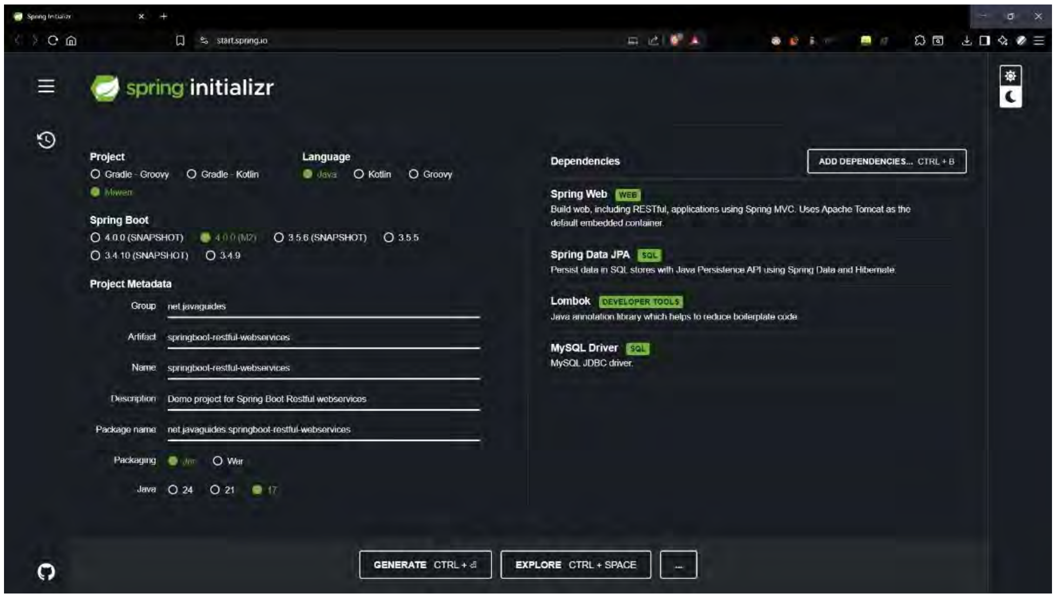
MARK ANGELO D. SIAZON IV-ACSAD

MARK ANGELO D. SIAZON IV-ACSAD ELEC-3

1. Create a Spring Boot Application and Import in IntelliJ IDEA or Eclipse or VS Code



Project: Gradle - Groovy, Gradle - Kotlin, [✓ Maven]

Language: [✓ Java], Kotlin, Groovy

Spring Boot: 4.0.0 (SNAPSHOT), [✓ 4.0.0 (M2)], 3.5.6 (SNAPSHOT), 3.5.5, 3.4.10 (SNAPSHOT), 3.4.9

Project Metadata

Group	net.javaguides
Artifact	springboot-restful-webservices
Name	springboot-restful-webservices
Description	Demo project for Spring Boot Restful webservices
Package name	net.javaguides.springboot-restful-webservices
Packaging	[✓ Jar], War
Java	24, 21, [✓ 17]

Dependencies — ADD DEPENDENCIES... CTRL + B

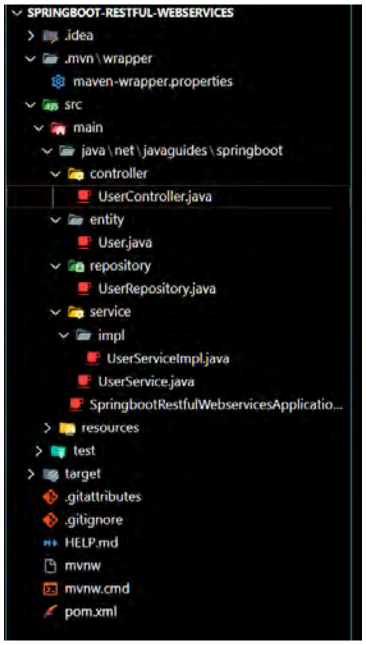
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

MySQL Driver SQL
MySQL JDBC driver.

