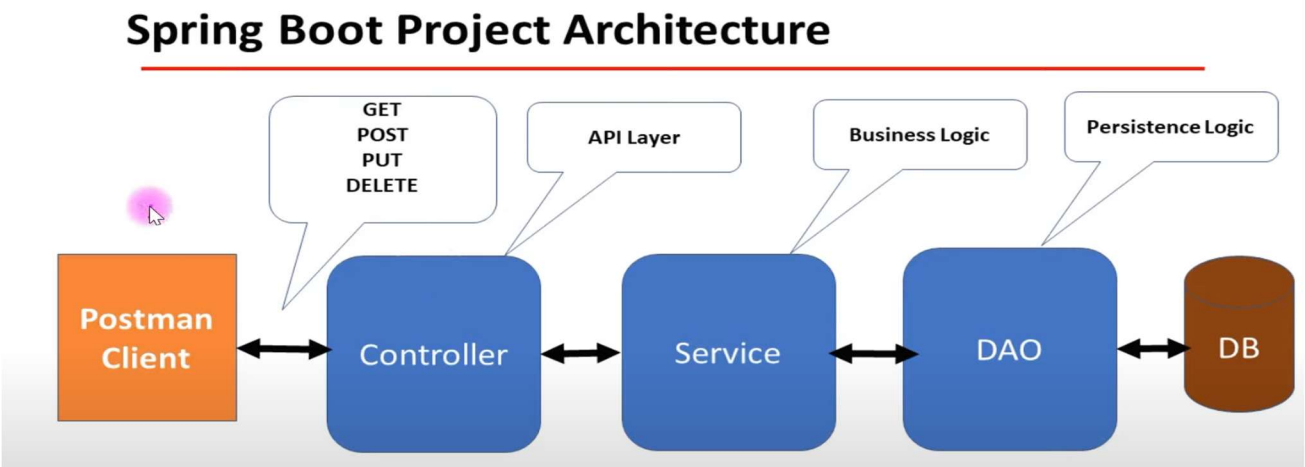
SPRING BOOT ASSIGNMENTS

6) Design a Spring Boot program to create a CRUD (Create, Read, Update, Delete) application using Hibernate for managing employee records. The program should allow users to perform the following operations on the employee database:

a) Add a new employee: The user can enter details like employee name, department, and salary, and the program should add the employee to the database.
b) Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.
c) Delete an employee: The user can delete an employee from the database based on their employee ID.
d) Display all employees: The program should retrieve and display a list of all employees and their details from the database.
e) Requirements:
   i) Use Spring Boot to create the application and Hibernate to manage the database.
   ii) Implement JPA (Java Persistence API) for data access.
   iii) Provide a RESTful API for performing CRUD operations on employees.
   iv) Implement exception handling to handle possible errors during database interactions.
   v) Cover Spring Boot and Hibernate topics, such as entity classes, repositories, services, and controllers.
f) Note: Before running the program, make sure you have set up the database and configured the connection in the application.properties file.

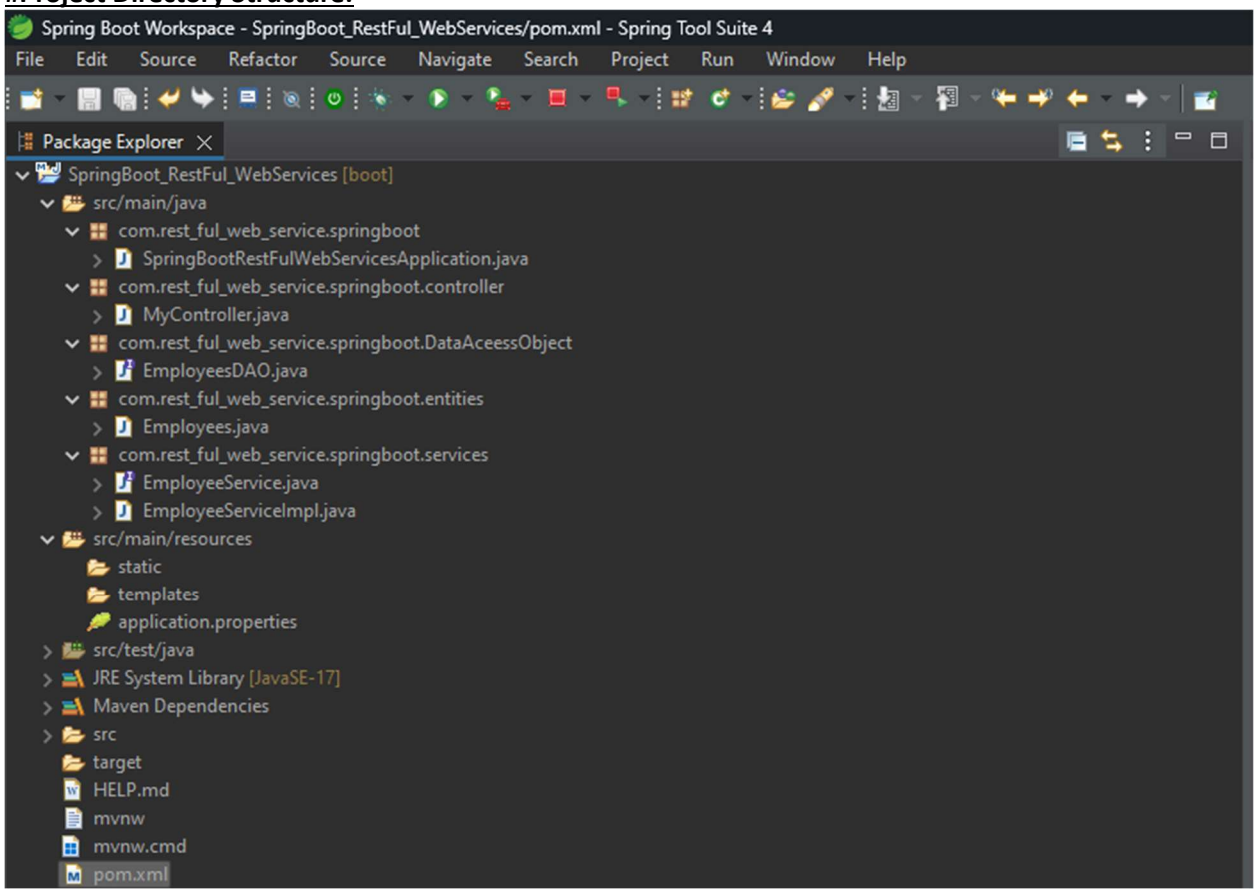
Implementation: As simple as that, created Employee Management system, via using Spring Boot, Hibernate and Oracle DB, as given in the task configured all prerequisites and followed MVC Design pattern.

