LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Rekayasa Perangkat Lunak 2
Kelas	: 4IA06
Praktikum ke-	: 5
Tanggal	: 12 November 2024
Materi	: Konsep Framework Spring, Pembuatan project spring dan hibernate.
NPM	: 51421055
Nama	: Muhammad Ridho
Ketua Asisten	: Gilbert Jefferson Faozato Mendrofa
Paraf Asisten	:
Nama Asisten	:
Jumlah Lembar	: 17 Lembar

LABORATORIUM TEKNIK INFORMATIKA
UNIVERSITAS GUNADARMA
2024

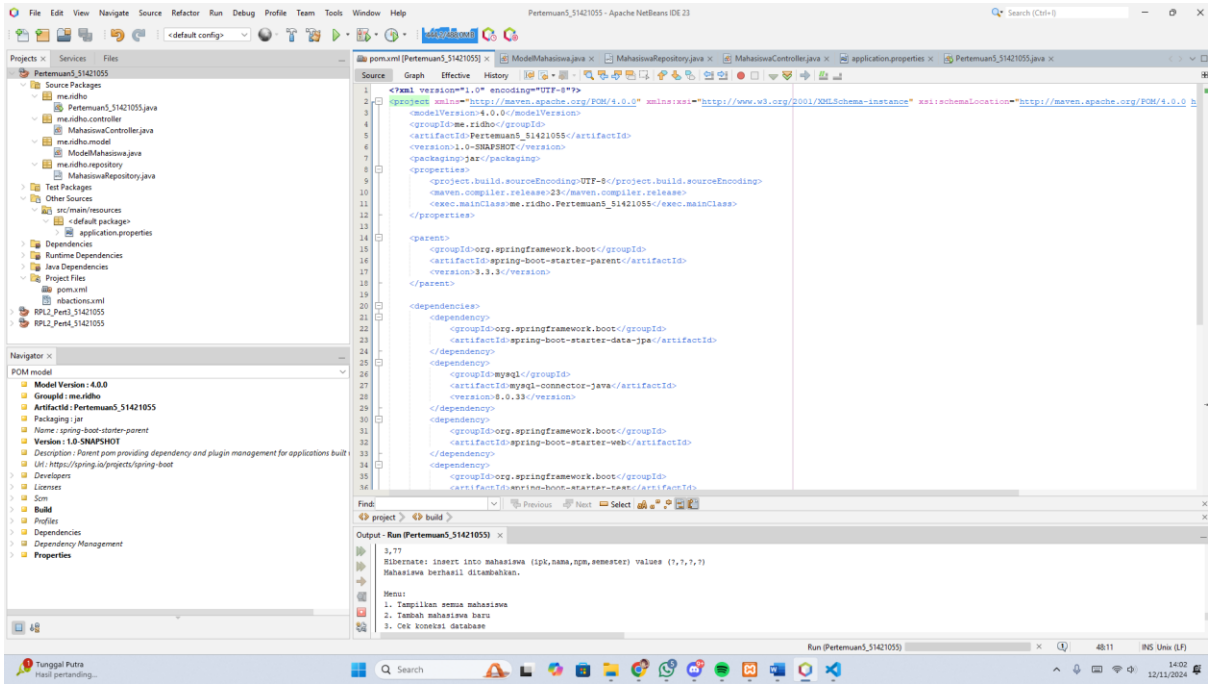
Soal:

1. Jelaskan apa itu SpringBoot dan bagaimana perbedaanya dengan kode pada pertemuan sebelumnya. Apa keuntungan utama yang ditawarkan SpringBoot bagi pengembang aplikasi.
2. Jelaskan kode dan Langkah-langkah program yang telah dibuat.

