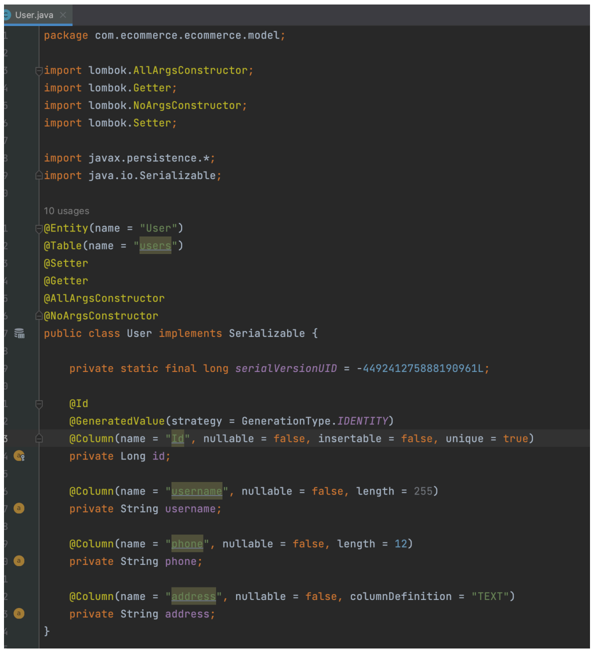
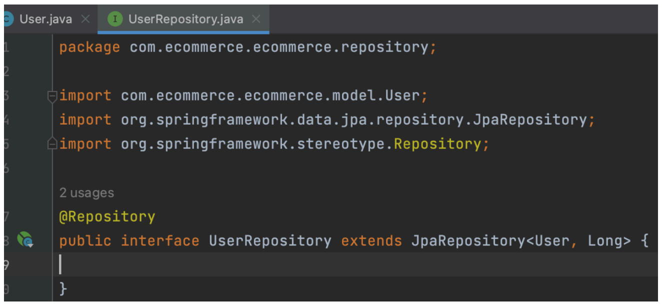
Spring Data JPA

1. CRUD Data User
    a. Create, Read, Update, Delete Code
        i. Model



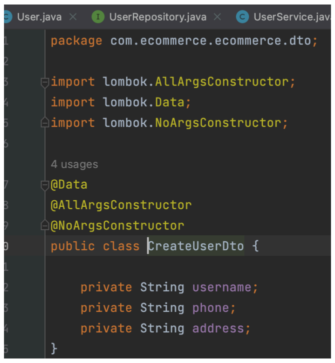
```java
package com.ecommerce.ecommerce.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.persistence.*;
import java.io.Serializable;

10 usages
@Entity(name = "User")
@Table(name = "users")
@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class User implements Serializable {

    private static final long serialVersionUID = -449241275888190961L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Id", nullable = false, insertable = false, unique = true)
    private Long id;

    @Column(name = "username", nullable = false, length = 255)
    private String username;

    @Column(name = "phone", nullable = false, length = 12)
    private String phone;

    @Column(name = "address", nullable = false, columnDefinition = "TEXT")
    private String address;
}
```

        ii. Repository

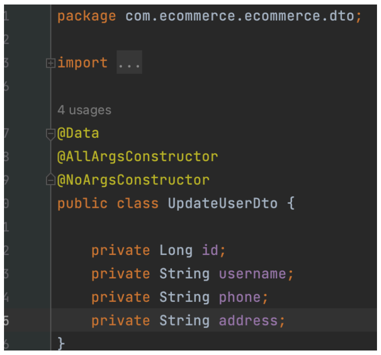


```java
package com.ecommerce.ecommerce.repository;

import com.ecommerce.ecommerce.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

2 usages
@Repository
public interface UserRepository extends JpaRepository<User, Long> {

}
```

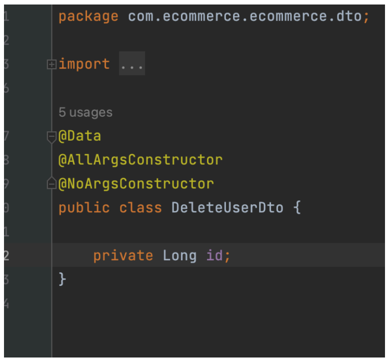
        iii. DTO



```java
package com.ecommerce.ecommerce.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

4 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CreateUserDto {

    private String username;
    private String phone;
    private String address;
}
```

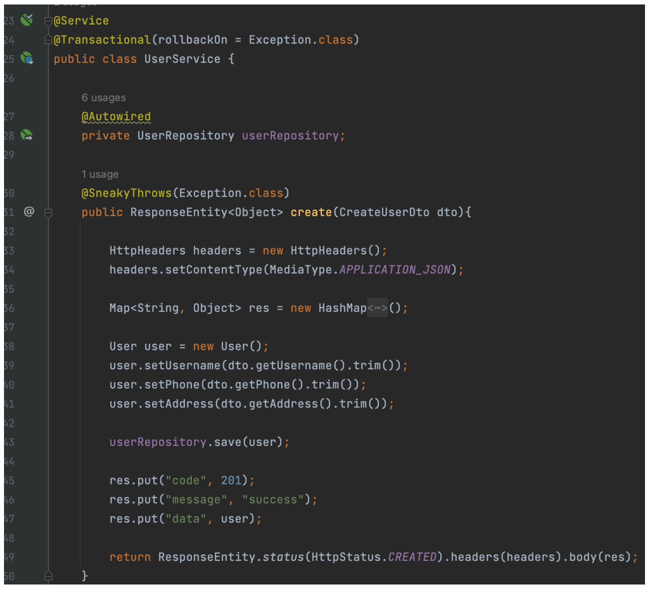


```
package com.ecommerce.ecommerce.dto;

import ...

4 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class UpdateUserDto {

    private Long id;
    private String username;
    private String phone;
    private String address;
}
```

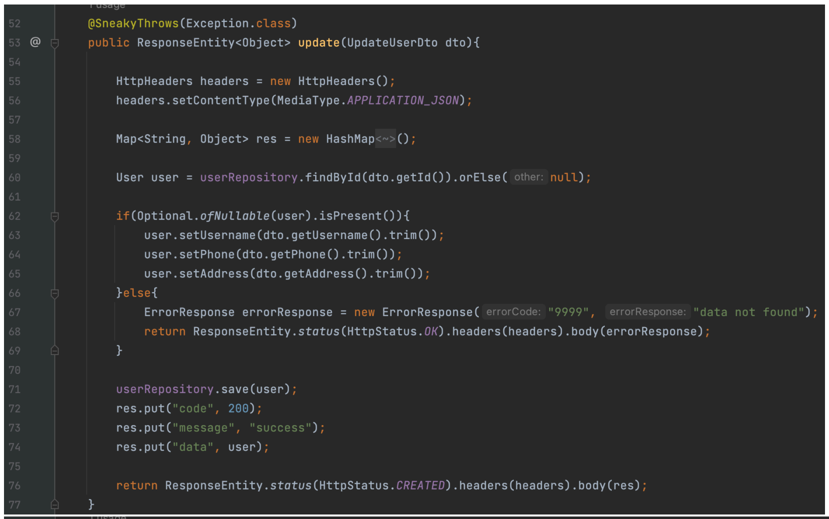


```
package com.ecommerce.ecommerce.dto;

import ...

5 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class DeleteUserDto {

    private Long id;
}
```

