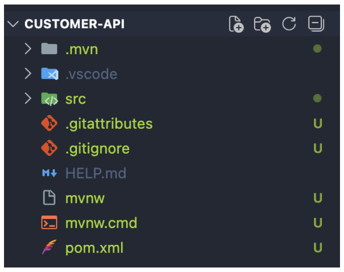
ITITWE23039 - Phan Kim Vĩnh Phúc

Lab 8

Github site & link

https://github.com/alexis0704/web-dev-practice-lab/tree/main/Lab8/customer-api

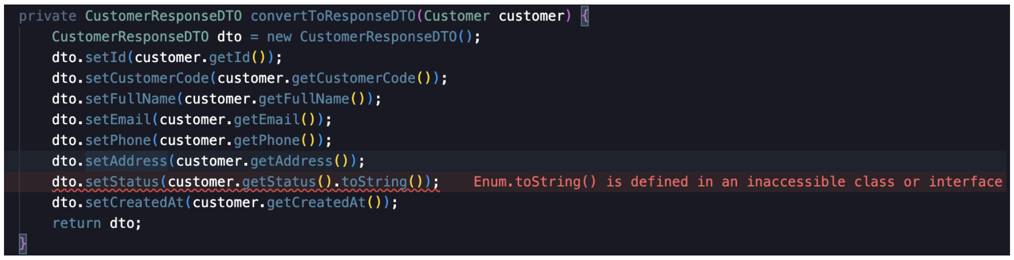
Task 1.1: Create Spring Boot Project



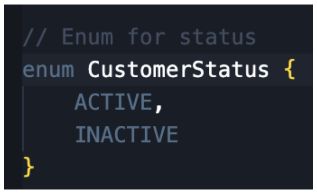
Configuration:
Spring Boot: 4.0.0
Group: com.example
Artifact: customer-api
Java: 17

Task 1.2, 1.3, 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, 4.1, 4.2: Database Setup, Create Customer Entity, Create Request DTO, Create Response DTO, Create Error Response DTO, Create Repository, Create Service Interface, Implement Service, Create Basic REST Controller, Add Exception Handling

I simply pasted in the given code in the guide.
However, there's a bug as follows

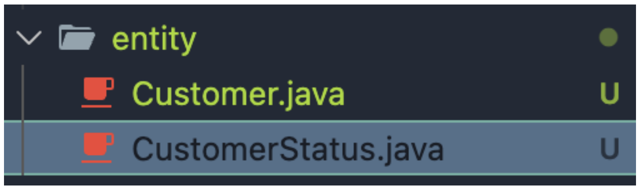


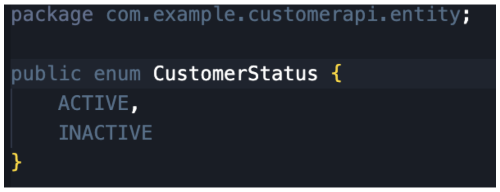
```
private CustomerResponseDTO convertToResponseDTO(Customer customer) {
    CustomerResponseDTO dto = new CustomerResponseDTO();
    dto.setId(customer.getId());
    dto.setCustomerCode(customer.getCustomerCode());
    dto.setFullName(customer.getFullName());
    dto.setEmail(customer.getEmail());
    dto.setPhone(customer.getPhone());
    dto.setAddress(customer.getAddress());
    dto.setStatus(customer.getStatus().toString());    Enum.toString() is defined in an inaccessible class or interface
    dto.setCreatedAt(customer.getCreatedAt());
    return dto;
}
```



```
// Enum for status
enum CustomerStatus {
    ACTIVE,
    INACTIVE
}
```

The service couldn't use the method of the CustomerStatus enum, defined in the Customer.java entity file. Here, I think we can either modify the Customer's getStatus() method to implement toString(), or we can create a new CustomerStatus enum class.
I go with the second approach, because it improves maintainability and readability. Also, it avoids modifying the entity's basic method unnecessarily.

