PROJECT SERVICE MAHASISWA

Disusun untuk memenuhi tugas

MATA KULIAH : SISTEM TERDISTRIBUSI

DOSEN PENGAMPU : ERVAN ASRI, S.Kom., M.KOM

NAMA : KURNIA ADILLA

NO.BP : 2111082021

KELAS : TRPL 3C

Program Studi DIV Teknologi Rekayasa Perangkat Lunak

Jurusan Teknologi Informasi

POLITEKNIK NEGERI PADANG

2022/2023

BAB I
PENDAHULUAN

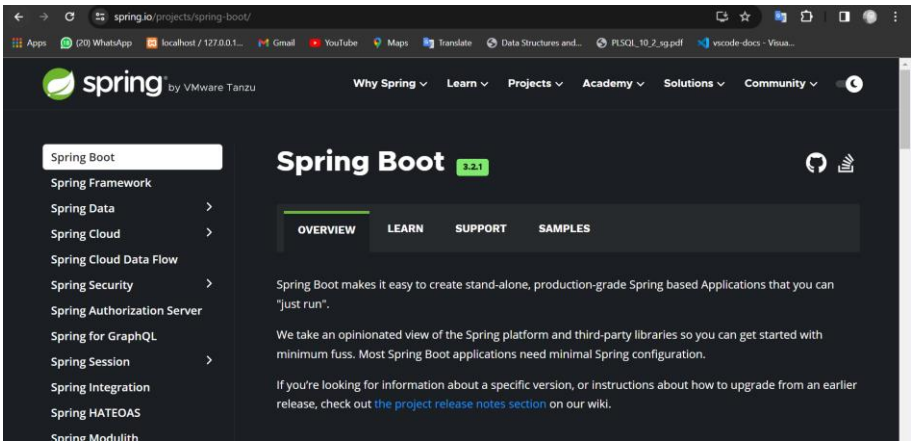
A. Tujuan

- Membuat Service Mahasiswa, Matakuliah dan Nilai

B. Teori Singkat

Service adalah bagian yang menyediakan layanan atau fungsi tertentu kepada pengguna atau aplikasi lainnya melalui jaringan. Layanan ini dapat mencakup berbagai hal, seperti pemrosesan data, penyimpanan, komunikasi, atau fungsionalitas lainnya yang dapat diakses dan digunakan oleh komponen sistem terdistribusi yang berbeda.

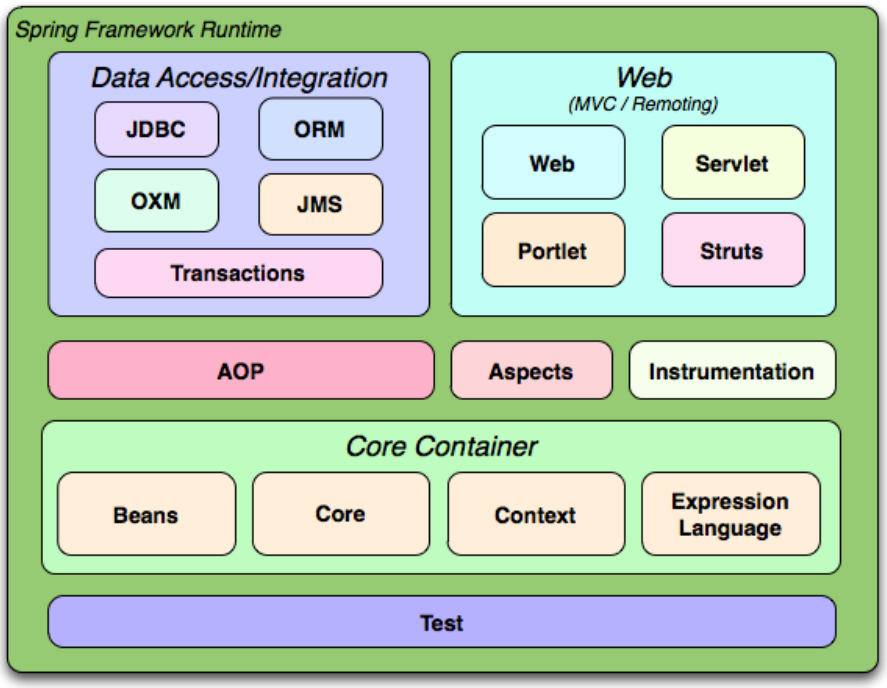
Spring Boot adalah framework aplikasi untuk platform Java yang bersifat open source. Spring Boot adalah bagian dari Spring Framework yang membantu pengembang Java membuat aplikasi dengan lebih mudah.



