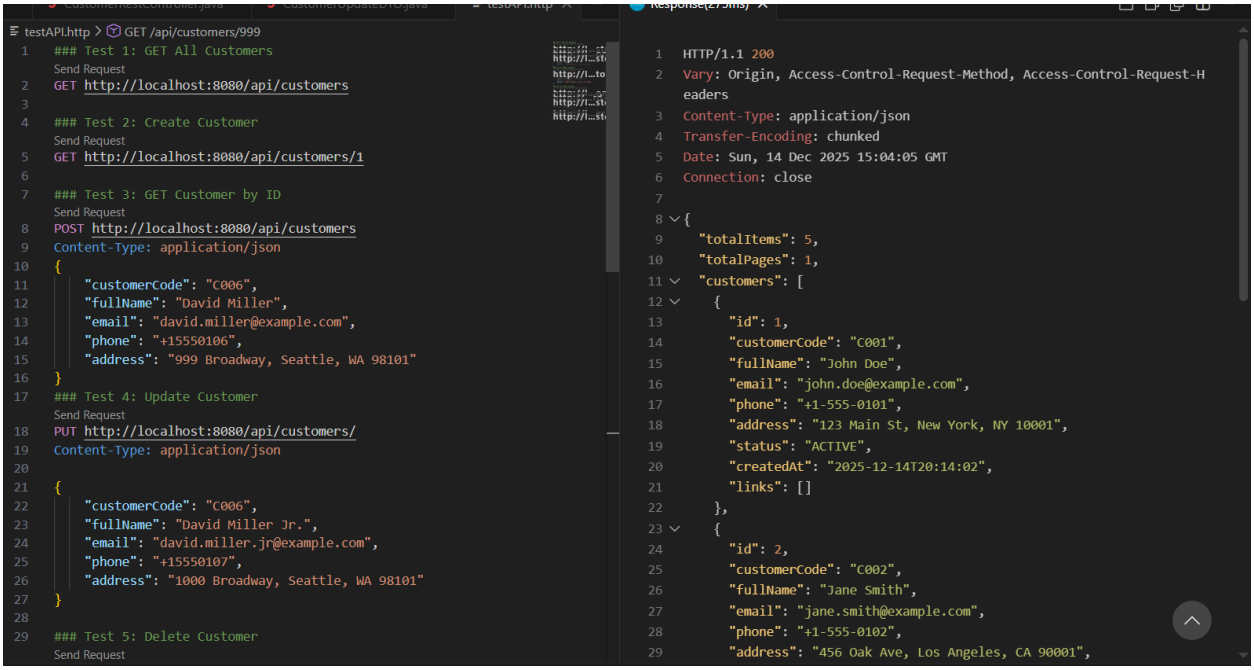
Test 1: GET All Customers

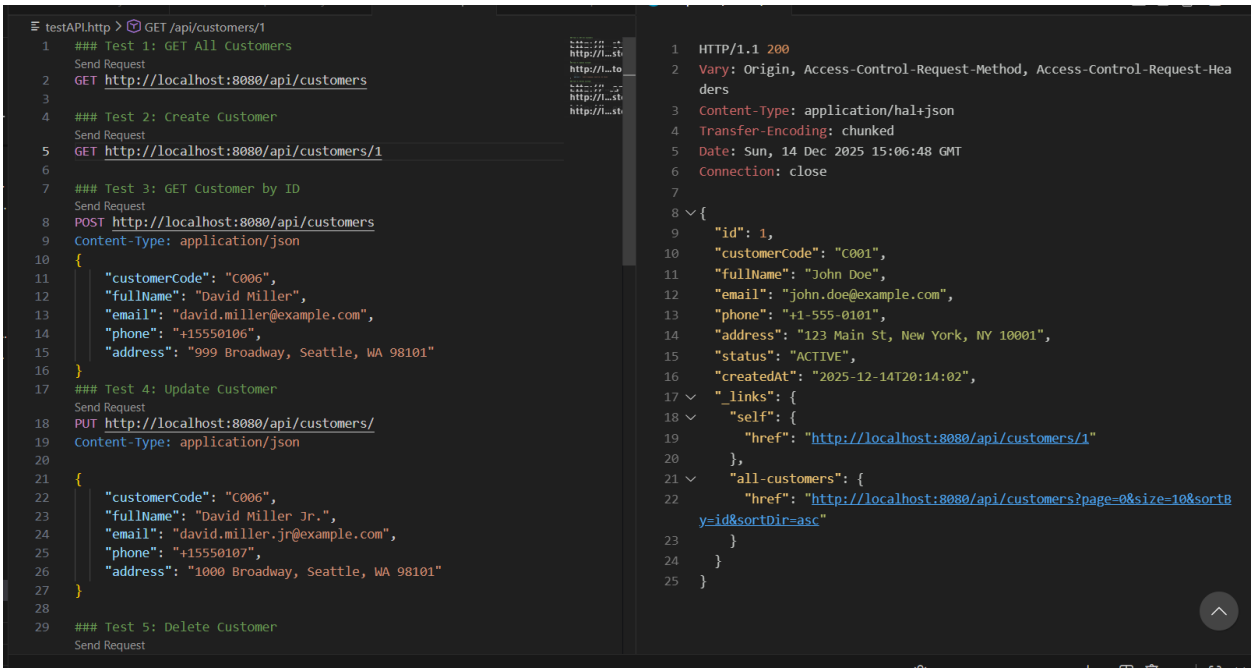


```
testAPI.http > GET /api/customers/999

### Test 1: GET All Customers
Send Request
GET http://localhost:8080/api/customers

### Test 2: Create Customer
Send Request
GET http://localhost:8080/api/customers/1

### Test 3: GET Customer by ID
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json
{
    "customerCode": "C006",
    "fullName": "David Miller",
    "email": "david.miller@example.com",
    "phone": "+15550106",
    "address": "999 Broadway, Seattle, WA 98101"
}
### Test 4: Update Customer
Send Request
PUT http://localhost:8080/api/customers/
Content-Type: application/json

{
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+15550107",
    "address": "1000 Broadway, Seattle, WA 98101"
}

### Test 5: Delete Customer
Send Request
```

```
HTTP/1.1 200
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:04:05 GMT
Connection: close

{
  "totalItems": 5,
  "totalPages": 1,
  "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": "2025-12-14T20:14:02",
      "links": []
    },
    {
      "id": 2,
      "customerCode": "C002",
      "fullName": "Jane Smith",
      "email": "jane.smith@example.com",
      "phone": "+1-555-0102",
      "address": "456 Oak Ave, Los Angeles, CA 90001",
```

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 O**K.

