International University

School of Computer Science and Engineering

Web Application Development Laboratory

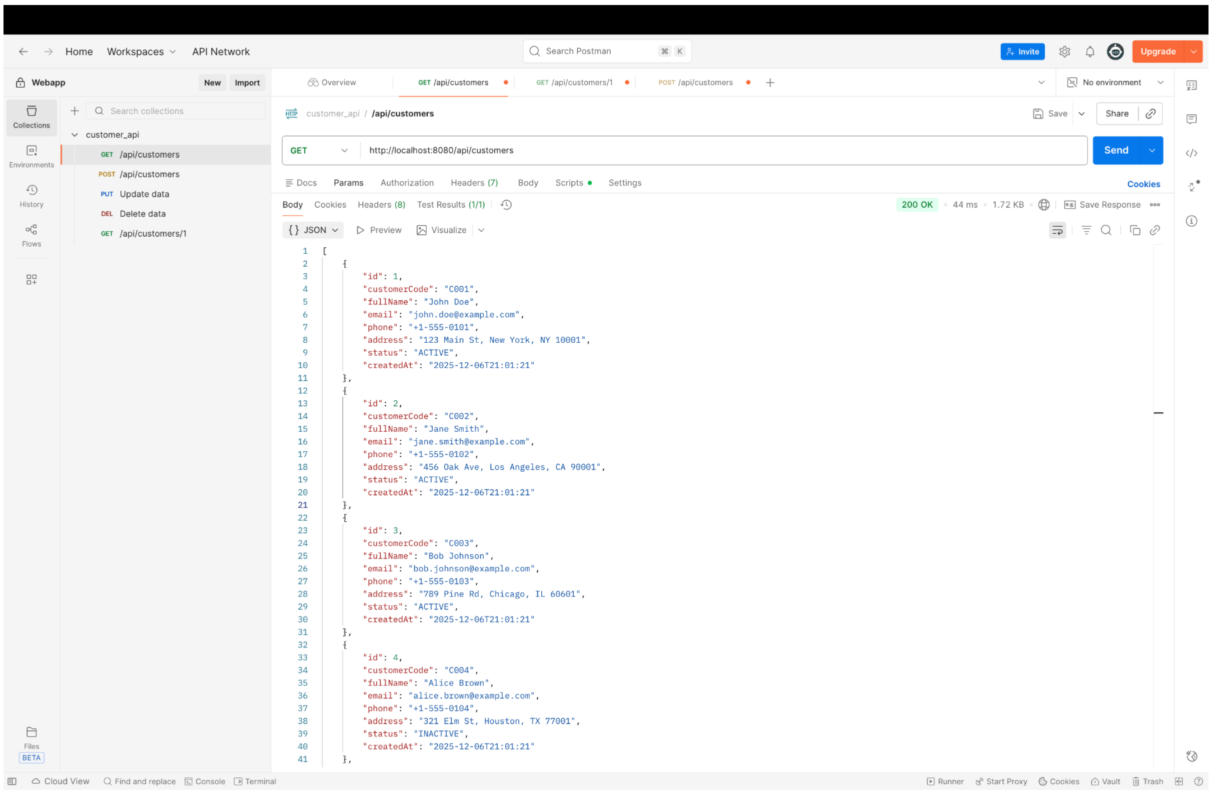
IT093IU

Lab #8 Practice

Submitted by

Nguyễn Hồng Ngọc Hân - ITCSIU22229

Test 1: GET All Customers



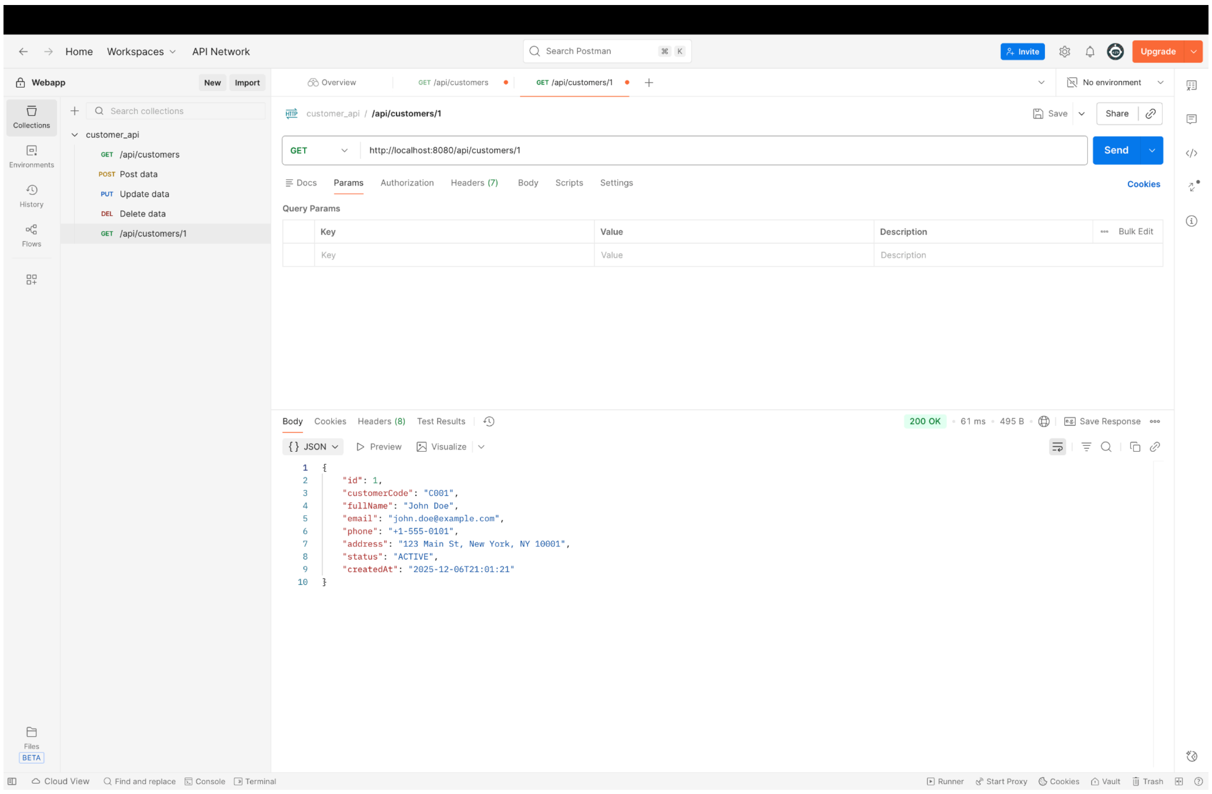
GET http://localhost:8080/api/customers

200 OK · 44 ms · 1.72 KB

```
[
    {
        "id": 1,
        "customerCode": "C001",
        "fullName": "John Doe",
        "email": "john.doe@example.com",
        "phone": "+1-555-0101",
        "address": "123 Main St, New York, NY 10001",
        "status": "ACTIVE",
        "createdAt": "2025-12-06T21:01:21"
    },
    {
        "id": 2,
        "customerCode": "C002",
        "fullName": "Jane Smith",
        "email": "jane.smith@example.com",
        "phone": "+1-555-0102",
        "address": "456 Oak Ave, Los Angeles, CA 90001",
        "status": "ACTIVE",
        "createdAt": "2025-12-06T21:01:21"
    },
    {
        "id": 3,
        "customerCode": "C003",
        "fullName": "Bob Johnson",
        "email": "bob.johnson@example.com",
        "phone": "+1-555-0103",
        "address": "789 Pine Rd, Chicago, IL 60601",
        "status": "ACTIVE",
        "createdAt": "2025-12-06T21:01:21"
    },
    {
        "id": 4,
        "customerCode": "C004",
        "fullName": "Alice Brown",
        "email": "alice.brown@example.com",
        "phone": "+1-555-0104",
        "address": "321 Elm St, Houston, TX 77001",
        "status": "INACTIVE",
        "createdAt": "2025-12-06T21:01:21"