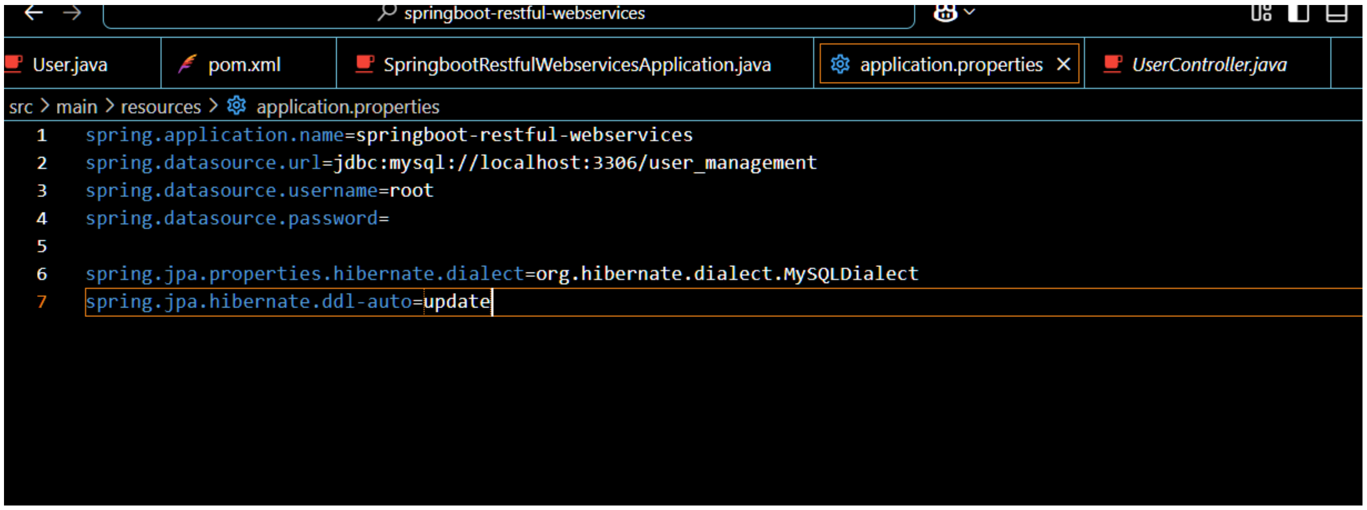
2.Project Structure



```
SPRINGBOOT-RESTFUL-WEBSERVICES
 > .idea
 v .mvn \ wrapper
     maven-wrapper.properties
 v src
   v main
     v java \ net \ javaguides \ springboot
       v controller
           UserController.java
       v entity
           User.java
       v repository
           UserRepository.java
       v service
         v impl
             UserServiceImpl.java
           UserService.java
         SpringbootRestfulWebservicesApplicatio...
     > resources
   > test
 > target
   .gitattributes
   .gitignore
   HELP.md
   mvnw
   mvnw.cmd
   pom.xml
```

