

Fredluis A. Obidos

IV – BCSAD

Assignment 1 - REST API

Codes:

Create JPA Entity - User.java:

```
src > main > java > net > javaguides > springboot > entity > User.java > java > User.java
1  package net.javaguides.springboot.entity;
2
3  import jakarta.persistence.*;
4  import lombok.AllArgsConstructor;
5  import lombok.Getter;
6  import lombok.NoArgsConstructor;
7  import lombok.Setter;
8
9  @Getter
10 @Setter
11 @NoArgsConstructor
12 @AllArgsConstructor
13 @Entity
14 @Table(name = "users")
15 public class User {
16
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long id;
20     @Column(nullable = false)
21     private String firstName;
22     @Column(nullable = false)
23     private String lastName;
24     @Column(nullable = false, unique = true)
25     private String email;
26 }
```

Create Spring Data JPA Repository for User JPA Entity

```
1 package net.javaguides.springboot.repository;
2
3 import net.javaguides.springboot.entity.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface UserRepository extends JpaRepository<User, Long> {
7 }
```

UserService Interface:

```
package net.javaguides.springboot.service;

import net.javaguides.springboot.entity.User;
import java.util.List;

public interface UserService {
    User createUser(User user);

    User getUserById(Long userId);

    List<User> getAllUsers();

    User updateUser(User user);

    void deleteUser(Long userId);
}
```

UserServiceImpl Class:

```
package net.javaguides.springboot.service.impl;

import lombok.AllArgsConstructor;
import net.javaguides.springboot.entity.User;
import net.javaguides.springboot.repository.UserRepository;
import net.javaguides.springboot.service.UserService;
import org.springframework.stereotype.Service;

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

@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;

    @Override
    public User createUser(User user) {
        return userRepository.save(user);
    }

    @Override
    public User getUserById(Long userId) {
        Optional<User> optionalUser = userRepository.findById(userId);
        return optionalUser.get();
    }

    @Override
    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    @Override
    public User updateUser(User user) {
        User existingUser = userRepository.findById(user.getId()).get();
        existingUser.setFirstName(user.getFirstName());
        existingUser.setLastName(user.getLastName());
        existingUser.setEmail(user.getEmail());
        User updatedUser = userRepository.save(existingUser);
        return updatedUser;
    }
}
```

```

    @Override
    public void deleteUser(Long userId) {
        userRepository.deleteById(userId);
    }
}

```

Creating UserController - Building CRUD Rest APIs:

```

package net.javaguides.springboot.controller;

import lombok.AllArgsConstructor;
import net.javaguides.springboot.entity.User;
import net.javaguides.springboot.service.UserService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@AllArgsConstructor
@RequestMapping("api/users")
public class UserController {

    private UserService userService;

    // build create User REST API
    @PostMapping
    public ResponseEntity<User> createUser(@RequestBody User user){
        User savedUser = userService.createUser(user);
        return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
    }

    // build get user by id REST API
    // http://localhost:8080/api/users/1
    @GetMapping("{id}")
    public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
        User user = userService.getUserById(userId);
        return new ResponseEntity<>(user, HttpStatus.OK);
    }
}

