LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Rekayasa Perangkat Lunak 2
Kelas	: 4IA07
Praktikum ke-	: 5
Tanggal	: 15 November 2025
Materi	: Spring Boot
NPM	: 51422084
Nama	: Muhammad Izzuddin Almansyur
Ketua Asisten	: Haikal Abizar
Jumlah Lembar	: 7 Halaman

LABORATORIUM INFORMATIKA

UNIVERSITAS GUNADARMA

2025

1. Jelaskan apa itu Spring Boot dan bagaimana perbedaannya dengan kode pada pertemuan sebelumnya. Apa keuntungan utama yang ditawarkan Spring Boot bagi pengembangan aplikasi?

Spring Boot adalah **framework berbasis Java** yang dibangun di atas **Spring Framework** dengan tujuan utama mempermudah proses pengembangan aplikasi. Spring Boot menyediakan konfigurasi otomatis (*auto-configuration*), *starter dependencies*, dan server bawaan sehingga developer dapat langsung fokus pada logika bisnis tanpa harus repot mengatur konfigurasi manual yang kompleks.

Pada pertemuan sebelumnya, fokus utama adalah **ORM** (misalnya Hibernate/JPA) yang digunakan untuk memetakan objek Java ke tabel database. Dengan ORM, developer tidak perlu menulis SQL secara manual karena operasi CRUD bisa dilakukan melalui objek. Namun, penggunaan ORM secara langsung biasanya masih membutuhkan banyak konfigurasi manual, seperti pengaturan koneksi database, `SessionFactory`, `EntityManager`, dan pengelolaan transaksi. Spring Boot hadir sebagai **lapisan di atas Spring Framework** yang menyederhanakan semua konfigurasi tersebut. Jika sebelumnya kita harus menulis banyak kode untuk mengatur ORM (Hibernate, JPA), maka dengan Spring Boot cukup menambahkan **starter dependency** seperti `spring-boot-starter-data-jpa`, lalu Spring Boot akan otomatis mengonfigurasi koneksi database, `EntityManager`, dan integrasi dengan ORM.

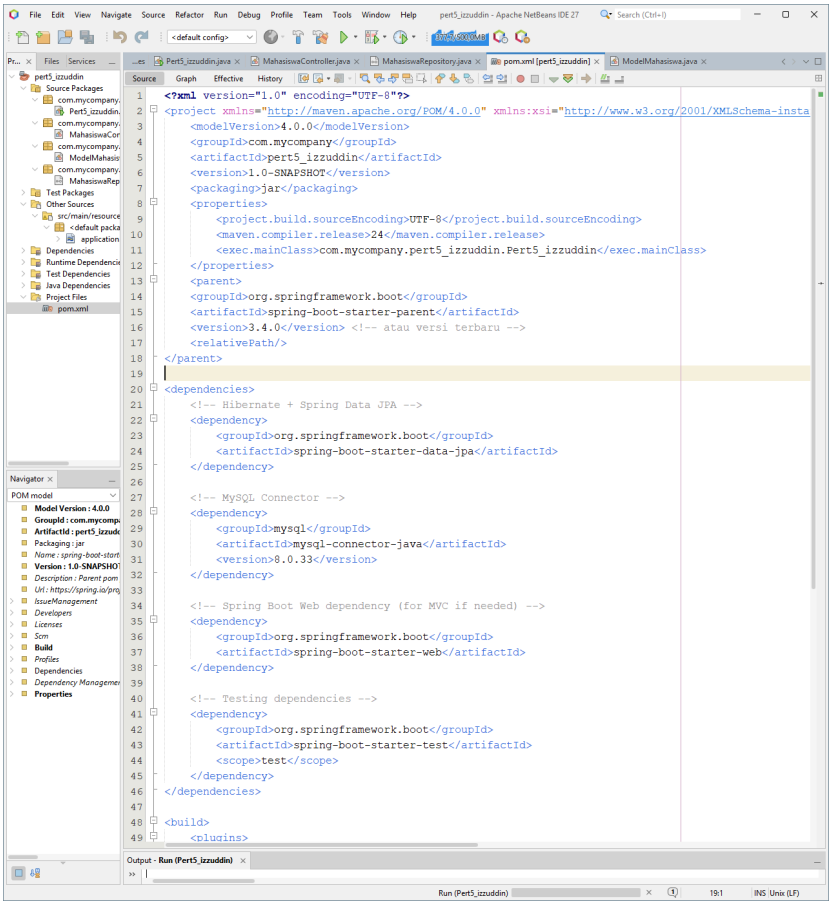
Spring Boot menawarkan keuntungan yang melengkapi penggunaan ORM:

1. **Konfigurasi Otomatis:** ORM seperti Hibernate langsung dikonfigurasi oleh Spring Boot tanpa setup manual.
2. **Produktivitas Tinggi:** Developer fokus pada logika bisnis, bukan konfigurasi teknis.
3. **Starter Dependencies:** Mempermudah integrasi ORM dengan database dan modul lain.
4. **Embedded Server:** Aplikasi bisa dijalankan mandiri tanpa perlu server eksternal.
5. **Microservices Ready:** Cocok untuk arsitektur modern karena ringan dan modular.

6. **Integrasi Ekosistem:** Mudah digabungkan dengan Spring Data JPA, REST API, dan Spring Security.

2. Jelaskan kode dan langkah-langkah program yang telah dibuat!

pom.xml



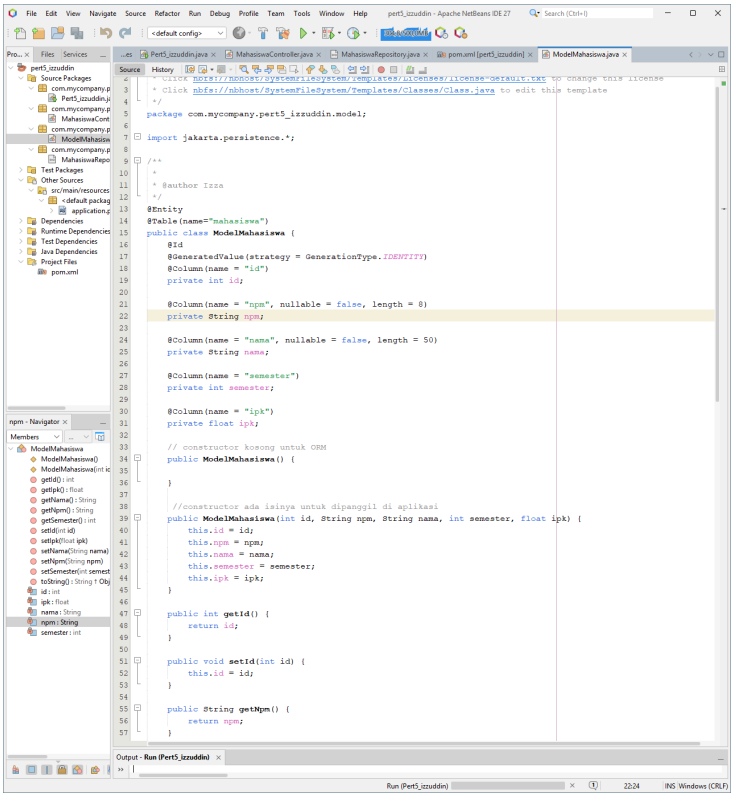
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>pert5_izzuddin</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.release>24</maven.compiler.release>
        <exec.mainClass>com.mycompany.pert5_izzuddin.Pert5_izzuddin</exec.mainClass>
    </properties>
    <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.0</version> <!-- atau versi terbaru -->
    <relativePath/>
</parent>

<dependencies>
    <!-- Hibernate + Spring Data JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- MySQL Connector -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>

    <!-- Spring Boot Web dependency (for MVC if needed) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