Jawaban:

1. Spring Boot adalah sebuah framework berbasis Java yang dirancang untuk mempermudah pengembangan aplikasi berbasis Spring. Spring Boot memudahkan pembuatan aplikasi berbasis Spring dengan mengurangi konfigurasi manual yang diperlukan, seperti pengaturan XML atau JavaConfig, dengan menyediakan konfigurasi otomatis (auto-configuration) yang memungkinkan aplikasi berjalan lebih cepat. Perbedaanya kalau ORM (Object-Relational Mapping) seperti Hibernate atau JPA merupakan teknik untuk memetakan objek-objek Java dengan tabel-tabel dalam database, sehingga pengembang dapat bekerja dengan data sebagai objek Java. Dalam pertemuan sebelumnya, ORM mungkin hanya berfokus pada interaksi dengan database. Sementara itu, Spring Boot adalah framework yang lebih besar yang menyediakan banyak fitur selain ORM, seperti sistem keamanan, REST API, MVC, hingga manajemen session, sehingga aplikasi lebih terstruktur dan komprehensif. Keuntungannya sendiri yakni Konfigurasi Otomatis (Auto-Configuration): Mengurangi waktu pengaturan yang kompleks, karena sebagian besar pengaturan dilakukan secara otomatis. Embedded Server: Mendukung embedded server seperti Tomcat, yang memudahkan developer untuk menjalankan aplikasi tanpa perlu setup server eksternal. Starter Dependencies: Mengelompokkan dependency sehingga memudahkan developer dalam menambahkan fitur seperti Spring Data, Spring Security, atau REST dengan konfigurasi minimal.
2.

Pom.xml



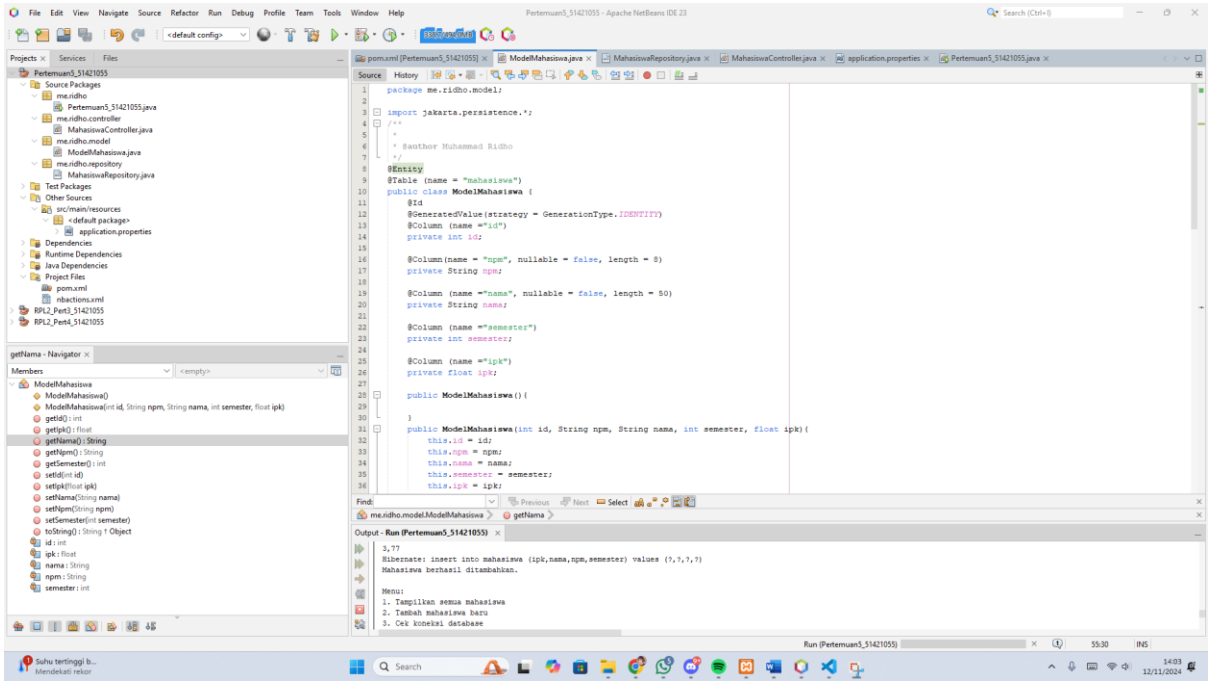
Sourcecode:

```
<?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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>me.ridho</groupId>
  <artifactId>Pertemuan5_51421055</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.release>23</maven.compiler.release>
    <exec.mainClass>me.ridho.Pertemuan5_51421055</exec.mainClass>
  </properties>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.3</version>
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
```

```
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
  </dependencies>

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

Penjelasan: Codingan ditas untuk mengatur dependencies dan konfigurasi springboot. Pada bagian dependencies yang ditambahkan terdapat library utama yang digunaka yakni spring-boot-starter-data-jpa untuk mengelola database dengan JPA, mysql-connector-java versi 8.0.33 untuk koneksi ke MySQL, spring-boot-starter-web untuk pengembangan aplikasi web, serta spring-boot-starter-test untuk keperluan testing. Pada bagian build, terdapat plugin spring-boot-maven-plugin yang memungkinkan tugas ACT ini dibangun atau dijalankan dengan integrasi maven

ModelMahasiswa.java



Sourcecode:

