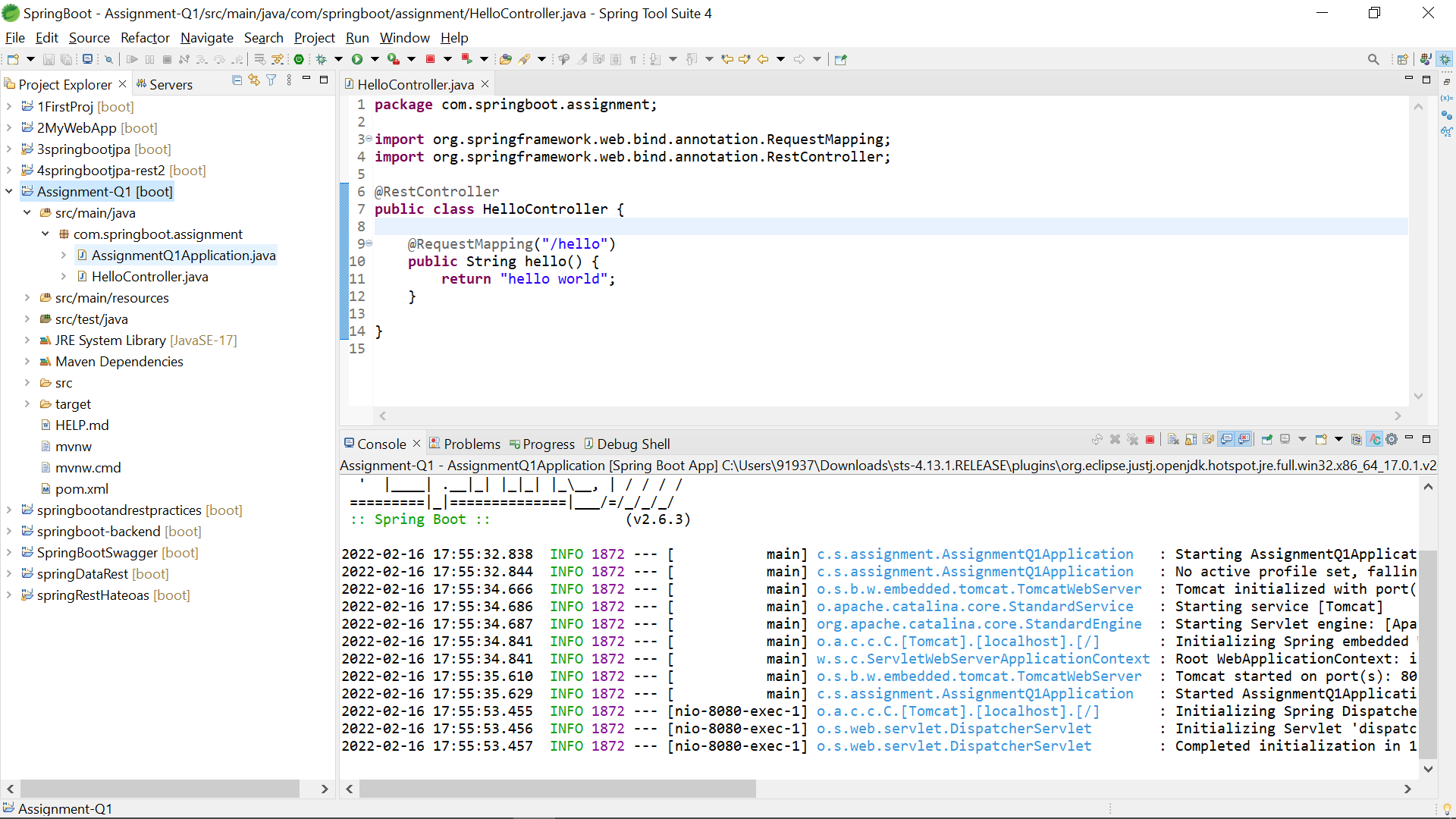
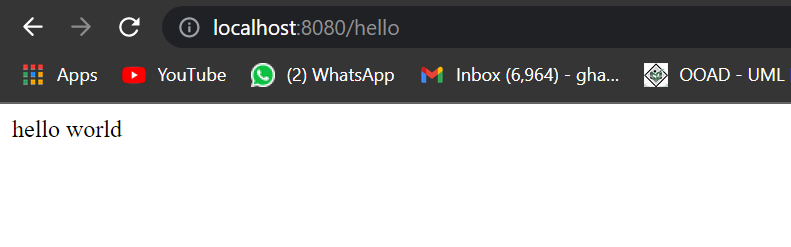
1. Create a RESTful web service that returns "Hello World" message.





2) Create a RESTful web service that authenticates an user. User will specify his/her credentials i.e. username and password. If username and password are correct, It should return "valid user" message, else "Invalid user" message.

Credentials.java

**package** springboot.assignment.model;

**public** **class** Credentials {

**private** String username;

**private** String password;

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

CredentialService.java

**package** springboot.assignment.service;

**public** **interface** CredentialService {

**public** String validate(String username,String password);

}

CredentialServiceImpl.java

**package** springboot.assignment.service.impl;

**import** org.springframework.stereotype.Service;

**import** springboot.assignment.service.CredentialService;

@Service

**public** **class** CredentialServiceImpl **implements** CredentialService {

**public** String user = "Prad";

**public** String pass = "Prad\_9";

@Override

**public** String validate(String username, String password) {

**if** (username.equals(username) && pass.equals(password)) {

**return** "Valid User";

}

**else**

**return** "Invalid User";

}

}

CredentialController.java

**package** springboot.assignment;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** springboot.assignment.service.CredentialService;

@RestController

@RequestMapping("/credentials")

**public** **class** CredentialController {

@Autowired

CredentialService credentialService;

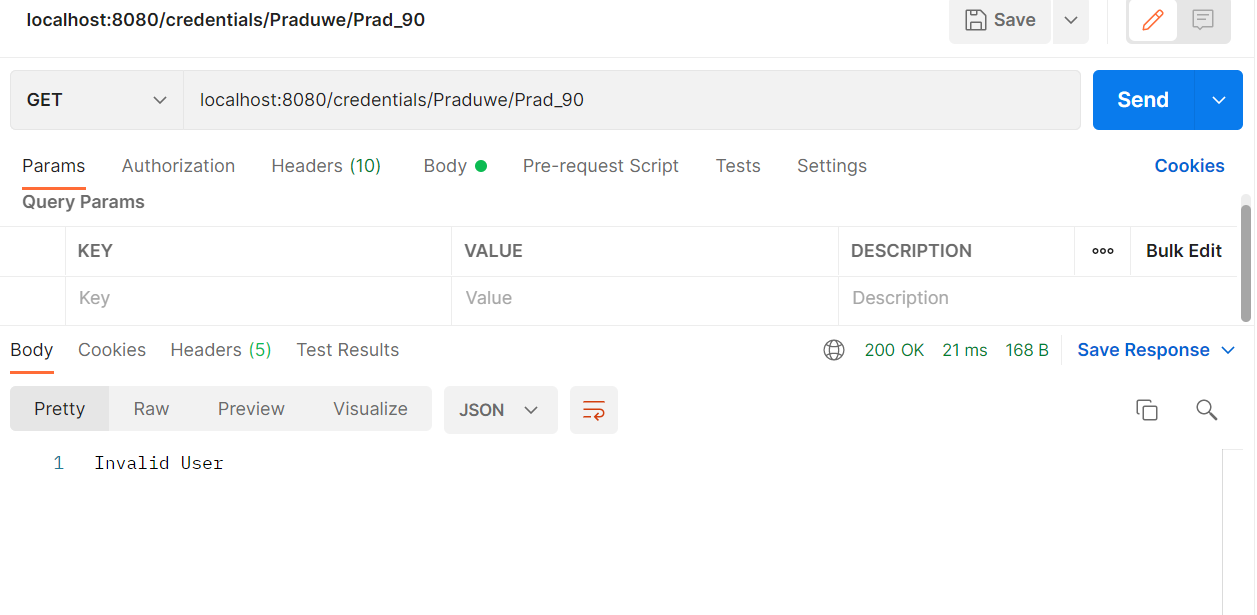
@GetMapping("/{username}/{password}")

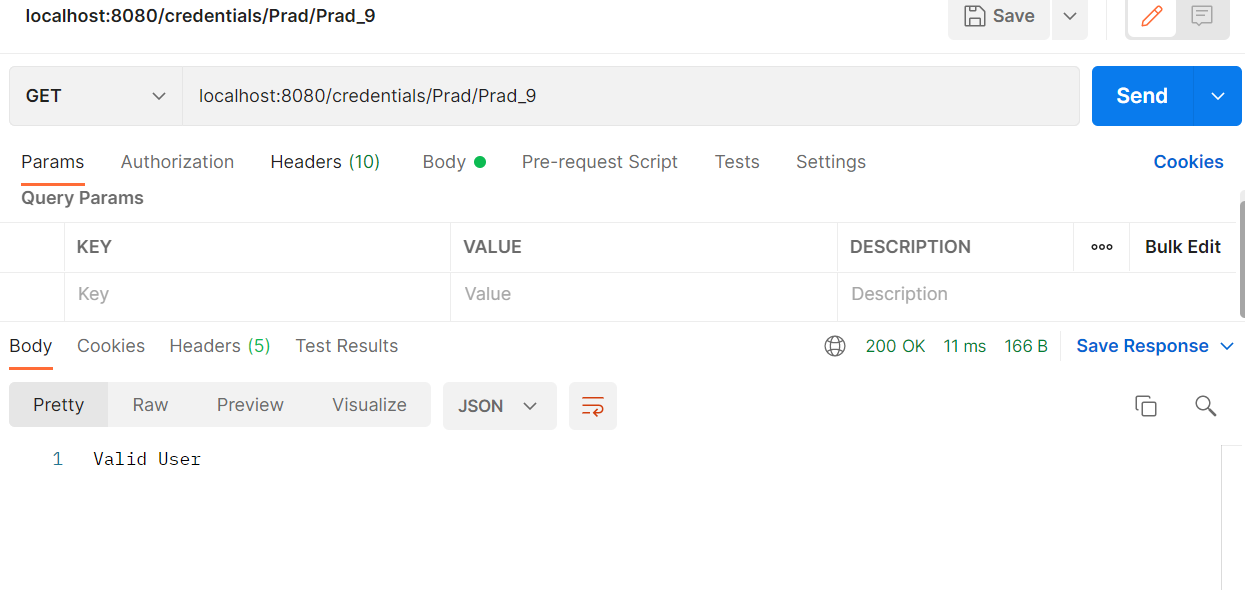
**public** String validate(@PathVariable("username") String username, @PathVariable("password") String password) {

**return** credentialService.validate(username, password);

}

}





3) Create a RESTful web service that returns state, city and country information when user passes zipcode. You can send state, city and country information in JSON format. Sample Input: 99501 Sample output: {"state": "AK", City: "ANCHORAGE", "country: "US"}

Address.java

**package** com.springboot.assignment.model;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name = "address")

**public** **class** Address {

@Id

@Column(name = "zipcode")

**private** **long** zipcode;

@Column(name = "city")

**private** String city;

@Column(name = "state")

**private** String state;

@Column(name = "country")

**private** String country;

**public** **long** getZipcode() {

**return** zipcode;

}

**public** **void** setZipcode(**long** zipcode) {

**this**.zipcode = zipcode;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

@Override

**public** String toString() {

**return** "Address [city=" + city + ", state=" + state + ", country=" + country + "]";

}

}

AddressRepository.java

**package** com.springboot.assignment.repo;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** com.springboot.assignment.model.Address;

@Repository

**public** **interface** AddressRepository **extends** JpaRepository<Address, Long> {

}

AddressController.java

**package** com.springboot.assignment.controller;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.springboot.assignment.model.Address;

**import** com.springboot.assignment.repo.AddressRepository;

@RestController

**public** **class** AddressController {

@Autowired

AddressRepository addressRepository;

@PostMapping("/address")

**public** Address addAddress(@RequestBody Address address) {

addressRepository.save(address);

**return** address;

}

@GetMapping("/address")

**public** List<Address> getAddresses() {

**return** addressRepository.findAll();

}

@GetMapping("/address/{zipcode}")

**public** Optional<Address> getAddress(@PathVariable("zipcode") **long** zipcode) {

**return** addressRepository.findById(zipcode);

}

}

Application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/cadd?useSSL=false

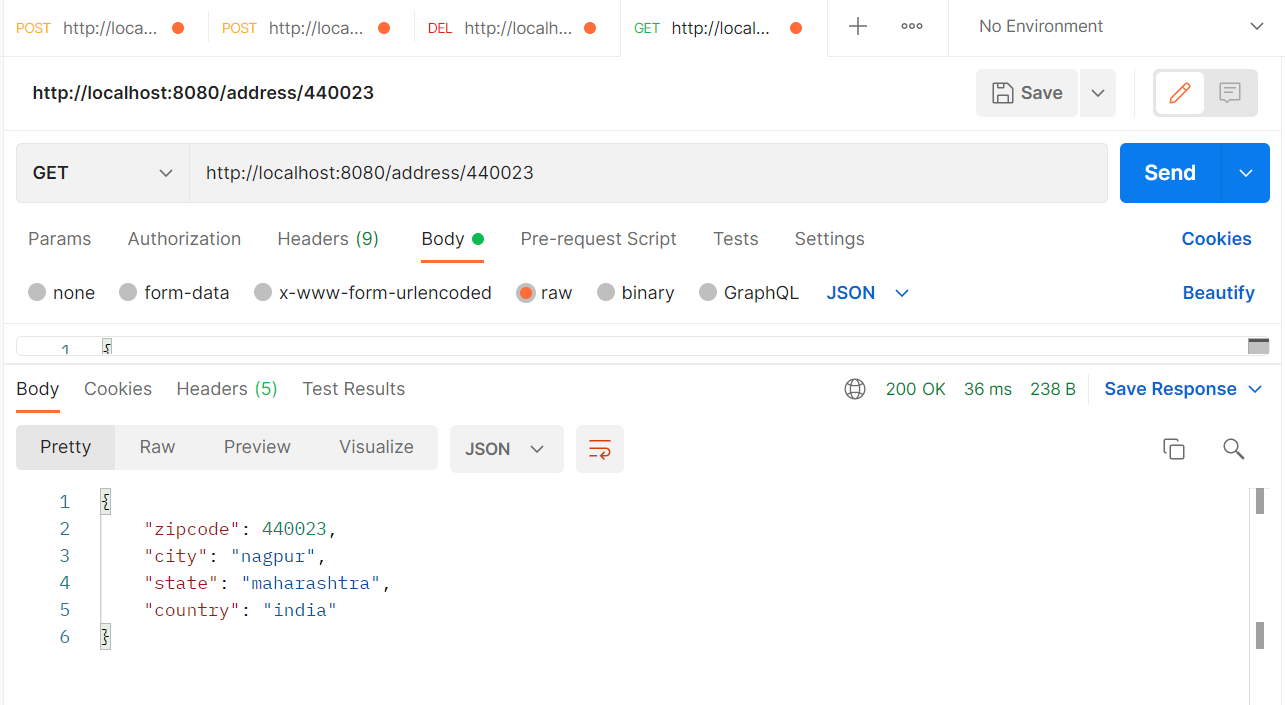
spring.datasource.username=root

spring.datasource.password=Pradumnya@9

#Hibernate properties

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=update



4) 4) Create a RESTful web service that validates the credit card. It means we need to check the type of credit card like American Express, Discover, Visa etc and it is valid or not. Valid length: 16 digits.