Test 2: GET Customer by ID

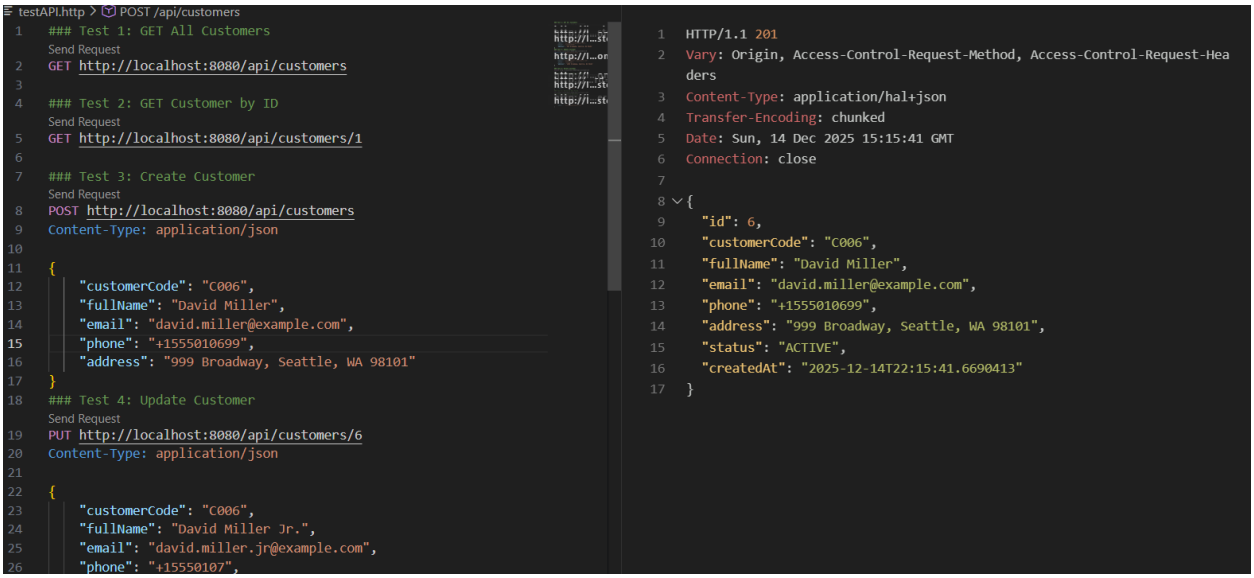


```
testAPI.http > GET /api/customers/1
### Test 1: GET All Customers
Send Request
GET http://localhost:8080/api/customers

### Test 2: Create Customer
Send Request
GET http://localhost:8080/api/customers/1

### Test 3: GET Customer by ID
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json
{
    "customerCode": "C006",
    "fullName": "David Miller",
    "email": "david.miller@example.com",
    "phone": "+15550106",
    "address": "999 Broadway, Seattle, WA 98101"
}
### Test 4: Update Customer
Send Request
PUT http://localhost:8080/api/customers/
Content-Type: application/json

{
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+15550107",
    "address": "1000 Broadway, Seattle, WA 98101"
}

### Test 5: Delete Customer
Send Request
```

```
HTTP/1.1 200