    },
```

1. **Controller: CustomerRestController.getAllCustomers() Receives GET /api/customers.**
2. **Service: CustomerServiceImpl.getAllCustomers() calls the repository to fetch all entities.**
3. **Repository: CustomerRepository.findAll() executes SQL to retrieve all records from the customers table.**
4. **Returns List<Customer>.**
5. **Service: CustomerServiceImpl streams the List<Customer> and maps each entity to a CustomerResponseDTO using helper methods.**
6. **Controller CustomerRestController returns the List<CustomerResponseDTO> with status 200** OK.

Test 2: GET Customer by ID



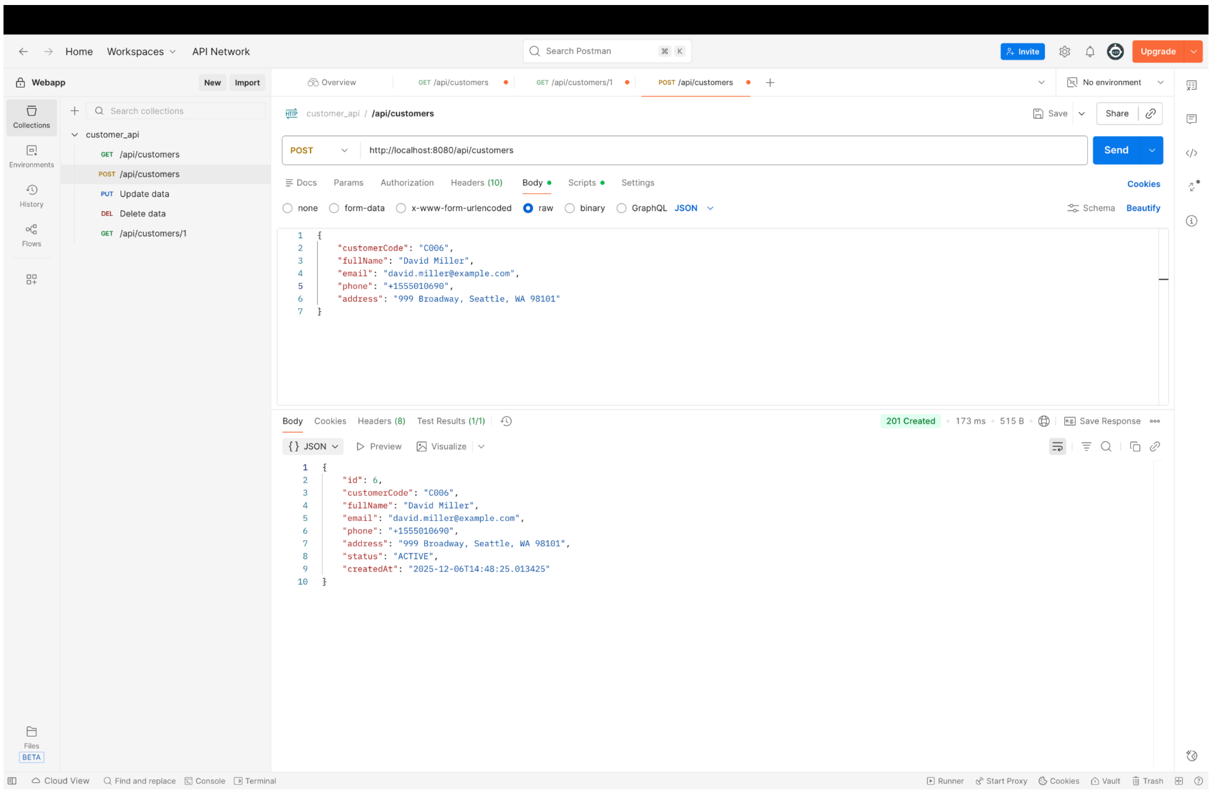
GET http://localhost:8080/api/customers/1

200 OK • 61 ms • 495 B

```
{
    "id": 1,
    "customerCode": "C001",
    "fullName": "John Doe",
    "email": "john.doe@example.com",
    "phone": "+1-555-0101",
    "address": "123 Main St, New York, NY 10001",
    "status": "ACTIVE",
    "createdAt": "2025-12-06T21:01:21"
}
```

The request to GET /api/customers/{id} starts at the Controller, where the path variable {id} is extracted. The Controller calls the Service's getCustomerById(id) method. The Service immediately calls the Repository's findById(id) method, which executes a SELECT query in the database. The Repository returns an Optional<Customer>. The Service unwraps the result (or throws a ResourceNotFoundException if empty), converts the resulting Customer entity into a CustomerResponseDTO, and passes it back. Finally, the Controller returns the CustomerResponseDTO with a 200 OK status.

