Product Management System

Konstantinos Tsampiras

Introduction

This is a simple product management system written in Java using the Spring framework. Admin users can retrieve all the products, add new products, update existing products, and delete products from the database. Simple users can only see existing products and add new ones. The running app has 2 users who can execute commands after proper authorization.

Role	**ADMIN**	**USER**
Username	admin	user
Password	Admin	password

API Documentation

- `GET /api/v1/products`

Returns all the products from the database, requires authorization from either an admin or a user.

Using Postman, Select the **GET** action, enter the URL **http://localhost:8080/api/products/** and in the Authorization tab, on Type, select **Basic Auth** and enter the **user** and **password** for the user you want to use, and press **Send**.

A list with the products will appear, each product has a product ID, a name, a description, and a price.

- `POST /api/v1/products`

Adds a new product to the database, requires authorization from either an admin or a user.

Using Postman, Select the **POST** action, enter the URL **http://localhost:8080/api/products/** and in the Authorization tab, on Type, select **Basic Auth** and enter the **user** and **password** for the user you want to use, in the Body tab, select the raw formatting and select the JSON format, paste the snippet below, and press **Send**.

The product will be returned with a new product ID.

```
{
    "name": "Laptop",
    "description": "A Laptop",
    "price": 999.99
}
```

- `PUT /api/v1/products/`*`{product ID}`*

Updates the product with the specified product ID, requires authorization from an admin.

Using Postman, Select the **PUT** action, enter the URL **http://localhost:8080/api/products/1**, in the Authorization tab, on Type, select **Basic Auth** and enter the **user** and **password** for the user you want to use, in the Body tab, select the raw formatting and select the JSON format, paste the snippet below, and press **Send**.

```
{
    "name": "Peach",
    "price": 1.99
}
```

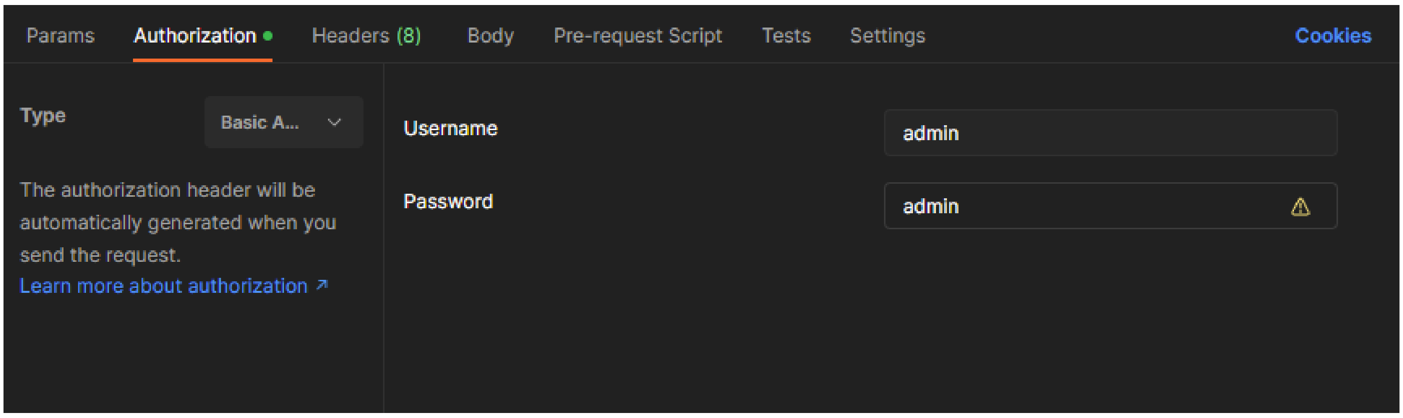
➢ `DELETE /api/v1/products/{product ID}`

Deletes the product with the specified product ID, requires authorization from an admin.

Using Postman, Select the **DELETE** action, enter the URL **http://localhost:8080/api/products/1**, in the Authorization tab, on Type, select **Basic Auth** and enter the **user** and **password** for the user you want to use, and press **Send**.

