8. Create a project to demonstrate microservices with Spring Boot=

BuildingRestApiWebserviceDemoApplication.java

```
package com.ecommerce;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;


@ComponentScan({"com.ecommerce.controllers","com.ecommerce.entity",
"com.ecommerce.repositries" })

@EnableJpaRepositories

@SpringBootApplication

public class BuildingRestApiWebserviceDemoApplication {


	public static void main(String[] args) {

		SpringApplication.run(BuildingRestApiWebserviceDemoApplication.class, args);

	}


}
```

MainRestController.java

```
package com.ecommerce.controllers;


import java.math.BigDecimal;


import org.springframework.http.HttpStatusCode;

import org.springframework.http.ResponseEntity;

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

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

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

```
import com.ecommerce.entity.EProduct;

@RestController
@RequestMapping("/main")
public class MainRestController {


	// Starter Helloworld Example
	@GetMapping(path="/data", produces = "application/json")
	public ResponseEntity<EProduct> displayData()
	{
		EProduct e1= new EProduct();
		e1.setName("Hockey Pads");
		e1.setPrice(new BigDecimal(1000.5));

		return new ResponseEntity<EProduct>(e1, HttpStatusCode.valueOf(200));

	}
	}
```

ProductRestController.java

```
package com.ecommerce.controllers;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatusCode;
import org.springframework.http.ResponseEntity;
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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ecommerce.entity.EProduct;
import com.ecommerce.repositries.EProductRepo;

@RestController
@RequestMapping("/products")
public class ProductRestController {
	@Autowired
	EProductRepo eProductRepo;

	// List all the products
	@GetMapping(path="/list", produces = "application/json")
	public ResponseEntity<List<EProduct>> listproducts()
	{
		List<EProduct> products = eProductRepo.findAll();

		return new ResponseEntity<List<EProduct>>(products,
HttpStatusCode.valueOf(200));
	}

	// Add a new product
	@PostMapping(path="/add", consumes = "application/json", produces = "application/json")
	public ResponseEntity<EProduct> addProduct(@RequestBody EProduct eProduct){
		eProduct = eProductRepo.save(eProduct);

		return new ResponseEntity<EProduct>(eProduct, HttpStatusCode.valueOf(200));

	}
```

```
	// Show details of one product
	@GetMapping(path="/details/{id}", produces = "application/json")
	public ResponseEntity<Object> showProduct(@PathVariable("id") int id){
		Optional<EProduct> productFromRepo = eProductRepo.findById(id);

		if (productFromRepo.isPresent()) {
			EProduct product = productFromRepo.get();
			return new ResponseEntity<>(product, HttpStatusCode.valueOf(200));
		}else {
			return new ResponseEntity<>("Product Not Found",
HttpStatusCode.valueOf(404));
		}
	}

	// Delete a product
	@GetMapping(path="/delete/{id}", produces = "application/json")
	public ResponseEntity<Object> deleteProduct(@PathVariable("id") int id){
		Optional<EProduct> productFromRepo = eProductRepo.findById(id);

		if (productFromRepo.isPresent()) {
			eProductRepo.deleteById(id);
			return new ResponseEntity<>("Product "+ id + " Deleted",
HttpStatusCode.valueOf(200));
		}else {
			return new ResponseEntity<>("Product "+ id + " Not Found",
HttpStatusCode.valueOf(404));
		}
	}

}
```

EProduct.java

```
package com.ecommerce.entity;

import java.math.BigDecimal;
import java.util.Date;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;



@Entity
@Table(name="eproduct")
public class EProduct {

		@Id
		@Column(name="id")
		@GeneratedValue(strategy = GenerationType.IDENTITY)
	private long ID;

		// @Column(name="name")
	private String name;

	private BigDecimal price;

	@Column(name="date_added")
```

```
    private Date dateAdded;

    public EProduct() {

    }

    public long getID() {return this.ID; }
    public String getName() { return this.name;}
    public BigDecimal getPrice() { return this.price;}
    public Date getDateAdded() { return this.dateAdded;}

    public void setID(long id) { this.ID = id;}
    public void setName(String name) { this.name = name;}
    public void setPrice(BigDecimal price) { this.price = price;}
    public void setDateAdded(Date date) { this.dateAdded = date;}