Test 3: POST Create Customer


POST http://localhost:8080/api/customers

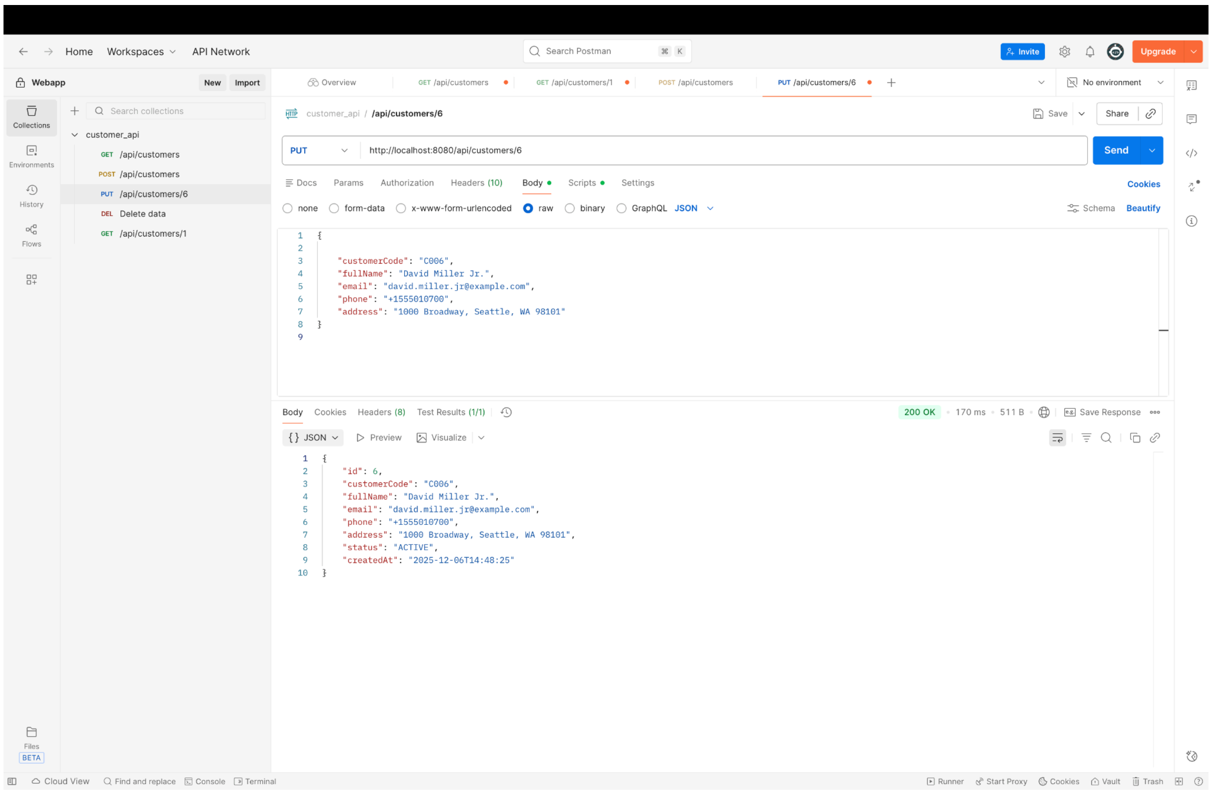
```
{
    "customerCode": "C006",
    "fullName": "David Miller",
    "email": "david.miller@example.com",
    "phone": "+1555010690",
    "address": "999 Broadway, Seattle, WA 98101"
}
```

201 Created · 173 ms · 515 B

```
{
    "id": 6,
    "customerCode": "C006",
    "fullName": "David Miller",
    "email": "david.miller@example.com",
    "phone": "+1555010690",
    "address": "999 Broadway, Seattle, WA 98101",
    "status": "ACTIVE",
    "createdAt": "2025-12-06T14:48:25.013425"
}
```

The POST /api/customers request is received by the Controller and is checked for Validation (@Valid). After successful validation, the Service executes the business logic: checking for duplicate customer code and email by calling existsByCustomerCode() and existsByEmail() in the Repository. If no duplicates are found, the Service converts the CustomerRequestDTO into a Customer entity, and the Repository performs save() (INSERT into DB). The Service converts the saved entity into a CustomerResponseDTO, and the Controller returns it with a 201 Created status.

Test 4: PUT Update Customer



PUT http://localhost:8080/api/customers/6

Body (raw, JSON):

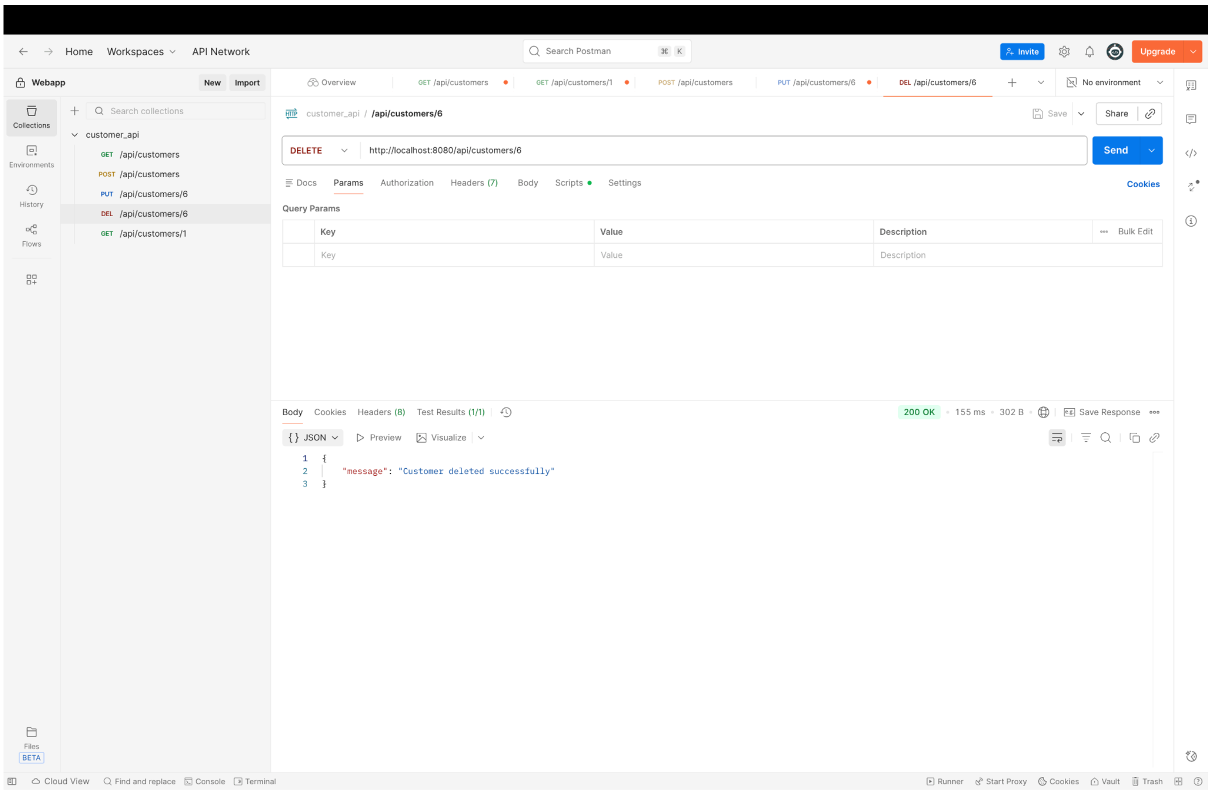
```
{

    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+1555010700",
    "address": "1000 Broadway, Seattle, WA 98101"
}

```

Response: 200 OK • 170 ms • 511 B

```
{
    "id": 6,
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+1555010700",
    "address": "1000 Broadway, Seattle, WA 98101",
    "status": "ACTIVE",
    "createdAt": "2025-12-06T14:48:25"
}
```

The PUT /api/customers/{id} request is received by the Controller, and the incoming CustomerRequestDTO undergoes Validation (@Valid). Upon success, the Controller calls the Service's updateCustomer(id, requestDTO) method. The Service first calls the Repository's findById(id) to retrieve the existing customer entity, ensuring it exists. The Service performs a duplicate check on the new email address, ensuring the new email doesn't belong to any other customer.
If checks pass, the Service updates the fields of the retrieved entity.
The Service then calls the Repository's save(existingCustomer) method, which executes an UPDATE query. The Service converts the updated entity to a CustomerResponseDTO, and the Controller returns it with a 200 OK status.