Spring Boot memudahkan pengembang aplikasi berbasis Java untuk memilih library Java yang akan digunakan. Spring Boot juga memudahkan pengembang untuk mengatur, mengkonfigurasi, dan menjalankan aplikasi berbasis Spring.

Spring Boot juga menyediakan Tomcat dan beberapa server lain, sehingga pengembang hanya perlu menjalankan program. Spring Boot juga menggunakan Maven sebagai build manager yang dapat diatur di Project Object Model (POM).

Spring Boot banyak digunakan oleh perusahaan Indonesia dan luar negeri. Spring Boot memudahkan proses web development di Java. Spring Boot juga memudahkan penggunaan kerangka kerja berbasis Java untuk membuat layanan mikro dan aplikasi web.



1. **SPRING FRAMEWORK**

- ***org.springframework.beans.factory.annotation.Autowired***

Mengindikasikan penggunaan fitur "Dependency Injection" dari Spring Framework.

- ***org.springframework.stereotype.Service***

Mengindikasikan bahwa kelas ini adalah bagian dari layanan (service) Spring.

- ***org.springframework.web.bind.annotation.RestController***

Anotasi ini menandakan bahwa kelas tersebut adalah bagian dari lapisan kontroller (controller) Spring untuk pengembangan aplikasi web.

- ***org.springframework.boot.SpringApplication***

Kelas ini adalah bagian dari Spring Boot, yang menyediakan utilitas untuk memulai dan mengelola aplikasi Spring Boot.

- ***org.springframework.boot.autoconfigure.SpringBootApplication***

Anotasi ini digunakan untuk menandai kelas utama aplikasi Spring Boot. Anotasi ini menggabungkan tiga anotasi lainnya: @Configuration, @EnableAutoConfiguration, dan @ComponentScan.

2. **SPRING FRAMEWORK (SPRING DATA JPA)**

- ***org.springframework.data.jpa.repository.JpaRepository***

Merupakan bagian dari Spring Data JPA, yang menyediakan antarmuka

- ***JpaRepository***

Untuk melakukan operasi dasar pada entitas JPA.

3. **SPRING FRAMEWORK (SPRING CORE)**

- ***org.springframework.stereotype.Repository***

Anotasi yang digunakan untuk menandai bahwa kelas tersebut adalah bagian dari lapisan repository Spring.

4. SPRING MVC (MODEL-VIEW-CONTROLLER)

- ***org.springframework.web.bind.annotation.GetMapping***

Anotasi yang menandakan bahwa metode tersebut menangani permintaan HTTP GET.

- ***org.springframework.web.bind.annotation.PostMapping***

Anotasi yang menandakan bahwa metode tersebut menangani permintaan HTTP POST.

- ***org.springframework.web.bind.annotation.PutMapping***

Anotasi yang menandakan bahwa metode tersebut menangani permintaan HTTP PUT.

- ***org.springframework.web.bind.annotation.DeleteMapping***

Anotasi yang menandakan bahwa metode tersebut menangani permintaan HTTP DELETE.

- ***org.springframework.web.bind.annotation.RequestMapping***

Anotasi ini digunakan untuk menetapkan jalur dasar untuk semua metode di dalam kelas ini.

5. JAKARTA TRANSACTION API (JTA)

- ***jakarta.transaction.Transactional***

Anotasi ini digunakan dalam konteks transaksi dan biasanya digunakan bersama dengan JTA.

6. JAVA STANDARD LIBRARY

- ***java.util.Optional***

Kelas ini termasuk dalam Java Standard Library dan digunakan untuk menyatakan opsional nilai, yang sesuai dengan praktik baik dalam Java modern.

- ***java.util.List***

Kelas ini termasuk dalam Java Standard Library dan digunakan untuk merepresentasikan daftar objek.

7. JAVA PERSISTENCE API (JPA) / JAKARTA PERSISTENCE API

- ***jakarta.persistence.Entity***

Anotasi ini menandakan bahwa kelas tersebut adalah entitas JPA atau Jakarta Persistence API, yang dapat dipersistensi ke dalam database.

- ***jakarta.persistence.GeneratedValue***

Anotasi ini menandakan bahwa nilai dari suatu atribut (biasanya yang berfungsi sebagai kunci utama) akan dihasilkan secara otomatis oleh database.

- ***jakarta.persistence.GenerationType***

  Enumerasi yang menyediakan strategi identitas (IDENTITY), sequence (SEQUENCE), atau tabel (TABLE) untuk menghasilkan nilai kunci.

- ***jakarta.persistence.Id***

Anotasi ini menandakan bahwa suatu atribut adalah identitas utama (primary key) dari entitas.

- ***jakarta.persistence.Table***

Anotasi ini dapat digunakan untuk menyesuaikan konfigurasi tabel database yang sesuai dengan entitas.

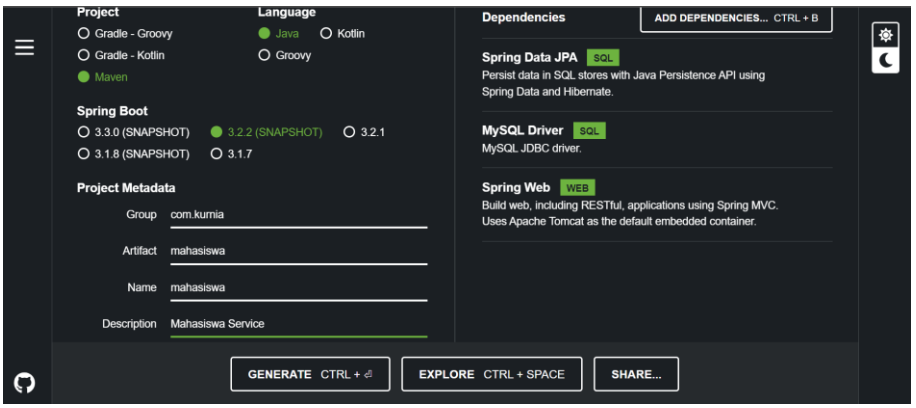
BAB II
PEMBAHASAN

A. Alat dan Bahan

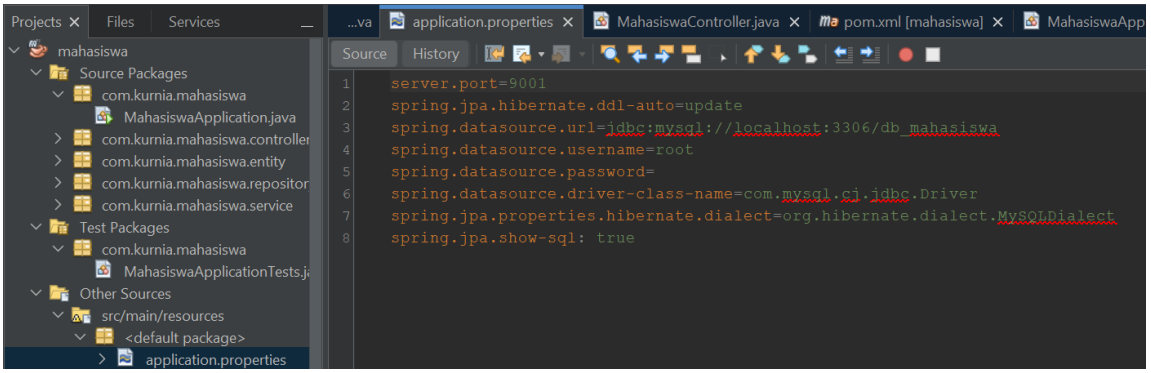
1. Laptop atau Komputer
2. Spring boot
3. XAMPP
4. Apache Neatbens
5. Dependencies (Spring Web, Spring Data JPA, MySQL JDBC driver)

B. Langkah Kerja

1. Langkah pertama dalam membuat service mahasiswa yaitu kita menggunakan start.spring.io menggunakan JDK 21 atau 17 serta spring boot 3.2.0, dan tambahkan dependencies, **spring data JPA, MySQL Driver, Spring Web.**



2. Setelah itu **Generate,** extrak rar dan jalankan melalui neatbens.

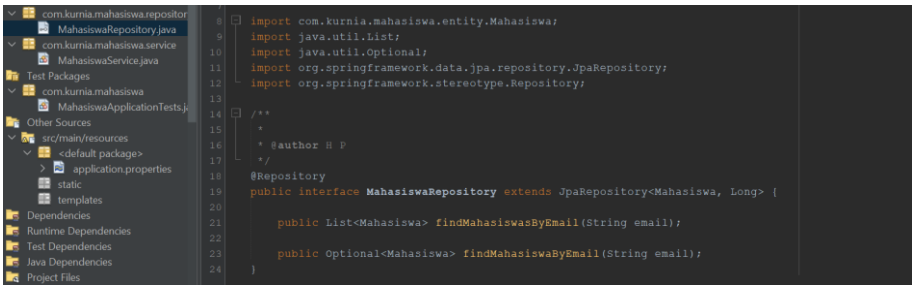


3. Gunakan web server xampp untuk membuat database sesuai dengan yang nama di application.properties

4. Selanjutnya buat package baru dengan nama mahasiswa.entity serta java class Mahasiswa.java untuk membuat table di dalam database dengan kolom id, nama, dan email serta constructor, setter dan getternya.Gunakan @Entity untuk menndakan bahwa java class ini adalah entity

```
public class Mahasiswa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nama;
    private String email;

    public Mahasiswa() {
    }

    public Mahasiswa(Long id, String nama, String email) {
        this.id = id;
        this.nama = nama;
        this.email = email;
    }
```

5. Setelah itu buat package baru lagi dengan nama mahasiswa.repository, dengan java interface MahasiswaRepository yang mewarisi class JpaRepository berfungsi untuk CRUD database seerta validasinya. Gunakan tag @Repository



6. Selanjutnya buat package baru dengan nama mahasiswa.service serta java class MahasiswaService.java untuk menerapkan logika dalam menangani dan memvalidasi data yang

diinsert, didelete, diupdate atau diread dari dan kedatabase menggunakan fungsi dari class repository yang telah dibuat sebelumnya @Service menandakan bahwa class ini adalah service

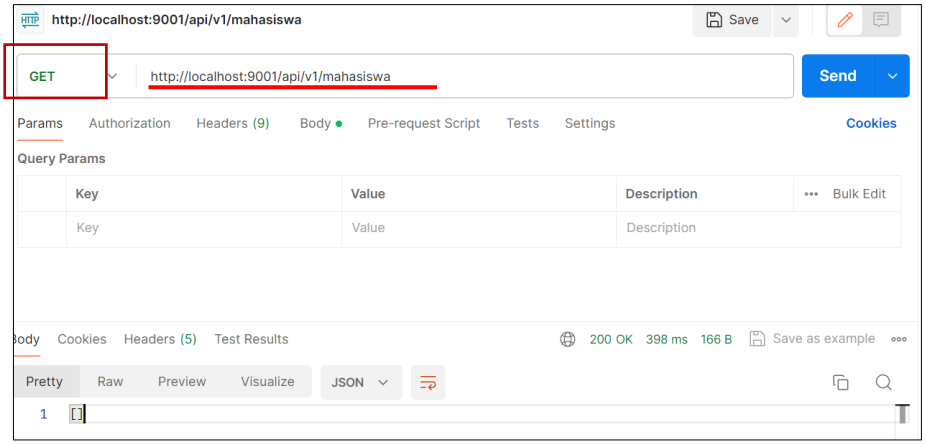
```
@Service
public class MahasiswaService {

    @Autowired
    private MahasiswaRepository mahasiswaRepository;

    public List<Mahasiswa> getAll() {
        return mahasiswaRepository.findAll();
    }

    public Mahasiswa getMahasiswa(Long id) {
        Optional<Mahasiswa> mahasiswaOptional = mahasiswaRepository.findById(id);
        return mahasiswaOptional.get();
    }

    public void insert(Mahasiswa mahasiswa) {
        Optional<Mahasiswa> mahasiswaOptional
                = mahasiswaRepository.findMahasiswaByEmail(email: mahasiswa.getEmail());
        if (mahasiswaOptional.isPresent()) {
            throw new IllegalStateException(s: "Email sudah ada");
        }
        mahasiswaRepository.save(entity: mahasiswa);
    }

    public void delete(Long id) {
        boolean ada = mahasiswaRepository.existsById(id);
        if (!ada) {
```

7. Terakhir membuat package controller dan java class controller dengan penanda @RestController untuk memanggil fungsi CRUD di service sesuai dengan methode HTTP yang digunakan @PostMapping, @PutMapping, @DeleteMapping atau @GetMapping serta menggunakan @RequestMapping untuk menentukan url yang akan digunakan untuk memanggil seluruh fungsi tersebut.

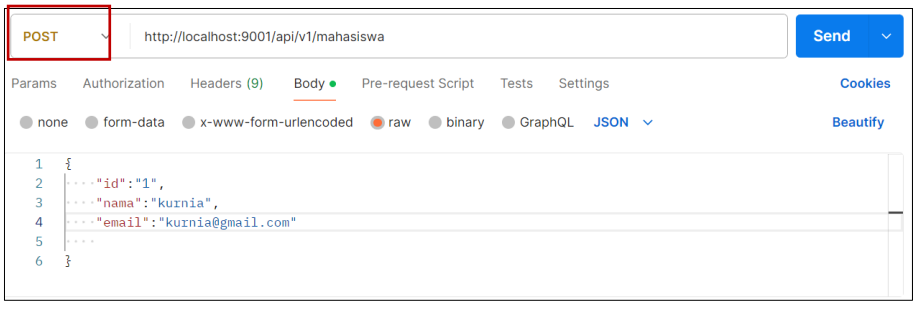
```
@RestController
@RequestMapping("api/v1/mahasiswa")
public class MahasiswaController {

    @Autowired
    private MahasiswaService mahasiswaService;

    @GetMapping
    public List<Mahasiswa> getAll(){
        return mahasiswaService.getAll();
    }

    @GetMapping(path = "{id}")
    public Mahasiswa getMahasiswa(@PathVariable("id") Long id){
        return mahasiswaService.getMahasiswa(id);
    }

    @PostMapping
    public void insert(@RequestBody Mahasiswa mahasiswa){
        mahasiswaService.insert(mahasiswa);
    }

    @DeleteMapping(path = "{id}")
    public void delete(@PathVariable("id") Long id){
        mahasiswaService.delete(id);
    }

    @PutMapping(path = "{id}")
    public void update(@PathVariable("id") Long id,
```

```
        @RequestParam(required = false) String nama,
        @RequestParam(required = false) String email) {
    mahasiswaService.update(id, nama, email);
}
```

8. Setelah langkah-langkah selesai, silakan eksekusi MahasiswaApplication di paket utama. Ini akan membuat tabel mahasiswa di database db_mahasiswa. Selanjutnya, jalankan Postman untuk menguji API yang telah dibuat dengan URL http://localhost:9001/api/v1/mahasiswa menggunakan metode **GET**. Aplikasi akan mengambil data dari tabel mahasiswa, tetapi karena belum ada data, respons yang dikembalikan akan berisi informasi yang sesuai.



9. Menggunakan metode POST untuk melakukan insert data di postman dengan url yang sama masuk ke tab body dan radio raw masukkan data dalam bentuk JSON untuk nama dan email. Ketika dikirim akan dikembalikan kosong tandanya insert berhasil Kembali ke metode GET untuk mengambil data.



10. Untuk mengupdate data gunakan metode PUT dan url yang sama ditambahkan dengan /id?namakolom=nilaibaru contohnya http://localhost:9001/api/v1/mahasiswa/2?nama=kurnia akan dikembalikan nilai kosong dan jika Kembali ke GET data akan berubah
11. Selanjutnya dapat ditambahkan untuk Matakuliah dan Nilai denagn cara yang sama.