

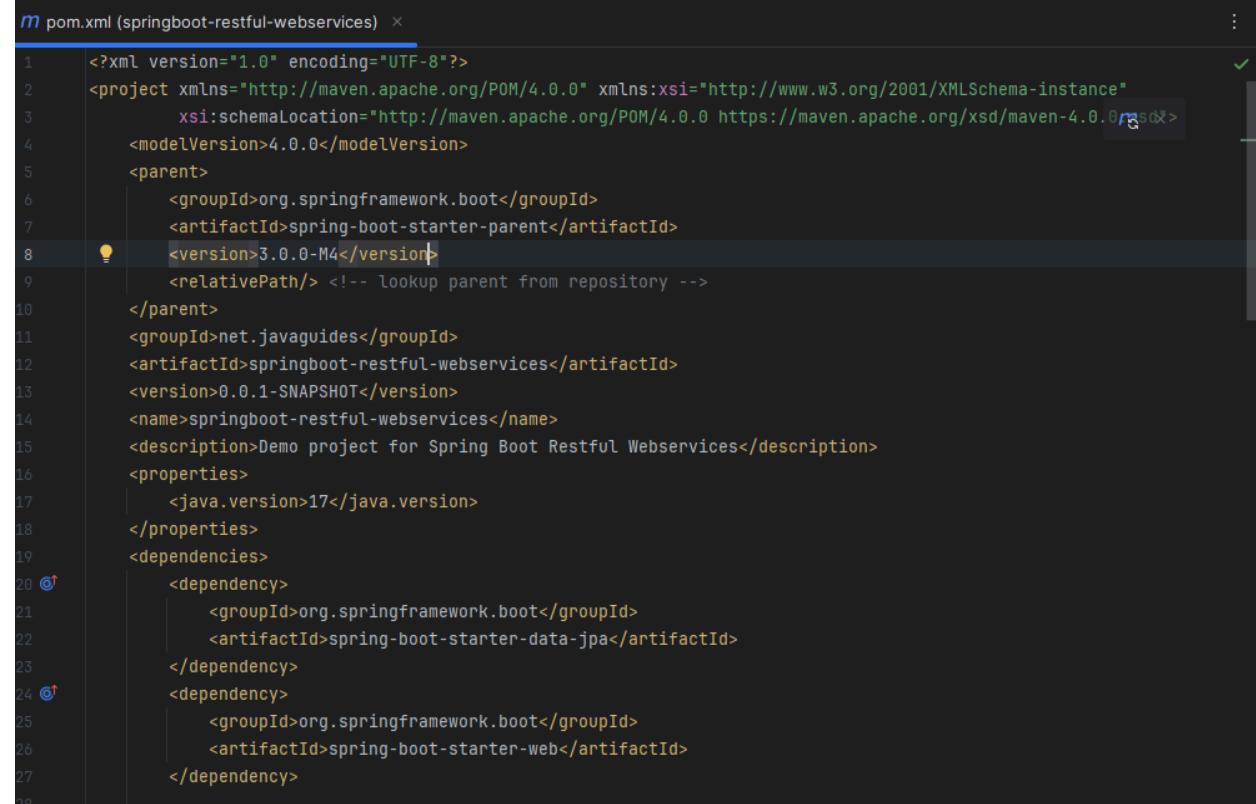
BARGABINO, MICHAEL JOSH V.