Test 5: DELETE Customer



customer_api
- GET /api/customers
- POST /api/customers
- PUT /api/customers/6
- DEL /api/customers/6
- GET /api/customers/1

customer_api / /api/customers/6

DELETE http://localhost:8080/api/customers/6

Query Params

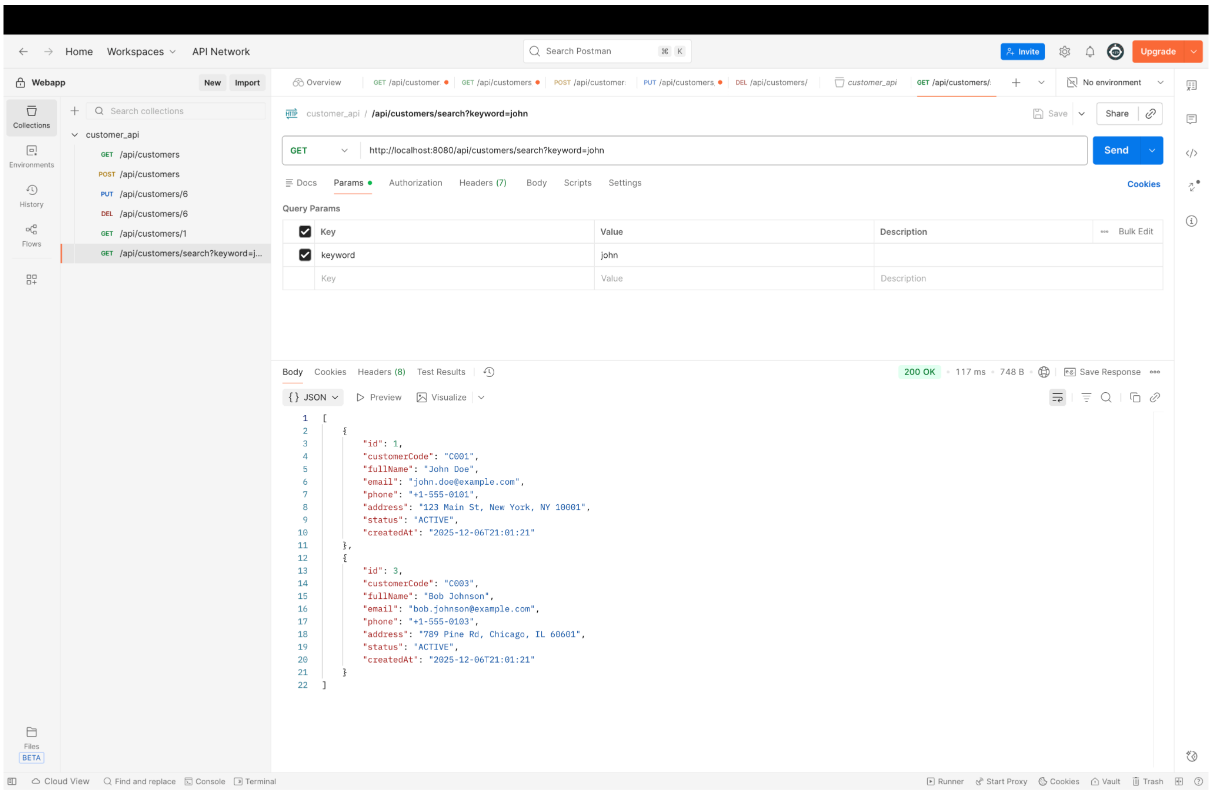
Key	Value	Description
Key	Value	Description

200 OK · 155 ms · 302 B

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

The DELETE /api/customers/{id} request is received by the Controller, which calls the Service's deleteCustomer(id) method. The Service first calls the Repository's existsById(id) to confirm the resource exists.
If the customer exists, the Service calls the Repository's deleteById(id) method, which executes a DELETE query. If the deletion is successful (or the operation completes), the Controller constructs a simple success JSON map and returns it with a 200 OK status.