Followed architecture shows actual Restful Service.



Same as given in Spring Boot Project Architecture, Created SpringBoot_RestFul_WebServices arranged in same manner.

#Project Directory Structure:



#pom.xml Responsible to resolved Maven Dependencies

```
<?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://ma-
ven.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>3.3.2</version>
            <relativePath /> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.SpringBootRest</groupId>
      <artifactId>SpringBoot_RestFul_WebServices</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>SpringBoot_RestFul_WebServices</name>
      <description>Spring Boot RestFul Web Services</description>
      <url />
      <licenses>
            <license />
      </licenses>
      <developers>
            <developer />
      </developers>
      <scm>
            <connection />
            <developerConnection />
            <tag />
            <url />
      </scm>
      <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.oracle.database.jdbc</groupId>
                  <artifactId>ojdbc11</artifactId>
                  <scope>runtime</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>
```

#application.properties: To configure prerequisites for application.

```
#Server Port Changed
server.port=9090

#Database Confiuration
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=8855

#Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

#Database table describtion generated by automatically

```
SQL> desc employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(5)
 NAME                                               VARCHAR2(255 CHAR)
 DEPT                                               VARCHAR2(255 CHAR)
 SALARY                                             FLOAT(53)

SQL>
```

#Employee.java class-showing entity of Spring Boot Rest Full Web Service application entities that physically present in Database and associated table.

```
package com.rest_ful_web_service.springboot.entities;

import jakarta.persistence.Column;
```

```java
import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Employees {

	@Id
	@Column(name = "ID", nullable = false)
	private long id;

	@Column(name = "name", nullable = false)
	private String name;


	@Column(name = "dept", nullable = false)
	private String dept;

	@Column(name = "salary", nullable = false)
	private double salary;

	public Employees() {
		super();
	}

	public Employees(long id, String name, String dept, double salary) {
		super();
		this.id = id;
		this.name = name;
		this.dept = dept;
		this.salary = salary;
	}

	public long getId() {
		return id;
	}

	public void setId(long id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getDept() {
		return dept;
```

```
	}

	public void setDept(String dept) {
		this.dept = dept;
	}

	public double getSalary() {
		return salary;
	}

	public void setSalary(double salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		// TODO Auto-generated method stub
		return super.toString();
	}
}
```

SPRING BOOT CONTROLLER LAYER:

MyController.java This API Layer class that responsible to interact with requests that can be GET, POST, PUT or DELETE.

```
package com.rest_ful_web_service.springboot.controller;

import java.util.List;

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

import com.rest_ful_web_service.springboot.entities.Employees;
import com.rest_ful_web_service.springboot.services.EmployeeService;

@RestController
public class MyController {

	@Autowired
	private EmployeeService employeeService;
```

```
	// Get all Employees
	@GetMapping("/Employees")
	public List<Employees> getEmployees() {
		return this.employeeService.getEmployees();
	}

	// Add Employee
	@PostMapping("/Employees")
	public Employees addEmployee(@RequestBody Employees employees) {
		return this.employeeService.addEmployee(employees);
	}

	// Update Employee
	@PutMapping("/Employees")
	public Employees updateEmployee(@RequestBody Employees employee) {
		return this.employeeService.updateEmployee(employee);
	}

	// DELETE Employee by ID
	@DeleteMapping("/Employees/{EmployeeID}")
	public ResponseEntity<HttpStatus> delEmployee(@PathVariable String Em-
ployeeID) {
		try {
			this.employeeService.delEmployee(Long.parseLong(Employ-
eeID));
			return new ResponseEntity<>(HttpStatus.OK);
		} catch (Exception e) {
			return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ER-
ROR);
		}
	}

}
```

SPRING BOOT SERVICE LAYER:

EmployeeService.java Interface responsible for business logic, interface shows the unimplemented methods that request can be come from controller layer.