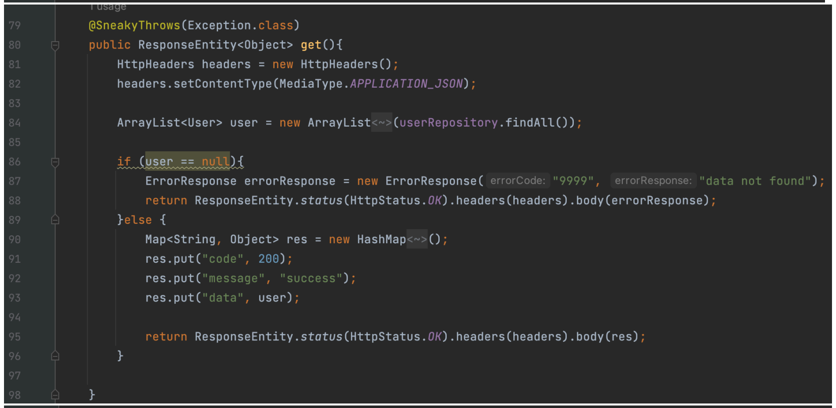
iv. Service



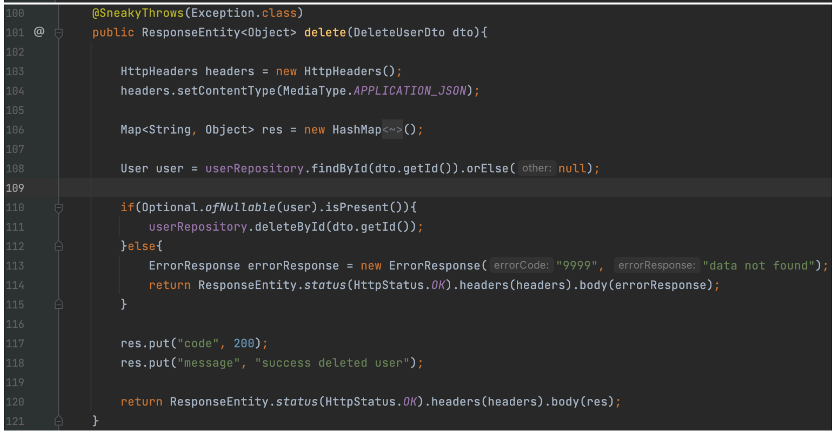
```
@Service
@Transactional(rollbackOn = Exception.class)
public class UserService {

    6 usages
    @Autowired
    private UserRepository userRepository;

    1 usage
    @SneakyThrows(Exception.class)
    public ResponseEntity<Object> create(CreateUserDto dto){

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        Map<String, Object> res = new HashMap<~>();

        User user = new User();
        user.setUsername(dto.getUsername().trim());
        user.setPhone(dto.getPhone().trim());
        user.setAddress(dto.getAddress().trim());

        userRepository.save(user);

        res.put("code", 201);
        res.put("message", "success");
        res.put("data", user);

        return ResponseEntity.status(HttpStatus.CREATED).headers(headers).body(res);
    }
```



```java
@SneakyThrows(Exception.class)
public ResponseEntity<Object> update(UpdateUserDto dto){

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    Map<String, Object> res = new HashMap<~>();

    User user = userRepository.findById(dto.getId()).orElse( other: null);

    if(Optional.ofNullable(user).isPresent()){
        user.setUsername(dto.getUsername().trim());
        user.setPhone(dto.getPhone().trim());
        user.setAddress(dto.getAddress().trim());
    }else{
        ErrorResponse errorResponse = new ErrorResponse( errorCode: "9999",  errorResponse: "data not found");
        return ResponseEntity.status(HttpStatus.OK).headers(headers).body(errorResponse);
    }

    userRepository.save(user);
    res.put("code", 200);
    res.put("message", "success");
    res.put("data", user);

    return ResponseEntity.status(HttpStatus.CREATED).headers(headers).body(res);
}
```

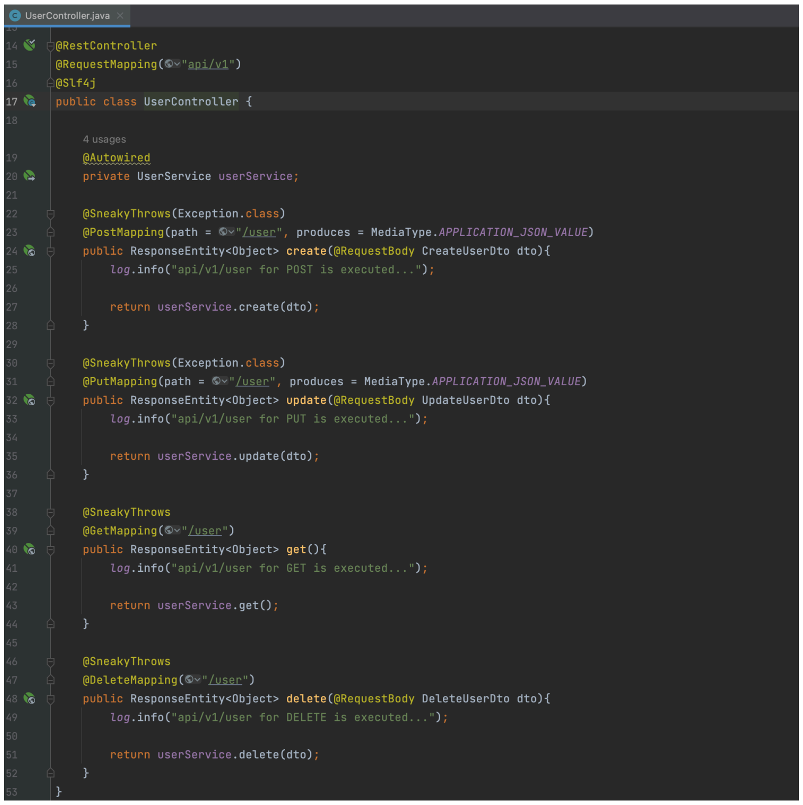


```java
@SneakyThrows(Exception.class)
public ResponseEntity<Object> get(){
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    ArrayList<User> user = new ArrayList<~>(userRepository.findAll());

    if (user == null){
        ErrorResponse errorResponse = new ErrorResponse( errorCode: "9999",  errorResponse: "data not found");
        return ResponseEntity.status(HttpStatus.OK).headers(headers).body(errorResponse);
    }else {
        Map<String, Object> res = new HashMap<~>();
        res.put("code", 200);
        res.put("message", "success");
        res.put("data", user);

        return ResponseEntity.status(HttpStatus.OK).headers(headers).body(res);
    }

}
```



```java
@SneakyThrows(Exception.class)
public ResponseEntity<Object> delete(DeleteUserDto dto){

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    Map<String, Object> res = new HashMap<~>();

    User user = userRepository.findById(dto.getId()).orElse( other: null);

    if(Optional.ofNullable(user).isPresent()){
        userRepository.deleteById(dto.getId());
    }else{
        ErrorResponse errorResponse = new ErrorResponse( errorCode: "9999",  errorResponse: "data not found");
        return ResponseEntity.status(HttpStatus.OK).headers(headers).body(errorResponse);
    }

    res.put("code", 200);
    res.put("message", "success deleted user");

    return ResponseEntity.status(HttpStatus.OK).headers(headers).body(res);
}
```