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-Type: application/hal+json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:06:48 GMT
Connection: close

{
  "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": "2025-12-14T20:14:02",
  "_links": {
    "self": {
      "href": "http://localhost:8080/api/customers/1"
    },
    "all-customers": {
      "href": "http://localhost:8080/api/customers?page=0&size=10&sortBy=id&sortDir=asc"
    }
  }
}
```

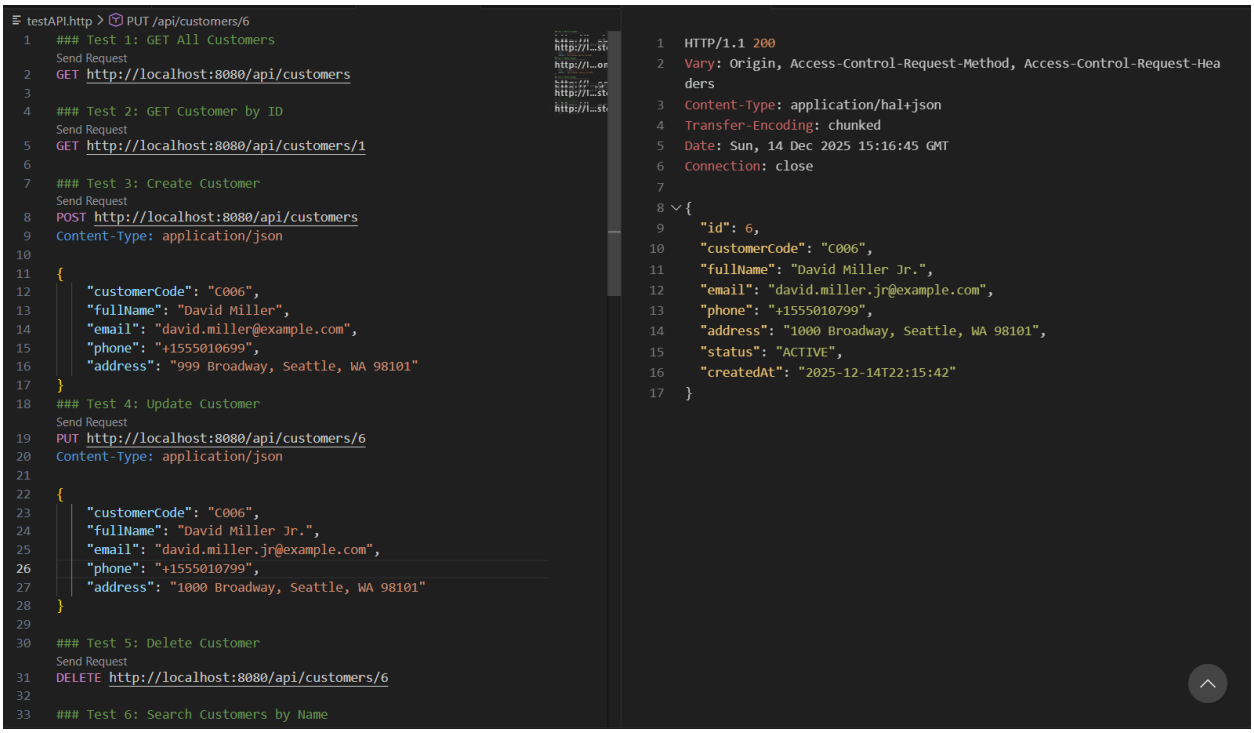
The request to GET /api/customers/{id} starts at the Controller, where the path variable {id} is extracted. The Controller calls the Service's getCustomerById(id) method. The Service immediately calls the Repository's findById(id) method, which executes a SELECT query in the database. The Repository 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