Test 6: Search Customers


GET http://localhost:8080/api/customers/search?keyword=john

Query Params

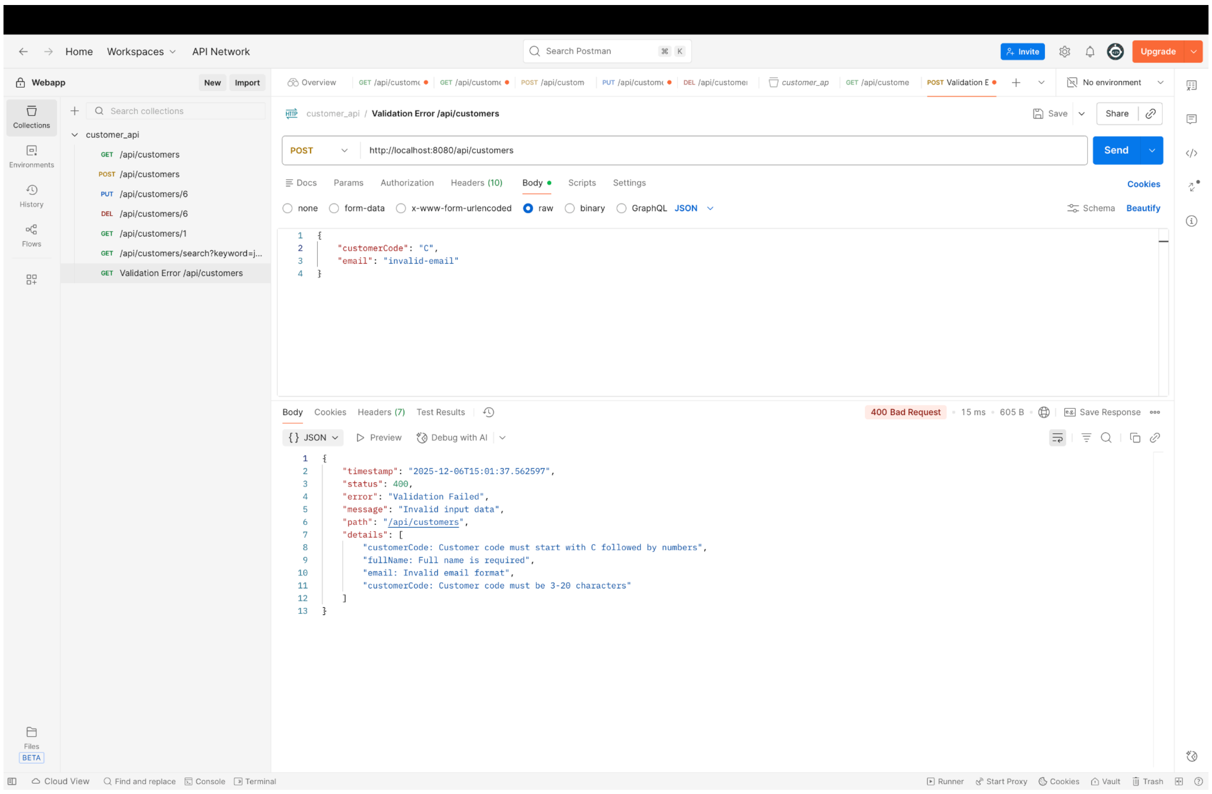
Key	Value	Description
keyword	john	

200 OK · 117 ms · 748 B

```
[
    {
        "id": 1,
        "customerCode": "C001",
        "fullName": "John Doe",
        "email": "john.doe@example.com",
        "phone": "+1-555-0101",
        "address": "123 Main St, New York, NY 10001",
        "status": "ACTIVE",
        "createdAt": "2025-12-06T21:01:21"
    },
    {
        "id": 3,
        "customerCode": "C003",
        "fullName": "Bob Johnson",
        "email": "bob.johnson@example.com",
        "phone": "+1-555-0103",
        "address": "789 Pine Rd, Chicago, IL 60601",
        "status": "ACTIVE",
        "createdAt": "2025-12-06T21:01:21"
    }
]
```

The GET /api/customers/search?keyword={keyword} request is received by the Controller, which extracts the keyword request parameter. The Controller calls the Service's searchCustomers(keyword) method. The Service immediately calls the Repository's searchCustomers(keyword) method. The Repository executes a specialized @Query (JPQL/HQL) that searches across multiple fields (fullName, email, customerCode) using LOWER(..) and LIKE %keyword% for case-insensitive partial matching. The Repository returns a List<Customer>. The Service converts this list to a List<CustomerResponseDTO>, and the Controller returns the results with a 200 OK status.

Test 7: Validation Error



customer_api / Validation Error /api/customers

POST http://localhost:8080/api/customers

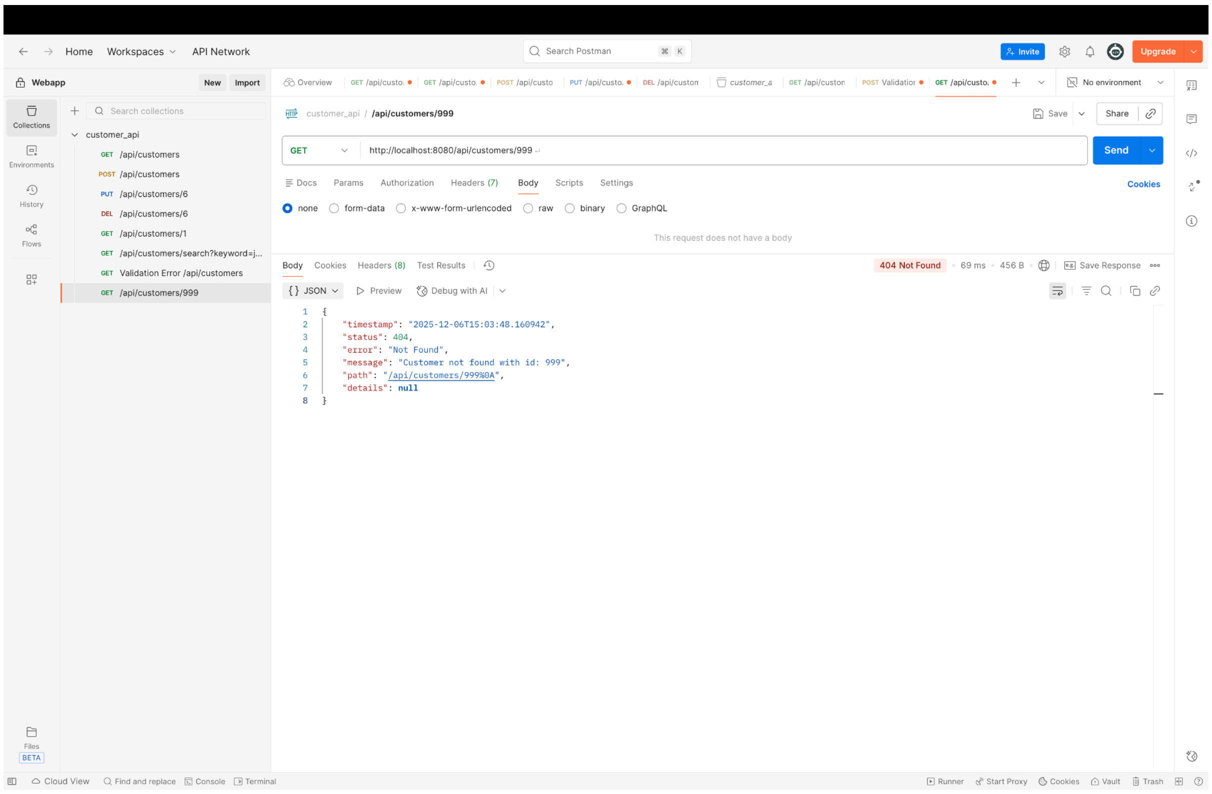
```
{
    "customerCode": "C",
    "email": "invalid-email"
}
```

400 Bad Request · 15 ms · 605 B

```
{
    "timestamp": "2025-12-06T15:01:37.562597",
    "status": 400,
    "error": "Validation Failed",
    "message": "Invalid input data",
    "path": "/api/customers",
    "details": [
        "customerCode: Customer code must start with C followed by numbers",
        "fullName: Full name is required",
        "email: Invalid email format",
        "customerCode: Customer code must be 3-20 characters"
    ]
}
```

When the POST /api/customers request contains invalid data (e.g., missing fullName), the validation (@Valid) fails immediately at the Controller. Instead of calling the Service, Spring throws a MethodArgumentNotValidException. This exception is caught by the GlobalExceptionHandler. The Handler extracts all detailed errors from the BindingResult, constructs an ErrorResponseDTO containing the error details, and returns it to the client with a 400 Bad Request status.