v. Controller

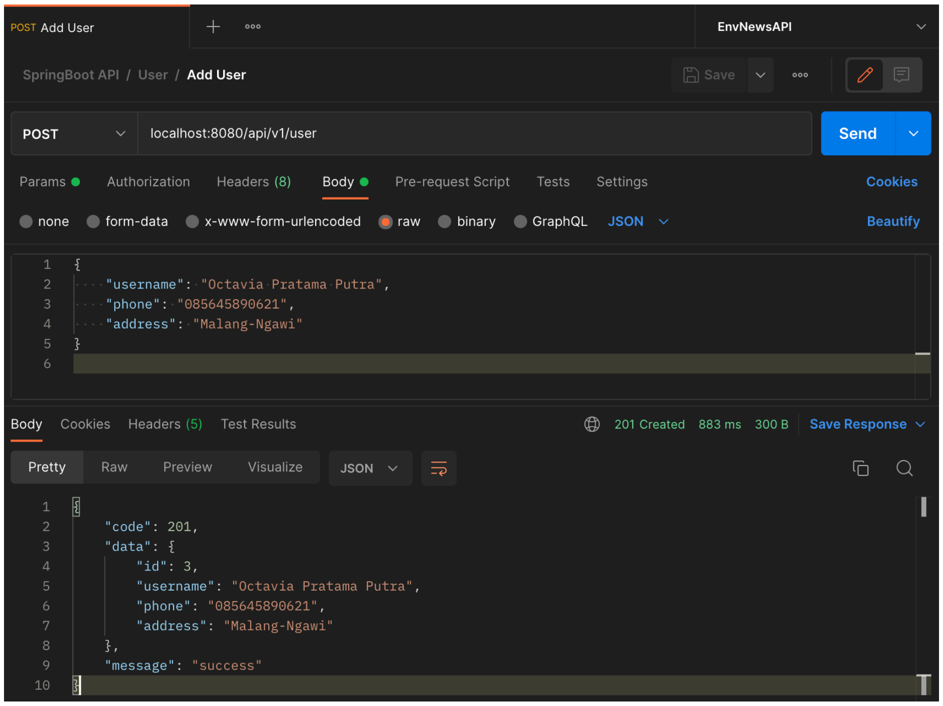


UserController.java

```java
@RestController
@RequestMapping("api/v1")
@Slf4j
public class UserController {

    4 usages
    @Autowired
    private UserService userService;

    @SneakyThrows(Exception.class)
    @PostMapping(path = "/user", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Object> create(@RequestBody CreateUserDto dto){
        log.info("api/v1/user for POST is executed...");

        return userService.create(dto);
    }

    @SneakyThrows(Exception.class)
    @PutMapping(path = "/user", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Object> update(@RequestBody UpdateUserDto dto){
        log.info("api/v1/user for PUT is executed...");

        return userService.update(dto);
    }

    @SneakyThrows
    @GetMapping("/user")
    public ResponseEntity<Object> get(){
        log.info("api/v1/user for GET is executed...");

        return userService.get();
    }

    @SneakyThrows
    @DeleteMapping("/user")
    public ResponseEntity<Object> delete(@RequestBody DeleteUserDto dto){
        log.info("api/v1/user for DELETE is executed...");

        return userService.delete(dto);
    }
}
```

b. Test API CRUD

i. Create



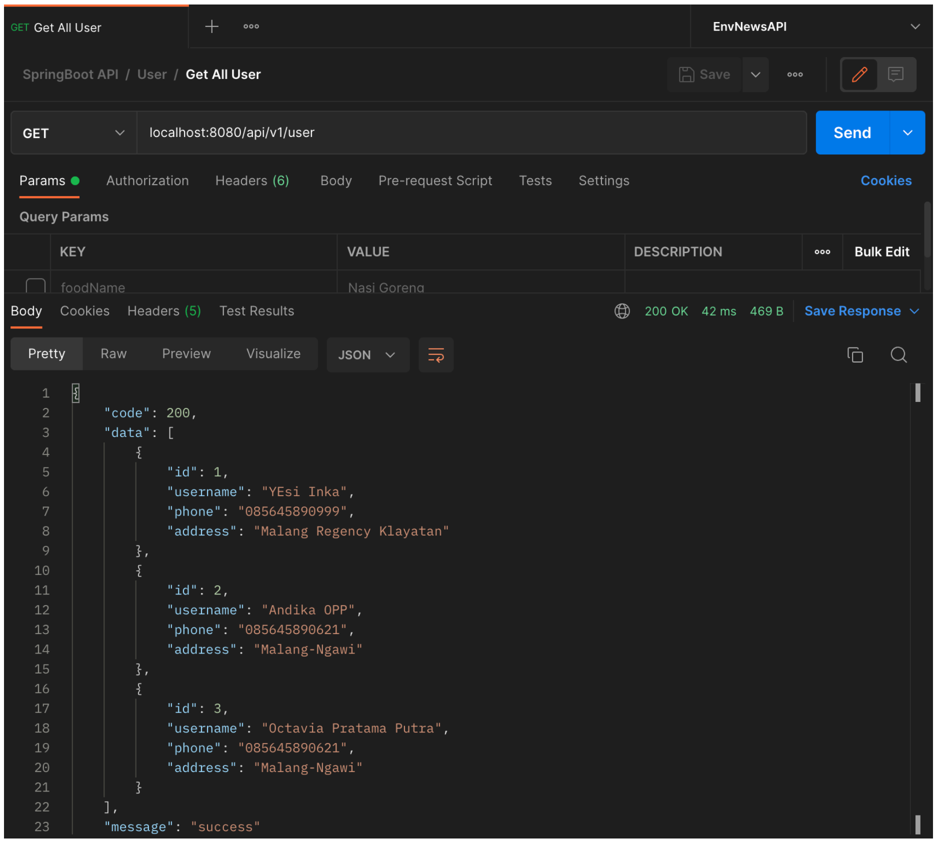
POST localhost:8080/api/v1/user

```json
{
    "username": "Octavia Pratama Putra",
    "phone": "085645890621",
    "address": "Malang-Ngawi"
}
```

201 Created 883 ms 300 B

```json
{
    "code": 201,
    "data": {
        "id": 3,
        "username": "Octavia Pratama Putra",
        "phone": "085645890621",
        "address": "Malang-Ngawi"
    },
    "message": "success"
}
```

