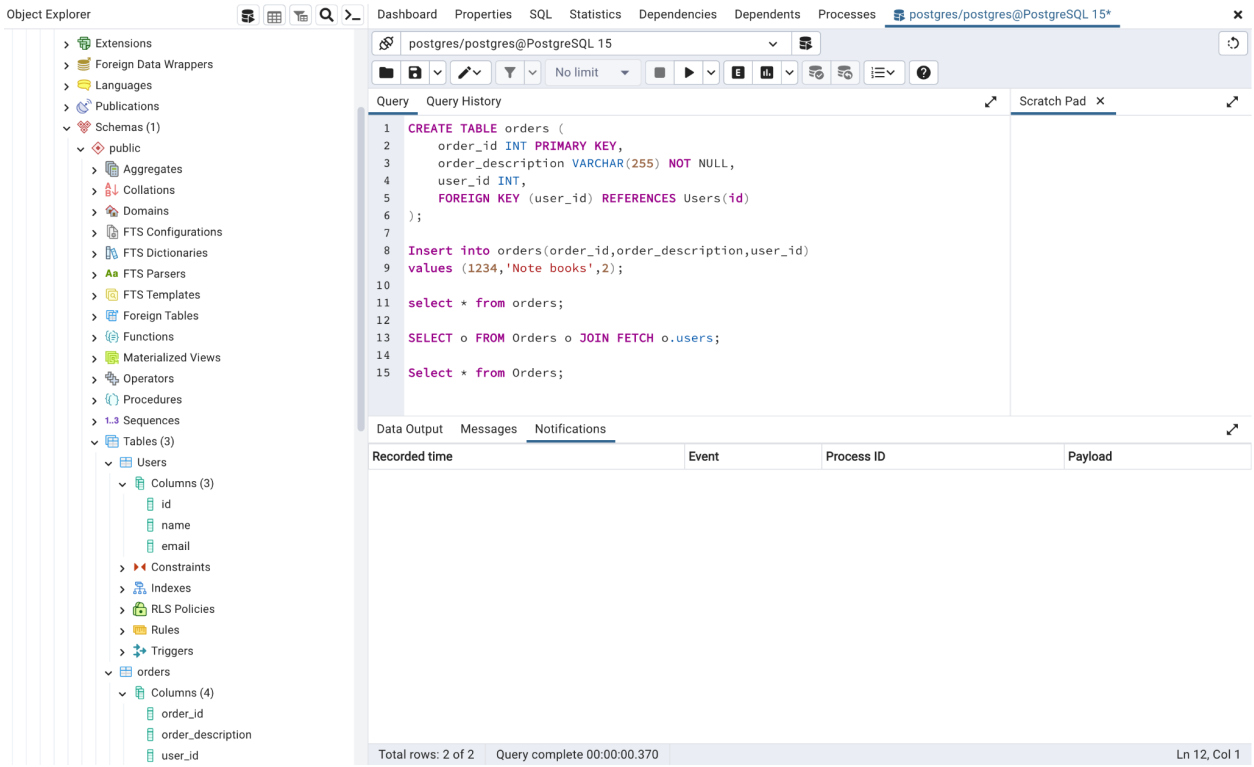
Advanced DataBase Systems- Project

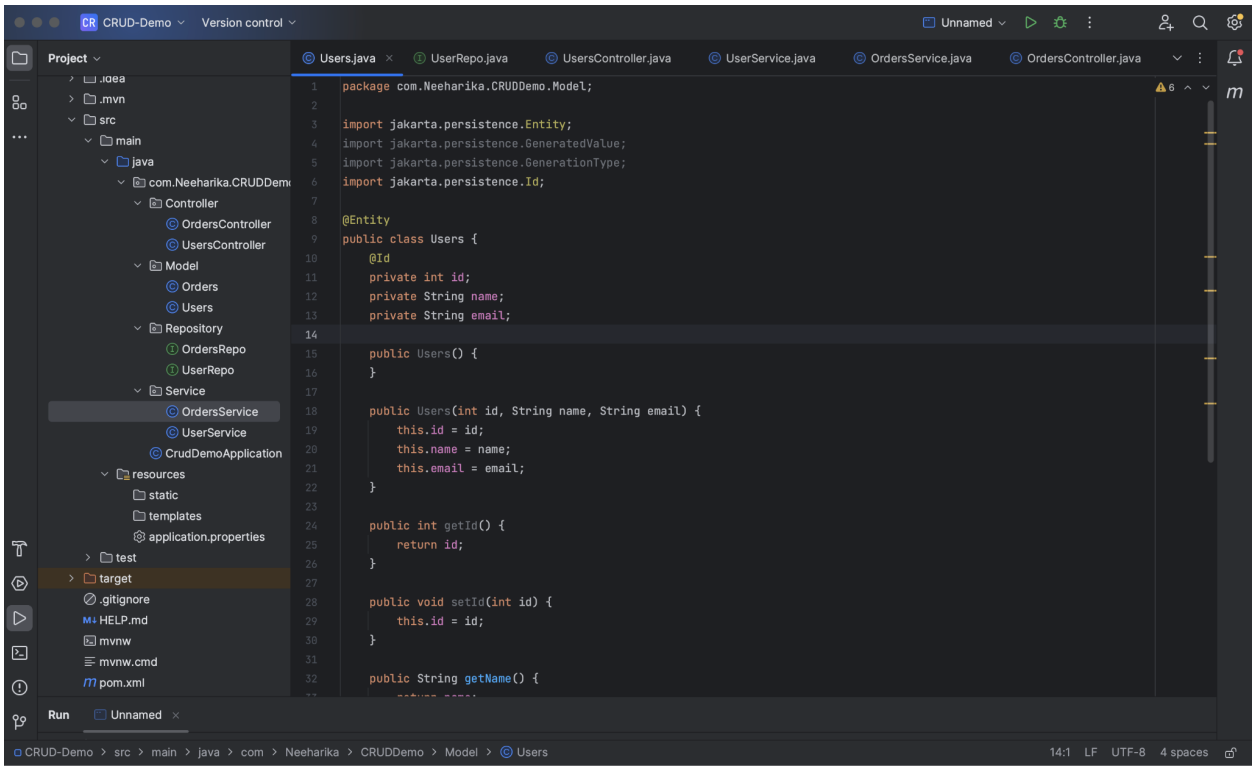
Adella Neeharika(nadella@kent.edu)