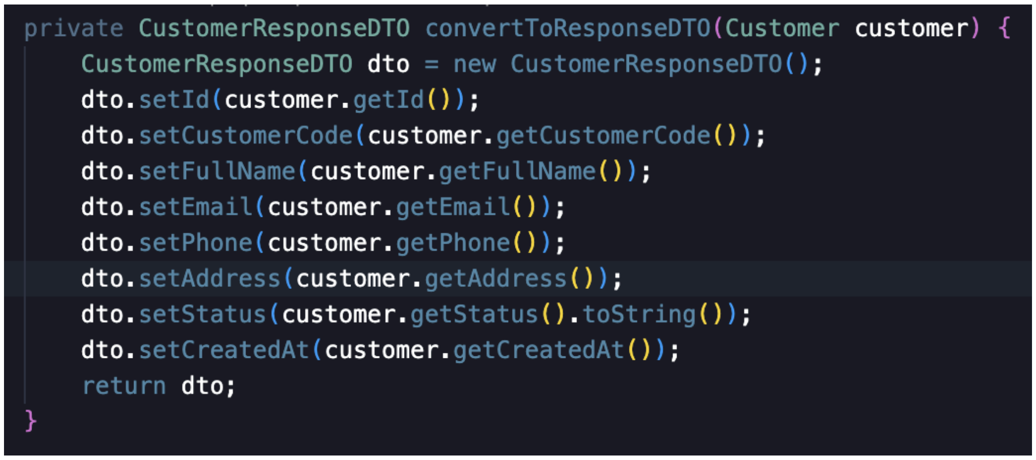




```java
package com.example.customerapi.entity;

public enum CustomerStatus {
    ACTIVE,
    INACTIVE
}
```

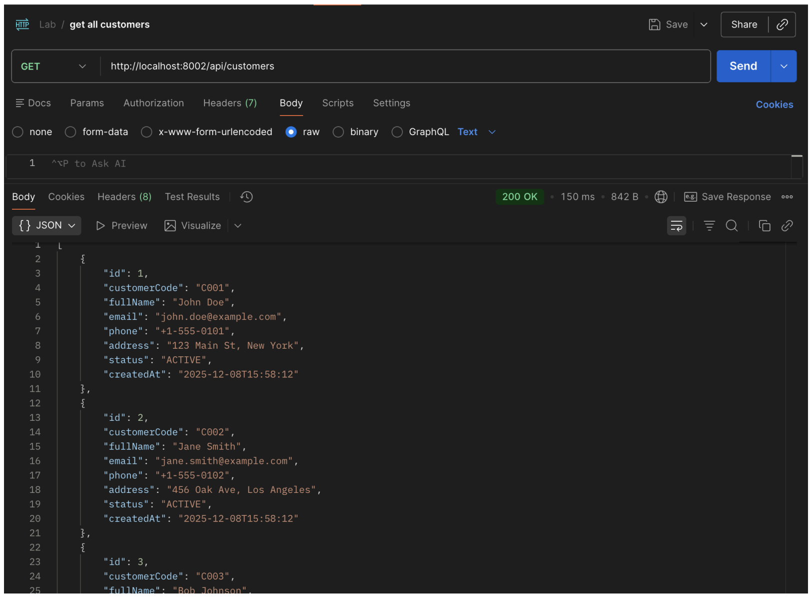
After moving CustomerStatus to its own public enum class, the method shows no bug anymore:



```java
private CustomerResponseDTO convertToResponseDTO(Customer customer) {
    CustomerResponseDTO dto = new CustomerResponseDTO();
    dto.setId(customer.getId());
    dto.setCustomerCode(customer.getCustomerCode());
    dto.setFullName(customer.getFullName());
    dto.setEmail(customer.getEmail());
    dto.setPhone(customer.getPhone());
    dto.setAddress(customer.getAddress());
    dto.setStatus(customer.getStatus().toString());
    dto.setCreatedAt(customer.getCreatedAt());
    return dto;
}
```