Controller.java

**package** springboot.assignment.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RestController;

**import** springboot.assignment.service.Service;

@RestController

**public** **class** Controller {

@Autowired

**private** Service service;

@GetMapping(path = "/credit-card/{id}")

**public** String creditCard(@PathVariable("id") **long** number) {

**boolean** ans = service.isValid(number);

**if** (ans == **true**) {

**return** "Credit card is valid";

}

**return** "Credit card is not valid";

}

}

Service.java

**package** springboot.assignment.service;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Service {

**public** **boolean** isValid(**long** number) {

**return** (*getSize*(number) >= 13 &&

*getSize*(number) <= 16) &&

(*prefixMatched*(number, 4) ||

*prefixMatched*(number, 5) ||

*prefixMatched*(number, 37) ||

*prefixMatched*(number, 6))

&&

((*sumOfDoubleEvenPlace*(number) +

*sumOfOddPlace*(number)) % 10 == 0);

}

// Get the result from Step 2

**public** **static** **int** sumOfDoubleEvenPlace(**long** number) {

**int** sum = 0;

String num = number + "";

**for** (**int** i = *getSize*(number) - 2; i >= 0; i -= 2)

sum += *getDigit*(Integer.*parseInt*(num.charAt(i) + "") \* 2);

**return** sum;

}

**public** **static** **int** getDigit(**int** number) {

**if** (number < 9)

**return** number;

**return** number / 10 + number % 10;

}

**public** **static** **int** sumOfOddPlace(**long** number) {

**int** sum = 0;

String num = number + "";

**for** (**int** i = *getSize*(number) - 1; i >= 0; i -= 2)

sum += Integer.*parseInt*(num.charAt(i) + "");

**return** sum;

}

**public** **static** **boolean** prefixMatched(**long** number, **int** d) {

**return** *getPrefix*(number, *getSize*(d)) == d;

}

**public** **static** **int** getSize(**long** d) {

String num = d + "";

**return** num.length();

}

**public** **static** **long** getPrefix(**long** number, **int** k) {

**if** (*getSize*(number) > k) {

String num = number + "";

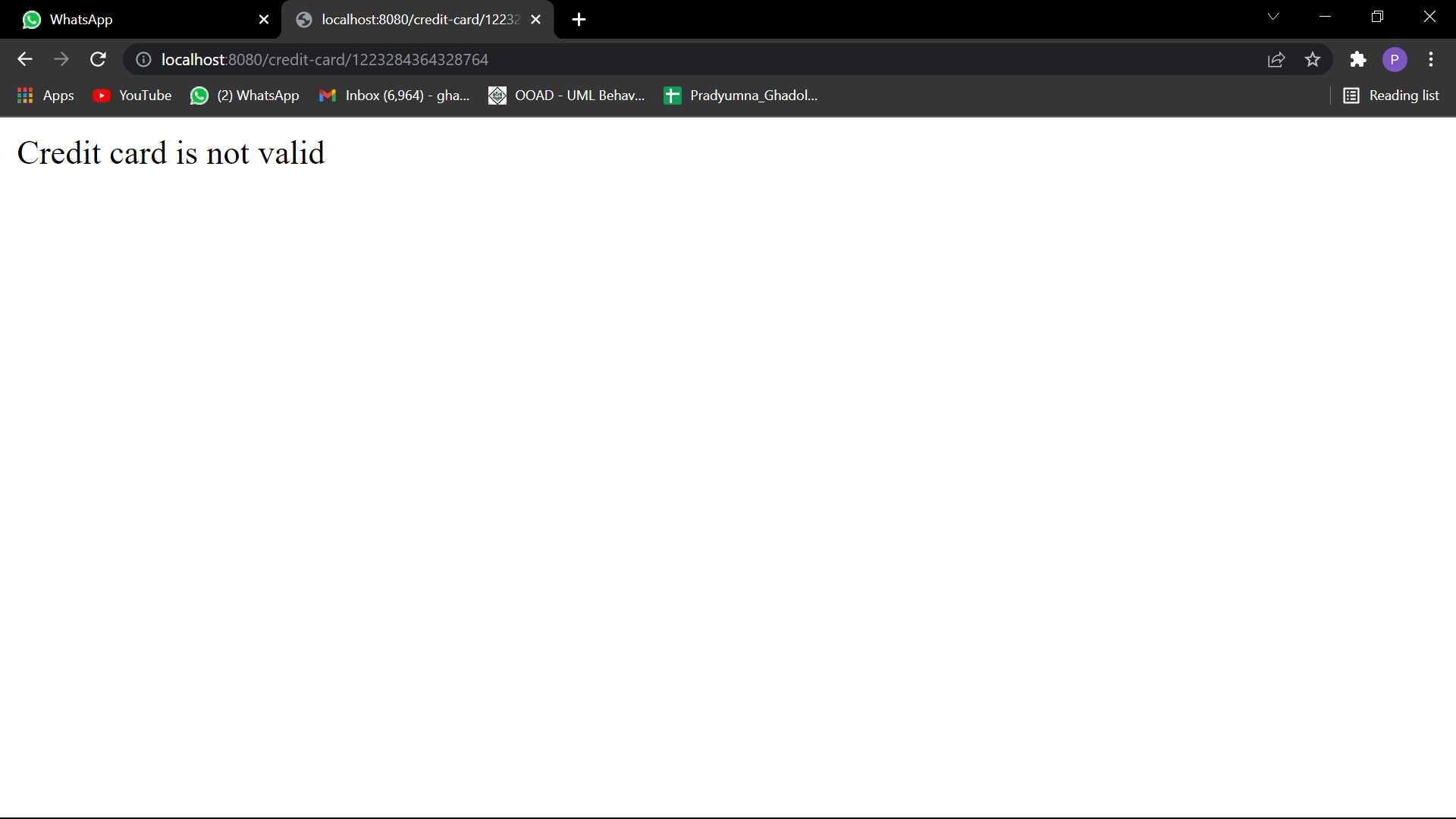
**return** Long.*parseLong*(num.substring(0, k));

}

**return** number;

}

}



5) Develop RESTful web services for "Employee Management System" that manages the information about employees

1. Add a new employee

2. Searching for specific employee

3. Deleting an existing employee

4. Finding all employees

5. Editing/updating employee information.

Create a Employee domain model class having following properties: employeeId, employeeName, employeeDepartment, employeeDesignation, employeeSalary. Employee Id should be generated automatically at database level.

Develop controller, service and repository layers classes.

Use CrudRepository from Spring Data.

Employee.java

package com.springboot.rest.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Data;

@Data

@Entity

@Table(name = "employee2")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int employeeId;

@Column(name = "empName")

private String employeeName;

@Column(name = "empDept")

private String employeeDepartment;

@Column(name = "empDesig")

private String employeeDesignation;

@Column(name = "empSalary")

private long employeeSalary;

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getEmployeeName() {

return employeeName;

}

public void setEmployeeName(String employeeName) {

this.employeeName = employeeName;

}

public String getEmployeeDepartment() {

return employeeDepartment;

}

public void setEmployeeDepartment(String employeeDepartment) {

this.employeeDepartment = employeeDepartment;

}

public String getEmployeeDesignation() {

return employeeDesignation;

}

public void setEmployeeDesignation(String employeeDesignation) {

this.employeeDesignation = employeeDesignation;

}

public long getEmployeeSalary() {

return employeeSalary;

}

public void setEmployeeSalary(long employeeSalary) {

this.employeeSalary = employeeSalary;

}

@Override

public String toString() {

return "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", employeeDepartment="

+ employeeDepartment + ", employeeDesignation=" + employeeDesignation + ", employeeSalary="

+ employeeSalary + "]";

}

}

employeeRepository.java

package com.springboot.rest.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Data;

@Data

@Entity

@Table(name = "employee2")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int employeeId;

@Column(name = "empName")

private String employeeName;

@Column(name = "empDept")

private String employeeDepartment;

@Column(name = "empDesig")

private String employeeDesignation;

@Column(name = "empSalary")

private long employeeSalary;

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getEmployeeName() {

return employeeName;

}

public void setEmployeeName(String employeeName) {

this.employeeName = employeeName;

}

public String getEmployeeDepartment() {

return employeeDepartment;

}

public void setEmployeeDepartment(String employeeDepartment) {

this.employeeDepartment = employeeDepartment;

}

public String getEmployeeDesignation() {

return employeeDesignation;

}

public void setEmployeeDesignation(String employeeDesignation) {

this.employeeDesignation = employeeDesignation;

}

public long getEmployeeSalary() {

return employeeSalary;

}

public void setEmployeeSalary(long employeeSalary) {

this.employeeSalary = employeeSalary;

}

@Override

public String toString() {

return "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", employeeDepartment="

+ employeeDepartment + ", employeeDesignation=" + employeeDesignation + ", employeeSalary="

+ employeeSalary + "]";

}

}

EmployeeController.java

package com.springboot.rest.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Data;

@Data

@Entity

@Table(name = "employee2")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int employeeId;

@Column(name = "empName")

private String employeeName;

@Column(name = "empDept")

private String employeeDepartment;

@Column(name = "empDesig")

private String employeeDesignation;

@Column(name = "empSalary")

private long employeeSalary;

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getEmployeeName() {

return employeeName;

}

public void setEmployeeName(String employeeName) {

this.employeeName = employeeName;

}

public String getEmployeeDepartment() {

return employeeDepartment;

}

public void setEmployeeDepartment(String employeeDepartment) {

this.employeeDepartment = employeeDepartment;

}

public String getEmployeeDesignation() {

return employeeDesignation;

}

public void setEmployeeDesignation(String employeeDesignation) {

this.employeeDesignation = employeeDesignation;

}

public long getEmployeeSalary() {

return employeeSalary;

}

public void setEmployeeSalary(long employeeSalary) {

this.employeeSalary = employeeSalary;

}

@Override

public String toString() {

return "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", employeeDepartment="

+ employeeDepartment + ", employeeDesignation=" + employeeDesignation + ", employeeSalary="

+ employeeSalary + "]";

}

}

EmployeeService.java

**package** com.springboot.rest.service;

**import** java.util.List;

**import** com.springboot.rest.model.Employee;

**public** **interface** EmployeeService {

Employee saveEmployee(Employee employee);

List<Employee> getAllEmployees();

Employee getEmployeeById(**int** id);

Employee updateEmployee(Employee employee, **int** id);

**void** deleteEmployee(**int** id);

}

EmployeeServiceImpl.java

**package** com.springboot.rest.service.impl;

**import** java.util.List;

**import** org.springframework.stereotype.Service;

**import** com.springboot.rest.exception.ResourceNotFoundException;

**import** com.springboot.rest.model.Employee;

**import** com.springboot.rest.repository.EmployeeRepository;

**import** com.springboot.rest.service.EmployeeService;

@Service

**public** **class** EmployeeServiceImpl **implements** EmployeeService {

**private** EmployeeRepository employeeRepository;

**public** EmployeeServiceImpl(EmployeeRepository employeeRepository) {

**super**();

**this**.employeeRepository = employeeRepository;

}

@Override

**public** Employee saveEmployee(Employee employee) {

**return** employeeRepository.save(employee);

}

@Override

**public** List<Employee> getAllEmployees() {

**return** employeeRepository.findAll();

}

@Override

**public** Employee getEmployeeById(**int** id) {

**return** employeeRepository.findById(id).orElseThrow(() -> **new** ResourceNotFoundException("Employee", "id", id));

}

@Override

**public** Employee updateEmployee(Employee employee, **int** id) {

Employee existingEmployee = employeeRepository.findById(id)

.orElseThrow(() -> **new** ResourceNotFoundException("Employee", "id", id));

existingEmployee.setEmployeeDepartment(employee.getEmployeeDepartment());

existingEmployee.setEmployeeDesignation(employee.getEmployeeDesignation());

existingEmployee.setEmployeeName(employee.getEmployeeName());

existingEmployee.setEmployeeSalary(employee.getEmployeeSalary());

employeeRepository.save(existingEmployee);

**return** existingEmployee;

}

@Override

**public** **void** deleteEmployee(**int** id) {

employeeRepository.findById(id).orElseThrow(() -> **new** ResourceNotFoundException("Employee", "id", id));

employeeRepository.deleteById(id);

}

}

ResourceNotFoundException.java

**package** com.springboot.rest.exception;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.***NOT\_FOUND***)

**public** **class** ResourceNotFoundException **extends** RuntimeException{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** String resourceName;

**private** String fieldName;

**private** Object fieldValue;

**public** ResourceNotFoundException(String resourceName, String fieldName, Object fieldValue) {

**super**(String.*format*("%s not found with %s : '%s'",resourceName,fieldName,fieldValue));

**this**.resourceName = resourceName;

**this**.fieldName = fieldName;

**this**.fieldValue = fieldValue;

}

**public** String getResourceName() {

**return** resourceName;

}

**public** String getFieldName() {

**return** fieldName;

}

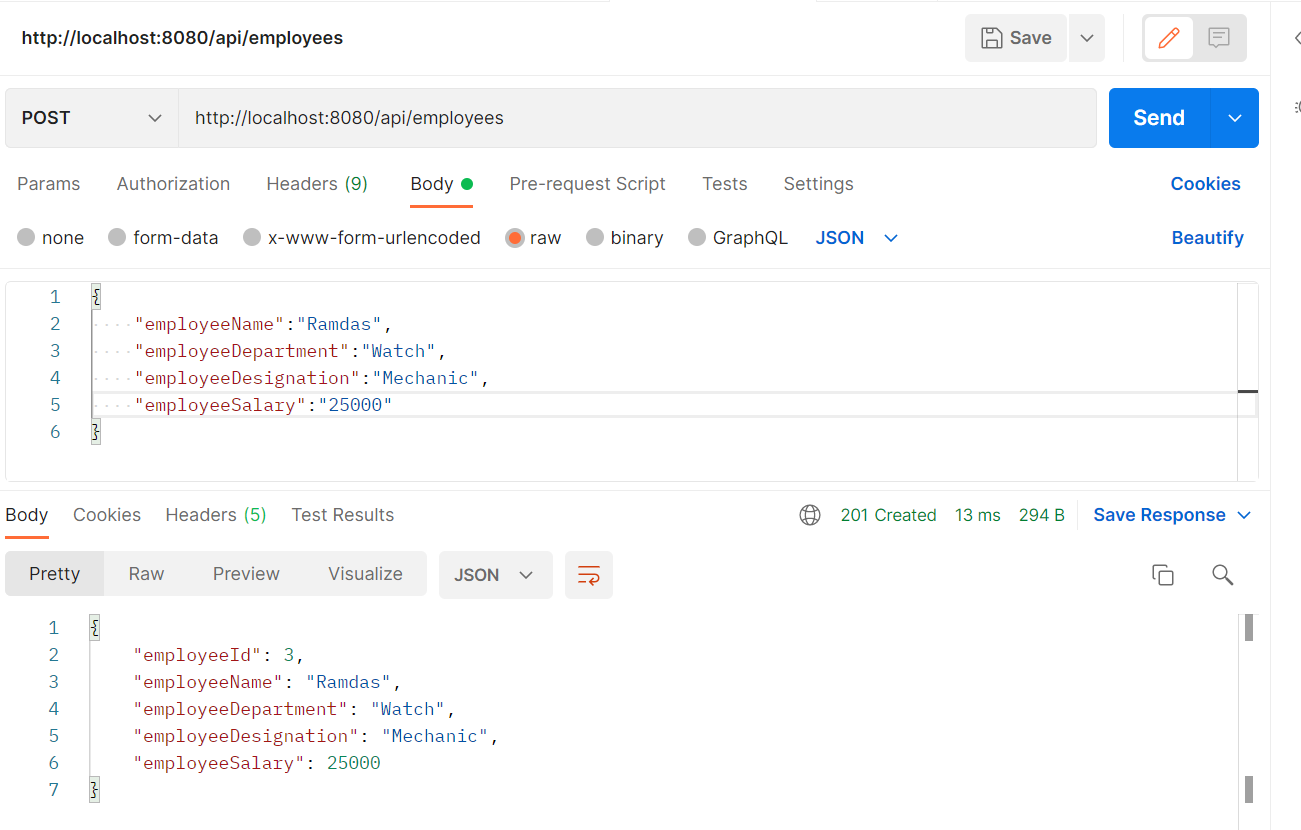
**public** Object getFieldValue() {

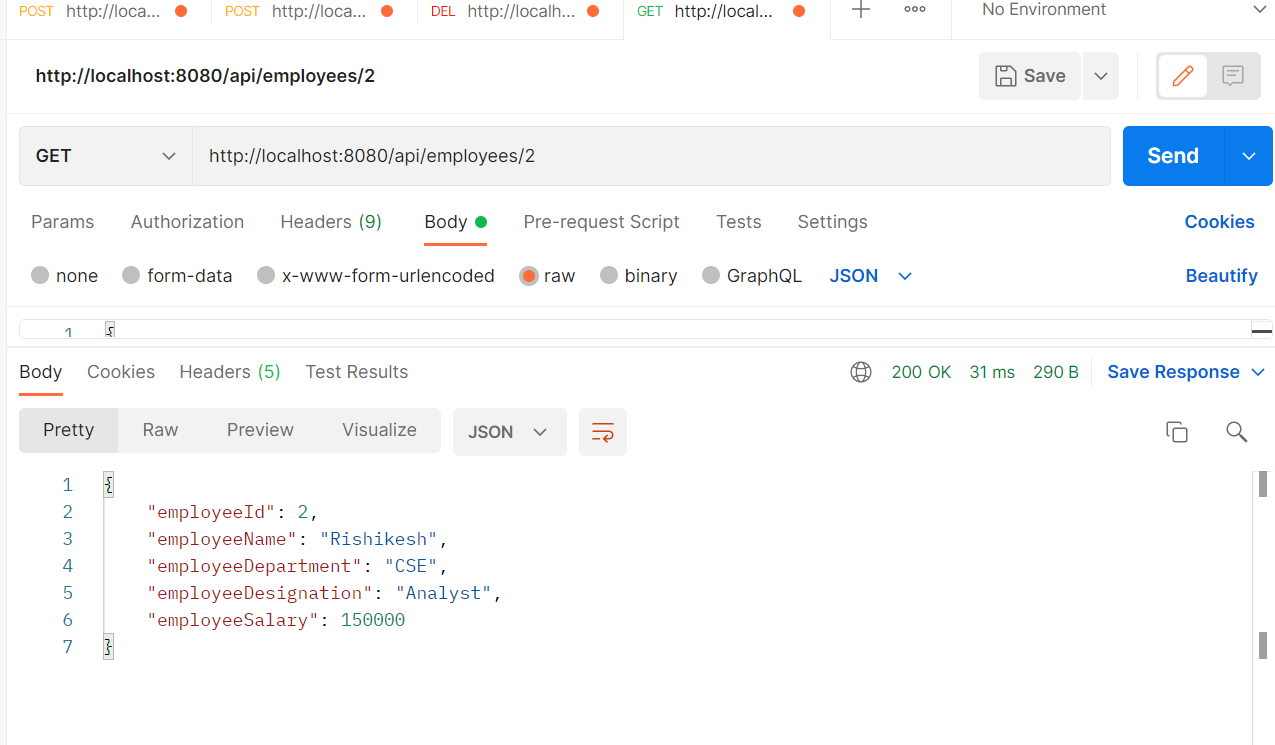
**return** fieldValue;

}

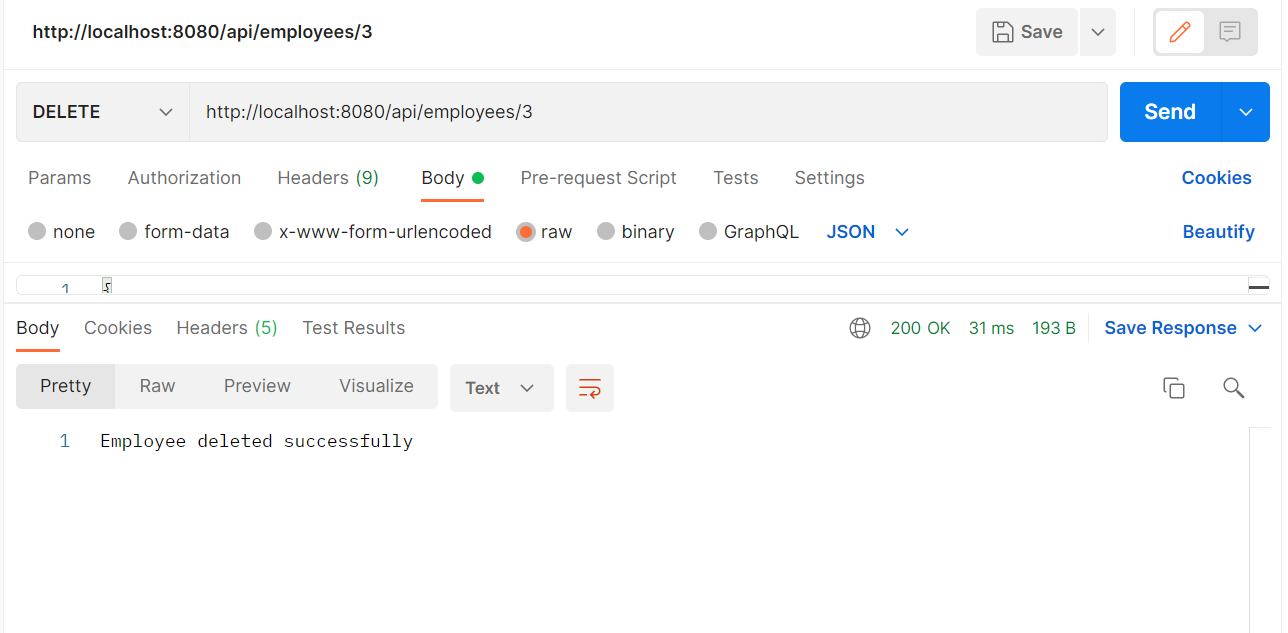
}

1. adding

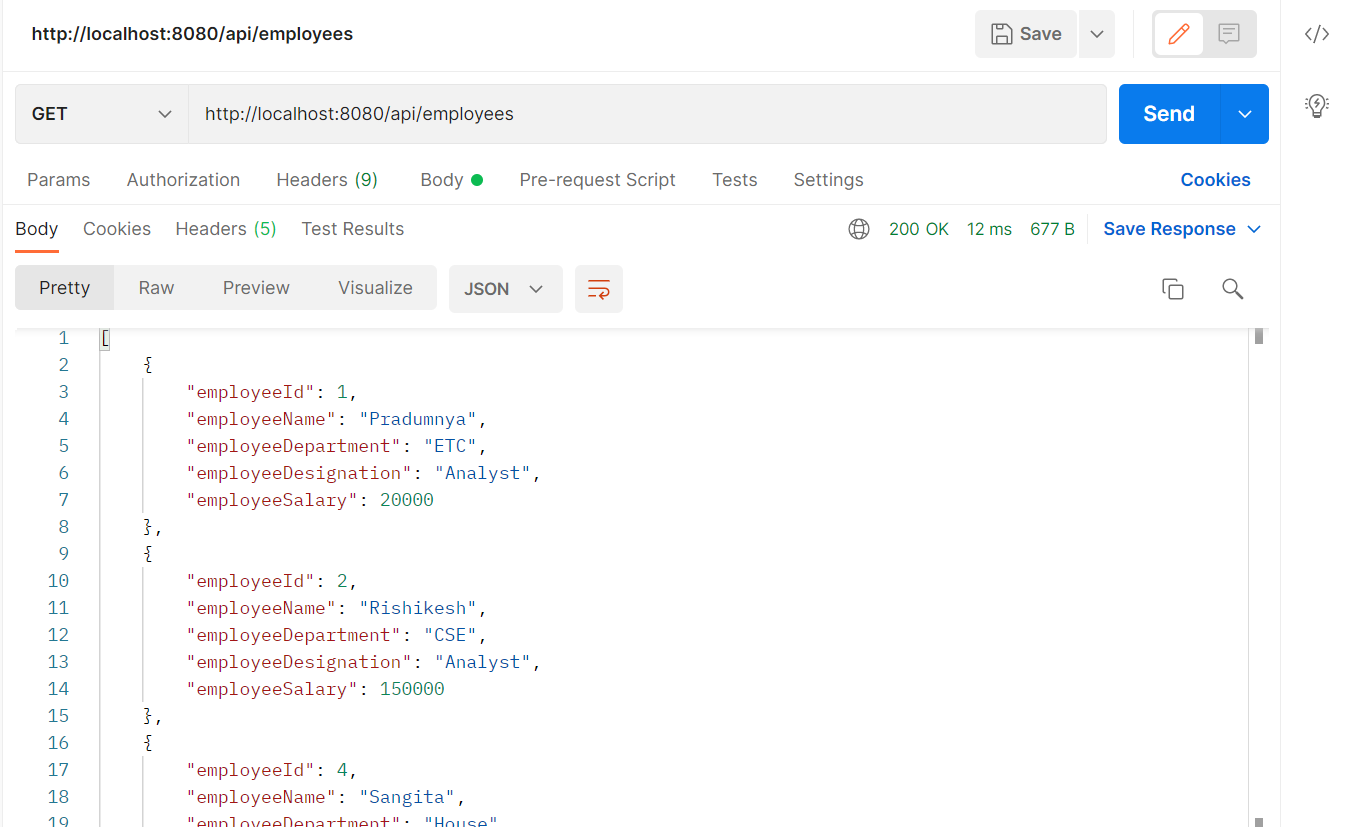


2. searching specific employee.

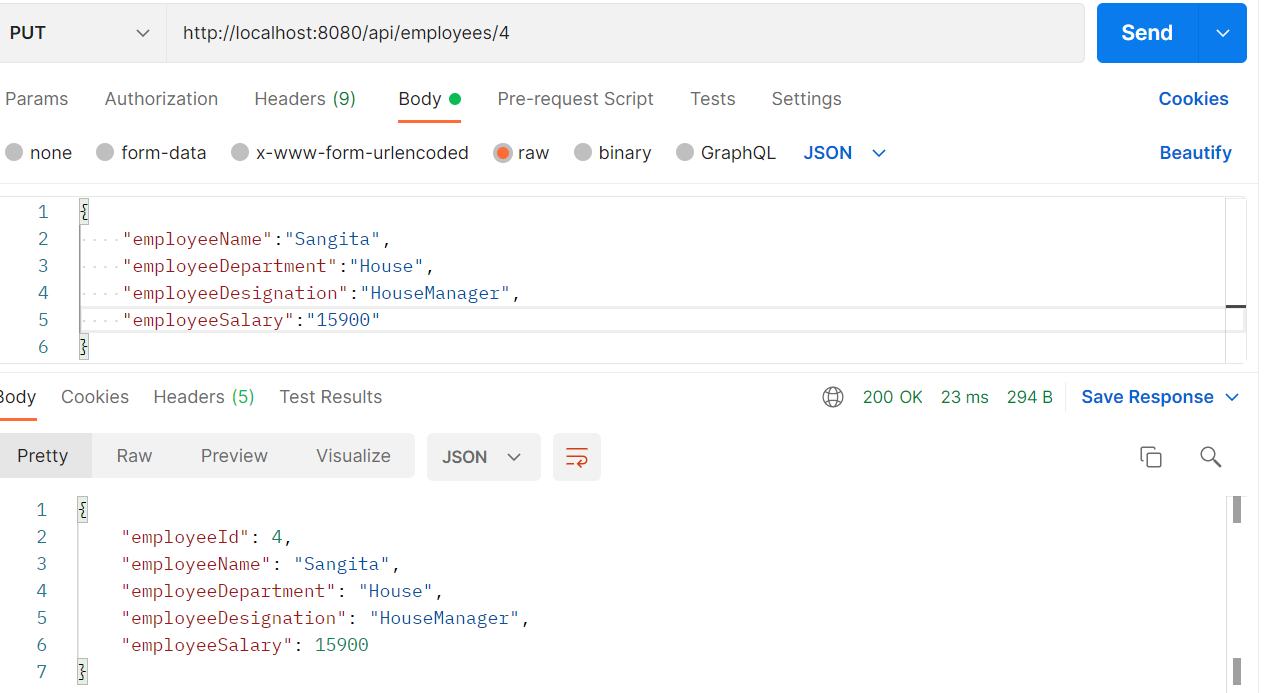
3. deleting specific employee



4.find all employees



5. update employee



6) Create a Calculator RESTful service that provides following functionality.

Calculator.java

**package** springboot.assignment.model;

**public** **class** Calculator {

**private** **int** num1;

**private** **int** num2;

**public** **int** getNum1() {

**return** num1;

}

**public** **void** setNum1(**int** num1) {

**this**.num1 = num1;

}

**public** **int** getNum2() {

**return** num2;

}

**public** **void** setNum2(**int** num2) {

**this**.num2 = num2;

}

}

CalculatorService.java

**package** springboot.assignment.service;

**public** **interface** CalculatorService {

**public** **int** add(**int** num1, **int** num2);

**public** **int** sub(**int** num1, **int** num2);

**public** **int** mul(**int** num1, **int** num2);

**public** **double** div(**int** num1, **int** num2);

**public** **double** sqrt(**int** num1);

}

CalculatorServiceImpl.java

**package** springboot.assignment.service.impl;

**import** org.springframework.stereotype.Service;

**import** springboot.assignment.service.CalculatorService;

@Service

**public** **class** CalculatorServiceImpl **implements** CalculatorService {

@Override

**public** **int** add(**int** num1, **int** num2) {

**return** num1 + num2;

}

@Override

**public** **int** sub(**int** num1, **int** num2) {

**return** num1 - num2;

}

@Override

**public** **int** mul(**int** num1, **int** num2) {

**return** num1 \* num2;

}

@Override

**public** **double** div(**int** num1, **int** num2) {

**double** num3 = 0;

**try** {

num3 = num1 / num2;

} **catch** (ArithmeticException e) {

e.printStackTrace();

}

**return** num3;

}

@Override

**public** **double** sqrt(**int** num1) {

**double** num3 = 0;

num3 = Math.*sqrt*(num1);

**return** num3;

}

}

CalculatorController.java

**package** springboot.assignment;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** springboot.assignment.service.CalculatorService;

@RestController

@RequestMapping("/calculate")

**public** **class** CalculatorController {

@Autowired

CalculatorService calculatorService;

**public** CalculatorController(CalculatorService calculatorService) {

**super**();

**this**.calculatorService = calculatorService;

}

@GetMapping("/add/{num1}/{num2}")

**public** **int** add(@PathVariable("num1") **int** num1, @PathVariable("num2") **int** num2) {

**return** calculatorService.add(num1, num2);

}

@GetMapping("/sub/{num1}/{num2}")

**public** **int** sub(@PathVariable("num1") **int** num1, @PathVariable("num2") **int** num2) {

**return** calculatorService.sub(num1, num2);

}

@GetMapping("/mul/{num1}/{num2}")

**public** **int** mul(@PathVariable("num1") **int** num1, @PathVariable("num2") **int** num2) {

**return** calculatorService.mul(num1, num2);

}

@GetMapping("/div/{num1}/{num2}")

**public** **double** div(@PathVariable("num1") **int** num1, @PathVariable("num2") **int** num2) {

**return** calculatorService.div(num1, num2);

}

@GetMapping("/sqrt/{num1}")

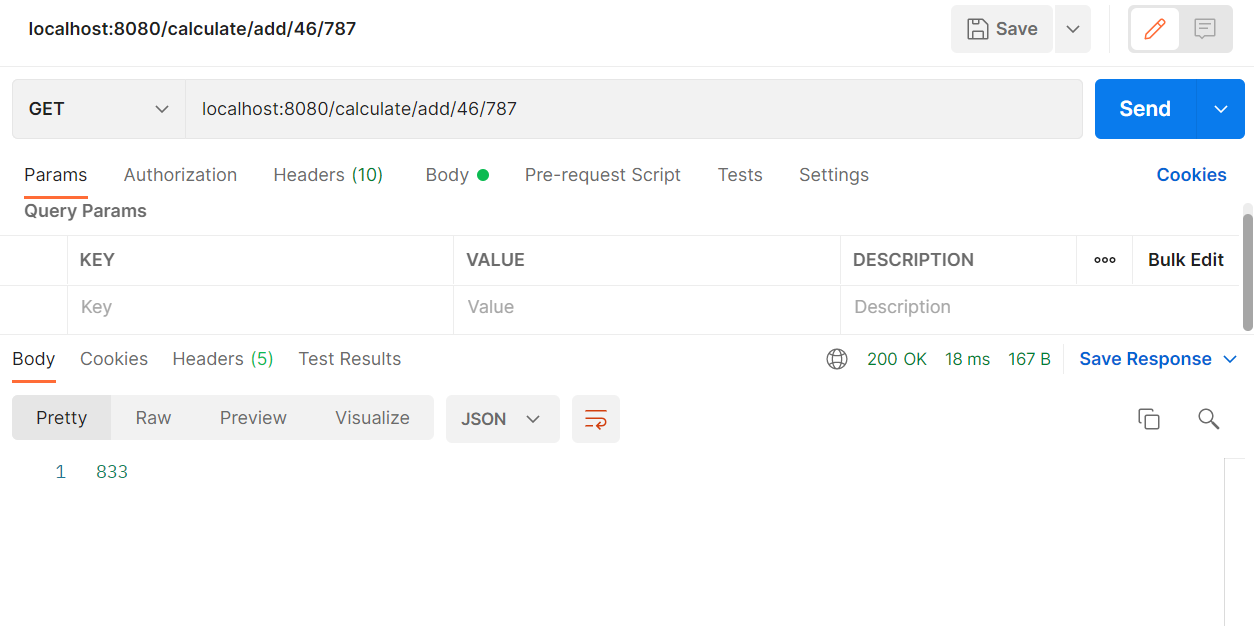
**public** **double** sqrt(@PathVariable("num1") **int** num1) {

**return** calculatorService.sqrt(num1);

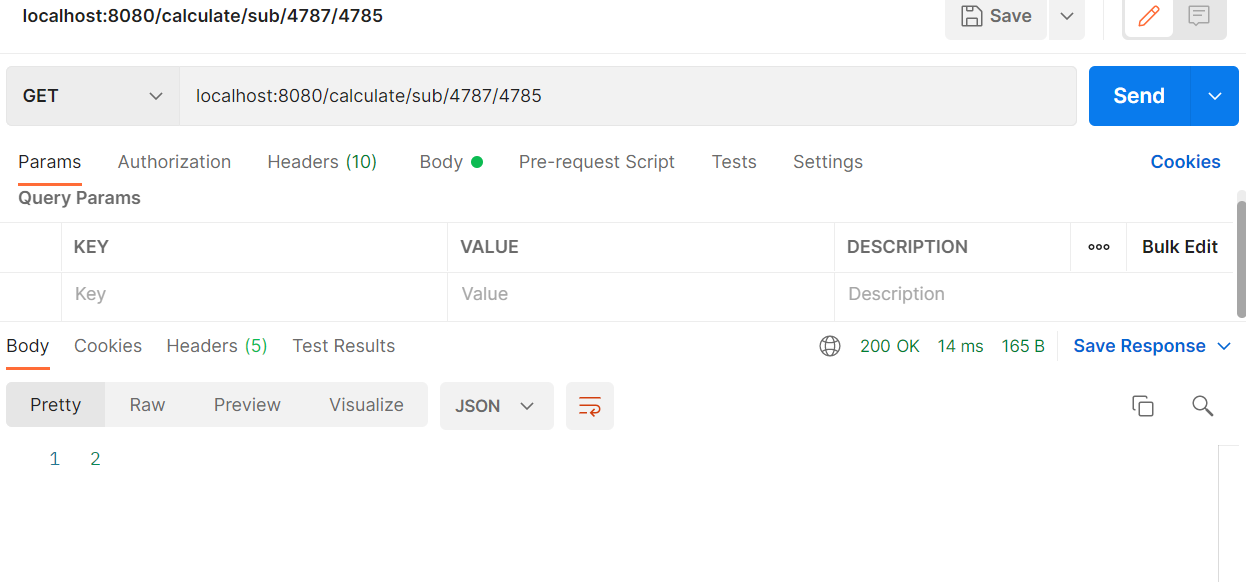
}

}

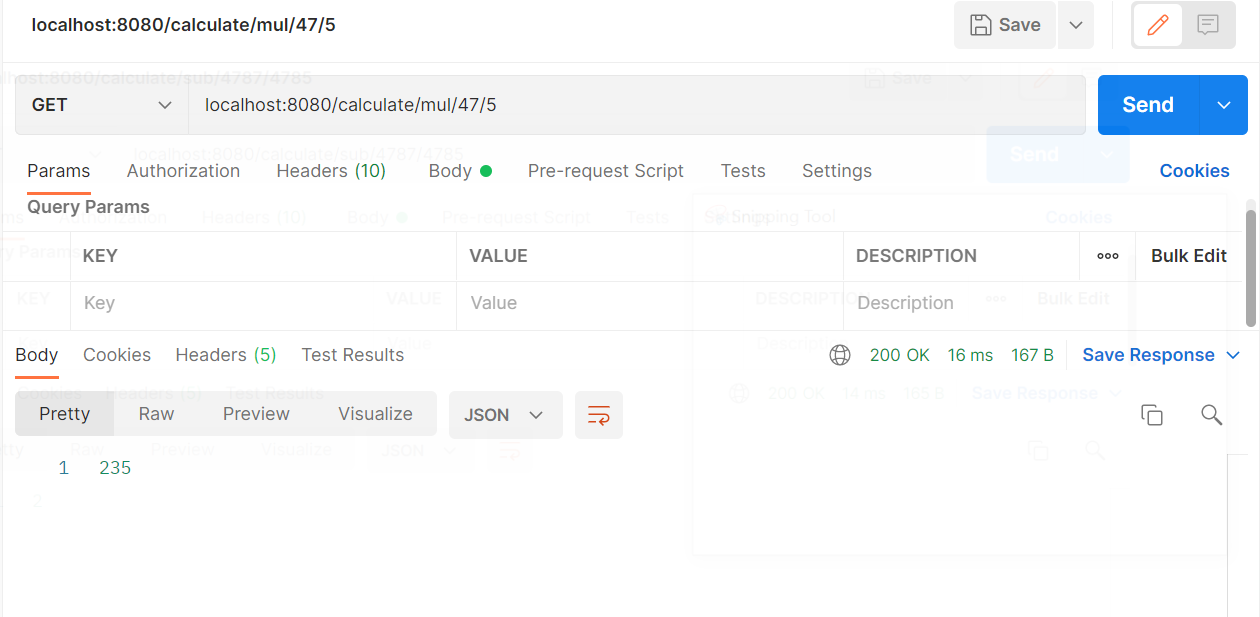
1. Addition of the 2 numbers



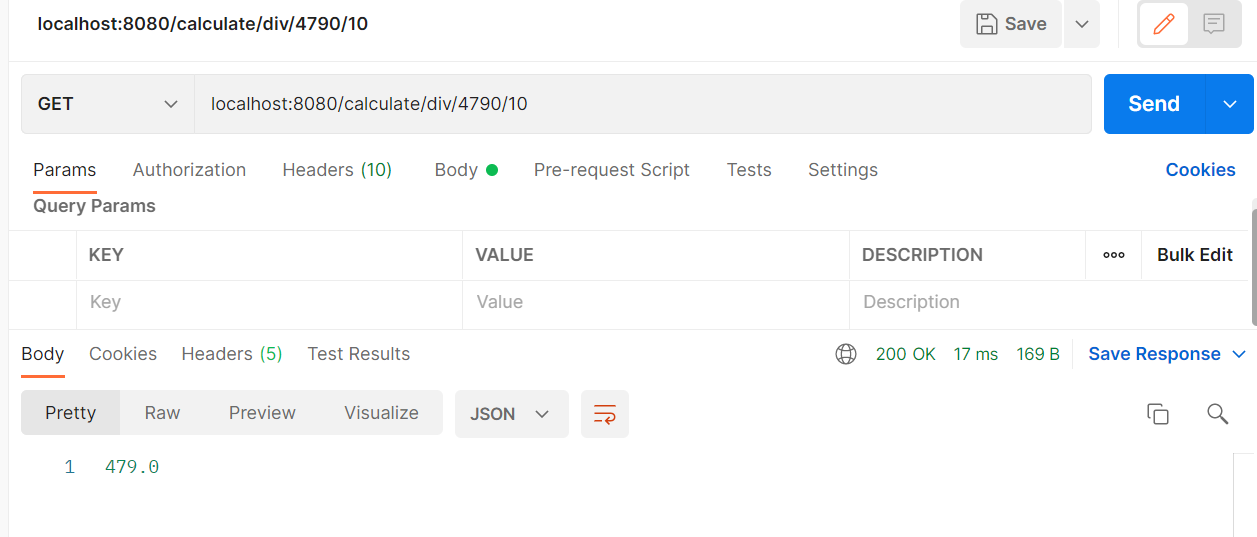
2. Subtraction of the 2 numbers



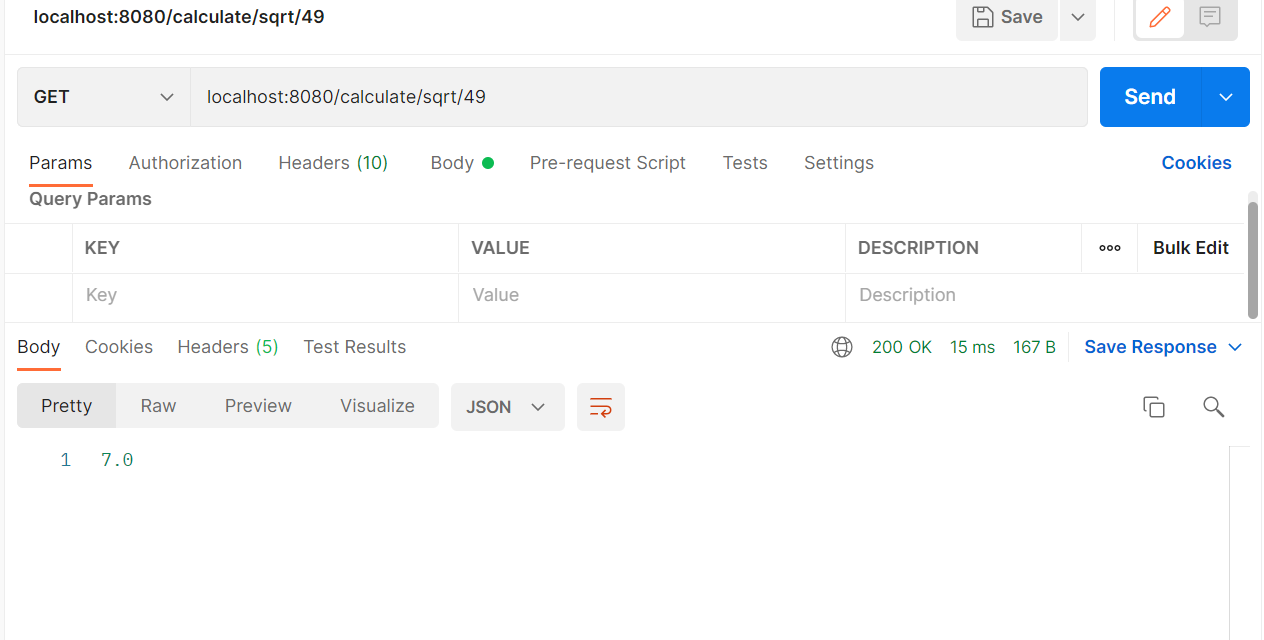
3. Multiplication of 2 numbers



4. Division of 2 numbers



5. Finding square root of a number.



Consume the above RESTful web service by using RestTemplate.

7) Design and develop RESTful web service as follows:

Orders.java

**package** mongodbexample.model;

**import** org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "order")

**public** **class** Orders {

**private** **int** id;

**private** String name;

**private** **int** price;

**private** String status;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** Orders(**int** id, String name, **int** price, String status) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.price = price;

**this**.status = status;

}

}

OrdersRepository.java

package mongodbexample.repo;

import org.springframework.data.mongodb.repository.MongoRepository;

import mongodbexample.model.Orders;

public interface OrderRepository extends MongoRepository<Orders, Integer>{

}

OrderService.java

package mongodbexample.service;

import java.util.List;

import mongodbexample.model.Orders;

public interface OrderService {

Orders saveOrders(Orders orders);

List<Orders> getAllOrders();

Orders getOrderById(int id);

Orders updateOrder(Orders orders, int id);

void deleteOrder(int id);

}

OrderServiceImpl.java

**package** mongodbexample.service.impl;

**import** java.util.List;

**import** org.springframework.stereotype.Service;

**import** mongodbexample.exception.ResourceNotFoundException;

**import** mongodbexample.model.Orders;

**import** mongodbexample.repo.OrderRepository;

**import** mongodbexample.service.OrderService;

@Service

**public** **class** OrderServiceImpl **implements** OrderService {

**private** OrderRepository orderRepository;

**public** OrderServiceImpl(OrderRepository orderRepository) {

**super**();

**this**.orderRepository = orderRepository;

}

@Override

**public** Orders saveOrders(Orders orders) {

**return** orderRepository.save(orders);

}

@Override

**public** List<Orders> getAllOrders() {

**return** orderRepository.findAll();

}

@Override

**public** Orders getOrderById(**int** id) {

**return** orderRepository.findById(id).orElseThrow(() -> **new** ResourceNotFoundException("Orders", "id", id));

}

@Override

**public** Orders updateOrder(Orders orders, **int** id) {

Orders existingOrders = orderRepository.findById(id)

.orElseThrow(() -> **new** ResourceNotFoundException("Orders", "id", id));

existingOrders.setId(orders.getId());

existingOrders.setName(orders.getName());

existingOrders.setPrice(orders.getPrice());

existingOrders.setStatus(orders.getStatus());

orderRepository.save(existingOrders);

**return** existingOrders;

}

@Override

**public** **void** deleteOrder(**int** id) {

orderRepository.findById(id).orElseThrow(() -> **new** ResourceNotFoundException("Orders", "id", id));

orderRepository.deleteById(id);

}

}

OrderController.java

**package** mongodbexample.controller;

**import** java.util.List;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** mongodbexample.model.Orders;

**import** mongodbexample.service.OrderService;

@RestController

@RequestMapping("/orders")

**public** **class** OrderController {

**private** OrderService orderService;

**public** OrderController(OrderService orderService) {

**super**();

**this**.orderService = orderService;

}

@PostMapping

**public** ResponseEntity<Orders> addOrders(@RequestBody Orders orders){

**return** **new** ResponseEntity<Orders>(orderService.saveOrders(orders),HttpStatus.***CREATED***);

}

@GetMapping

**public** List<Orders> getAllOrders(){

**return** orderService.getAllOrders();

}

@GetMapping("{id}")

**public** ResponseEntity<Orders> getOrderById(@PathVariable("id") **int** id) {

**return** **new** ResponseEntity<Orders>(orderService.getOrderById(id),HttpStatus.***OK***);

}

@PutMapping("{id}")

**public** ResponseEntity<Orders> updateOrders(@PathVariable("id") **int** id, @RequestBody Orders orders) {

**return** **new** ResponseEntity<Orders>(orderService.updateOrder(orders, id), HttpStatus.***OK***);

}

// build delete employee REST API

@DeleteMapping("{id}")

**public** ResponseEntity<String> deleteOrders(@PathVariable("id") **int** id) {

orderService.deleteOrder(id);

**return** **new** ResponseEntity<String>("Order cancelled successfully",HttpStatus.***OK***);

}

}

ResourceNotFoundException.java

package mongodbexample.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT\_FOUND)

public class ResourceNotFoundException extends RuntimeException {

private static final long serialVersionUID = 1L;

public String resourceName;

public String fieldName;

public Object fieldValue;

public ResourceNotFoundException(String resourceName, String fieldName, Object fieldValue) {

super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue));

this.resourceName = resourceName;

this.fieldName = fieldName;

this.fieldValue = fieldValue;

}

public String getResourceName() {

return resourceName;

}

public void setResourceName(String resourceName) {

this.resourceName = resourceName;

}

public String getFieldName() {

return fieldName;

}

public void setFieldName(String fieldName) {

this.fieldName = fieldName;

}

public Object getFieldValue() {

return fieldValue;

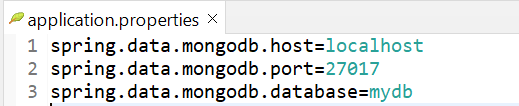
}

public void setFieldValue(Object fieldValue) {

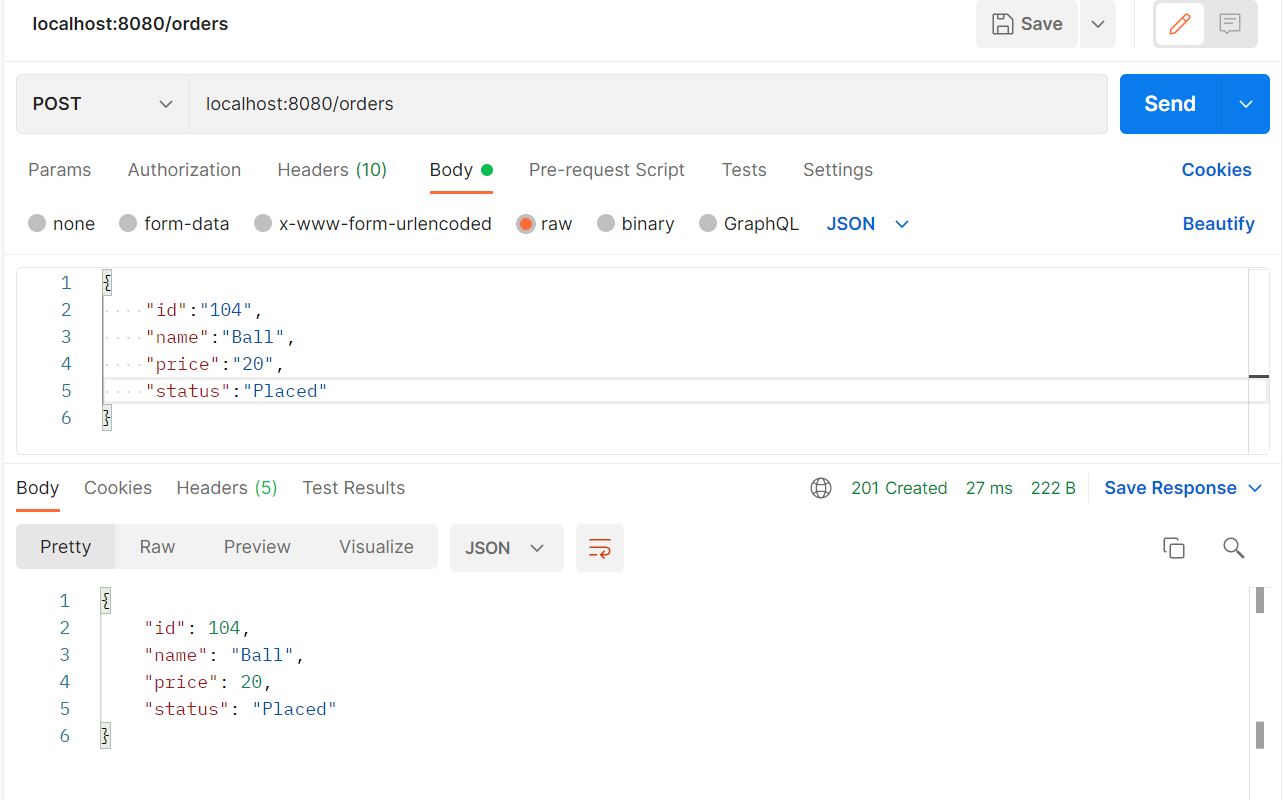
this.fieldValue = fieldValue;

}

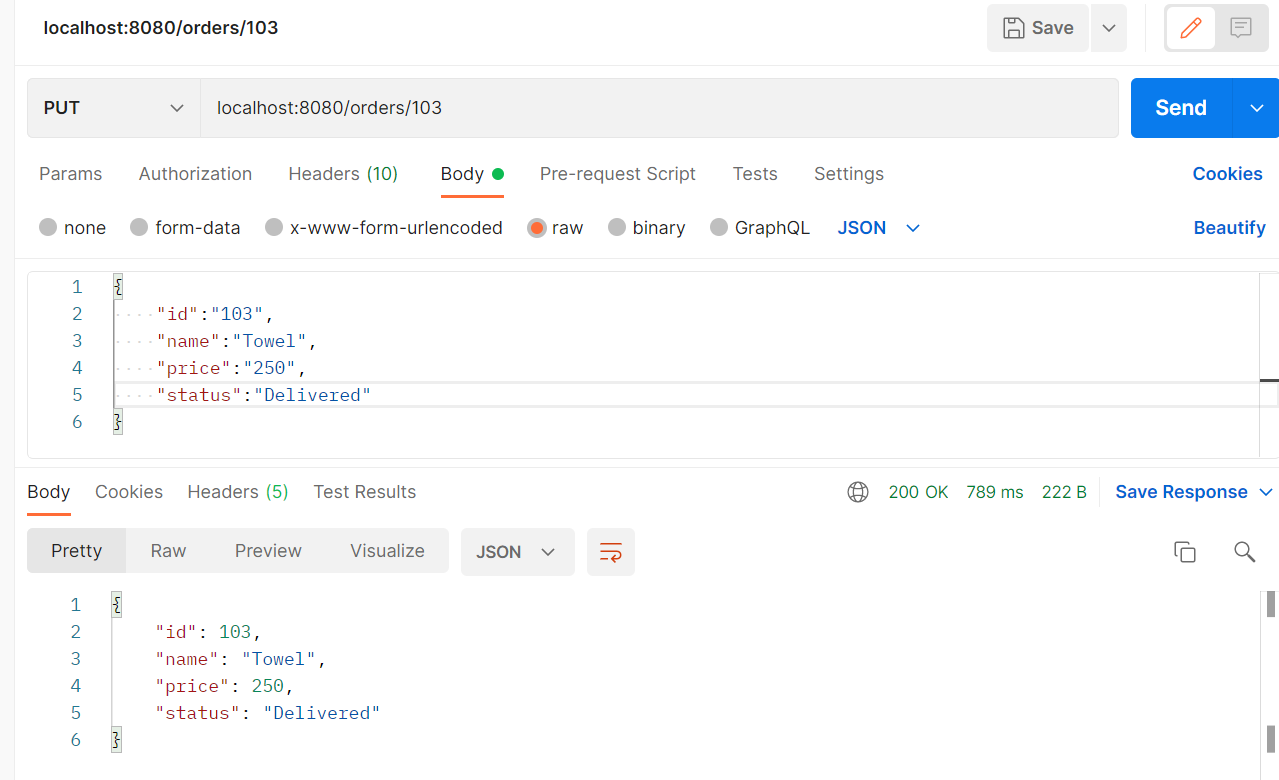
}



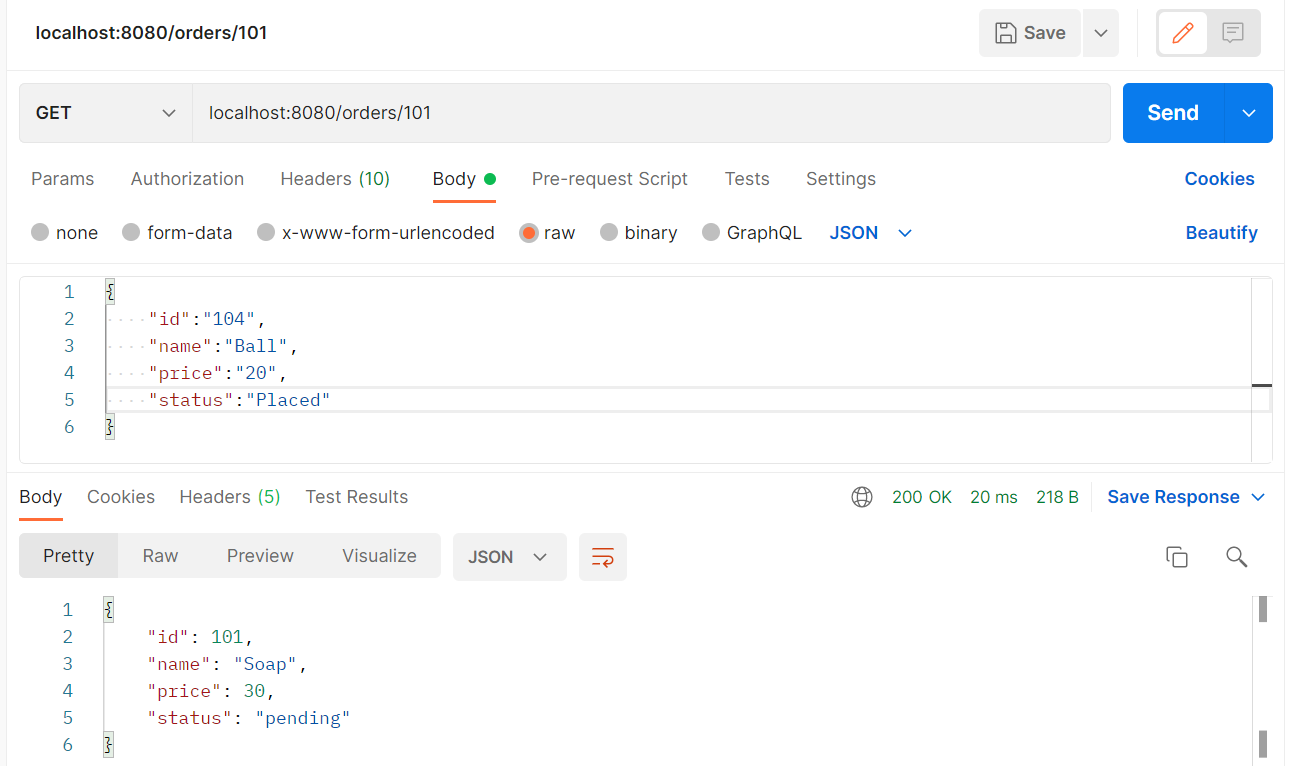
1. A user can place an order



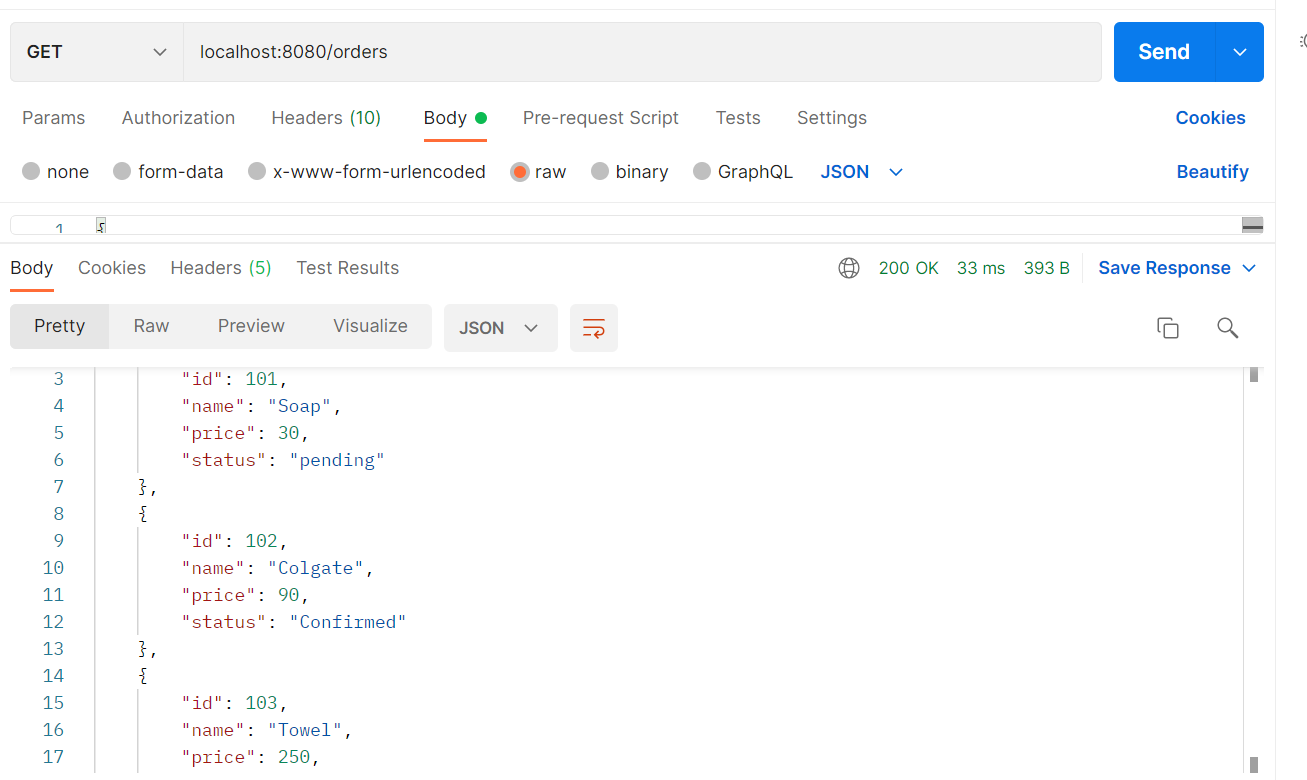
2. A user can update an order



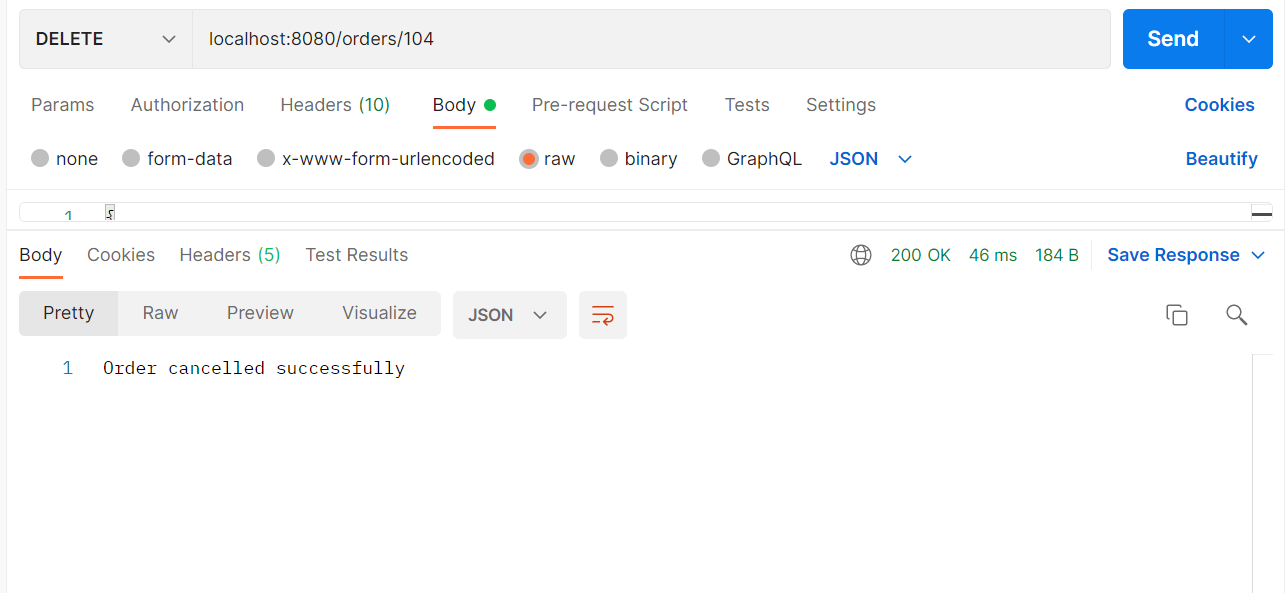
3. A user can view specific order



4. A user can view all the orders



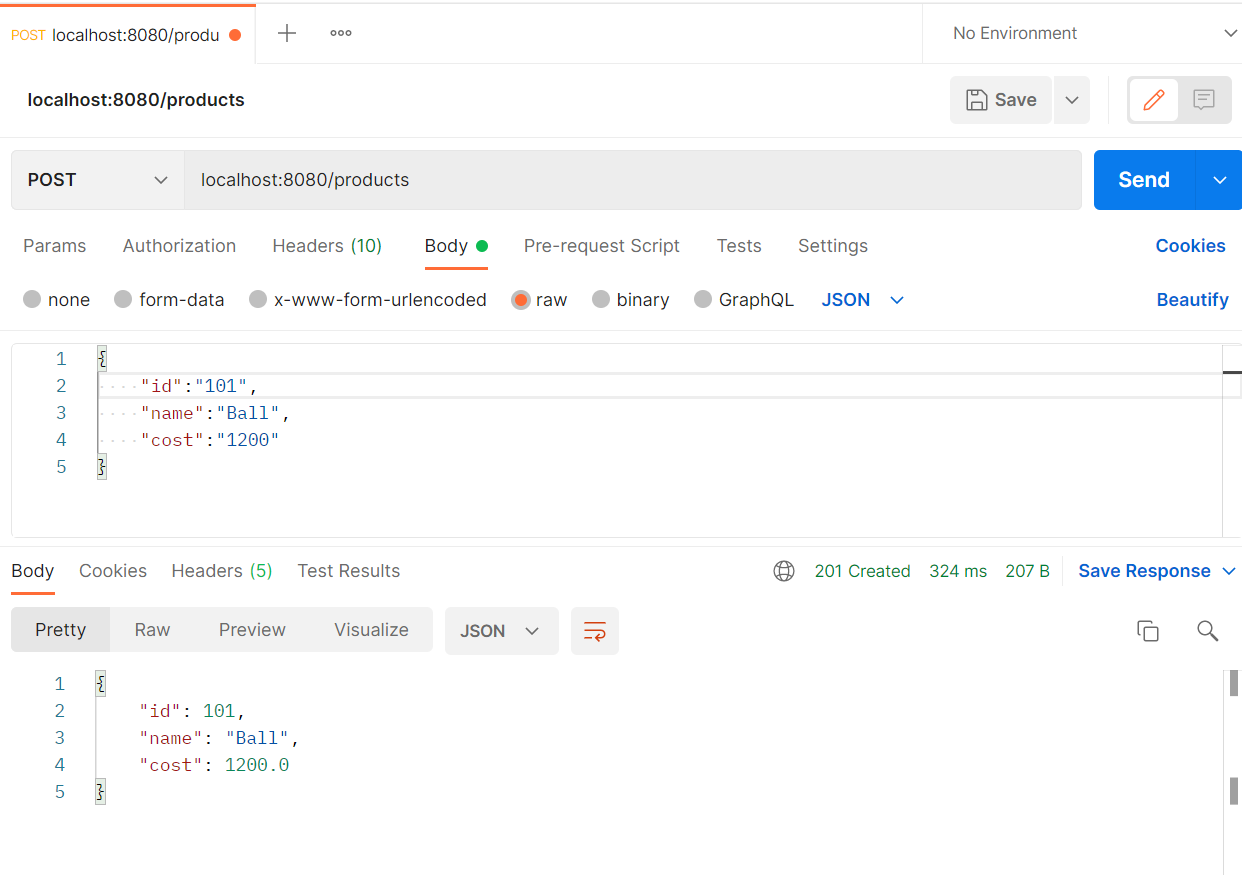
5. A user can delete a specific order.



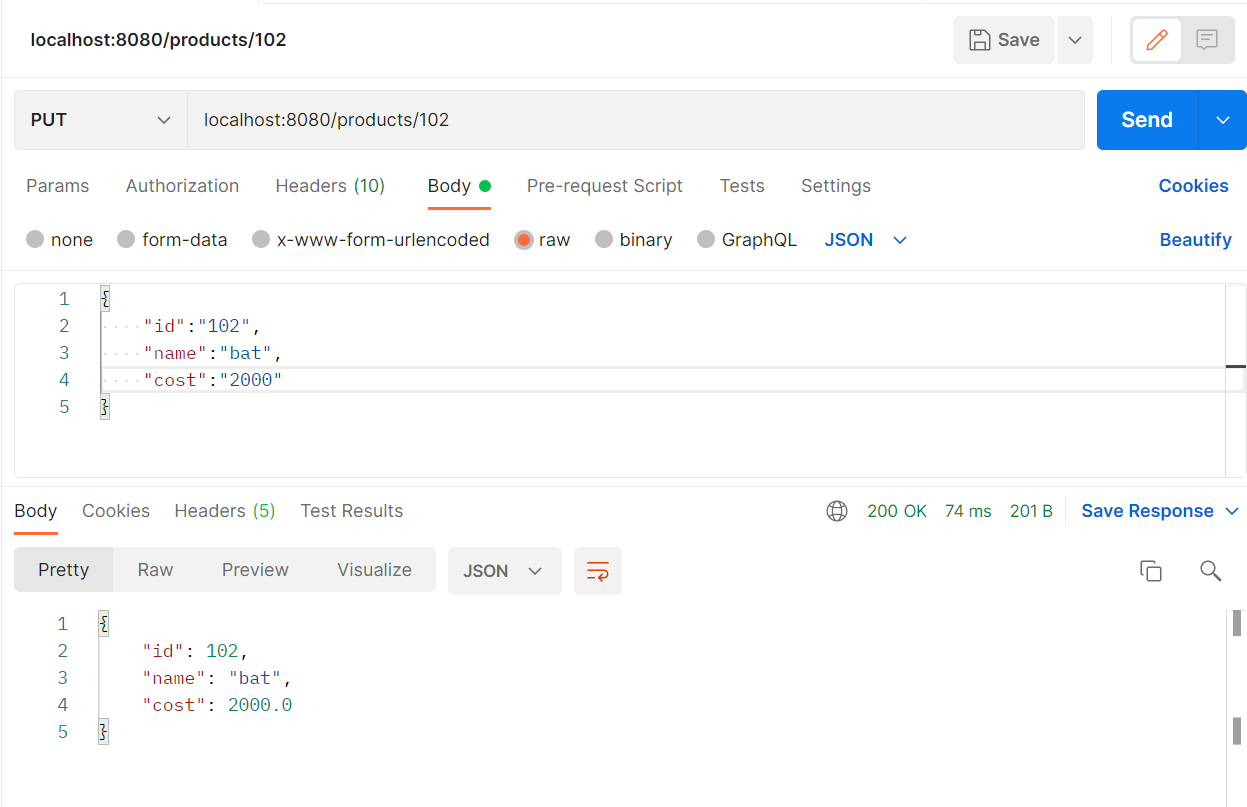
Note: Use MongoRepository of Spring data to store order details

8) Design and develop RESTful web service as follows:

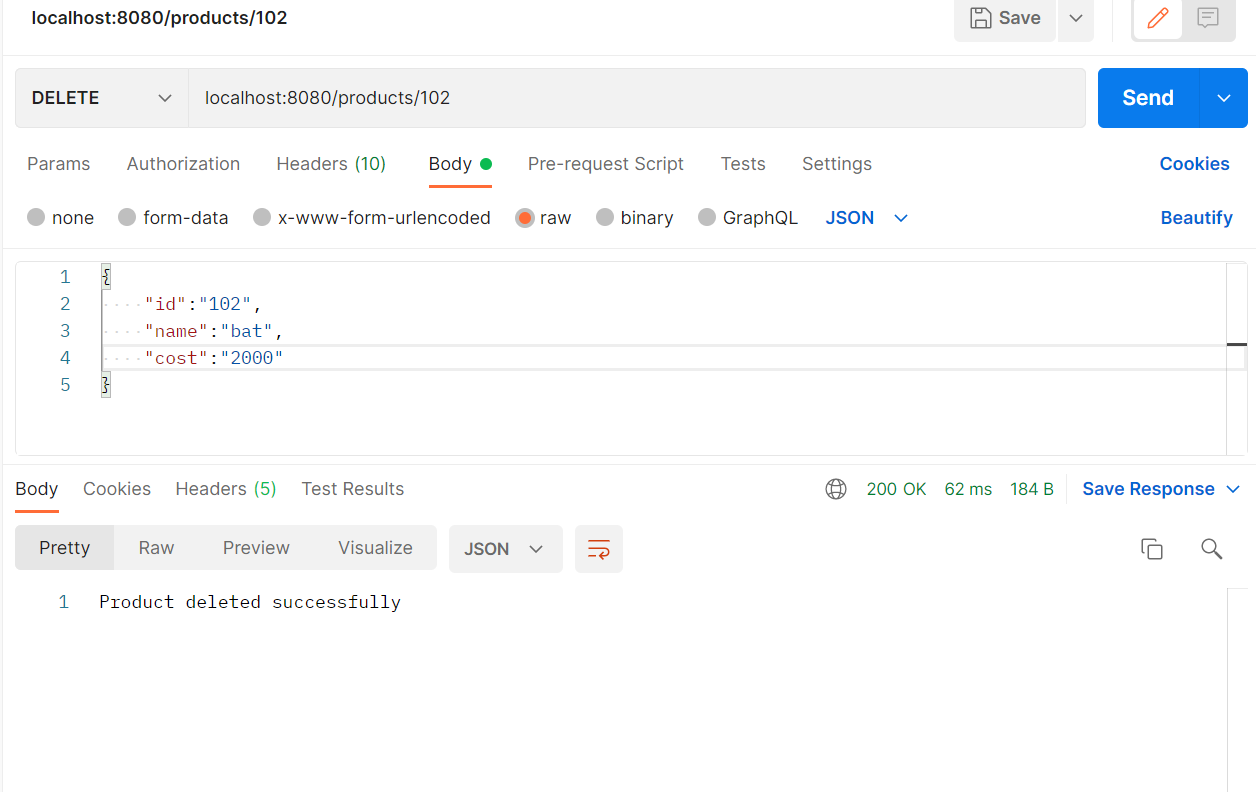
1. An admin can add a new product.



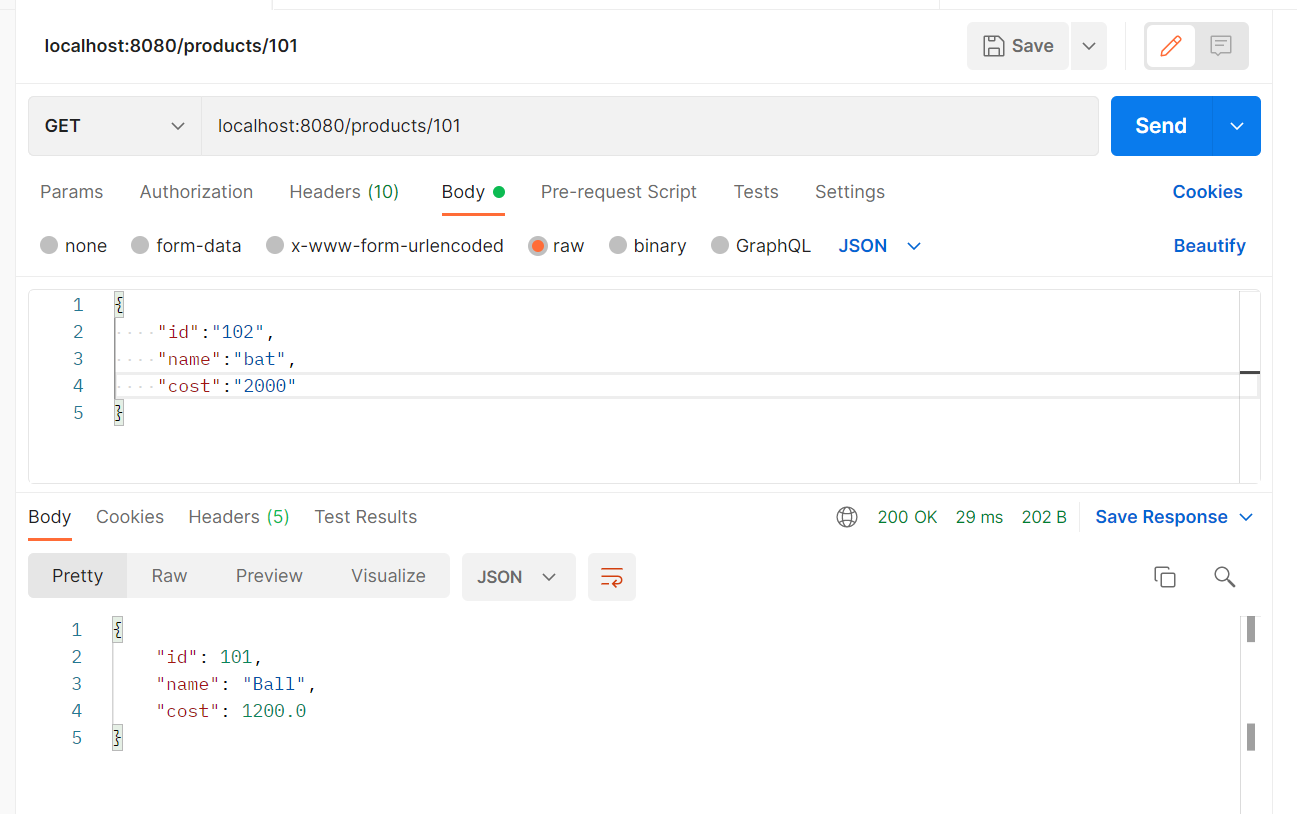
2. An admin can update details of existing product.



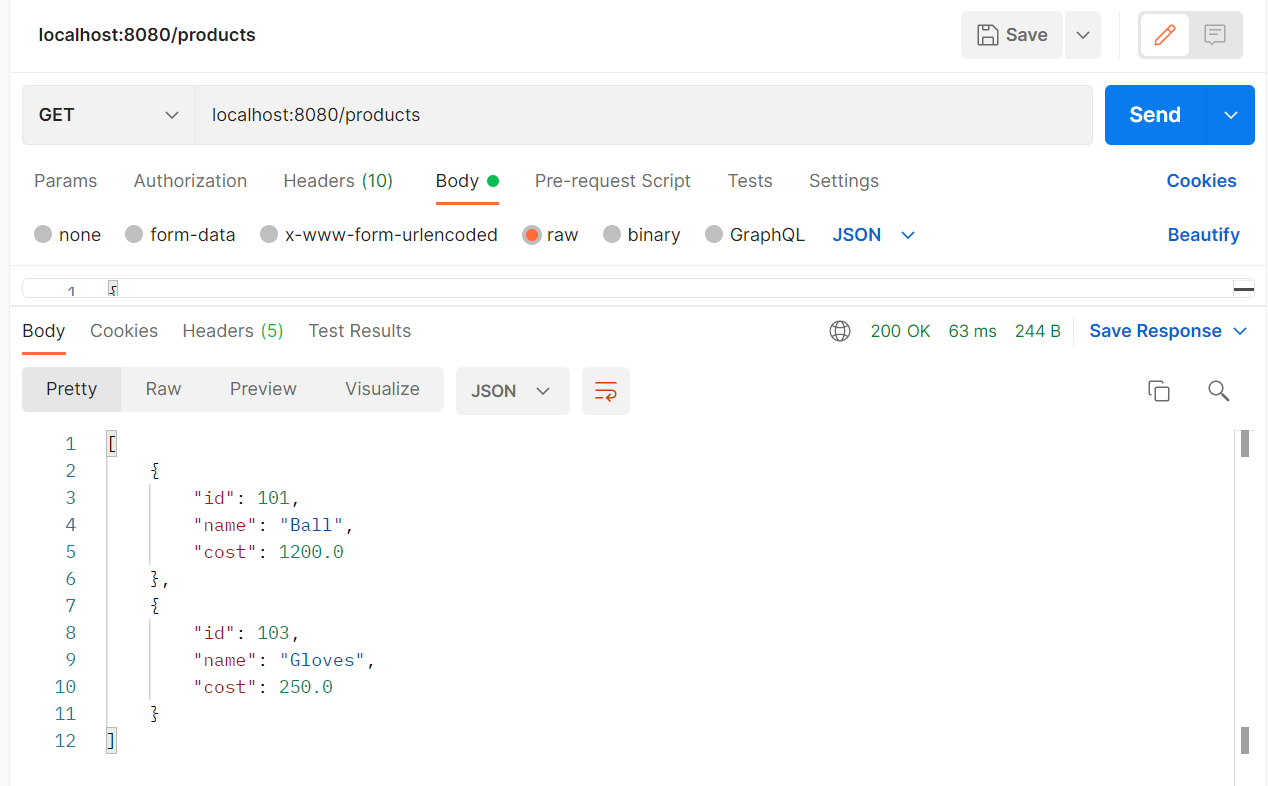
3. An admin can delete existing product



4. An admin can view specific product



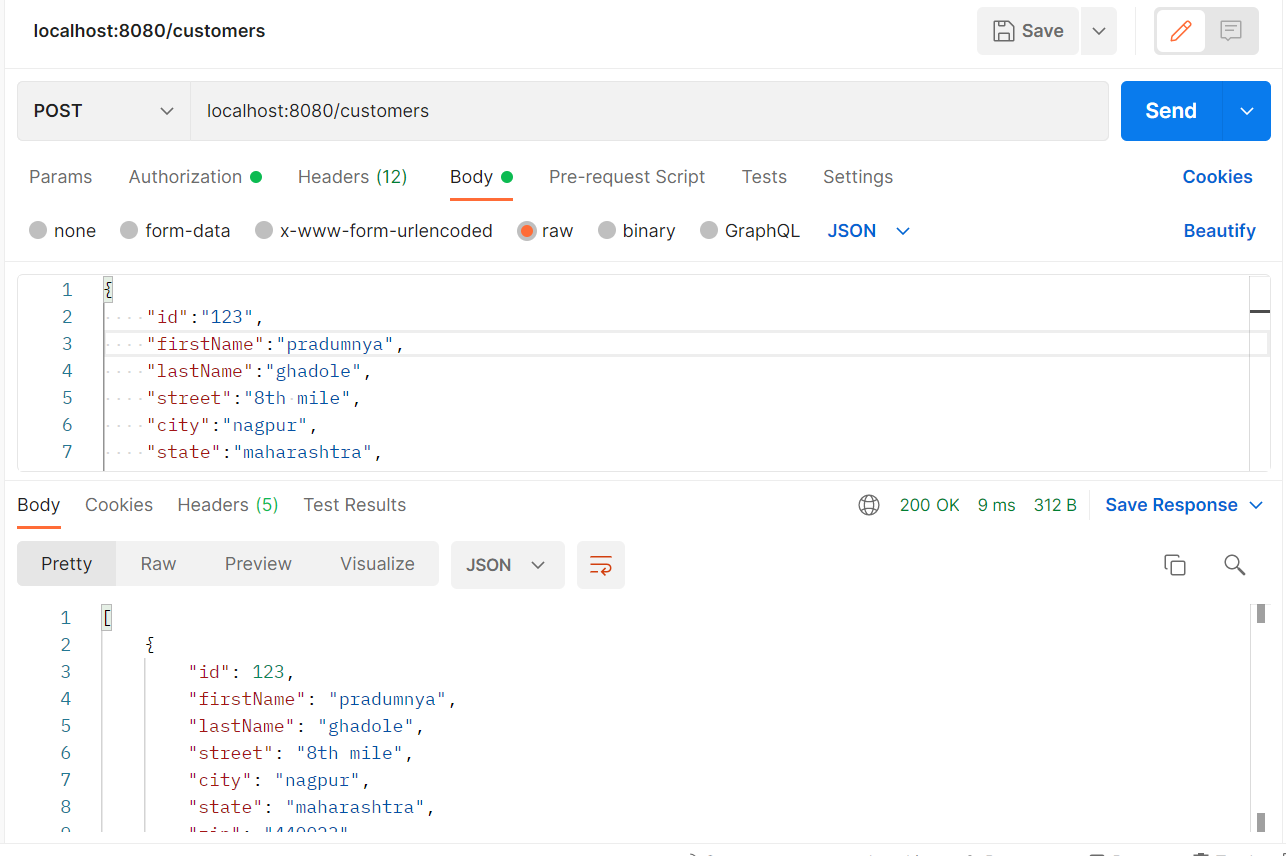
5. An admin can view all the products



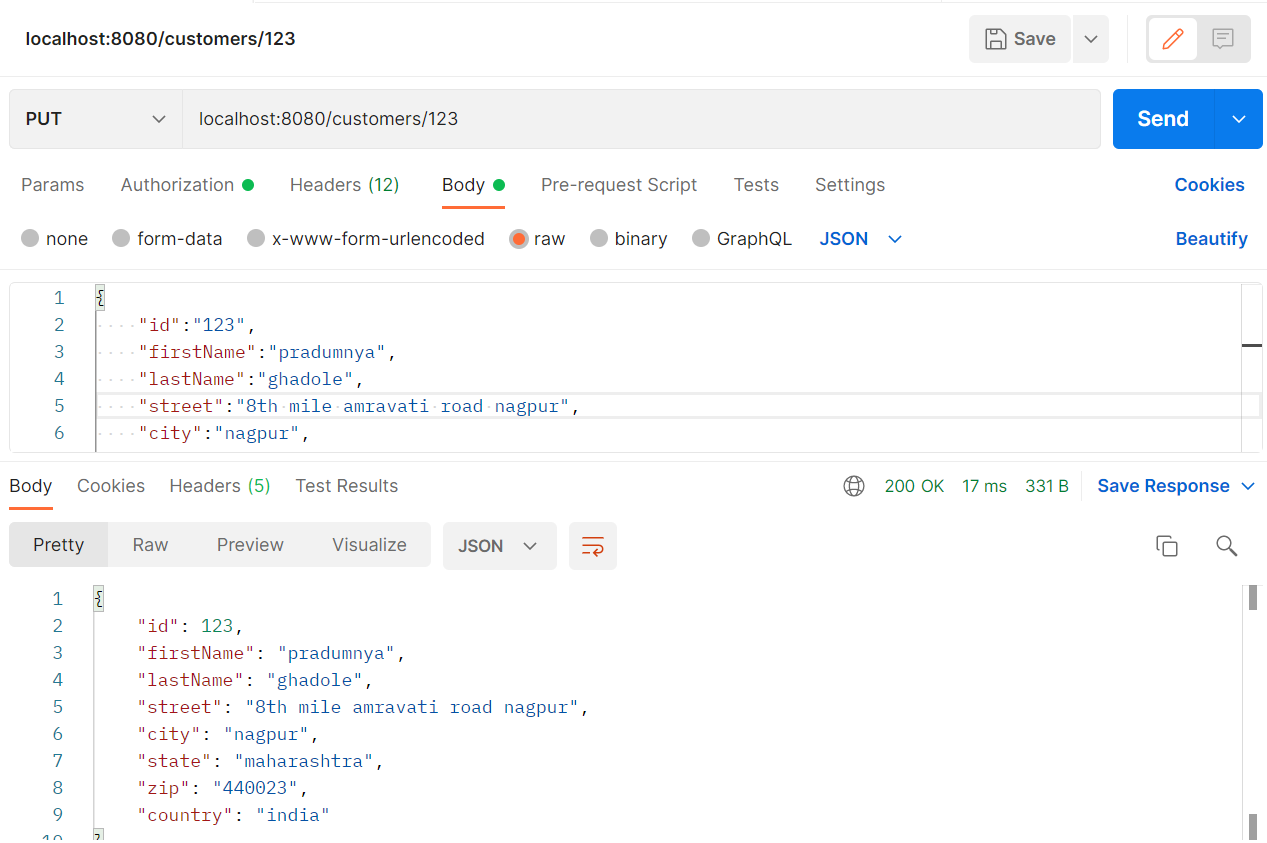
Note:Use MongoRepository of Spring data to store product details

9) Design and develop RESTful web service as follows:

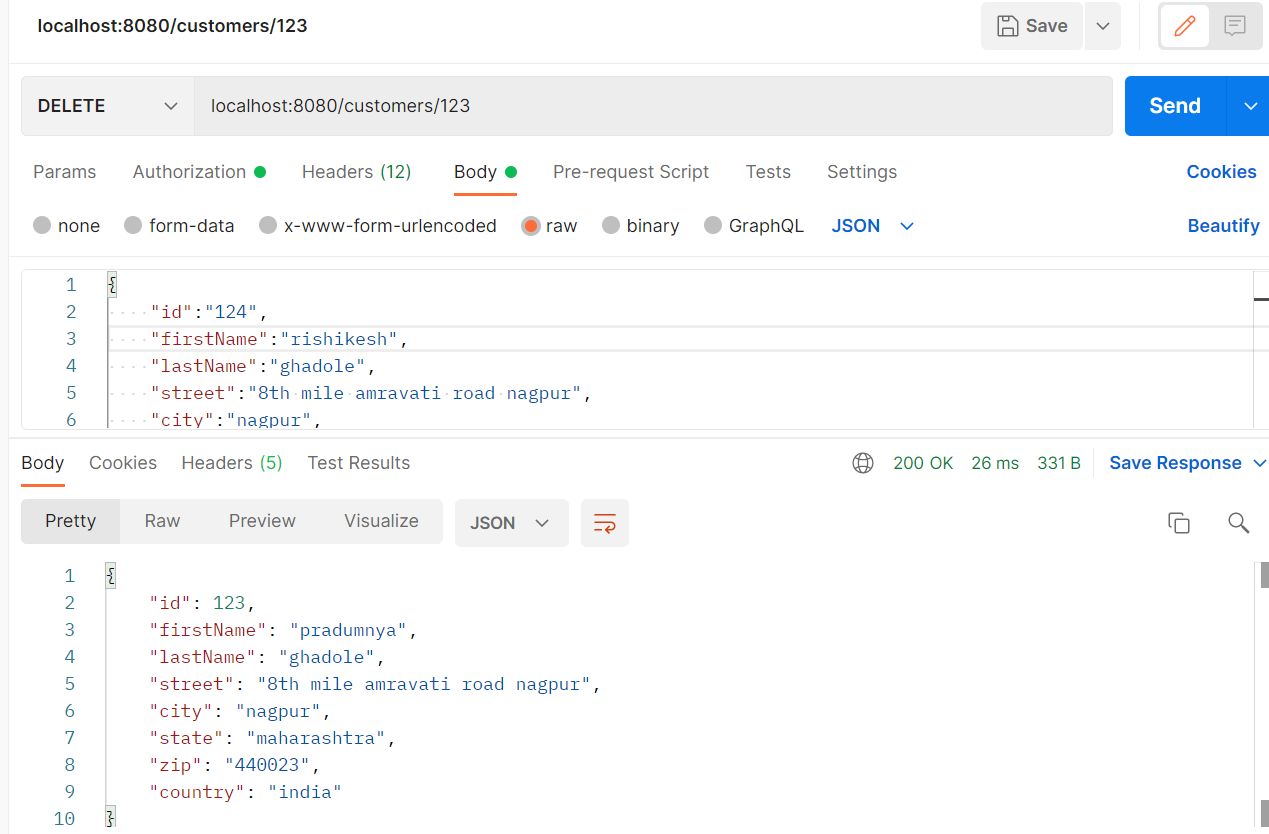
1. Add a new customer information.



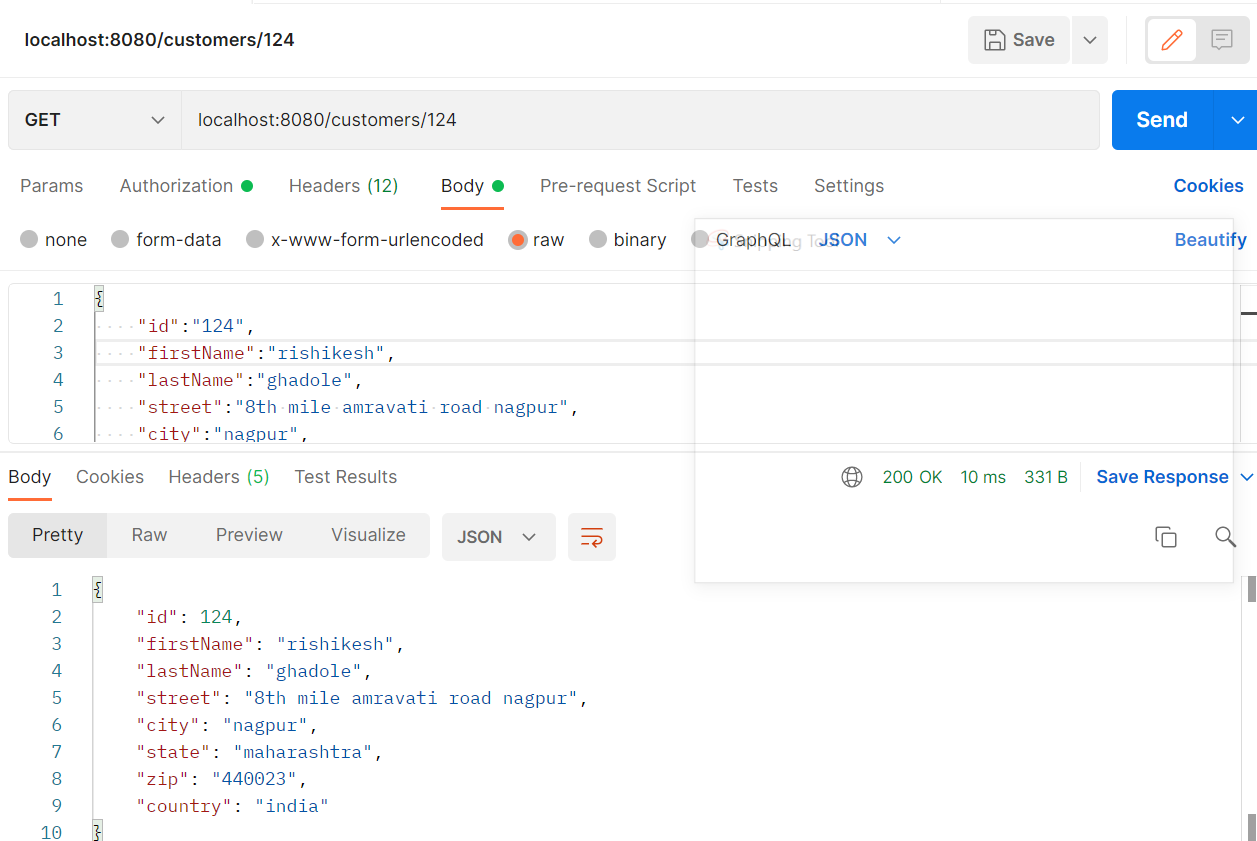
2. Update customer information.



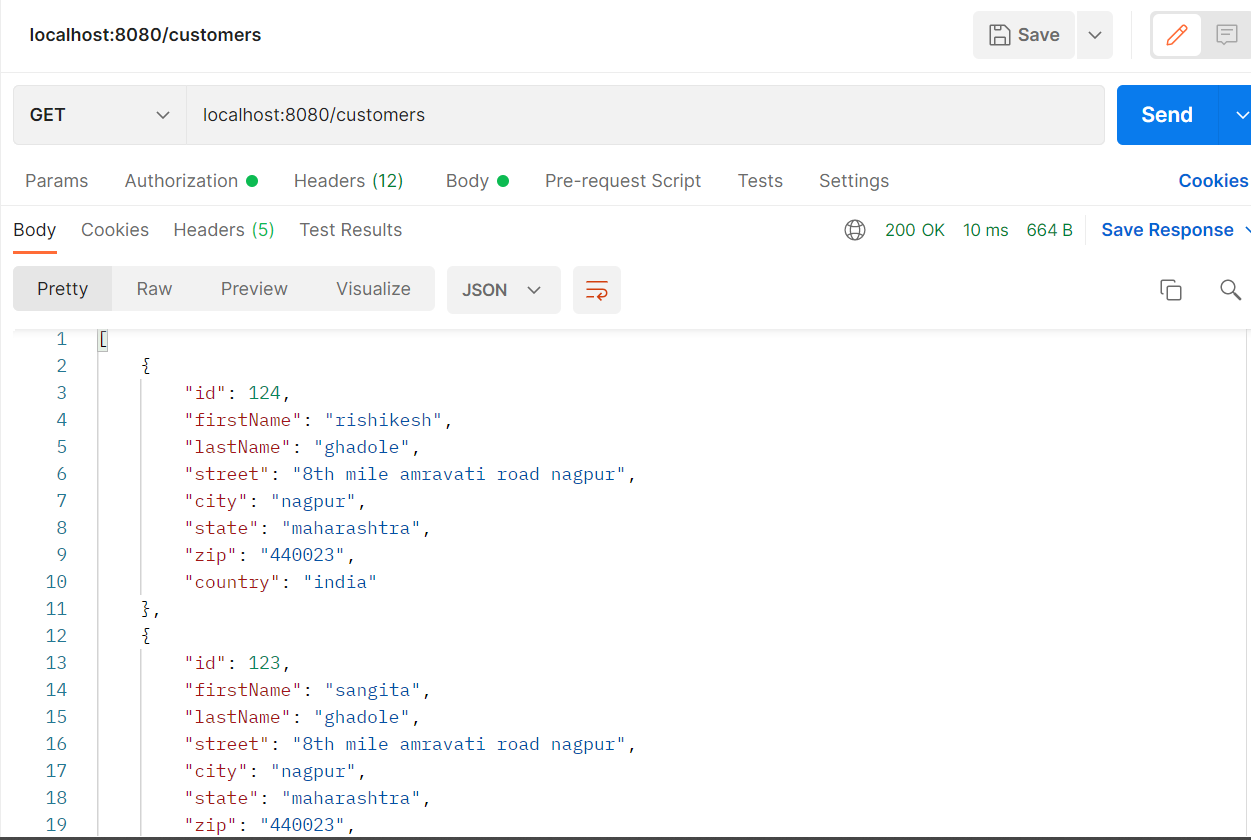
3. Delete existing customer information.



4. Fetch information of specific customer.



5. Fetch information of all customers.



Note: Use CrudRepository of Spring Data to store customer details

10) Test all the above RESTful web services by using MockMvc.