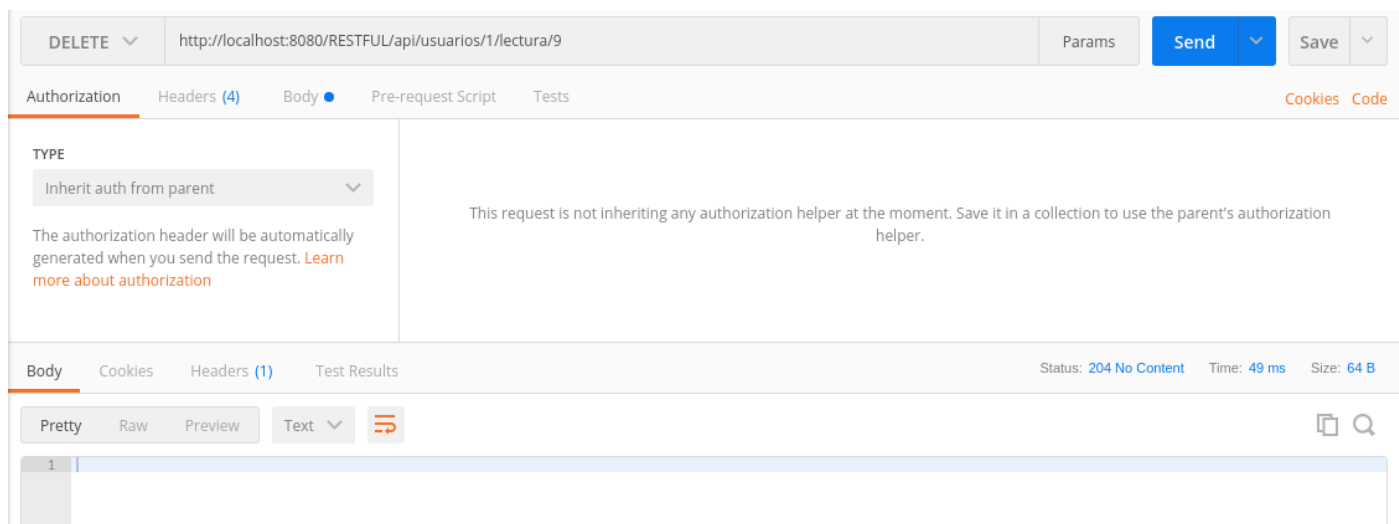


Figura 1: Elimina una de las lecturas del usuario con id=1, en este caso eliminar la lectura con idLectura=9. (Eliminar la lectura de un libro por un usuario)

The image shows a REST client interface with two panels. The top panel displays an XML PUT request to the endpoint `http://localhost:8080/RESTFUL/api/usuarios/1/lectura/8`. The request body is an XML document with the following structure:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <lectura>
3   <autor>Federico Garcia</autor>
4   <calificacion>10</calificacion>
5   <categoria>Novela</categoria>
6   <fechaLectura>2004-04-04</fechaLectura>
7 </lectura>
```

The bottom panel shows the response, which is a JSON object. The status is 200 OK, with a response time of 123 ms and a size of 367 B. The response body is:

```
1 {
2   "autor": "Federico Garcia",
3   "calificacion": 10,
4   "categoria": "Novela",
5   "fechaLectura": "2021-04-25T07:49:55.161Z[UTC]"
6 }
7
```

The interface also shows tabs for Authorization, Headers (4), Body, Pre-request Script, and Tests. The Body tab is selected, and the content is displayed in a Pretty view. The status bar at the bottom indicates the response status, time, and size.

Figura 1: Modifica la lectura con idLectura=8 del usuario con id=1, en el resultado aparecen los datos ya modificados con toda la información de la lectura que se está modificando. **(Editar un libro de la red)**

Figura 2: Misma representación, pero en JSON. **(Editar un libro de la red)**

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/1/lectura?fecha=2021-04-21 Params Send Save ▼

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets ▼
<input type="checkbox"/> Content-Type	application/json				
<input type="checkbox"/> Accept	application/json				
<input checked="" type="checkbox"/> Accept	application/xml				
<input type="checkbox"/> Content-Type	xm				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 334 ms Size: 386 B

Pretty Raw Preview XML ▼ ≡

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <lecturas>
3   <lecturas date="2004-02-05" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1"/>
4   <lecturas date="2004-04-05" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"/>
5 </lecturas>

