

Jeremiah Dela Vega

IV-BCSAD

Elective 3: Assignment 1 - REST API

Output:

POST (User Creation)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (selected), 'Environments', 'Flows', and 'History'. The main area has a header 'Jeremiah Dela Vega's Workspace' with tabs for 'Home', 'Workspaces', and 'API Network'. A search bar at the top right says 'Search Postman'. Below it, a collection named 'New Collection' is selected, with a sub-item 'POST http://localhost:8000/api/users'. The main panel shows a POST request to 'http://localhost:8000/api/users'. The 'Body' tab is active, showing raw JSON input:

```
1 {
2   "firstName": "Jeremiah",
3   "lastName": "Dela Vega",
4   "email": "jdelavegaa.k5150204@gmail.com"
5 }
```

Below the request, the response is shown under the 'Body' tab, labeled '201 Created'. The response body is identical to the input:

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

At the bottom of the interface, there are various status indicators and links like 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (selected), 'Environments', 'Flows', and 'History'. The main area has tabs for 'Overview', 'POST http://localhost:8000/...', and 'New Collection / http://localhost:8000/api/users'. Below these are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Scripts', and 'Settings'. Under 'Body', the 'raw' tab is selected, showing a JSON payload:

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

Below the body, the response status is '201 Created' with a timestamp of '11 ms' and a size of '259 B'. The response body is also shown in JSON format:

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

At the bottom, there are buttons for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

GET (Single User)

The screenshot shows the Postman application interface. In the top navigation bar, 'Jeremiah Dela Vega's Workspace' is selected. The main workspace displays a collection named 'New Collection' with a single endpoint: 'GET http://localhost:8000/api/users/4'. The 'Body' tab is selected, showing the raw JSON response. The response status is '200 OK' with a timestamp of '19 ms'. The JSON data returned is:

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

GET (All Users)

The screenshot shows the Postman application interface. On the left, the sidebar displays 'Jeremiah Dela Vega's Workspace' with sections for Collections, Environments, Flows, and History. The main area shows a collection named 'New Collection / http://localhost:8000/api/users'. A GET request is selected with the URL 'http://localhost:8000/api/users'. The 'Body' tab is active, showing the raw JSON response. The response is a list of user objects:

```
[{"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": "Mish", "lastName": "", "email": ""}]
```

The status bar at the bottom indicates a 200 OK response with 117 ms latency and 537 B size.

PUT (Update)

The screenshot shows the Postman application interface. On the left, the sidebar displays 'Jeremiah Dela Vega's Workspace' with sections for Collections, Environments, Flows, and History. A 'New Collection' option is also present. The main workspace shows a collection titled 'New Collection / http://localhost:8000/api/users'. Inside this collection, there is a single request labeled 'PUT http://localhost:8000/api/users/4'. The 'Body' tab is selected, showing a raw JSON payload:

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

Below the request, the response details are shown: '200 OK' status, '15 ms' duration, and '255 B' size. The response body is identical to the request body, indicating a successful update.

DELETE

The screenshot shows the Postman application interface. On the left, the sidebar displays 'Jeremiah Dela Vega's Workspace' with sections for Collections, Environments, Flows, and History. A 'New Collection' is currently selected. In the main workspace, a collection named 'New Collection' is shown with a single item: a POST request to `http://localhost:8000/api/users`. This item has been modified to a DELETE request to `http://localhost:8000/api/users/4`. The 'Body' tab is selected, showing the following JSON payload:

```
1 {  
2   "firstName": "Mishh",  
3   "lastName": "Lifakha",  
4   "email": "jdelavega2a.k5150284@gmail.com"  
5 }
```

Below the body, the response status is **200 OK**, with a response time of 21 ms and a size of 190 B. The response content is: **User successfully deleted!**

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

A screenshot of a Java IDE showing the main application class. The code is as follows:

```
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     public static void main(String[] args) {
10         SpringApplication.run(SpringbootRestfulWebservicesApplication.class, args);
11     }
12
13 }
14
```

A screenshot of a Java IDE showing the service interface. The code is as follows:

```
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 }
```

The screenshot shows a Java code editor with three tabs at the top: `SpringbootRestfulWebservicesApplication.java`, `UserService.java`, and `UserServiceImpl.java`. The `UserServiceImpl.java` tab is active. The code is a Spring Boot service implementation for managing users.

```
src > main > java > net > javaguides > springboot > service > impl > 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

```
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

```
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 }
```