

# Manual de Usuario Prueba Técnica

## 1. Resumen

Este esquema de directorios y archivos refleja una estructura típica de proyecto Java con Spring Boot, organizada por capas (controladores, servicios, repositorios) y componentes específicos para la autenticación JWT, gestión de usuarios y seguridad, junto a un servicio relacionado a generación de Tickets, el cual nos podrá simular la compra de un objeto en específico. Cada directorio y archivo juega un papel crucial en la arquitectura y funcionalidad general del Proyecto

## 2. Tecnologías

- Java 11(SpringBoot) - Maven
- MySql
- Docker
- Tomcat

## 3. Diseño Arquitectura

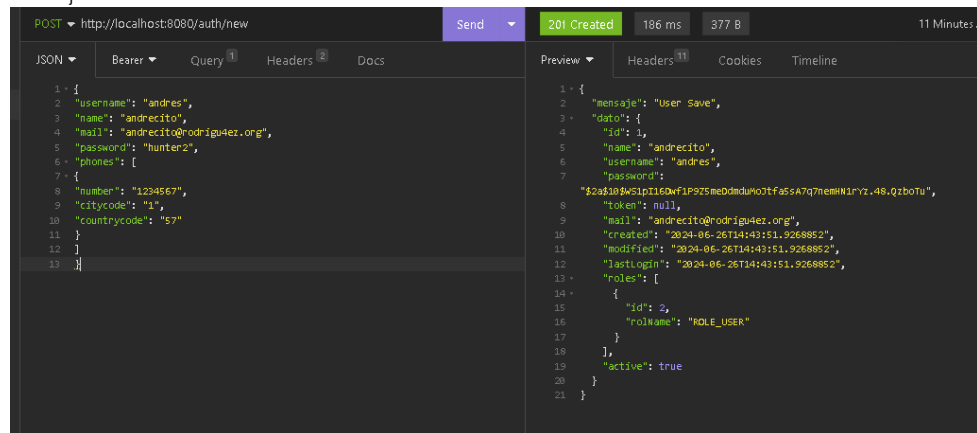


## 1. EndPoints User(Security)

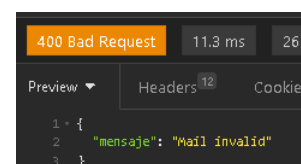
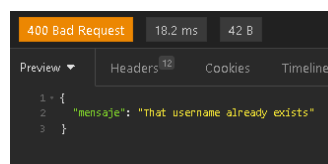
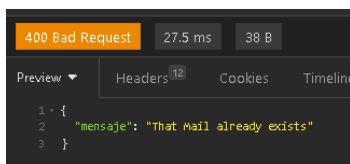
1. New User (PostMethod)

- <http://localhost:8080/auth/new>
- No necesita token de autentificación
- Crear nuevo usuario
- Json:

```
{
  "username": "juan123",
  "name": "Juan Rodriguez",
  "mail": "juan@rodriguez.org",
  "password": "hunter2",
}
```



- Username duplicado tendrá su respectiva validación
- Mail duplicado tendrá su respectiva Validación
- Mail mal formado tendrá su respectiva Validación