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



testAPI.http > PUT /api/customers/6

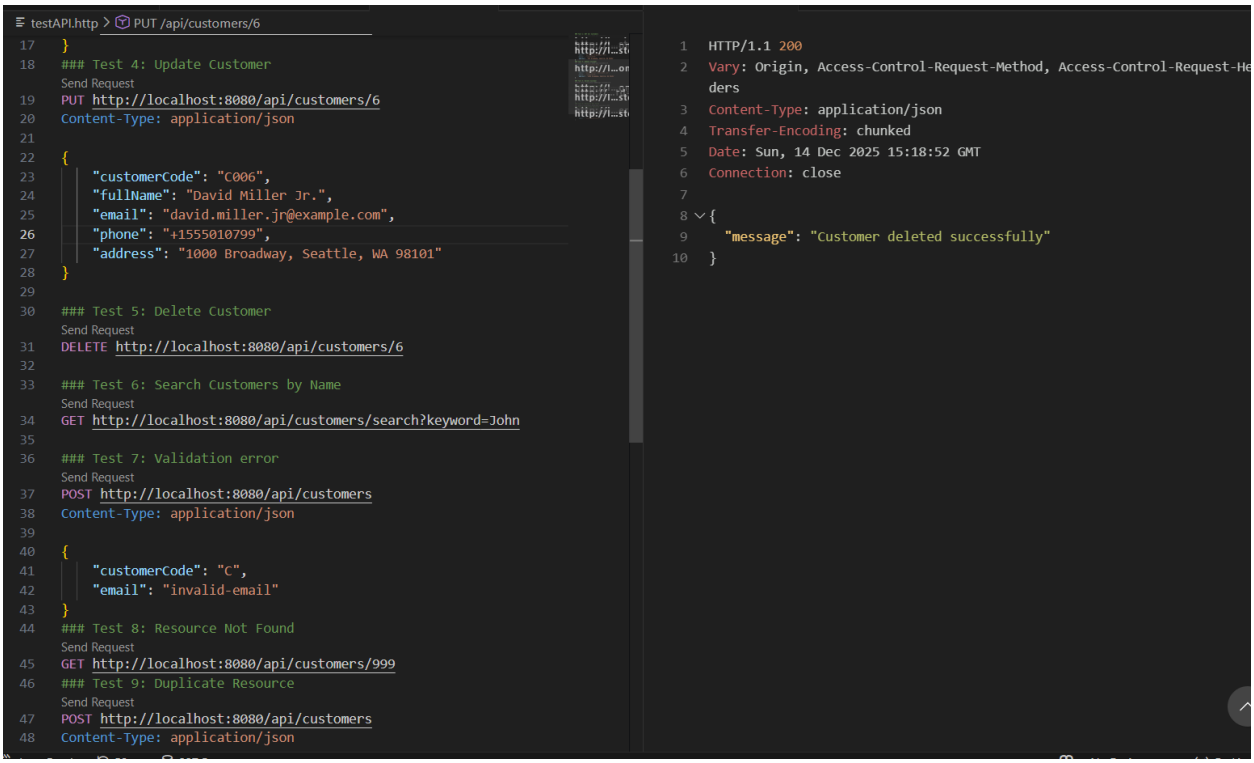
```
### Test 1: GET All Customers
Send Request
GET http://localhost:8080/api/customers

### Test 2: GET Customer by ID
Send Request
GET http://localhost:8080/api/customers/1

### Test 3: Create Customer
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json

{
    "customerCode": "C006",
    "fullName": "David Miller",
    "email": "david.miller@example.com",
    "phone": "+1555010699",
    "address": "999 Broadway, Seattle, WA 98101"
}
### Test 4: Update Customer
Send Request
PUT http://localhost:8080/api/customers/6
Content-Type: application/json

{
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+1555010799",
    "address": "1000 Broadway, Seattle, WA 98101"
}

### Test 5: Delete Customer
Send Request
DELETE http://localhost:8080/api/customers/6

### Test 6: Search Customers by Name
```

```
HTTP/1.1 200
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-Type: application/hal+json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:16:45 GMT
Connection: close

{
  "id": 6,
  "customerCode": "C006",
  "fullName": "David Miller Jr.",
  "email": "david.miller.jr@example.com",
  "phone": "+1555010799",
  "address": "1000 Broadway, Seattle, WA 98101",
  "status": "ACTIVE",
  "createdAt": "2025-12-14T22:15:42"
}
```