Testing all endpoints with Postman:

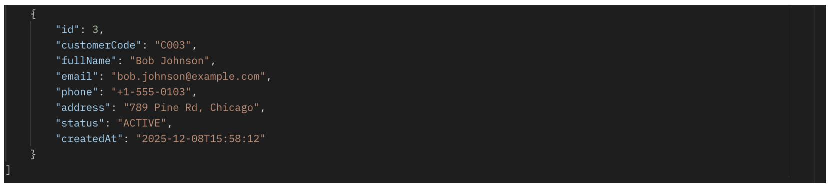
Get all customers



GET http://localhost:8002/api/customers

200 OK • 150 ms • 842 B

```json
[
    {
        "id": 1,
        "customerCode": "C001",
        "fullName": "John Doe",
        "email": "john.doe@example.com",
        "phone": "+1-555-0101",
        "address": "123 Main St, New York",
        "status": "ACTIVE",
        "createdAt": "2025-12-08T15:58:12"
    },
    {
        "id": 2,
        "customerCode": "C002",
        "fullName": "Jane Smith",
        "email": "jane.smith@example.com",
        "phone": "+1-555-0102",
        "address": "456 Oak Ave, Los Angeles",
        "status": "ACTIVE",
        "createdAt": "2025-12-08T15:58:12"
    },
    {
        "id": 3,
        "customerCode": "C003",
        "fullName": "Bob Johnson",
```

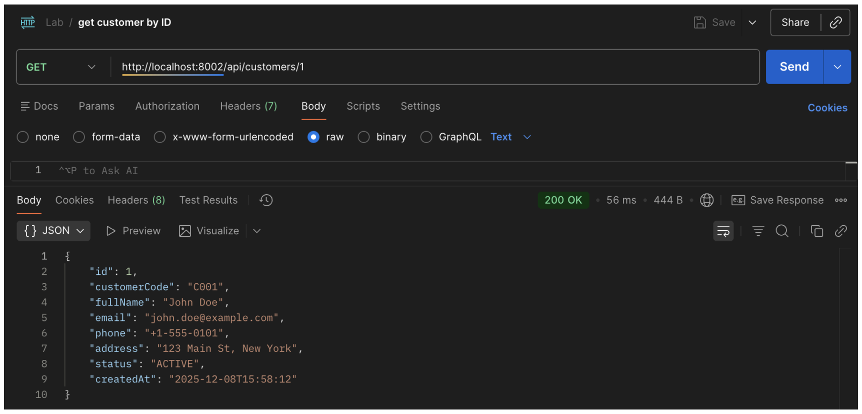


```
  {
      "id": 3,
      "customerCode": "C003",
      "fullName": "Bob Johnson",
      "email": "bob.johnson@example.com",
      "phone": "+1-555-0103",
      "address": "789 Pine Rd, Chicago",
      "status": "ACTIVE",
      "createdAt": "2025-12-08T15:58:12"
  }
]
```

Flow:

- Controller (CustomerRestController.getAllCustomers()) calls customerService.getAllCustomers()
- Service (CustomerServiceImpl.getAllCustomers()) calls customerRepository.findAll() to fetch all customer entities, converts each Customer entity to CustomerResponseDTO
- Repository (CustomerRepository.findAll()) runs SQL: SELECT * FROM customers
- Service returns list of DTOs
- Controller returns 200 OK with JSON list