GitHub repository Url :https://github.com/neehariko/ADBS_Project.git

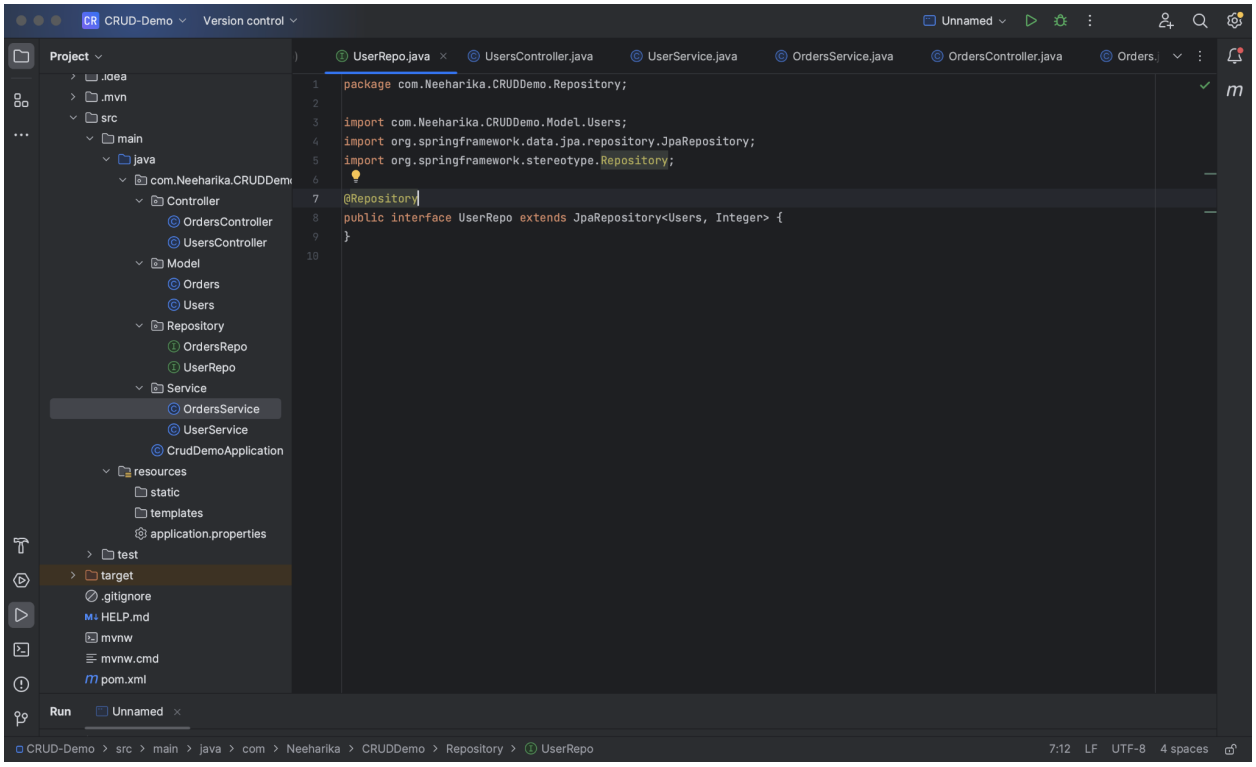
This is the Postgres Admin application(pgadmin). Here I have used Users and Orders tables.

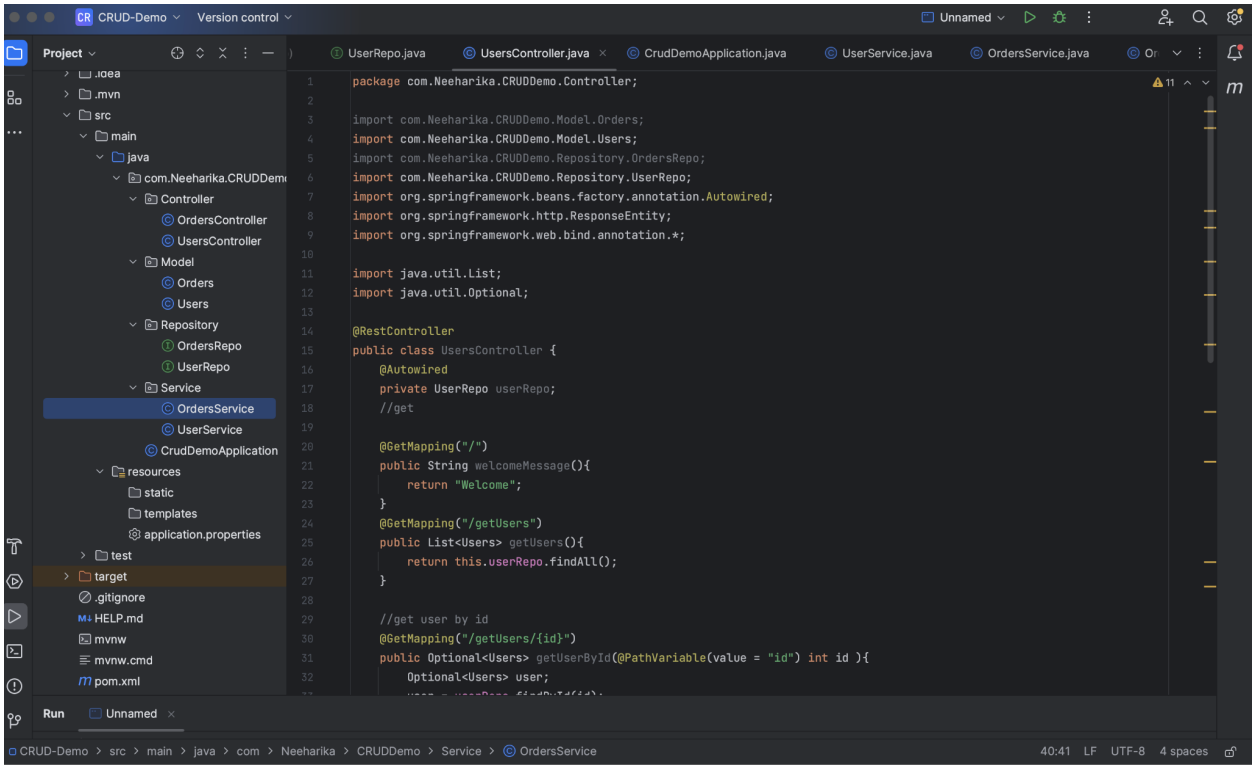


Below attached are Users tables's model class, controller class, repository class