Running the commands

As an admin



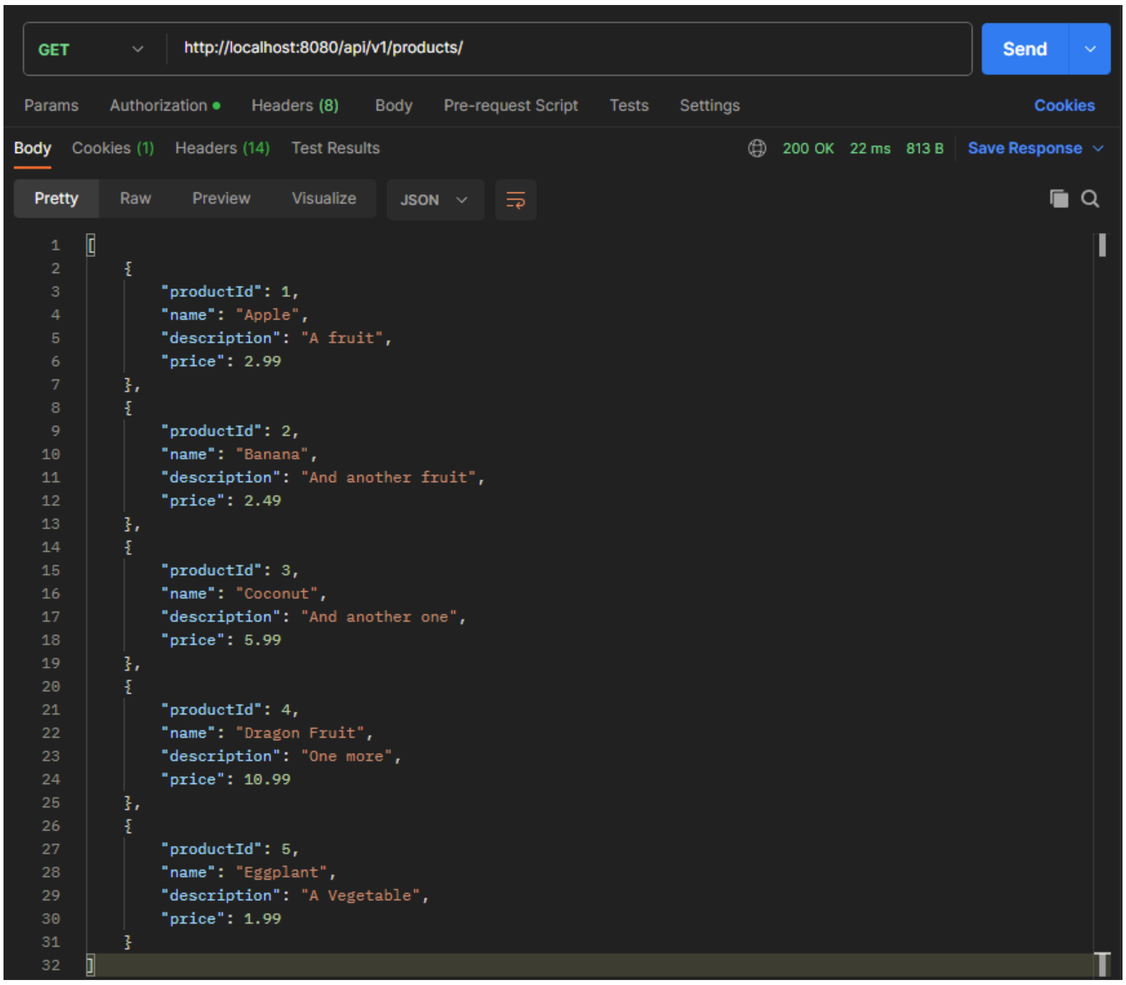
Type: Basic A...

The authorization header will be automatically generated when you send the request.