Get customer by ID



GET http://localhost:8002/api/customers/1

Body: raw, Text

200 OK • 56 ms • 444 B

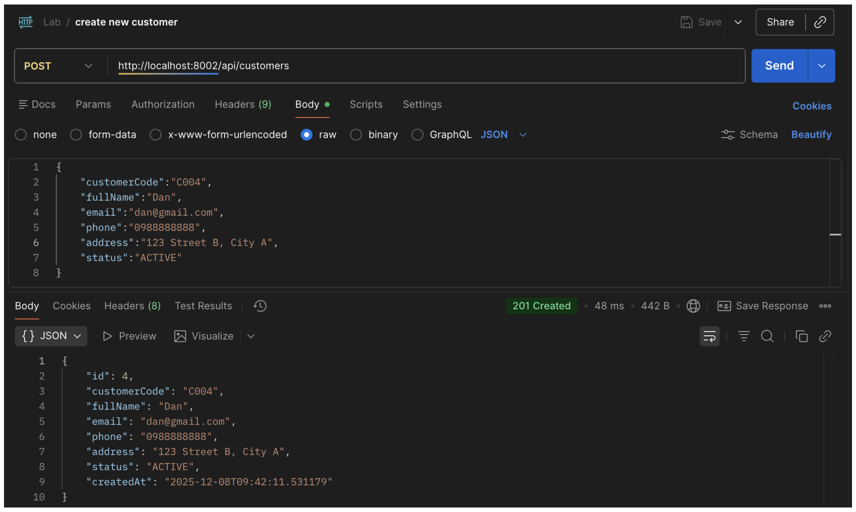
{} JSON

```
{
    "id": 1,
    "customerCode": "C001",
    "fullName": "John Doe",
    "email": "john.doe@example.com",
    "phone": "+1-555-0101",
    "address": "123 Main St, New York",
    "status": "ACTIVE",
    "createdAt": "2025-12-08T15:58:12"
}
```

Flow:

- Controller (getCustomerById()) calls customerService.getCustomerById(id)
- Service (getCustomerById()) calls customerRepository.findById(id)
    - If not found, throws ResourceNotFoundException (caught by GlobalExceptionHandler, shows 404)
    - If found, converts entity to DTO
- Controller returns 200 OK with the DTO

Create new customer


Lab / create new customer

POST http://localhost:8002/api/customers

Body: raw JSON

```
{
    "customerCode":"C004",
    "fullName":"Dan",
    "email":"dan@gmail.com",
    "phone":"0988888888",
    "address":"123 Street B, City A",
    "status":"ACTIVE"
}
```

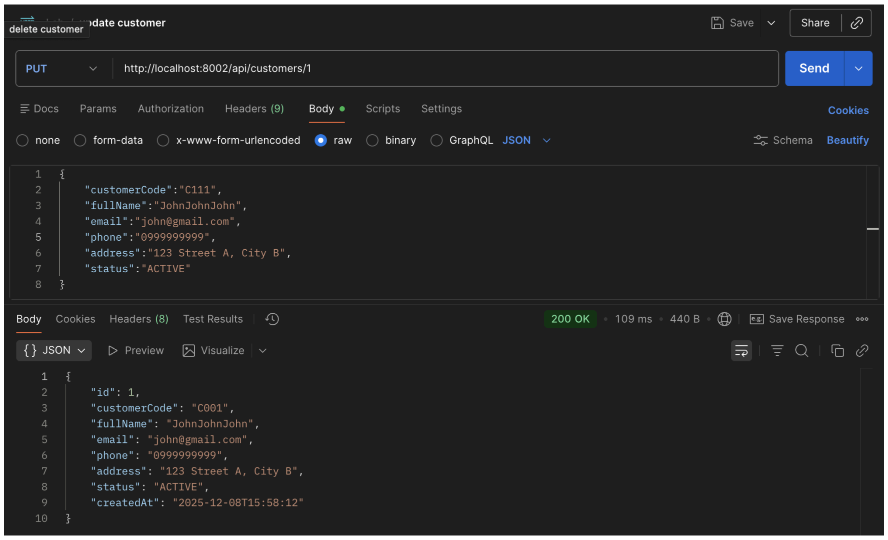
201 Created · 48 ms · 442 B

```
{
    "id": 4,
    "customerCode": "C004",
    "fullName": "Dan",
    "email": "dan@gmail.com",
    "phone": "0988888888",
    "address": "123 Street B, City A",
    "status": "ACTIVE",
    "createdAt": "2025-12-08T09:42:11.531179"
}
```