ii. Read



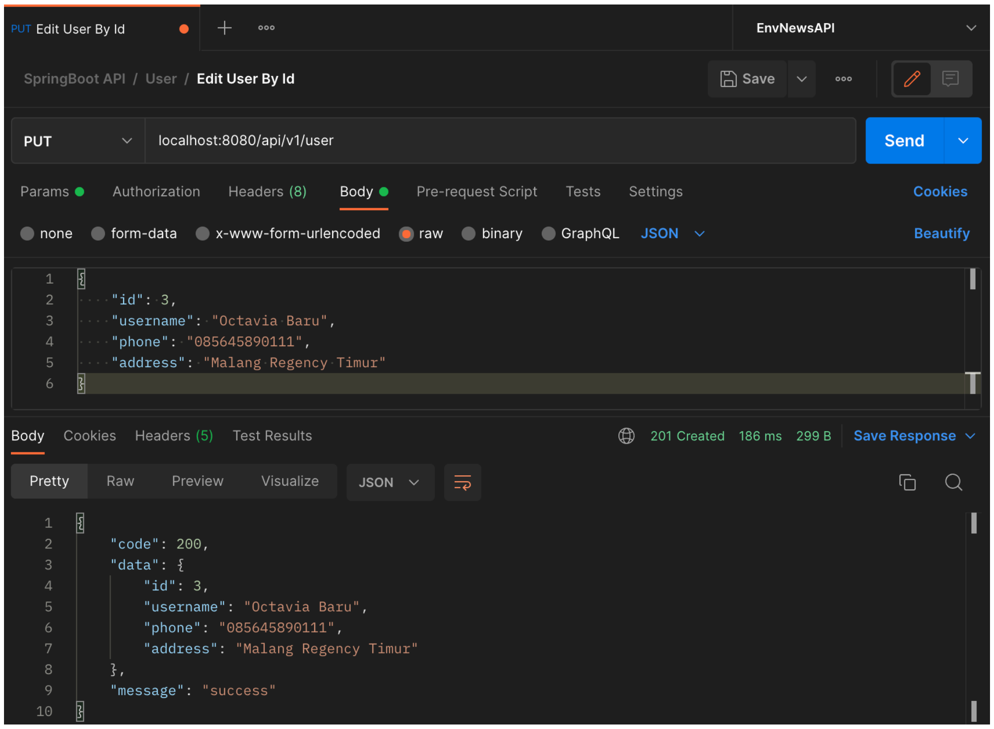
GET localhost:8080/api/v1/user

KEY	VALUE	DESCRIPTION
foodName	Nasi Goreng	

200 OK 42 ms 469 B

```
{
    "code": 200,
    "data": [
        {
            "id": 1,
            "username": "YEsi Inka",
            "phone": "085645890999",
            "address": "Malang Regency Klayatan"
        },
        {
            "id": 2,
            "username": "Andika OPP",
            "phone": "085645890621",
            "address": "Malang-Ngawi"
        },
        {
            "id": 3,
            "username": "Octavia Pratama Putra",
            "phone": "085645890621",
            "address": "Malang-Ngawi"
        }
    ],
    "message": "success"
```

iii. Update



PUT localhost:8080/api/v1/user

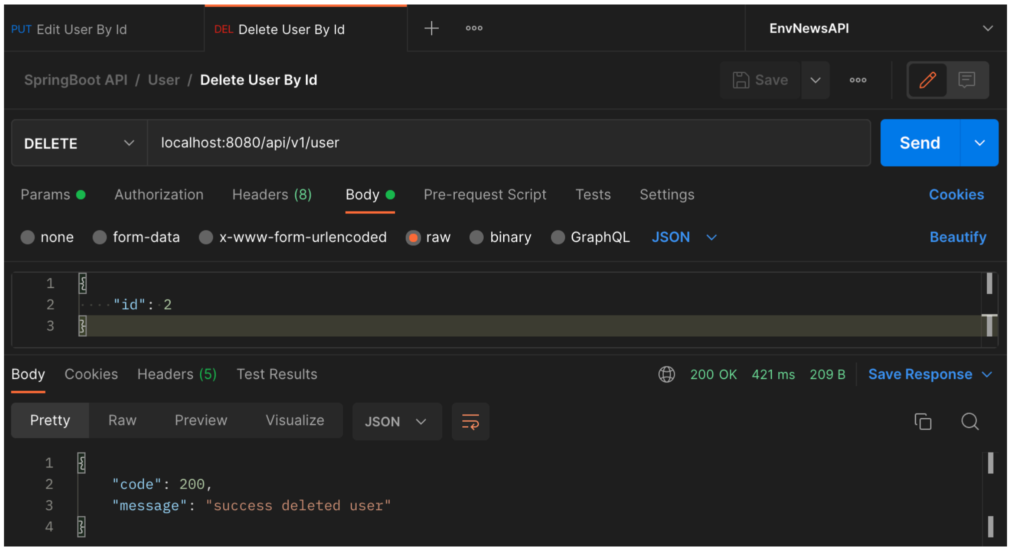
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "id": 3,
    "username": "Octavia Baru",
    "phone": "085645890111",
    "address": "Malang Regency Timur"
}
```

201 Created 186 ms 299 B

```
{
    "code": 200,
    "data": {
        "id": 3,
        "username": "Octavia Baru",
        "phone": "085645890111",
        "address": "Malang Regency Timur"
    },
    "message": "success"
}
```

iv. Delete



PUT Edit User By Id | DEL Delete User By Id | EnvNewsAPI

SpringBoot API / User / Delete User By Id

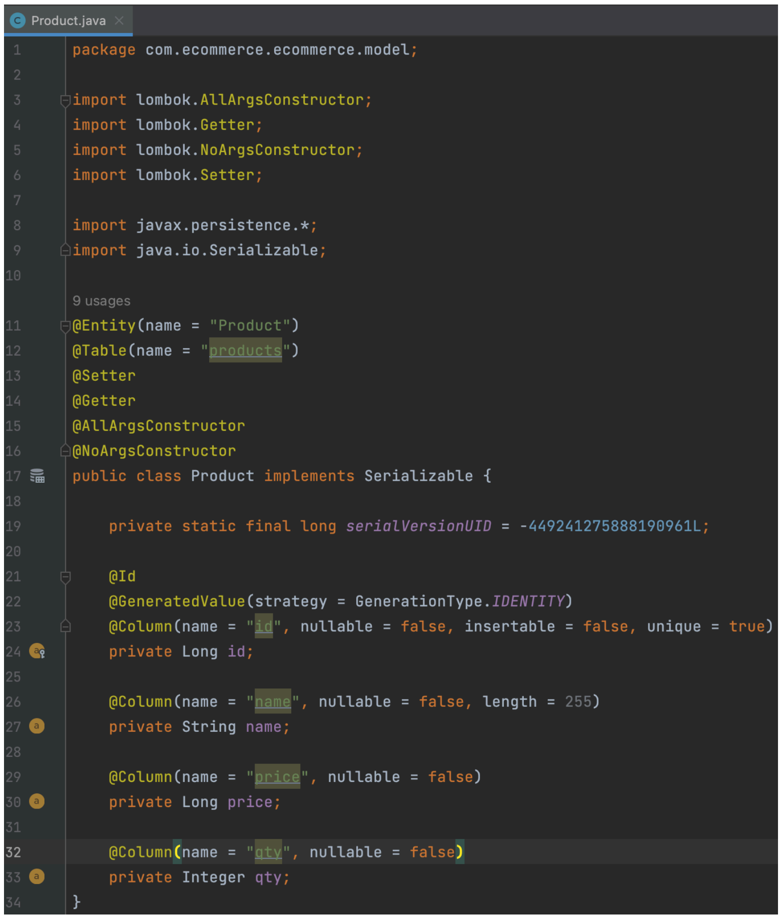
DELETE localhost:8080/api/v1/user

