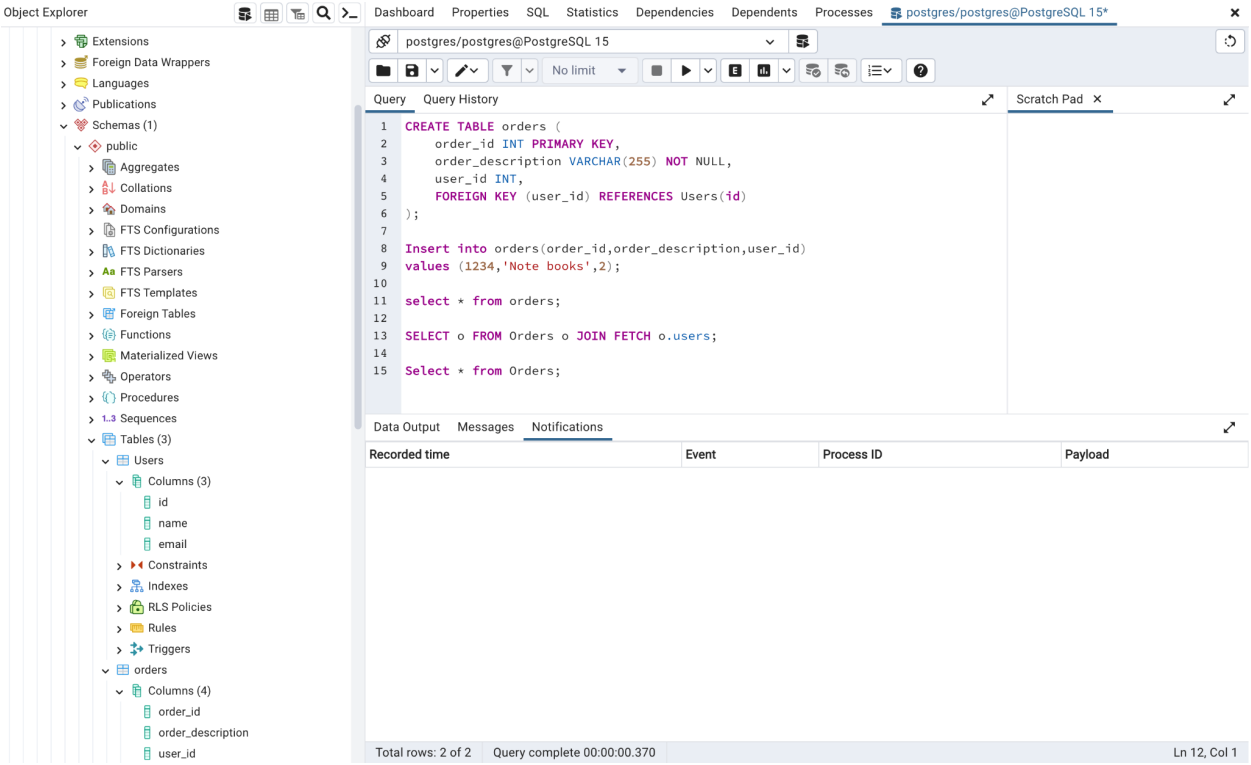
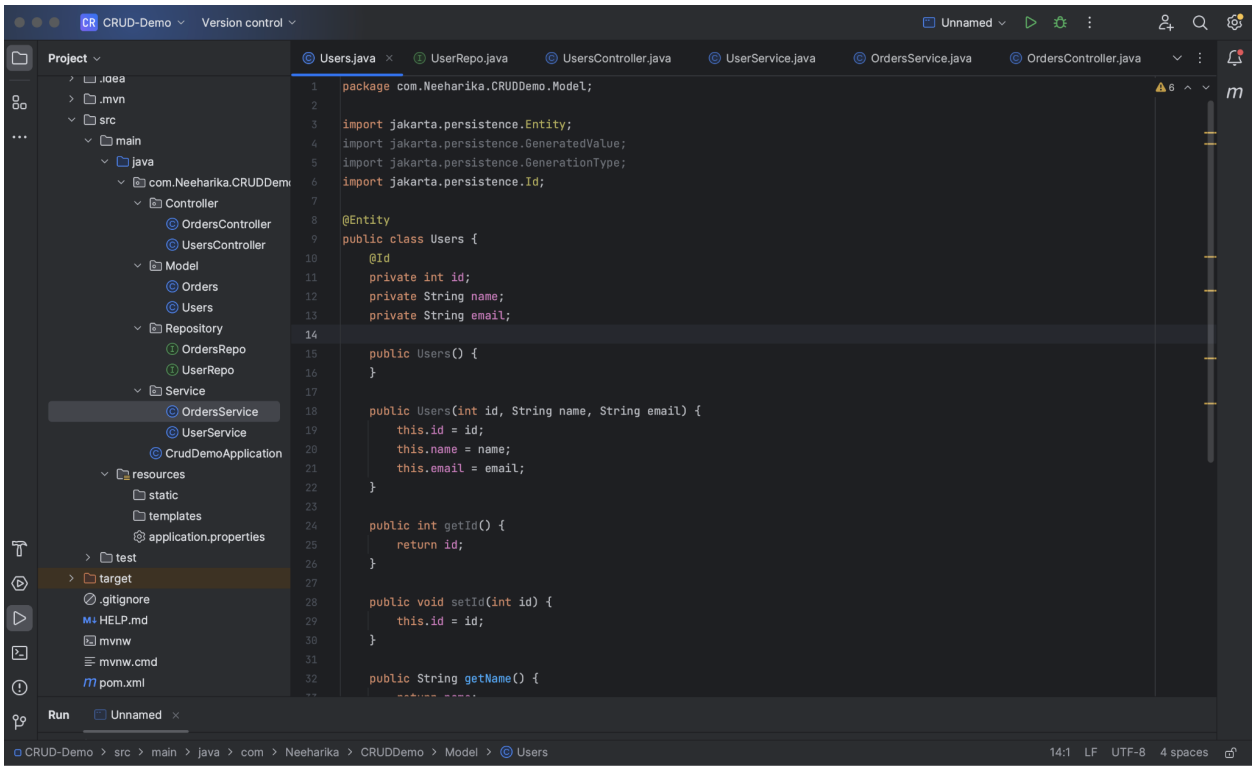
Advanced DataBase Systems- Project

Adella Neeharika(nadella@kent.edu)