```

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/1/lectura?fecha=2021-04-21 Params Send Save ▼

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

TYPE

Inherit auth from parent ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Body Cookies Headers (3) Test Results Status: 200 OK Time: 93 ms Size: 319 B

Pretty Raw Preview JSON ▼ ≡

```

1 {
2   "lecturas": [
3     {
4       "date": "2004-02-05",
5       "rel": "self",
6       "url": "http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1"
7     },
8     {
9       "date": "2004-04-05",
10      "rel": "self",
11      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"
12    }
13  ]
14 }

```

Figura 1: Muestra las lecturas del usuario con id=1 pero además filtra por fecha, para mostrar los libros anteriores a 2021-04-21. (Consultar los últimos libros leídos por un usuario, donde se podrá filtrar por fecha)

Figura 2: misma representación, pero en JSON. (Consultar los últimos libros leídos por un usuario, donde se podrá filtrar por fecha)

The image displays two screenshots of a REST client interface, likely Postman, showing the results of a GET request to the endpoint `http://localhost:8080/RESTFUL/api/usuarios/1/lectura/1`. The request headers are set to `Accept: application/xml` and `Content-Type: xm`.

Figure 1 (Top Screenshot): Shows the response in XML format. The status is 200 OK, time is 84 ms, and size is 342 B. The XML body is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lectura fechaLectura="2021-04-25T09:54:55.583+02:00" idLectura="1">
  <autor>Cervantez</autor>
  <calificacion>5</calificacion>
  <categoria>Novela</categoria>
  <idLect>1</idLect>
</lectura>
```

Figure 2 (Bottom Screenshot): Shows the same response in JSON format. The status is 200 OK, time is 45 ms, and size is 240 B. The JSON body is as follows:

```
{
  "autor": "Cervantez",
  "calificacion": 5,
  "categoria": "Novela",
  "fechaLectura": "2021-04-25T07:55:29.679Z[UTC]",
  "idLect": 1,
  "idLectura": 1
}
```

Figura 1: Muestra la información del libro con idLectura=1 del usuario con id=1. **(Obtener información del libro de cierto usuario)**

Figura 2: Misma representación, pero usando JSON. **(Obtener información del libro de cierto usuario)**

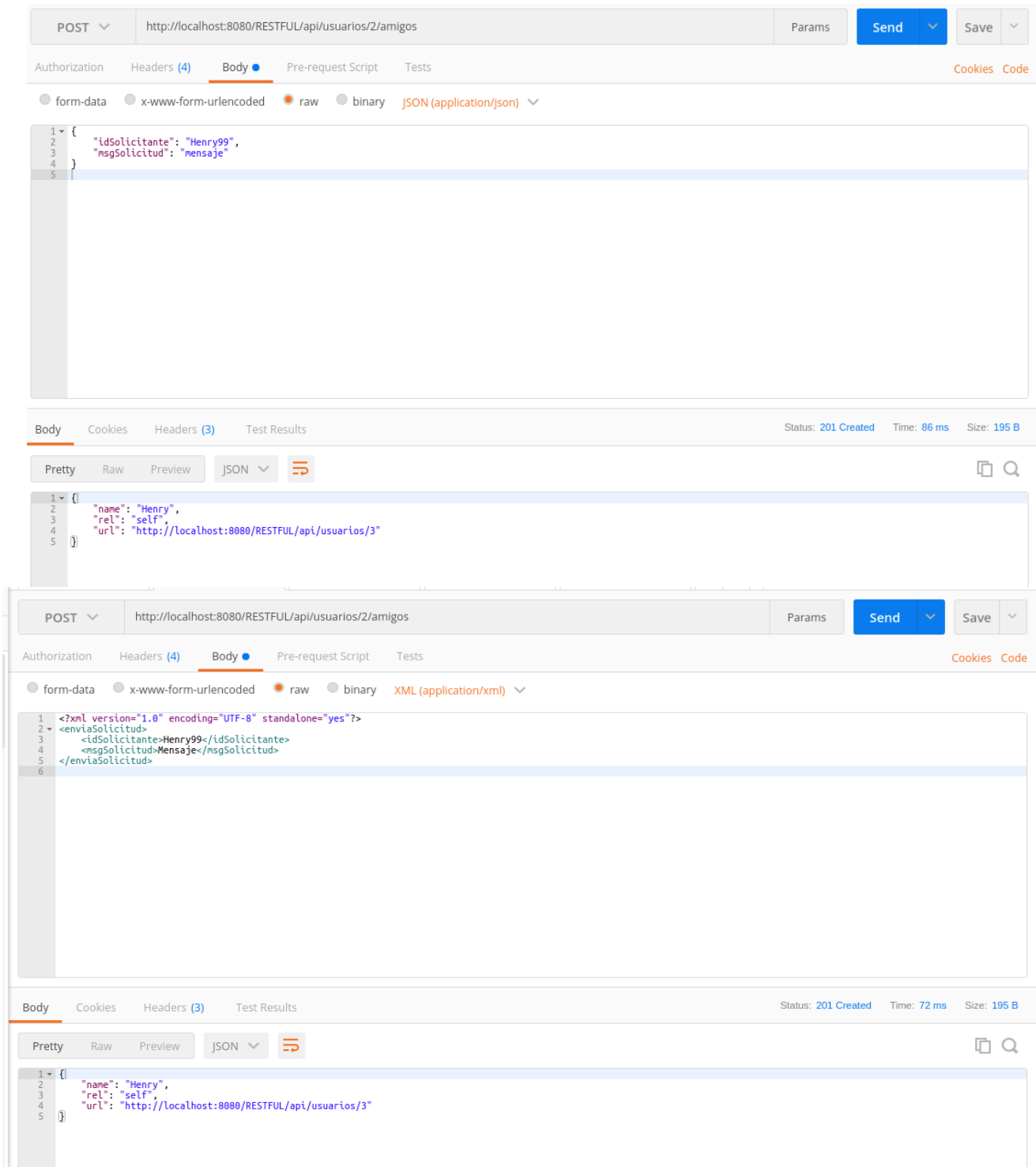


Figura 1: Se añade un amigo al usuario con id=2, para realizar dicho proceso de amistad se pasará el **nombre de usuario** del usuario del que quiere ser amigo junto con un mensaje de solicitud, y se obtendrá como resultado la información (una uri accesible a esta información) del usuario al que se ha enviado la solicitud de amistad. **(Añadir amigo dentro de la red de lectura)**

Figura 2: Misma representación, pero usando JSON. **(Añadir amigo dentro de la red de lectura)**

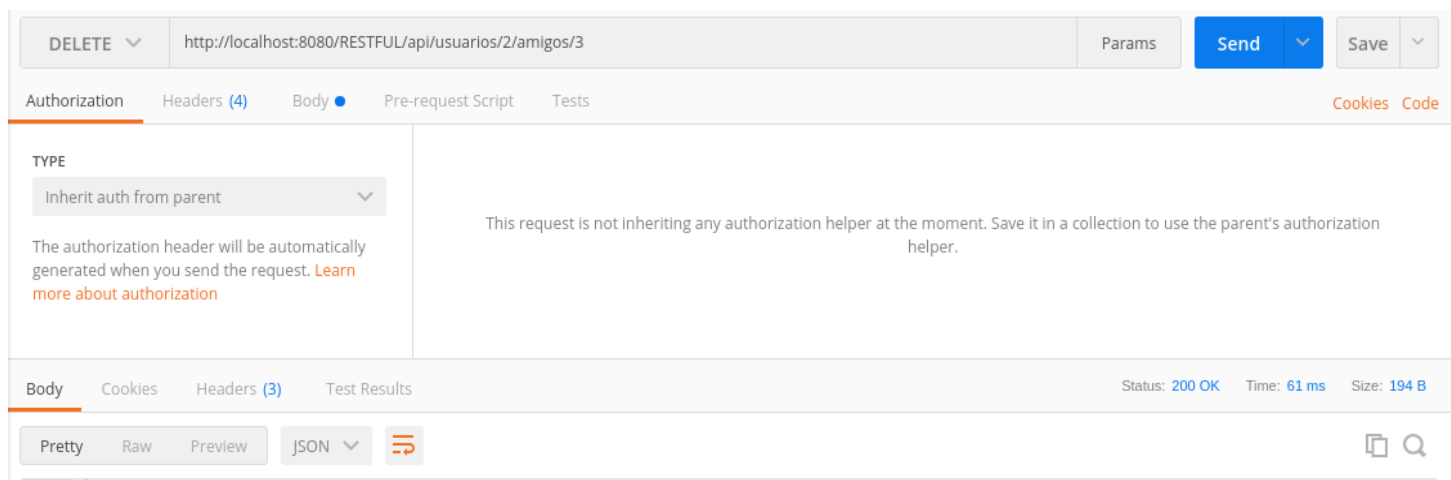


Figura 1: Elimina de la lista de amigos del usuario con id=2, eliminando al usuario con id=3. **(Eliminar amigo)**

Figura 2: Misma representación, pero usando JSON. **(Obtener información del libro de cierto usuario)**

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/1/amigos?limite=2 Params Send ▼ Save ▼

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> limite	2			
New key	Value	Description		

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets ▼
<input type="checkbox"/> Content-Type	application/json				
<input type="checkbox"/> Accept	application/json				
<input checked="" type="checkbox"/> Accept	application/xml				
<input type="checkbox"/> Content-Type	xml				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 85 ms Size: 348 B

Pretty Raw Preview XML ▼

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <usuarios>
3   <usuario rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/amigos/2"/>
4   <usuario rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/amigos/3"/>
5 </usuarios>