FORTE, TIMOTHY

IV-ACSAD

CODE

POM.XML



The screenshot shows a code editor window with the file 'pom.xml' open. The file is a Maven Project Object Model (POM) XML document. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.0-M4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>net.javaguides</groupId>
  <artifactId>springboot-restful-webservices</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>springboot-restful-webservices</name>
  <description>Demo project for Spring Boot Restful Webservices</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>

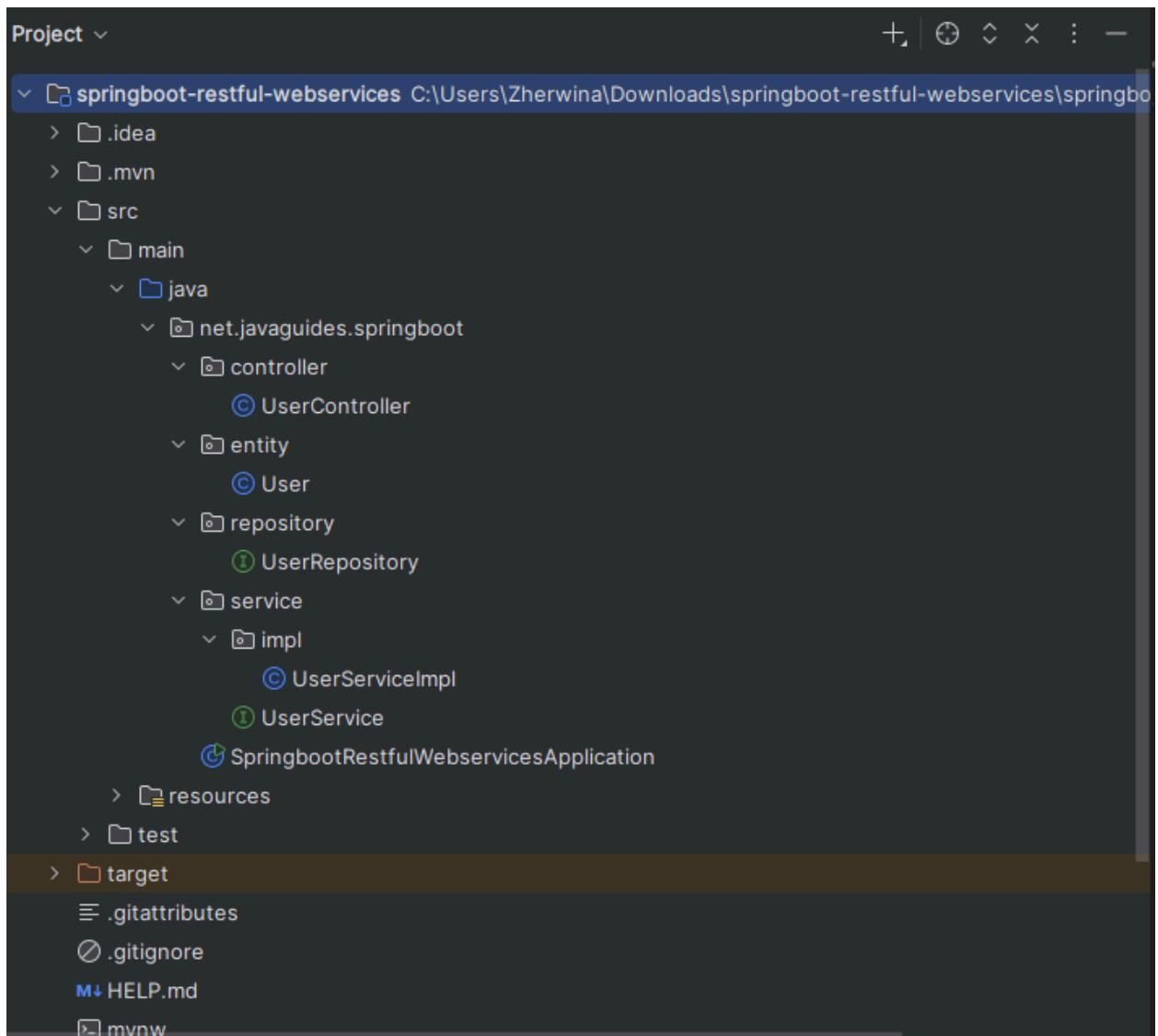
```

```
m pom.xml (springboot-restful-webservices) ×
2   <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ✓
19      <dependencies>
29         <dependency>
31             <artifactId>mysql-connector-java</artifactId>
32             <scope>runtime</scope>
33         </dependency>
34     ⏷         <dependency>
35             <groupId>org.projectlombok</groupId>
36             <artifactId>lombok</artifactId>
37             <optional>true</optional>
38         </dependency>
39     ⏷         <dependency>
40             <groupId>org.springframework.boot</groupId>
41             <artifactId>spring-boot-starter-test</artifactId>
42             <scope>test</scope>
43         </dependency>
44     </dependencies>
45
46     <build>
47         <plugins>
48             <plugin>
49                 <groupId>org.springframework.boot</groupId>
50     ⏷                 <artifactId>spring-boot-maven-plugin</artifactId>
51                 <configuration>
52                     <excludes>
53                         <exclude>
54                             <groupId>org.projectlombok</groupId>
55                             <artifactId>lombok</artifactId>

```

```
m pom.xml (springboot-restful-webservices) ×
2   <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ✓
46      <build>
47          <plugins>
48              <plugin>
49                  <configuration>
51                      </excludes>
52                      </configuration>
53                  </plugin>
54              </plugins>
55          </build>
56          <repositories>
57              <repository>
58                  <id>spring-milestones</id>
59                  <name>Spring Milestones</name>
60                  <url>https://repo.spring.io/milestone</url>
61                  <snapshots>
62                      <enabled>false</enabled>
63                  </snapshots>
64              </repository>
65          </repositories>
66          <pluginRepositories>
67              <pluginRepository>
68                  <id>spring-milestones</id>
69                  <name>Spring Milestones</name>
70                  <url>https://repo.spring.io/milestone</url>
71                  <snapshots>
72                      <enabled>false</enabled>
73                  </snapshots>
74          </pluginRepositories>
75      </project>
76  
```

PROJECT STRUCTURE:



APPLICATION.PROPERTIES:

The screenshot shows a code editor interface with two tabs: 'pom.xml (springboot-restful-webservices)' and 'application.properties'. The 'application.properties' tab is active and contains the following configuration:

```
1 spring.application.name=springboot-restful-webservices
2 spring.datasource.url=jdbc:mysql://localhost:3306/user_management
3 spring.datasource.username=root
4 spring.datasource.password=
5
6 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
7 spring.jpa.hibernate.ddl-auto=update
```

The configuration includes the application name, database connection details (MySQL), and Hibernate properties for JPA.

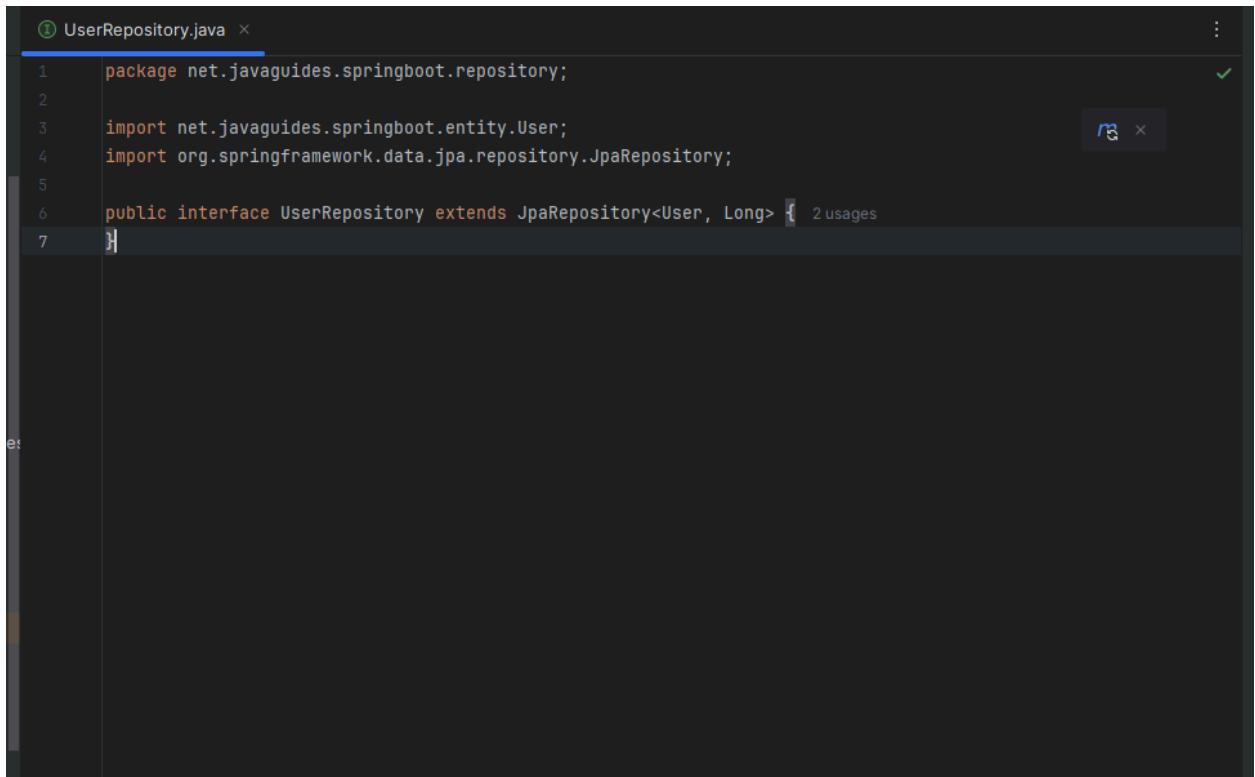
USER.JAVA

The screenshot shows a Java code editor interface with three tabs: `pom.xml`, `application.properties`, and `User.java`. The `User.java` tab is active, displaying the following code:

```
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 30 usages
10 @Setter
11 @NoArgsConstructor
12 @AllArgsConstructor
13 @Entity
14 @Table(name = "users")
15 public class User {
16
17     @Id no usages
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long id;
20
21     @Column(nullable = false) no usages
22     private String firstName;
23
24     @Column(nullable = false) no usages
25     private String lastName;
26     @Column(nullable = false, unique = true) no usages
27     private String email;
28 }
```

The code defines a `User` entity with an `@Id` annotated `id` field, `firstName`, `lastName`, and `email` fields. The `email` field is annotated with `@Column(nullable = false, unique = true)`. Lombok annotations like `@Getter`, `@Setter`, `@NoArgsConstructor`, and `@AllArgsConstructor` are used.

USER REPOSITORY:



A screenshot of a code editor showing a Java file named `UserRepository.java`. The code defines a repository interface for a `User` entity, extending `JpaRepository`. The code is as follows:

```
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> { 2 usages
7 }
```

USERSERVICE:

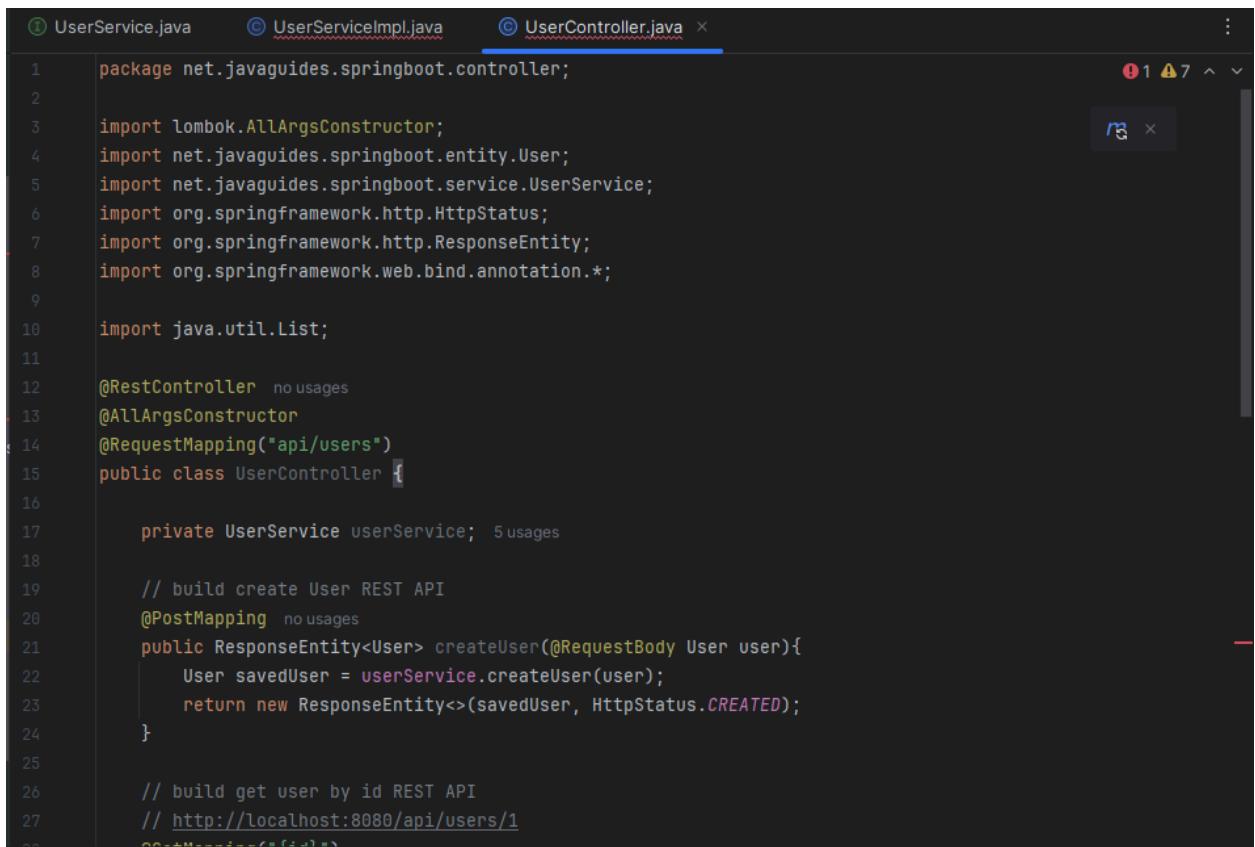
```
UserService.java
1 package net.javaguides.springboot.service;
2
3 import net.javaguides.springboot.entity.User;
4
5 import java.util.List;
6
7 @Service
8 public interface UserService {
9     User createUser(User user);
10    User getUserById(Long userId);
11    List<User> getAllUsers();
12    User updateUser(User user);
13    void deleteUser(Long userId);
14 }
15
```

USERSERVICEIMPL:

```
① UserService.java      ② UserServiceImpl.java ×
1  package net.javaguides.springboot.service.impl;
2
3  import lombok.AllArgsConstructor;
4  import net.javaguides.springboot.entity.User;
5  import net.javaguides.springboot.repository.UserRepository;
6  import net.javaguides.springboot.service.UserService;
7  import org.apache.logging.log4j.util.Strings;
8  import org.springframework.stereotype.Service;
9  import org.springframework.util.StringUtils;
10
11 import java.util.List;
12 import java.util.Objects;
13 import java.util.Optional;
14
15 @Service no usages
16 @AllArgsConstructor
17 public class UserServiceImpl implements UserService {
18
19     private UserRepository userRepository; 6 usages
20
21     @Override 1 usage
22     public User createUser(User user) {
23         return userRepository.save(user);
24     }
25
26     @Override 1 usage
27     public User getUserById(Long userId) {
28         return userRepository.findById(userId);
29     }
30
31     @Override 1 usage
32     public List<User> getAllUsers() {
33         return userRepository.findAll();
34     }
35
36
37     @Override 1 usage
38     public User updateUser(User user) {
39         User existingUser = userRepository.findById(user.getId()).get();
40         existingUser.setFirstName(user.getFirstName());
41         existingUser.setLastName(user.getLastName());
42         existingUser.setEmail(user.getEmail());
43         User updatedUser = userRepository.save(existingUser);
44         return updatedUser;
45     }
46
47     @Override 1 usage
48     public void deleteUser(Long userId) {
49         userRepository.deleteById(userId);
50     }
51 }
```