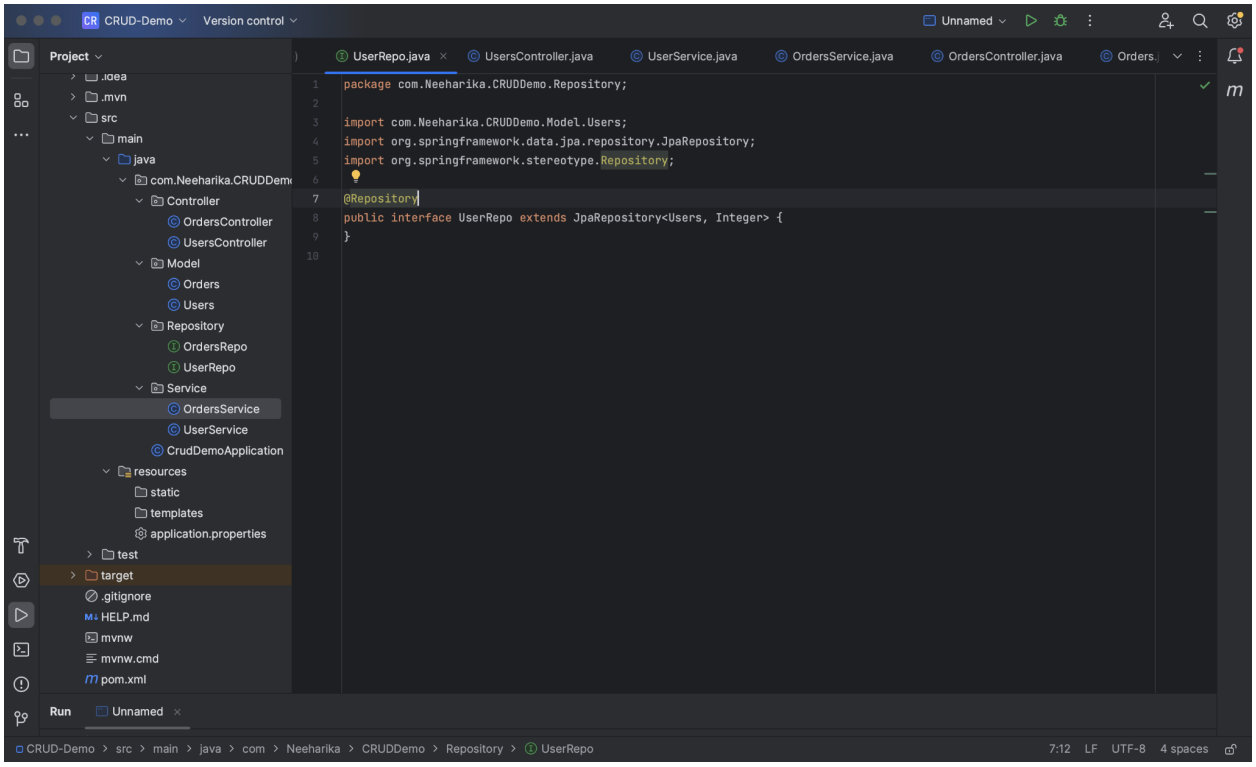
This is the Postgres Admin application(pgadmin). Here I have used Users and Orders tables.

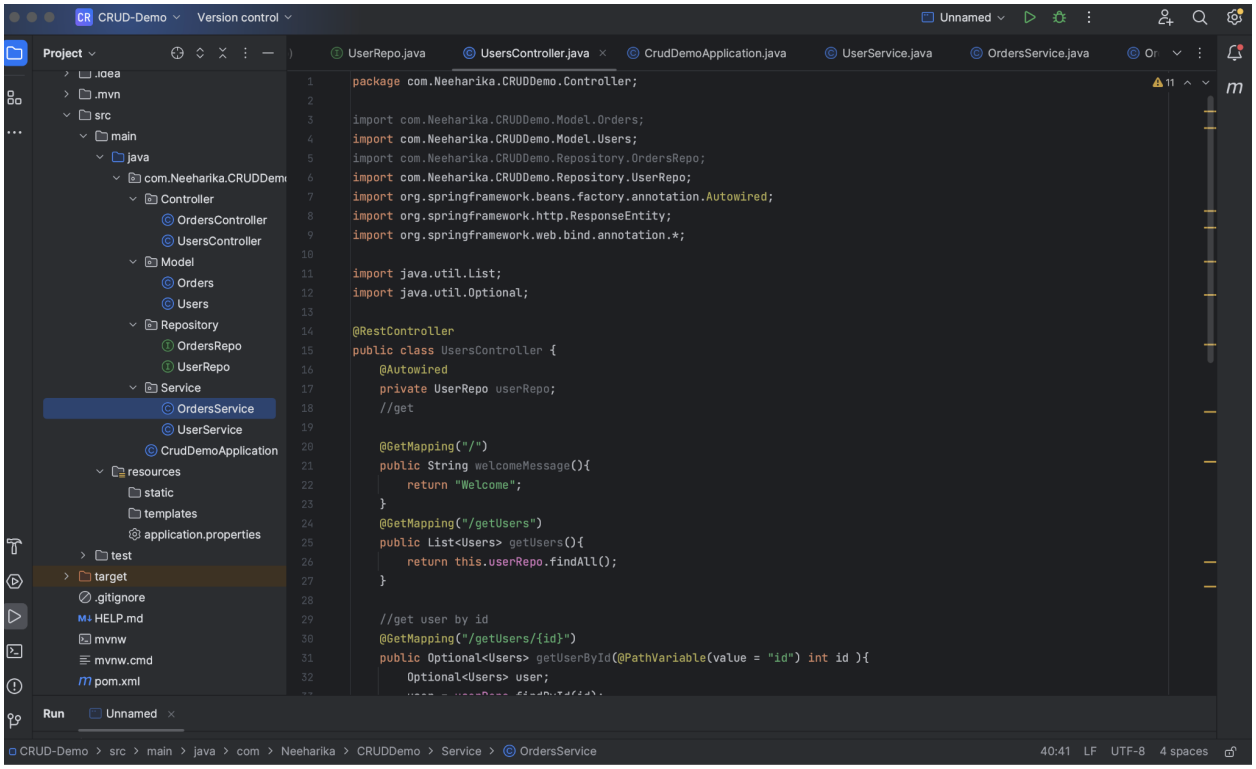


Below attached are Users tables's model class, controller class, repository class