MARK ANGELO D. SIAZON IV-ACSAD

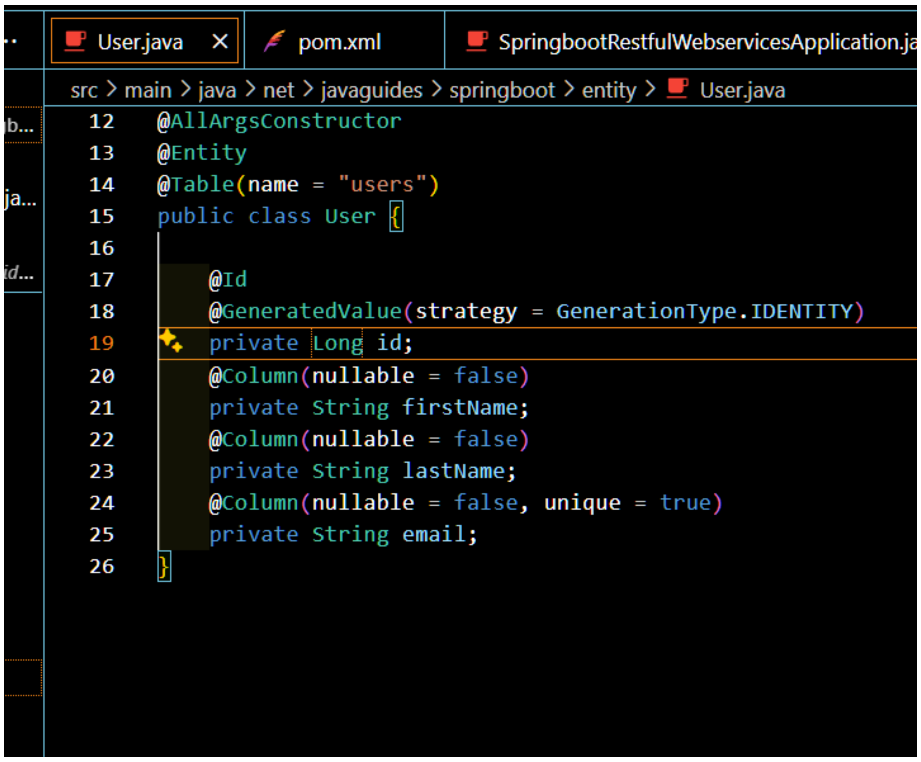
3. Configuring MySQL Database



springboot-restful-webservices
User.java
pom.xml
SpringbootRestfulWebservicesApplication.java
application.properties
UserController.java
src > main > resources > application.properties

```
spring.application.name=springboot-restful-webservices
spring.datasource.url=jdbc:mysql://localhost:3306/user_management
spring.datasource.username=root
spring.datasource.password=

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
```

4. Create JPA Entity - User.java

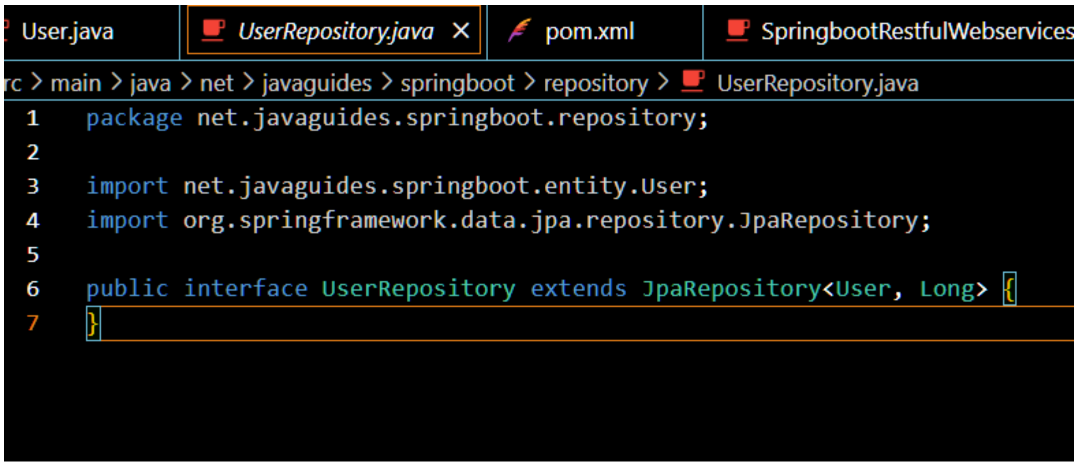


User.java
pom.xml
SpringbootRestfulWebservicesApplication.ja
src > main > java > net > javaguides > springboot > entity > User.java

```
@AllArgsConstructor
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String firstName;
    @Column(nullable = false)
    private String lastName;
    @Column(nullable = false, unique = true)
    private String email;
}
```

