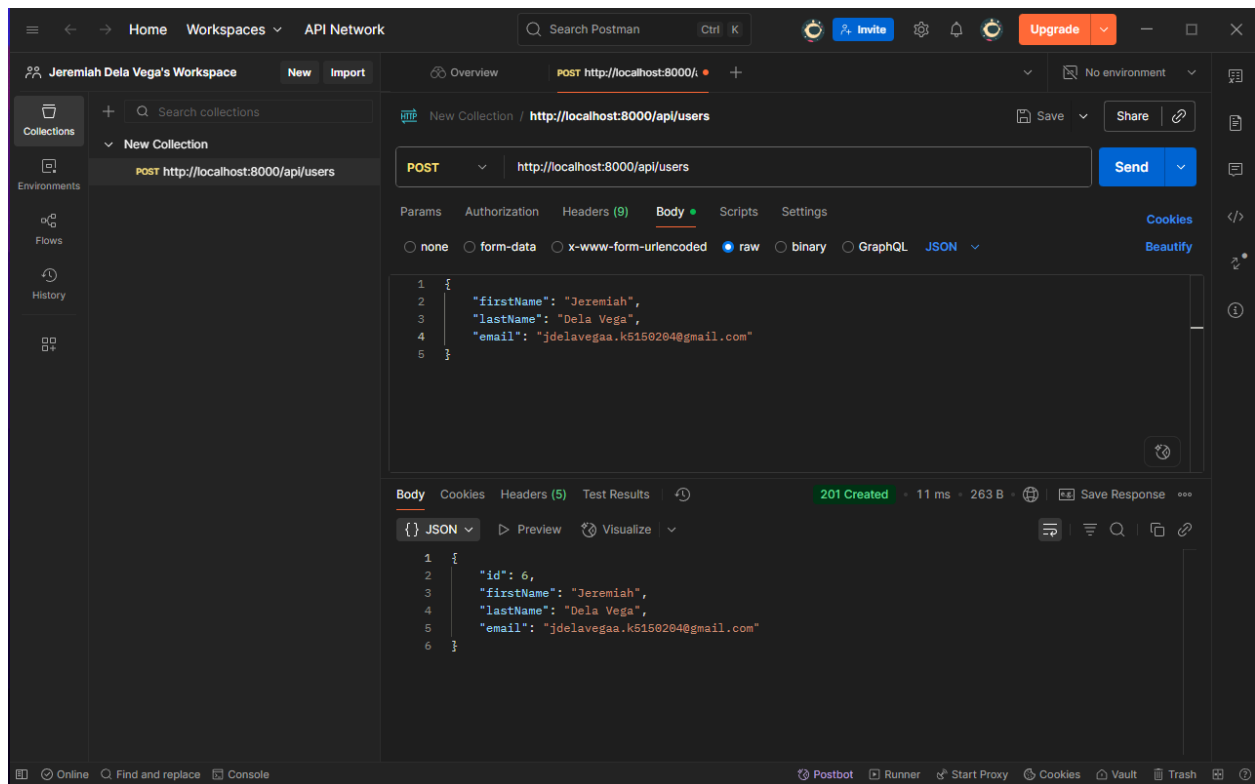


Jeremiah Dela Vega  
IV-BCSAD  
Elective 3: Assignment 1 - REST API

Output:

POST (User Creation)



Jeremiah Dela Vega's WorkspaceNewImport

OverviewPOST http://localhost:8000/

Search PostmanCtrl K

Invite

Upgrade

Collections

Environments

Flows

History

Search collections

New Collection

POST http://localhost:8000/api/users

New Collectionhttp://localhost:8000/api/users

SaveShare

POSThttp://localhost:8000/api/users

Send

ParamsAuthorizationHeaders (9)BodyScriptsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

cookies

Beautify

```
1 {
2   "firstName": "Miah",
3   "lastName": "Lifakha",
4   "email": "jdelavegaaa.k5150204@gmail.com"
5 }
```