Test 8: Resource Not Found



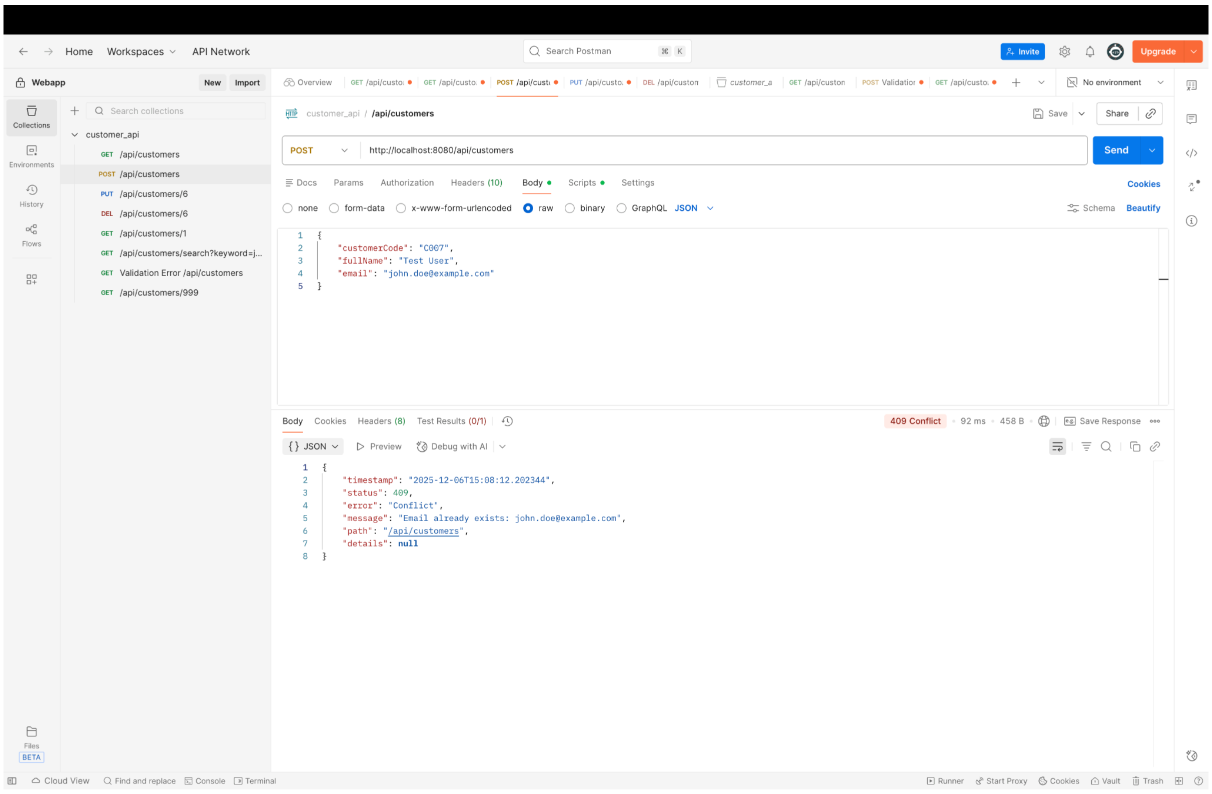
GET http://localhost:8080/api/customers/999

404 Not Found · 69 ms · 456 B

```
{
    "timestamp": "2025-12-06T15:03:48.160942",
    "status": 404,
    "error": "Not Found",
    "message": "Customer not found with id: 999",
    "path": "/api/customers/999%0A",
    "details": null
}
```

When the GET /api/customers/999 request (non-existent ID) is sent, the Service calls findById(999L) on the Repository. The Repository returns an Optional.empty(). The Service checks the result and throws a ResourceNotFoundException. This exception is caught by the GlobalExceptionHandler, which then creates an ErrorResponseDTO with the message "Customer not found..." and returns it with a 404 Not Found status.

Test 9: Duplicate Resource



customer_api / /api/customers

POST http://localhost:8080/api/customers

```
{
    "customerCode": "C007",
    "fullName": "Test User",
    "email": "john.doe@example.com"
}
```

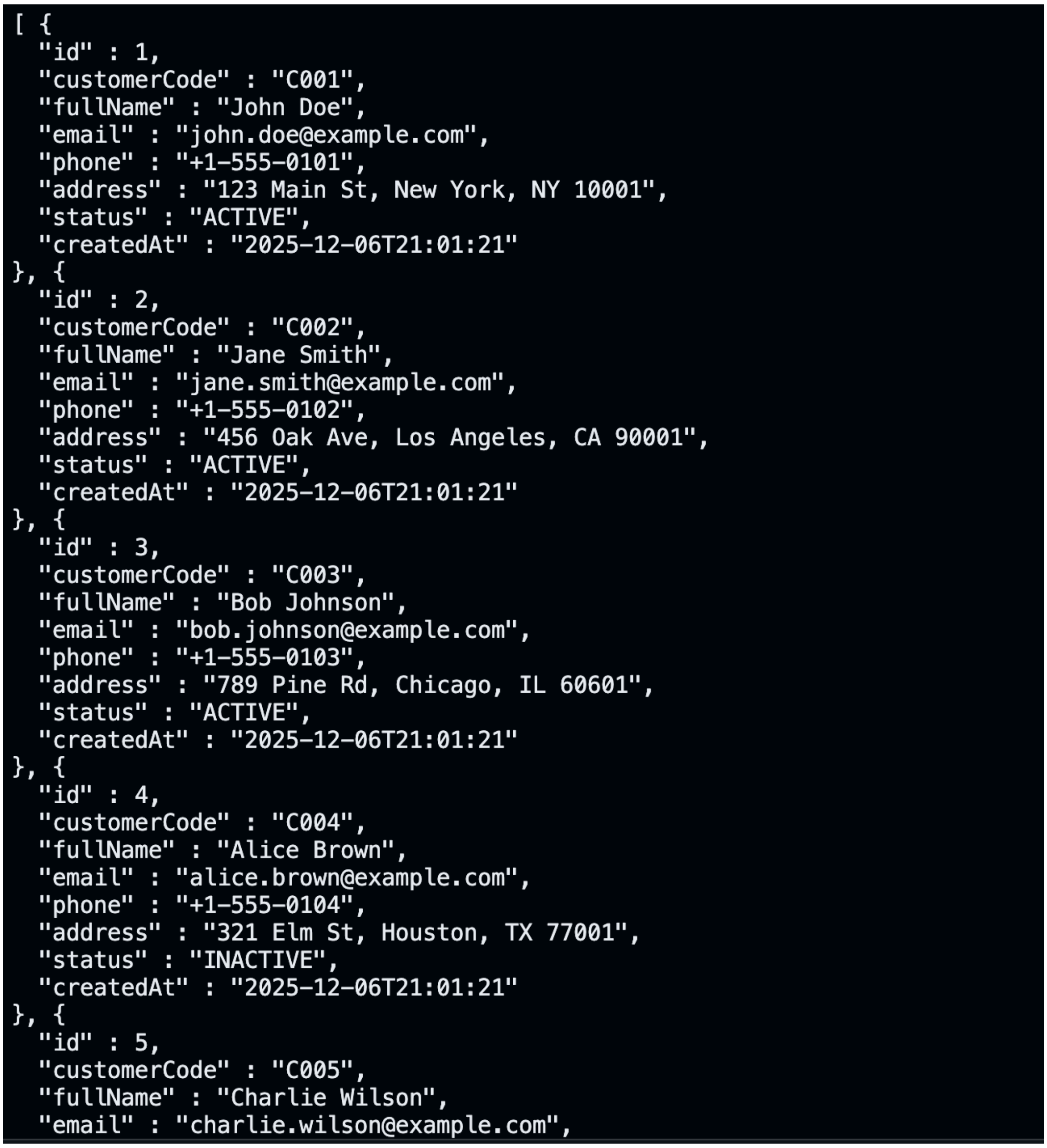
409 Conflict · 92 ms · 458 B

```
{
    "timestamp": "2025-12-06T15:08:12.202344",
    "status": 409,
    "error": "Conflict",
    "message": "Email already exists: john.doe@example.com",
    "path": "/api/customers",
    "details": null
}
```

When the POST /api/customers request contains an existing email, basic validation succeeds. The Service calls existsByEmail() on the Repository, and the Repository returns true. The Service immediately throws a DuplicateResourceException. This exception is caught by the GlobalExceptionHandler, which constructs an ErrorResponseDTO with a conflict message and returns it with a 409 Conflict status.