```
{
    "id": 2
}
```

Response: 200 OK 421 ms 209 B

```
{
    "code": 200,
    "message": "success deleted user"
}
```

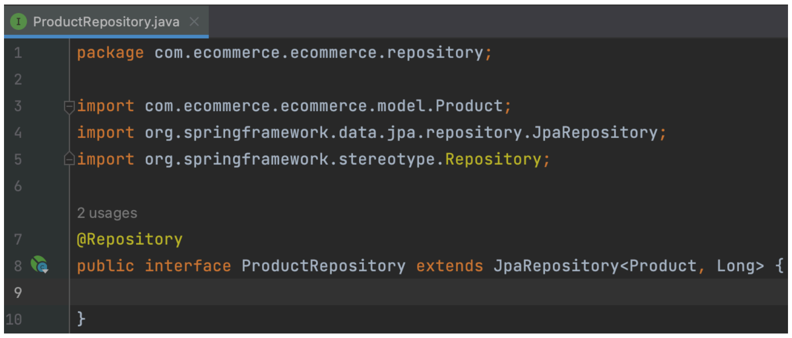
2. CRUD Data Product
   a. Create, Read, Update, Delete Code
      i. Model



Product.java

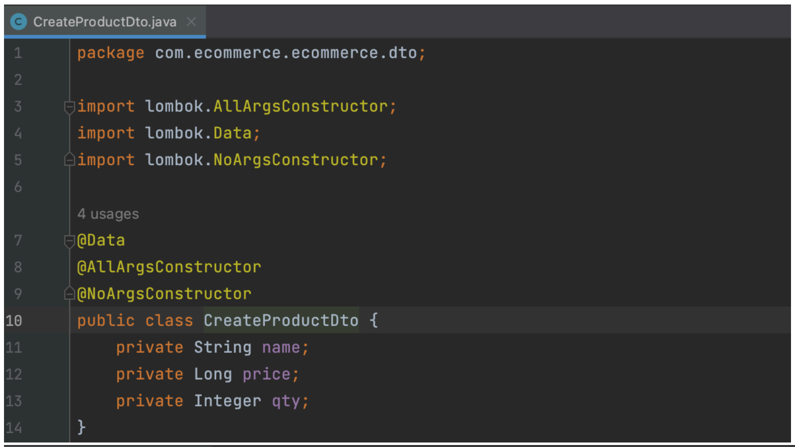
```java
package com.ecommerce.ecommerce.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.persistence.*;
import java.io.Serializable;

9 usages
@Entity(name = "Product")
@Table(name = "products")
@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class Product implements Serializable {

    private static final long serialVersionUID = -449241275888190961L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false, insertable = false, unique = true)
    private Long id;

    @Column(name = "name", nullable = false, length = 255)
    private String name;

    @Column(name = "price", nullable = false)
    private Long price;

    @Column(name = "qty", nullable = false)
    private Integer qty;
}
```

ii. Repository

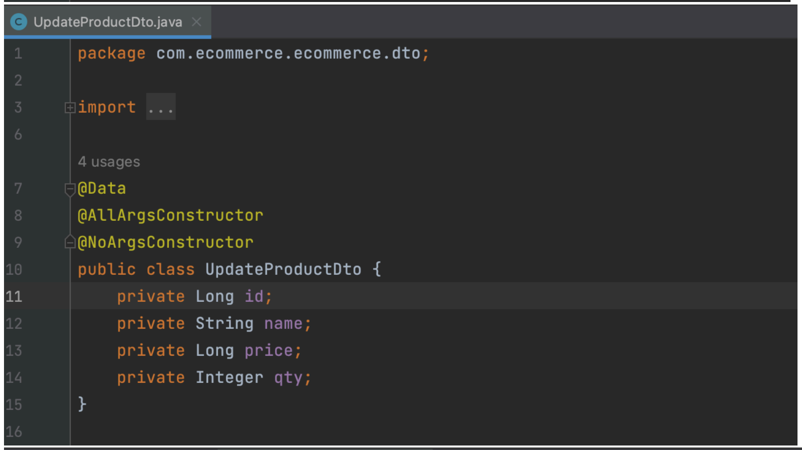

ProductRepository.java

```java
package com.ecommerce.ecommerce.repository;

import com.ecommerce.ecommerce.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

2 usages
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {

}
```

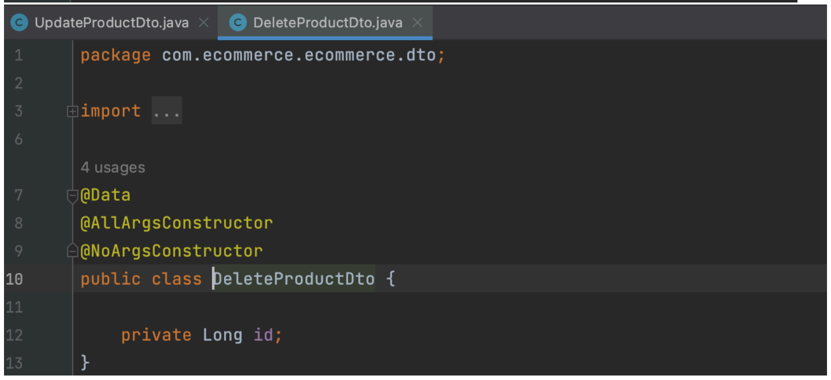
iii. DTO


CreateProductDto.java

```java
package com.ecommerce.ecommerce.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

4 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CreateProductDto {
    private String name;
    private Long price;
    private Integer qty;
}
```

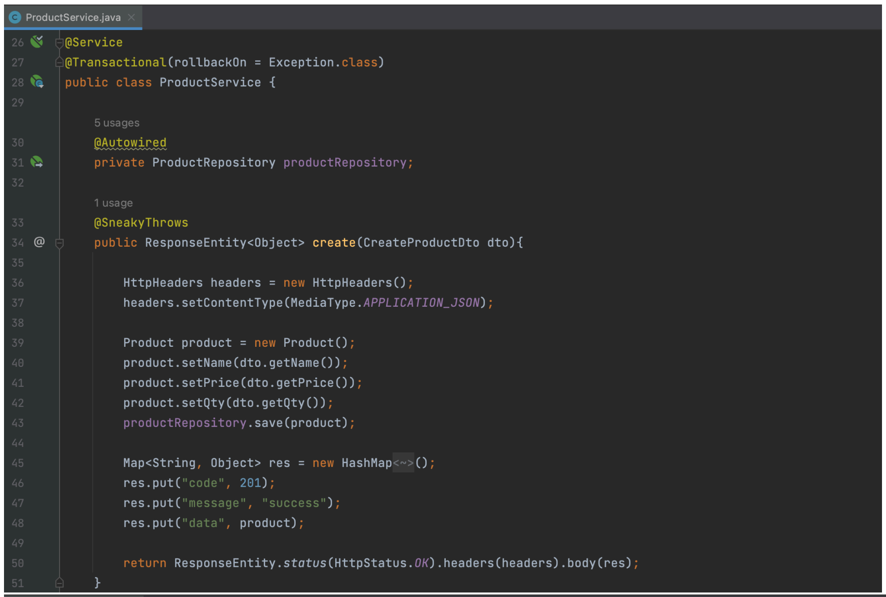

UpdateProductDto.java

```java
package com.ecommerce.ecommerce.dto;

import ...

4 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class UpdateProductDto {
    private Long id;
    private String name;
    private Long price;
    private Integer qty;
}

```


UpdateProductDto.java DeleteProductDto.java

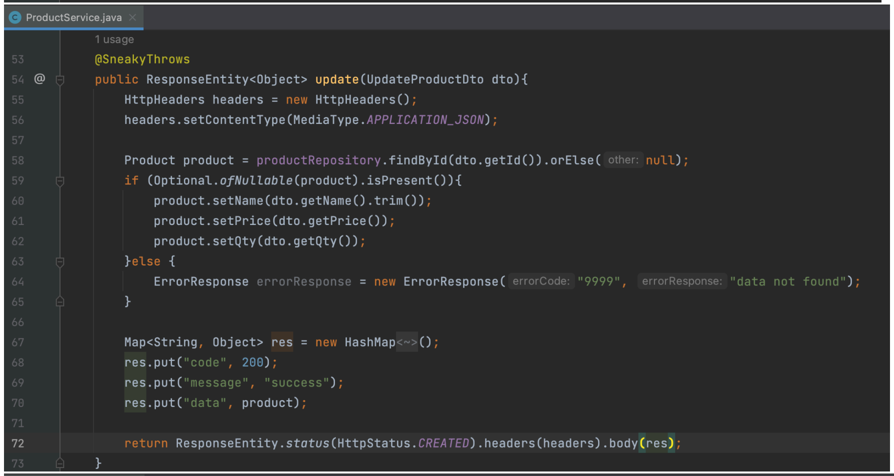
```java
package com.ecommerce.ecommerce.dto;

import ...