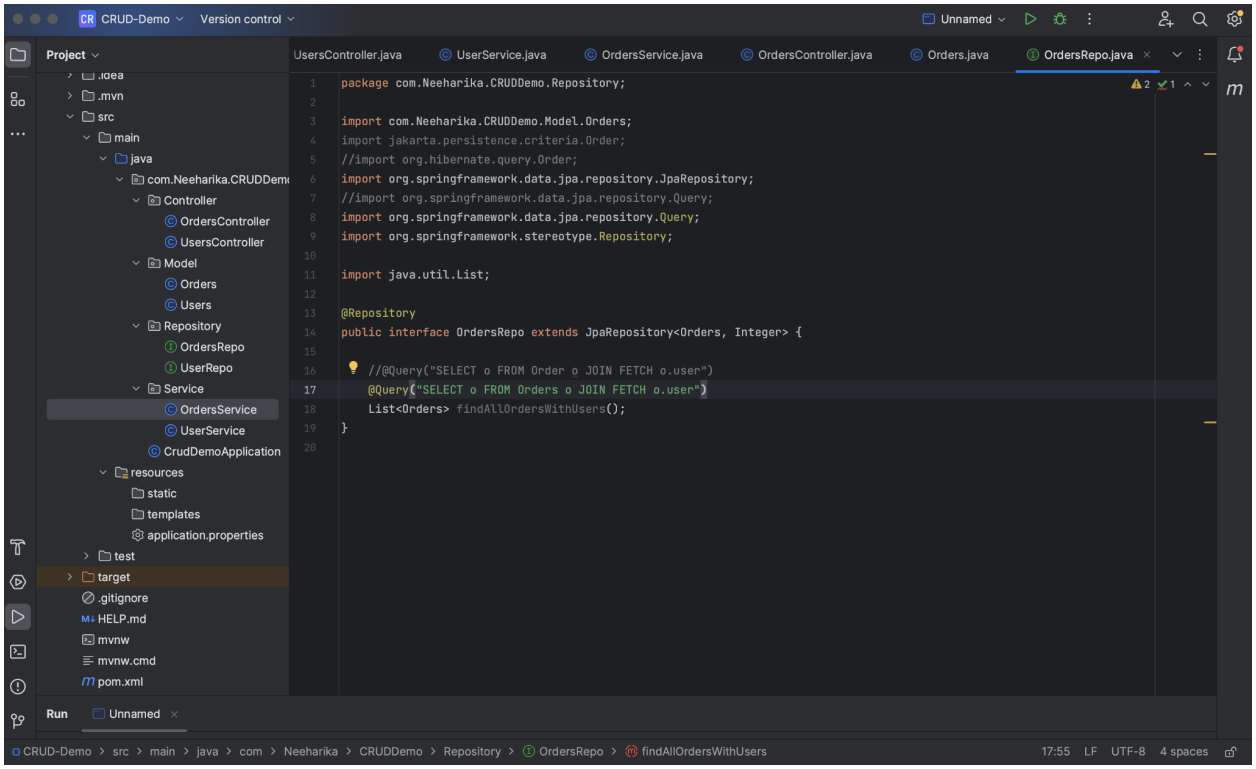


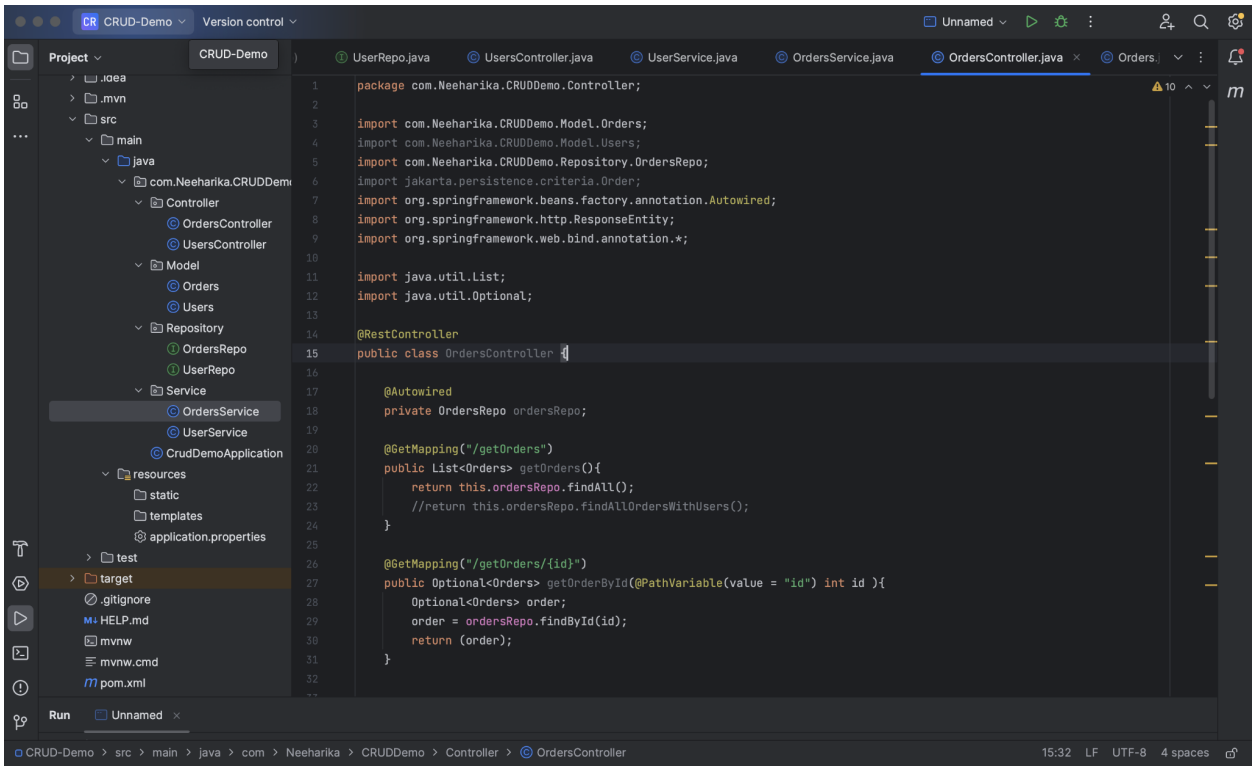

```java
package com.Neeharika.CRUDDemo.Model;

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() {
```

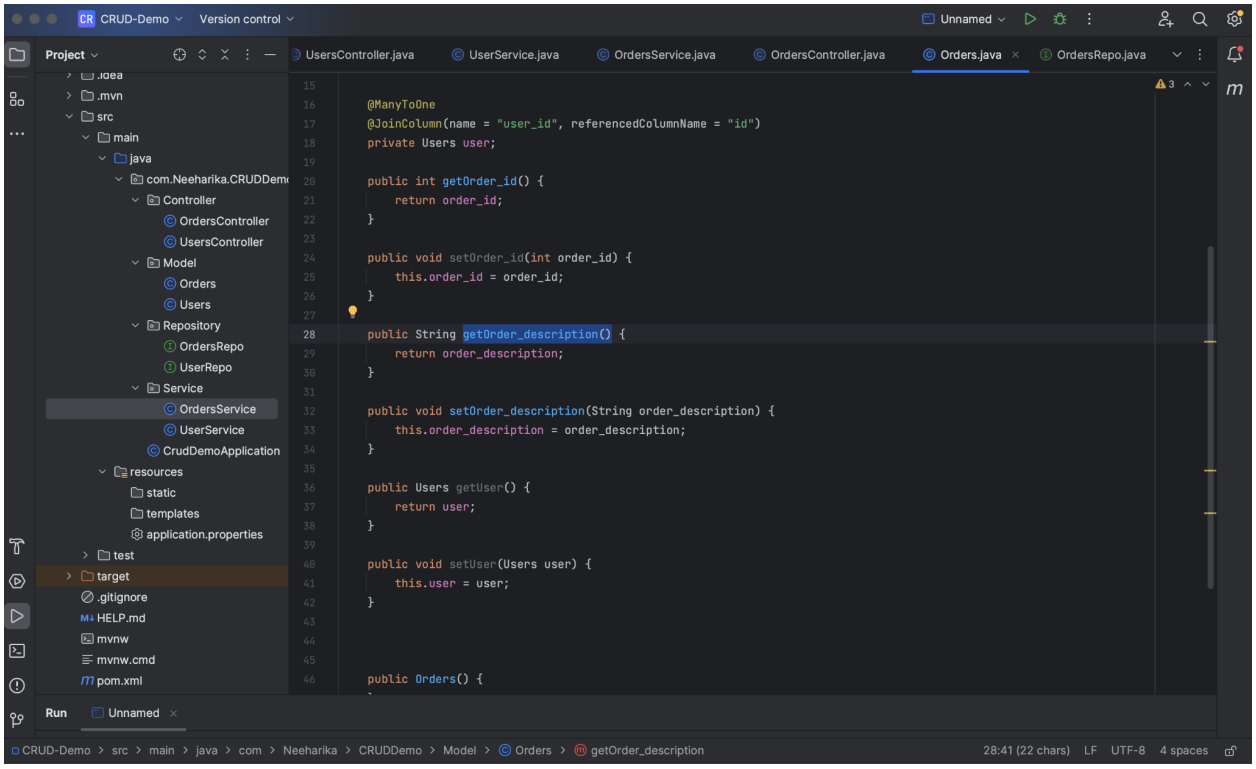

CRUD-Demo
Version control
Unnamed
Project
UserRepo.java
UsersController.java
UserService.java
OrdersService.java
OrdersController.java
Orders.
CRUD-Demo
src
main
java
com.Neeharika.CRUDDemo
Controller
OrdersController
UsersController
Model
Orders
Users
Repository
OrdersRepo
UserRepo
Service
OrdersService
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