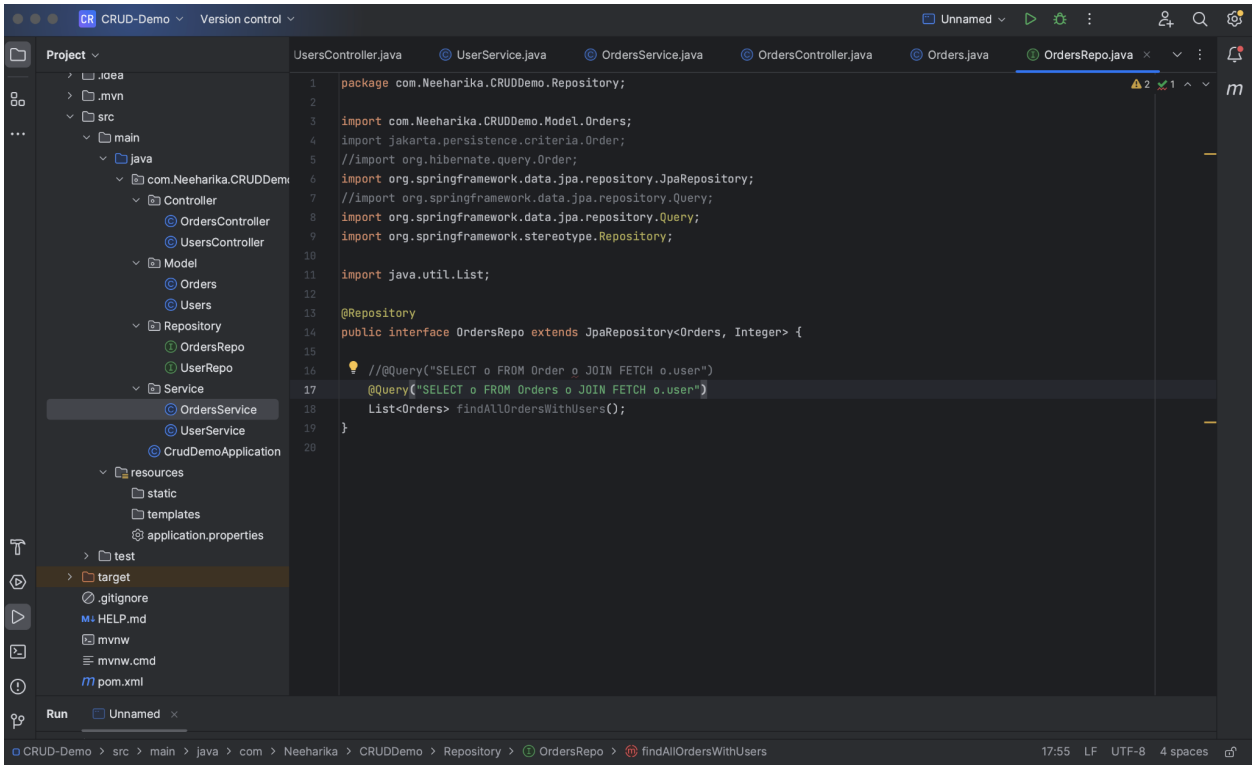


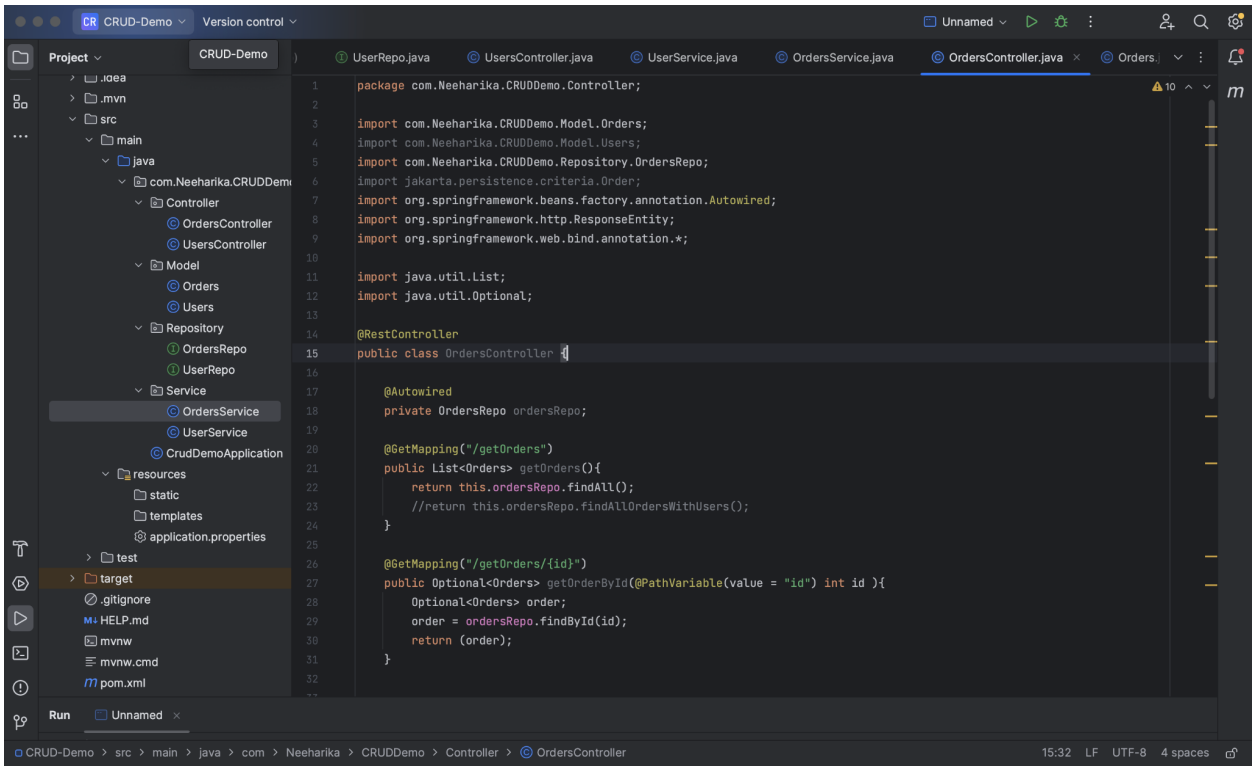

```java
package com.Neeharika.CRUDDemo.Model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Users {
    @Id
    private int id;
    private String name;
    private String email;

    public Users() {
    }

    public Users(int id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
```

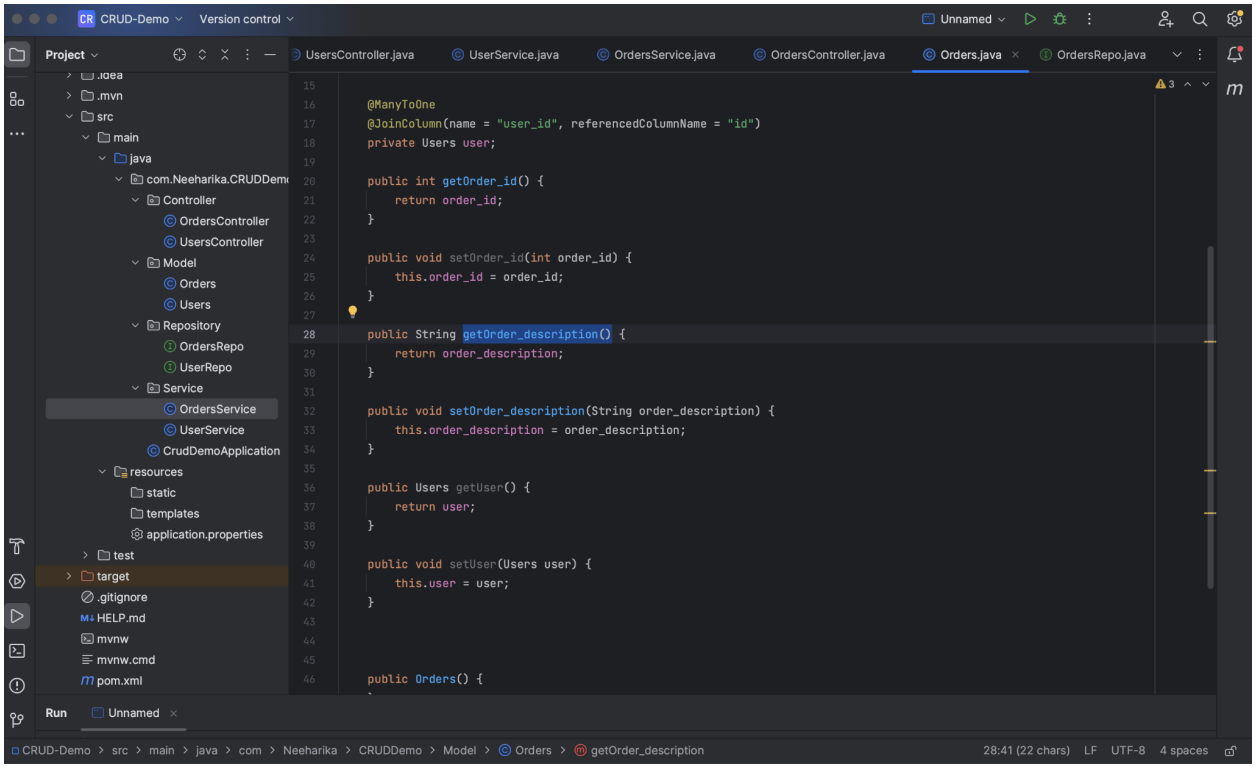

CRUD-Demo
Version control
Unnamed
Project
UserRepo.java
UsersController.java
UserService.java
OrdersService.java
OrdersController.java
Orders.
CRUD-Demo
src
main
java
com.Neeharika.CRUDDemo
Controller
OrdersController
UsersController
Model
Orders
Users
Repository
OrdersRepo
UserRepo
Service
OrdersService
UserService
CrudDemoApplication
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
1 package com.Neeharika.CRUDDemo.Repository;
2
3 import com.Neeharika.CRUDDemo.Model.Users;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface UserRepo extends JpaRepository<Users, Integer> {
9 }
10
Run
Unnamed
CRUD-Demo > src > main > java > com > Neeharika > CRUDDemo > Repository > UserRepo
7:12 LF UTF-8 4 spaces


CRUD-Demo
Version control
Unnamed
Project
UserRepo.java
UsersController.java
CrudDemoApplication.java
UserService.java
OrdersService.java
CRUD-Demo
src
main
java
com.Neeharika.CRUDDemo
Controller
OrdersController
UsersController
Model
Orders
Users
Repository
OrdersRepo
UserRepo
Service
OrdersService
UserService
CrudDemoApplication
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
1 package com.Neeharika.CRUDDemo.Controller;
2
3 import com.Neeharika.CRUDDemo.Model.Orders;
4 import com.Neeharika.CRUDDemo.Model.Users;
5 import com.Neeharika.CRUDDemo.Repository.OrdersRepo;
6 import com.Neeharika.CRUDDemo.Repository.UserRepo;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.web.bind.annotation.*;
10
11 import java.util.List;
12 import java.util.Optional;
13
14 @RestController
15 public class UsersController {
16 @Autowired
17 private UserRepo userRepo;
18 //get
19
20 @GetMapping("/")
21 public String welcomeMessage(){
22 return "Welcome";
23 }
24 @GetMapping("/getUsers")
25 public List<Users> getUsers(){
26 return this.userRepo.findAll();
27 }
28
29 //get user by id
30 @GetMapping("/getUsers/{id}")
31 public Optional<Users> getUserById(@PathVariable(value = "id") int id){
32 Optional<Users> user;
Run
Unnamed
CRUD-Demo > src > main > java > com > Neeharika > CRUDDemo > Service > OrdersService
40:41 LF UTF-8 4 spaces

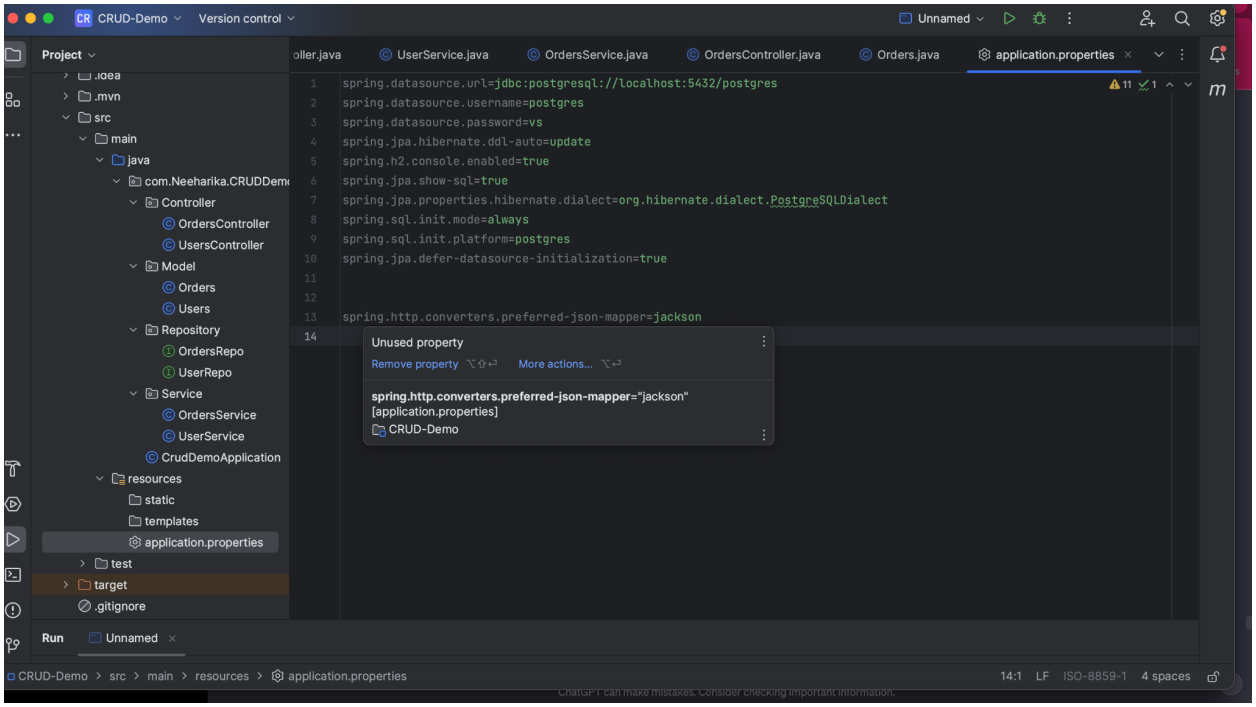
Below attached are Orders table's model,repository and controller classes