Flow:

- Controller (createCustomer()) receives CustomerRequestDTO, triggers validation annotations (@Valid) and calls customerService.createCustomer(requestDTO)
- Service (createCustomer()) checks duplicates: existsByCustomerCode() and existsByEmail()
    - If duplicate, throws DuplicateResourceException (409 Conflict)
    - If not, converts DTO to Entity (convertToEntity()), sets default enum status = ACTIVE, saves new customer using save() and converts saved entity to Response DTO
- Controller returns 201 Created with the DTO

Update Customer


delete customer ›date customer

PUT http://localhost:8002/api/customers/1

Body: raw JSON

```
{
    "customerCode":"C111",
    "fullName":"JohnJohnJohn",
    "email":"john@gmail.com",
    "phone":"0999999999",
    "address":"123 Street A, City B",
    "status":"ACTIVE"
}
```

200 OK · 109 ms · 440 B

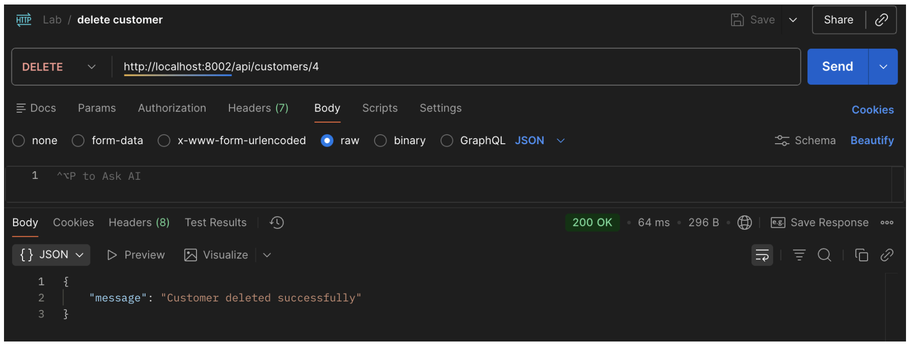
```
{
    "id": 1,
    "customerCode": "C001",
    "fullName": "JohnJohnJohn",
    "email": "john@gmail.com",
    "phone": "0999999999",
    "address": "123 Street A, City B",
    "status": "ACTIVE",
    "createdAt": "2025-12-08T15:58:12"
}
```

Flow:

- Controller (updateCustomer()) receives DTO and calls customerService.updateCustomer(id, requestDTO)

- Service (updateCustomer())
    - findById(id), if missing, throw ResourceNotFoundException (404)
    - Checks if the email is being changed to an existing one. If yes, throw DuplicateResourceException (409)
    - Updates necessary fields, saves the updated entity, converts entity to DTO
- Controller returns 200 OK with updated DTO

Delete customer



DELETE http://localhost:8002/api/customers/4

Body: none / form-data / x-www-form-urlencoded / [✓ raw] / binary / GraphQL — JSON