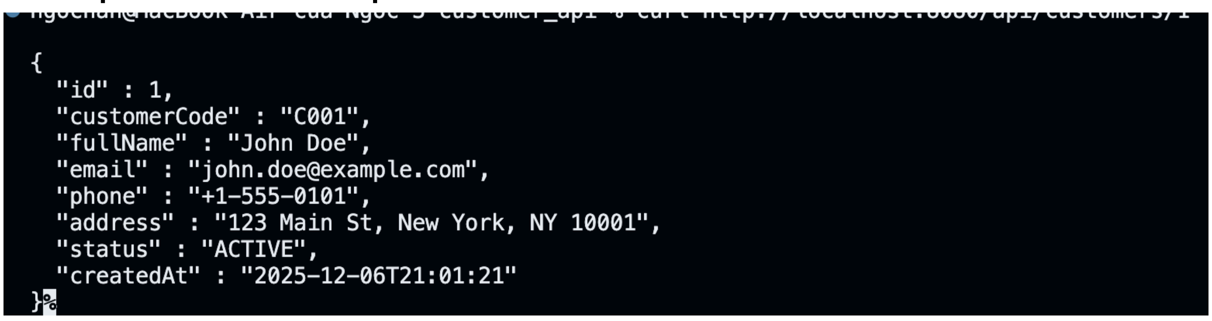
Using cURL (Command Line)

# GET all customers
curl http://localhost:8080/api/customers



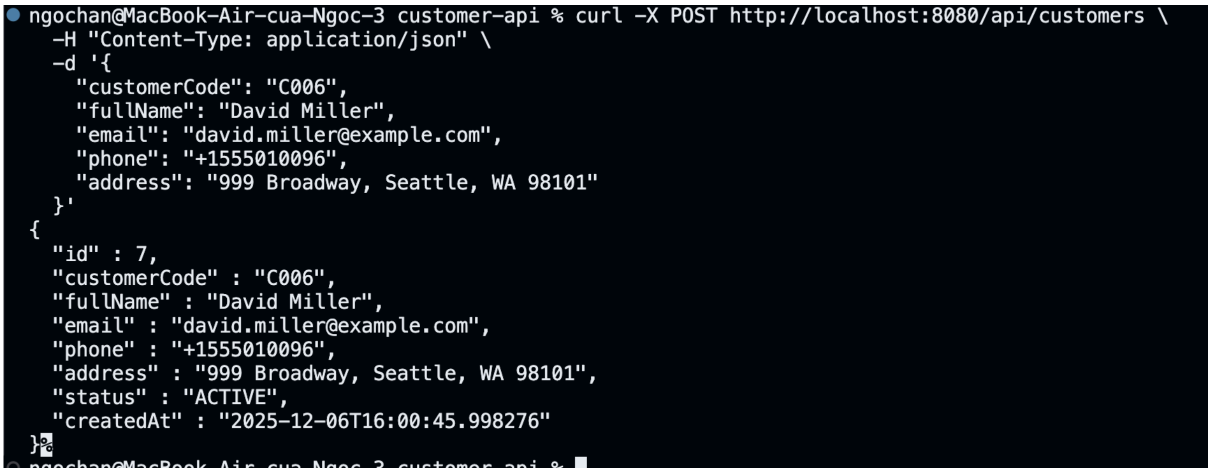
```
[ {
  "id" : 1,
  "customerCode" : "C001",
  "fullName" : "John Doe",
  "email" : "john.doe@example.com",
  "phone" : "+1-555-0101",
  "address" : "123 Main St, New York, NY 10001",
  "status" : "ACTIVE",
  "createdAt" : "2025-12-06T21:01:21"
}, {
  "id" : 2,
  "customerCode" : "C002",
  "fullName" : "Jane Smith",
  "email" : "jane.smith@example.com",
  "phone" : "+1-555-0102",
  "address" : "456 Oak Ave, Los Angeles, CA 90001",
  "status" : "ACTIVE",
  "createdAt" : "2025-12-06T21:01:21"
}, {
  "id" : 3,
  "customerCode" : "C003",
  "fullName" : "Bob Johnson",
  "email" : "bob.johnson@example.com",
  "phone" : "+1-555-0103",
  "address" : "789 Pine Rd, Chicago, IL 60601",
  "status" : "ACTIVE",
  "createdAt" : "2025-12-06T21:01:21"
}, {
  "id" : 4,
  "customerCode" : "C004",
  "fullName" : "Alice Brown",
  "email" : "alice.brown@example.com",
  "phone" : "+1-555-0104",
  "address" : "321 Elm St, Houston, TX 77001",
  "status" : "INACTIVE",
  "createdAt" : "2025-12-06T21:01:21"
}, {
  "id" : 5,
  "customerCode" : "C005",
  "fullName" : "Charlie Wilson",
  "email" : "charlie.wilson@example.com",
```