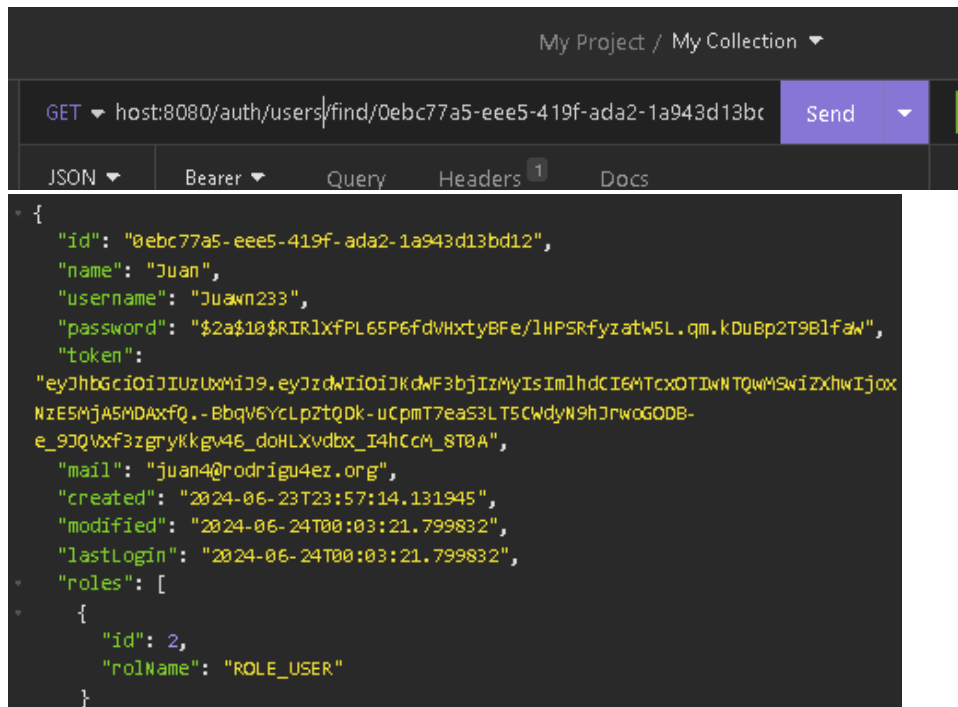


## 2. Login (PostMethod)

- <http://localhost:8080/auth/login>
- Acceder usando username y password, generará los datos de acceso y un token
- Json:

```
{
  "username": "Juawn233",
  "password": "hunter2"
}
```





My Project / My Collection ▾

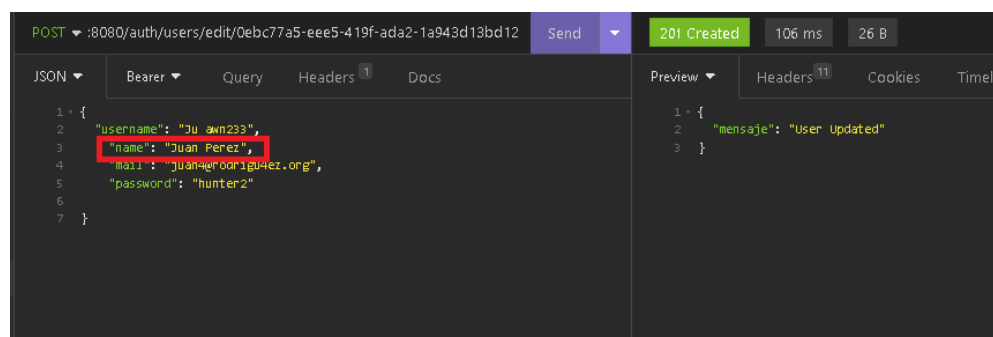
GET ▾ host:8080/auth/users/find/0ebc77a5-eee5-419f-ada2-1a943d13bd12 Send ▾

JSON ▾ Bearer ▾ Query Headers 1 Docs

```
{
  "id": "0ebc77a5-eee5-419f-ada2-1a943d13bd12",
  "name": "Juan",
  "username": "Juan233",
  "password": "$2a$10$RIR1xfPL65P6fdVHxtyBFe/1HPSRfyzatw5L.qm.kDuBp2T9B1faw",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWQ6IjE6MTcxOTIwNTQwMSwiZmxhbnIjOiBzE5MjA5MDAxZjQ6BbQv6YcLpztQDk-uCpMT7eaS3LT5CWdyN9hJrwoGODB-e_9JQVxf3zgrykkgw46_doHLXvdbx_I4hCcM_8T0A",
  "mail": "juan4@rodrigu4ez.org",
  "created": "2024-06-23T23:57:14.131945",
  "modified": "2024-06-24T00:03:21.799832",
  "lastLogin": "2024-06-24T00:03:21.799832",
  "roles": [
    {
      "id": 2,
      "rolName": "ROLE_USER"
    }
  ]
}
```

##### 5. UpdateById (PostMethod)

- <http://localhost:8080/auth/users/edit/{id}>
- Se deberá usar Bearer como método de acceso mediante token y registrar el token perteneciente a la sesión iniciada para poder así acceder y realizar la edición.
- Se editará usuario mediante el uso del PathVariable para búsqueda del registro, y se realizar modificaciones del objeto según sea la necesidad
- Datos básicos accesibles a modificaciones (name, username, password, mail), usando el método GetUsers, se podrá evidenciar el cambio en los datos.