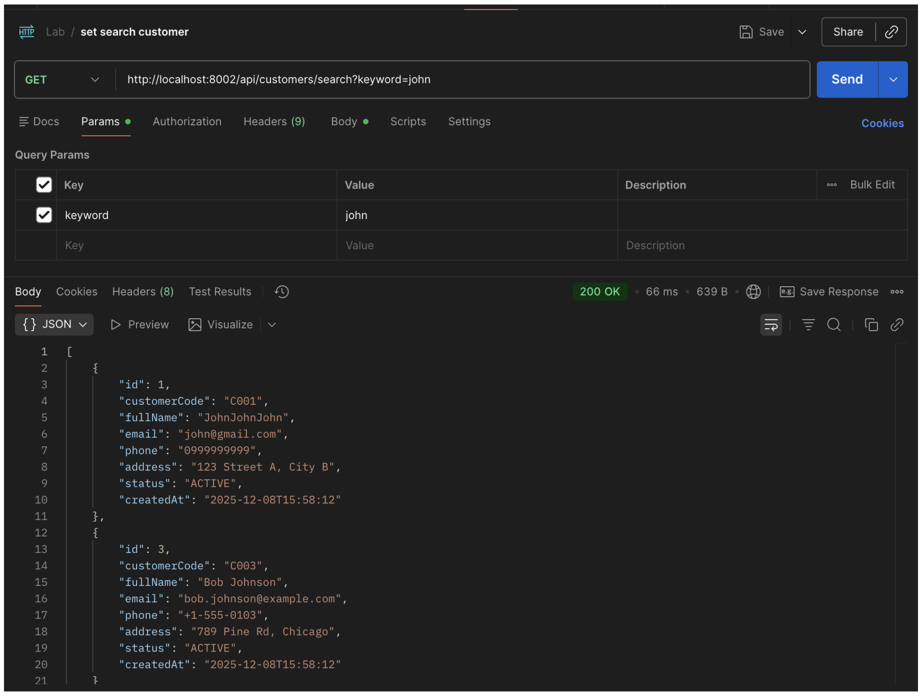
200 OK • 64 ms • 296 B

```
{
    "message": "Customer deleted successfully"
}
```

Flow:

- Controller (deleteCustomer()) → calls customerService.deleteCustomer(id)
- Service (deleteCustomer())
    - Checks existsById(id), if not found, throw ResourceNotFoundException (404)
    - Calls deleteById(id)
- Controller returns 200 OK with success message

Search customer



GET http://localhost:8002/api/customers/search?keyword=john

Query Params

	Key	Value	Description
✓	keyword	john	
	Key	Value	Description

200 OK • 66 ms • 639 B

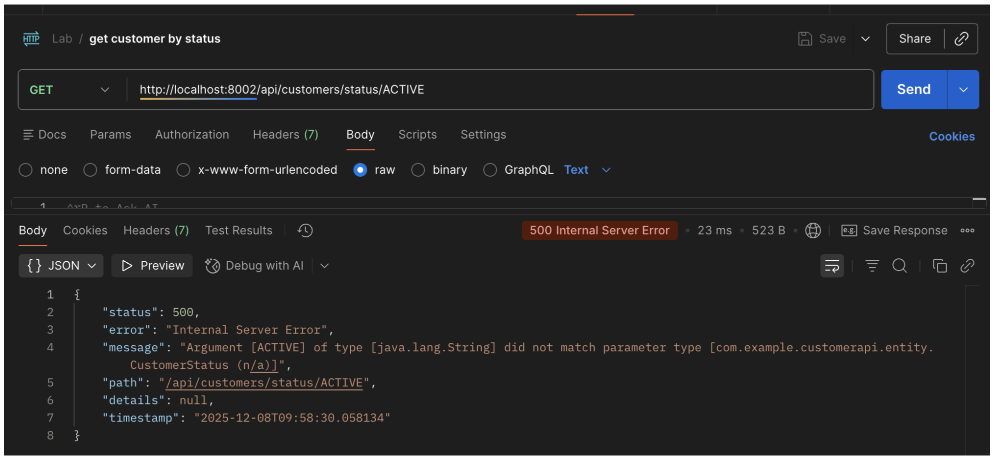
```
[
    {
        "id": 1,
        "customerCode": "C001",
        "fullName": "JohnJohnJohn",
        "email": "john@gmail.com",
        "phone": "0999999999",
        "address": "123 Street A, City B",
        "status": "ACTIVE",
        "createdAt": "2025-12-08T15:58:12"
    },
    {
        "id": 3,
        "customerCode": "C003",
        "fullName": "Bob Johnson",
        "email": "bob.johnson@example.com",
        "phone": "+1-555-0103",
        "address": "789 Pine Rd, Chicago",
        "status": "ACTIVE",
        "createdAt": "2025-12-08T15:58:12"
    }
```