```
package com.rest_ful_web_service.springboot.services;

import java.util.List;

import com.rest_ful_web_service.springboot.entities.Employees;

public interface EmployeeService {

	public List<Employees> getEmployees();

	public Employees addEmployee(Employees employees);
```

```
	public Employees updateEmployee(Employees employee);

	public void delEmployee(long parseLong);

}
```

EmployeeServiceImpl.java class responsible to implement unimplemented methods of interface.

```
package com.rest_ful_web_service.springboot.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.rest_ful_web_service.springboot.DataAceessObject.EmployeesDAO;
import com.rest_ful_web_service.springboot.entities.Employees;

@Service
public class EmployeeServiceImpl implements EmployeeService {

	@Autowired
	private EmployeesDAO employeeDAO;

	public EmployeeServiceImpl() {
		super();
	}

	// GET all employees
	@Override
	public List<Employees> getEmployees() {
		return employeeDAO.findAll();
	}


	// Add Employees by ID
	@Override
	public Employees addEmployee(Employees employee) {
		employeeDAO.save(employee);
		return employee;
	}

	// Update Employee by ID
	@Override
	public Employees updateEmployee(Employees employee) {
		employeeDAO.save(employee);
		return employee;
	}

	// DELETE Employee by ID
```

```
	@Override
	public void delEmployee(long parseLong) {
		Employees employee = employeeDAO.getReferenceById(parseLong);
		employeeDAO.delete(employee);
	}

}
```

SPRING BOOT DAO(Data Access Object) LAYER:

EmployeeDAO.java Interface responsible for persistence logic, interface extends another interface that is JpaRepository that offers convenient way to interact with database.

```
package com.rest_ful_web_service.springboot.DataAceessObject;

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

import com.rest_ful_web_service.springboot.entities.Employees;

public interface EmployeesDAO extends JpaRepository<Employees, Long> {

}
```

SPRING BOOT DAO(Data Access Object) LAYER:

SpringBootRestFulWebServicesApplication.java hold main method. JVM find the main method to kick-start Application.

```
package com.rest_ful_web_service.springboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringBootRestFulWebServicesApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootRestFulWebServicesApplica-
tion.class, args);
	}

}
```

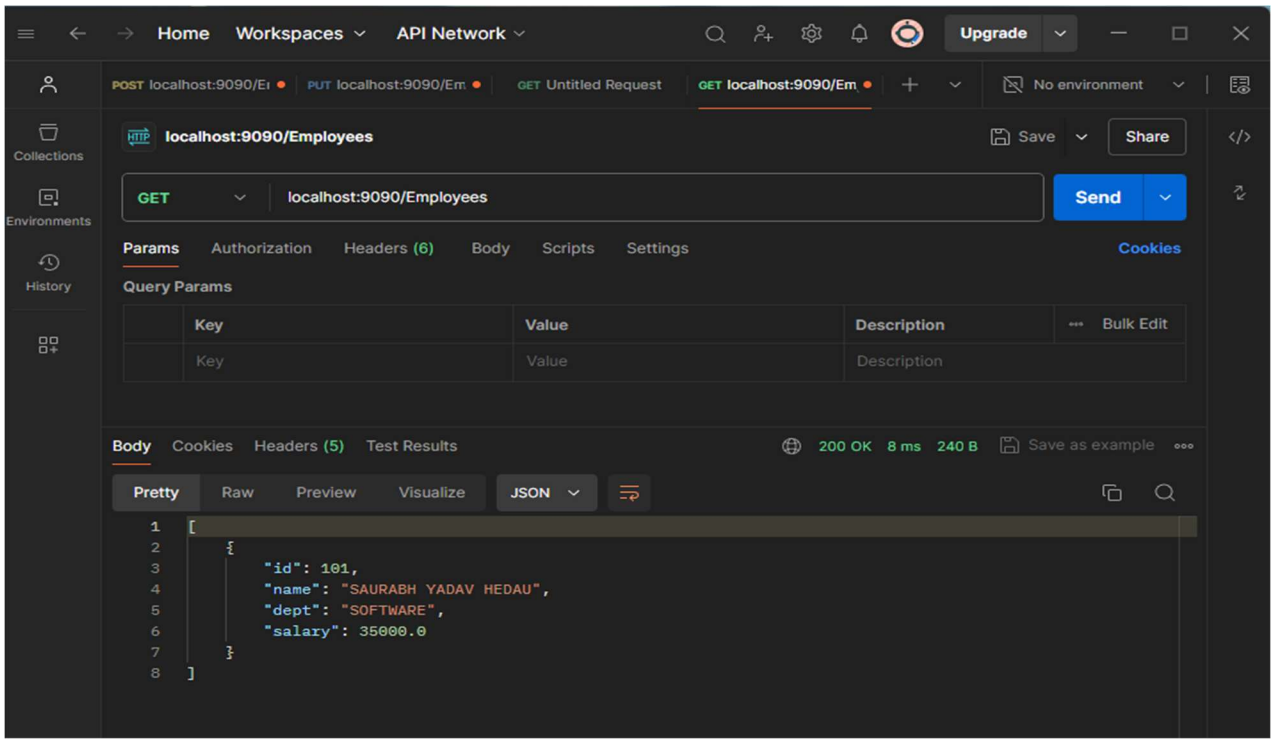
OUTPUT: Spring Boot Project Starter