POST ▾ :8080/auth/users/edit/0ebc77a5-eee5-419f-ada2-1a943d13bd12 Send ▾ 201 Created 106 ms 26 B

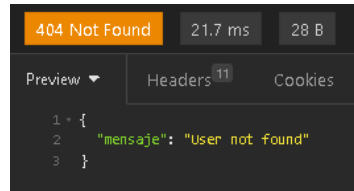
JSON ▾ Bearer ▾ Query Headers 1 Docs

```
1 {
2   "username": "Juan233",
3   "name": "Juan Perez",
4   "mail": "juan4@rodrigu4ez.org",
5   "password": "hunter2"
6 }
7 }
```

Preview ▾ Headers 11 Cookies Timeline

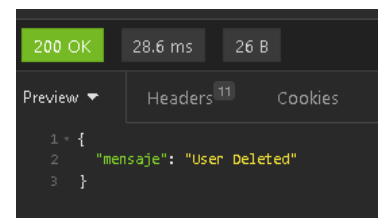
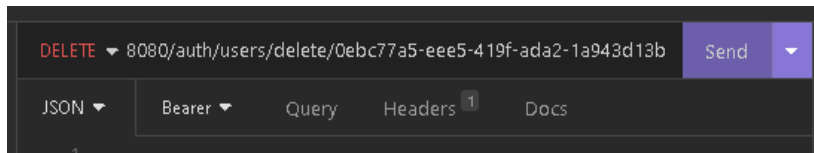
```
1 {
2   "mensaje": "User Updated"
3 }
```

- En caso de no encontrar el usuario, mostrará mensaje de User Not Found

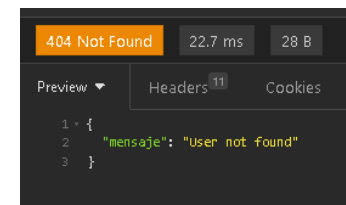


#### 6. DeleteUserById (DeleteMethod)

- <http://localhost:8080/auth/users/delete/{id}>
- Se deberá usar Bearer como método de acceso mediante token y registrar el token perteneciente a la sesión iniciada para poder así acceder y realizar la eliminación.
- Se realiza búsqueda de los datos del usuario a eliminar de forma lógica (Desactivacion) mediante id ()



- En caso de no encontrar el respectivo usuario a eliminar, se mostrará un mensaje de User Not Found.



#### 6. Endpoint Store(Product)

##### 1. CreateProduct (PostMethod)

- <http://localhost:8080/auth/product/new>
- Json:

```
{
  "name": "frutas",
  "description": "la sexta",
  "price": 50,
  "imageUrl": "http://example.com/product.jpg",
  "stock": true,
  "amount": 50
}
```

Stock: Manejara un estado de disponibilidad para hacer uso del producto durante la creación de un ticket

Amount: Cantidad disponible del producto en Stock.

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/auth/product/new`. The request body is a JSON object representing a product. The response is a 201 Created status with a response body containing a success message and the created product details.

```
POST http://localhost:8080/auth/product/new
{
  "name": "frutas",
  "description": "la sexta",
  "price": 50,
  "imageUrl": "http://example.com/product.jpg",
  "stock": true,
  "amount": 50
}
```

```
{
  "mensaje": "ProductModel Save",
  "dato": {
    "id": 2,
    "name": "frutas",
    "description": "la sexta",
    "price": 50.0,
    "category": null,
    "imageUrl": null,
    "stock": true,
    "amount": 50
  }
}
```