Username: admin

Password: admin

`GET /api/v1/products`

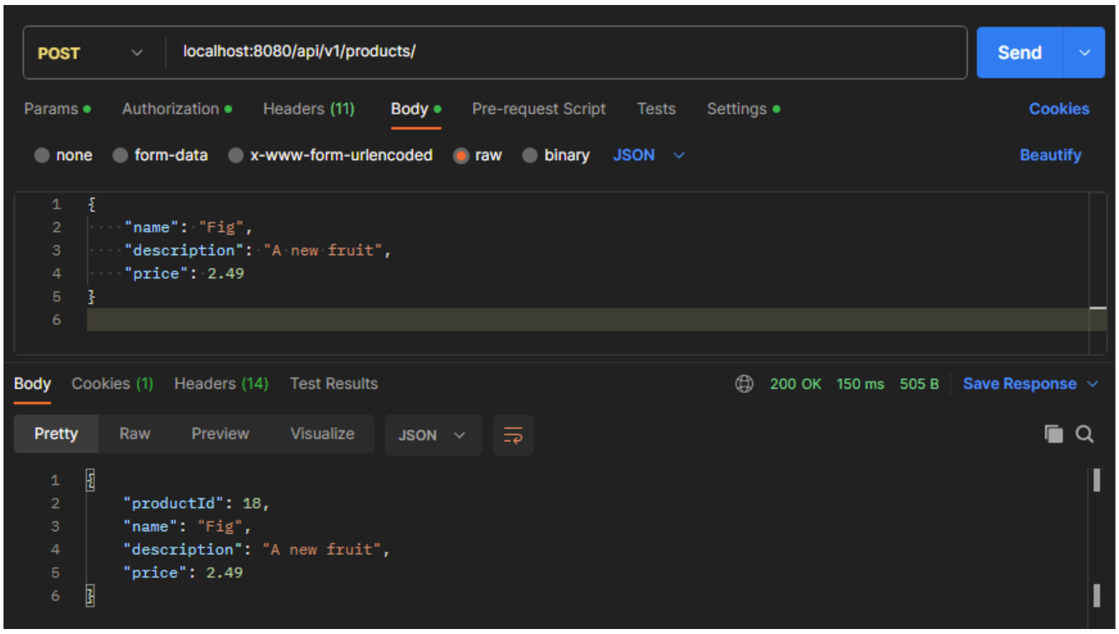


GET http://localhost:8080/api/v1/products/

200 OK 22 ms 813 B

```
[
    {
        "productId": 1,
        "name": "Apple",
        "description": "A fruit",
        "price": 2.99
    },
    {
        "productId": 2,
        "name": "Banana",
        "description": "And another fruit",
        "price": 2.49
    },
    {
        "productId": 3,
        "name": "Coconut",
        "description": "And another one",
        "price": 5.99
    },
    {
        "productId": 4,
        "name": "Dragon Fruit",
        "description": "One more",
        "price": 10.99
    },
    {
        "productId": 5,
        "name": "Eggplant",
        "description": "A Vegetable",
        "price": 1.99
    }
]
```

```
POST /api/v1/products
```



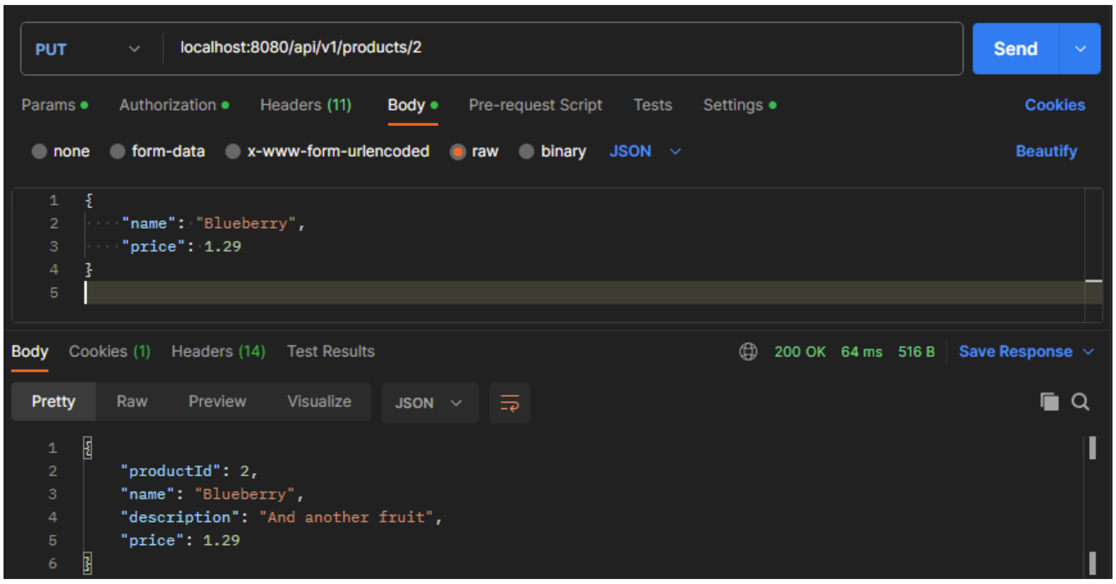
POST localhost:8080/api/v1/products/

```
{
    "name": "Fig",
    "description": "A new fruit",
    "price": 2.49
}
```

200 OK 150 ms 505 B

```
{
    "productId": 18,
    "name": "Fig",
    "description": "A new fruit",
    "price": 2.49
}
```

```
PUT /api/v1/products/{productid}
```



