

Test 1: GET All Customers

```
testAPI.http > ⌂ GET /api/customers/999
1  ### Test 1: GET All Customers
2  Send Request
3  GET http://localhost:8080/api/customers
4
5  ### Test 2: Create Customer
6  Send Request
7  GET http://localhost:8080/api/customers/1
8
9  ### Test 3: GET Customer by ID
10 Send Request
11 POST http://localhost:8080/api/customers
12 Content-Type: application/json
13 {
14   "customerCode": "C006",
15   "fullName": "David Miller",
16   "email": "david.miller@example.com",
17   "phone": "+15550106",
18   "address": "999 Broadway, Seattle, WA 98101"
19
20  ### Test 4: Update Customer
21 Send Request
22 PUT http://localhost:8080/api/customers/
23 Content-Type: application/json
24
25  {
26   "customerCode": "C006",
27   "fullName": "David Miller Jr.",
28   "email": "david.miller.jr@example.com",
29   "phone": "+15550107",
30   "address": "1000 Broadway, Seattle, WA 98101"
31
32  ### Test 5: Delete Customer
33 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",
      "links": []
    }
  ]
}
```

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

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

```
testAPI.http > POST /api/customers
1  ### Test 1: GET All Customers
2  Send Request
3  GET http://localhost:8080/api/customers
4
5  ### Test 2: GET Customer by ID
6  Send Request
7  GET http://localhost:8080/api/customers/1
8
9  ### Test 3: Create Customer
10 Send Request
11 POST http://localhost:8080/api/customers
12 Content-Type: application/json
13
14 {
15     "customerCode": "C006",
16     "fullName": "David Miller",
17     "email": "david.miller@example.com",
18     "phone": "+1555010699",
19     "address": "999 Broadway, Seattle, WA 98101"
20 }
21
22 ### Test 4: Update Customer
23 Send Request
24 PUT http://localhost:8080/api/customers/6
25 Content-Type: application/json
26
27 {
28     "customerCode": "C006",
29     "fullName": "David Miller Jr.",
30     "email": "david.miller.jr@example.com",
31     "phone": "+15550107",
```

HTTP/1.1 201
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-type: application/hal+json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:15:41 GMT
Connection: close

{
 "id": 6,
 "customerCode": "C006",
 "fullName": "David Miller",
 "email": "david.miller@example.com",
 "phone": "+1555010699",
 "address": "999 Broadway, Seattle, WA 98101",
 "status": "ACTIVE",
 "createdAt": "2025-12-14T22:15:41.6690413"
}

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

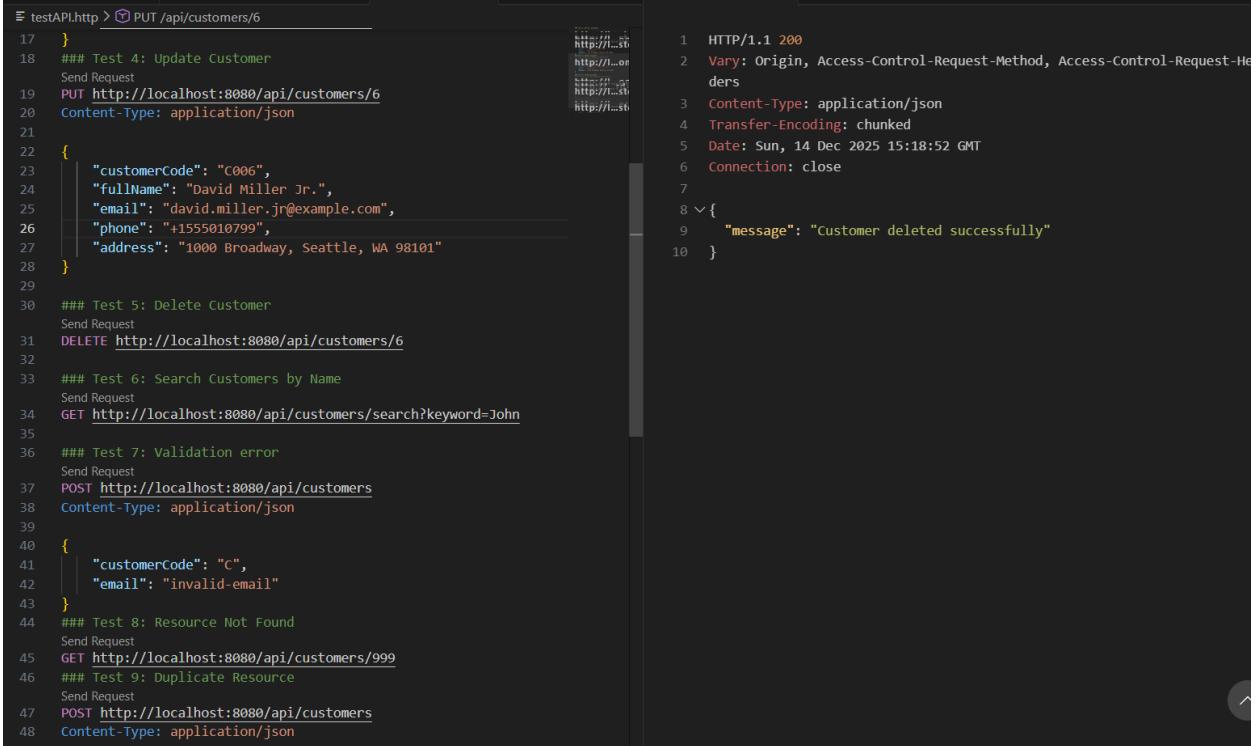
```
testAPI.http > ⏪ PUT /api/customers/6
1  ## Test 1: GET All Customers
2  Send Request
3  GET http://localhost:8080/api/customers
4
5  ## Test 2: GET Customer by ID
6  Send Request
7  GET http://localhost:8080/api/customers/1
8
9  ## Test 3: Create Customer
10 Send Request
11 POST http://localhost:8080/api/customers
12 Content-Type: application/json
13
14 {
15     "customerCode": "C006",
16     "fullName": "David Miller",
17     "email": "david.miller@example.com",
18     "phone": "+1555010699",
19     "address": "999 Broadway, Seattle, WA 98101"
20 }
21
22 {
23     "customerCode": "C006",
24     "fullName": "David Miller Jr.",
25     "email": "david.miller.jr@example.com",
26     "phone": "+1555010799",
27     "address": "1000 Broadway, Seattle, WA 98101"
28 }
29
30 ## Test 4: Update Customer
31 Send Request
32 PUT http://localhost:8080/api/customers/6
33 Content-type: application/json
34
35 {
36     "id": 6,
37     "customerCode": "C006",
38     "fullName": "David Miller Jr.",
39     "email": "david.miller.jr@example.com",
40     "phone": "+1555010799",
41     "address": "1000 Broadway, Seattle, WA 98101"
42 }
43
44 ## Test 5: Delete Customer
45 Send Request
46 DELETE http://localhost:8080/api/customers/6
47
48 ## Test 6: Search Customers by Name
```

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

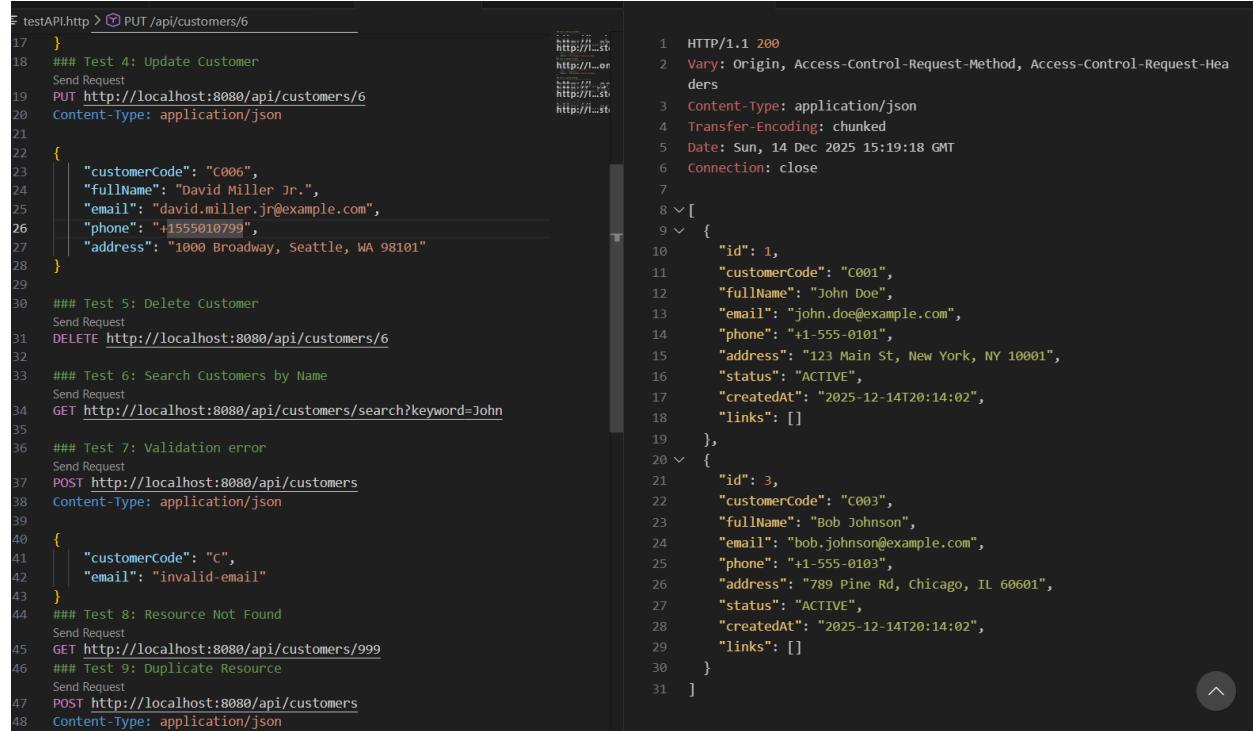


```
testAPI.http > PUT /api/customers/6
17 }
18 ### Test 4: Update Customer
Send Request
19 PUT http://localhost:8080/api/customers/6
20 Content-Type: application/json
21
22 {
23   "customerCode": "C006",
24   "fullName": "David Miller Jr.",
25   "email": "david.miller.jr@example.com",
26   "phone": "+1555010799",
27   "address": "1000 Broadway, Seattle, WA 98101"
28 }
29
30 ### Test 5: Delete Customer
Send Request
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
```

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 15:18:52 GMT
Connection: close
{"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



The screenshot shows a terminal window with two panes. The left pane displays a series of API test cases numbered 17 through 48. The right pane shows the corresponding HTTP responses. The responses include headers like 'Content-Type: application/json' and 'Transfer-Encoding: chunked', and body content such as customer details and a list of customers.

```
testAPIhttp > PUT /api/customers/6
17 }
18 ## Test 4: Update Customer
Send Request
19 PUT http://localhost:8080/api/customers/6
20 Content-Type: application/json
21
22 {
| "customerCode": "C006",
| "fullName": "David Miller Jr.",
| "email": "david.miller.jr@example.com",
23 "phone": "+155010799",
24 "address": "1000 Broadway, Seattle, WA 98101"
25
26
27 }
28 }

### Test 5: Delete Customer
Send Request
31 DELETE http://localhost:8080/api/customers/6
32

### Test 6: Search Customers by Name
Send Request
34 GET http://localhost:8080/api/customers/search?keyword=John
35

### Test 7: Validation error
Send Request
37 POST http://localhost:8080/api/customers
38 Content-Type: application/json
39
40 {
| "customerCode": "C",
| "email": "invalid-email"
41
42 }

### 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
```

```
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

```
CustomerController.java      CustomerUpdatedDTO.java      testAPI.http      CustomerAPI.java
1  testAPI.http > ...
2  PUT http://localhost:8080/api/customers/6
3  {
4      "customerCode": "C006",
5      "fullName": "David Miller Jr.",
6      "email": "david.miller.jr@example.com",
7      "phone": "+1555010799",
8      "address": "1000 Broadway, Seattle, WA 98101"
9  }
10
11  ### Test 5: Delete Customer
12  Send Request
13  DELETE http://localhost:8080/api/customers/6
14
15  ### Test 6: Search Customers by Name
16  Send Request
17  GET http://localhost:8080/api/customers/search?keyword=John
18
19  ### Test 7: Validation error
20  Send Request
21  POST http://localhost:8080/api/customers
22  Content-Type: application/json
23
24  {
25      "customerCode": "C",
26      "email": "invalid-email"
27  }
28
29  ### Test 8: Resource Not Found
30  Send Request
31  GET http://localhost:8080/api/customers/999
32
33  ### Test 9: Duplicate Resource
34  Send Request
35  POST http://localhost:8080/api/customers
36  Content-Type: application/json
37
38  {
39      "customerCode": "C007",
40      "fullName": "Test User",
41      "email": "john.doe@example.com"
42
43
44
45
46
47
48
49
50 }
```

Response(SIMs) ▾

```
1  HTTP/1.1 400
2  Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
3  Content-Type: application/json
4  Transfer-Encoding: chunked
5  Date: Sun, 14 Dec 2025 15:19:38 GMT
6  Connection: close
7
8  {
9      "timestamp": "2025-12-14T22:19:38.4036155",
10     "status": 400,
11     "error": "Validation Failed",
12     "message": "Invalid input data",
13     "path": "/api/customers",
14     "details": [
15         "fullName: Full name is required",
16         "email: Invalid email format",
17         "customerCode: Customer code must start with C followed by number s",
18         "customerCode: customer code must be 3-20 characters"
19     ]
20 }
```

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

The screenshot shows a code editor with two tabs: 'testAPI.http' and 'Response(20ms)'. The 'testAPI.http' tab contains a series of HTTP requests and their responses. The 'Response(20ms)' tab shows the detailed response for the 404 error.

```
testAPI.http > ...
31  DELETE http://localhost:8080/api/customers/6
32
33  ### Test 6: Search Customers by Name
34  Send Request
35  GET http://localhost:8080/api/customers/search?keyword=John
36
37  ### Test 7: Validation error
38  Send Request
39  POST http://localhost:8080/api/customers
40  Content-Type: application/json
41
42  {
43  |   "customerCode": "C",
44  |   "email": "invalid-email"
45  }
46  ### Test 8: Resource Not Found
47  Send Request
48  GET http://localhost:8080/api/customers/999
49  ### Test 9: Duplicate Resource
50  Send Request
51  POST http://localhost:8080/api/customers
52  Content-Type: application/json
53
54  {
55  |   "customerCode": "C007",
56  |   "fullName": "Test User",
57  |   "email": "john.doe@example.com"
58 }

Response(20ms) > ...
1  HTTP/1.1 404
2  Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Hea
3  ders
4  Content-Type: application/json
5  Transfer-Encoding: chunked
6  Date: Sun, 14 Dec 2025 15:19:58 GMT
7  Connection: close
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

```
testAPI.http > ...
31  DELETE http://localhost:8080/api/customers/6
32
33  ### Test 6: Search Customers by Name
34  Send Request
35  GET http://localhost:8080/api/customers/search?keyword=John
36
37  ### Test 7: Validation error
38  Send Request
39  POST http://localhost:8080/api/customers
40  Content-Type: application/json
41
42  {
43    "customerCode": "C",
44    "email": "invalid-email"
45  }
46  ### Test 8: Resource Not Found
47  Send Request
48  GET http://localhost:8080/api/customers/999
49  ### Test 9: Duplicate Resource
50  Send Request
51  POST http://localhost:8080/api/customers
52  Content-Type: application/json
53
54  {
55    "customerCode": "C007",
56    "fullName": "Test User",
57    "email": "john.doe@example.com"
58 }
```

HTTP/1.1 409
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sun, 14 Dec 2025 15:20:17 GMT
Connection: close

{"timestamp": "2025-12-14T22:20:17.111316",
"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
```

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"
```