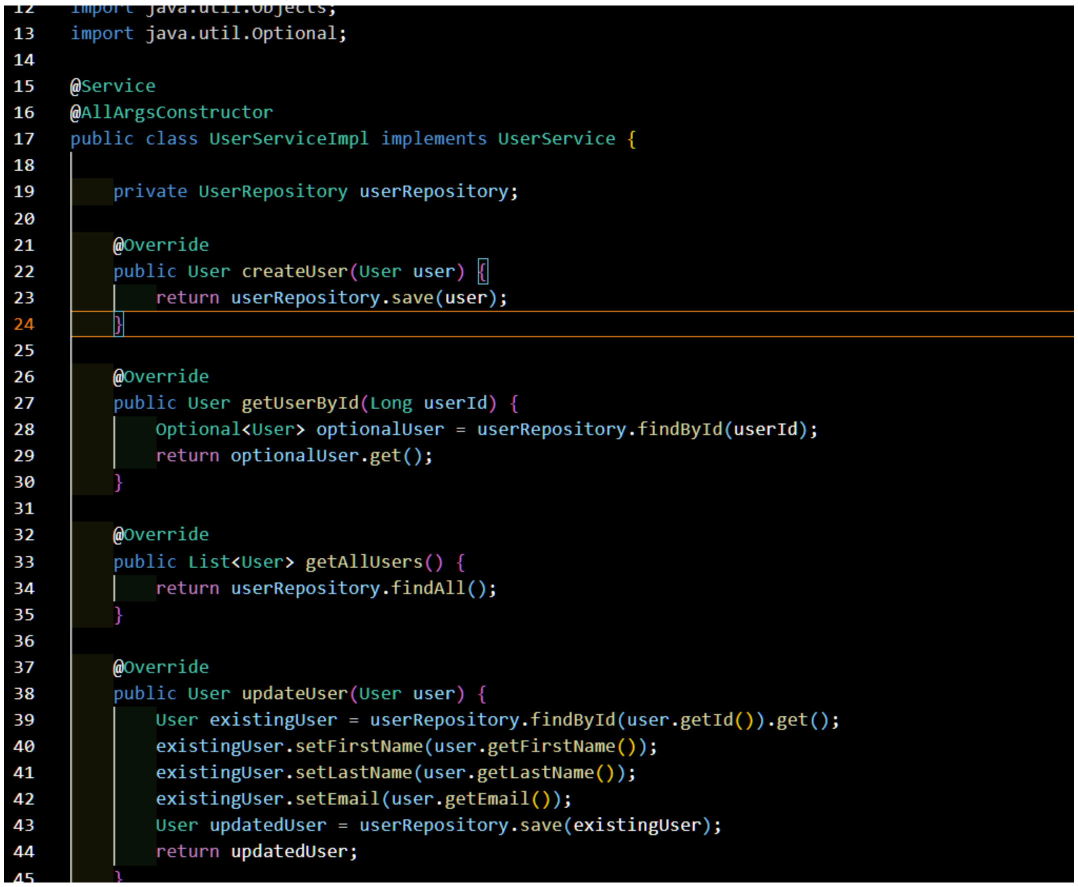
MARK ANGELO D. SIAZON IV-ACSAD

5. Create Spring Data JPA Repository for User JPA Entity


User.java
UserRepository.java
pom.xml
SpringbootRestfulWebservices
rc > main > java > net > javaguides > springboot > repository > UserRepository.java

```java
package net.javaguides.springboot.repository;

import net.javaguides.springboot.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
}
```

6. Service Layer Implementation

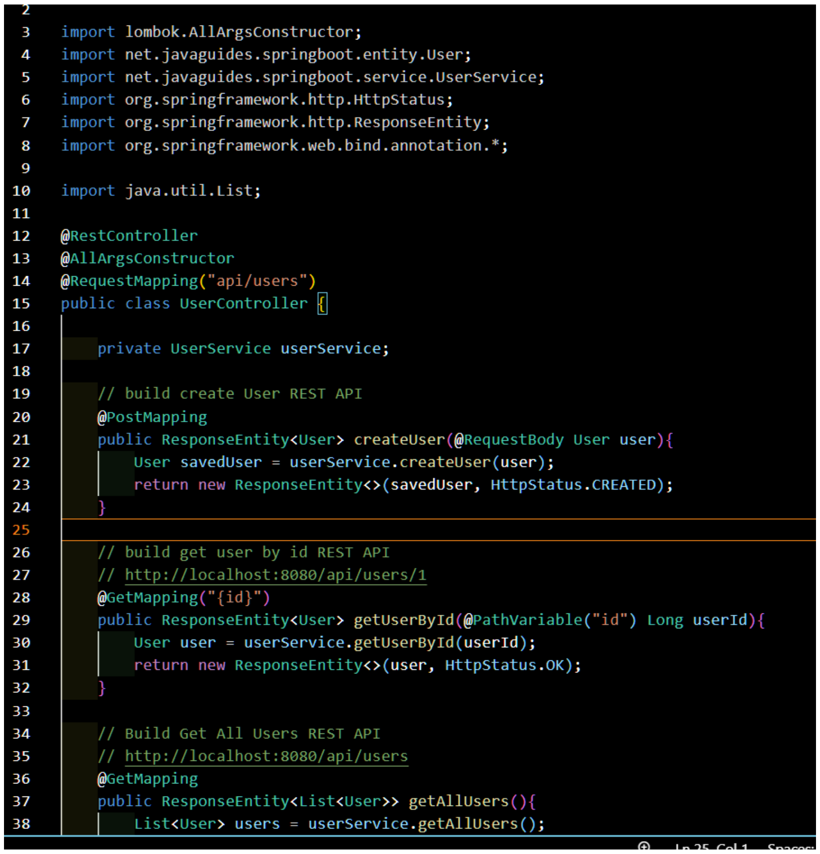


```java
import java.util.Objects;
import java.util.Optional;

@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;
    }
```