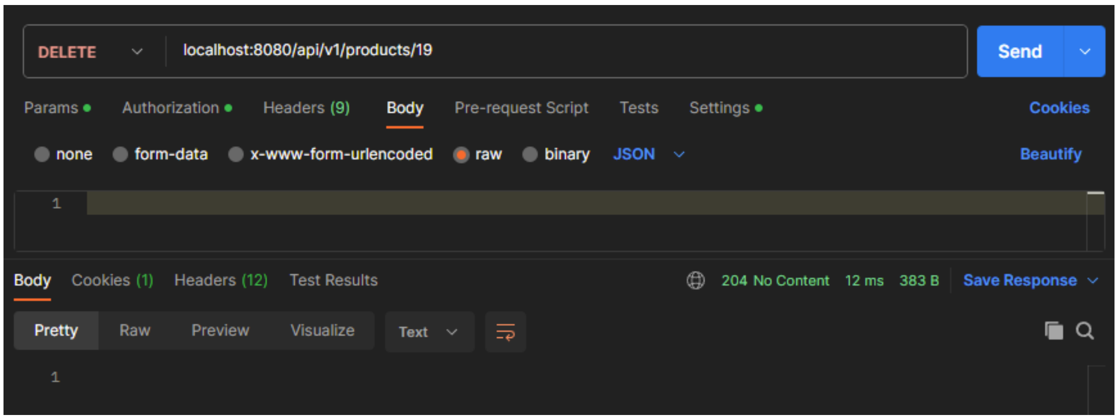
PUT localhost:8080/api/v1/products/2

```
{
    "name": "Blueberry",
    "price": 1.29
}
```

200 OK 64 ms 516 B

```
{
    "productId": 2,
    "name": "Blueberry",
    "description": "And another fruit",
    "price": 1.29
}
```

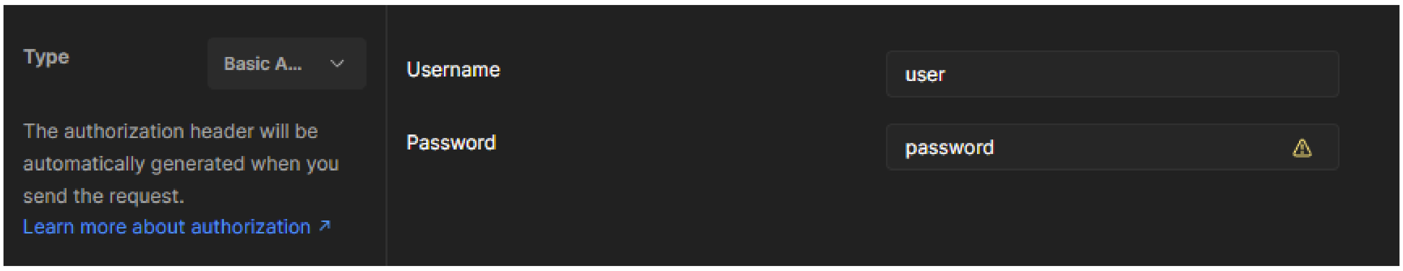
```
DELETE /api/v1/products/{product ID}
```



DELETE localhost:8080/api/v1/products/19

204 No Content 12 ms 383 B

As a user



Type: Basic A...

The authorization header will be automatically generated when you send the request.