4 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class DeleteProductDto {

    private Long id;
}
```

iv. Service



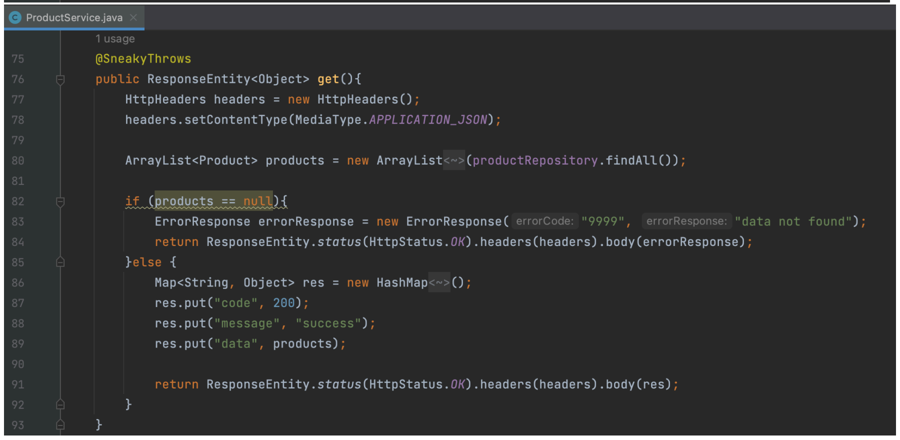
ProductService.java

```java
@Service
@Transactional(rollbackOn = Exception.class)
public class ProductService {

    5 usages
    @Autowired
    private ProductRepository productRepository;

    1 usage
    @SneakyThrows
    public ResponseEntity<Object> create(CreateProductDto dto){

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        Product product = new Product();
        product.setName(dto.getName());
        product.setPrice(dto.getPrice());
        product.setQty(dto.getQty());
        productRepository.save(product);

        Map<String, Object> res = new HashMap<~>();
        res.put("code", 201);
        res.put("message", "success");
        res.put("data", product);

        return ResponseEntity.status(HttpStatus.OK).headers(headers).body(res);
    }
```



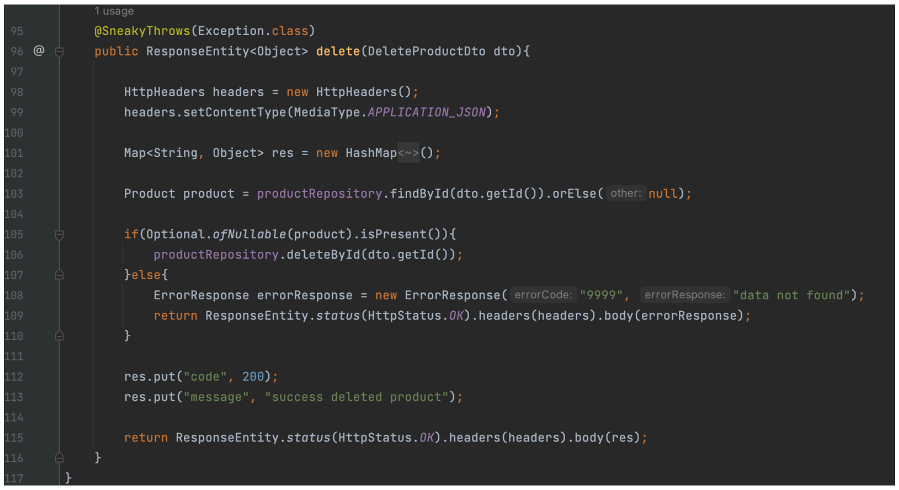
ProductService.java

```java
    1 usage
    @SneakyThrows
    public ResponseEntity<Object> update(UpdateProductDto dto){
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        Product product = productRepository.findById(dto.getId()).orElse( other: null);
        if (Optional.ofNullable(product).isPresent()){
            product.setName(dto.getName().trim());
            product.setPrice(dto.getPrice());
            product.setQty(dto.getQty());
        }else {
            ErrorResponse errorResponse = new ErrorResponse( errorCode: "9999",  errorResponse: "data not found");
        }

        Map<String, Object> res = new HashMap<~>();
        res.put("code", 200);
        res.put("message", "success");
        res.put("data", product);

        return ResponseEntity.status(HttpStatus.CREATED).headers(headers).body(res);
    }
```



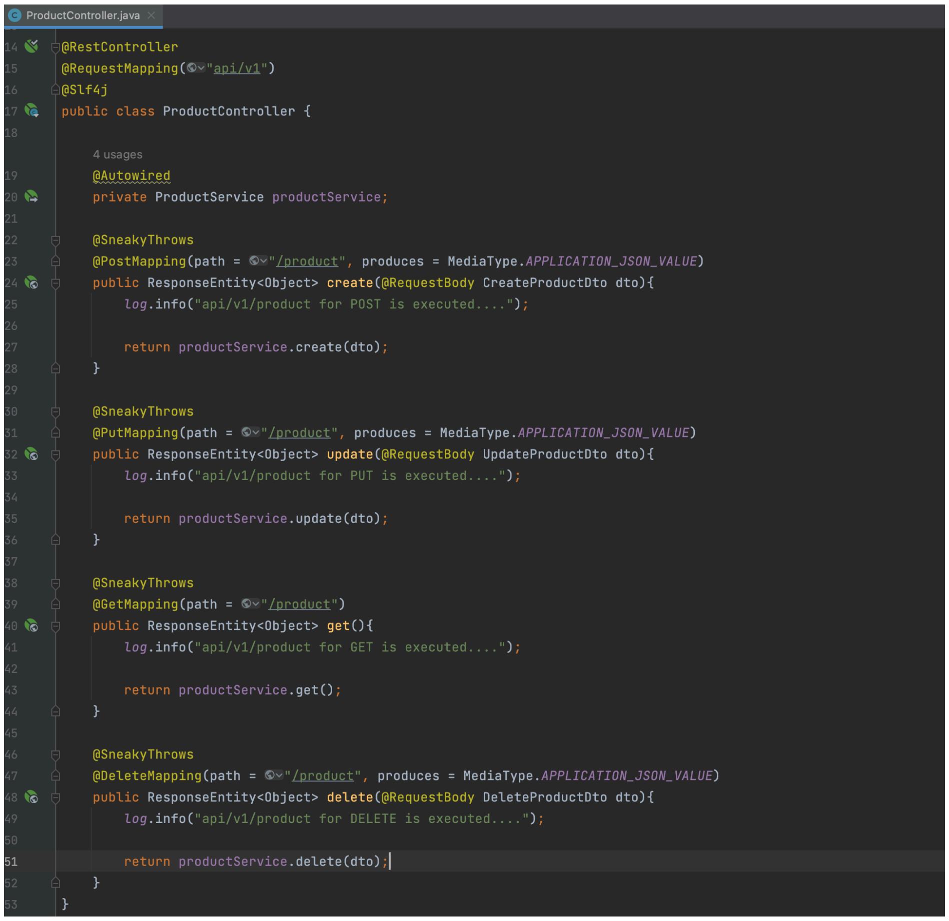
ProductService.java

