

FULLSTACK I: Microservicios

Leonel Briones Palacios

Profesor: Ignacio Cuturrufo

Índice

Características.....	4
Requisitos previos.....	4
Instalación.....	5
Base URL.....	5
Ejecución con Docker.....	7
Requisitos previos.....	7
Notas importantes.....	8
Endpoints.....	8
Endpoints disponibles.....	8
Estructura del proyecto.....	9
Dependencias principales.....	10
Excepciones personalizadas.....	11
Pruebas de Funcionalidad.....	12
Testing.....	17
Anexos.....	17

API REST: User

Este proyecto es una API RESTful desarrollada con **Spring Boot** que permite la gestión de usuarios. Incluye operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y utiliza el patrón DTO para la transferencia de datos entre las capas de la aplicación.

URL: https://github.com/jarodsmdev/API_REST_USER_EDUTECH

Características

- **CRUD de usuarios:** Crear, obtener, actualizar y eliminar usuarios.
- **Validación de datos:** Validación de entradas mediante anotaciones de `jakarta.validation`.
- **Patrón DTO:** Uso de `UserDto` para transferir datos entre la capa de servicio y el controlador.
- **Mapeo automático:** Implementación de `MapStruct` para mapear entre entidades y DTOs.
- **Manejo de excepciones:** Gestión de errores como `DuplicateKeyException` y `UserNotFoundException`.
- **Base de datos:** Persistencia de datos utilizando JPA con una base de datos relacional.

Requisitos previos

- **Java:** Versión 17 o superior.
- **Maven:** Versión 3.8 o superior.
- **Base de datos:** Configurada en el archivo `application.properties`. Por defecto viene la configuración para MySQL. (Puedes realizar los cambios a otro motor de bases de datos si así lo prefieres).

Instalación

1. Clona este repositorio:

```
git clone https://github.com/jarodsmdev/API_REST_USER_EDUTECH.git
```

2. Configura la base de datos en el archivo

`src/main/resources/application.properties` si deseas usar una base de datos diferente a MySQL.

- Configura las variables del entorno indicadas en este archivo.
 - `${DB_ENDPOINT}` : Dirección del host donde se encuentra la base de datos (por ejemplo, IP o nombre del servidor).
 - `${DB_PORT}` : Puerto en el que escucha la base de datos (por defecto 3306 para MySQL).
 - `${DB_NAME}` : Nombre de la base de datos a la que se va a conectar la aplicación.
 - `${DB_USERNAME}` : Usuario con el que se autentica la conexión a la base de datos.
 - `${DB_PASSWORD}` : Contraseña del usuario para acceder a la base de datos.