```
package me.ridho.model;


import jakarta.persistence.*;
/**
 *
 * @author Muhammad Ridho
 */
@Entity
@Table (name = "mahasiswa")
public class ModelMahasiswa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column (name ="id")
    private int id;
```

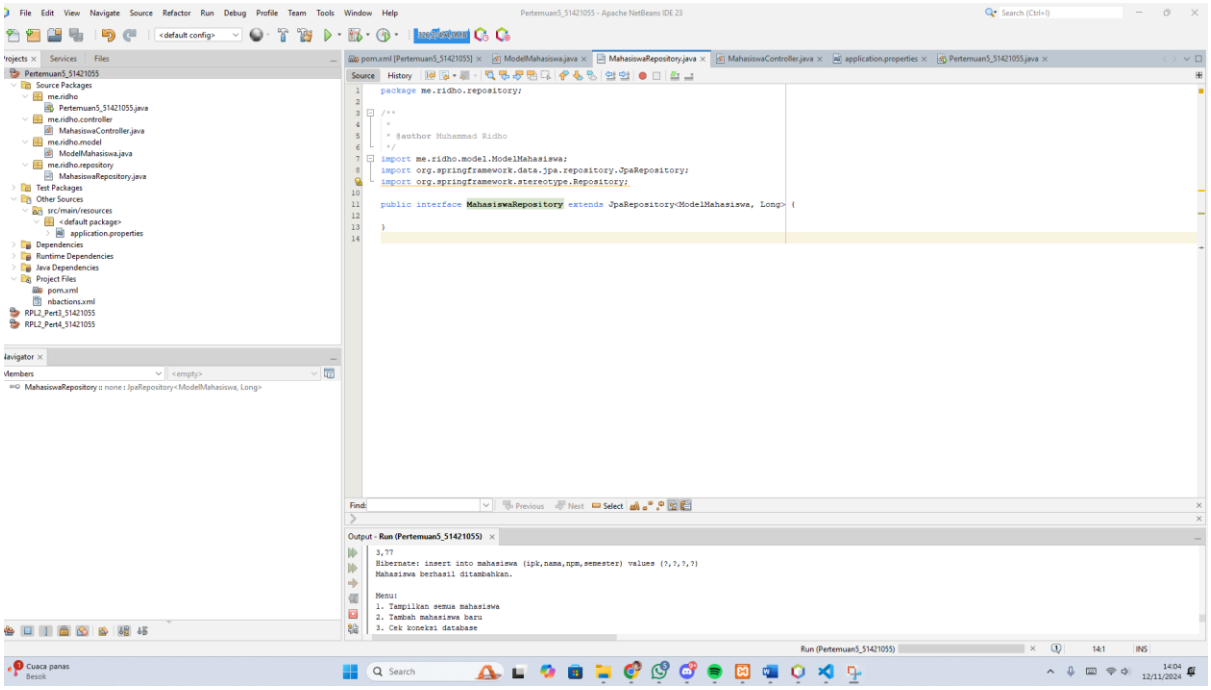
```
@Column(name = "npm", nullable = false, length = 8)
private String npm;