```java
    1 usage
    @SneakyThrows
    public ResponseEntity<Object> get(){
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        ArrayList<Product> products = new ArrayList<~>(productRepository.findAll());

        if (products == null){
            ErrorResponse errorResponse = new ErrorResponse( errorCode: "9999",  errorResponse: "data not found");
            return ResponseEntity.status(HttpStatus.OK).headers(headers).body(errorResponse);
        }else {
            Map<String, Object> res = new HashMap<~>();
            res.put("code", 200);
            res.put("message", "success");
            res.put("data", products);

            return ResponseEntity.status(HttpStatus.OK).headers(headers).body(res);
        }
    }
```



```
    1 usage
    @SneakyThrows(Exception.class)
    public ResponseEntity<Object> delete(DeleteProductDto dto){

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        Map<String, Object> res = new HashMap<~>();

        Product product = productRepository.findById(dto.getId()).orElse( other: null);

        if(Optional.ofNullable(product).isPresent()){
            productRepository.deleteById(dto.getId());
        }else{
            ErrorResponse errorResponse = new ErrorResponse( errorCode: "9999",  errorResponse: "data not found");
            return ResponseEntity.status(HttpStatus.OK).headers(headers).body(errorResponse);
        }

        res.put("code", 200);
        res.put("message", "success deleted product");

        return ResponseEntity.status(HttpStatus.OK).headers(headers).body(res);
    }
}
```

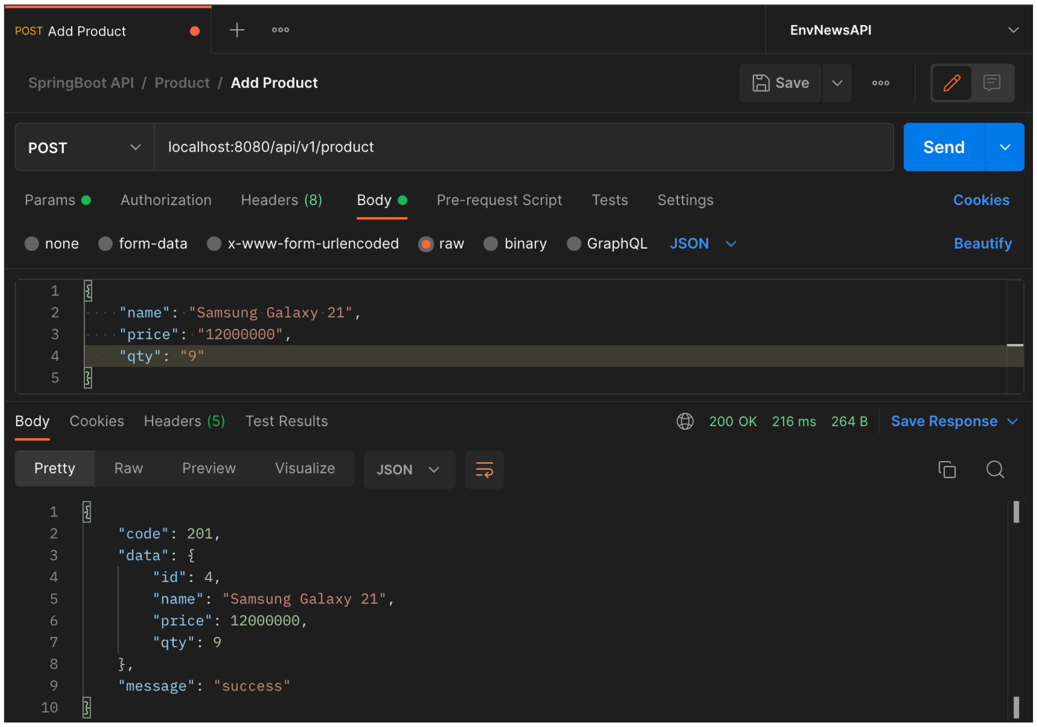
v. Controller



```
ProductController.java

@RestController
@RequestMapping("api/v1")
@Slf4j
public class ProductController {

    4 usages
    @Autowired
    private ProductService productService;

    @SneakyThrows
    @PostMapping(path = "/product", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Object> create(@RequestBody CreateProductDto dto){
        log.info("api/v1/product for POST is executed....");

        return productService.create(dto);
    }

    @SneakyThrows
    @PutMapping(path = "/product", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Object> update(@RequestBody UpdateProductDto dto){
        log.info("api/v1/product for PUT is executed....");

        return productService.update(dto);
    }

    @SneakyThrows
    @GetMapping(path = "/product")
    public ResponseEntity<Object> get(){
        log.info("api/v1/product for GET is executed....");

        return productService.get();
    }

    @SneakyThrows
    @DeleteMapping(path = "/product", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Object> delete(@RequestBody DeleteProductDto dto){
        log.info("api/v1/product for DELETE is executed....");

        return productService.delete(dto);
    }
}
```

b. Test API CRUD

i. Create



POST Add Product
EnvNewsAPI
SpringBoot API / Product / Add Product
Save
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (5) Test Results
Save Response
Pretty Raw Preview Visualize JSON

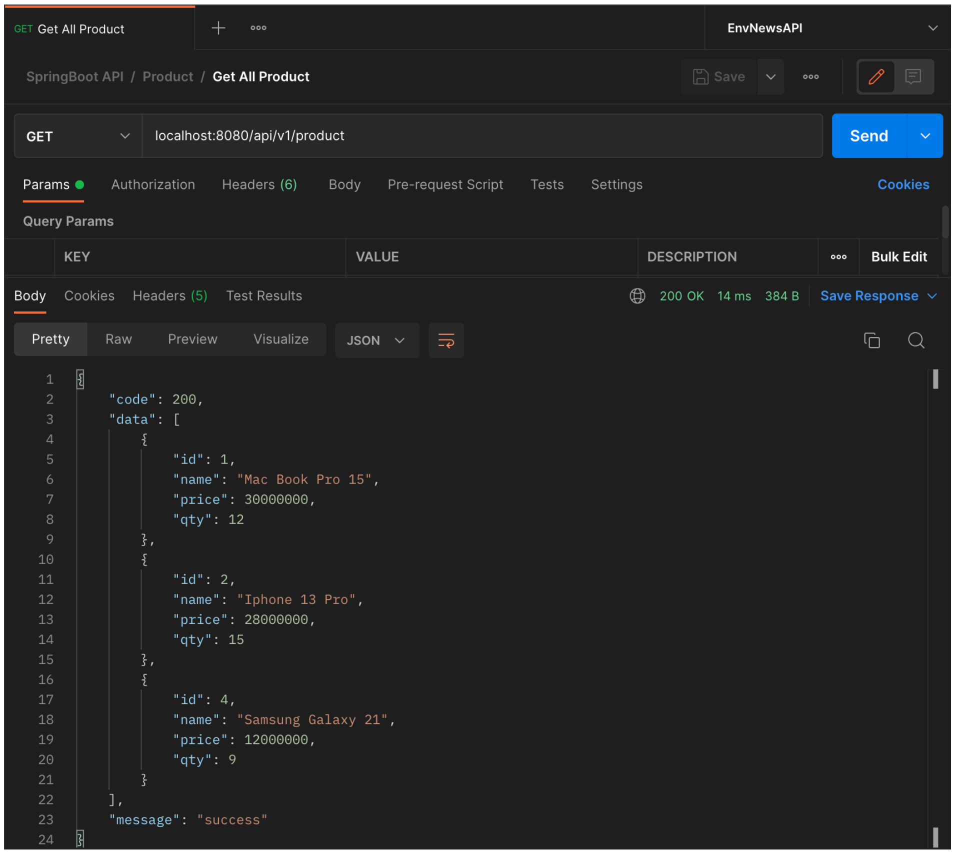
POST localhost:8080/api/v1/product