}
```

EProductRepo.java

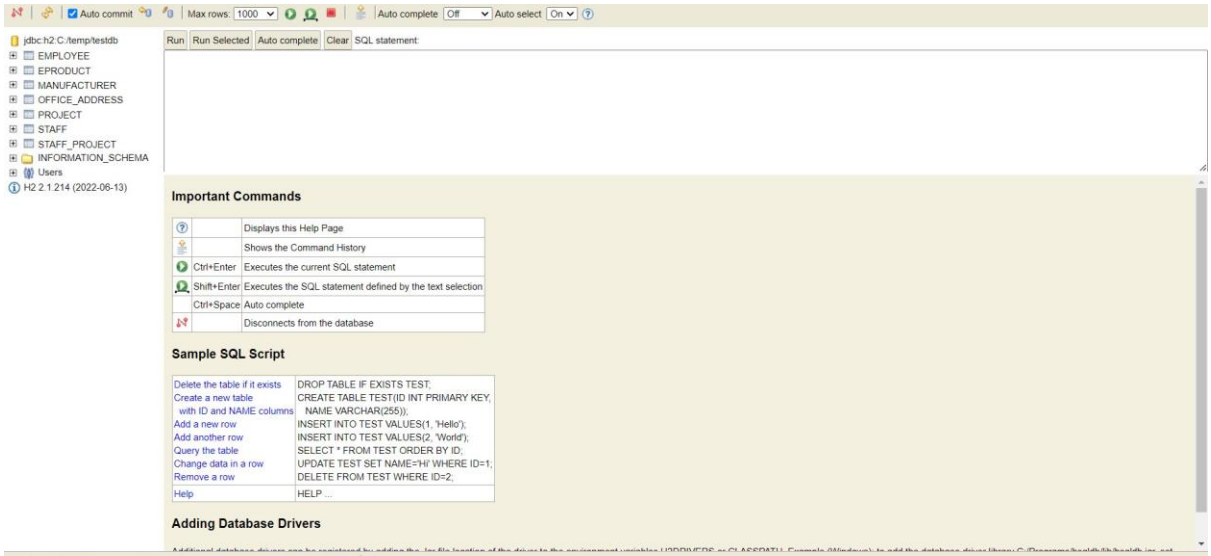
```
package com.ecommerce.repositries;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import com.ecommerce.entity.EProduct;

@Repository
public interface EProductRepo extends JpaRepository<EProduct, Integer>, JpaSpecificationExecutor {
```

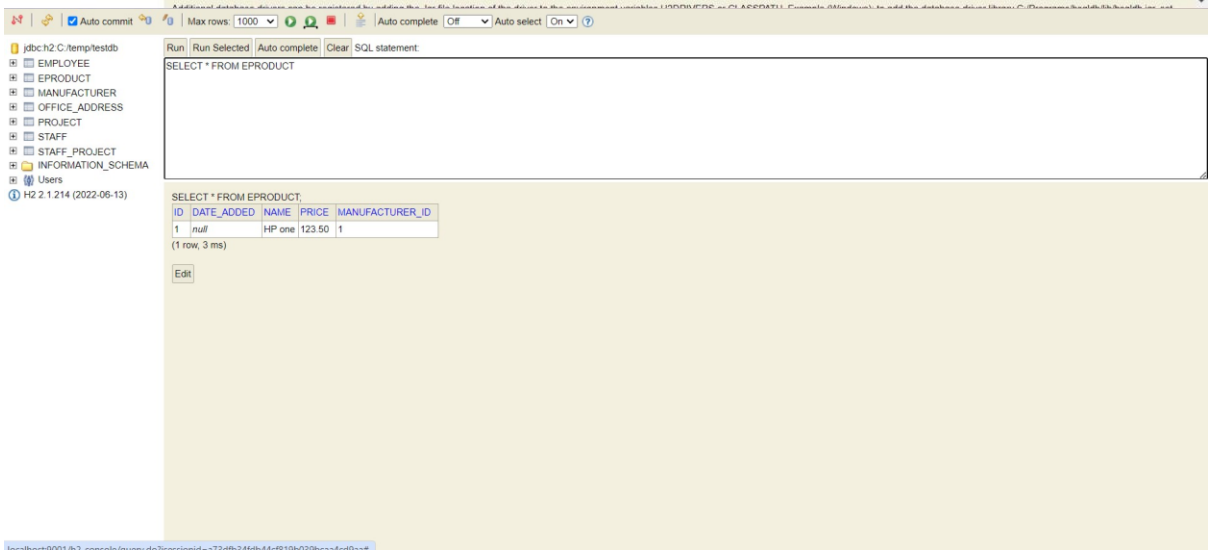
```
List<EProduct> findAllByName(String name);


List<EProduct> findAllByPrice(float price);


List<EProduct> findAllByPriceGreaterThan(float price);


// Partial name match using JPQL

@Query("SELECT p FROM EProduct p WHERE p.name LIKE %:name%")

List<EProduct> getAllProductHavingNameAnywhere(String name);


// Partial name match and price match using JPQL

@Query("SELECT p FROM EProduct p WHERE p.name LIKE %:name% and price < :price")

List<EProduct> getAllProductsHavingNameAnywhereAndPriceLT(String name, float price);


// Partial name match and price match using SQL

@Query(value="SELECT * FROM eproduct WHERE name LIKE %:name% and price > :price",
nativeQuery=true)

List<EProduct> getAllProductsHavingNameAnywhereAndPriceGT(String name, float price);


}
```

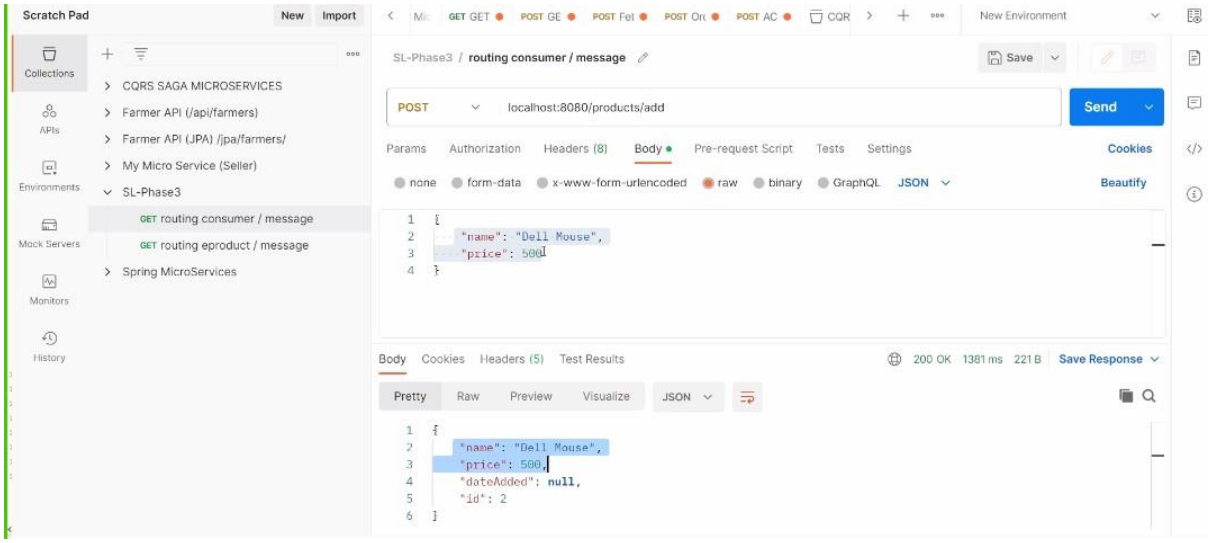
Pom.xml

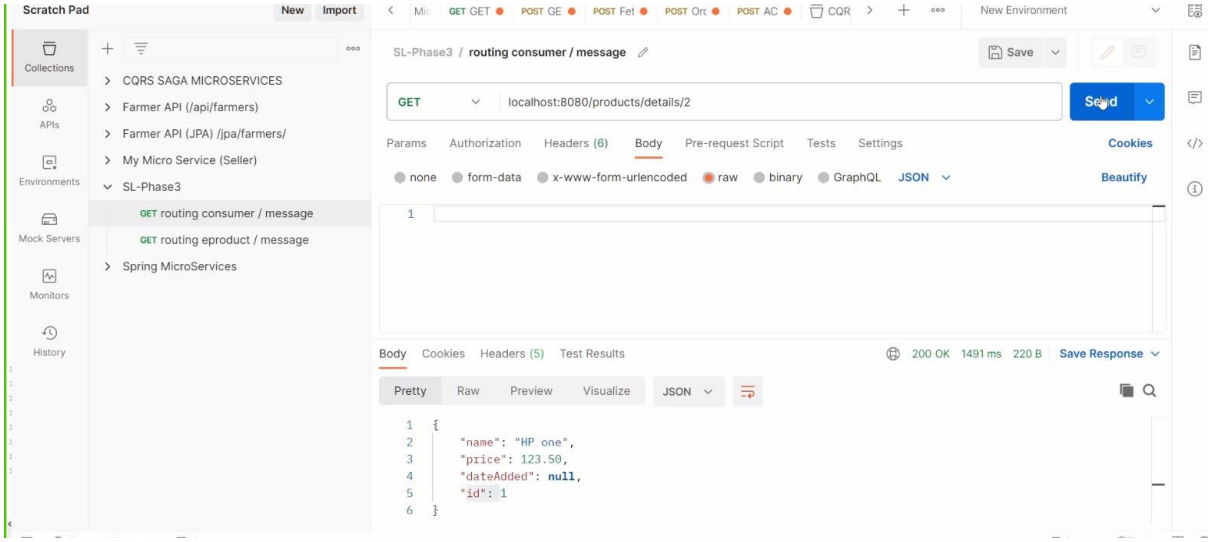
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>3.1.0</version>
            <relativePath/> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.example</groupId>
      <artifactId>building-rest-api-webservice-demo</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>building-rest-api-webservice-demo</name>
      <description>Demo project for Spring Boot</description>
      <properties>
```

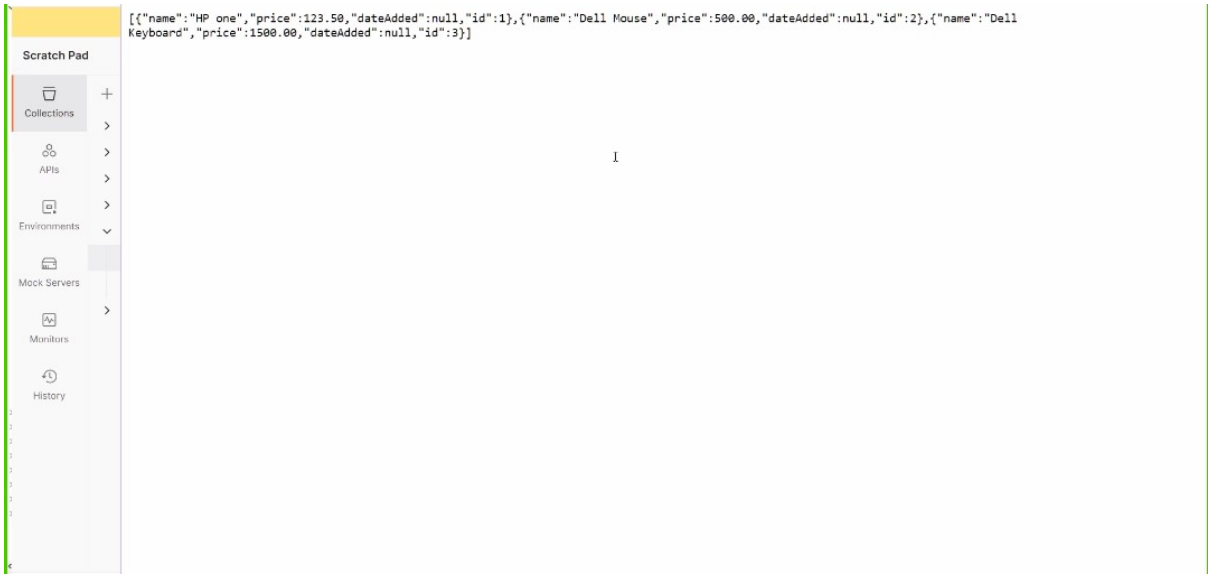
```
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>com.h2database</groupId>
			<artifactId>h2</artifactId>
			<scope>runtime</scope>
		</dependency>

		<!--

	https://mvnrepository.com/artifact/jakarta.servlet.jsp.jstl/jakarta.s
ervlet.jsp.jstl-api -->
		<dependency>
			<groupId>jakarta.servlet.jsp.jstl</groupId>
			<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
			<version>3.0.0</version>
		</dependency>

		<!-- JSP support in Spring -->
		<dependency>
			<groupId>org.apache.tomcat.embed</groupId>
			<artifactId>tomcat-embed-jasper</artifactId>
			<scope>provided</scope>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```


Auto commit
Max rows: 1000
Auto complete Off
Auto select On
jdbc:h2:C:/temp/testdb
EMPLOYEE
EPRODUCT
MANUFACTURER
OFFICE_ADDRESS
PROJECT
STAFF
STAFF_PROJECT
INFORMATION_SCHEMA
Users
H2 2.1.214 (2022-06-13)
Run
Run Selected
Auto complete
Clear
SQL statement:
Important Commands
Displays this Help Page
Shows the Command History
Ctrl+Enter
Executes the current SQL statement
Shift+Enter
Executes the SQL statement defined by the text selection
Ctrl+Space
Auto complete
Disconnects from the database
Sample SQL Script
Delete the table if it exists
DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row
INSERT INTO TEST VALUES(1, 'Hello');
Add another row
INSERT INTO TEST VALUES(2, 'World');
Query the table
SELECT * FROM TEST ORDER BY ID;
Change data in a row
UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row
DELETE FROM TEST WHERE ID=2;
Help
HELP ...
Adding Database Drivers


Auto commit
Max rows: 1000
Auto complete Off
Auto select On
jdbc:h2:C:/temp/testdb
EMPLOYEE
EPRODUCT
MANUFACTURER
OFFICE_ADDRESS
PROJECT
STAFF
STAFF_PROJECT
INFORMATION_SCHEMA
Users
H2 2.1.214 (2022-06-13)
Run
Run Selected
Auto complete
Clear
SQL statement:
SELECT * FROM EPRODUCT
SELECT * FROM EPRODUCT;
ID DATE_ADDED NAME PRICE MANUFACTURER_ID
1 null HP one 123.50 1
(1 row, 3 ms)
Edit


Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
CQRS SAGA MICROSERVICES
Farmer API (/api/farmers)
Farmer API (JPA) /jpa/farmers/
My Micro Service (Seller)
SL-Phase3
GET routing consumer / message
GET routing eproduct / message
Spring MicroServices
New Environment
SL-Phase3 / routing consumer / message
Save
POST
localhost:8080/products/add
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Dell Mouse",
"price": 500
}
Body
Cookies
Headers (5)
Test Results
200 OK
1381 ms
221 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": "Dell Mouse",
"price": 500,
"dateAdded": null,
"id": 2
}


Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
CQRS SAGA MICROSERVICES
Farmer API (/api/farmers)
Farmer API (JPA) /jpa/farmers/
My Micro Service (Seller)
SL-Phase3
GET routing consumer / message
GET routing eproduct / message
Spring MicroServices
New Environment
SL-Phase3 / routing consumer / message
Save
GET
localhost:8080/products/details/2
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Body
Cookies
Headers (5)
Test Results
200 OK
1491 ms
220 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": "HP one",
"price": 123.50,
"dateAdded": null,
"id": 1
}


[{"name":"HP one","price":123.50,"dateAdded":null,"id":1},{"name":"Dell Mouse","price":500.00,"dateAdded":null,"id":2},{"name":"Dell Keyboard","price":1500.00,"dateAdded":null,"id":3}]
Scratch Pad
Collections
APIs
Environments
Mock Servers
Monitors
History