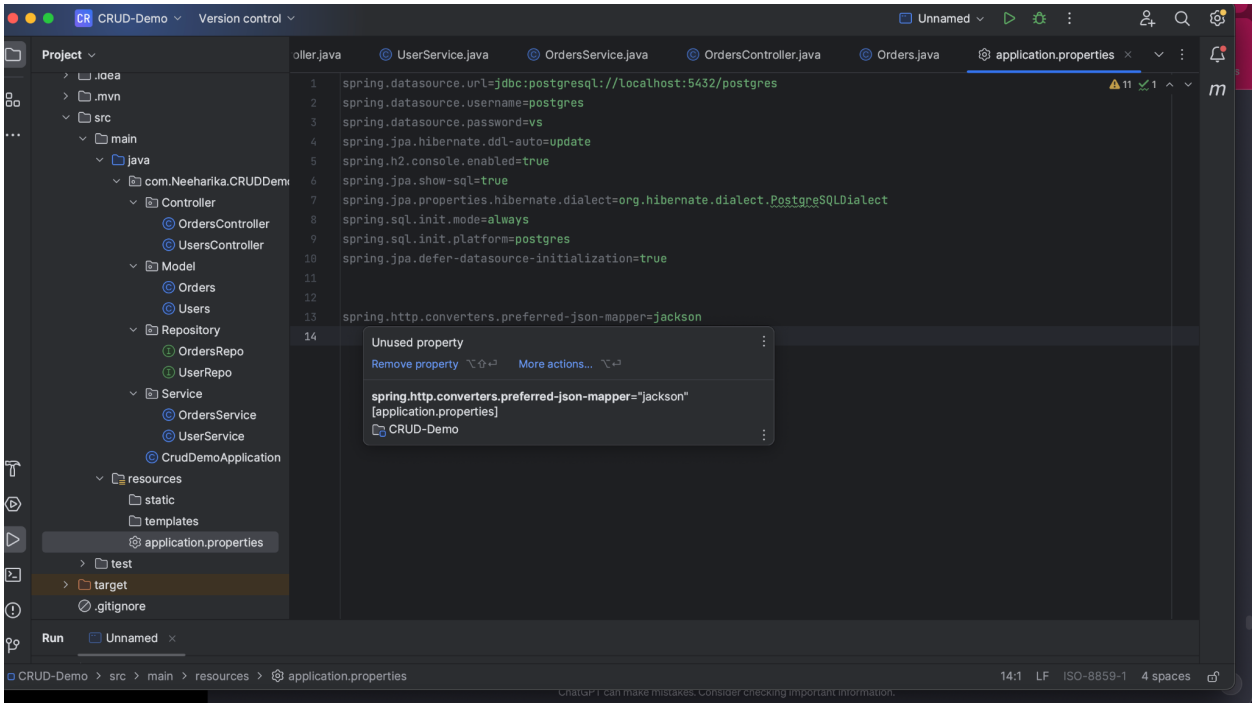
Below attached are Orders table's model,repository and controller classes



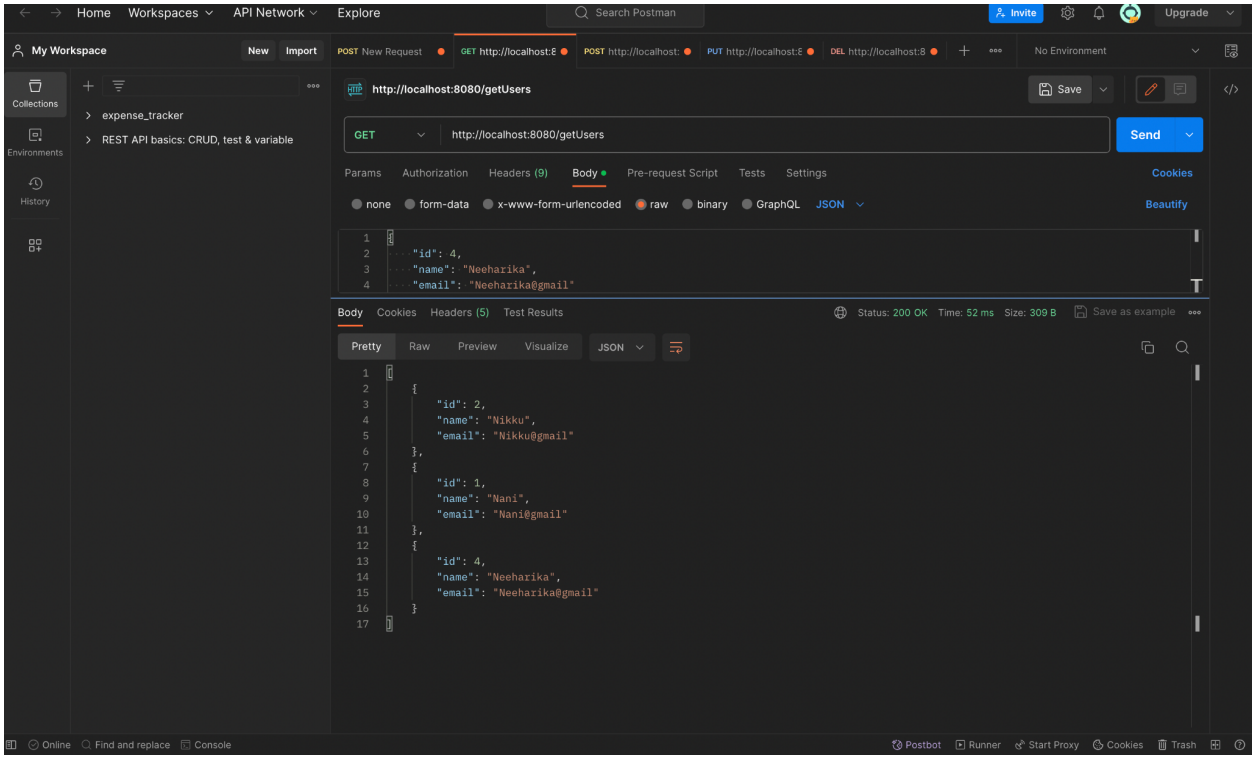




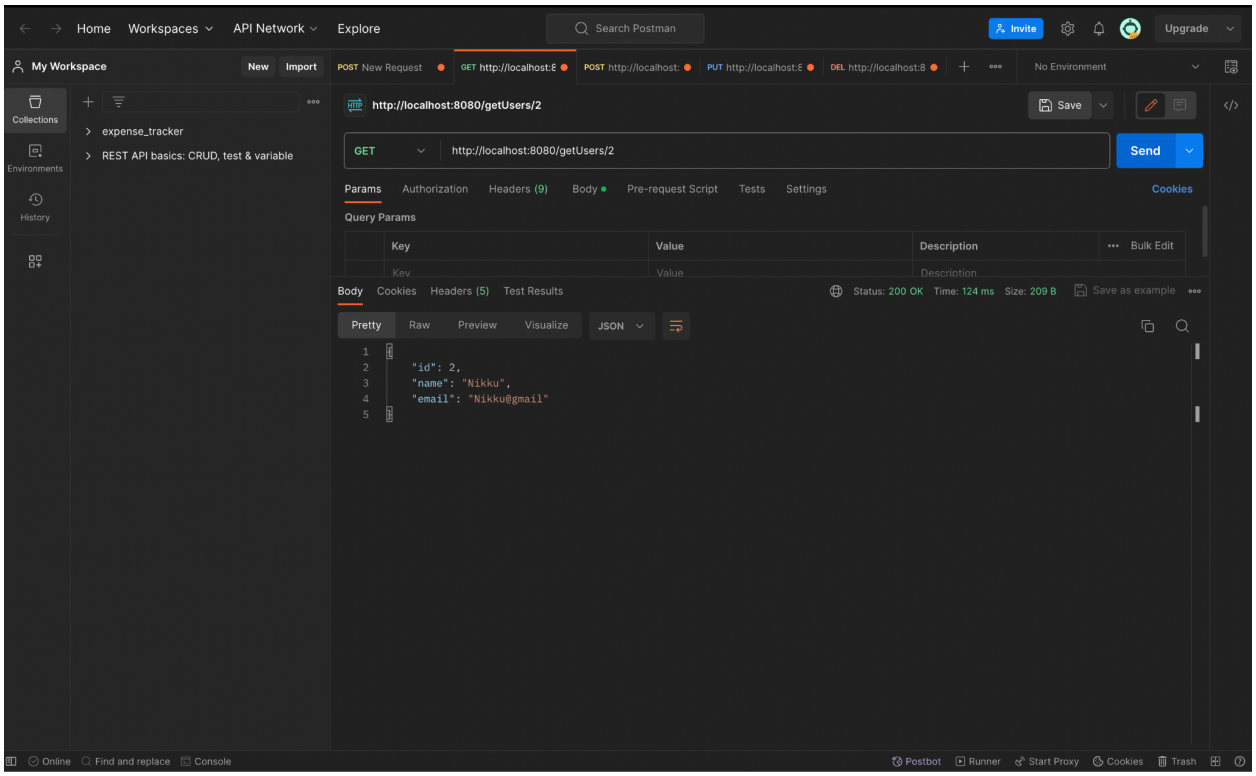
Below attached is the properties of database connection