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.

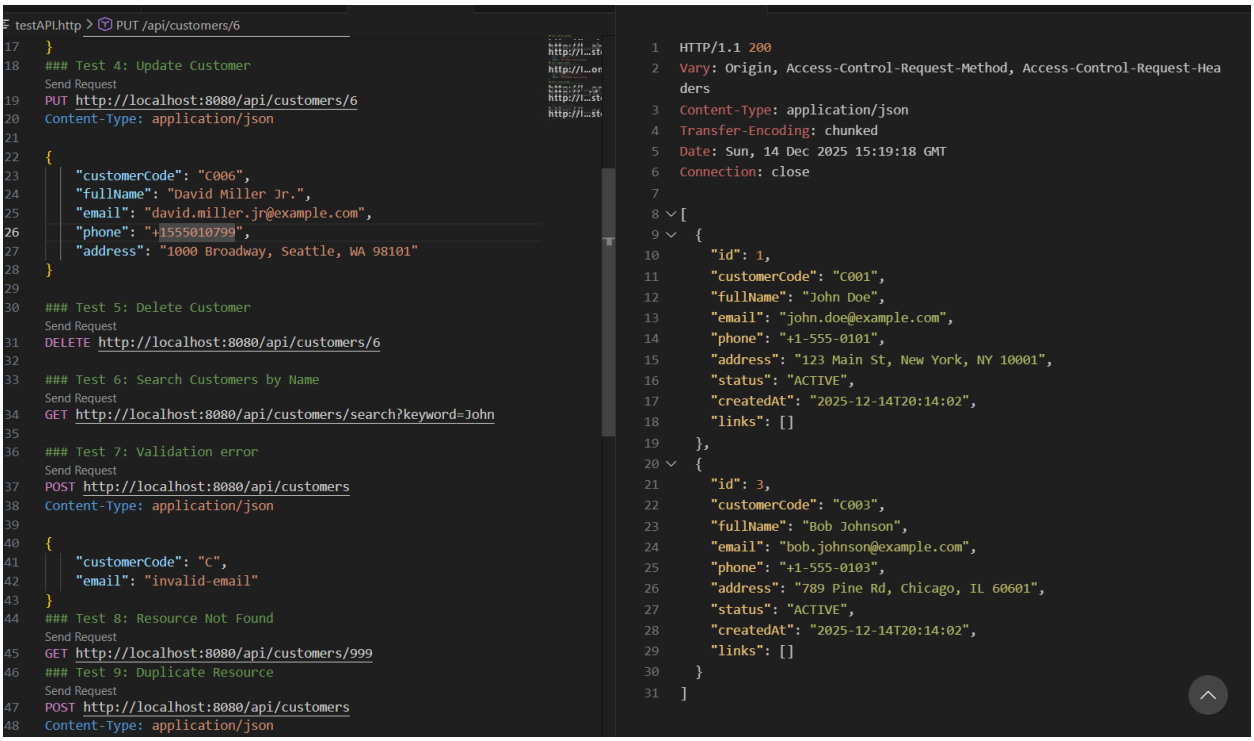
If checks pass, the Service updates the fields of the retrieved entity. The Service then calls the Repository's save(existingCustomer) method, which executes an UPDATE query. The Service converts the updated entity to a CustomerResponseDTO, and the Controller returns it with a 200 OK status.