```
① UserService.java      ② UserServiceImpl.java ×
17    public class UserServiceImpl implements UserService {
18
19        private UserRepository userRepository; 6 usages
20
21        @Override 1 usage
22        public List<User> getAllUsers() {
23            return userRepository.findAll();
24        }
25
26
27        @Override 1 usage
28        public User updateUser(User user) {
29            User existingUser = userRepository.findById(user.getId()).get();
30            existingUser.setFirstName(user.getFirstName());
31            existingUser.setLastName(user.getLastName());
32            existingUser.setEmail(user.getEmail());
33            User updatedUser = userRepository.save(existingUser);
34            return updatedUser;
35        }
36
37        @Override 1 usage
38        public void deleteUser(Long userId) {
39            userRepository.deleteById(userId);
40        }
41
42    }
```

USERCONTROLLER:



The screenshot shows a code editor interface with three tabs at the top: `UserService.java`, `UserServiceImpl.java`, and `UserController.java`. The `UserController.java` tab is active. The code in the editor is as follows:

```
1 package net.javaguides.springboot.controller;
2
3 import lombok.AllArgsConstructor;
4 import net.javaguides.springboot.entity.User;
5 import net.javaguides.springboot.service.UserService;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.List;
11
12 @RestController no usages
13 @AllArgsConstructor
14 @RequestMapping("api/users")
15 public class UserController {
16
17     private UserService userService; 5 usages
18
19     // build create User REST API
20     @PostMapping no usages
21     public ResponseEntity<User> createUser(@RequestBody User user){
22         User savedUser = userService.createUser(user);
23         return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
24     }
25
26     // build get user by id REST API
27     // http://localhost:8080/api/users/1
28 }
```

The code defines a `UserController` class with a constructor annotated with `@AllArgsConstructor`. It contains a single method `createUser` annotated with `@PostMapping` that returns a `ResponseEntity` containing a `User` object and a `HttpStatus.CREATED`. The code editor shows line numbers from 1 to 28.

```
① UserService.java    ② UserServiceImpl.java    ③ UserController.java ×
15  public class UserController {
29      public ResponseEntity<User> getUserId(@PathVariable("id") Long userId){
30          User user = userService.getUserId(userId);
31          return new ResponseEntity<>(user, HttpStatus.OK);
32      }
33
34      // Build Get All Users REST API
35      // http://localhost:8080/api/users
36      @GetMapping no usages
37      public ResponseEntity<List<User>> getAllUsers(){
38          List<User> users = userService.getAllUsers();
39          return new ResponseEntity<>(users, HttpStatus.OK);
40      }
41
42      // Build Update User REST API
43      @PutMapping("{id}") no usages
44      // http://localhost:8080/api/users/1
45      @
46      public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
47                                              @RequestBody User user){
48          user.setId(userId);
49          User updatedUser = userService.updateUser(user);
50          return new ResponseEntity<>(updatedUser, HttpStatus.OK);
51      }
52
53      // Build Delete User REST API
54      @DeleteMapping("{id}") no usages
55      public ResponseEntity<String> deleteUser(@PathVariable("id") Long userId){
56          userService.deleteUser(userId);
57
58      }
```

```
// Build Delete User REST API
@DeleteMapping("{id}") no usages
public ResponseEntity<String> deleteUser(@PathVariable("id") Long userId){
    userService.deleteUser(userId);
    return new ResponseEntity<>( body: "User successfully deleted!", HttpStatus.OK);
}
```

TESTING

Create User REST API:

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8080/api/users` is entered. Below it, a POST request is selected. The 'Body' tab is active, showing raw JSON data:

```
1 {
2   "firstName": "umesh",
3   "lastName": "fadatare",
4   "email": "umesh@gmail.com"
5 }
```

On the right side, the response status is `201 Created`, and the response body is displayed as:

```
1 {
2   "id": 1,
3   "firstName": "umesh",
4   "lastName": "fadatare",
5   "email": "umesh@gmail.com"
6 }
```

Get Single User REST API:

The screenshot shows the Postman application interface. At the top, there are tabs for 'Overview', 'New Collection', and a selected tab 'GET http://localhost:8080/api/users/1'. To the right of the tabs are buttons for 'Save' and 'Share'. Below the tabs, a search bar contains 'http://localhost:8080/api/users/1'. On the left, a dropdown menu shows 'GET' is selected. To the right of the dropdown are 'Send' and a 'Send' button. Below the dropdown, tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Scripts', and 'Settings' are visible, with 'Body' being the active tab. Under 'Body', options for 'none', 'form-data', 'x-www-form-urlencoded', 'raw', 'binary', 'GraphQL', and 'JSON' are shown, with 'raw' selected. A JSON editor displays the following data:

```
1 {  
2   "firstName": "umesh",  
3   "lastName": "fadatare",  
4   "email": "umesh@gmail.com"  
5 }  
6 }
```

On the right side of the body editor, there are 'Cookies' and 'Beautify' buttons. Below the body editor, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Test Results' tab is active, showing a green '200 OK' status with a response time of '51 ms' and a size of '240 B'. To the right of the status are icons for copy, refresh, search, and link.

Update User REST API:

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Overview', 'New Collection', and a search bar containing 'PUT http://localhost:8080/api/users/1'. To the right of the search bar are 'Save' and 'Share' buttons. Below the search bar, the method 'PUT' is selected, and the URL 'http://localhost:8080/api/users/1' is displayed.

The main workspace is divided into sections: 'Params', 'Authorization', 'Headers (8)', 'Body', 'Scripts', and 'Settings'. The 'Body' section is active, showing the raw JSON payload:

```
1 {  
2   "firstName": "umesh",  
3   "lastName": "fadatare",  
4   "email": "umesh@gmail.com"  
5 }  
6 }
```

Below the body, there are tabs for 'Cookies', 'Beautify', and 'Raw'. The 'Raw' tab is currently selected. The 'Headers' tab shows 5 headers. The 'Test Results' tab is also visible.

At the bottom right, the response status is shown as '200 OK' with a green background, along with performance metrics: 27 ms, 240 B, and three dots indicating more details.

Get All Users REST API:

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/users`. The response status is `200 OK` with a response time of `15 ms` and a size of `242 B`.

Body (JSON)

```
1 [  
2 {  
3   "id": 1,  
4   "firstName": "umesh",  
5   "lastName": "fadatare",  
6   "email": "umesh@gmail.com"  
7 }  
8 ]
```

Delete User REST API:

Overview | New Collection | **DEL** <http://localhost:8080/api/users/1> | + | No environment | [Save](#) | [Share](#)

DELETE <http://localhost:8080/api/users/1> | **Send**

Params Authorization Headers (8) **Body** Scripts Settings Cookies Beautify

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

```
1 {  
2   "firstName": "umesh",  
3   "lastName": "fadatare",  
4   "email": "umesh@gmail.com"  
5 }  
6 }
```

Body Cookies Headers (5) Test Results | [Raw](#) | [Preview](#) | [Visualize](#) | [200 OK](#) • 79 ms • 190 B | [...](#)

```
1 User successfully deleted!
```