# GET customer by ID
curl http://localhost:8080/api/customers/1



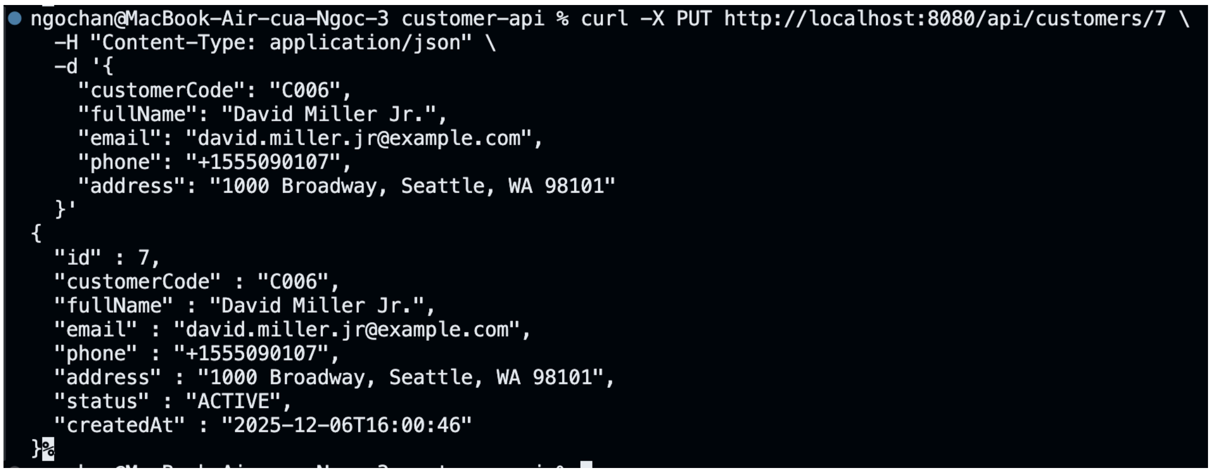
```
ngocnan@MacBook-Air-cua-Ngoc-3 customer_api % curl http://localhost:8080/api/customers/1

 {
   "id" : 1,
   "customerCode" : "C001",
   "fullName" : "John Doe",
   "email" : "john.doe@example.com",
   "phone" : "+1-555-0101",
   "address" : "123 Main St, New York, NY 10001",
   "status" : "ACTIVE",
   "createdAt" : "2025-12-06T21:01:21"
 }%
```

# POST create customer
curl -X POST http://localhost:8080/api/customers \
  -H "Content-Type: application/json" \
  -d '{
    "customerCode": "C006",
    "fullName": "David Miller",
    "email": "david.miller@example.com",
    "phone": "+1-555-0106",
    "address": "999 Broadway, Seattle, WA 98101"
  }'



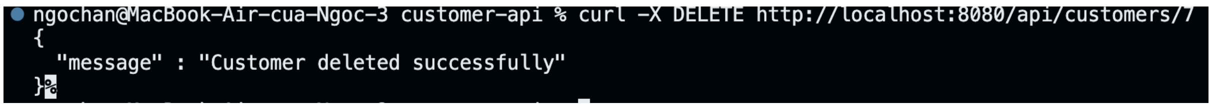
```
● ngochan@MacBook-Air-cua-Ngoc-3 customer-api % curl -X POST http://localhost:8080/api/customers \
    -H "Content-Type: application/json" \
    -d '{
      "customerCode": "C006",
      "fullName": "David Miller",
      "email": "david.miller@example.com",
      "phone": "+1555010096",
      "address": "999 Broadway, Seattle, WA 98101"
    }'
  {
    "id" : 7,
    "customerCode" : "C006",
    "fullName" : "David Miller",
    "email" : "david.miller@example.com",
    "phone" : "+1555010096",
    "address" : "999 Broadway, Seattle, WA 98101",
    "status" : "ACTIVE",
    "createdAt" : "2025-12-06T16:00:45.998276"
  }%
○ ngochan@MacBook-Air-cua-Ngoc-3 customer-api %
```

# PUT update customer
curl -X PUT http://localhost:8080/api/customers/6 \
  -H "Content-Type: application/json" \
  -d '{
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+1-555-0107",
    "address": "1000 Broadway, Seattle, WA 98101"
  }'



```
● ngochan@MacBook-Air-cua-Ngoc-3 customer-api % curl -X PUT http://localhost:8080/api/customers/7 \
    -H "Content-Type: application/json" \
    -d '{
      "customerCode": "C006",
      "fullName": "David Miller Jr.",
      "email": "david.miller.jr@example.com",
      "phone": "+1555090107",
      "address": "1000 Broadway, Seattle, WA 98101"
    }'
  {
    "id" : 7,
    "customerCode" : "C006",
    "fullName" : "David Miller Jr.",
    "email" : "david.miller.jr@example.com",
    "phone" : "+1555090107",
    "address" : "1000 Broadway, Seattle, WA 98101",
    "status" : "ACTIVE",
    "createdAt" : "2025-12-06T16:00:46"
  }%
```

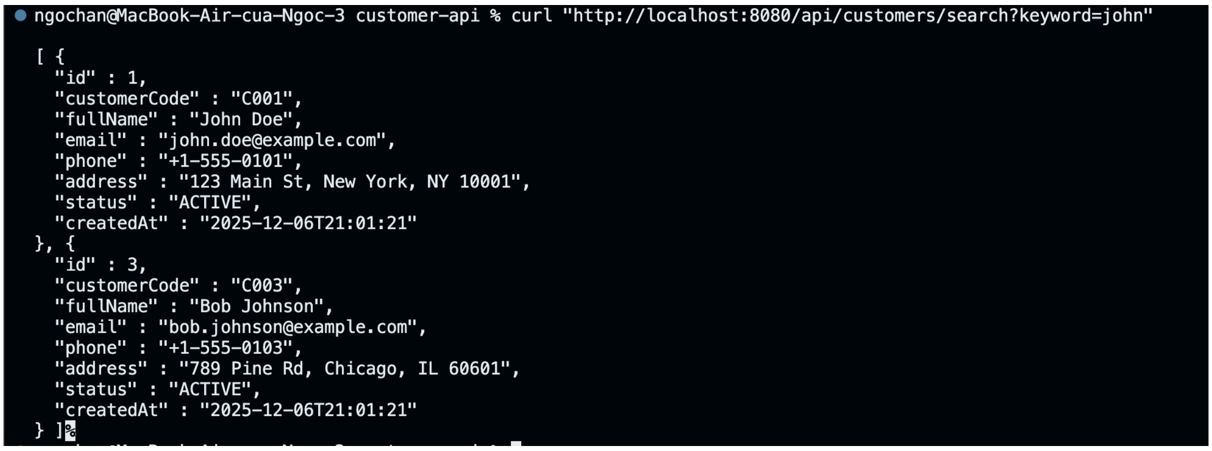
# DELETE customer
curl -X DELETE http://localhost:8080/api/customers/6



```
ngochan@MacBook-Air-cua-Ngoc-3 customer-api % curl -X DELETE http://localhost:8080/api/customers/7
{
  "message" : "Customer deleted successfully"
}%
```

# Search customers

curl "http://localhost:8080/api/customers/search?keyword=john"



```
ngochan@MacBook-Air-cua-Ngoc-3 customer-api % curl "http://localhost:8080/api/customers/search?keyword=john"

[ {
  "id" : 1,
  "customerCode" : "C001",
  "fullName" : "John Doe",
  "email" : "john.doe@example.com",
  "phone" : "+1-555-0101",
  "address" : "123 Main St, New York, NY 10001",
  "status" : "ACTIVE",
  "createdAt" : "2025-12-06T21:01:21"
}, {
  "id" : 3,
  "customerCode" : "C003",
  "fullName" : "Bob Johnson",
  "email" : "bob.johnson@example.com",
  "phone" : "+1-555-0103",
  "address" : "789 Pine Rd, Chicago, IL 60601",
  "status" : "ACTIVE",
  "createdAt" : "2025-12-06T21:01:21"
} ]%
```