Flow:

- Controller (searchCustomers()) calls customerService.searchCustomers(keyword)
- Service (searchCustomers()) calls customerRepository.searchCustomers(keyword)
- Repository (@Query) runs JPQL search across fullName, email, customerCode
- Service converts results to DTOs
- Controller returns 200 OK with the list

Get customer by status

There's a type mismatch bug here for enum.



GET http://localhost:8002/api/customers/status/ACTIVE

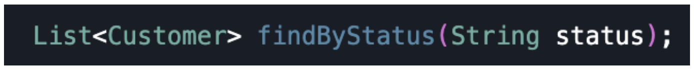
Body: none ( ) form-data ( ) x-www-form-urlencoded ( ) [✓ raw] binary ( ) GraphQL ( ) Text

500 Internal Server Error • 23 ms • 523 B

```
{
    "status": 500,
    "error": "Internal Server Error",
    "message": "Argument [ACTIVE] of type [java.lang.String] did not match parameter type [com.example.customerapi.entity.
        CustomerStatus (n/a)]",
    "path": "/api/customers/status/ACTIVE",
    "details": null,
    "timestamp": "2025-12-08T09:58:30.058134"
}
```

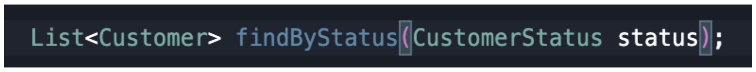
This is because the entity's attribute is CustomerStatus, but the whole flow from the Controller, to Service and to Repository throw in String status.
To fix the bug, I make the Service class convert the string to the enum type and make the Repository class save the enum:

- Old repository method



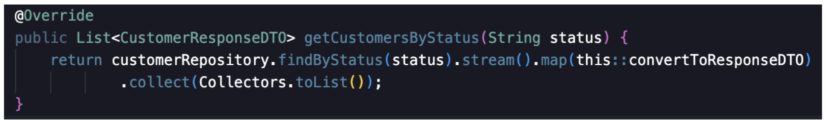
```java
List<Customer> findByStatus(String status);
```

- New repository method



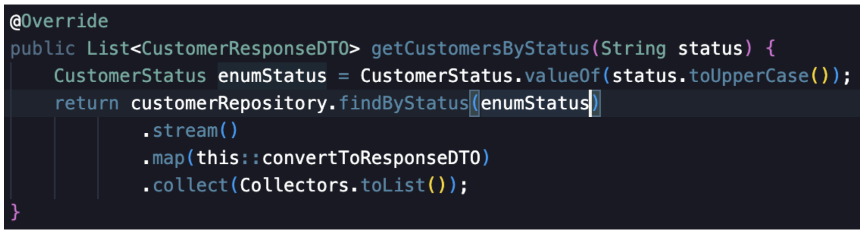
```java
List<Customer> findByStatus(CustomerStatus status);
```