## 2. GetAllProducts: Obtener todos los productos mediante paginación: (GetMethod)

- <http://localhost:8080/auth/product/products?page=0&size=3>

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/auth/product/products?page=0&size=3`. The response is a 200 OK status with a response body containing a list of products and pagination information.

```
200 OK
{
  "content": [
    {
      "id": 1,
      "name": "carnes",
      "description": "la sexta",
      "price": 50.0,
      "category": null,
      "imageUrl": null,
      "stock": false,
      "amount": 50
    },
    {
      "id": 2,
      "name": "frutas",
      "description": "la sexta",
      "price": 50.0,
      "category": null,
      "imageUrl": null,
      "stock": false,
      "amount": 50
    }
  ],
  "pageable": {
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "offset": 0,
    "page": 0,
    "size": 3,
    "totalElements": 2
  }
}
```

## 3. EditProduct(PostMethod)

- <http://localhost:8080/auth/product/products/edit/{id}>

```
POST http://localhost:8080/auth/product/products/edit/1 201 Created 18.2 ms 156 B

JSON ▼ Bearer Query Headers Docs Preview Headers Cookies T
1 {
2   "id": 1,
3   "name": "frutas",
4   "description": "la sexta",
5   "price": 60.0,
6   "stock": true,
7   "amount": 50
8 }
9
10
```

#### 4. DeleteProductLogic(DeleteMethod)

- <http://localhost:8080/auth/product/products/delete?id=1&status=false>
- False para desactivar, true para activar

```
200 OK 48.6 ms 58 B

Preview ▼ Headers Cookies Timeline
1 {
2   "mensaje": "Product #:1Deleted(Desactivated)",
3   "dato": null
4 }
```

#### 5. FindById(PostMethod)

- <http://localhost:8080/auth/product/products/find/1>

```
200 OK 19.8 ms 119 B

Preview ▼ Headers Cookies
1 {
2   "id": 1,
3   "name": "frutas",
4   "description": "la sexta",
5   "price": 60.0,
6   "category": null,
7   "imageUrl": null,
8   "stock": true,
9   "amount": 50
10 }
```

#### 7. EndPoint Store(Tickets)

##### 1. CreateTicket (PostMethod)

- <http://localhost:8080/auth/ticket/new>
- Json:

```
{
  "amount": 50,
  "idUser": "1",
```

"idProduct": "2"

}

- Amount = Cantidad a “comprar-registrar” en el ticket ( Se descontara del stock del producto)
- idUser: identificador del usuario a elegir
- idProduct: identificador del producto a elegir
- Si el valor Amount, excede el stock del producto, generará un mensaje de “cantidad fuera del stock disponible”
- Si el producto no está disponible, generará mensaje de “producto no disponible”
- Si el producto o el usuario no existe, generará mensaje de “producto/usuario no disponible”

```
201 Created 37 ms 274 B
Preview Headers 11 Cookies Timeline
1 {
2   "mensaje": "Ticket Saved",
3   "dato": {
4     "id": 1,
5     "created": "2024-06-26T16:13:10.5683334",
6     "modified": "2024-06-26T16:13:10.5683334",
7     "amount": 10,
8     "price": 60.0,
9     "total": 600.0,
10    "idProduct": 0,
11    "idUser": 0,
12    "nameUser": "andrecito",
13    "nameProduct": "frutas",
14    "mensaje": null,
15    "isValid": 0,
16    "active": true
17  }
18 }
```

```
BeautifyJSON
409 Conflict 19.1 ms 257 B
Preview Headers 11 Cookies Timeline
1 {
2   "mensaje": "Required quantity out of stock",
3   "dato": {
4     "id": 0,
5     "created": null,
6     "modified": null,
```

```
400 Bad Request 27.7 ms 44 B
Preview Headers 12 Cookies Timeline
1 {
2   "mensaje": "Fail Saving Ticket",
3   "dato": null
4 }
```

