

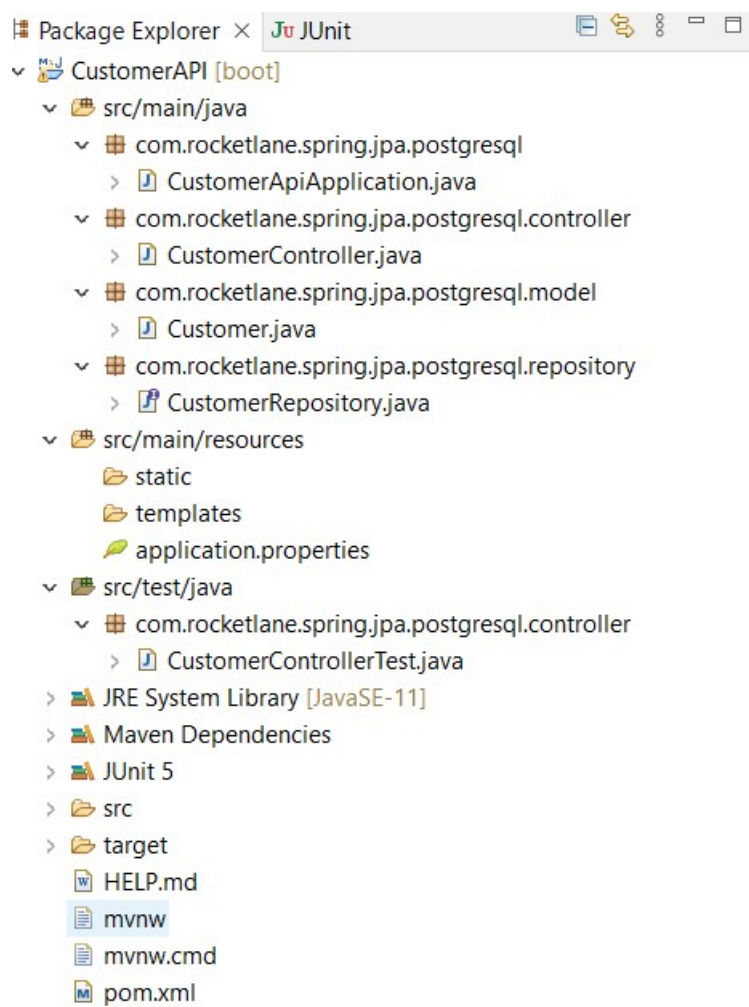
ROCKETLANE CODING ASSIGNMENT

SERVICE TO MANAGE CUSTOMER INFORMATION

Framework used: Spring Boot

Database: Postgresql

Project Layout:



PACKAGES AND LAYERS IN THIS PROJECT

1. DOMAIN LAYER – Creating Customer Model

Package: COM.ROCKETLANE.SPRING.JPA.POSTGRESQL.MODEL

The data model (Customer) is defined with the fields: ID, FIRSTNAME, LASTNAME, EMAILID, MOBILENO, CITY and ADDRESS.

2. PERSISTENCE LAYER – Creating Customer Repository

Package: COM.ROCKETLANE.SPRING.JPA.POSTGRESQL.REPOSITORY

In the repository package, CustomerRepository interface extends JpaRepository .

It provides method for easy retrieval from the database without the need for manual implementation.

3. BUSINESS LAYER- Create Customer Controller

Package: COM.ROCKETLANE.SPRING.JPA.POSTGRESQL.CONTROLLER

Customer controller is the controller which exposes REST endpoints for the CRUD Operation.

Create- HTTP POST METHOD

One POST method to **create a new customer record** in the database.

Read- HTTP GET METHOD

Two GET methods to **retrieve customer records** from the database. One gets all the customer details and the other gets a single record given a customer ID.

Update- HTTP PUT METHOD

One PUT method to **update a customer record** in the database given the customer id of the record to be updated is given.

Delete- HTTP DELETE METHOD

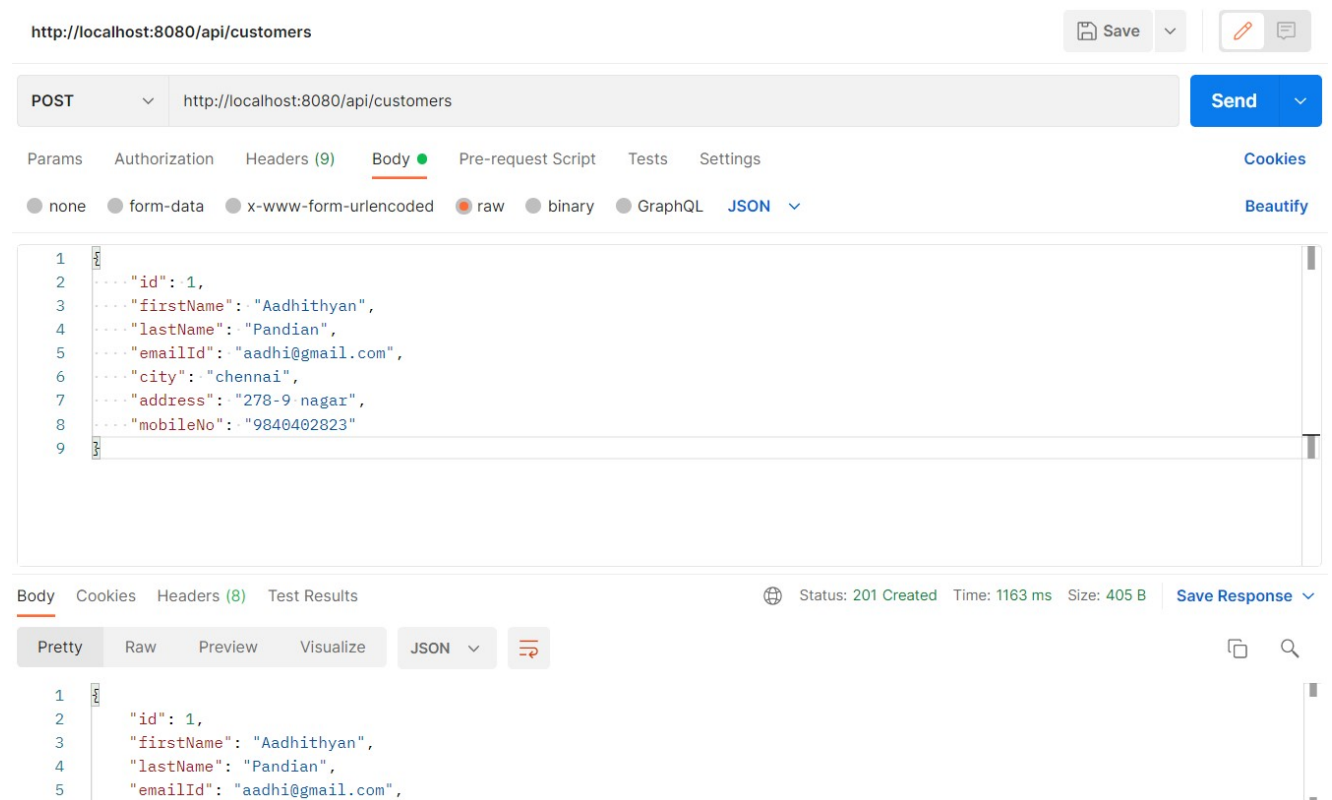
One DELETE method to **delete a customer record** in the database given the customer id of the record to be deleted.

IMPLEMENTED API'S:

1. HTTP POST <http://localhost:8080/customers>
2. HTTP GET <http://localhost:8080/customers/2>
3. HTTP GET <http://localhost:8080/customers>
3. HTTP PUT <http://localhost:8080/customers/2>
4. HTTP DELETE <http://localhost:8080/customers/2>

TESTING THE APIs IN POSTMAN:

Creating a new record using POST method:



Retrieving all customers detail with GET method:

http://localhost:8080/api/customers

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

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results Status: 200 OK Time: 544 ms Size: 706 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "firstName": "Aadhithyan",
5     "lastName": "Pandian",
6     "emailId": "aadhi@gmail.com",
7     "city": "chennai",
8     "address": "278-9 nagar",
9     "mobileNo": "9840402823"
10  },
11  {
12    "id": 2,
13    "firstName": "Aadil",
14    "lastName": "Rasheed",
15    "emailId": "aadil@gmail.com",
16    "city": "Srinagar",
17    "address": "Dal Street",
18    "mobileNo": "9840486534"
19  },
20  {
21    "id": 3,
22    "firstName": "Harikrishnan"
```

Retrieving customer details based on ID using GET method:

GET http://localhost:8080... + ... No Environment

http://localhost:8080/api/customers/3

GET http://localhost:8080/api/customers/3

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results Status: 200 OK Time: 54 ms Size: 413 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "firstName": "Harikrishnan",
4   "lastName": "Srikrishnan",
5   "emailId": "hksk@gmail.com",
6   "city": "Dubai",
7   "address": "Vivekanandar Street",
8   "mobileNo": "342343212343"
9 }
```

Updating customer detail with given id using PUT method:

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/api/customers/2`. The request body is a JSON object representing a customer with the following details:

```
1 {
2   "id": 2,
3   "firstName": "Aadil",
4   "lastName": "Rasheed Lone",
5   "emailId": "arl@gmail.com",
6   "city": "Banglore",
7   "address": "Indiranagar",
8   "mobileNo": "9850502823"
9 }
```

The response status is 200 OK, with a time of 56 ms and a size of 399 B. The response body is displayed in a pretty-printed JSON format:

```
1 {
2   "id": 2,
3   "firstName": "Aadil",
4   "lastName": "Rasheed Lone",
5   "emailId": "arl@gmail.com",
6   "city": "Banglore",
7   "address": "Indiranagar",
8   "mobileNo": "9850502823"
9 }
```

Deleting a customer detail with the given ID using DELETE method:

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8080/api/customers/1`. The interface includes tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings. The Body tab is currently selected, and the response status is 200 OK.

After deleting the customer with id 1:

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

Save

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

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

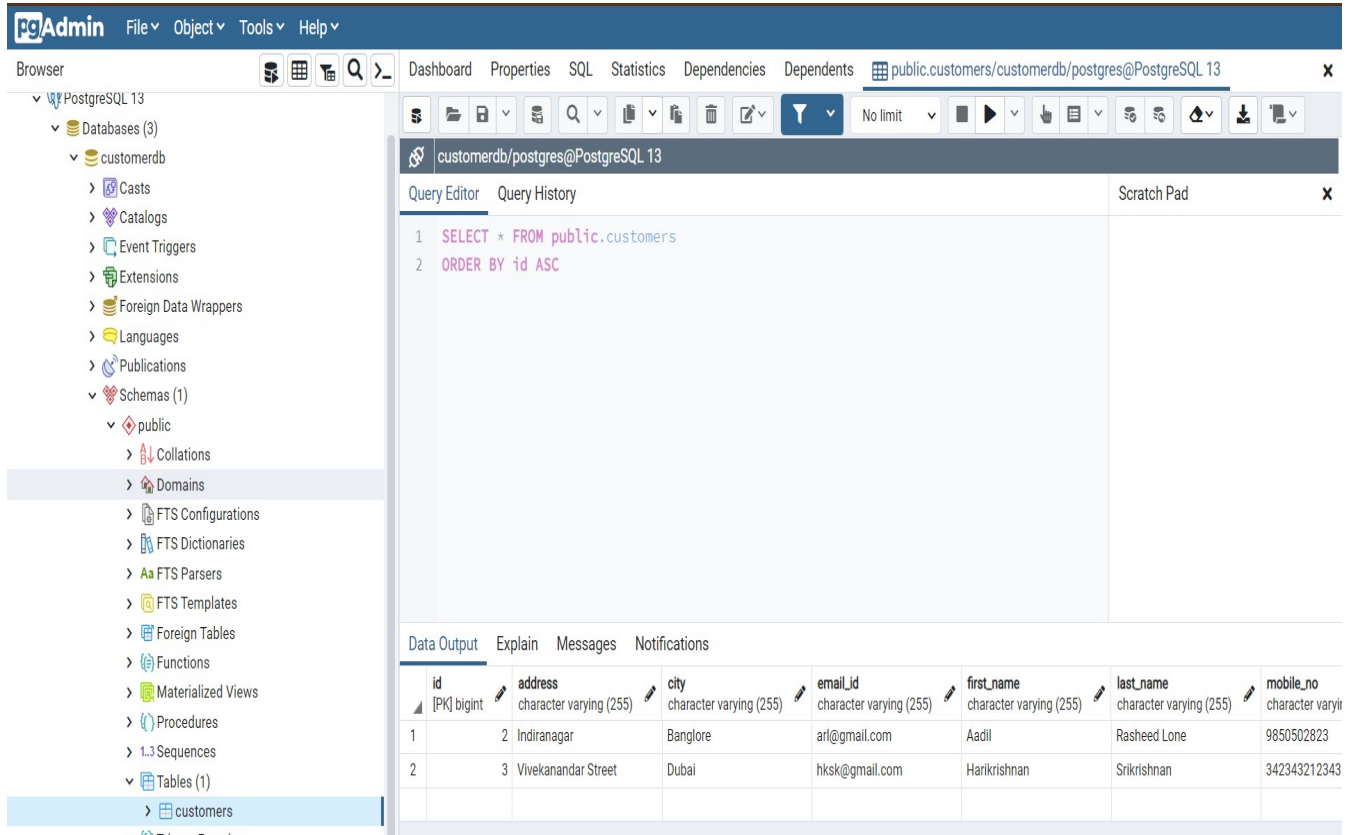
Body Cookies Headers (8) Test Results

Status: 200 OK Time: 28 ms Size: 562 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 3,
4     "firstName": "Harikrishnan",
5     "lastName": "Srikrishnan",
6     "emailId": "hkstk@gmail.com",
7     "city": "Dubai",
8     "address": "Vivekanandar Street",
9     "mobileNo": "342343212343"
10  },
11  {
12    "id": 2,
13    "firstName": "Aadil",
14    "lastName": "Rasheed Lone",
15    "emailId": "arl@gmail.com",
16    "city": "Banglore",
17    "address": "Indiranagar",
18    "mobileNo": "9850502823"
```

SNAPSHOT OF THE PGADMIN IN POSTGRESQL:



pgAdmin File Object Tools Help

Browser PostgreSQL 13

- Databases (3)
 - customerdb
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (1)
 - customers

customerdb/postgres@PostgreSQL 13

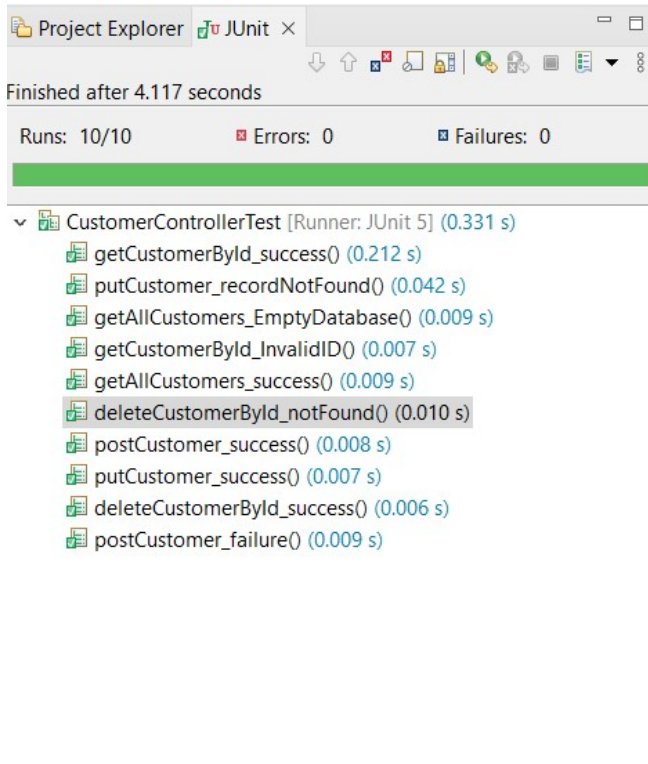
Query Editor Query History Scratch Pad

```
1 SELECT * FROM public.customers
2 ORDER BY id ASC
```

Data Output Explain Messages Notifications

| id | address | city | email_id | first_name | last_name | mobile_no |
|-------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| [PK] bigint | character varying (255) | character varying (255) | character varying (255) | character varying (255) | character varying (255) | character varying (255) |
| 1 | Indiranagar | Bangalore | arl@gmail.com | Aadil | Rasheed Lone | 9850502823 |
| 2 | Vivekanandar Street | Dubai | hksk@gmail.com | Harikrishnan | Srikrishnan | 342343212343 |

Unit Testing:



Project Explorer JUnit

Finished after 4.117 seconds

Runs: 10/10 Errors: 0 Failures: 0

CustomerControllerTest [Runner: JUnit 5] (0.331 s)

- getCustomerById_success() (0.212 s)
- putCustomer_recordNotFound() (0.042 s)
- getAllCustomers_EmptyDatabase() (0.009 s)
- getCustomerById_InvalidID() (0.007 s)
- getAllCustomers_success() (0.009 s)
- deleteCustomerById_notFound() (0.010 s)
- postCustomer_success() (0.008 s)
- putCustomer_success() (0.007 s)
- deleteCustomerById_success() (0.006 s)
- postCustomer_failure() (0.009 s)

| Testcase | Purpose | Result |
|------------------------------------|---|--------|
| 1. getAllCustomers_success() | To check if all customer record is retrieved and status code is OK | ✓ |
| 2. getAllCustomers_EmptyDatabase() | Checking if NoContent status is sent if the database is empty. | ✓ |
| 3. getCustomerById_success() | Testing if the getCustomerById works correctly and returns success status code if given a valid ID. | ✓ |
| 4. getCustomerById_InvalidId() | Testing if the getCustomerById method returns not found status given an incorrect id. | ✓ |
| 5. postCustomer_success() | Checking if a new customer record is created successfully given the valid info | ✓ |
| 6. postCustomer_failure() | Checking if creating a incorrect record results in bad request. | ✓ |
| 7. putCustomer_success() | Checking if put method is working perfectly if given the correct id to be updated. | ✓ |
| 8. putCustomer_recordNotFound() | Checking if not found status is sent if incorrect id is given. | ✓ |
| 9. deleteCustomerById_success | Checking if deletion works fine if given a valid ID. | ✓ |
| 10. deleteCustomerById_notFound() | Checking if bad request status is sent if given a invalid id | ✓ |

___ THE END ___