- Old service method



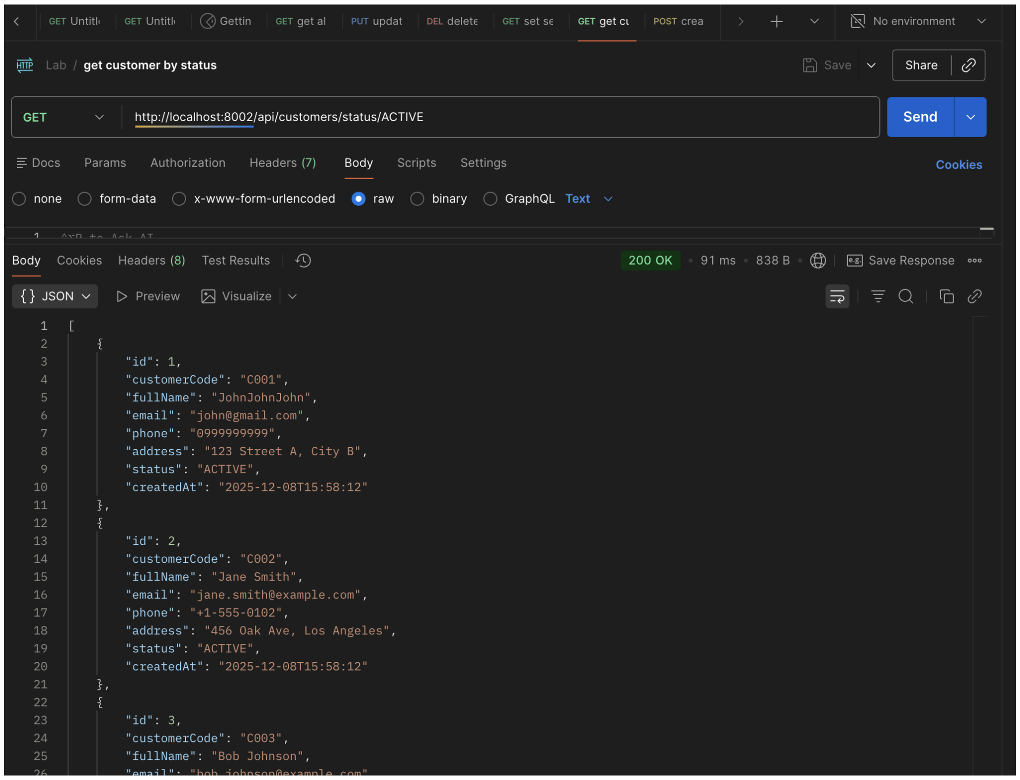
```java
@Override
public List<CustomerResponseDTO> getCustomersByStatus(String status) {
    return customerRepository.findByStatus(status).stream().map(this::convertToResponseDTO)
            .collect(Collectors.toList());
}
```

- New service method



```java
@Override
public List<CustomerResponseDTO> getCustomersByStatus(String status) {
    CustomerStatus enumStatus = CustomerStatus.valueOf(status.toUpperCase());
    return customerRepository.findByStatus(enumStatus)
            .stream()
            .map(this::convertToResponseDTO)
            .collect(Collectors.toList());
}
```

Result of the bug resolve:



```
GET http://localhost:8002/api/customers/status/ACTIVE
```

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

Body Cookies Headers (8) Test Results — 200 OK • 91 ms • 838 B — Save Response

JSON Preview Visualize

```
[
    {
        "id": 1,
        "customerCode": "C001",
        "fullName": "JohnJohnJohn",
        "email": "john@gmail.com",
        "phone": "0999999999",
        "address": "123 Street A, City B",
        "status": "ACTIVE",
        "createdAt": "2025-12-08T15:58:12"
    },
    {
        "id": 2,
        "customerCode": "C002",
        "fullName": "Jane Smith",
        "email": "jane.smith@example.com",
        "phone": "+1-555-0102",
        "address": "456 Oak Ave, Los Angeles",
        "status": "ACTIVE",
        "createdAt": "2025-12-08T15:58:12"
    },
    {
        "id": 3,
        "customerCode": "C003",
        "fullName": "Bob Johnson",
        "email": "bob.johnson@example.com",
```

Flow:

- Controller (getCustomersByStatus()), receives path variable string "ACTIVE" or "INACTIVE", calls customerService.getCustomersByStatus(status)
- Service (getCustomersByStatus()) converts input string to enum, calls customerRepository.findByStatus(enumStatus), converts each entity to DTO and returns the list
- Repository (findByStatus(CustomerStatus status)): JPA generates query based on enum column in DB, returns matching customers
- Controller returns 200 OK with filtered results