3. Compila el proyecto con Maven:

```
mvn clean install
```

4. Ejecuta la aplicación:

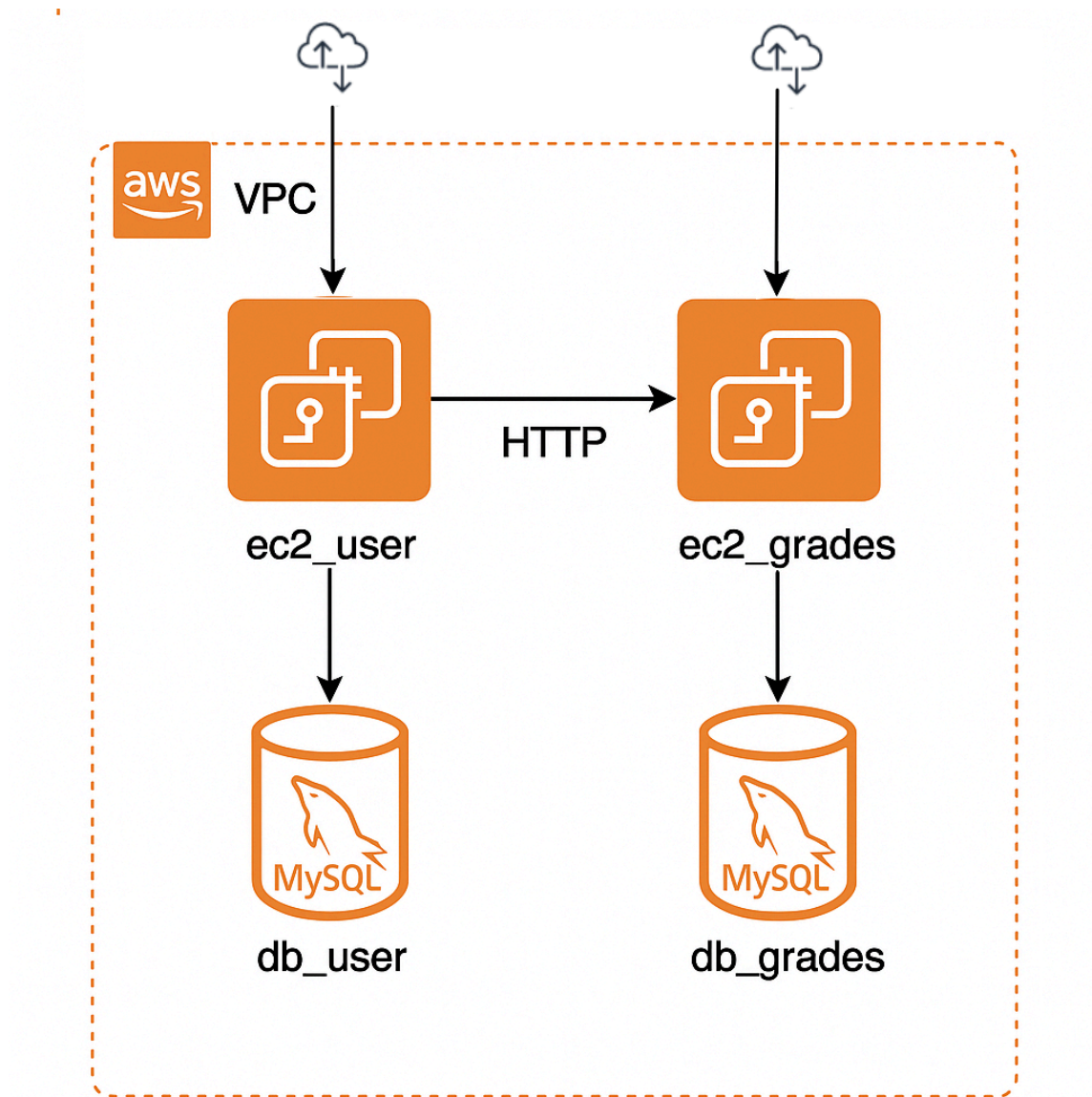
Debes configurar tu conexión a la base de datos en tu archivo `application.properties` ya que su configuración actual está realizada para ser utilizada con **Docker compose** (se indica más adelante), es sólo reemplazar las variables indicadas en el punto anterior.

```
mvn spring-boot:run
```

Base URL

```
http://localhost:8081/api/v1/users
```

Diagrama de despliegue



Ejecución con Docker

Este proyecto incluye un `Dockerfile` y un archivo `docker-compose.yml` para facilitar la construcción y ejecución de la aplicación en un contenedor Docker.

Requisitos previos

- **Docker:** Asegúrate de tener Docker instalado en tu sistema. [Guía de instalación de Docker](#)
- **Docker Compose:** Asegúrate de tener Docker Compose instalado. [Guía de instalación de Docker Compose](#)
- **Archivo .env :** Es necesario crear un archivo `.env` en la raíz del proyecto con las siguientes variables de entorno configuradas:

```
DB_ENDPOINT=<endpoint_de_tu_base_de_datos>
DB_PORT=<puerto_de_tu_base_de_datos>
DB_NAME=<nombre_de_tu_base_de_datos>
DB_USERNAME=<usuario_de_tu_base_de_datos>
DB_PASSWORD=<contraseña_de_tu_base_de_datos>
```

Construcción de la imagen Docker y ejecución

1. Abre una terminal y navega hasta la raíz del proyecto.
2. Ejecuta el siguiente comando para construir la imagen Docker:

```
docker compose up -d --build
```

3. Una vez que la imagen se haya construido y el contenedor esté en ejecución, puedes acceder a la API en la siguiente URL:

```
http://localhost/api/v1/users
```

4. Para detener y eliminar el contenedor, ejecuta:

```
docker compose down
```

Esto detendrá y eliminará el contenedor, pero no eliminará la imagen.