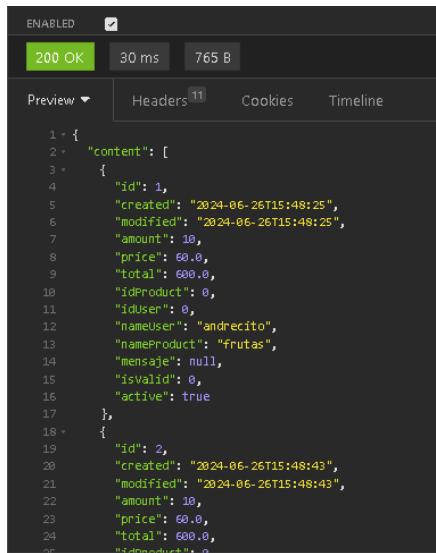
```
404 Not Found 18.3 ms 247 B
Preview Headers 11 Cookies Timeline
1 {
2   "mensaje": "User or Product not found",
3   "dato": {
4     "id": 0,
```

## 2. GetAllTickets(GetMethod)

- <http://localhost:8080/auth/ticket/tickets?page=0&size=10&status=all>
- <http://localhost:8080/auth/ticket/tickets?page=0&size=10&status=active>
- <http://localhost:8080/auth/ticket/tickets?page=0&size=10&status=desactivated>
- Se maneja un estado de disponibilidad en el cual según sea el estado que se requiera ver, mostrará los tickets



- Estados(All, active, desactivated)



ENABLED ☒

200 OK 30 ms 765 B

Preview Headers Cookies Timeline

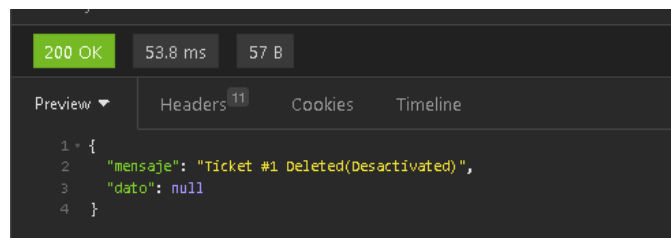
```

1 {
2   "content": [
3     {
4       "id": 1,
5       "created": "2024-06-26T15:48:25",
6       "modified": "2024-06-26T15:48:25",
7       "amount": 10,
8       "price": 60.0,
9       "total": 600.0,
10      "idProduct": 0,
11      "idUser": 0,
12      "nameUser": "andrecito",
13      "nameProduct": "frutas",
14      "mensaje": null,
15      "isValid": 0,
16      "active": true
17    },
18    {
19      "id": 2,
20      "created": "2024-06-26T15:48:43",
21      "modified": "2024-06-26T15:48:43",
22      "amount": 10,
23      "price": 60.0,
24      "total": 600.0,
25      "idProduct": 0,

```

## 2. DeleteLogicTicket(DeleteMethod)

- <http://localhost:8080/auth/ticket/delete?id=1&status=true>
- <http://localhost:8080/auth/ticket/delete?id=1&status=false>
- Se maneja un estado de disponibilidad en el cual según sea el estado que se requiera ver, mostrará los tickets
- False: Desactivara el ticket
- True: Activara el ticket



200 OK 53.8 ms 57 B

Preview Headers Cookies Timeline

```

1 {
2   "mensaje": "Ticket #1 Deleted(Desactivated)",
3   "dato": null
4 }

```

## 6. DockerFile

- Archivo de configuración Docker con el propósito de crear una imagen y poder ser desplegada en el servidor Tomcat (Configuracion de preferencia)

Dockerfile X

src > main > resources > Dockerfile > ...

```
1  # Usa la imagen oficial de Tomcat 9
2  FROM tomcat:9.0.90
3
4  # Elimina las aplicaciones de ejemplo de Tomcat
5  RUN rm -rf /usr/local/tomcat/webapps/*
6
7  # Copia el archivo WAR al directorio webapps de Tomcat
8  COPY target/demo-0.0.1-SNAPSHOT.war /usr/local/tomcat/webapps/demo.war
9
10 # Exponer el puerto 8080
11 EXPOSE 8080
12
13 # Comando para iniciar Tomcat
14 CMD ["catalina.sh", "run"]
15
```