MARK ANGELO D. SIAZON IV-ACSAD

7. Creating UserController - Building CRUD Rest APIs7.

Creating UserController - Building CRUD Rest APIs

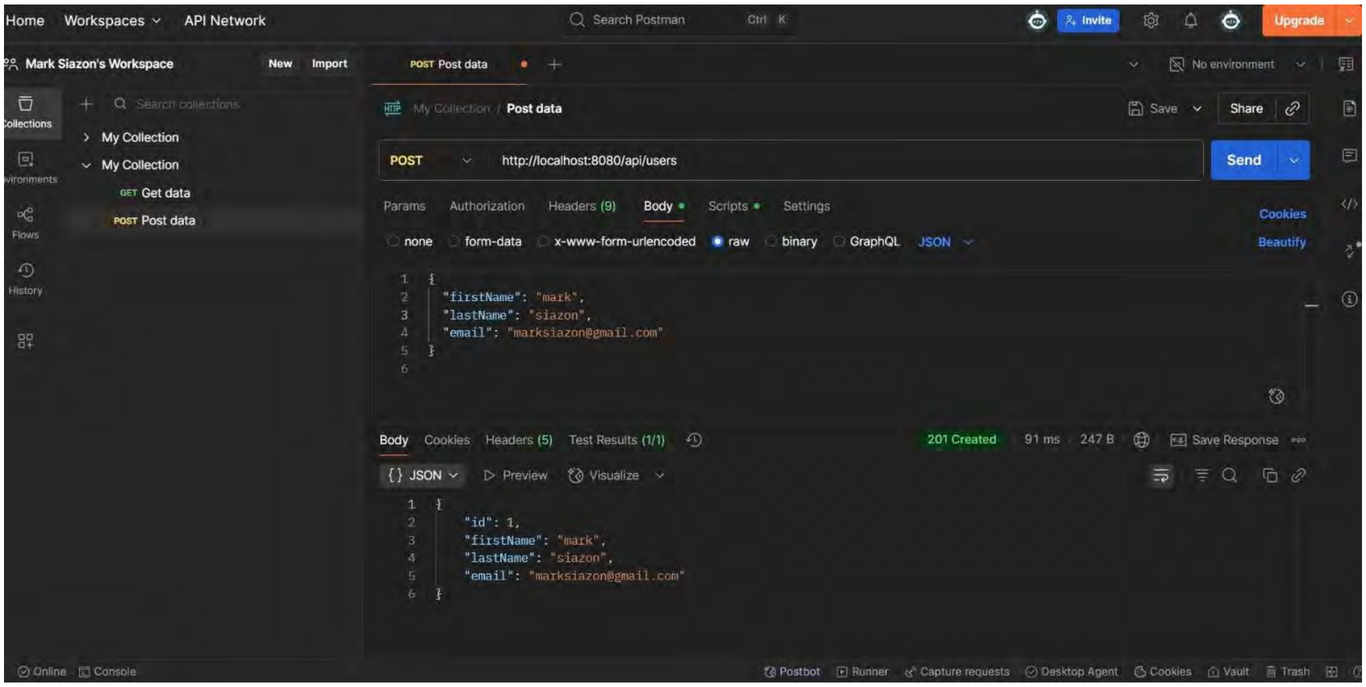


```
import lombok.AllArgsConstructor;
import net.javaguides.springboot.entity.User;
import net.javaguides.springboot.service.UserService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

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();
```