Test 5: DELETE Customer



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

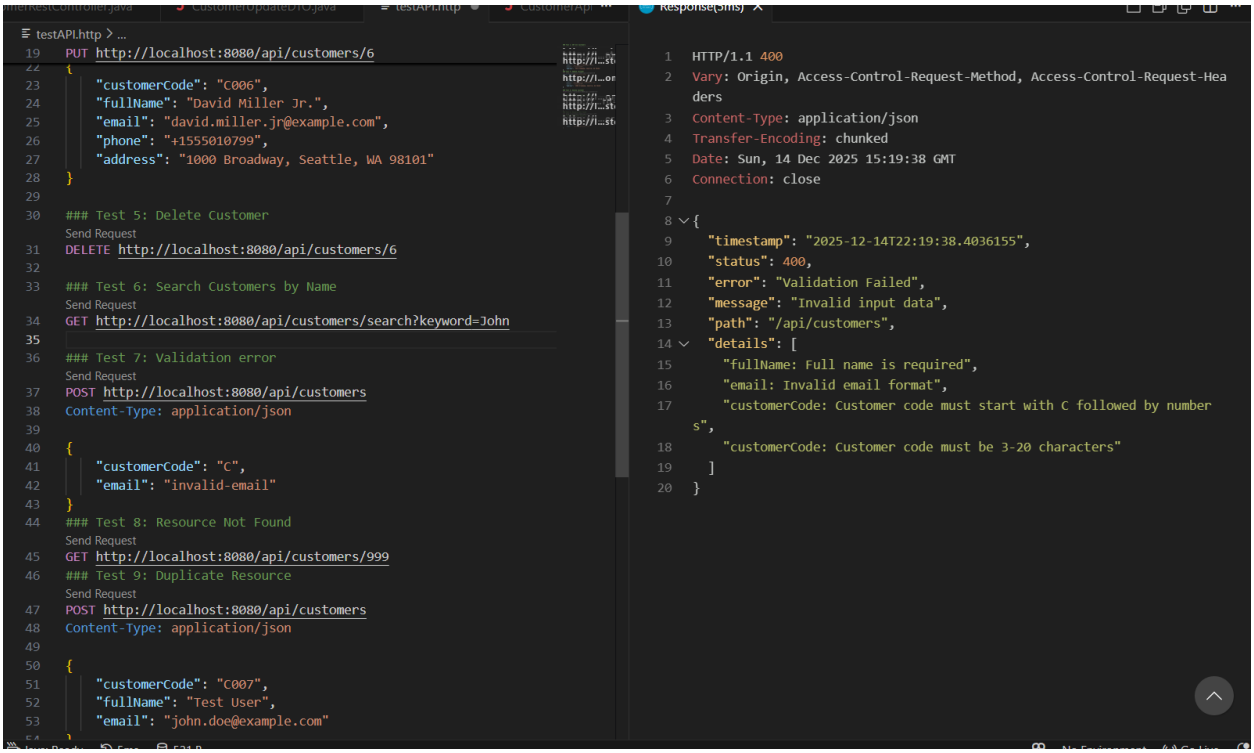


```
testAPI.http > PUT /api/customers/6
}
### Test 4: Update Customer
Send Request
PUT http://localhost:8080/api/customers/6
Content-Type: application/json

{
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+1555010799",
    "address": "1000 Broadway, Seattle, WA 98101"
}

### Test 5: Delete Customer
Send Request
DELETE http://localhost:8080/api/customers/6

### Test 6: Search Customers by Name
Send Request
GET http://localhost:8080/api/customers/search?keyword=John

### Test 7: Validation error
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json

{
    "customerCode": "C",
    "email": "invalid-email"
}
### Test 8: Resource Not Found
Send Request
GET http://localhost:8080/api/customers/999
### Test 9: Duplicate Resource
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json
```

```
HTTP/1.1 200
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:19:18 GMT
Connection: close

[
  {
    "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": "2025-12-14T20:14:02",
    "links": []
  },
  {
    "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": "2025-12-14T20:14:02",
    "links": []
  }
]
```