```
{
    "name": "Samsung Galaxy 21",
    "price": "12000000",
    "qty": "9"
}
```

200 OK 216 ms 264 B

```
{
    "code": 201,
    "data": {
        "id": 4,
        "name": "Samsung Galaxy 21",
        "price": 12000000,
        "qty": 9
    },
    "message": "success"
}
```

ii. Read



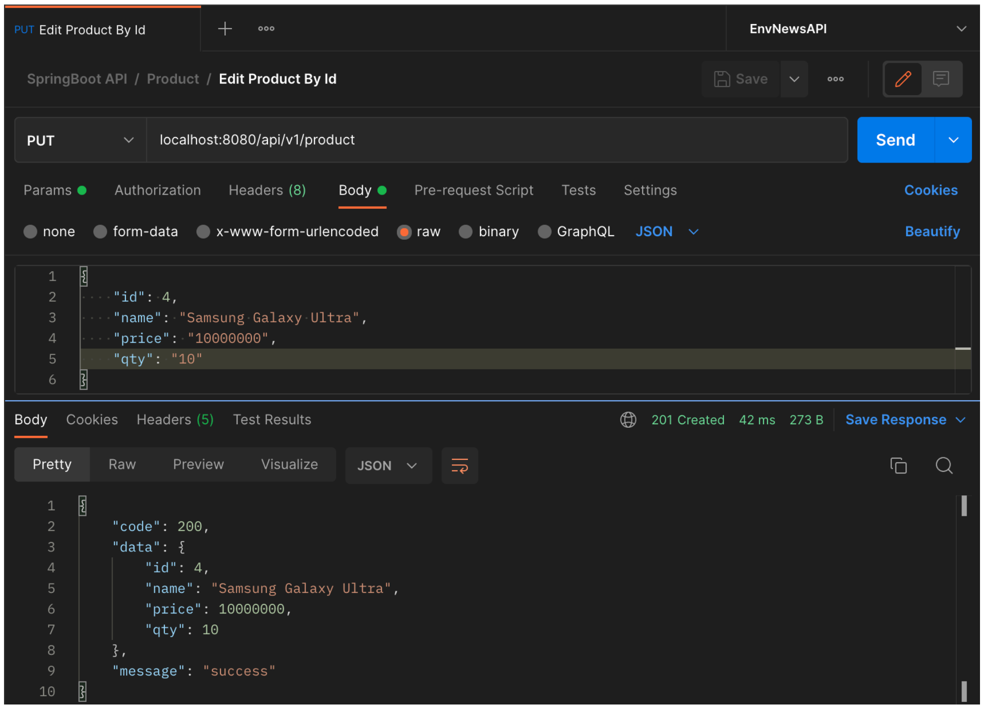
GET Get All Product
EnvNewsAPI
SpringBoot API / Product / Get All Product
Save
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Body Cookies Headers (5) Test Results
Save Response
Pretty Raw Preview Visualize JSON

GET localhost:8080/api/v1/product

200 OK 14 ms 384 B

```
{
    "code": 200,
    "data": [
        {
            "id": 1,
            "name": "Mac Book Pro 15",
            "price": 30000000,
            "qty": 12
        },
        {
            "id": 2,
            "name": "Iphone 13 Pro",
            "price": 28000000,
            "qty": 15
        },
        {
            "id": 4,
            "name": "Samsung Galaxy 21",
            "price": 12000000,
            "qty": 9
        }
    ],
    "message": "success"
}
```

iii. Update


PUT Edit Product By Id

EnvNewsAPI

SpringBoot API / Product / **Edit Product By Id**

PUT localhost:8080/api/v1/product

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

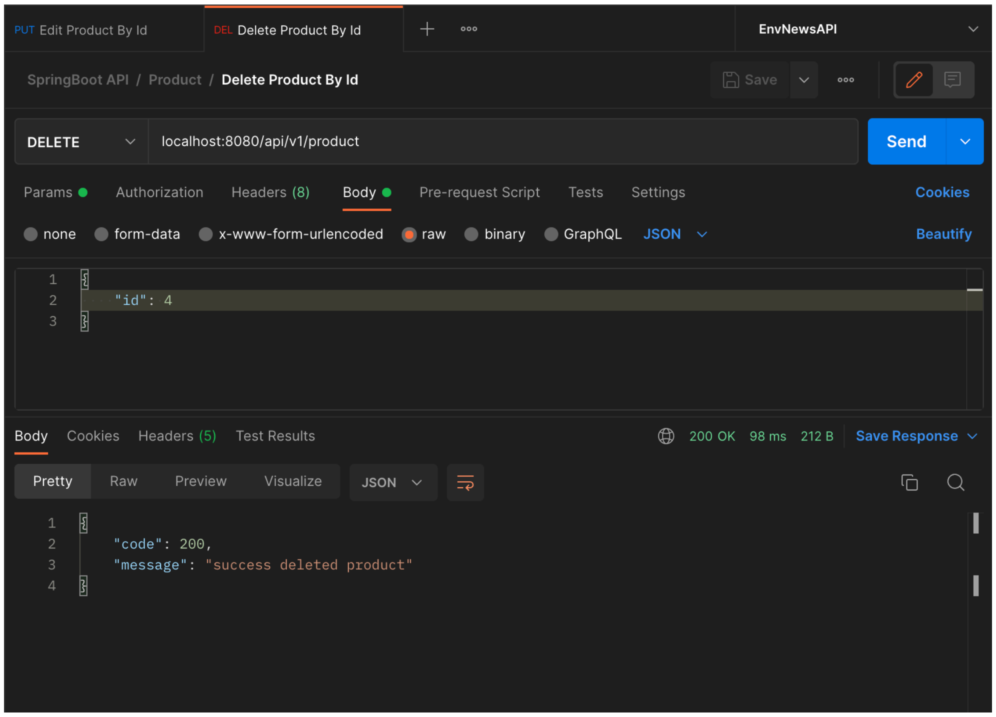
```
{
    "id": 4,
    "name": "Samsung Galaxy Ultra",
    "price": "10000000",
    "qty": "10"
}
```

Body Cookies Headers (5) Test Results 201 Created 42 ms 273 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "code": 200,
    "data": {
        "id": 4,
        "name": "Samsung Galaxy Ultra",
        "price": 10000000,
        "qty": 10
    },
    "message": "success"
}
```

iv. Delete


PUT Edit Product By Id DEL Delete Product By Id

EnvNewsAPI

SpringBoot API / Product / **Delete Product By Id**

DELETE localhost:8080/api/v1/product

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "id": 4
}
```

Body Cookies Headers (5) Test Results 200 OK 98 ms 212 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "code": 200,
    "message": "success deleted product"
}
```