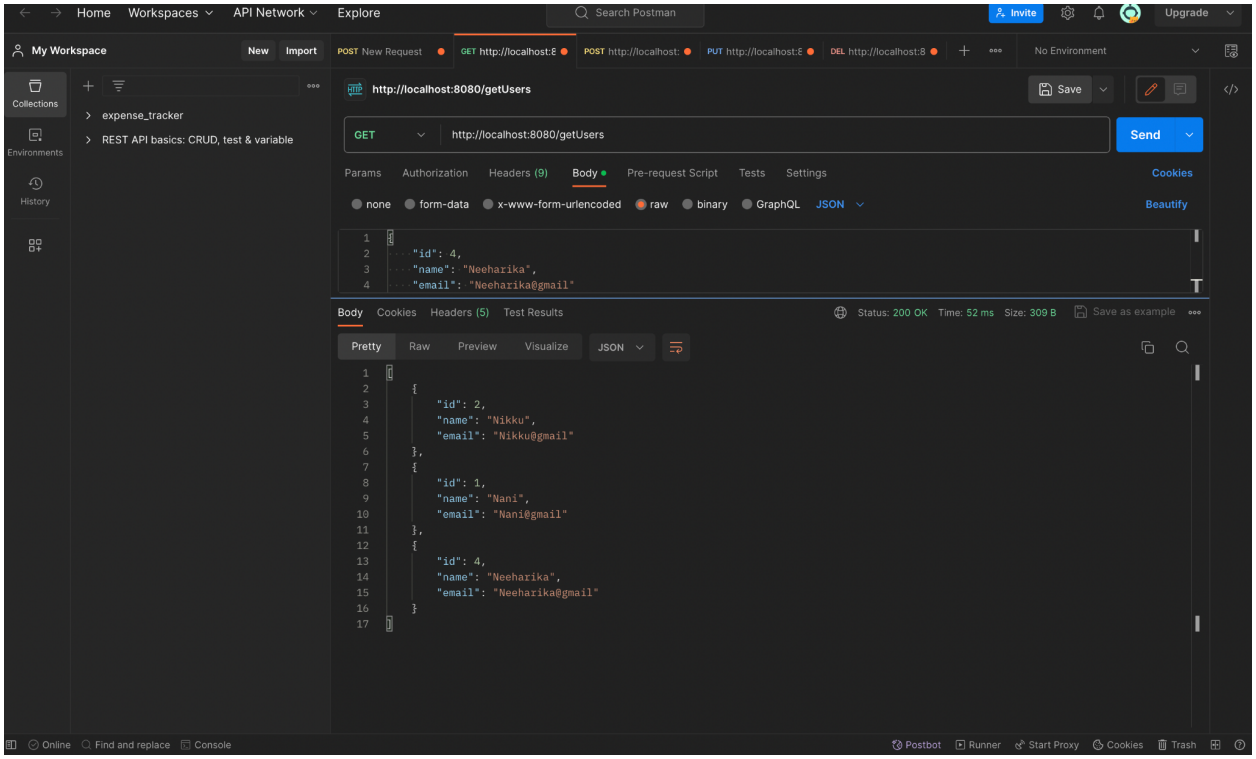




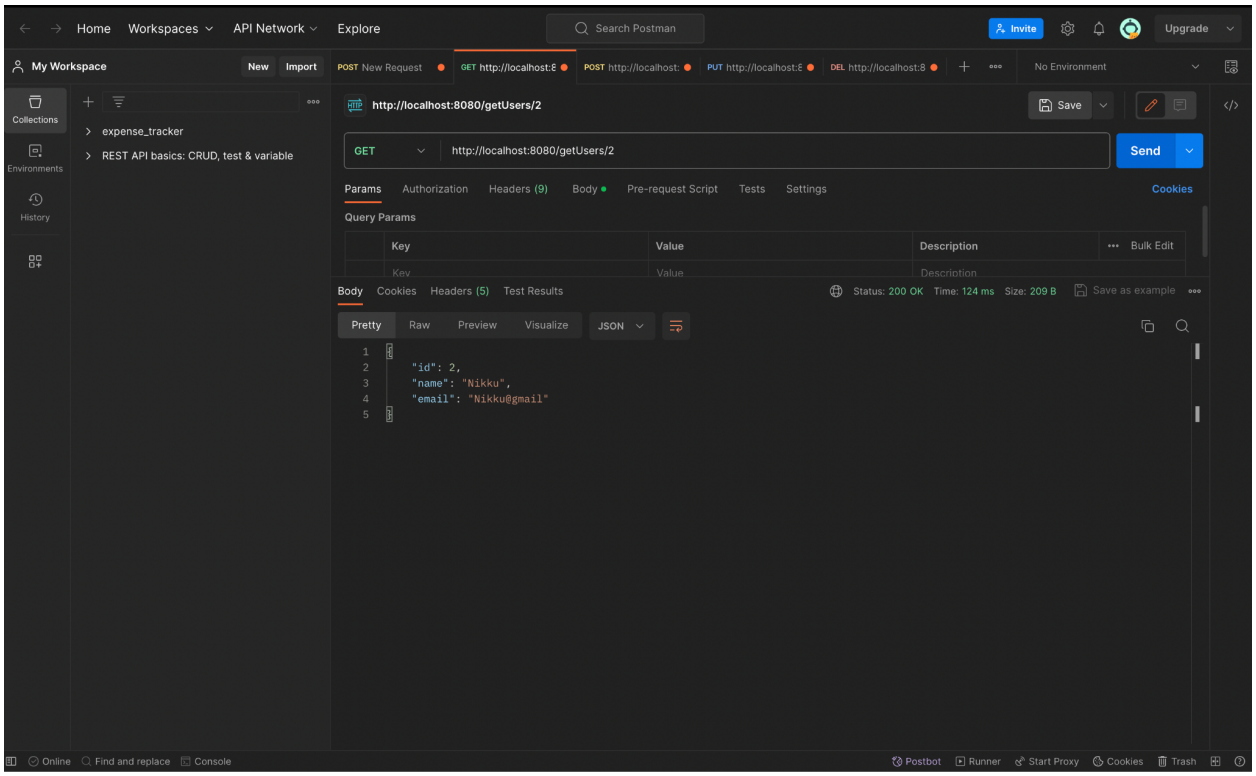
Below attached is the properties of database connection



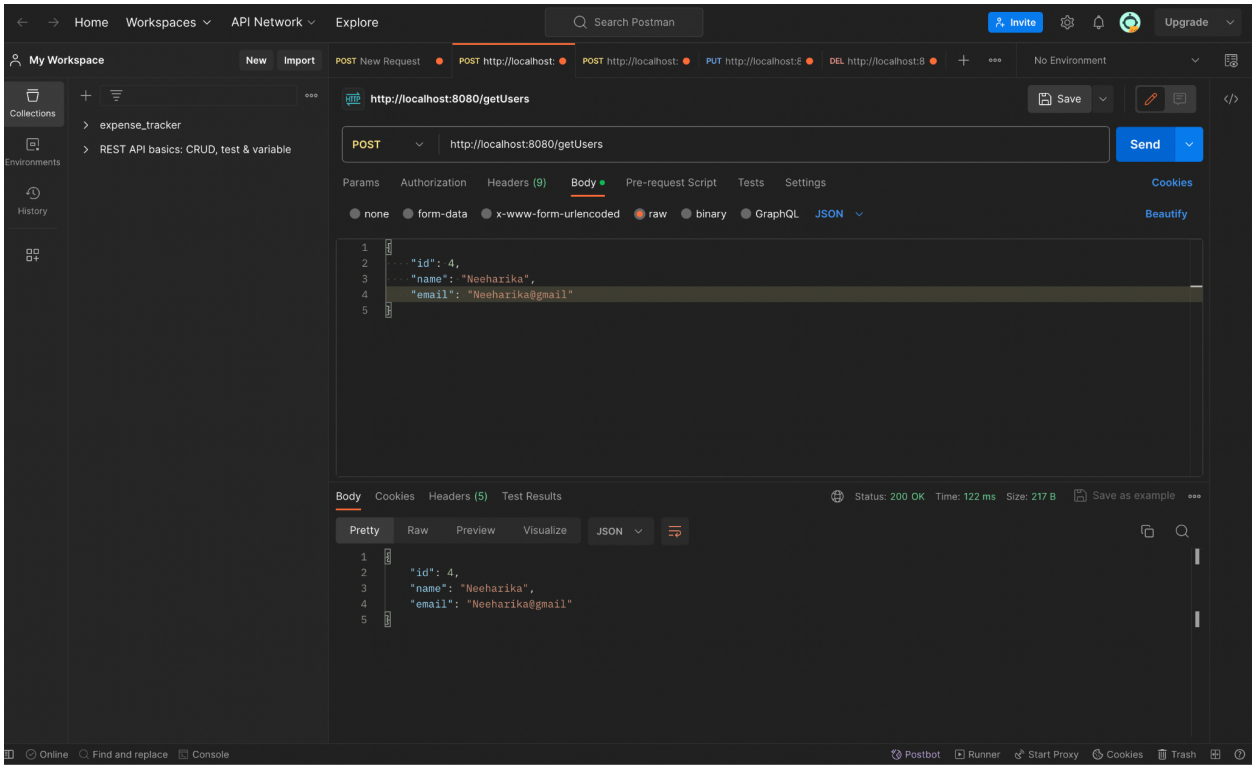
Read All Users



Read particular User

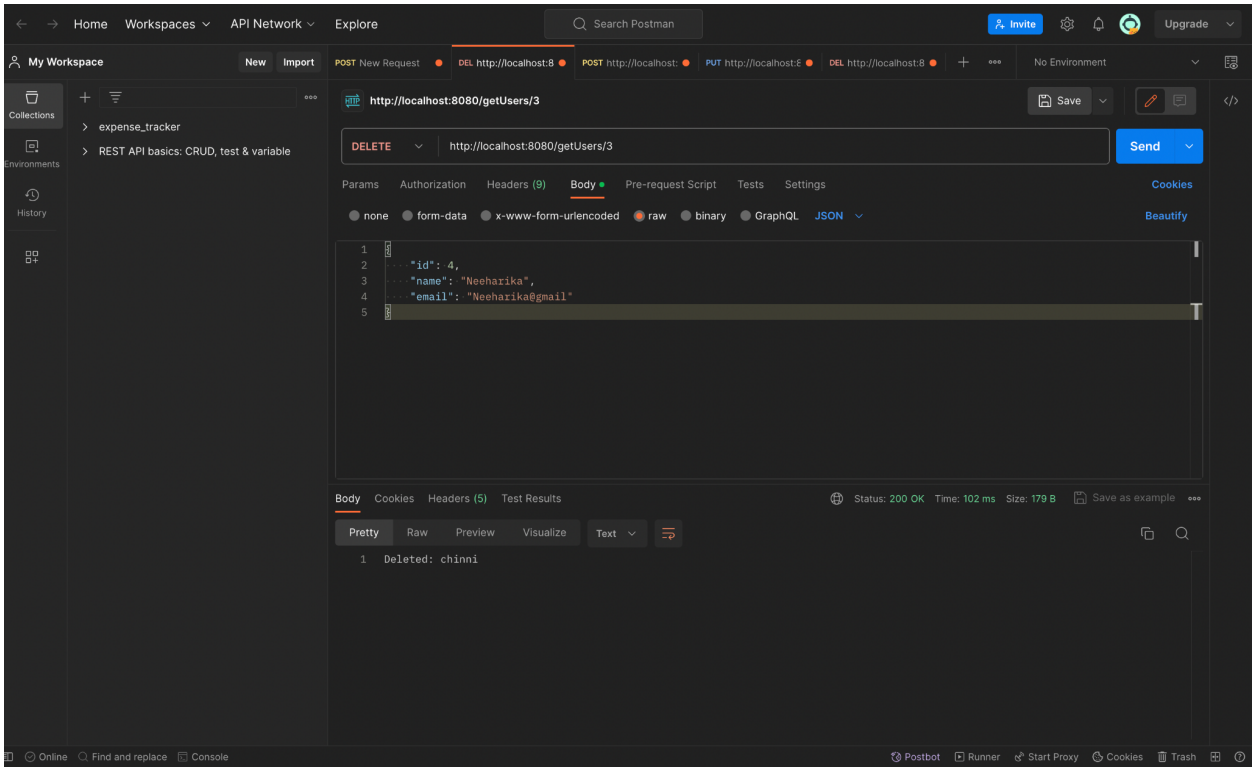


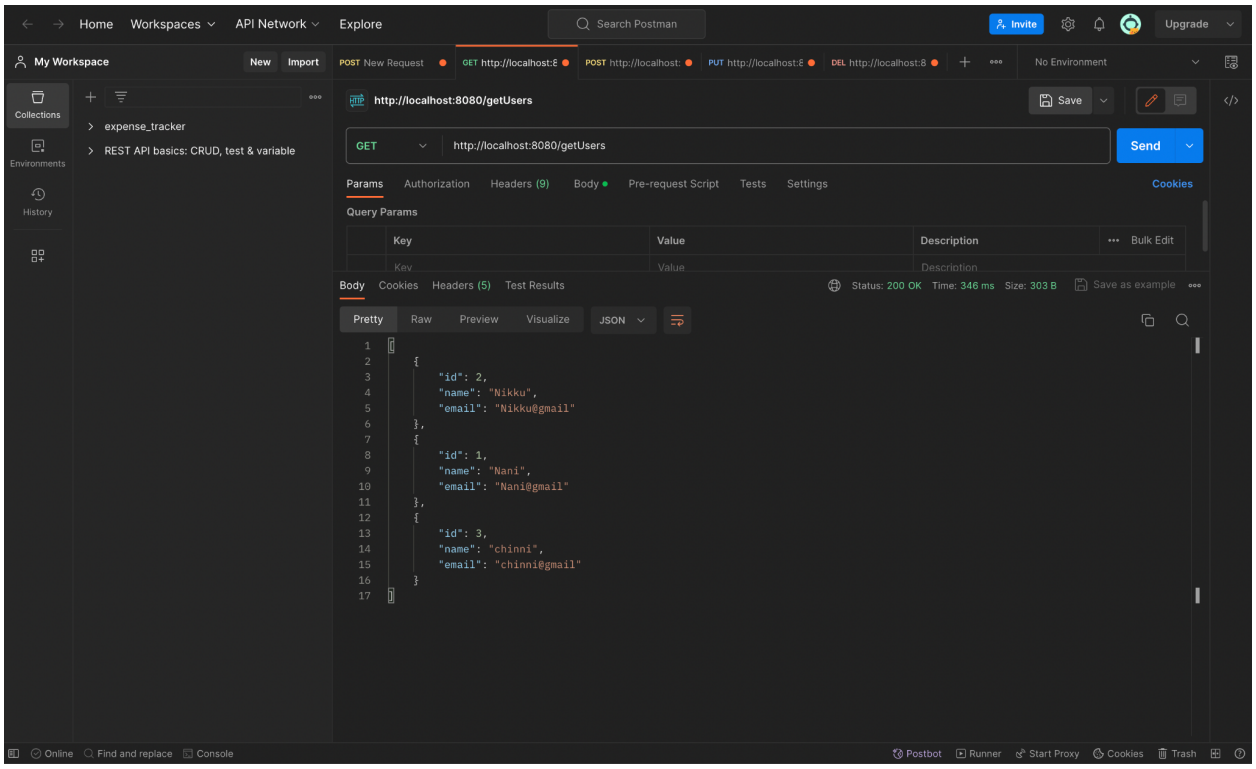
Insert one user



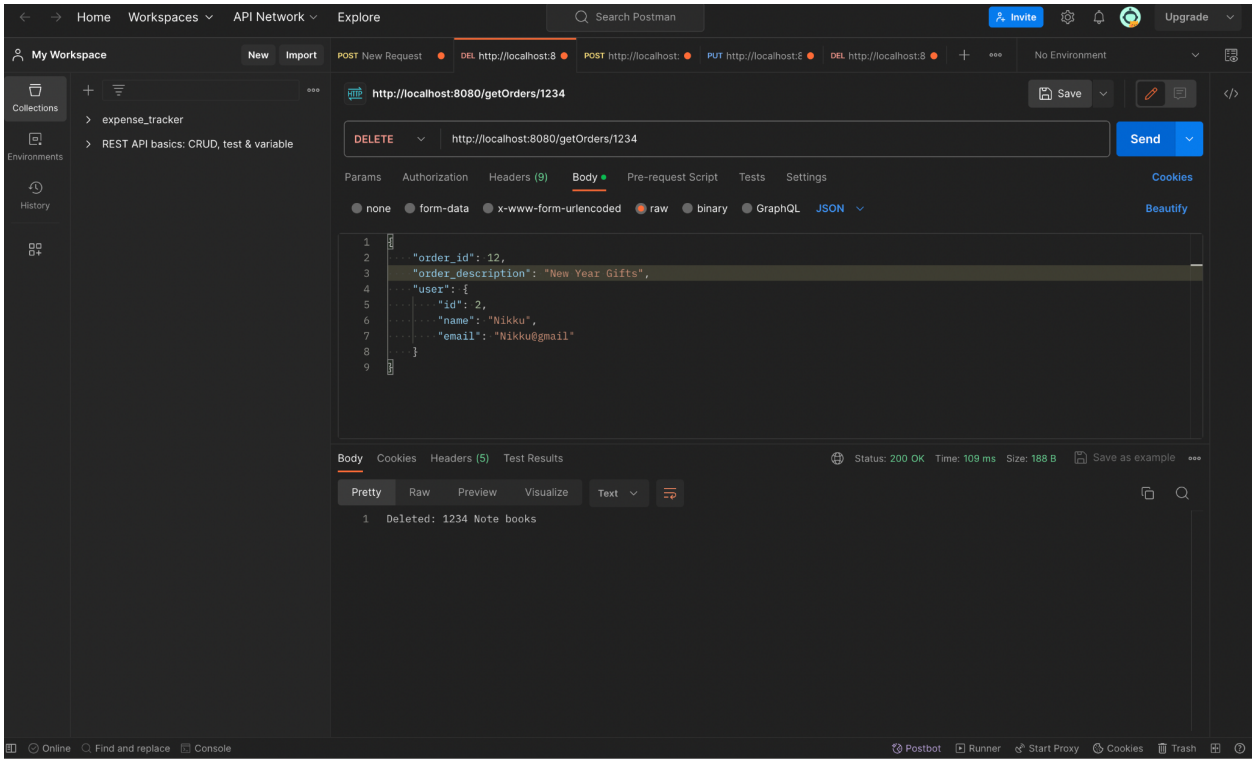
Update user

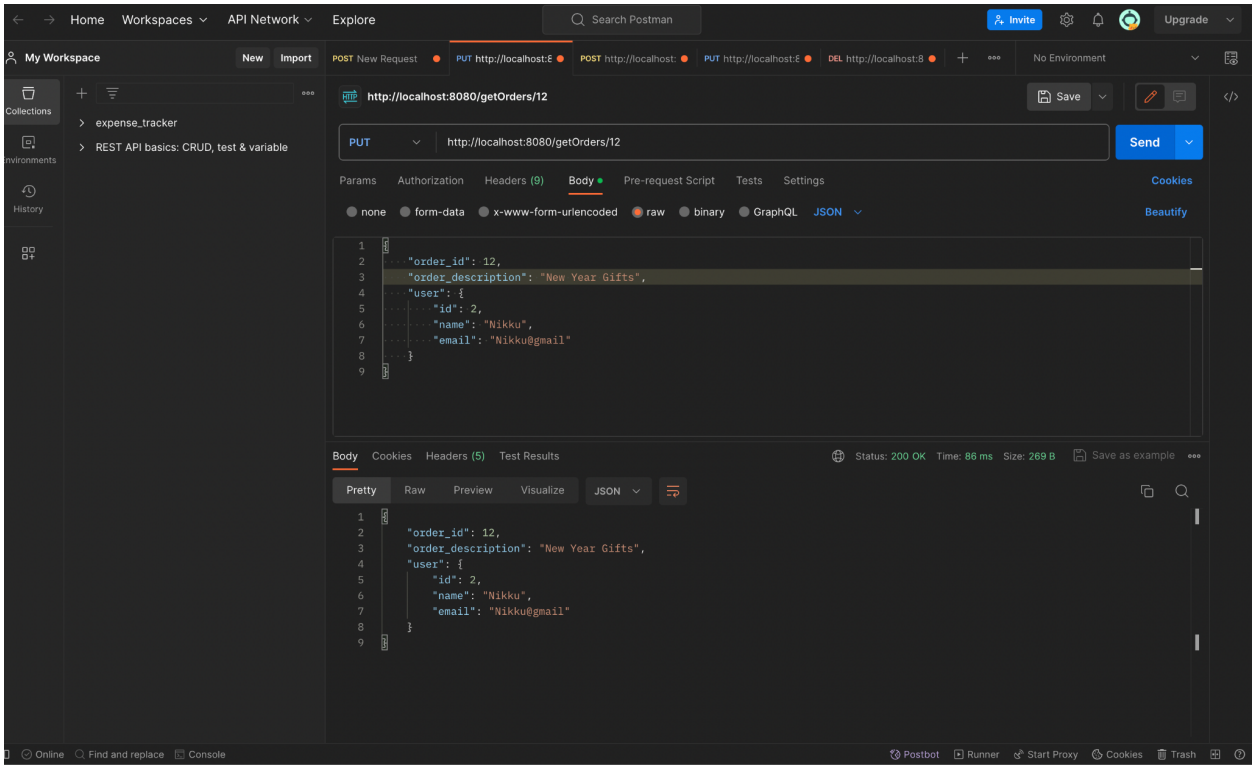
Delete User


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
No Environment
Collections
Environments
History
expense_tracker
REST API basics: CRUD, test & variable
http://localhost:8080/getUsers/3
Save
DELETE
http://localhost:8080/getUsers/3
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 4,
"name": "Neeharika",
"email": "Neeharika@gmail"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 102 ms
Size: 179 B
Save as example
Pretty
Raw
Preview
Visualize
Text
Deleted: chinni
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
No Environment
Collections
Environments
History
expense_tracker
REST API basics: CRUD, test & variable
http://localhost:8080/getUsers
Save
GET
http://localhost:8080/getUsers
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 346 ms
Size: 303 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 2,
"name": "Nikku",
"email": "Nikku@gmail"
},
{
"id": 1,
"name": "Nani",
"email": "Nani@gmail"
},
{
"id": 3,
"name": "chinni",
"email": "chinni@gmail"
}
]
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