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

<build>
    <plugins>
```

File **pom.xml** adalah pusat konfigurasi ketika membuat proyek berbasis Maven di NetBeans. File ini mendefinisikan seluruh dependensi, versi Java, struktur project, serta plugin yang diperlukan agar aplikasi Spring Boot berjalan. Pada bagian awal terdapat deklarasi *groupId*, *artifactId*, dan *version* yang menandakan identitas project. Bagian penting lainnya yaitu deklarasi *parent* yang menggunakan **spring-boot-starter-parent**, di mana elemen ini memberikan konfigurasi default Spring Boot seperti versi library, manajemen dependency, dan plugin bawaan Spring.

Di dalam blok *dependencies* terdapat beberapa pustaka penting, yaitu **spring-boot-starter-data-jpa** yang menyediakan seluruh fitur Hibernate dan ORM, **mysql-connector-java** sebagai penghubung aplikasi ke database MySQL, dan **spring-boot-starter-web** jika sewaktu-waktu ingin menambahkan REST API atau

MVC. Ada juga *spring-boot-starter-test* untuk keperluan unit testing. Bagian akhir dari file ini adalah konfigurasi *build*, khususnya *spring-boot-maven-plugin* yang memungkinkan aplikasi dijalankan menggunakan perintah `mvn spring-boot:run` dan menghasilkan file jar siap running.

ModelMahasiswa.java

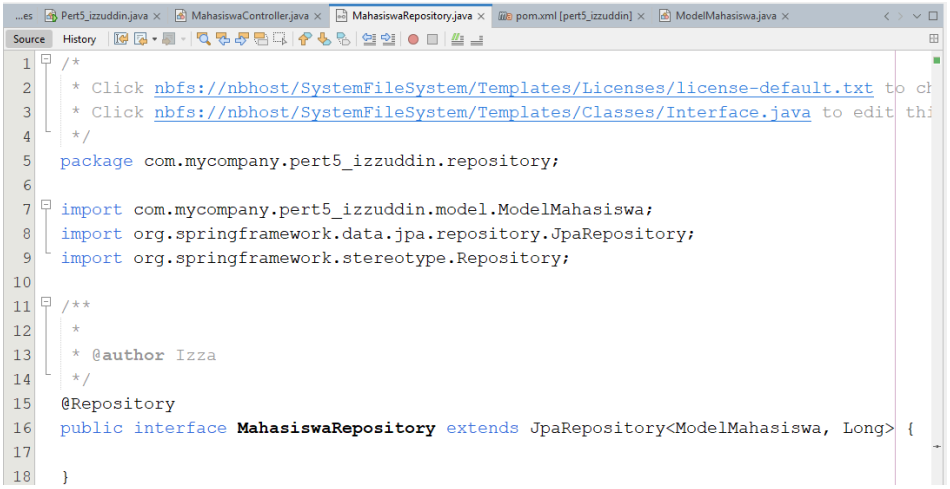


```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.pert5_izzuddin.model;

import jakarta.persistence.*;

/**
 *
 * @author Izza
 */
@Entity
@Table(name="mahasiswa")
public class ModelMahasiswa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "npm", nullable = false, length = 8)
    private String npm;

    @Column(name = "nama", nullable = false, length = 50)
    private String nama;

    @Column(name = "semester")
    private int semester;

    @Column(name = "ipk")
    private float ipk;

    // constructor kosong untuk ORM
    public ModelMahasiswa() {

    }

    //constructor ada isinya untuk dipanggil di aplikasi
    public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
        this.id = id;
        this.npm = npm;
        this.nama = nama;
        this.semester = semester;
        this.ipk = ipk;
    }

    public int getId() {
        return id;
    }

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

    public String getNpm() {
        return npm;
    }
```

Kelas ModelMahasiswa merupakan representasi tabel database menggunakan konsep ORM (Object Relational Mapping) dari Hibernate. Anotasi **@Entity** menunjukkan bahwa kelas ini adalah entitas yang akan dipetakan ke tabel database, sedangkan **@Table(name = "mahasiswa")** mengatur nama tabel dalam database MySQL. Variabel seperti *id*, *npm*, *nama*, *semester*, dan *ipk* dipetakan ke kolom tabel menggunakan anotasi **@Column**, dan kolom *id* dibuat otomatis bertambah dengan anotasi **@GeneratedValue(strategy = GenerationType.IDENTITY)**.

Di dalam kelas ini terdapat dua jenis konstruktor, yaitu konstruktor kosong yang diwajibkan oleh Hibernate untuk membuat objek secara otomatis dari database, serta konstruktor berparameter untuk mempermudah pembuatan objek secara manual ketika menambah data. Selain itu, terdapat getter dan setter yang digunakan Spring dan Hibernate untuk mengakses setiap field entitas. Method **toString()** digunakan agar data mahasiswa tampil rapi ketika dicetak ke konsol.

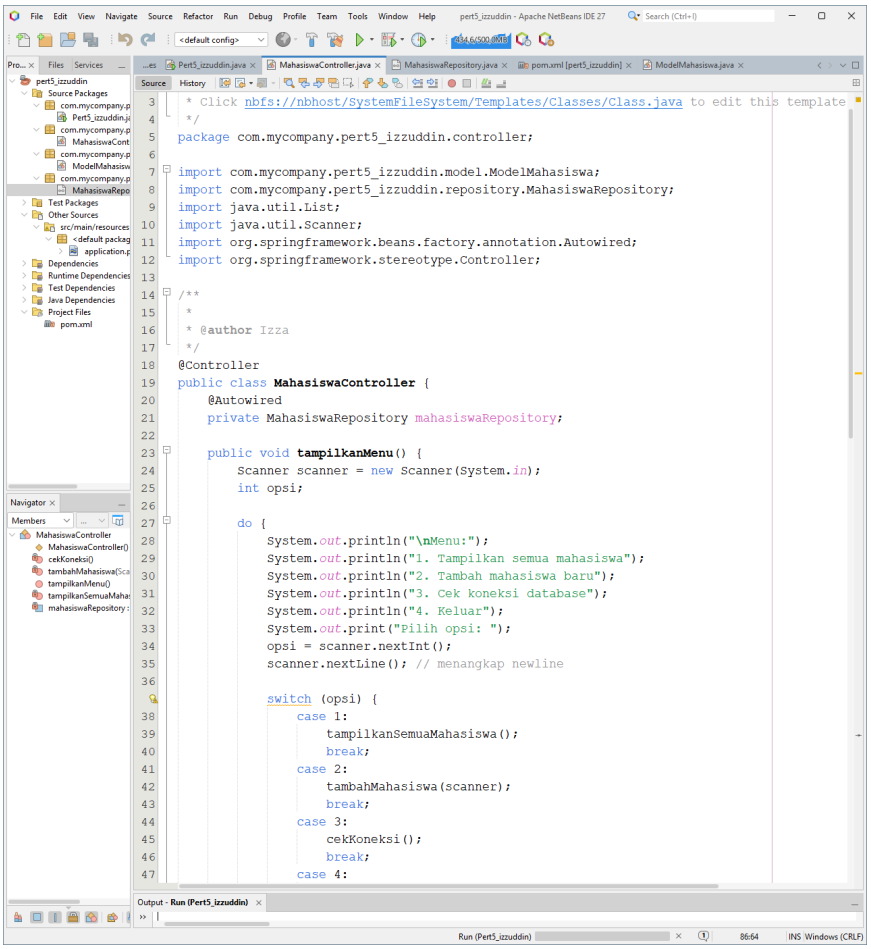
MahasiswaRepository.java



```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to ch
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit thi
 */
package com.mycompany.pert5_izzuddin.repository;

import com.mycompany.pert5_izzuddin.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 *
 * @author Izza
 */
@Repository
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {

}
```

File interface MahasiswaRepository berfungsi sebagai lapisan akses data (DAO) yang menghubungkan aplikasi dengan database tanpa perlu menulis query SQL secara manual. Interface ini diperluas dari **JpaRepository<ModelMahasiswa, Long>**, yang secara otomatis memberi fitur CRUD lengkap seperti findAll(), save(), deleteById(), dan findById() tanpa menuliskan satu pun query. Anotasi **@Repository** menunjukkan bahwa kelas ini dikelola oleh Spring sebagai komponen repository.

MahasiswaController.java

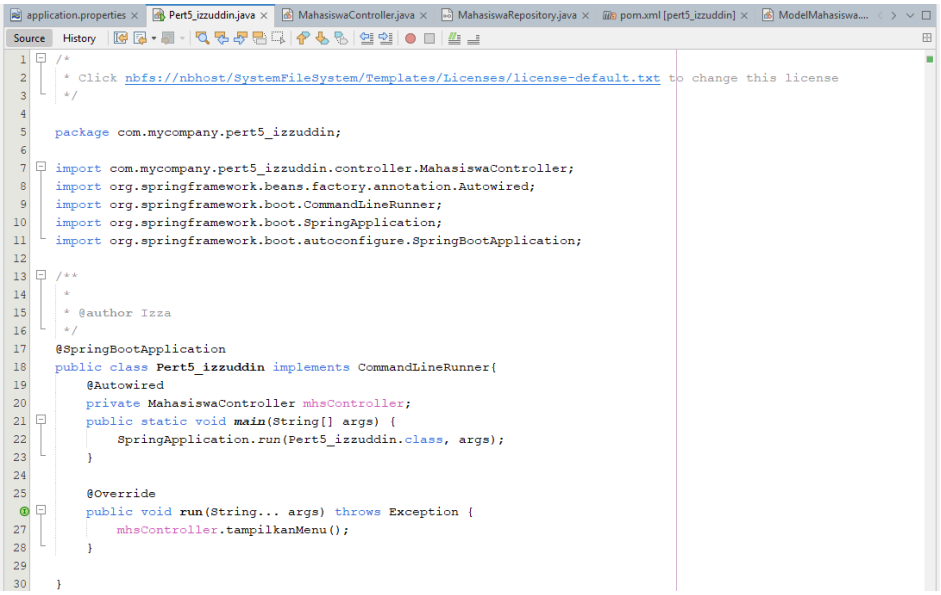


```java
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.pert5_izzuddin.controller;

import com.mycompany.pert5_izzuddin.model.ModelMahasiswa;
import com.mycompany.pert5_izzuddin.repository.MahasiswaRepository;
import java.util.List;
import java.util.Scanner;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

/**
 *
 * @author Izza
 */
@Controller
public class MahasiswaController {
    @Autowired
    private MahasiswaRepository mahasiswaRepository;

    public void tampilkanMenu() {
        Scanner scanner = new Scanner(System.in);
        int opsi;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Tampilkan semua mahasiswa");
            System.out.println("2. Tambah mahasiswa baru");
            System.out.println("3. Cek koneksi database");
            System.out.println("4. Keluar");
            System.out.print("Pilih opsi: ");
            opsi = scanner.nextInt();
            scanner.nextLine(); // menangkap newline