```

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/1/amigos?limite=2 Params Send ▼ Save ▼

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> limite	2			
New key	Value	Description		

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets ▼
<input type="checkbox"/> Content-Type	application/json				
<input checked="" type="checkbox"/> Accept	application/json				
<input type="checkbox"/> Accept	application/xml				
<input type="checkbox"/> Content-Type	xml				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 44 ms Size: 274 B

Pretty Raw Preview JSON ▼

```

1 {
2   "users": [
3     {
4       "rel": "self",
5       "url": "http://localhost:8080/RESTFUL/api/usuarios/1/amigos/2"
6     },
7     {
8       "rel": "self",
9       "url": "http://localhost:8080/RESTFUL/api/usuarios/1/amigos/3"
10    }
11  ]
12 }

```

Figura 1: Muestra la lista de amigos del usuario con id=2, además filtra limitando los datos, en este caso muestra solo 2 amigos, ya que “limite” =2. **(Obtener una lista de todos Nuestros Amigos, donde se puede limitar el número de amigos)**

Figura 2: Misma representación, pero usando JSON **(Obtener una lista de todos Nuestros Amigos, donde se puede limitar el número de amigos)**

GET Params

Key	Value	Description
<input checked="" type="checkbox"/> calificacion	5	
<input checked="" type="checkbox"/> categoria	Novela	
<input checked="" type="checkbox"/> autor	Charles	
New key	Value	Description

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

Key	Value	Description
<input type="checkbox"/> Content-Type	application/json	
<input type="checkbox"/> Accept	application/json	
<input checked="" type="checkbox"/> Accept	application/xml	
<input type="checkbox"/> Content-Type	xm	
New key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 130 ms Size: 357 B

Pretty Raw Preview XML

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <recomendaciones>
3   <recomendacion autor="Charles" calificaciones="10" categoria="Novela" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/amigos/recomendaciones/5"/>
4 </recomendaciones>

```

GET Params

Key	Value	Description
<input checked="" type="checkbox"/> calificacion	5	
<input checked="" type="checkbox"/> categoria	Novela	
<input checked="" type="checkbox"/> autor	Charles	
New key	Value	Description

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

Key	Value	Description
<input type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Accept	application/json	
<input type="checkbox"/> Accept	application/xml	
<input type="checkbox"/> Content-Type	xm	
New key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 72 ms Size: 280 B

Pretty Raw Preview JSON