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

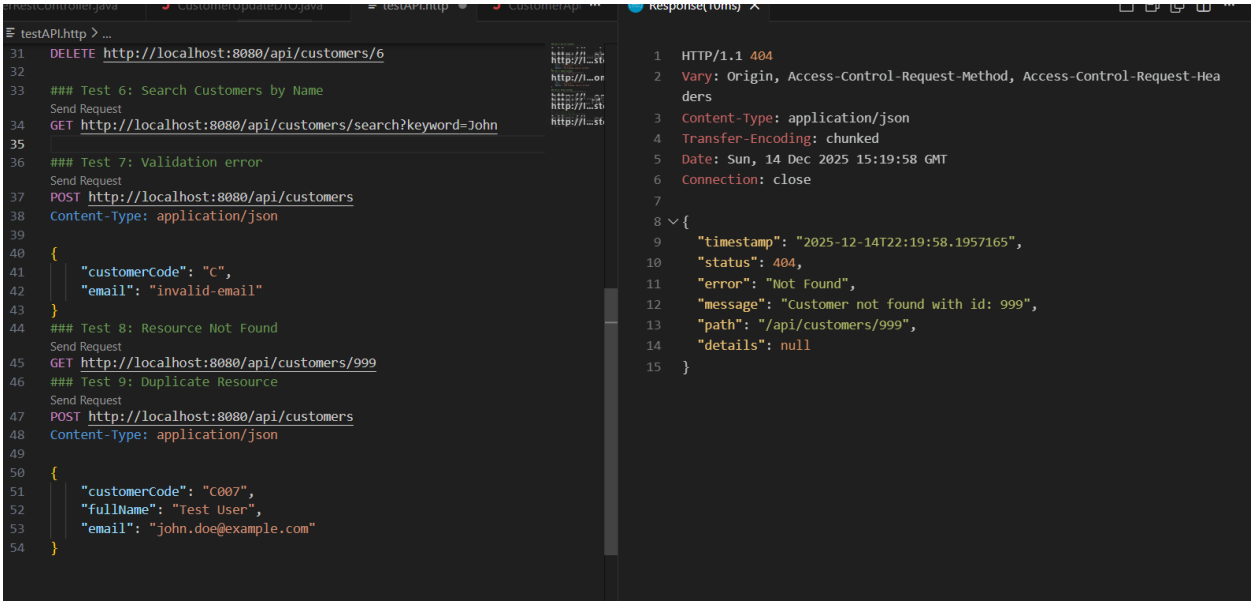


```
PUT http://localhost:8080/api/customers/6
{
    "customerCode": "C006",
    "fullName": "David Miller Jr.",
    "email": "david.miller.jr@example.com",
    "phone": "+1555010799",
    "address": "1000 Broadway, Seattle, WA 98101"
}

### Test 5: Delete Customer
Send Request
DELETE http://localhost:8080/api/customers/6

### Test 6: Search Customers by Name
Send Request
GET http://localhost:8080/api/customers/search?keyword=John

### Test 7: Validation error
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json

{
    "customerCode": "C",
    "email": "invalid-email"
}
### Test 8: Resource Not Found
Send Request
GET http://localhost:8080/api/customers/999
### Test 9: Duplicate Resource
Send Request
POST http://localhost:8080/api/customers
Content-Type: application/json

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

```
HTTP/1.1 400
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:19:38 GMT
Connection: close

{
  "timestamp": "2025-12-14T22:19:38.4036155",
  "status": 400,
  "error": "Validation Failed",
  "message": "Invalid input data",
  "path": "/api/customers",
  "details": [
    "fullName: Full name is required",
    "email: Invalid email format",
    "customerCode: Customer code must start with C followed by numbers",
    "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