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

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

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

public ModelMahasiswa(){

}
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;
}

public void setNpm(String npm) {
    this.npm = npm;
}

public String getNama() {
    return nama;
}

public void setNama(String nama) {
    this.nama = nama;
}

public int getSemester() {
    return semester;
}

public void setSemester(int semester) {
    this.semester = semester;
}

public float getIpk() {
    return ipk;
}
```

```
    public void setIpk(float ipk) {
        this.ipk = ipk;
    }

    @Override
    public String toString(){
        return "Mahasiswa{" +
                "id=" + id +
                ", npm='" + npm + '\'' +
                ", nama='" + nama + '\'' +
                ", semester='" + semester + '\'' +
                ", ipk='" + ipk + '\'' +
                '}';
    }
}
```

Penjelasan: Codingan diatas terdapat Class *ModelMahasiswa* yang mempresentasikan entitas “mahasiswa” menggunakan framework Jakarta Persistence untuk mengelola data dalam table “mahasiswa” didatabase. Pada Class ini ada anotasi @Entity untuk menunjukkan class ini adalah entitas dari framework diatas, dan @Table untuk mengatur nama table dalam database. Atribut-atribut ini diatur dengan anotasi @Column yang menentukan nama kolom di database, serta pengaturan tambahan seperti panjang maksimum dan ketentuan nullable. Class ini memiliki konstruktor default dan konstruktor parameter untuk inisialisasi objek dengan nilai awal. Setter dan getter disediakan untuk tiap atribut guna mengakses dan mengubah data. Kelas ini juga mengoverride metode `toString()` untuk menghasilkan representasi string dari objek `ModelMahasiswa`, sehingga memudahkan saat melakukan debugging atau menampilkan data objek.

MahasiswaRepository.java

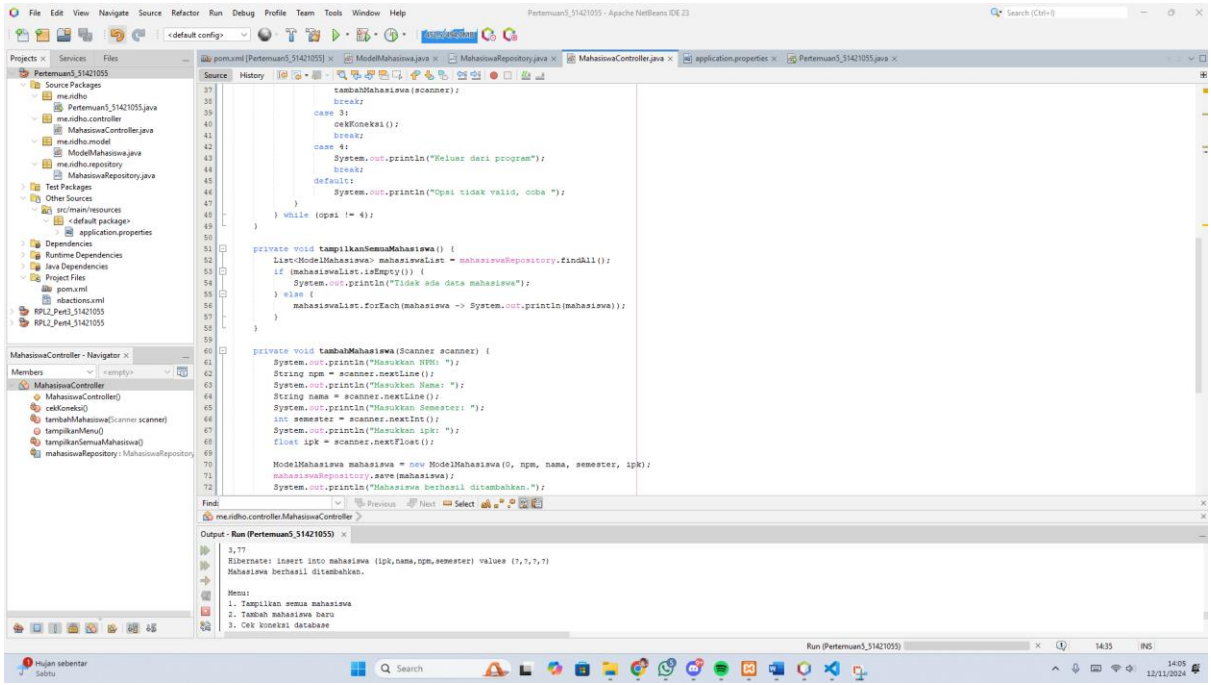


Sourcecode:

```
package me.ridho.repository;

/**
 *
 * @author Muhammad Ridho
 */
import me.ridho.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {


}
```

Penjalasan: Codingan diatas adalah bagian dari repository di Spring Data JPA untuk entitas *ModelMahasiswa*. Repository ini merupakan antarmuka (interface) yang berfungsi untuk melakukan operasi database pada entitas *ModelMahasiswa*. Dengan menggunakan JpaRepository, kita tidak perlu membuat implementasi metode CRUD secara manual karena semua operasi dasar (seperti save, delete, findAll, findById) telah disediakan oleh Spring Data JPA.

mahasiswaController.java



Sourcecode:

```
package me.ridho.controller;

import me.ridho.model.ModelMahasiswa;

import me.ridho.repository.MahasiswaRepository;

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

import org.springframework.stereotype.Controller;


import java.util.List;

import java.util.Scanner;

/**

 *

 * @author Muhammad Ridho

 */

@Controller

public class MahasiswaController {

    @Autowired

    private MahasiswaRepository mahasiswaRepository;
```

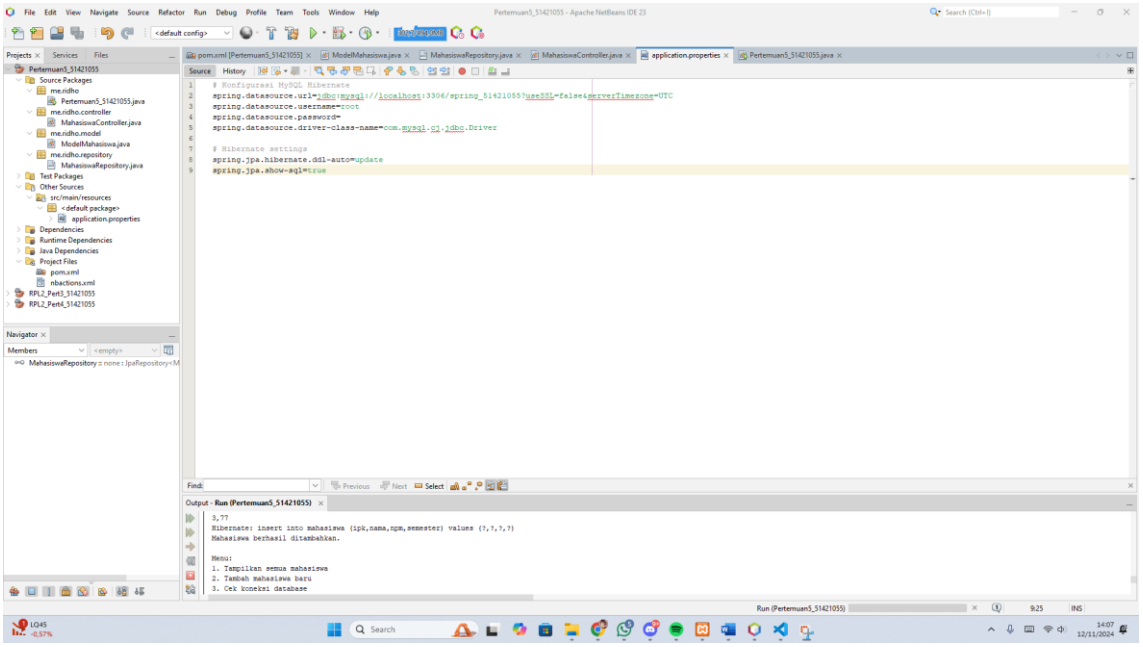
```
public void tampilkanMenu() {
    Scanner scanner = new Scanner(System.in);
    int opsi;

    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.println("Pilih Opsi: ");
        opsi = scanner.nextInt();
        scanner.nextLine();

        switch (opsi) {
            case 1:
                tampilkanSemuaMahasiswa();
                break;
            case 2:
                tambahMahasiswa(scanner);
                break;
            case 3:
                cekKoneksi();
                break;
            case 4:
                System.out.println("Keluar dari program");
                break;
            default:
```