            switch (opsi) {
                case 1:
                    tampilkanSemuaMahasiswa();
                    break;
                case 2:
                    tambahMahasiswa(scanner);
                    break;
                case 3:
                    cekKoneksi();
                    break;
                case 4:
```

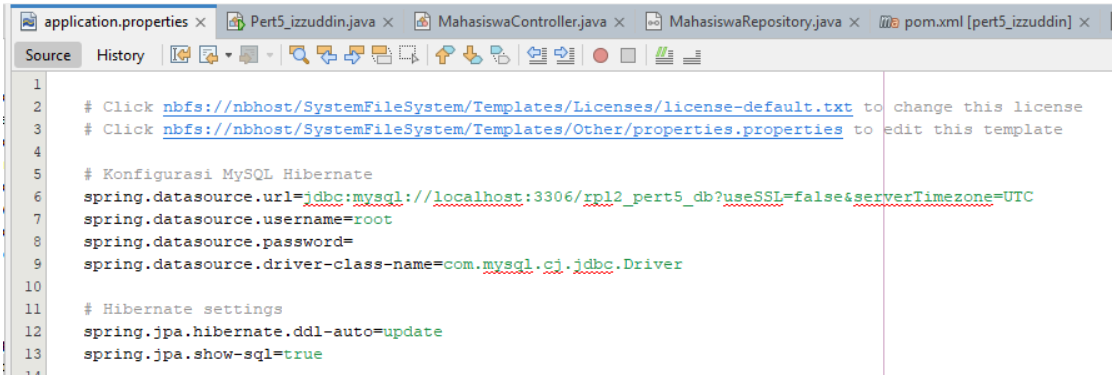
MahasiswaController bertanggung jawab mengatur logika aplikasi untuk menjalankan menu berbasis konsol. Controller ini di-inject dengan **@Autowired** agar Spring memberikan instance MahasiswaRepository secara otomatis. Metode tampilkanMenu() digunakan untuk menampilkan menu interaktif seperti menampilkan mahasiswa, menambah data, mengecek koneksi, dan keluar aplikasi. Pemrosesan input dilakukan menggunakan Scanner. Saat pengguna memilih opsi tertentu, controller memanggil metode lainnya seperti tampilkanSemuaMahasiswa() untuk mengambil data menggunakan repository, tambahMahasiswa() untuk menyimpan data baru, dan cekKoneksi() untuk memastikan database dapat diakses.

Pert5_izzuddin.java (main)

Kelas ini merupakan titik awal aplikasi Spring Boot. Anotasi **@SpringBootApplication** mengaktifkan auto-configuration Spring, component scanning, dan konfigurasi default lainnya. Di dalam method main(), perintah `SpringApplication.run()` digunakan untuk menjalankan container Spring Boot. Kelas ini juga mengimplementasikan **CommandLineRunner**, yang artinya metode run() akan otomatis dipanggil setelah konteks Spring selesai diinisialisasi. Dari sinilah controller menu dipanggil untuk menjalankan aplikasi berbasis teks.



application.propertis



File application.properties adalah pusat konfigurasi Spring Boot pada sisi database. Pada bagian datasource, aplikasi dihubungkan dengan server MySQL menggunakan URL **`jdbc:mysql://localhost:3306/rpl2_pert5_db`**, diikuti username dan password. Driver yang digunakan adalah **`com.mysql.cj.jdbc.Driver`**. Kemudian Spring JPA diatur agar Hibernate otomatis membuat atau memperbarui tabel sesuai entitas melalui **`spring.jpa.hibernate.ddl-auto=update.`** Properti **`spring.jpa.show-sql=true`** berfungsi menampilkan query SQL yang dijalankan pada konsol agar lebih mudah melakukan debugging.

Langkah - langkah pembuatan:

1. Pertama buka NetBeans lalu buat proyek baru dengan memilih menu *New Project*, kemudian pilih kategori Maven dan template *Java Application*. Setelah proyek terbentuk, buka file pom.xml dan ubah isinya menjadi konfigurasi Spring Boot dengan menambahkan parent Spring Boot, dependencies JPA, MySQL Connector, Web, dan plugin Spring Boot Maven seperti contoh yang sudah diberikan.
2. Setelah folder project siap, buat package baru bernama **`model`**. Di dalam package tersebut buat *Java Class* dengan nama ModelMahasiswa. Tambahkan anotasi JPA seperti @Entity dan @Table untuk memetakan kelas ke tabel database. Tambahkan variabel id, npm, nama, semester, dan ipk lalu beri anotasi @Column dan @Id sesuai kebutuhan. Tambahkan pula dua konstruktor (kosong dan berparameter), getter-setter, dan method toString.
3. Selanjutnya buat package **`controller`** dan tambahkan *Java Class* bernama MahasiswaController. Import model dan repository, lalu tambahkan anotasi @Controller dan injeksi @Autowired. Tuliskan logika menu menggunakan Scanner dan buat metode tampilkanMenu, tampilkanSemuaMahasiswa, tambahMahasiswa, dan cekKoneksi sesuai contoh yang sudah diberikan.
4. Selanjutnya buat package **`controller`** dan tambahkan *Java Class* bernama MahasiswaController. Import model dan repository, lalu tambahkan anotasi @Controller dan injeksi @Autowired. Tuliskan logika menu menggunakan Scanner dan buat metode tampilkanMenu, tampilkanSemuaMahasiswa, tambahMahasiswa, dan cekKoneksi sesuai contoh yang sudah diberikan.
5. Setelah controller selesai dibuat, buka file utama yang otomatis dibuat oleh NetBeans atau buat kelas baru dengan nama Pert5_izzuddin. Tambahkan anotasi @SpringBootApplication agar Spring Boot dapat melakukan scanning otomatis. Implementasikan CommandLineRunner lalu pada method run() panggil controller melalui mhsController.tampilkanMenu(). Pada method main, jalankan aplikasi menggunakan SpringApplication.run.
6. Terakhir buka folder **`src/main/resources`** lalu buat file baru dengan nama application.properties. Isikan konfigurasi koneksi database MySQL berupa datasource URL, username, password, driver, pengaturan Hibernate seperti ddl-auto dan show-sql. Dan program akan bisa di run.