```
application.properties   Employees.java   MyController.java   EmployeeService.java   EmployeeServiceImpl.java   EmployeesDAO.java   SpringBootRestFulWebServicesApplication.java   Console
SpringBoot_RestFul_WebServices - SpringBootRestFulWebServicesApplication [Spring Boot App] C:\Spring Tool Suite\sts-4.24.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (Aug 18, 2024, 3:08:07 PM) [pid: 21056]

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.2)

2024-08-18T15:08:09.148+05:30  INFO 21056 --- [                main] .SpringBootRestFulWebServicesApplication : Starting SpringBootRestFulWebServicesApplicatio
2024-08-18T15:08:09.151+05:30  INFO 21056 --- [                main] .SpringBootRestFulWebServicesApplication : No active profile set, falling back to 1 default
2024-08-18T15:08:09.598+05:30  INFO 21056 --- [                main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DE
2024-08-18T15:08:09.639+05:30  INFO 21056 --- [                main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 35 m
2024-08-18T15:08:10.026+05:30  INFO 21056 --- [                main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 9090 (http)
2024-08-18T15:08:10.048+05:30  INFO 21056 --- [                main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-08-18T15:08:10.049+05:30  INFO 21056 --- [                main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.26]
2024-08-18T15:08:10.107+05:30  INFO 21056 --- [                main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationCont
2024-08-18T15:08:10.108+05:30  INFO 21056 --- [                main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization comp
2024-08-18T15:08:10.259+05:30  INFO 21056 --- [                main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name
2024-08-18T15:08:10.319+05:30  INFO 21056 --- [                main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.5.2.Fin
2024-08-18T15:08:10.355+05:30  INFO 21056 --- [                main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2024-08-18T15:08:10.624+05:30  INFO 21056 --- [                main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class tra
2024-08-18T15:08:10.650+05:30  INFO 21056 --- [                main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-18T15:08:10.900+05:30  INFO 21056 --- [                main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection oracle.jdbc.dri
2024-08-18T15:08:10.901+05:30  INFO 21056 --- [                main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-18T15:08:11.793+05:30  INFO 21056 --- [                main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibe
2024-08-18T15:08:12.167+05:30  INFO 21056 --- [                main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persis
2024-08-18T15:08:12.346+05:30  WARN 21056 --- [                main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default.
2024-08-18T15:08:12.615+05:30  INFO 21056 --- [                main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 9090 (http) with context
2024-08-18T15:08:12.624+05:30  INFO 21056 --- [                main] .SpringBootRestFulWebServicesApplication : Started SpringBootRestFulWebServicesApplication
```

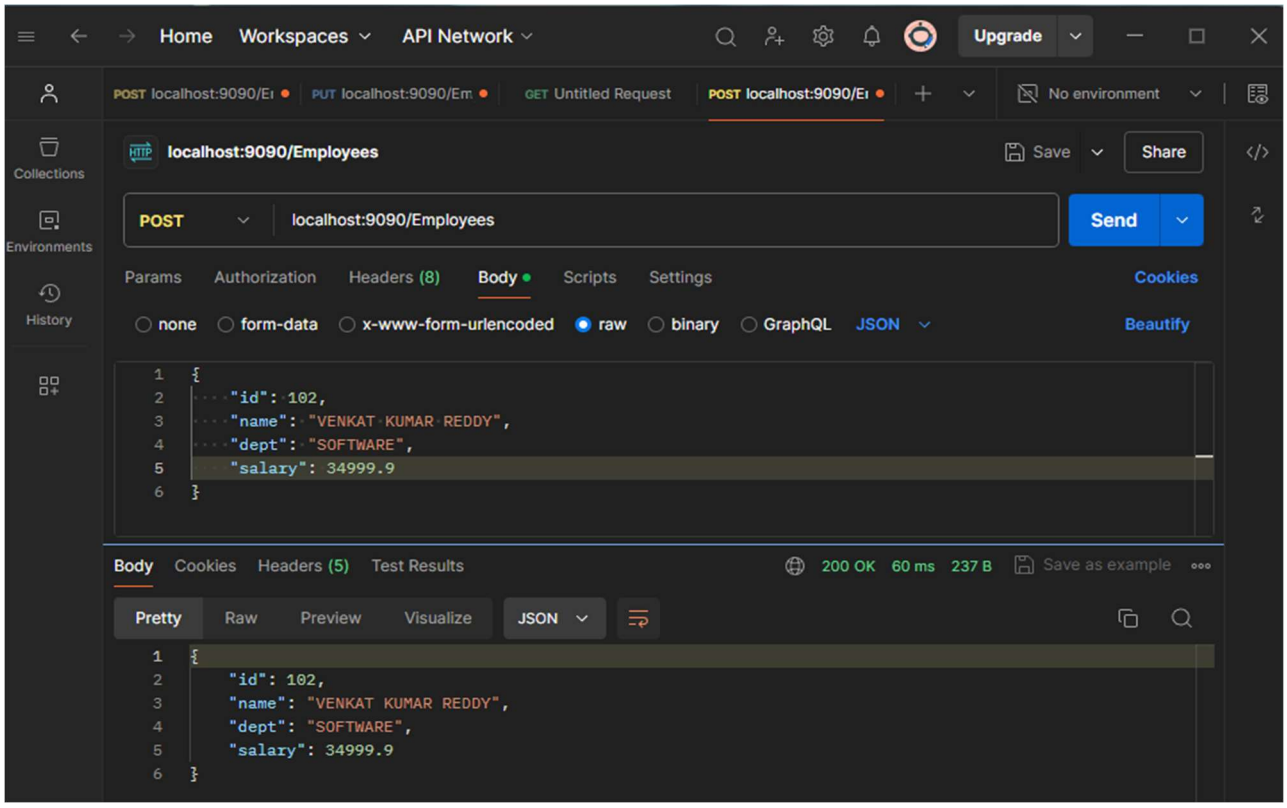
Application Testing:

Postman Rest-line testing tool that confirming API working with expected results.

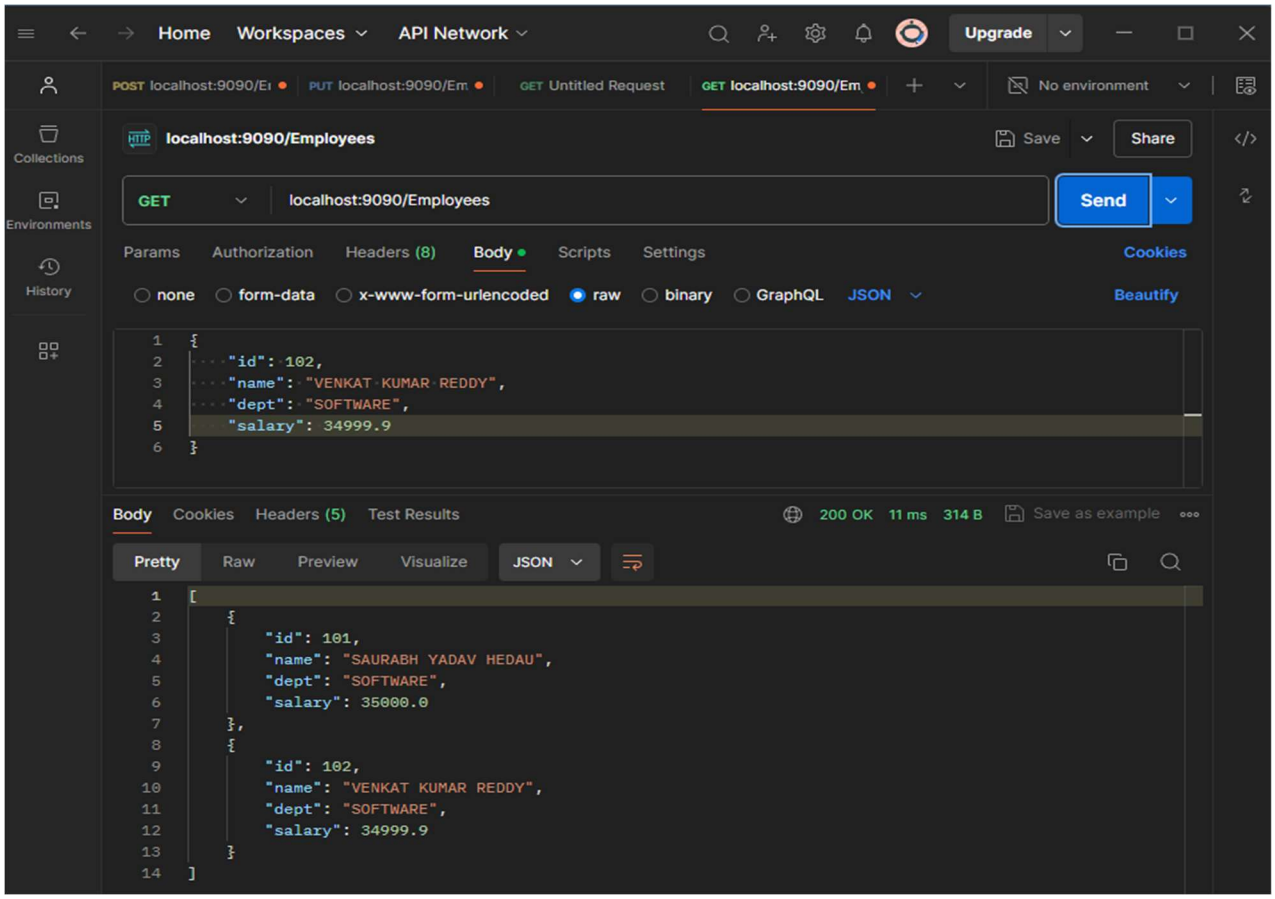
#GET method:



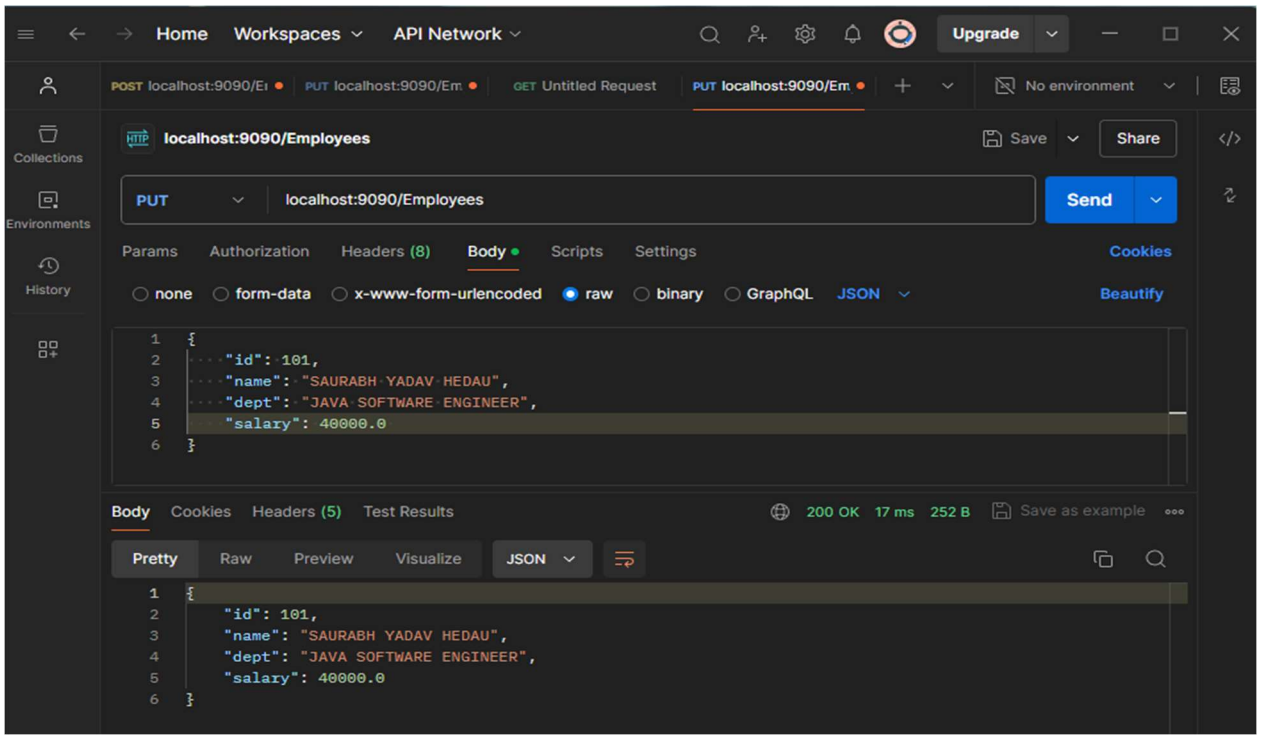
#POST method:



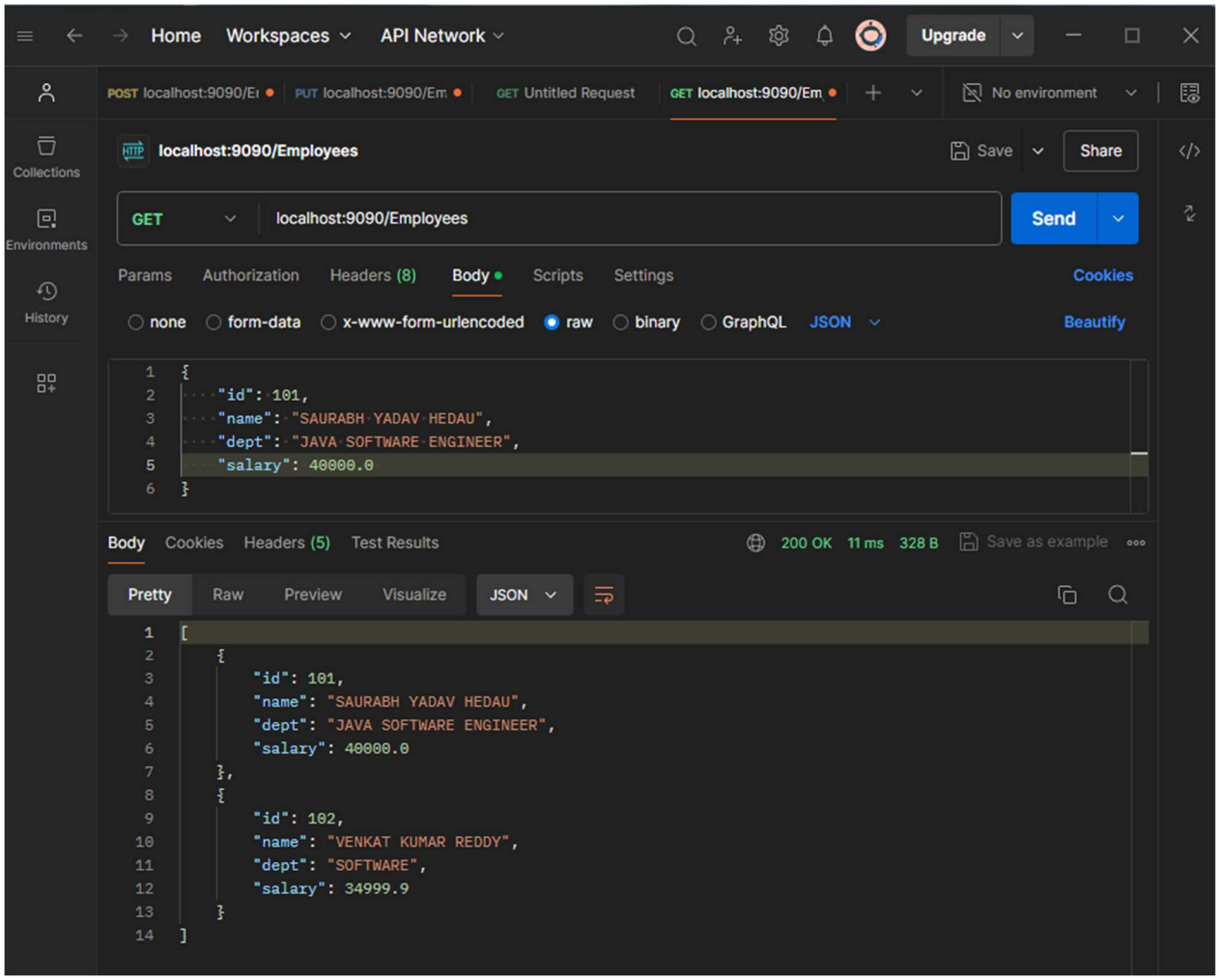
#GET method after POST method:


Home
Workspaces
API Network
Upgrade
POST localhost:9090/E
PUT localhost:9090/Em
GET Untitled Request
GET localhost:9090/Em
No environment
Collections
Environments
History
localhost:9090/Employees
Save
Share
GET
localhost:9090/Employees
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 102,
"name": "VENKAT KUMAR REDDY",
"dept": "SOFTWARE",
"salary": 34999.9
}
Body
Cookies
Headers (5)
Test Results
200 OK
11 ms
314 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 101,
"name": "SAURABH YADAV HEDAU",
"dept": "SOFTWARE",
"salary": 35000.0
},
{
"id": 102,
"name": "VENKAT KUMAR REDDY",
"dept": "SOFTWARE",
"salary": 34999.9
}
]

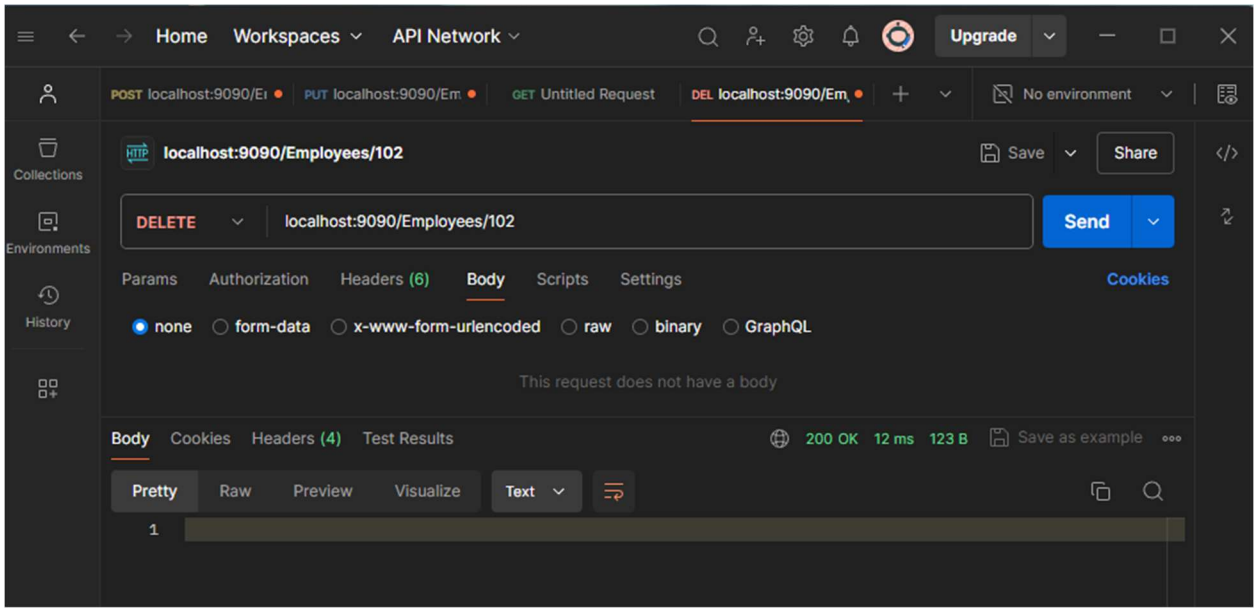
#PUT method:


Home
Workspaces
API Network
Upgrade
POST localhost:9090/E
PUT localhost:9090/Em
GET Untitled Request
PUT localhost:9090/Em
No environment
Collections
Environments
History
localhost:9090/Employees
Save
Share
PUT
localhost:9090/Employees
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 101,
"name": "SAURABH YADAV HEDAU",
"dept": "JAVA SOFTWARE ENGINEER",
"salary": 40000.0
}
Body
Cookies
Headers (5)
Test Results
200 OK
17 ms
252 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 101,
"name": "SAURABH YADAV HEDAU",
"dept": "JAVA SOFTWARE ENGINEER",
"salary": 40000.0
}

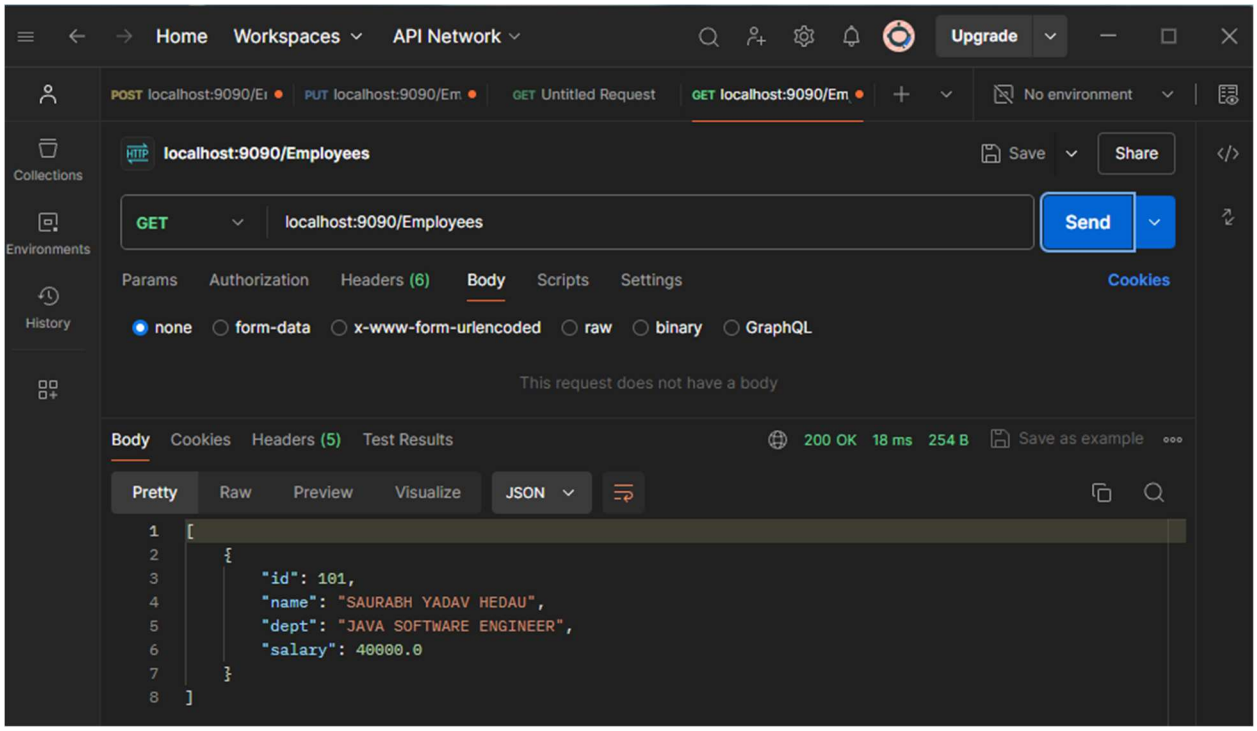
#GET method after PUT method:



#DELETE method:



#GET method after DELETE method:



#DATABASE TABEL