```
                System.out.println("Opsi tidak valid, coba ");
            }
        } while (opsi != 4);
    }

    private void tampilkanSemuaMahasiswa() {
        List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
        if (mahasiswaList.isEmpty()) {
            System.out.println("Tidak ada data mahasiswa");
        } else {
            mahasiswaList.forEach(mahasiswa -> System.out.println(mahasiswa));
        }
    }

    private void tambahMahasiswa(Scanner scanner) {
        System.out.println("Masukkan NPM: ");
        String npm = scanner.nextLine();
        System.out.println("Masukkan Nama: ");
        String nama = scanner.nextLine();
        System.out.println("Masukkan Semester: ");
        int semester = scanner.nextInt();
        System.out.println("Masukkan ipk: ");
        float ipk = scanner.nextFloat();

        ModelMahasiswa mahasiswa = new ModelMahasiswa(0, npm, nama, semester, ipk);
        mahasiswaRepository.save(mahasiswa);
        System.out.println("Mahasiswa berhasil ditambahkan.");
    }
```

```
    private void cekKoneksi() {
        try {
            mahasiswaRepository.findAll();
            System.out.println("Koneksi ke database berhasil");
        } catch (Exception e) {
            System.out.println("Gagal terhubung ke database");
        }
    }
}
```

Penjelasan: Codingan diatas adalah sebuah controller untuk mengelola data mahasiswa menggunakan *MahasiswaRepository* sebagai akses data. Controller ini berfungsi untuk menampilkan menu untuk mengelola data mahasiswa. Terdapat Anotasi @Controller menunjukkan bahwa class ini adalah bagian dari komponen Controller dalam aplikasi Spring, yang berperan mengatur logika bisnis. @Autowired digunakan untuk melakukan injeksi dependensi dari MahasiswaRepository, yang memungkinkan kelas ini menggunakan mahasiswaRepository untuk berinteraksi dengan database. Method *tampilkanMenu()* Method ini menampilkan menu utama aplikasi yang memungkinkan pengguna memilih opsi untuk mengelola data mahasiswa. Dengan menggunakan Scanner, pengguna dapat memilih opsi 1. Tampilkan semua mahasiswa, 2. Tambah mahasiswa baru, 3. Cek koneksi database, dan 4. Keluar. Method *tampilkanSemuaMahasiswa()* Method ini mengambil semua data mahasiswa dari database menggunakan findAll() dari JpaRepository. Jika tidak ada data mahasiswa, akan mencetak pesan "Tidak ada data mahasiswa", sebaliknya, akan mencetak informasi tiap mahasiswa. Method *tambahMahasiswa()* Method ini menerima input pengguna untuk data mahasiswa baru. Setelah data dimasukkan, ModelMahasiswa dibuat, dan *mahasiswaRepository.save()* digunakan untuk menyimpan data mahasiswa ke database. Method *cekKoneksi()* Method ini memeriksa apakah aplikasi dapat terhubung ke database dengan mencoba mengambil semua data mahasiswa. Jika berhasil, akan menampilkan "Koneksi ke database berhasil". Jika gagal, akan mencetak pesan kesalahan.