BodyCookiesHeaders (5)Test Results

201 Created11 ms259 B

Save Response

JSONPreviewVisualize

```
1 {
2   "id": 10,
3   "firstName": "Miah",
4   "lastName": "Lifakha",
5   "email": "jdelavegaaa.k5150204@gmail.com"
6 }
```

OnlineFind and replaceConsole

PostbotRunnerStart ProxyCookiesVaultTrash

## GET (Single User)

The screenshot displays the Postman application interface. On the left sidebar, the 'Collections' tab is active, showing a collection named 'New Collection' with a single item 'POST http://localhost:8000/api/users'. The main workspace shows a GET request to 'http://localhost:8000/api/users/4'. The request is configured with the 'raw' body type and 'JSON' format. The response is a 200 OK status with a response time of 19 ms and a body size of 257 B. The response body is displayed in a JSON format, showing the details of a user with ID 4, first name 'Jeremiah', last name 'DeLa Vega', and email 'jdelavega.k5150204@gmail.com'.

Jeremiah Dela Vega's Workspace   New   Import   Overview   GET http://localhost:8000/api/users/4   No environment

New Collection / http://localhost:8000/api/users   Save   Share

GET   http://localhost:8000/api/users/4   Send

Params   Authorization   Headers (9)   Body   Scripts   Settings   Cookies   Beautify

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

Body   Cookies   Headers (5)   Test Results   200 OK   19 ms   257 B   Save Response

JSON   Preview   Visualize

```
1 {
2   "id": 4,
3   "firstName": "Jeremiah",
4   "lastName": "DeLa Vega",
5   "email": "jdelavega.k5150204@gmail.com"
6 }
```

Postbot   Runner   Start Proxy   Cookies   Vault   Trash

## GET (All Users)

The screenshot displays the Postman interface for a workspace named "Jeremiah Dela Vega's Workspace". The main view shows a GET request to the endpoint `http://localhost:8000/api/users`. The request is configured with the "raw" body type and "JSON" format. The response is a 200 OK status, indicating a successful request. The response body is displayed in JSON format, showing a list of four users.

**Request Details:**

- Method: GET
- URL: `http://localhost:8000/api/users`
- Body Type: raw
- Format: JSON

**Response Details:**

- Status: 200 OK
- Time: 117 ms
- Size: 537 B

**Response Body (JSON):**

```
[{"id": 1, "firstName": "Jeremiah", "lastName": "Dela Vega", "email": "bakitakomayemail@gmail.com"}, {"id": 4, "firstName": "Jeremiah", "lastName": "Dela Vega", "email": "jdelavega.k5150204@gmail.com"}, {"id": 6, "firstName": "Jeremiah", "lastName": "Dela Vega", "email": "jdelavegaa.k5150204@gmail.com"}, {"id": 10, "firstName": "Miah", "lastName": "Dela Vega", "email": "jdelavega.k5150204@gmail.com"}]
```

## PUT (Update)

The screenshot displays the Postman interface for a workspace named "Jeremiah Dela Vega's Workspace". The main view shows a PUT request to the endpoint `http://localhost:8000/api/users/4`. The request body is a JSON object with the following fields:

```
1 {
2   "firstName": "Miahhh",
3   "lastName": "Lifakha",
4   "email": "jdelavegaa2a.k5150204@gmail.com"
5 }
```

The response is a 200 OK status, indicating a successful update. The response body is a JSON object with the following fields:

```
1 {
2   "id": 4,
3   "firstName": "Miahhh",
4   "lastName": "Lifakha",
5   "email": "jdelavegaa2a.k5150204@gmail.com"
6 }
```

The interface includes a sidebar with "Collections" and "Environments", a top bar with navigation and search options, and a bottom bar with various tool icons.