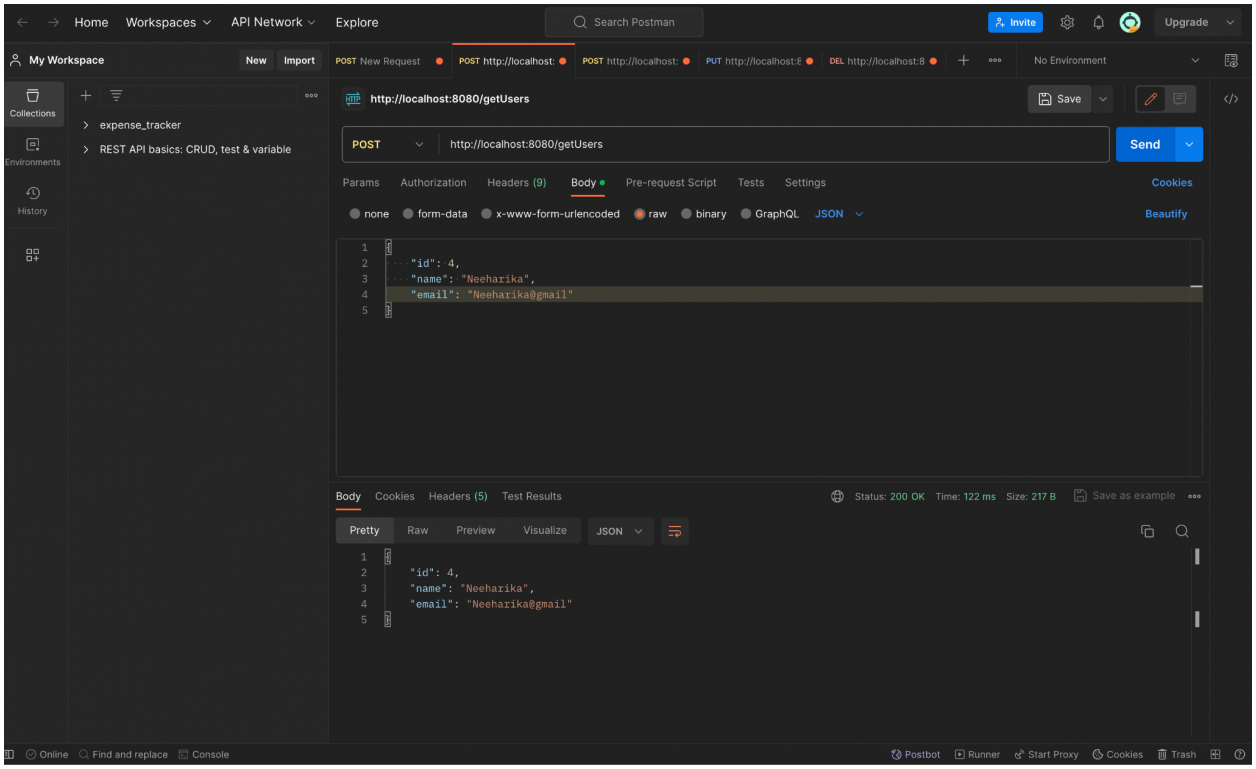
Read All Users

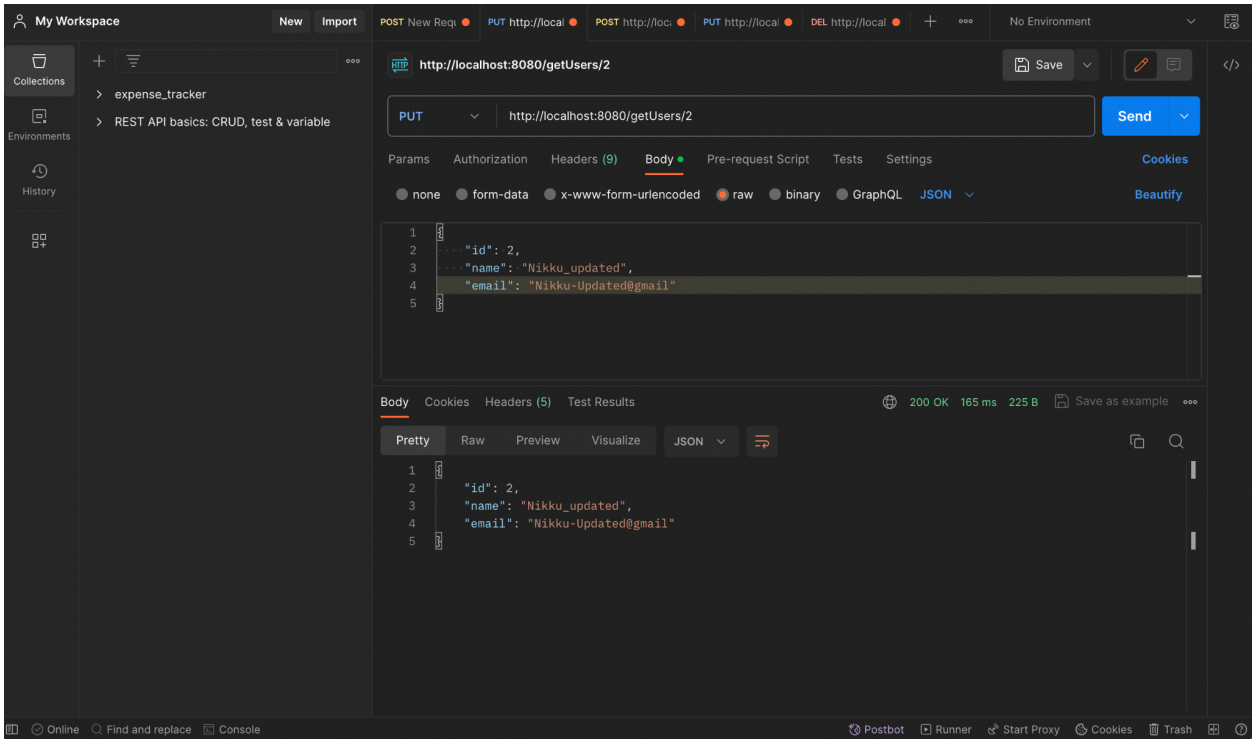


Read particular User



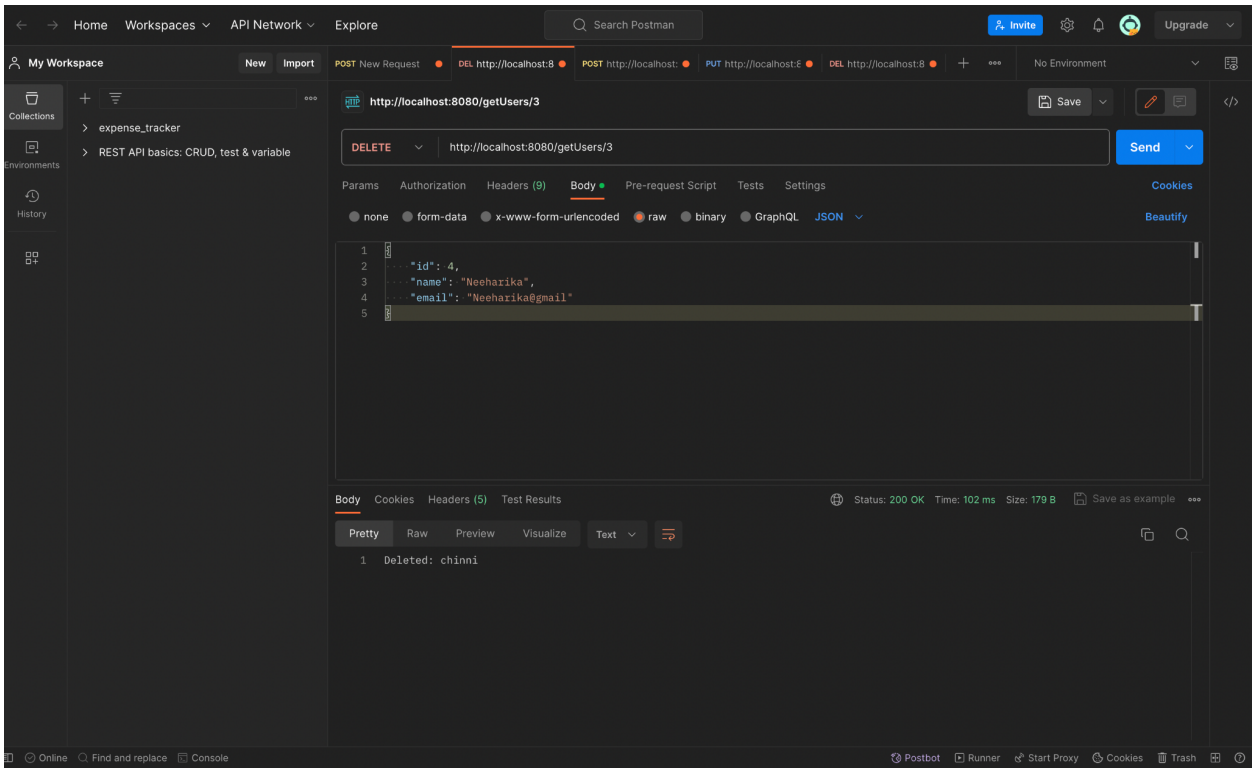
Insert one user

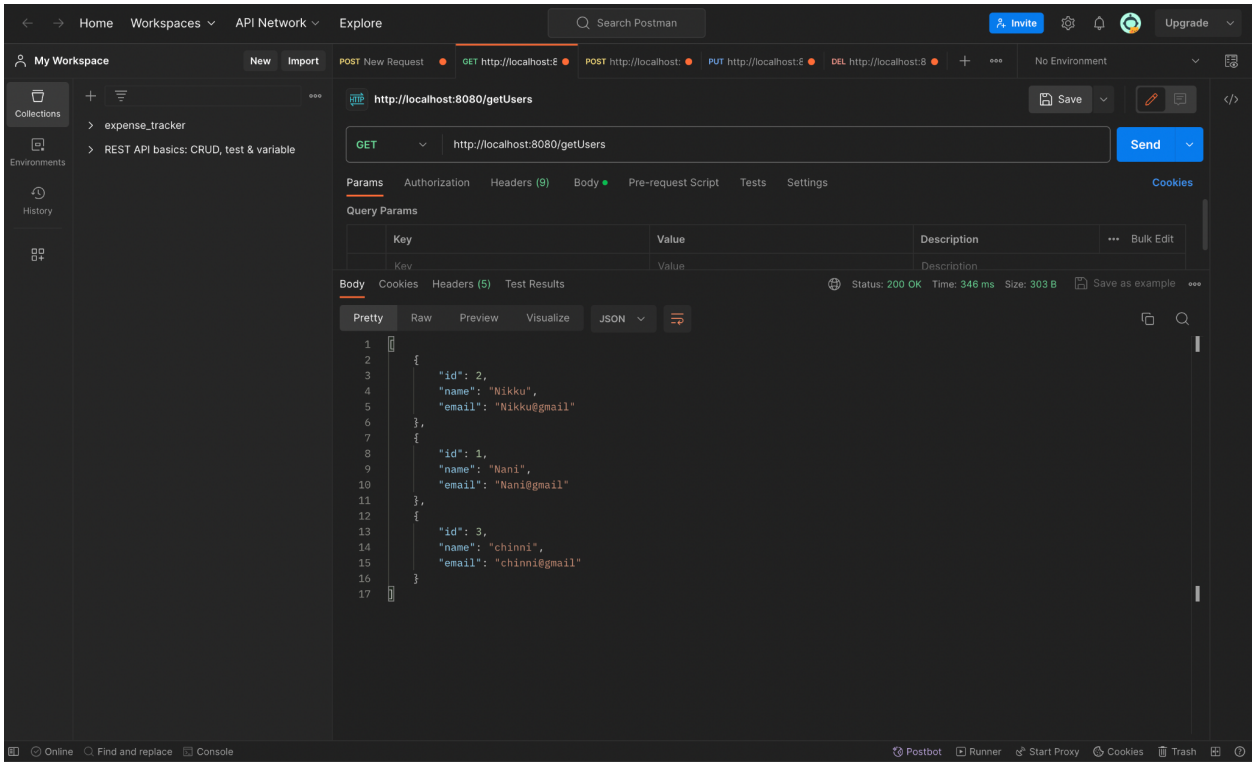




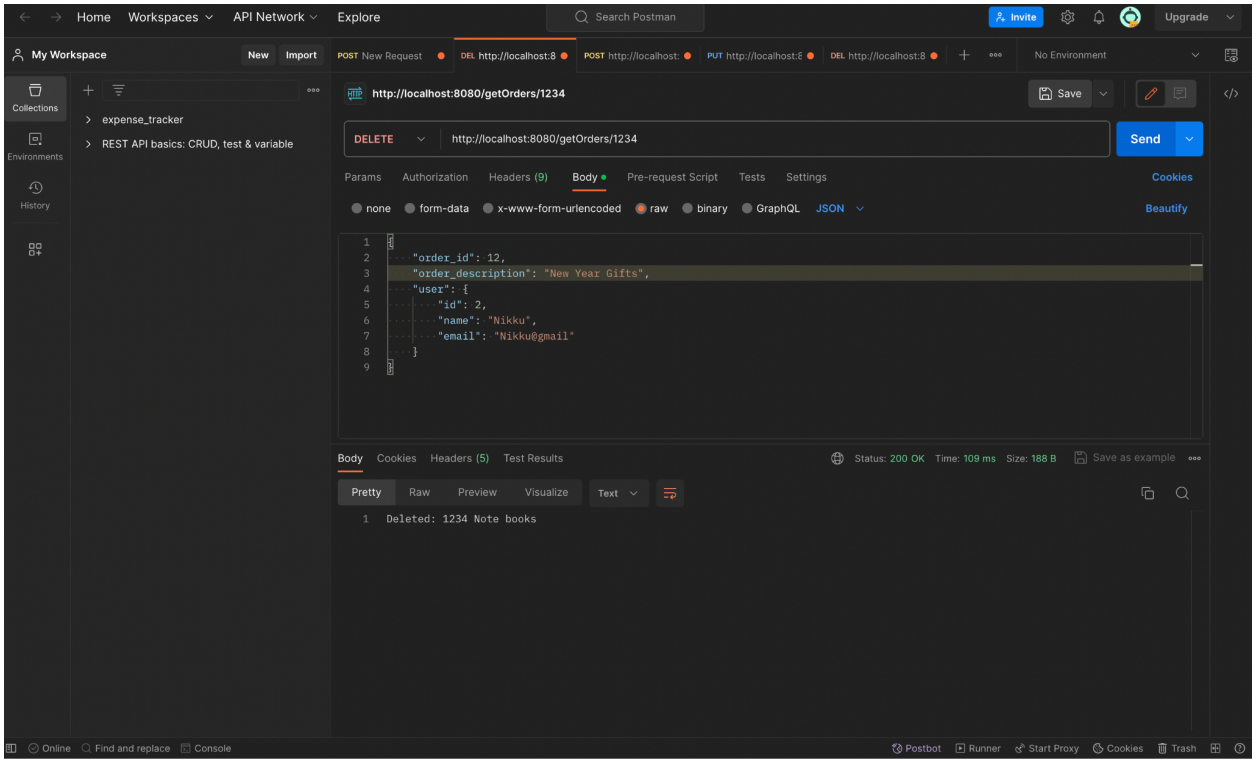