application.properties



Sourcecode:

 # Konfigurasi MySQL Hibernate

spring.datasource.url=jdbc:mysql://localhost:3306/spring_51421055?useSSL=false&serverTimezone=UTC

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

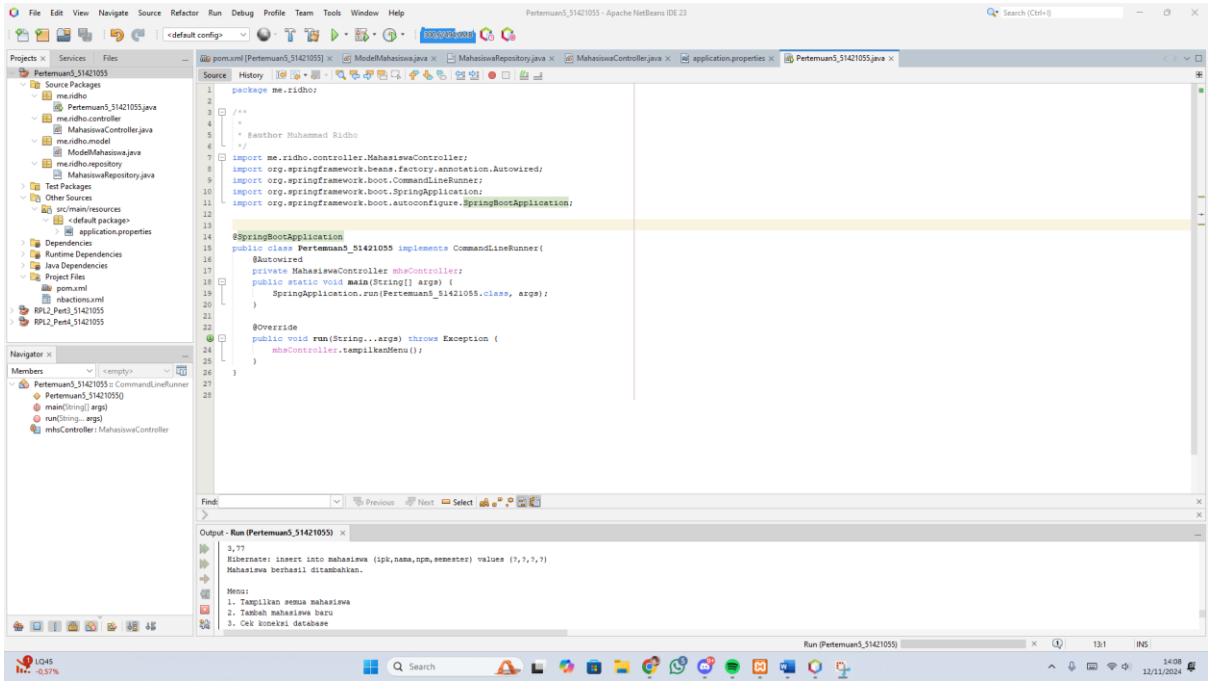
 # Hibernate settings

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Penjelasan: Codingan diatas berfungsi untuk menghubungkan Springboot ke database MySQL dengan menggunakan hibernate sebagai JPA Provider. spring.datasource.url menentukan URL koneksi JDBC ke database spring_51421055 di localhost, dengan useSSL=false (menonaktifkan SSL) dan serverTimezone=UTC untuk menjaga konsistensi zona waktu. Username dan password ditentukan dengan spring.datasource.username dan spring.datasource.password. Class driver MySQL (com.mysql.cj.jdbc.Driver) digunakan untuk koneksi, dan Hibernate dikonfigurasi untuk memperbarui skema database secara otomatis (spring.jpa.hibernate.ddl-auto=update). Pengaturan spring.jpa.show-sql=true menampilkan perintah SQL di konsol untuk memudahkan debugging.

Pertemuan5_51421055.java



Sourcecode:

```
package me.ridho;


/**
 *
 * @author Muhammad Ridho
 */
import me.ridho.controller.MahasiswaController;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class Pertemuan5_51421055 implements CommandLineRunner{
    @Autowired
```

```
    private MahasiswaController mhsController;
    public static void main(String[] args) {
        SpringApplication.run(Pertemuan5_51421055.class, args);
    }

    @Override
    public void run(String...args) throws Exception {
        mhsController.tampilkanMenu();
    }
}
```

Penjelasan: Codingan diatas adalah semacam bagian utama seperti mahasiswa view dipertemuan sebelumnya, disini terdapat Anotasi *@SpringBootApplication* Anotasi ini mengaktifkan konfigurasi otomatis Spring Boot, komponen pemindaian, dan mendefinisikan aplikasi sebagai aplikasi Spring Boot. *CommandLineRunner* implements untuk code dalam method run untuk pemanggilan setelah Springboot dimulai. Injeksi *MahasiswaController* i-inject menggunakan *@Autowired*, sehingga bisa langsung digunakan di class ini. Method main method titik masuk Springboot. Method run Ketika code dijalankan run akan dipanggil, dan *mhsController.tampilkanMenu()* akan menampilkan menu utama.

OUTPUT SOURCECODE: