LANGKAH-LANGKAH MEMBUAT SPRING BOOT + MYSQL BACKEND MENGGUNAKAN VS CODE

1. Install setiap tools yang dibutuhkan:

Pastikan tools di bawah ini sudah terinstall:

a. Java Development Kit (JDK)

- Install **JDK 17 or 21**
- Untuk verifikasi bisa melakukan perintah ini dalam cmd/terminal: java -version

b. Visual Studio Code

- Install ekstensi VS Code:
  - **Extension Pack for Java**
  - **Spring Boot Dashboard**
  - **Spring Initializr**
  - **Lombok Annotations Support** (opsional tapi direkomendasikan)

c. MySQL Server + MySQL Workbench

- Buat database baru: CREATE DATABASE dbmarket;

2. Membuat Spring Boot Project (Menggunakan Spring Initializr in VS Code)

1. Buka VS Code
2. Tekan **Ctrl + Shift + P**
3. Pilih **"Spring Initializr: Create a Maven Project"**
4. Pilih **Spring Boot version** (3.x recommended)
5. Pilih **Project language :** Java
6. Isi:
   - Group: `com.rmn`
   - Artifact: `crud`
7. Pilih **Packaging Type:** JAR
8. Pilih versi **Java 17** atau **21**
9. Tambahkan dependencies:
   - **Spring Web**
   - **Spring Data JPA**
   - **Spring Boot DevTools**
   - **Spring Security**
   - **MySQL Driver**
   - **Lombok**
10. Generate the project → choose a folder → Finish.

3. Konfigurasi MySQL Connection dalam application.properties

Dalam src/main/resources/application.properties:

```
spring.application.name=crud
# --- Konfigurasi Server ---
server.port=8080

# --- Konfigurasi MySQL ---
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/dbmarket?useSSL=false&ser
verTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

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

4. Membuat Model (Entity)

Model (Entity): Digunakan untuk merepresentasikan data. Buat folder model dalam package com.rmn dan masukkan kelas baru. Contoh: Employee.java yang merepresentasikan tabel Employee pada database dbmarket

```
package com.rmn.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import java.time.LocalDate;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @Column(name = "EmpCode")
    private String empCode;

    @Column(name = "EmpName")
    private String empName;

    @Column(name = "EmpAddress")
    private String empAddress;

    @Column(name = "EmpZipCode")
    private String empZipCode;

    @Column(name = "EmpDOB")
    private LocalDate empDOB;

    public Employee() {}

    public String getEmpCode() {
        return empCode;
    }

    public void setEmpCode(String empCode) {
        this.empCode = empCode;
    }

    public String getEmpName() {
        return empName;
```

```
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    public String getEmpAddress() {
        return empAddress;
    }

    public void setEmpAddress(String empAddress) {
        this.empAddress = empAddress;
    }

    public String getEmpZipCode() {
        return empZipCode;
    }

    public void setEmpZipCode(String empZipCode) {
        this.empZipCode = empZipCode;
    }

    public LocalDate getEmpDOB() {
        return empDOB;
    }

    public void setEmpDOB(LocalDate empDOB) {
        this.empDOB = empDOB;
    }
}
```

5. Membuat Repository

Dalam Spring Framework, Repository adalah komponen yang digunakan untuk tujuan utama yaitu mengakses data (persistance) dan mengabstraksi logika interaksi dengan database. Sekarang buatlah folder baru bernama repository dalam package com.rmn dan kelas EmployeeRepository.java lalu masukkan kode berikut:

```
package com.rmn.repository;

import com.rmn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, String>
{
    List<Employee> findByEmpNameContainingIgnoreCase(String empName);

    List<Employee> findByEmpZipCode(String empZipCode);

    List<Employee> findByEmpNameContainingIgnoreCaseAndEmpZipCode(
            String empName,
            String empZipCode
    );
```

```
}
```

6. Membuat Service

Jika Repository digunakan untuk mengakses data, maka Service (atau *Service Layer*) adalah komponen inti di Spring yang digunakan untuk tujuan utama yaitu mengimplementasikan logika bisnis (Business Logic) dan mengkoordinasikan alur kerja aplikasi. Sekarang buatlah folder baru bernama service dalam package com.rmn dan kelas EmployeeService.java lalu masukkan kode berikut:

```
package com.rmn.service;

import com.rmn.model.Employee;
import com.rmn.repository.EmployeeRepository;
import org.springframework.stereotype.Service;

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

@Service
public class EmployeeService {

    private final EmployeeRepository employeeRepo;

    public EmployeeService(EmployeeRepository employeeRepo) {
        this.employeeRepo = employeeRepo;
    }

    public List<Employee> getAll() {
        return employeeRepo.findAll();
    }

    public Optional<Employee> getByCode(String code) {
        return employeeRepo.findById(code);
    }

    public Employee create(Employee emp) {
        return employeeRepo.save(emp);
    }

    public Employee update(String code, Employee updated) {
        return employeeRepo.findById(code)
            .map(emp -> {
                emp.setEmpName(updated.getEmpName());
                emp.setEmpAddress(updated.getEmpAddress());
                emp.setEmpZipCode(updated.getEmpZipCode());
                emp.setEmpDOB(updated.getEmpDOB());
                return employeeRepo.save(emp);
            })
            .orElseThrow(() -> new RuntimeException("Employee not found"));
    }

    public void delete(String code) {
        employeeRepo.deleteById(code);
```

```
    }

    public List<Employee> searchEmployees(String name, String zipcode) {
        if (name != null && zipcode != null) {
                                                                    return
employeeRepo.findByEmpNameContainingIgnoreCaseAndEmpZipCode(name, zipcode);
        }
        if (name != null) {
            return employeeRepo.findByEmpNameContainingIgnoreCase(name);
        }
        if (zipcode != null) {
            return employeeRepo.findByEmpZipCode(zipcode);
        }

        return employeeRepo.findAll();
    }
}
```

7. Membuat Controller REST API

Ini adalah komponen yang bertugas menerima permintaan HTTP dari klien dan mengirimkan respons kembali. *Controller* digunakan untuk mengekspos *endpoint* REST.

Buat folder baru (misalnya com.rmn.controller) dan buat kelas berikut:

Contoh: EmployeeController.java

```
package com.rmn.controller;

import com.rmn.model.Employee;
import com.rmn.service.EmployeeService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/employees")
public class EmployeeController {

    private final EmployeeService service;

    public EmployeeController(EmployeeService service) {
        this.service = service;
    }

    @GetMapping
    public List<Employee> getAll() {
        return service.getAll();
    }

    @GetMapping("/{code}")
    public Employee getByCode(@PathVariable String code) {
        return service.getByCode(code)
                .orElseThrow(()   ->   new   RuntimeException("Employee   not
found"));
    }
```

```
    @PostMapping
    public Employee create(@RequestBody Employee emp) {
        return service.create(emp);
    }

    @PutMapping("/{code}")
    public Employee update(@PathVariable String code, @RequestBody Employee
emp) {
        return service.update(code, emp);
    }

    @DeleteMapping("/{code}")
    public String delete(@PathVariable String code) {
        service.delete(code);
        return "Deleted: " + code;
    }

    @GetMapping("/search")
    public List<Employee> searchEmployees(
        @RequestParam(required = false) String name,
        @RequestParam(required = false) String zipcode
    ) {
        return service.searchEmployees(name, zipcode);
    }
}
```

8. Membuat SecurityConfig

Ini adalah konfigurasi yang kita buat untuk menonaktifkan form *login* Spring Security, memungkinkan akses ke API Anda. Buat *package* baru (misalnya com.rmn.config) dan buat kelas berikut:

```
package com.rmn.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebS
ecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        http
            .csrf(csrf -> csrf.disable())

            .authorizeHttpRequests(authorize -> authorize
```

```
                .anyRequest().permitAll()
            );

        return http.build();
    }
}
```

9. Memindahkan Main Class

Pindahkan main class (Misal: CrudApplication) ke dalam package com.rmn sehingga kodenya seperti berikut:

```
package com.rmn;

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

@SpringBootApplication
public class CrudApplication {

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

}
```

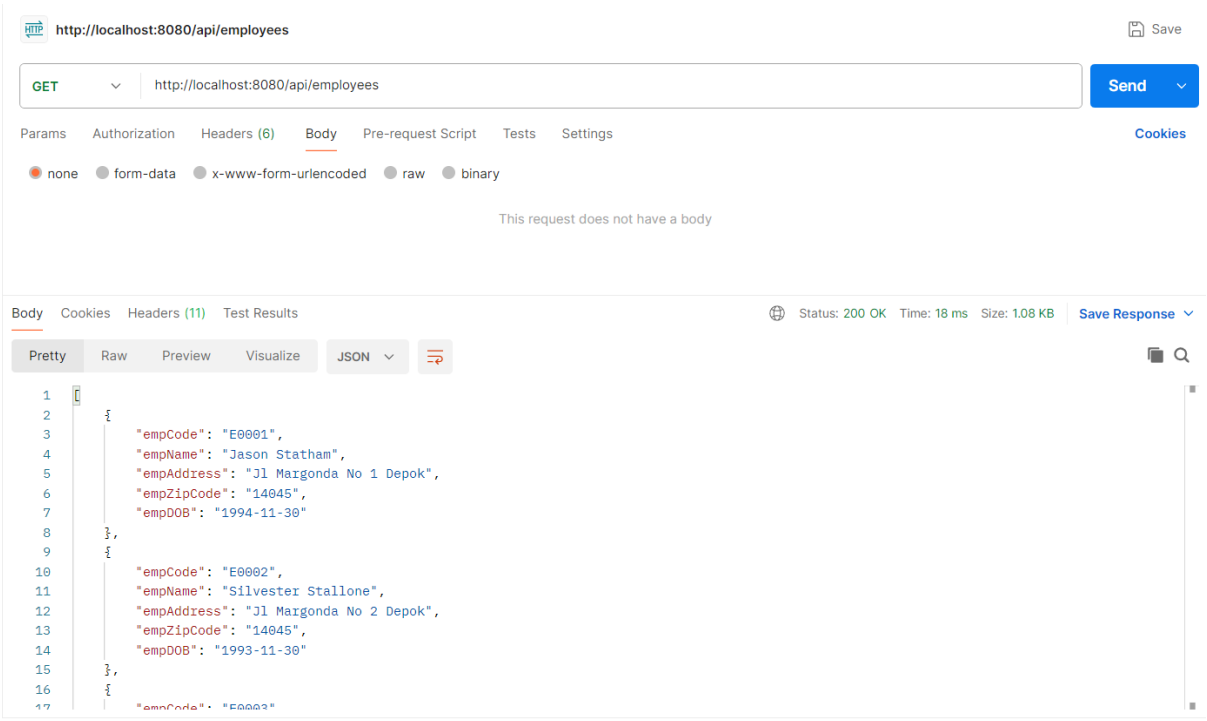
Struktur Akhir Projek

10. Menjalankan Aplikasi

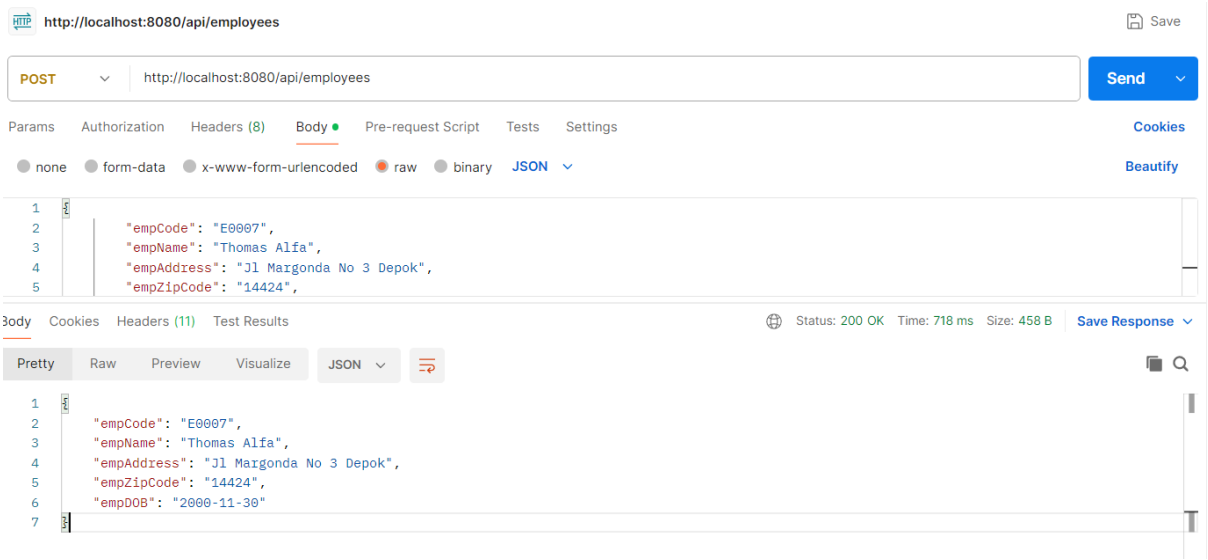
Pada VS Code:

- Gunakan terminal:
  `./mvnw spring-boot:run`

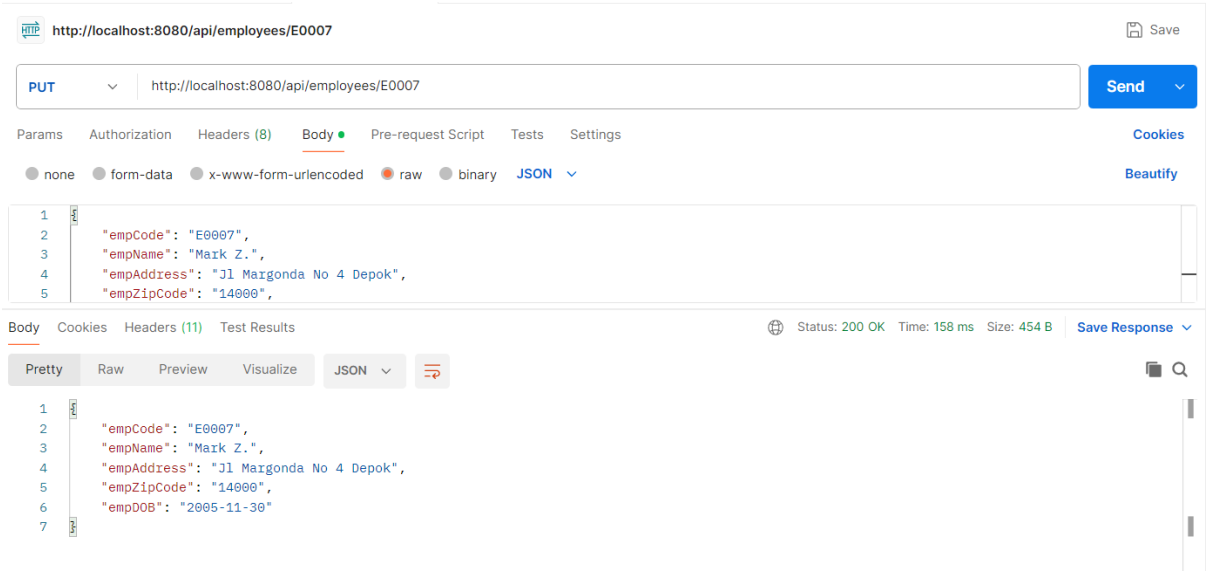
Check API menggunakan Postman:



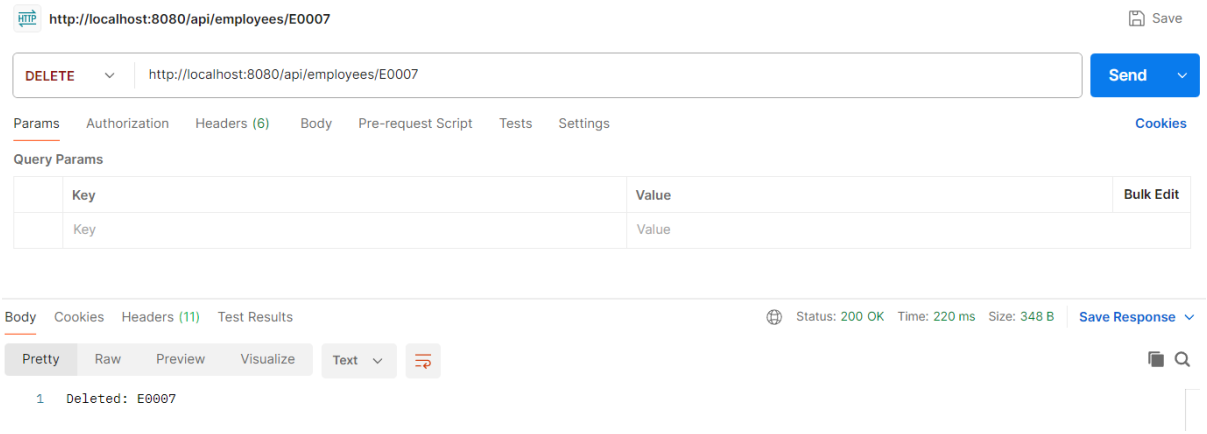
Menampilkan semua data Employee



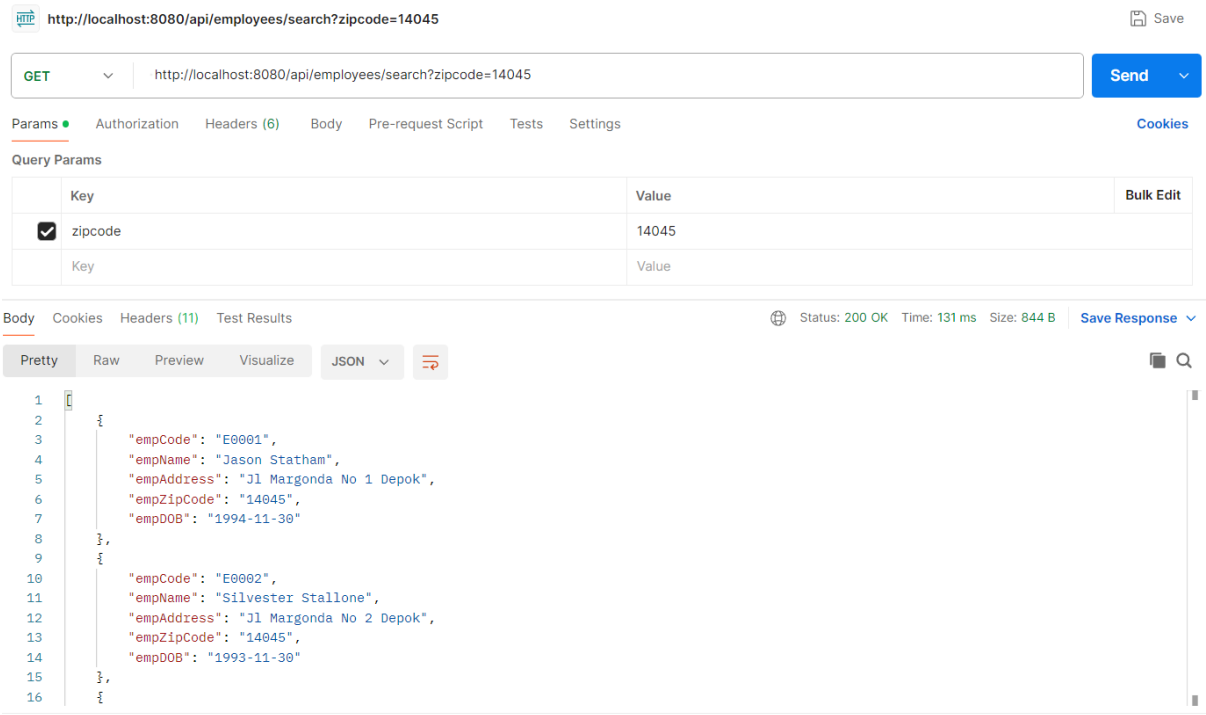
Memasukkan data Employee



Mengedit data Employee



Menghapus data Employee



Mencari data Employee