```

```

// Build Get All Users REST API
// http://localhost:8080/api/users
@GetMapping
public ResponseEntity<List<User>> getAllUsers(){
    List<User> users = userService.getAllUsers();
    return new ResponseEntity<>(users, HttpStatus.OK);
}

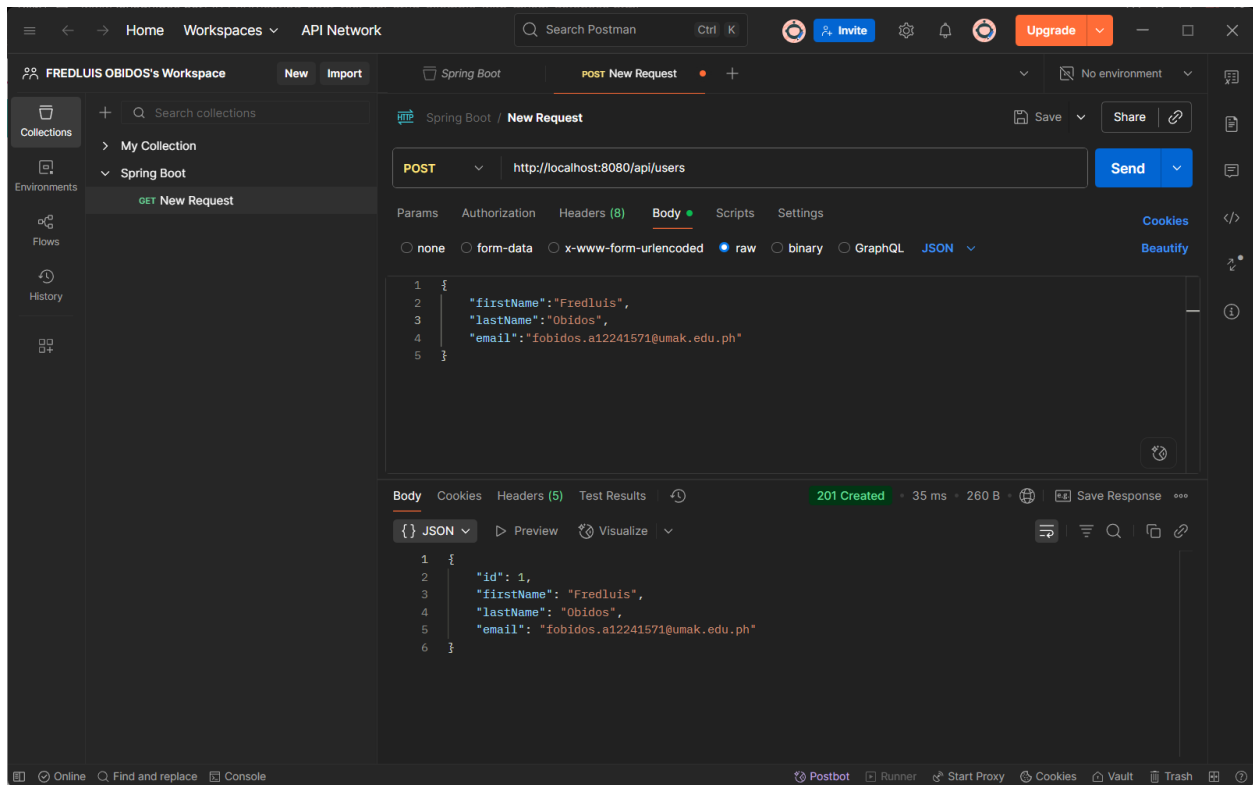
// Build Update User REST API
@PutMapping("{id}")
// http://localhost:8080/api/users/1
public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
                                      @RequestBody User user){
    user.setId(userId);
    User updatedUser = userService.updateUser(user);
    return new ResponseEntity<>(updatedUser, HttpStatus.OK);
}

// Build Delete User REST API
@DeleteMapping("{id}")
public ResponseEntity<String> deleteUser(@PathVariable("id") Long userId){
    userService.deleteUser(userId);
    return new ResponseEntity<>("User successfully deleted!", HttpStatus.OK);
}
}

```

Testing Screen Shots:

Create User REST API:



Get Single User REST API:

The screenshot shows the Postman interface with a workspace named "FREDLUIS OBIDOS's Workspace". A "New Request" is configured with the method "GET" and the URL "http://localhost:8080/api/users/1". The request is sent, and the response is a 200 OK status with a 6 ms response time and 255 B of data. The response body is in JSON format, showing a user object with the following details:

```
{  "id": 1,  "firstName": "Fredluis",  "lastName": "Obidos",  "email": "fobidos.a12241571@umak.edu.ph"}
```

Update User REST API:

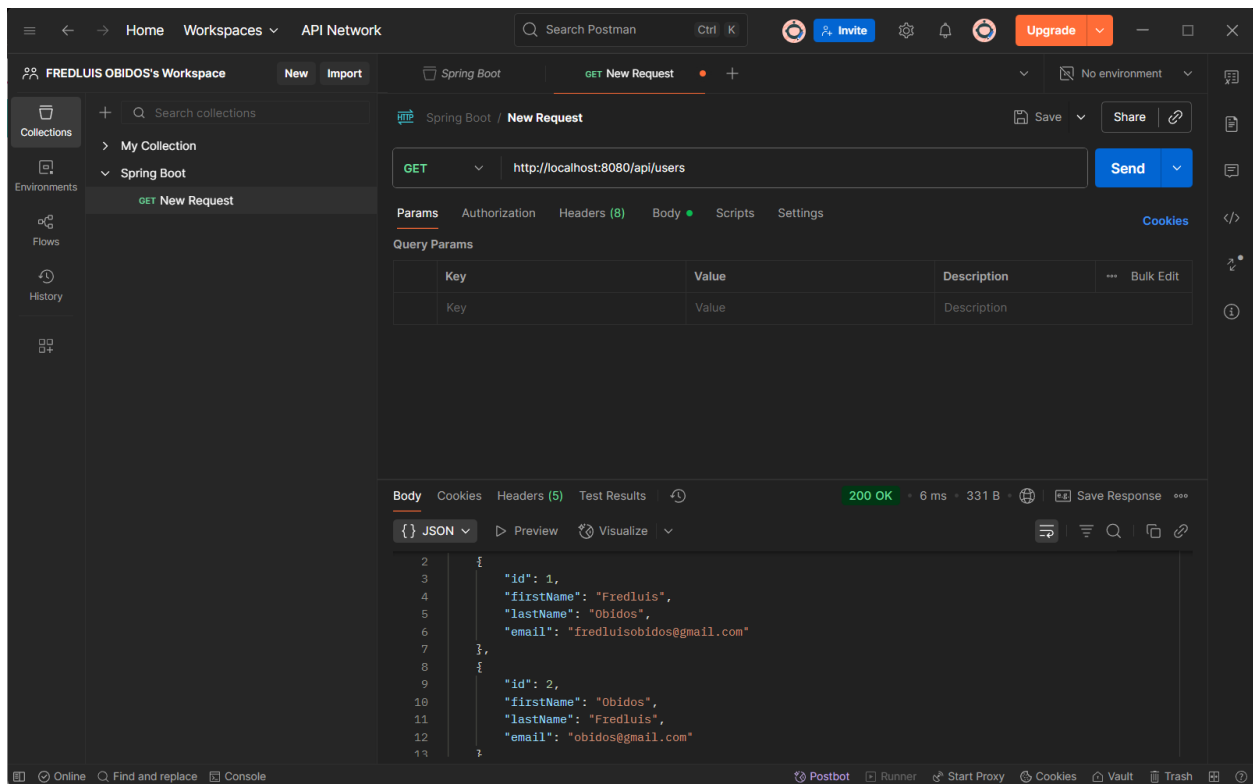
The screenshot shows the Postman interface with a workspace named "FREDLUIS OBIDOS's Workspace". A "New Request" is configured with the method "PUT" and the URL "http://localhost:8080/api/users/1". The request is sent, and the response is a 200 OK status with a 15 ms response time and 250 B of data. The request body is in JSON format, showing a user object with the following details:

```
{  "firstName": "Fredluis",  "lastName": "Obidos",  "email": "fredluisobidos@gmail.com"}
```

The response body is in JSON format, showing a user object with the following details:

```
{  "id": 1,  "firstName": "Fredluis",  "lastName": "Obidos",  "email": "fredluisobidos@gmail.com"}
```

Get All Users REST API:

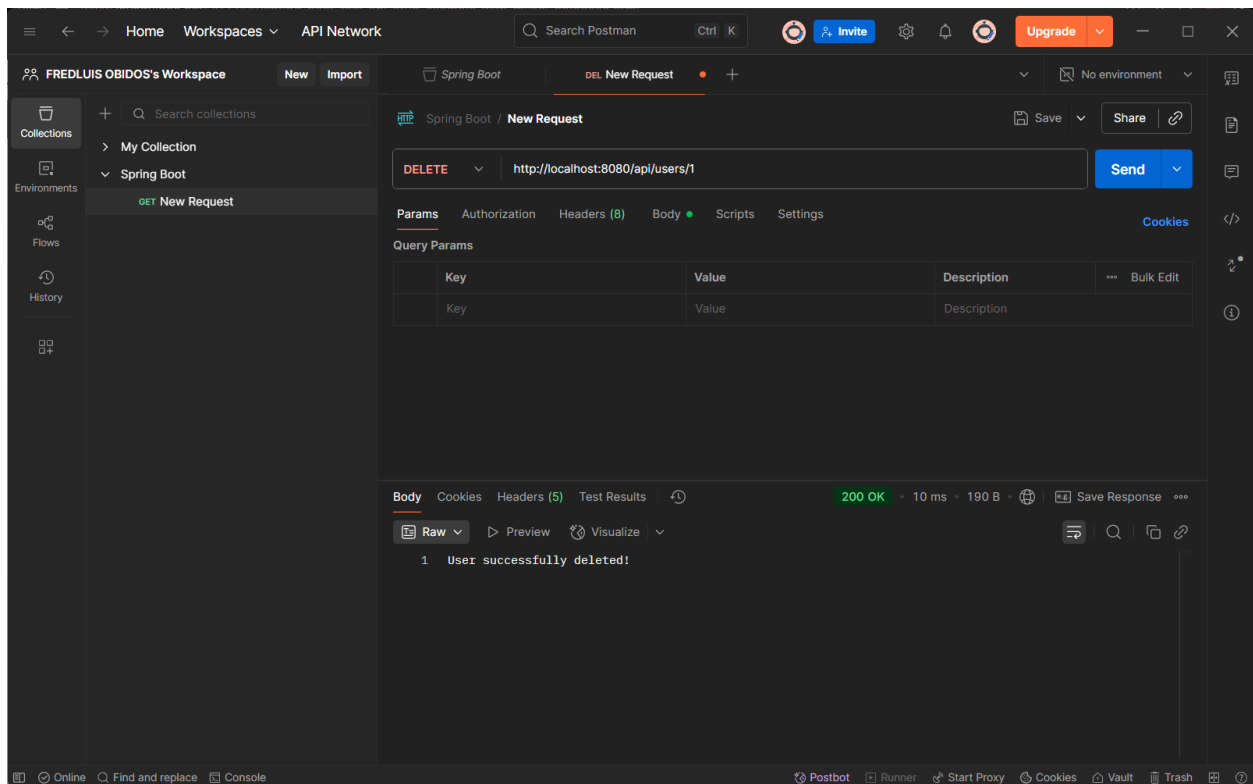


The screenshot shows the Postman interface with a workspace named "FREDLUIS OBIDOS's Workspace". A "New Request" is configured with the method "GET" and the URL "http://localhost:8080/api/users". The "Send" button is visible. Below the request, the "Body" tab is selected, showing a JSON response:

```
{
  "id": 1,
  "firstName": "Fredluis",
  "lastName": "Obidos",
  "email": "fredluisobidos@gmail.com"
},
{
  "id": 2,
  "firstName": "Obidos",
  "lastName": "Fredluis",
  "email": "obidos@gmail.com"
}
```

The response status is "200 OK" with a response time of 6 ms and a body size of 331 B. The interface also shows a sidebar with "Collections" and "Environments", and a bottom bar with various tool icons.

Delete User REST API:



The screenshot shows the Postman interface with the same workspace. A "New Request" is configured with the method "DELETE" and the URL "http://localhost:8080/api/users/1". The "Send" button is visible. Below the request, the "Body" tab is selected, showing a raw text response:

```
1 User successfully deleted!
```

The response status is "200 OK" with a response time of 10 ms and a body size of 190 B. The interface also shows a sidebar with "Collections" and "Environments", and a bottom bar with various tool icons.