Username: user

Password: password

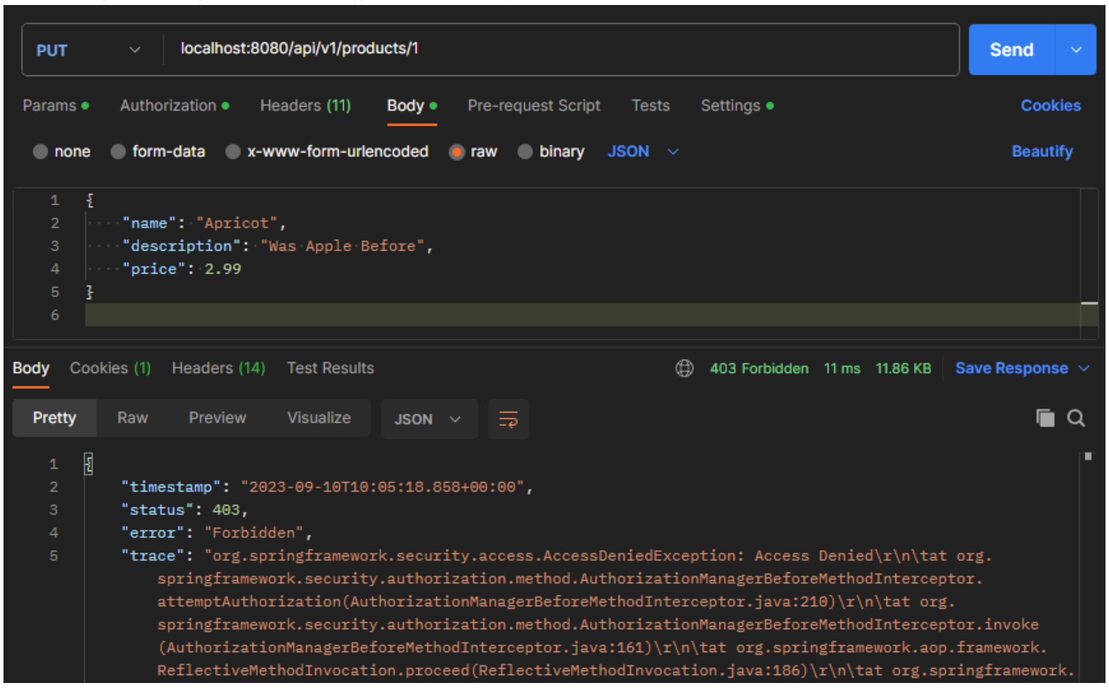
`GET /api/v1/products`

Same as admin

`POST /api/v1/products`

Same as admin

`PUT /api/v1/products/{productid}`



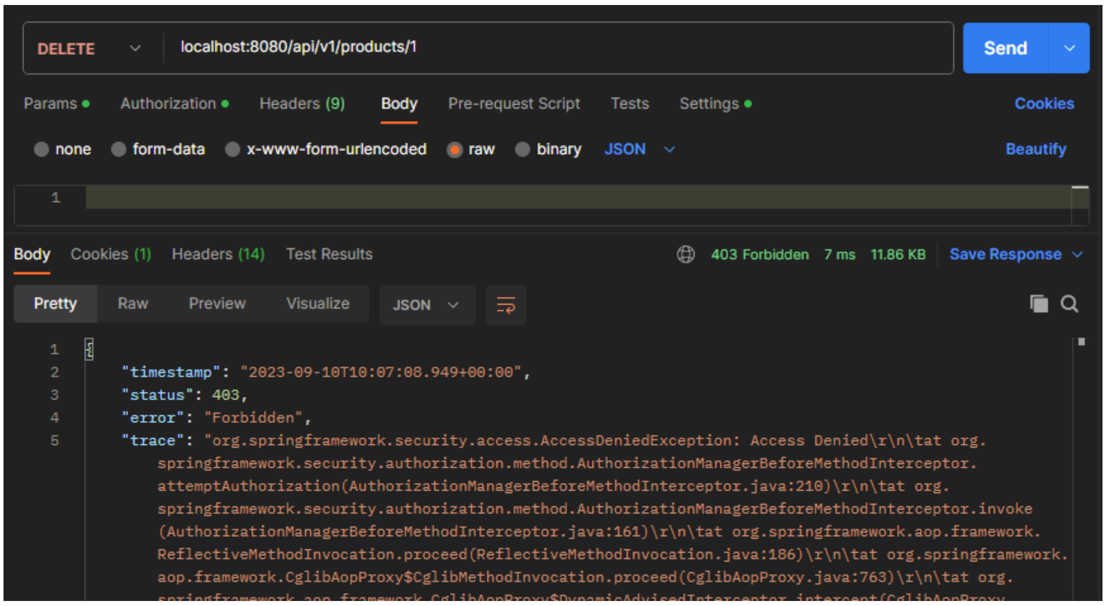
PUT localhost:8080/api/v1/products/1

```
{
    "name": "Apricot",
    "description": "Was Apple Before",
    "price": 2.99
}
```

403 Forbidden 11 ms 11.86 KB

```
{
    "timestamp": "2023-09-10T10:05:18.858+00:00",
    "status": 403,
    "error": "Forbidden",
    "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\r\n\tat org.
        springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.
        attemptAuthorization(AuthorizationManagerBeforeMethodInterceptor.java:210)\r\n\tat org.
        springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke
        (AuthorizationManagerBeforeMethodInterceptor.java:161)\r\n\tat org.springframework.aop.framework.
        ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)\r\n\tat org.springframework.
```

`DELETE /api/v1/products/{productid}`



DELETE localhost:8080/api/v1/products/1

403 Forbidden 7 ms 11.86 KB

```
{
    "timestamp": "2023-09-10T10:07:08.949+00:00",
    "status": 403,
    "error": "Forbidden",
    "trace": "org.springframework.security.access.AccessDeniedException: Access Denied\r\n\tat org.
        springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.
        attemptAuthorization(AuthorizationManagerBeforeMethodInterceptor.java:210)\r\n\tat org.
        springframework.security.authorization.method.AuthorizationManagerBeforeMethodInterceptor.invoke
        (AuthorizationManagerBeforeMethodInterceptor.java:161)\r\n\tat org.springframework.aop.framework.
        ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)\r\n\tat org.springframework.
        aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:763)\r\n\tat org.
        springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy
```