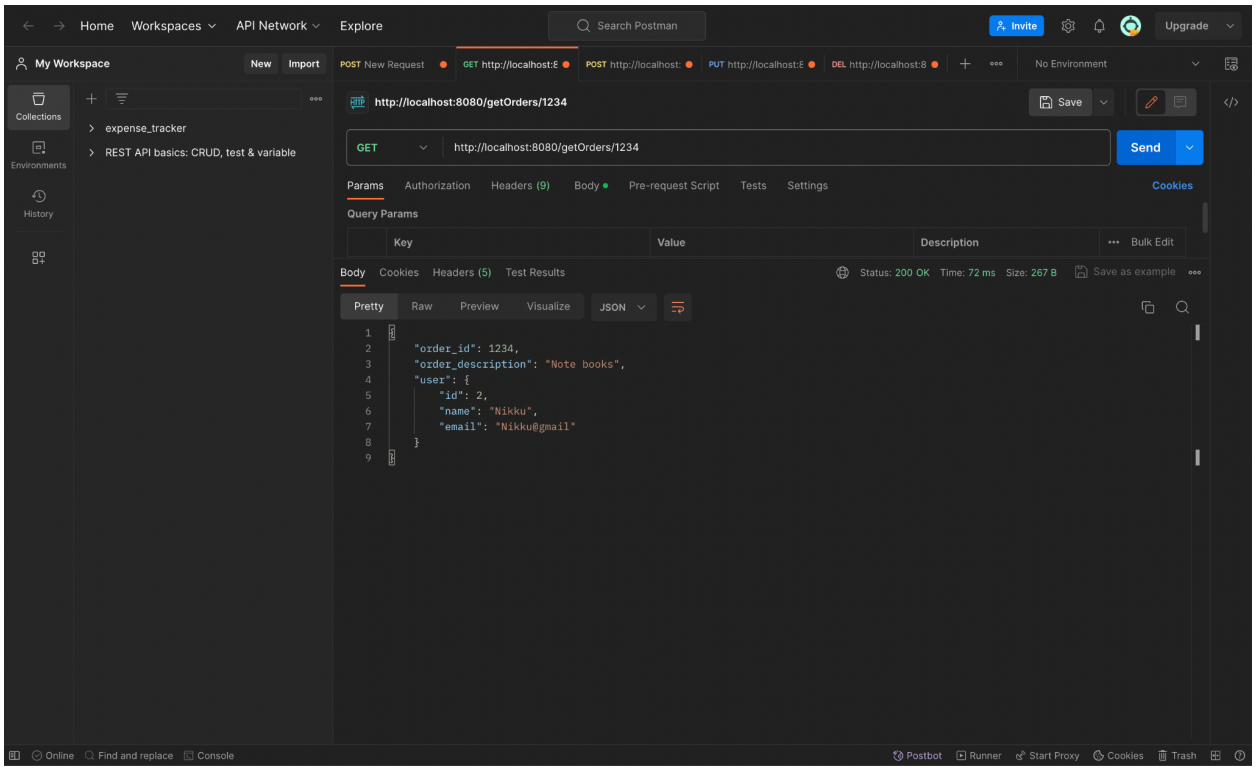
Delete order

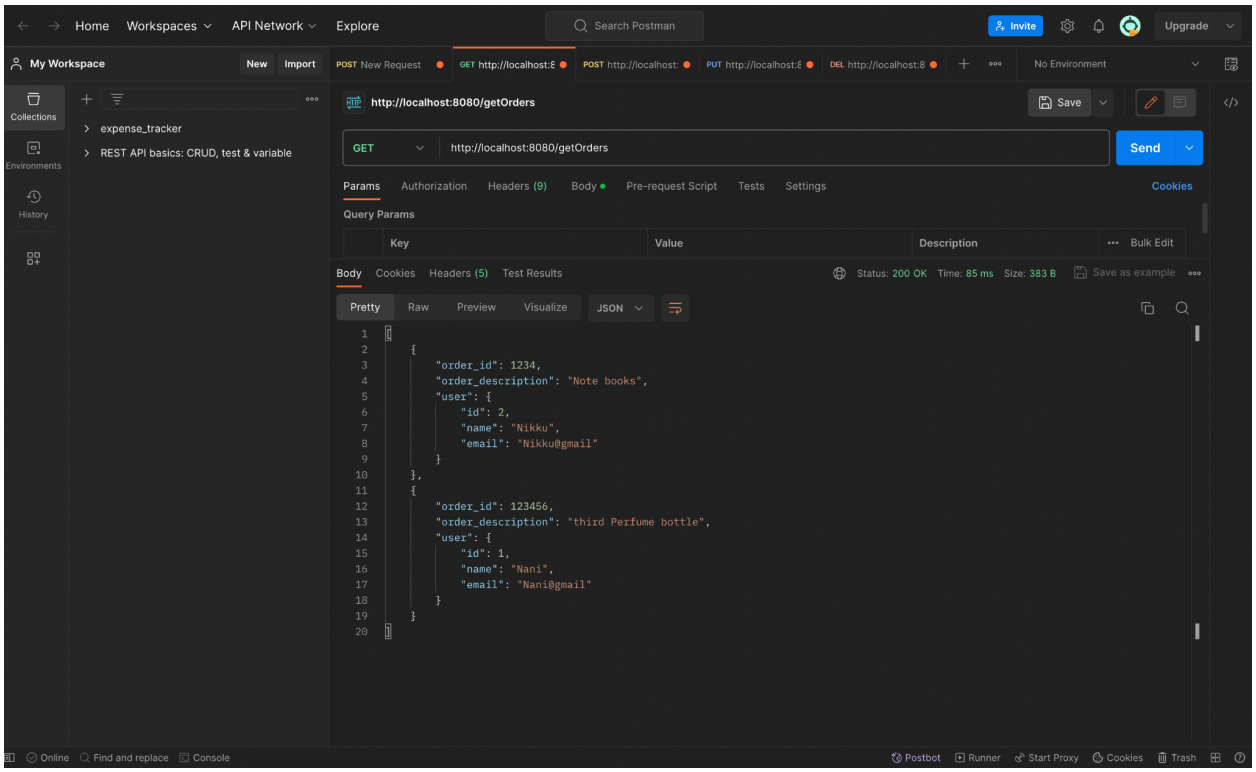




Above is Update particular order

Read particular order





Above is read all orders

1.I have used SpringBoot with Postgres and Postman tool
2.Used two tables- Users and Orders
3.Users(id,name,email)
Orders(order_id,order_description,User_id) user_id references to id in Users table acts as foreign key
4. This is the simple web api, we can integrate it to any service who are in need of it.(I am working on it to add Simple UI to it), I will push my remaining work to GIT.
5. The reason why I used two tables is: I don't want to make ambiguity by putting orders and users in the same table, another one is I want to create many to one relations from orders to user because users may have many orders.
The last important one is, i wanted to show relations between them maintain scalability and achieve Normalisation.

6. I have implemented controllers for CRUD operations on Users and Orders separately. You can create a user separately but when you create an Order, you have to associate its user as well(Thats is how I have implemented these operations).

7.The reasons why i choose this database is that PostgreSQL is reliable and powerful dbms, spring boot is known for its enterprise
grade features, scalability, it provides comprehensive ecosystem that includes Spring Data JPA for easy database integrations
which simplifies CRUD operations.