```

1 {
2   "recomendacion": [
3     {
4       "autor": "Charles",
5       "calificaciones": 10,
6       "categoria": "Novela",
7       "rel": "self",
8       "url": "http://localhost:8080/RESTFUL/api/usuarios/1/amigos/recomendaciones/5"
9     }
10  ]
11 }

```

Figura 1: Muestra los libros recomendados por nuestros amigos, que se pueden filtrar por **calificación**, en este caso libros con **calificación** superior a 5, **autor**= "Charles" y con **categoría**= "novela". (Buscar en libros recomendados por nuestros amigos)

Figura 2: Misma representación, pero usando JSON (Buscar en libros recomendados por nuestros amigos.)

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/1/movil Params Send ▼ Save ▼

Key	Value	Description	...	Bulk Edit
New key	Value	Description		

Authorization Headers (4) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets
<input type="checkbox"/> Content-Type	application/json				
<input type="checkbox"/> Accept	application/json				
<input checked="" type="checkbox"/> Accept	application/xml				
<input type="checkbox"/> Content-Type	xml				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 83 ms Size: 593 B

Pretty Raw Preview XML ▼ ↺

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <movil>
3   <amigos>2</amigos>
4   <LibrosAmigos idAmigo="2" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/2/lectura/4"/>
5   <LibrosAmigos idAmigo="3" rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/3/lectura/6"/>
6   <ultimoLibro rel="self" href="http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"/>
7   <user id="1">
8     <correo>pepe@gmail.com</correo>
9     <edad>21</edad>
10    <name>Pepe</name>
11    <userName>Pepe99</userName>
12  </user>
13 </movil>

```

GET ▼ http://localhost:8080/RESTFUL/api/usuarios/1/movil Params Send ▼ Save ▼

Key	Value	Description	...	Bulk Edit	Presets
<input type="checkbox"/> Content-Type	application/json				
<input checked="" type="checkbox"/> Accept	application/json				
<input type="checkbox"/> Accept	application/xml				
<input type="checkbox"/> Content-Type	xml				
New key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 149 ms Size: 498 B

Pretty Raw Preview JSON ▼ ↺

```

1 {
2   "amigos": 2,
3   "listaLectura": [
4     {
5       "idAmigo": 2,
6       "rel": "self",
7       "url": "http://localhost:8080/RESTFUL/api/usuarios/2/lectura/4"
8     },
9     {
10      "idAmigo": 3,
11      "rel": "self",
12      "url": "http://localhost:8080/RESTFUL/api/usuarios/3/lectura/6"
13    }
14  ],
15  "ultimoLibro": [
16    {
17      "rel": "self",
18      "url": "http://localhost:8080/RESTFUL/api/usuarios/1/lectura/2"
19    }
20  ],
21  "user": {
22    "ID": 1,
23    "correo": "pepe@gmail.com",
24    "edad": 21,
25    "name": "Pepe",
26    "userName": "Pepe99"
27  }
28 }

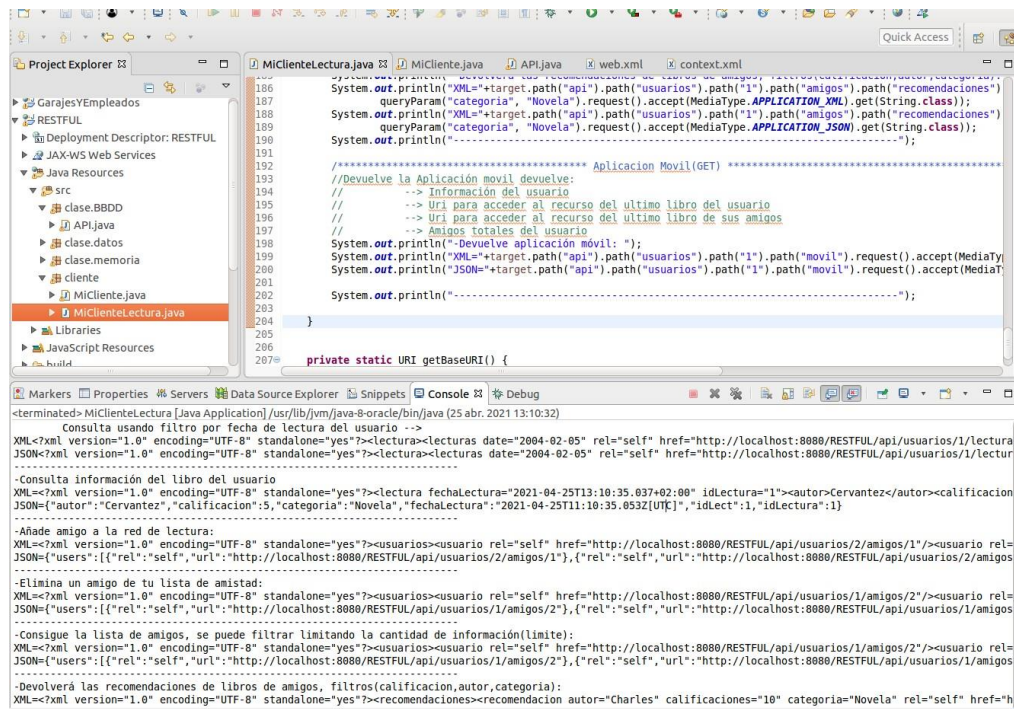
```

Figura 1: Muestra los datos básicos del usuario con id= 1, el número de amigos. Y también el último libro leído por el usuario y sus amigos, devolviendo URIS totalmente navegables para acceder a la información de esos últimos libros. **(Aplicación Móvil)**

Figura 2: Misma representación, pero usando JSON **(Aplicación Móvil)**

3. Capturas de la ejecución de las operaciones anteriores en el cliente.

NOTA: Es muy difícil realizar capturas de pantalla del cliente (al menos el cliente que se nos ha enseñado en clase) debido a que las tiras de XML y JSON son enormes. Así que se me ha ocurrido adjuntarle un documento **CLIENTE.txt** que contiene la ejecución entera del cliente donde se explica las operaciones que se realizan y el resultado tanto en XML como en JSON. Además, por cada operación que se realice como deletes o puts, he realizado gets para que se compruebe que las operaciones se realizan con éxito.



The screenshot shows an IDE with a project named 'GarajesYEmpleados'. The 'Project Explorer' on the left shows a package structure with 'src' containing 'clase.BBDD', 'API.java', 'clase.datos', 'clase.memoria', 'cliente', and 'MiClienteLectura.java'. The 'MiClienteLectura.java' file is open in the editor, showing Java code that interacts with a REST API. The code includes comments in Spanish and uses `System.out.println` for logging. The 'Console' at the bottom displays the output of the program, which is a mix of XML and JSON responses from the API. The output includes headers like 'Content-Type: application/xml' and 'Content-Type: application/json', and body content representing user data, book information, and API responses for various operations like 'Consulta usando filtro por fecha de lectura del usuario' and 'Consulta información del libro del usuario'.

Figura 1: Ejecución del cliente

4. Datos de la entrega

Se adjuntará en el .zip:

- Código (**RESTFUL**): También contendrá el **cliente**.
- Memoria (**MemoriaSOS.pdf**).
- En cuanto al usuario y password de la base de datos: como he trabajado como root solo se necesitará en cualquier caso la propia contraseña de la máquina virtual “restuser”.
- Y los **WAR** fuentes generados por ECLIPSE.
- Código SQL: tanto un **sql.sql** como un **sql.txt**. que contiene la creación de la base de datos y también INSERTS que son necesarios para la ejecución de la práctica.

5. Comentario final

Quiero recalcar algunas cosas de la práctica, y es que el cliente tiene alguna prueba distinta a las que he realizado en las capturas del postman, espero que no sea un inconveniente.

También decir, que en las lecturas de los libros se me ha colado introducir en la base de datos una columna para el nombre de los libros, al percatarme tarde he preferido no realizarlo ya que tendría que realizar varios cambios, no del código, pero si de la memoria y volver a realizar todas las capturas.

Además, he realizado algunas operaciones extras que yo creía, como por ejemplo obtener la información de un amigo o devolver la información de los libros recomendados, no están documentados ya que no sabía si era necesario, pero se hacen uso de ellos.

Asimismo, soy consciente de que quizás es necesario más ejemplos en la memoria, pero al haberlo realizado tanto en XML como en JSON ha hecho que me consumiera bastante tiempo, sobre todo por el cliente.