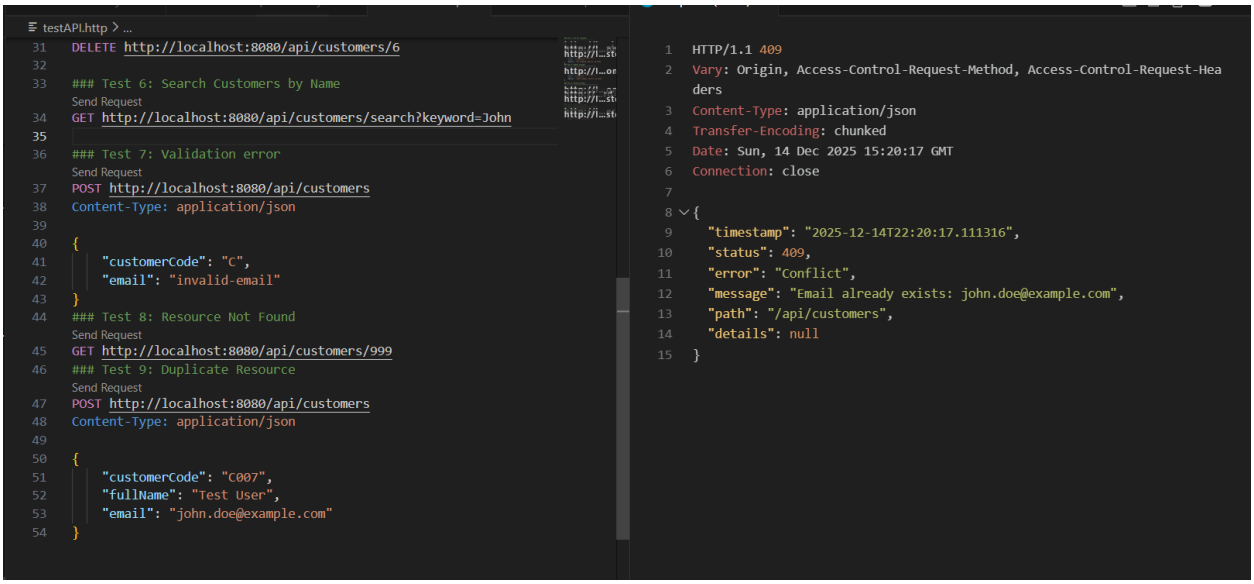


```
testAPI.http > ...
31  DELETE http://localhost:8080/api/customers/6
32
33  ### Test 6: Search Customers by Name
    Send Request
34  GET http://localhost:8080/api/customers/search?keyword=John
35
36  ### Test 7: Validation error
    Send Request
37  POST http://localhost:8080/api/customers
38  Content-Type: application/json
39
40  {
41      "customerCode": "C",
42      "email": "invalid-email"
43  }
44  ### Test 8: Resource Not Found
    Send Request
45  GET http://localhost:8080/api/customers/999
46  ### Test 9: Duplicate Resource
    Send Request
47  POST http://localhost:8080/api/customers
48  Content-Type: application/json
49
50  {
51      "customerCode": "C007",
52      "fullName": "Test User",
53      "email": "john.doe@example.com"
54  }
```

```
1   HTTP/1.1 404
2   Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Hea
    ders
3   Content-Type: application/json
4   Transfer-Encoding: chunked
5   Date: Sun, 14 Dec 2025 15:19:58 GMT
6   Connection: close
7
8   {
9     "timestamp": "2025-12-14T22:19:58.1957165",
10    "status": 404,
11    "error": "Not Found",
12    "message": "Customer not found with id: 999",
13    "path": "/api/customers/999",
14    "details": null
15  }
```

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



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
# GET customer by ID
curl http://localhost:8080/api/customers/1
# 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"
}'
# 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"
}'
# DELETE customer
curl -X DELETE http://localhost:8080/api/customers/6
# Search customers
curl "http://localhost:8080/api/customers/search?keyword=john"
```