# DELETE

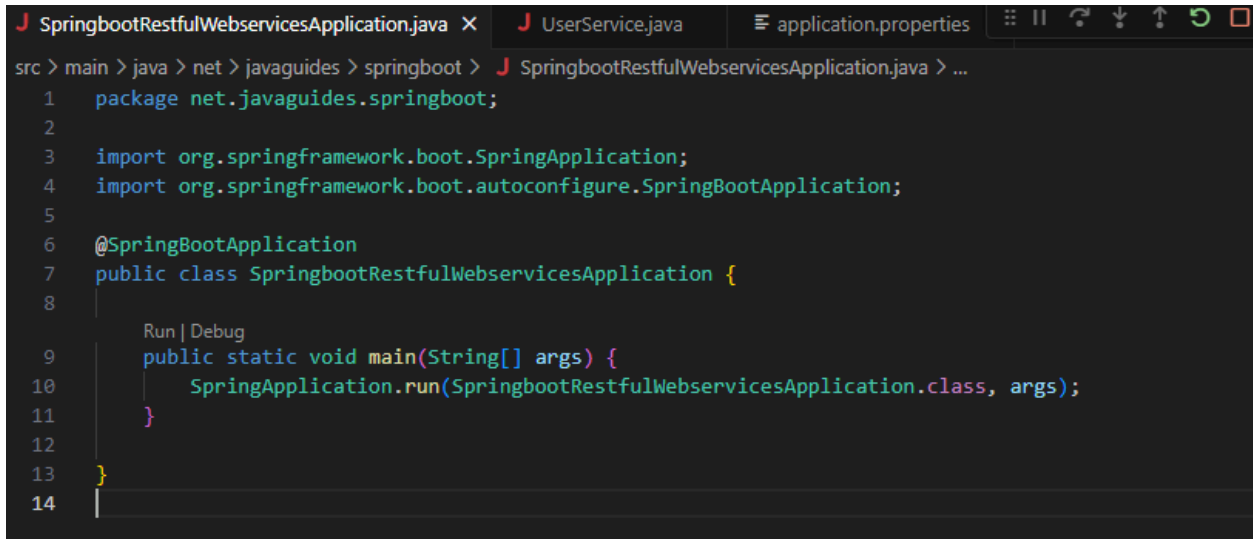
The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:8000/api/users/4`. The request body is a JSON object:

```
1 {
2   "firstName": "Miah",
3   "lastName": "Lifakha",
4   "email": "jdelavegaa2a.k5150204@gmail.com"
5 }
```

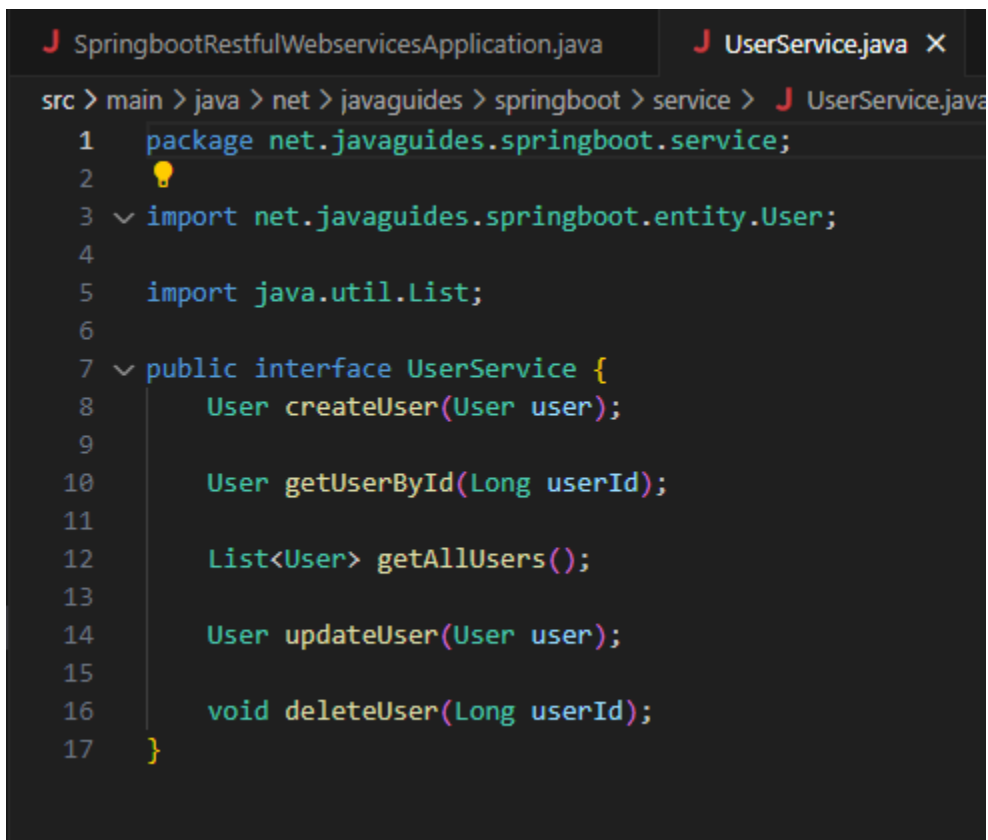
The response is **200 OK** with a status of **21 ms** and **190 B**. The response body is:

```
1 User successfully deleted!
```