Notas importantes

El archivo `.env` no se incluye en el repositorio por razones de seguridad. Asegúrate de crearlo y configurarlo correctamente antes de ejecutar los contenedores.

El contenedor de la aplicación utiliza las variables de entorno definidas en el archivo `.env` para conectarse a la base de datos

Endpoints

Endpoints disponibles

Método	Endpoint	Descripción	Entrada JSON
GET	/	Obtiene todos los usuarios.	N/A
GET	/{"uuid"}	Obtiene un usuario por su ID.	N/A
POST	/	Crea un nuevo usuario.	Ver Ejemplo
PUT	/	Actualiza un usuario existente.	Ver Ejemplo
DELETE	/{"uuid"}	Elimina un usuario por su ID.	N/A

Ejemplo de JSON de entrada/salida

Crear o actualizar un usuario

```
{
  "userId": "b29052e1-2df8-4b2d-aa2f-6dd5cbac3c64",
  "firstName": "Susana",
  "lastName": "Oria",
  "birthDate": "1991-07-22",
  "email": "susana.oria@example.com",
  "phone": "+987654321",
  "address": "Calle Ficticia 321, Ciudad",
  "active": false,
  "rol": "ROLE_ADMIN"
}
```

Estructura del proyecto

```
src/main/java/com/briones/users/management
├─ controller    # Controladores REST
├─ exception     # Clases de manejo de excepciones
├─ model         # Entidades JPA y DTOs
│   └─ dto       # Clases DTO
├─ repository    # Repositorios JPA
├─ service       # Lógica de negocio
└─ UserManagementApiApplication.java # Clase principal
```


Dependencias principales

- **Spring Boot Starter Web:** Para construir la API REST.
- **Spring Boot Starter Data JPA:** Para la persistencia de datos.
- **Spring Boot Starter Validation:** Para la validación de entrada de datos.
- **MySQL Database:** Base de datos MySQL para desarrollo y pruebas.
- **MapStruct:** Para el mapeo entre entidades y DTOs.
- **Lombok:** Para reducir el código boilerplate.
- **JaCoCo:** Para generar reportes de cobertura de código.
- **Swagger UI:** Para documentar y probar los endpoints de la API de manera interactiva.

Una vez que la aplicación esté en ejecución, puedes acceder a la interfaz de Swagger UI en la siguiente URL:

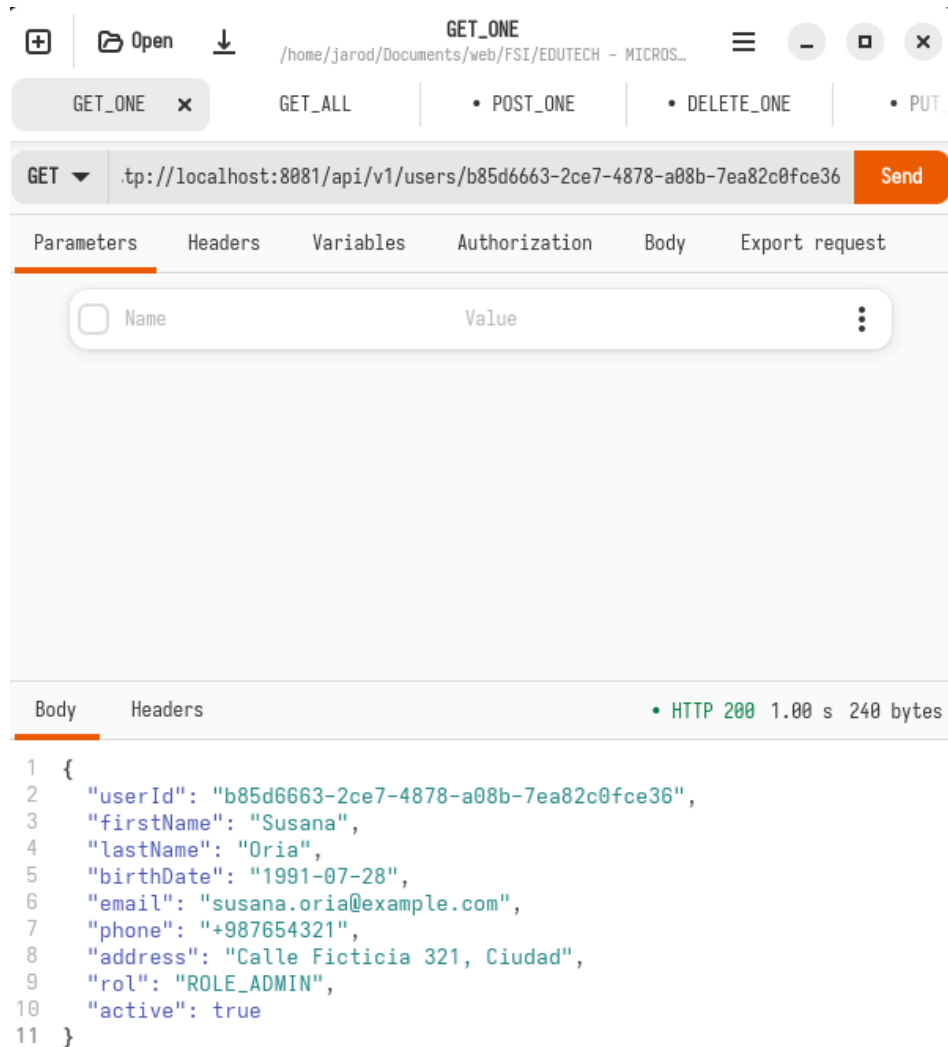
`http://localhost:8081/swagger-ui/index.html`

Excepciones personalizadas

- **DuplicateKeyException:** Se lanza cuando se intenta crear o actualizar un usuario con un correo electrónico ya existente.
- **UserNotFoundException:** Se lanza cuando no se encuentra un usuario con el ID o correo proporcionado.

Pruebas de Funcionalidad

• GET : Por ID de usuario



The screenshot shows a REST client interface with a GET request to `http://localhost:8081/api/v1/users/b85d6663-2ce7-4878-a08b-7ea82c0fce36`. The response is a JSON object representing a user.

Request:

- Method: GET
- URL: `http://localhost:8081/api/v1/users/b85d6663-2ce7-4878-a08b-7ea82c0fce36`

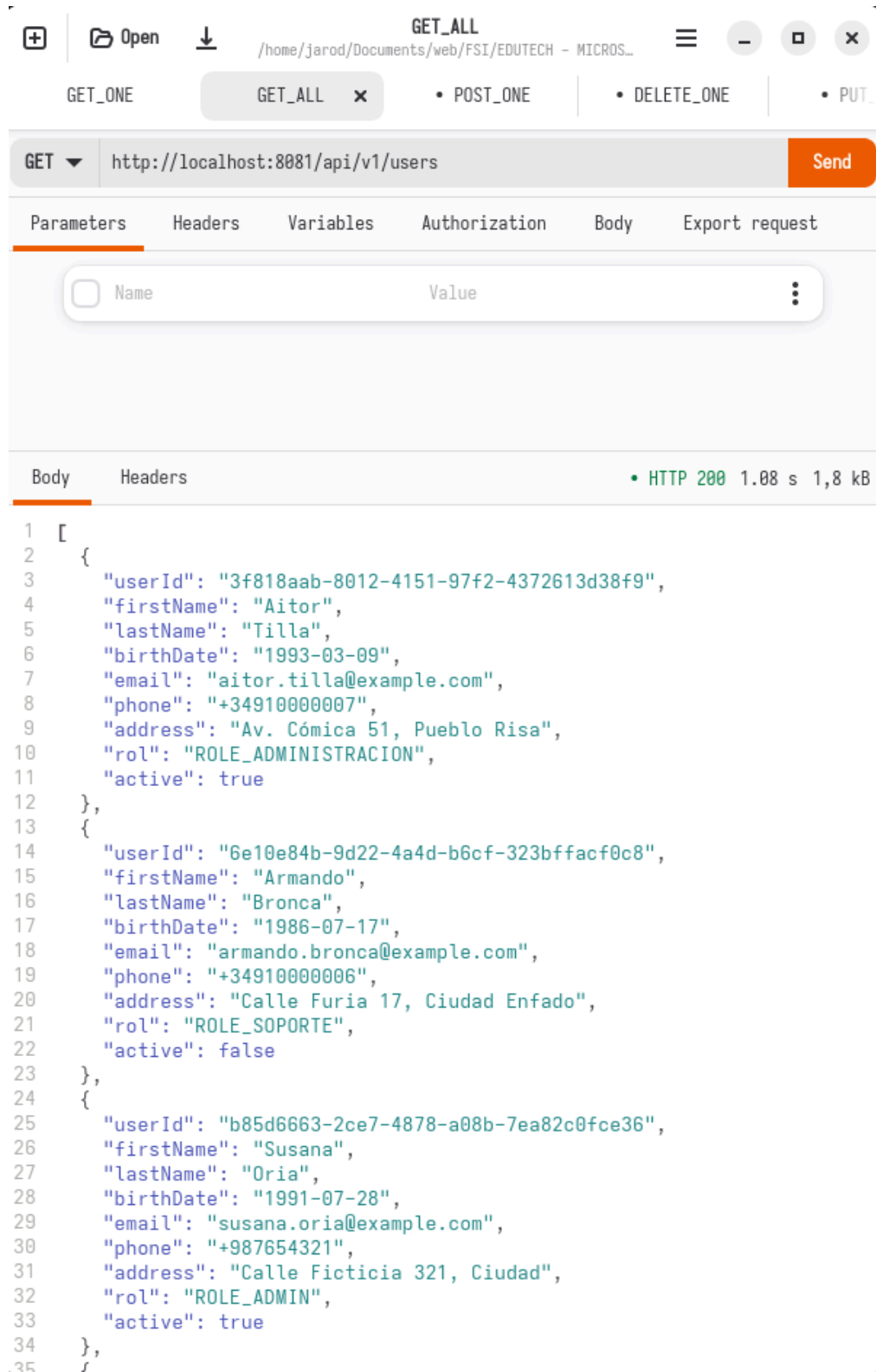
Response:

```

1 {
2   "userId": "b85d6663-2ce7-4878-a08b-7ea82c0fce36",
3   "firstName": "Susana",
4   "lastName": "Oria",
5   "birthDate": "1991-07-28",
6   "email": "susana.oria@example.com",
7   "phone": "+987654321",
8   "address": "Calle Ficticia 321, Ciudad",
9   "rol": "ROLE_ADMIN",
10  "active": true
11 }
```

• HTTP 200 1.00 s 240 bytes

- **GET ALL** : Listar todos los usuarios



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8081/api/v1/users
- Response Status:** HTTP 200, 1.08 s, 1,8 kB
- Response Body (JSON):**

```

1  [
2  {
3    "userId": "3f818aab-8012-4151-97f2-4372613d38f9",
4    "firstName": "Aitor",
5    "lastName": "Tilla",
6    "birthDate": "1993-03-09",
7    "email": "aitor.tilla@example.com",
8    "phone": "+34910000007",
9    "address": "Av. Cómica 51, Pueblo Risa",
10   "rol": "ROLE_ADMINISTRACION",
11   "active": true
12 },
13 {
14   "userId": "6e10e84b-9d22-4a4d-b6cf-323bfffac0c8",
15   "firstName": "Armando",
16   "lastName": "Bronca",
17   "birthDate": "1986-07-17",
18   "email": "armando.bronca@example.com",
19   "phone": "+34910000006",
20   "address": "Calle Furia 17, Ciudad Enfado",
21   "rol": "ROLE_SOPORTE",
22   "active": false
23 },
24 {
25   "userId": "b85d6663-2ce7-4878-a08b-7ea82c0fce36",
26   "firstName": "Susana",
27   "lastName": "Oria",
28   "birthDate": "1991-07-28",
29   "email": "susana.oria@example.com",
30   "phone": "+987654321",
31   "address": "Calle Ficticia 321, Ciudad",
32   "rol": "ROLE_ADMIN",
33   "active": true
34 },
35 ]

```

• **POST : Guardar usuario**

The screenshot shows a REST client interface with a POST request to `http://localhost:8081/api/v1/users`. The request body is a JSON object representing a user. The response is an HTTP 201 status with a JSON body containing the created user's details, including a unique `userId`.

Request:

```

1 {
2   "firstName": "Anacleto",
3   "lastName": "Mojado",
4   "birthDate": "1988-03-03",
5   "email": "anacleto.mojado@example.com",
6   "phone": "+34910000015",
7   "address": "Camino Empapado 2, Villa Chapuzón",
8   "rol": "ROLE_PROFESOR",
9   "active": true,
10  "password": "123456789A!"
11 }

```

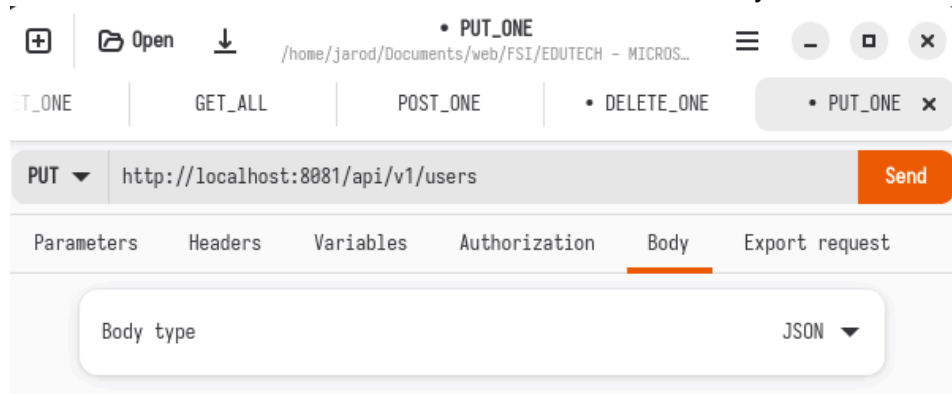
Response:

```

1 {
2   "userId": "ebf21f1f-0105-4cd0-9543-8be874e251d1",
3   "firstName": "Anacleto",
4   "lastName": "Mojado",
5   "birthDate": "1988-03-03",
6   "email": "anacleto.mojado@example.com",
7   "phone": "+34910000015",
8   "address": "Camino Empapado 2, Villa Chapuzón",
9   "password": "123456789A!",
10  "rol": "ROLE_PROFESOR",
11  "active": true
12 }

```

- **PUT** : Actualiza un usuario enviándole JSON incluyendo su ID

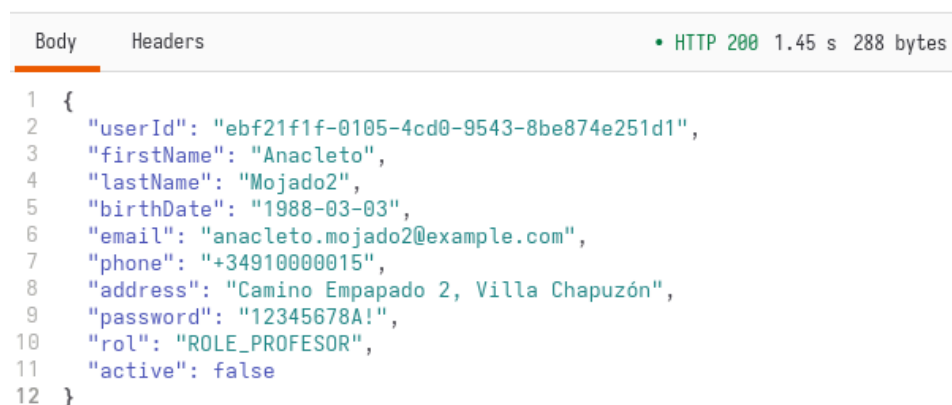


The screenshot shows a REST client interface with a tab labeled "PUT_ONE". The URL bar displays "http://localhost:8081/api/v1/users" with a "Send" button to the right. Below the URL bar, there are tabs for "Parameters", "Headers", "Variables", "Authorization", "Body", and "Export request". The "Body" tab is selected, and a dropdown menu shows "Body type" set to "JSON".

```

1 {
2   "userId": "ebf21f1f-0105-4cd0-9543-8be874e251d1",
3   "firstName": "Anacleto",
4   "lastName": "Mojado2",
5   "birthDate": "1988-03-03",
6   "email": "anacleto.mojado2@example.com",
7   "phone": "+34910000015",
8   "address": "Camino Empapado 2, Villa Chapuzón",
9   "password": "12345678A!",
10  "rol": "ROLE_PROFESOR",
11  "active": false
12 }

```



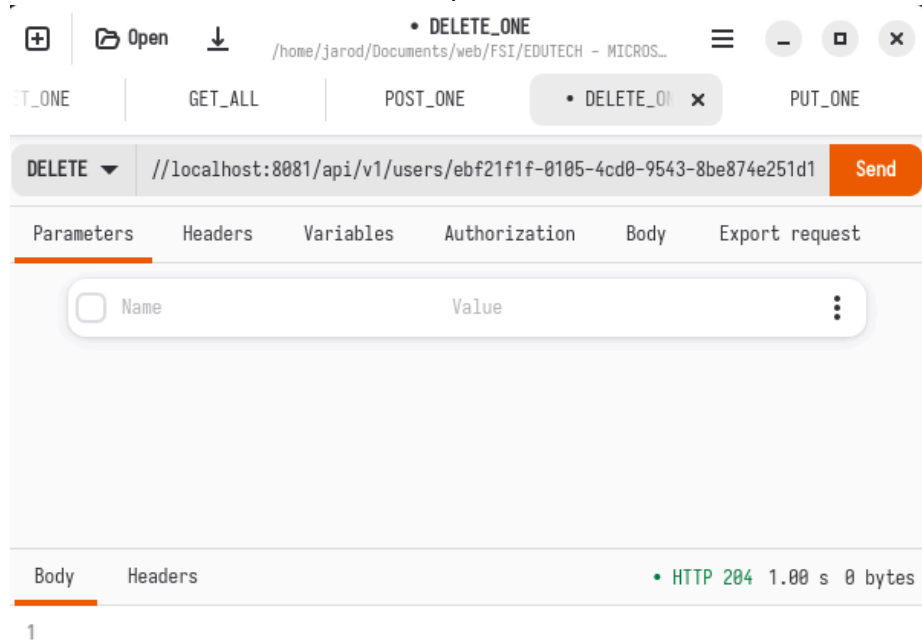
The screenshot shows the response of the PUT request. The "Body" tab is selected, and the response is displayed as JSON. The status bar at the top right indicates "HTTP 200 1.45 s 288 bytes".

```

1 {
2   "userId": "ebf21f1f-0105-4cd0-9543-8be874e251d1",
3   "firstName": "Anacleto",
4   "lastName": "Mojado2",
5   "birthDate": "1988-03-03",
6   "email": "anacleto.mojado2@example.com",
7   "phone": "+34910000015",
8   "address": "Camino Empapado 2, Villa Chapuzón",
9   "password": "12345678A!",
10  "rol": "ROLE_PROFESOR",
11  "active": false
12 }

```

- **DELETE** : Elimina usuario por su ID



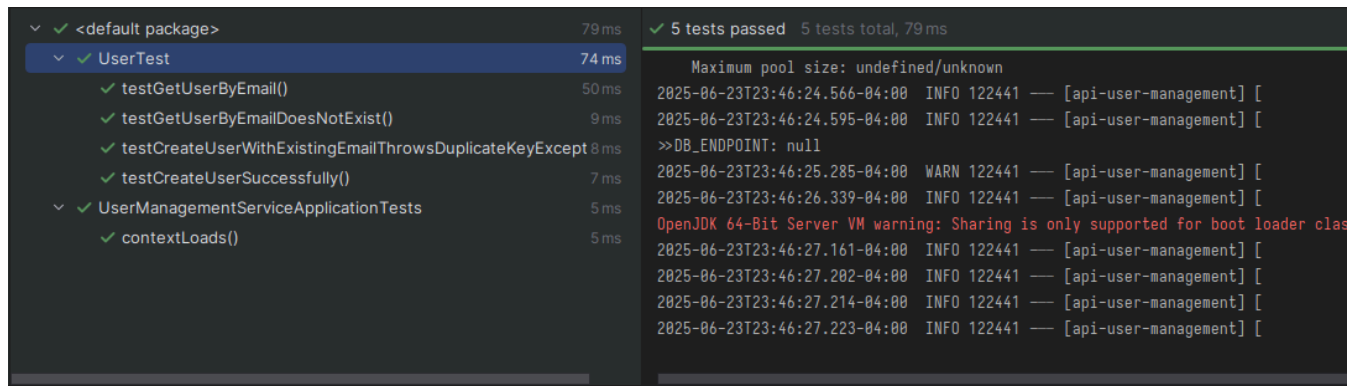
The screenshot shows a REST client interface with a tab titled "DELETE_ONE". The active tab is "DELETE", and the URL is "localhost:8081/api/v1/users/ebf21f1f-0105-4cd0-9543-8be874e251d1". The "Parameters" tab is selected, showing a table with columns "Name" and "Value". The "Body" tab is also visible, showing the response "HTTP 204 1.00 s 0 bytes".

Name	Value
------	-------

Body Headers

• HTTP 204 1.00 s 0 bytes

Testing



```

✓ <default package> 79 ms
  ✓ UserTest 74 ms
    ✓ testGetUserByEmail() 50 ms
    ✓ testGetUserByEmailDoesNotExist() 9 ms
    ✓ testCreateUserWithExistingEmailThrowsDuplicateKeyExcept 8 ms
    ✓ testCreateUserSuccessfully() 7 ms
  ✓ UserManagementServiceApplicationTests 5 ms
    ✓ contextLoads() 5 ms

✓ 5 tests passed 5 tests total, 79 ms

Maximum pool size: undefined/unknown
2025-06-23T23:46:24.566-04:00 INFO 122441 — [api-user-management] [
2025-06-23T23:46:24.595-04:00 INFO 122441 — [api-user-management] [
>>DB_ENDPOINT: null
2025-06-23T23:46:25.285-04:00 WARN 122441 — [api-user-management] [
2025-06-23T23:46:26.339-04:00 INFO 122441 — [api-user-management] [
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader clas
2025-06-23T23:46:27.161-04:00 INFO 122441 — [api-user-management] [
2025-06-23T23:46:27.202-04:00 INFO 122441 — [api-user-management] [
2025-06-23T23:46:27.214-04:00 INFO 122441 — [api-user-management] [
2025-06-23T23:46:27.223-04:00 INFO 122441 — [api-user-management] [
  
```

Anexos

Diagrama de Clases