Now Test Spring Boot CRUD REST APIs using Postman Client

MARK ANGELO D. SIAZON IV-ACSAD

1-Create User REST API



POST http://localhost:8080/api/users

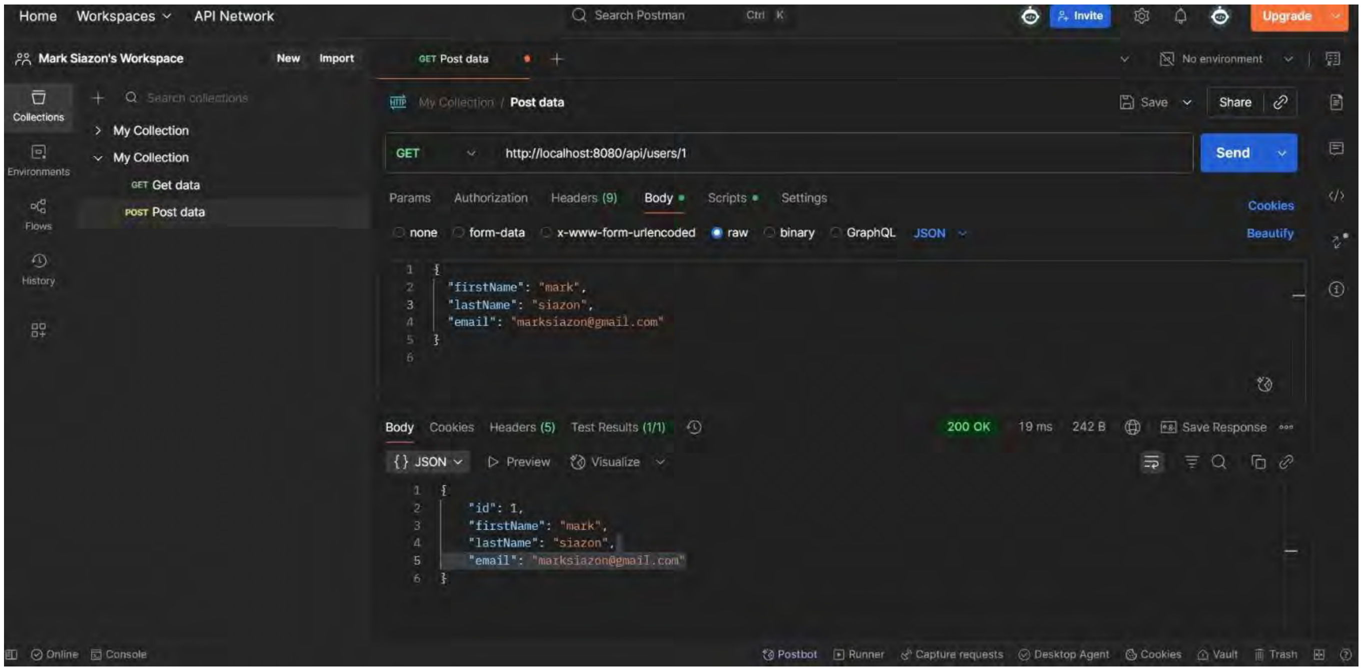
Body: raw JSON

```
{
  "firstName": "mark",
  "lastName": "siazon",
  "email": "marksiazon@gmail.com"
}
```

201 Created 91 ms 247 B

```
{
  "id": 1,
  "firstName": "mark",
  "lastName": "siazon",
  "email": "marksiazon@gmail.com"
}
```

2-Get Single User REST API



GET http://localhost:8080/api/users/1

Body: raw JSON

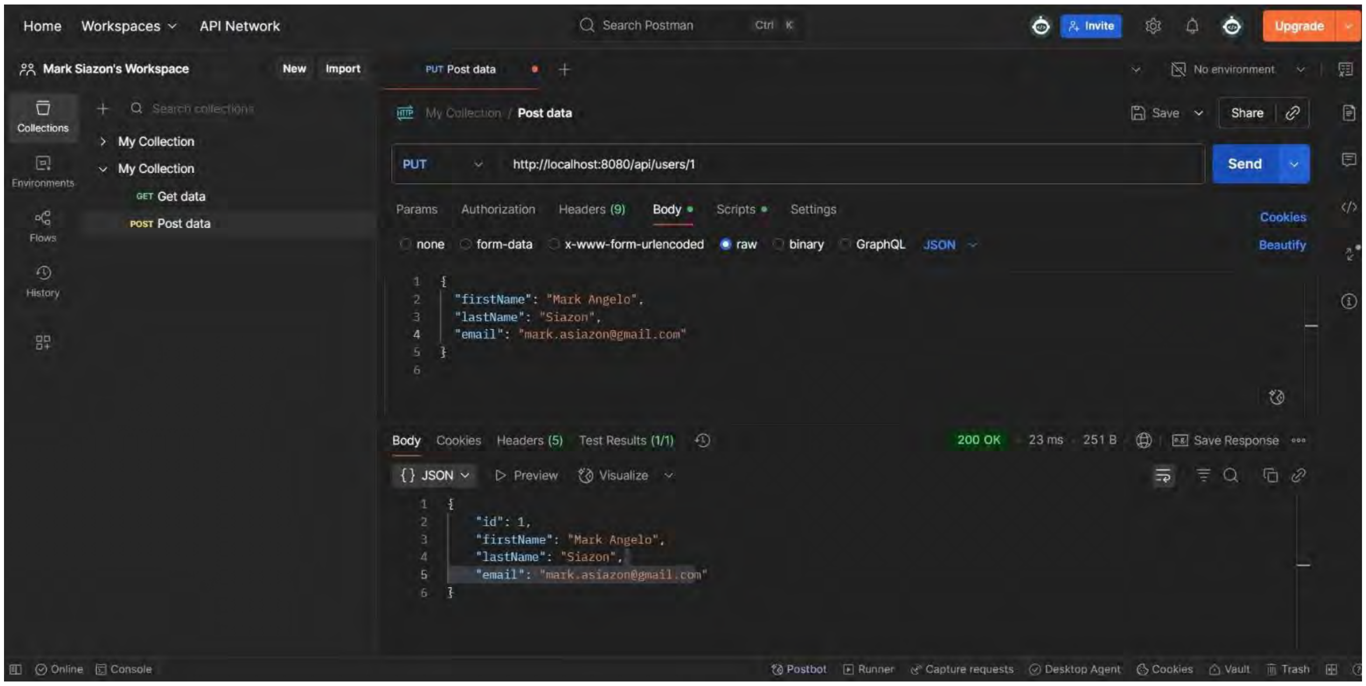
```
{
  "firstName": "mark",
  "lastName": "siazon",
  "email": "marksiazon@gmail.com"
}
```

200 OK 19 ms 242 B

```
{
  "id": 1,
  "firstName": "mark",
  "lastName": "siazon",
  "email": "marksiazon@gmail.com"
}
```

MARK ANGELO D. SIAZON   IV-ACSAD

3-Update User REST API



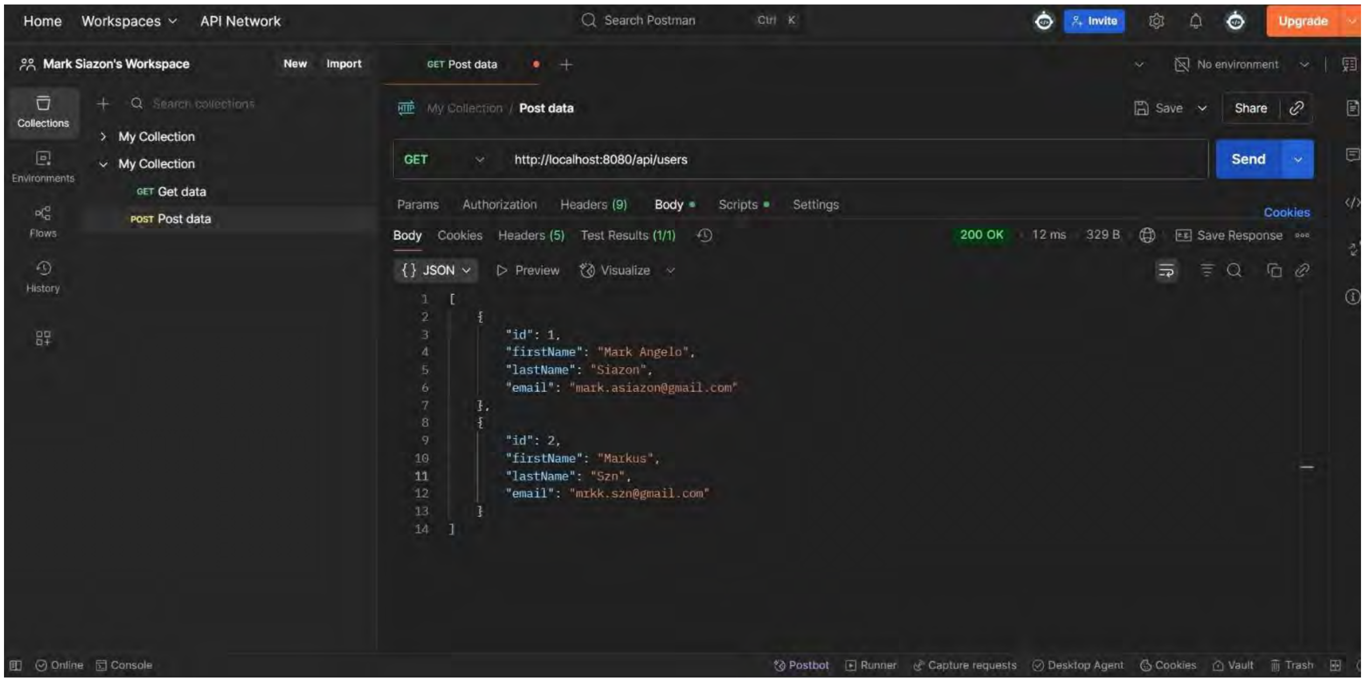
PUT http://localhost:8080/api/users/1

```
{
  "firstName": "Mark Angelo",
  "lastName": "Siazon",
  "email": "mark.asiazon@gmail.com"
}
```

200 OK 23 ms 251 B

```
{
    "id": 1,
    "firstName": "Mark Angelo",
    "lastName": "Siazon",
    "email": "mark.asiazon@gmail.com"
}
```

4-Get All Users REST API



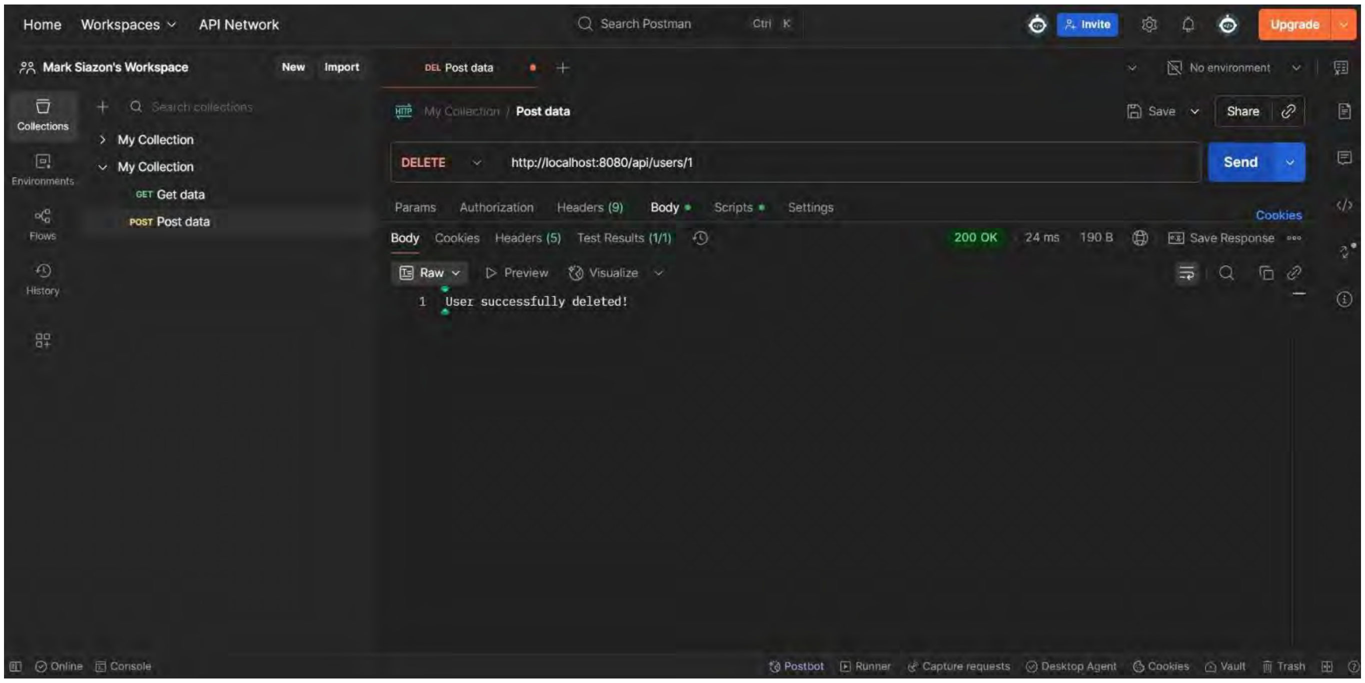
GET http://localhost:8080/api/users

200 OK 12 ms 329 B

```
[
    {
        "id": 1,
        "firstName": "Mark Angelo",
        "lastName": "Siazon",
        "email": "mark.asiazon@gmail.com"
    },
    {
        "id": 2,
        "firstName": "Markus",
        "lastName": "Szn",
        "email": "mrkk.szn@gmail.com"
    }
]
```

MARK ANGELO D. SIAZON IV-ACSAD

5 - Delete User REST API



DELETE http://localhost:8080/api/users/1

```
User successfully deleted!
```