Above is for Update user

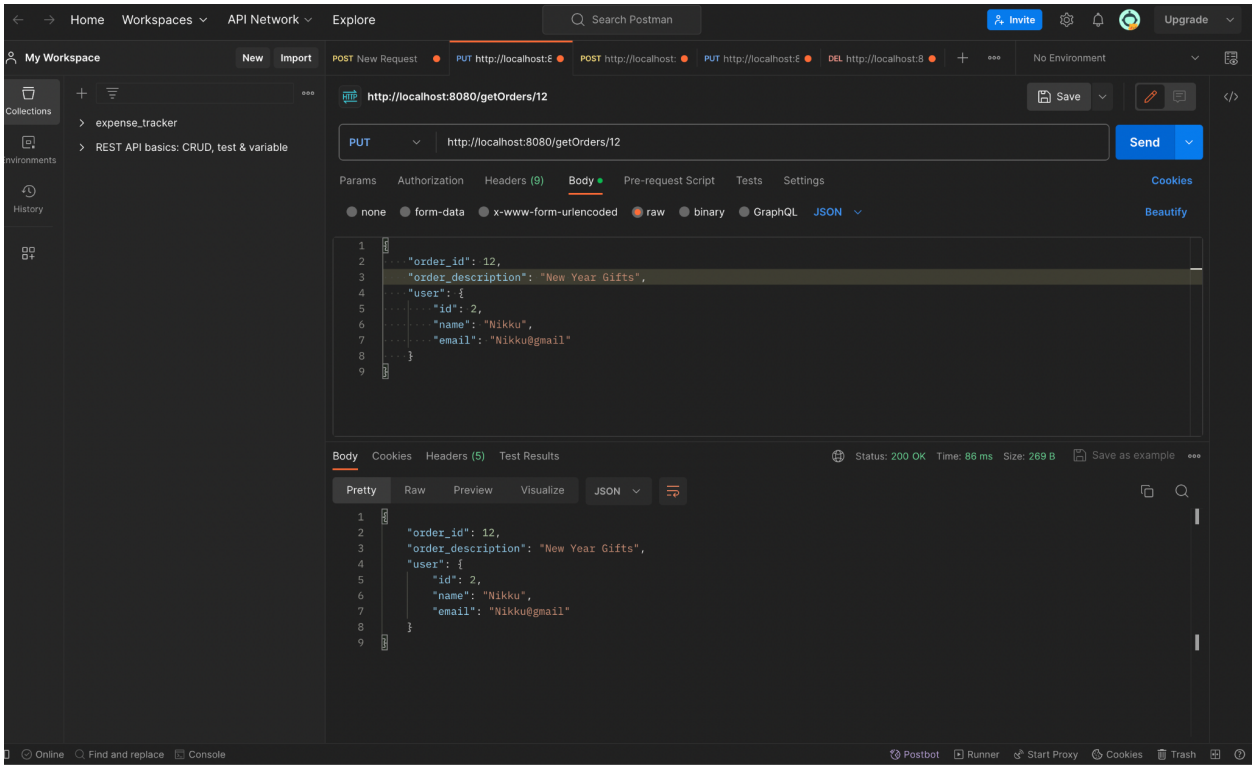
Delete User


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST New Request
DEL http://localhost:8
POST http://localhost:
PUT http://localhost:8
DEL http://localhost:8
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
1 {
2 "id": 4,
3 "name": "Neeharika",
4 "email": "Neeharika@gmail"
5 }
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 102 ms
Size: 179 B
Save as example
Pretty
Raw
Preview
Visualize
Text
1 Deleted: chinni
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash



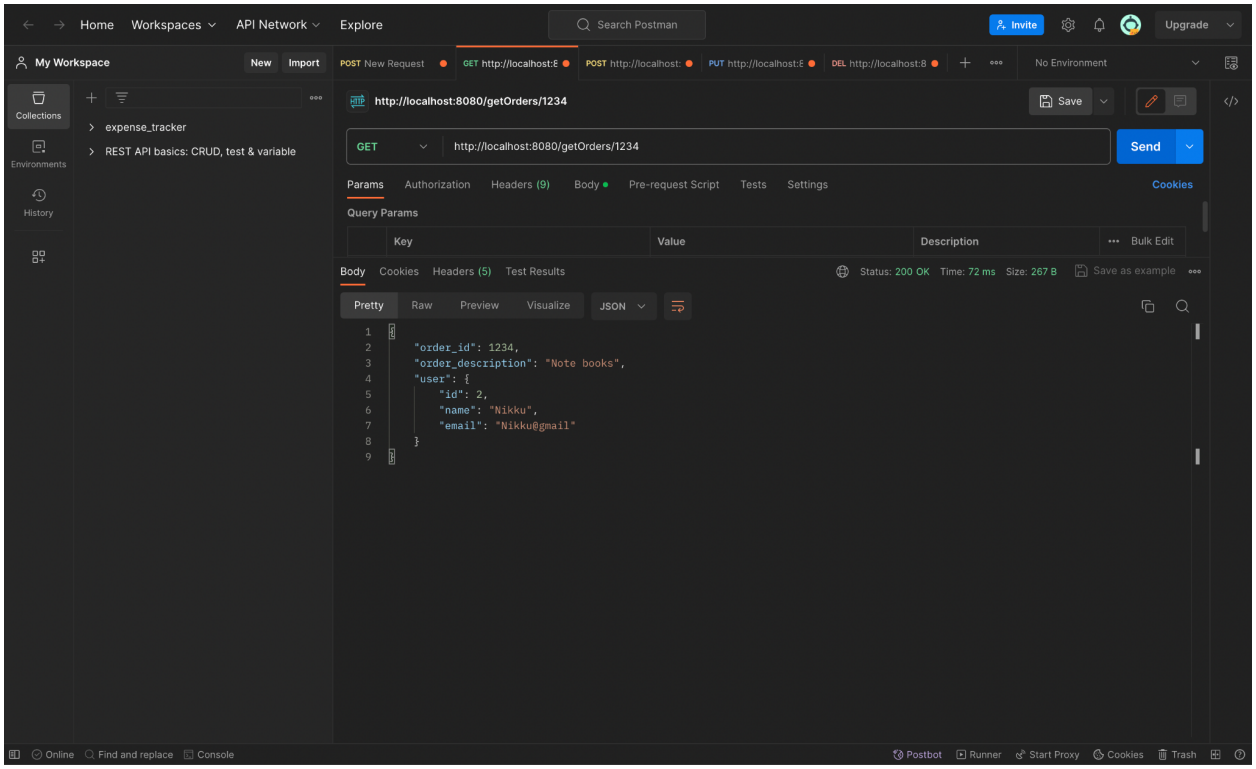
Delete order

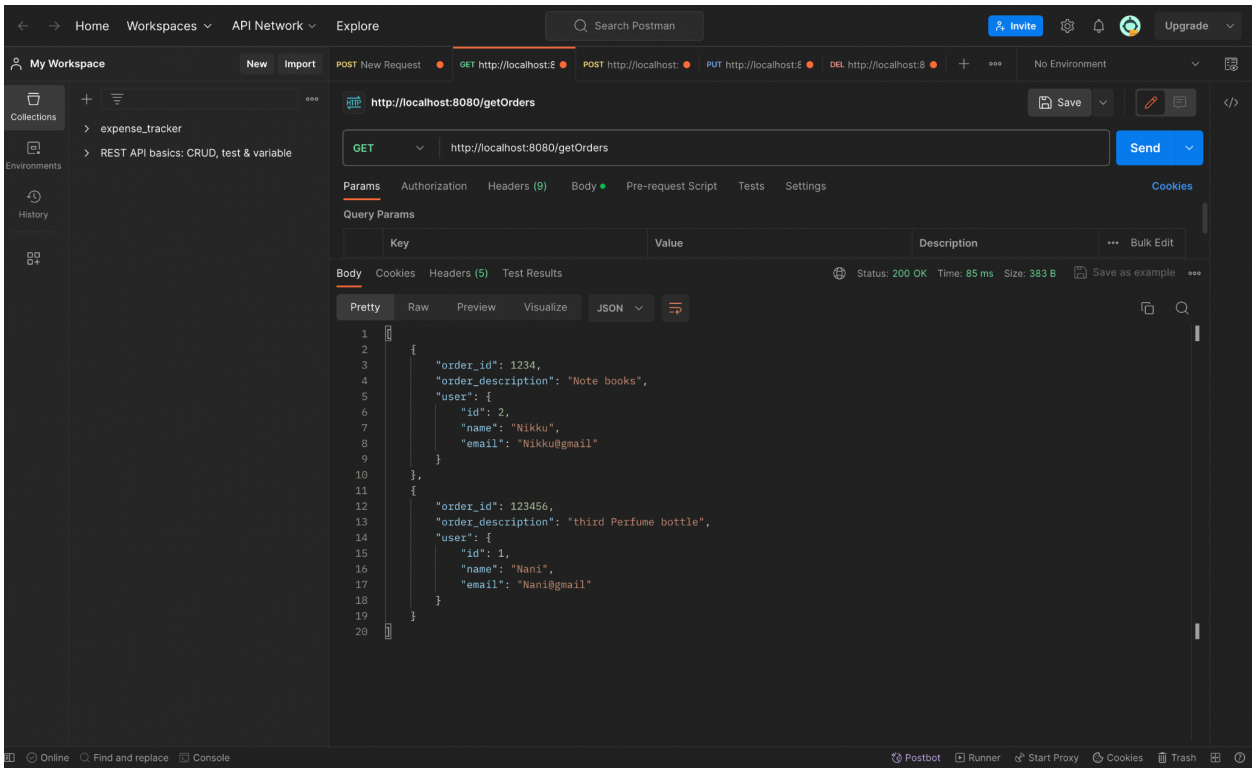




Above is Update particular order

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.

Below is the code

Orders.java

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


import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class Orders {

    @Id
    private int order_id;
    private String order_description;
```

```
    @ManyToOne
    @JoinColumn(name = "user_id", referencedColumnName = "id")
    private Users user;

    public int getOrder_id() {
        return order_id;
    }

    public void setOrder_id(int order_id) {
        this.order_id = order_id;
    }

    public String getOrder_description() {
        return order_description;
    }

    public void setOrder_description(String order_description) {
        this.order_description = order_description;
    }

    public Users getUser() {
        return user;
    }

    public void setUser(Users user) {
        this.user = user;
    }



    public Orders() {
    }


}
```

OrdersController.java

```
 package com.Neeharika.CRUDDemo.Controller;

import com.Neeharika.CRUDDemo.Model.Orders;
import com.Neeharika.CRUDDemo.Model.Users;
import com.Neeharika.CRUDDemo.Repository.OrdersRepo;
import jakarta.persistence.criteria.Order;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;
import java.util.Optional;

@RestController
public class OrdersController {

    @Autowired
    private OrdersRepo ordersRepo;

    @GetMapping("/getOrders")
    public List<Orders> getOrders(){
        return this.ordersRepo.findAll();
        //return this.ordersRepo.findAllOrdersWithUsers();
    }

    @GetMapping("/getOrders/{id}")
    public Optional<Orders> getOrderById(@PathVariable(value = "id") int id ){
        Optional<Orders> order;
        order = ordersRepo.findById(id);
        return (order);
    }


    @PostMapping("/getOrders")
    public Orders createOrder(@RequestBody Orders order){
        return this.ordersRepo.save(order);
    }

    @PutMapping("/getOrders/{id}")
    public ResponseEntity<Orders> updateOrder(@PathVariable(value = "id") int id, @RequestBody
Orders newOrder ){
        Orders order;
        order = ordersRepo.findById(id).orElse(null);
        assert order != null;
        order.setOrder_description(newOrder.getOrder_description());
        //order.setName(newUser.getName());
        return ResponseEntity.ok(this.ordersRepo.save(order));
    }

    @DeleteMapping("/getOrders/{id}")
    public String deleteEmployee( @PathVariable (value ="id") int id){
        Orders order;
        order = ordersRepo.findById(id).orElse(null);
        this.ordersRepo.delete(order);
        return "Deleted: "+ order.getOrder_id()+" "+ order.getOrder_description();
    }
```

```
}
```

OrdersRepo.java

```
package com.Neeharika.CRUDDemo.Repository;

import com.Neeharika.CRUDDemo.Model.Orders;
import jakarta.persistence.criteria.Order;
//import org.hibernate.query.Order;
import org.springframework.data.jpa.repository.JpaRepository;
//import org.springframework.data.jpa.repository.Query;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface OrdersRepo extends JpaRepository<Orders, Integer> {

    //@Query("SELECT o FROM Order o JOIN FETCH o.user")
    @Query("SELECT o FROM Orders o JOIN FETCH o.user")
    List<Orders> findAllOrdersWithUsers();
}
```

UserRepo.java

```
package com.Neeharika.CRUDDemo.Repository;

import com.Neeharika.CRUDDemo.Model.Users;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepo extends JpaRepository<Users, Integer> {
}
```

Users.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() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

UsersController.java

```
package com.Neeharika.CRUDDemo.Controller;
```

```
import com.Neeharika.CRUDDemo.Model.Orders;
import com.Neeharika.CRUDDemo.Model.Users;
import com.Neeharika.CRUDDemo.Repository.OrdersRepo;
import com.Neeharika.CRUDDemo.Repository.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
public class UsersController {
    @Autowired
    private UserRepo userRepo;
    //get

    @GetMapping("/")
    public String welcomeMessage(){
        return "Welcome";
    }
    @GetMapping("/getUsers")
    public List<Users> getUsers(){
        return this.userRepo.findAll();
    }

    //get user by id
    @GetMapping("/getUsers/{id}")
    public Optional<Users> getUserById(@PathVariable(value = "id") int id ){
        Optional<Users> user;
        user = userRepo.findById(id);
        return (user);
    }

    //create user
    @PostMapping("/getUsers")
    public Users createUser(@RequestBody  Users user){
        return this.userRepo.save(user);
    }

    //Update User
    @PutMapping("/getUsers/{id}")
```

```
    public ResponseEntity<Users> updateUser(@PathVariable(value = "id") int id,
@RequestBody Users newUser ){
        Users user;
        user = userRepo.findById(id).orElse(null);
        assert user != null;
        user.setEmail(newUser.getEmail());
        user.setName(newUser.getName());
        return ResponseEntity.ok(this.userRepo.save(user));
    }


    //Delete User
    @DeleteMapping("/getUsers/{id}")
    public String deleteEmployee( @PathVariable (value ="id") int id){
        Users user;
        user = userRepo.findById(id).orElse(null);
        this.userRepo.delete(user);
        return "Deleted: "+ user.getName();
    }



}
```