Code (Please see attached .rar file in TBL submission):



```
SpringbootRestfulWebServicesApplication.java X UserService.java application.properties
src > main > java > net > javaguides > springboot > SpringbootRestfulWebServicesApplication.java > ...
1 package net.javaguides.springboot;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class SpringbootRestfulWebServicesApplication {
8
9     Run | Debug
10     public static void main(String[] args) {
11         SpringApplication.run(SpringbootRestfulWebServicesApplication.class, args);
12     }
13 }
14
```



```
SpringbootRestfulWebServicesApplication.java UserService.java X
src > main > java > net > javaguides > springboot > service > UserService.java
1 package net.javaguides.springboot.service;
2
3 import net.javaguides.springboot.entity.User;
4
5 import java.util.List;
6
7 public interface UserService {
8     User createUser(User user);
9
10    User getUserById(Long userId);
11
12    List<User> getAllUsers();
13
14    User updateUser(User user);
15
16    void deleteUser(Long userId);
17 }
```

```
J SpringbootRestfulWebServicesApplication.java  J UserService.java  J UserServiceImpl.java x
src > main > java > net > javaguides > springboot > service > impl > J UserServiceImpl.java > {} net.javaguides.springboot.service.impl
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.springframework.stereotype.Service;
8
9 import java.util.List;
10 import java.util.Optional;
11
12 @Service
13 @AllArgsConstructor
14 public class UserServiceImpl implements UserService {
15
16     private UserRepository userRepository;
17
18     @Override
19     public User createUser(User user) {
20         return userRepository.save(user);
21     }
22
23     @Override
24     public User getUserById(Long userId) {
25         Optional<User> optionalUser = userRepository.findById(userId);
26         return optionalUser.get();
27     }
28
29     @Override
30     public List<User> getAllUsers() {
31         return userRepository.findAll();
32     }
33
34     @Override
35     public User updateUser(User user) {
36         User existingUser = userRepository.findById(user.getId()).get();
37         existingUser.setFirstName(user.getFirstName());
38         existingUser.setLastName(user.getLastName());
39         existingUser.setEmail(user.getEmail());
40         User updatedUser = userRepository.save(existingUser);
41         return updatedUser;
42     }
43
44     @Override
45     public void deleteUser(Long userId) {
46         userRepository.deleteById(userId);
47     }
48 }
```



```
SpringbootRestfulWebservicesApplication.java  UserRepository.java  application.properties X
src > main > resources > application.properties
1  spring.application.name=springboot-restful-webservices
2  spring.datasource.url=jdbc:mysql://localhost:3308/user_management
3  spring.datasource.username=root
4  spring.datasource.password=Jeremiah213123
5  server.port=8000
6
7  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
8  spring.jpa.hibernate.ddl-auto=update
```

```
SpringbootRestfulWebservicesApplication.java  application.properties  SpringbootRestfulWebservicesApplicationTests.java X
src > test > java > net > javaguides > springboot > SpringbootRestfulWebservicesApplicationTests.java > {} net.javaguides.springboot
1  package net.javaguides.springboot;
2
3  import org.junit.jupiter.api.Test;
4  import org.springframework.boot.test.context.SpringBootTest;
5
6  @SpringBootTest
7  class SpringbootRestfulWebservicesApplicationTests {
8
9      @Test
10     void contextLoads() {
11     }
12
13 }
14
```