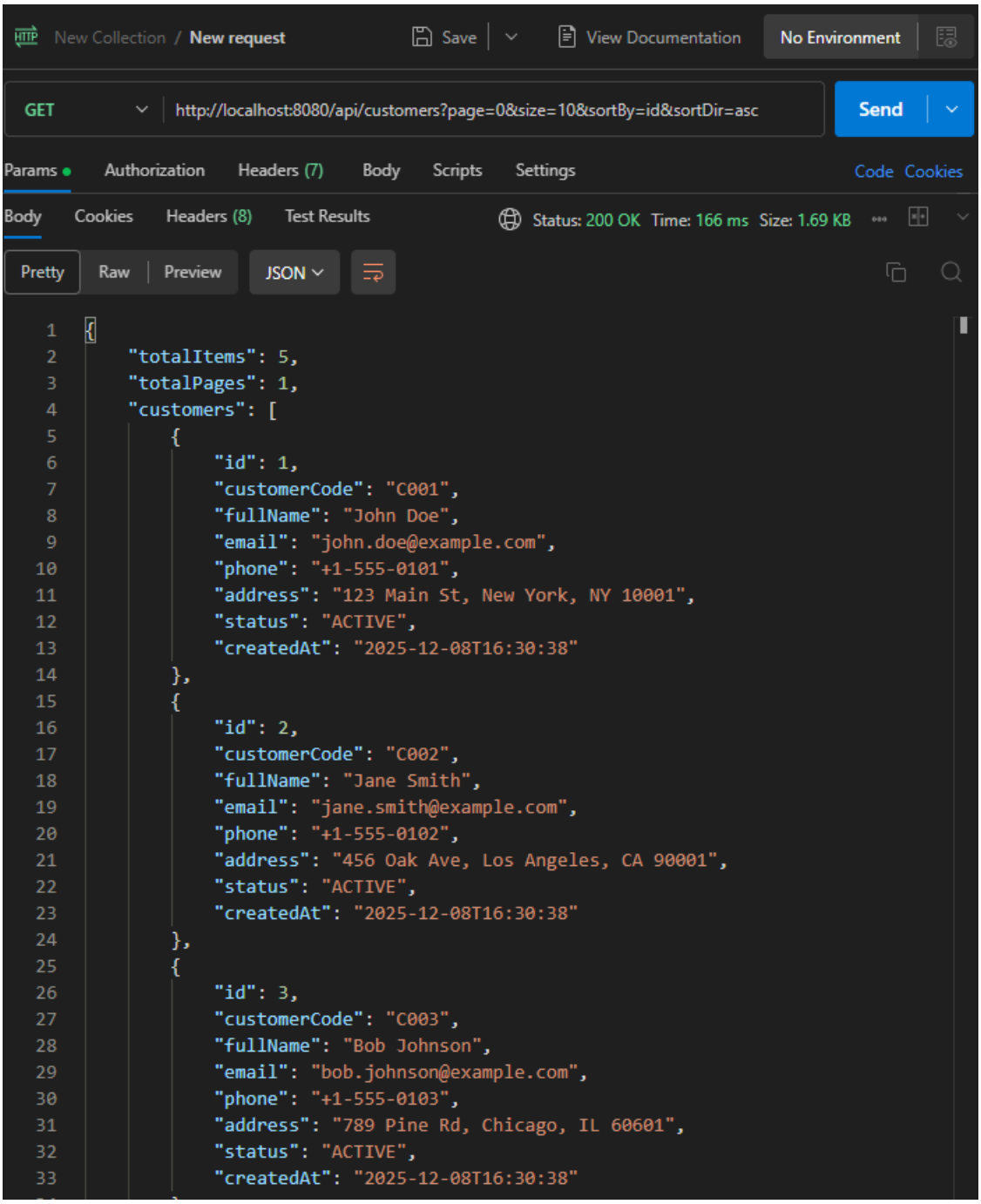
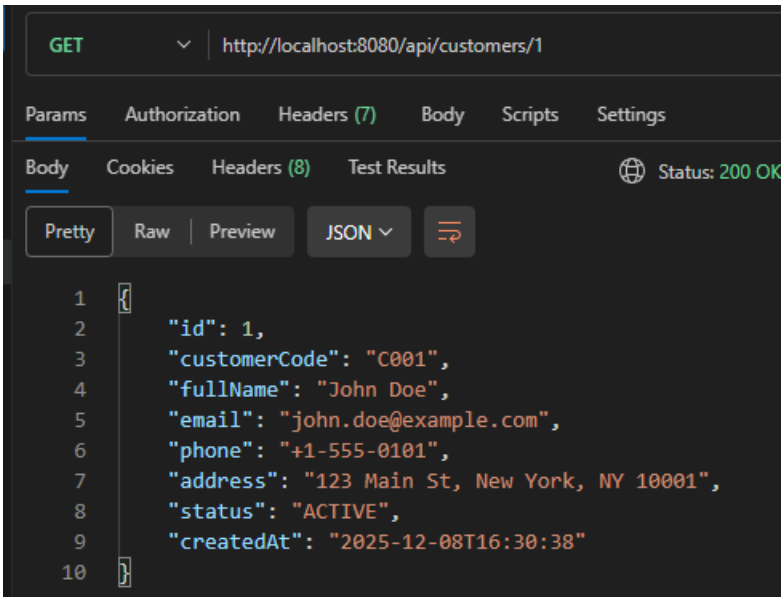
A1 — GET all customers (200) + Pagination/Sorting

This request retrieves the customer list using pagination and sorting parameters. The response returns the first page of customers sorted by id in ascending order, including metadata like totalItems and totalPages.



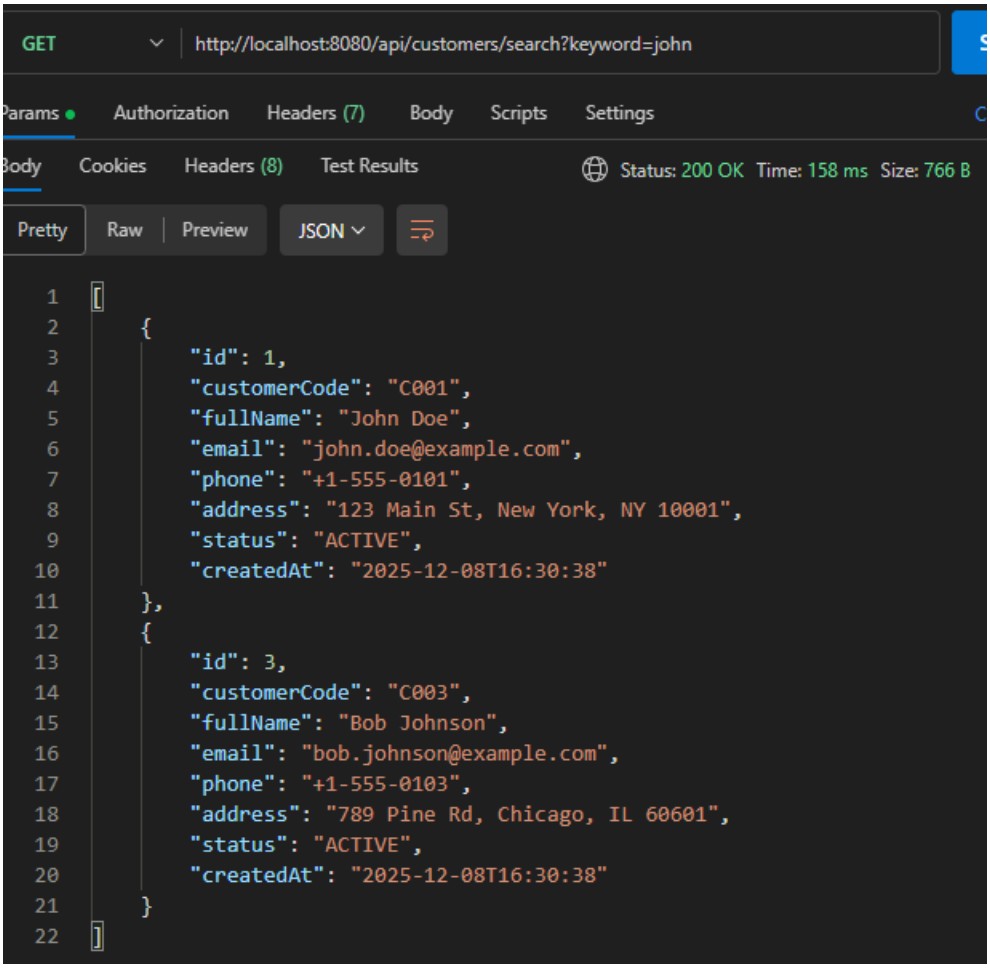
B1 — GET customer by ID (200)

This request fetches a single customer by a specific ID. The response confirms the customer exists and returns that customer's details (e.g., customerCode C001).



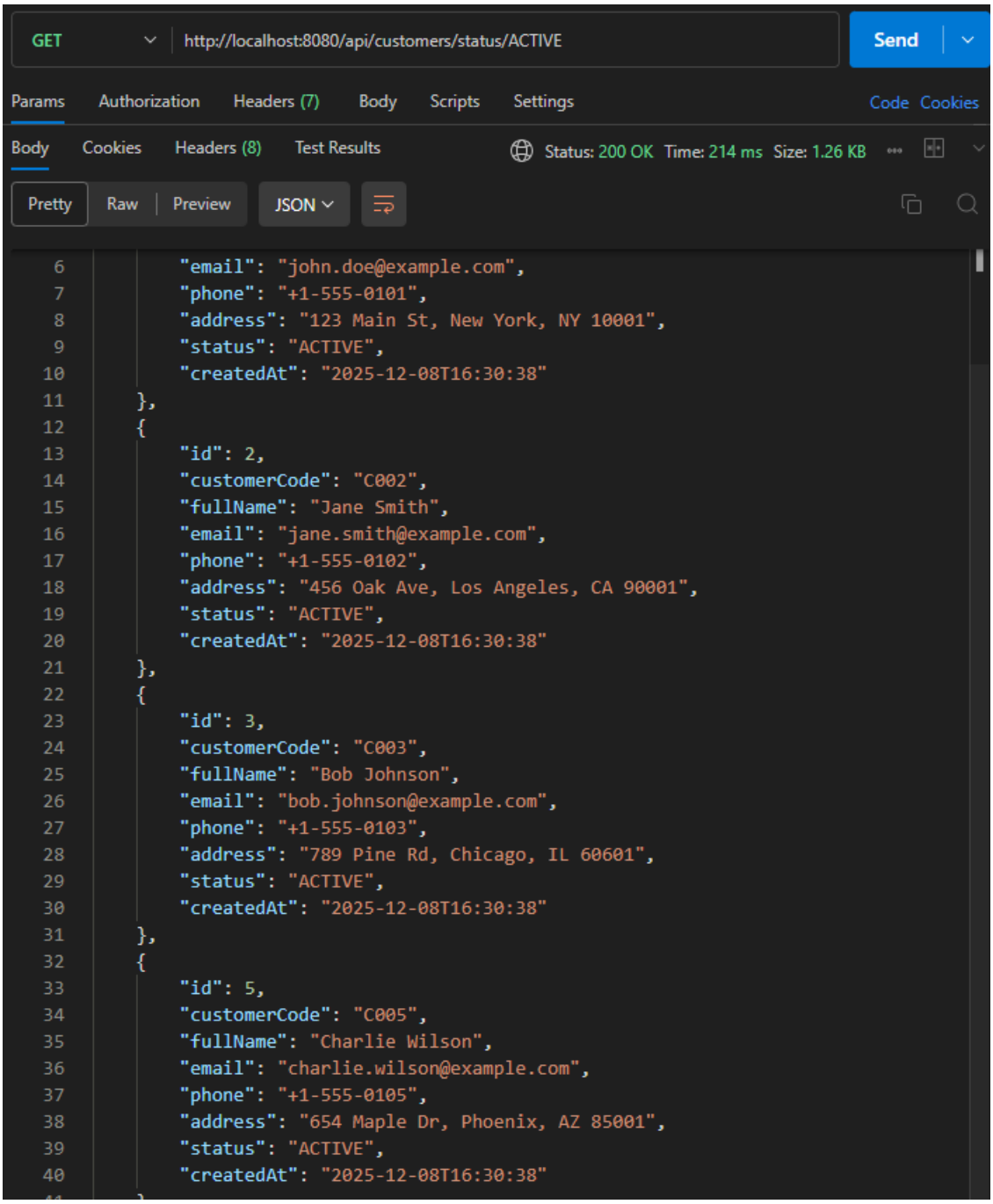
C1 — Search customers by keyword (200)

This request searches customers using a keyword across name, email, and customer code. The response returns a list of matching customers that contain the keyword (e.g., "john").



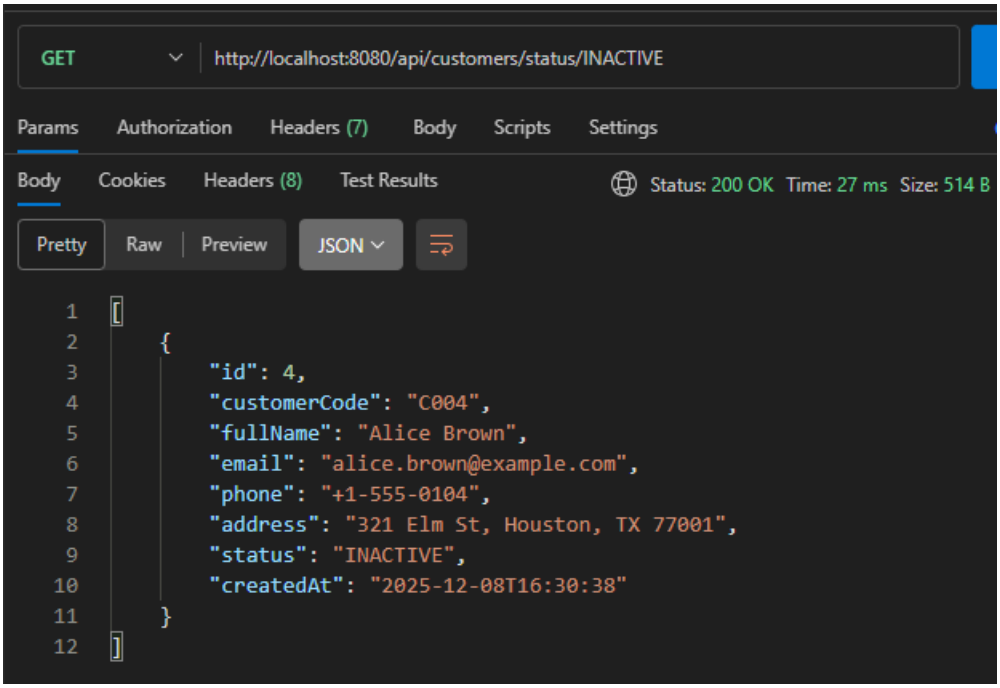
D1 — Filter customers by status ACTIVE (200)

This request filters customers by status ACTIVE. The response returns only customers whose status is ACTIVE.



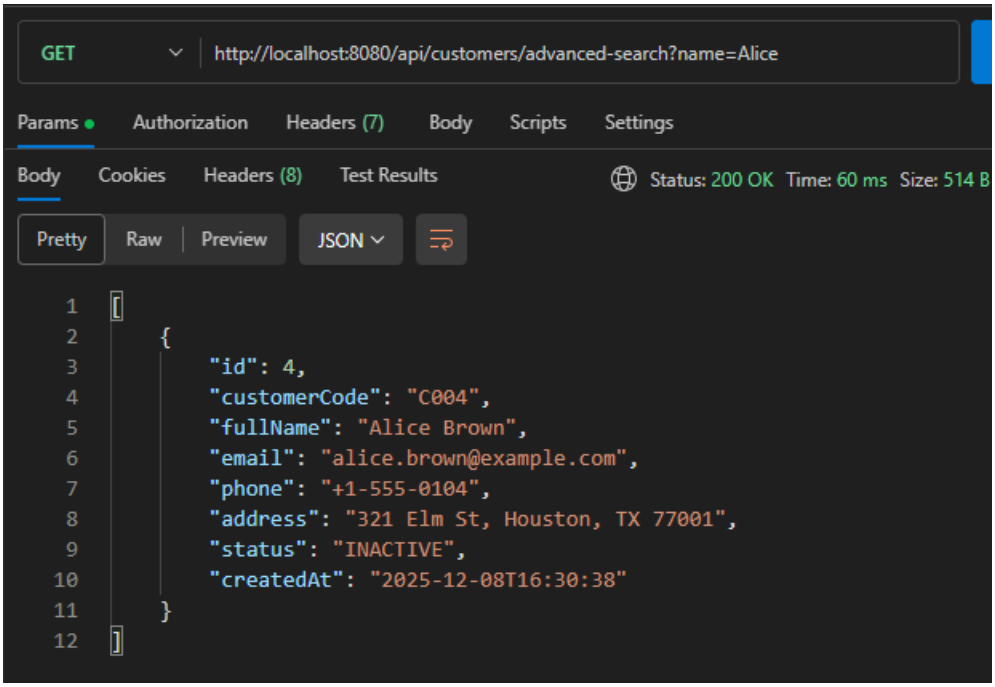
D2 — Filter customers by status INACTIVE (200)

This request filters customers by status INACTIVE. The response returns only customers whose status is INACTIVE.



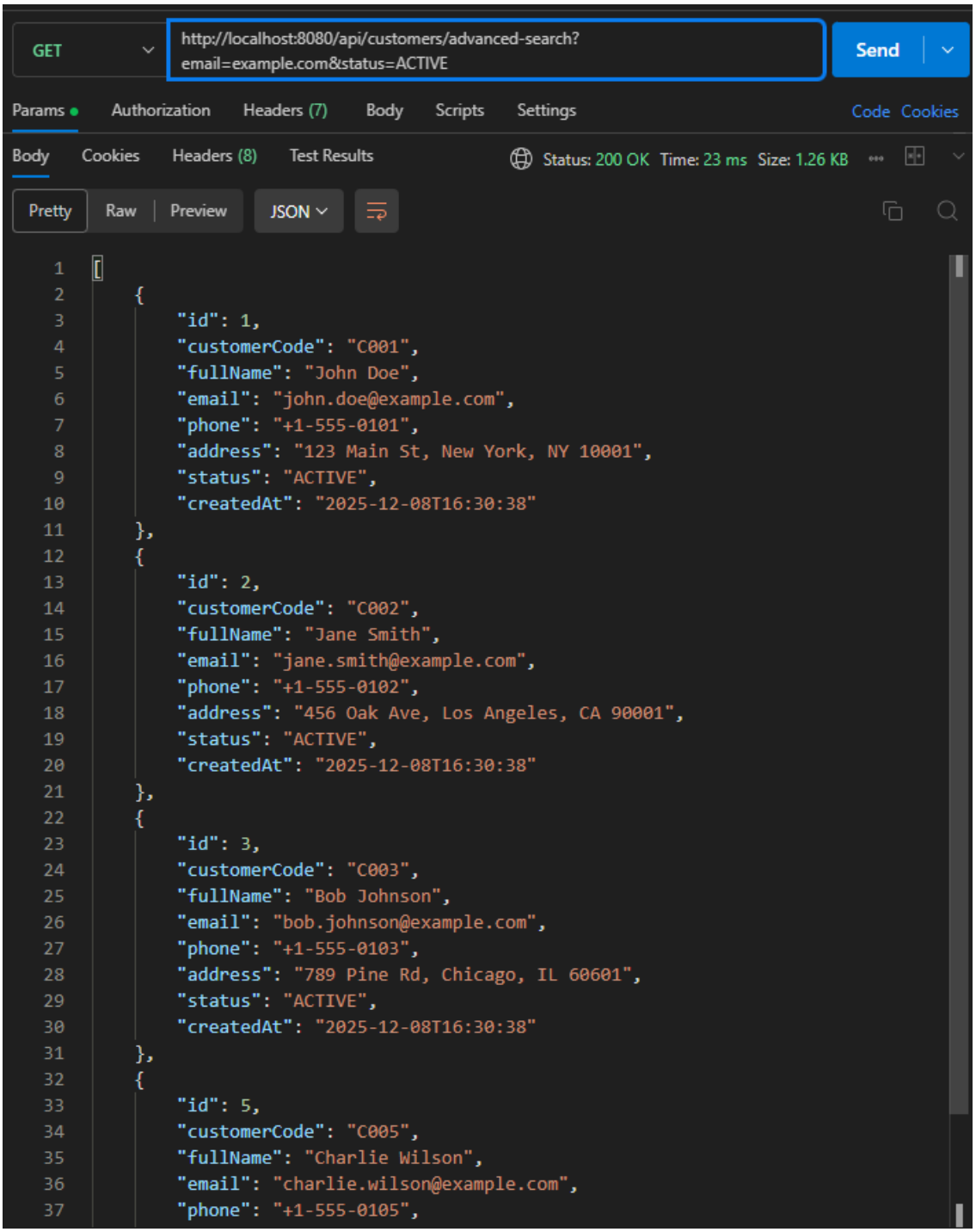
E1 — Advanced search by name (200)

This request performs an advanced search using an optional name parameter. The response returns customers whose full name matches the provided keyword (e.g., “Alice”).



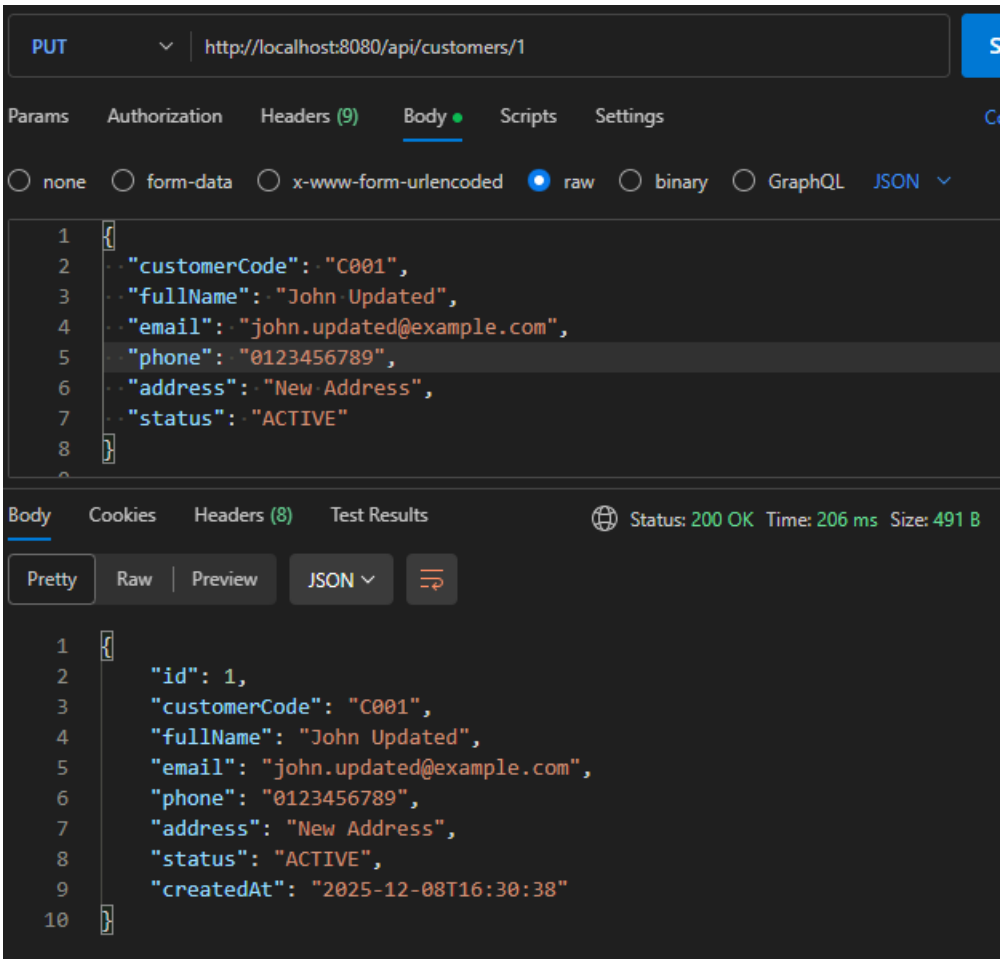
E2 — Advanced search by email + status (200)

This request performs an advanced search using multiple criteria (email and status). The response returns customers that match all provided conditions (e.g., email contains “example.com” and status is ACTIVE).



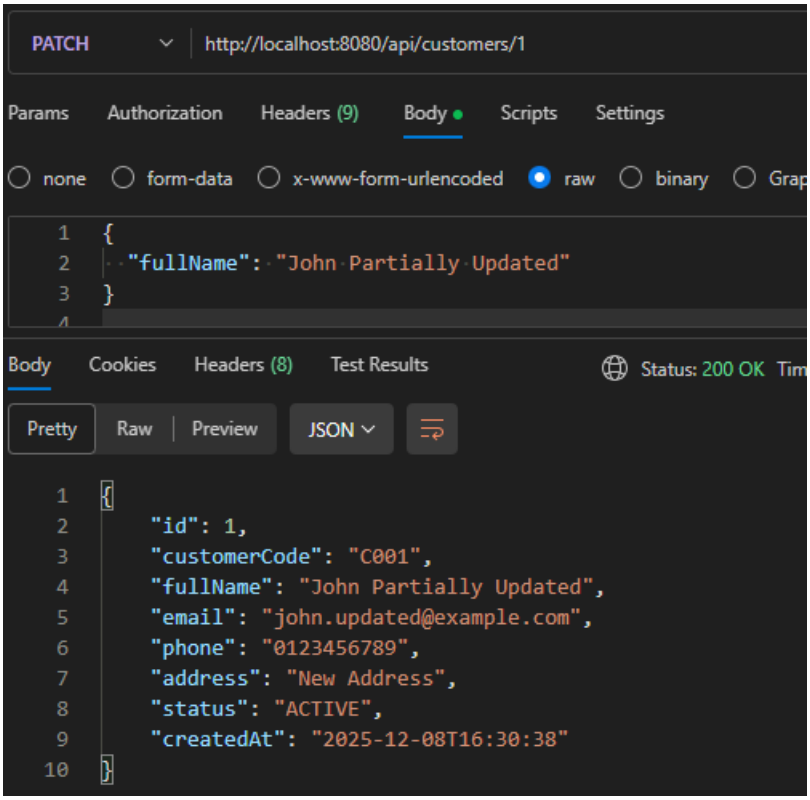
PUT1 — PUT full update customer (200)

This request fully updates a customer record using PUT. The response confirms the update succeeded and returns the updated customer data.



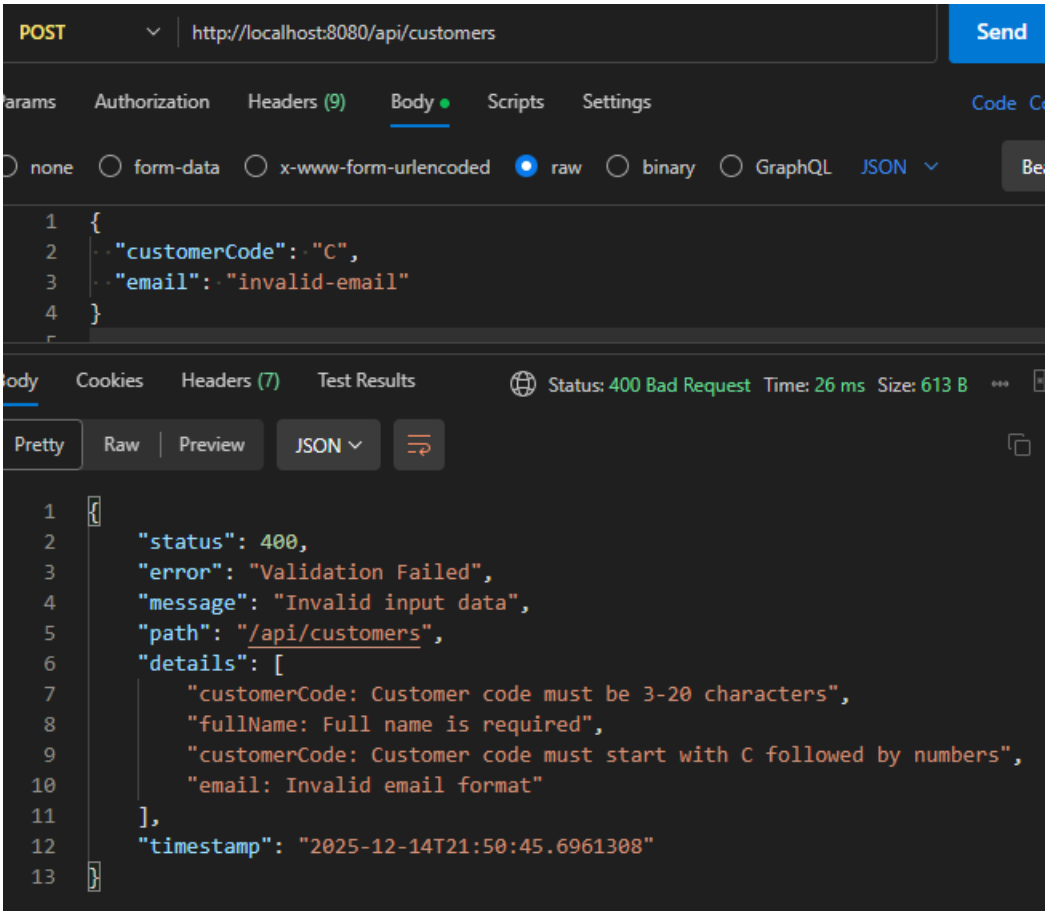
PATCH1 — PATCH partial update customer (200)

This request partially updates a customer by sending only the fields that need to change. The response confirms only the provided fields (e.g., fullName) were updated while other fields remain unchanged.



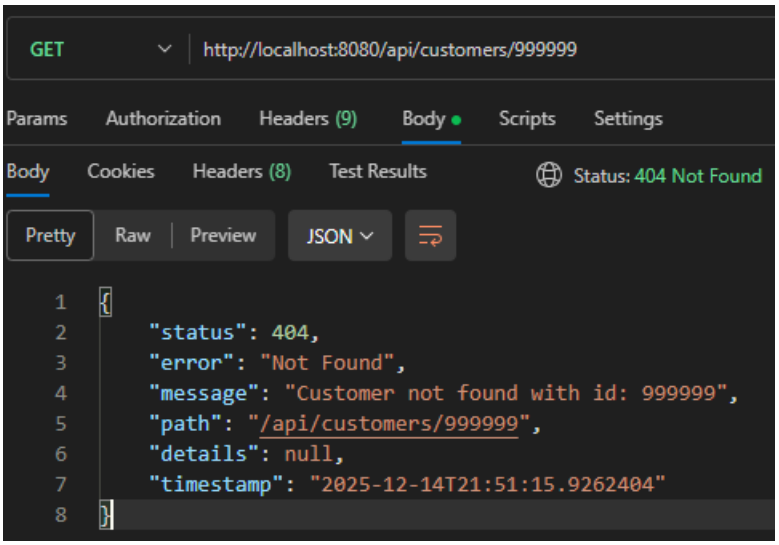
ERR400 — POST validation error (400)

This request sends invalid input data to test validation rules. The server responds with 400 Bad Request and returns validation error details for the incorrect/missing fields.



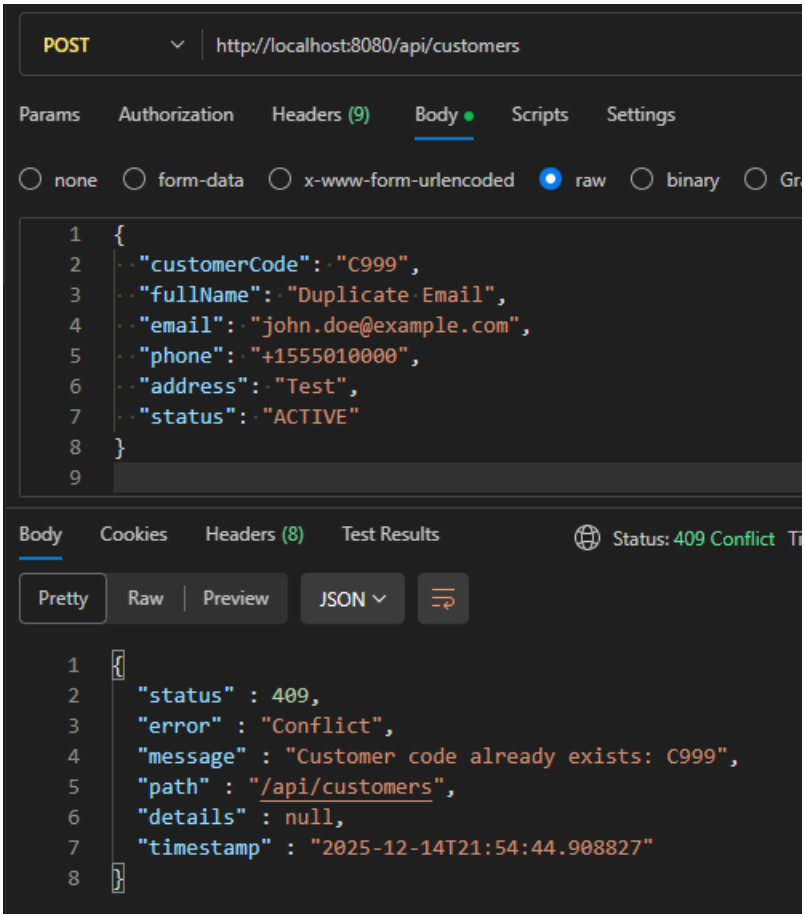
ERR404 — GET not found (404)

This request tries to fetch a customer using a non-existent ID. The server responds with 404 Not Found, confirming the resource does not exist.



ERR409 — POST duplicate resource (409)

This request attempts to create a customer with an email that already exists in the database. The server responds with 409 Conflict to indicate a duplicate resource constraint violation.


POST
http://localhost:8080/api/customers
Params
Authorization
Headers (9)
Body
Scripts
Settings
none
form-data
x-www-form-urlencoded
raw
binary
{
"customerCode": "C999",
"fullName": "Duplicate Email",
"email": "john.doe@example.com",
"phone": "+1555010000",
"address": "Test",
"status": "ACTIVE"
}
Body
Cookies
Headers (8)
Test Results
Status: 409 Conflict
Pretty
Raw
Preview
JSON
{
"status" : 409,
"error" : "Conflict",
"message" : "Customer code already exists: C999",
"path" : "/api/customers",
"details" : null,
"timestamp" : "2025-12-14T21:54:44.908827"
}