



**PROJECT SERVICE MAHASISWA  
SISTEM TERDISTRIBUSI**

**SEMESTER V**

**DISUSUN OLEH  
JESICA SANDITIA PUTRI  
2111083014**

**JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNOLOGI REKAYASA PERANGKAT  
LUNAK  
POLITEKNIK NEGERI PADANG  
2023**

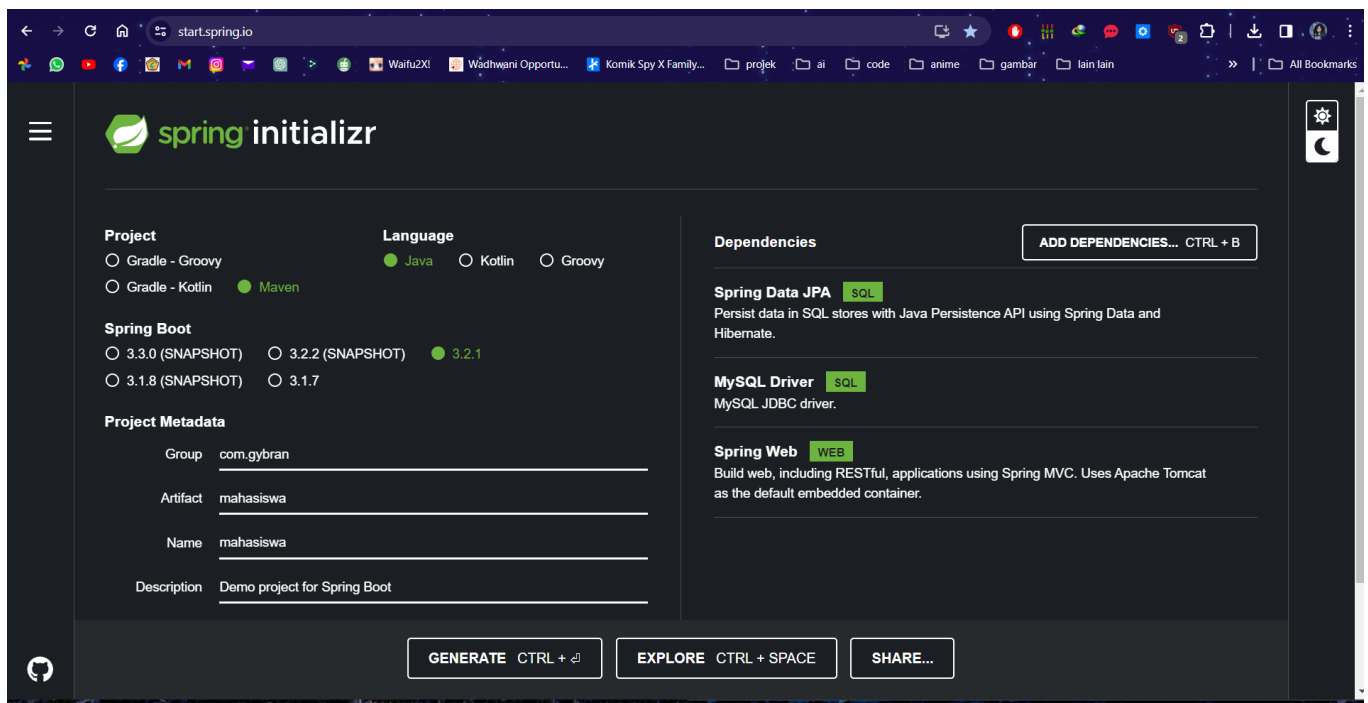
## A. Pengertian Spring Boot

salah satu framework dari spring yang memberikan kemudahan kita untuk memilih library java yang akan kita gunakan baik dari spring atau library lainnya. Dengan spring boot kita dapat dengan lebih cepat dan mudah untuk mengatur, mengkonfigurasi, dan menjalankan aplikasi berbasis spring. pada dasarnya spring boot adalah framework untuk menggabungkan dan membundel library-library spring dengan web server dan spring konfigurasi.

Meskipun terlihat sederhana tetapi selama saya menggunakan framework ini, saya sangat merasa sangat terbantu karena kita tidak perlu mencari daftar library, memilih, menggunakannya dan tidak perlu pusing akan *compatibility* antar library. Pekerjaan untuk menentukan library yang akan digunakan, dulu sebelum ada spring boot merupakan kerjaan wajib para Solution Architect atau minimal Senior Developer.

## B. Tool

Menggunakan layanan berbasis web yang memungkinkan pengguna untuk membuat proyek Spring Boot baru dengan cepat dengan dependensi yang diperlukan.



## C. Library

### 1. Spring Framework

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

Ini menunjukkan bahwa Spring Framework menggunakan fitur "Dependency Injection." Dengan kata lain, Spring akan secara otomatis menyuntikkan dependensi yang diperlukan ke dalam kelas ini, mengurangi ketergantungan manual pada objek lain.

- `org.springframework.stereotype.Service`

Dengan adanya anotasi ini, kelas tersebut diidentifikasi sebagai bagian dari layanan (service) dalam lingkungan Spring. Layanan ini dapat digunakan untuk mengelola logika bisnis atau tugas-tugas tertentu, dan Spring akan menyediakan fungsi-fungsi tambahan di sekitarnya.

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

Anotasi ini menandakan bahwa kelas tersebut berfungsi sebagai bagian dari lapisan kontrol (controller) dalam pengembangan aplikasi web menggunakan Spring. Ini berarti kelas ini akan menangani permintaan HTTP dan memberikan respons yang sesuai.

- `org.springframework.boot.SpringApplication`

Kelas ini adalah bagian dari Spring Boot, suatu kerangka kerja yang menyediakan utilitas untuk memulai dan mengelola aplikasi Spring Boot. `SpringApplication` membantu dalam mengatur dan menjalankan aplikasi dengan konfigurasi yang minimal.

- `org.springframework.boot.autoconfigure.SpringBootApplication`

Anotasi ini digunakan untuk menandai kelas utama aplikasi Spring Boot. Dengan menggunakan anotasi ini, menggabungkan tiga anotasi sekaligus: `@Configuration`, `@EnableAutoConfiguration`, dan `@ComponentScan`. Ini menyederhanakan konfigurasi dan memungkinkan Spring Boot secara otomatis menemukan dan mengonfigurasi komponen aplikasi.

## 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 menunjukkan bahwa metode tersebut menangani permintaan HTTP GET.

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

Anotasi yang menunjukkan bahwa metode tersebut menangani permintaan HTTP POST.

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

Anotasi yang menunjukkan bahwa metode tersebut menangani permintaan HTTP PUT.

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

Anotasi yang menunjukkan 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

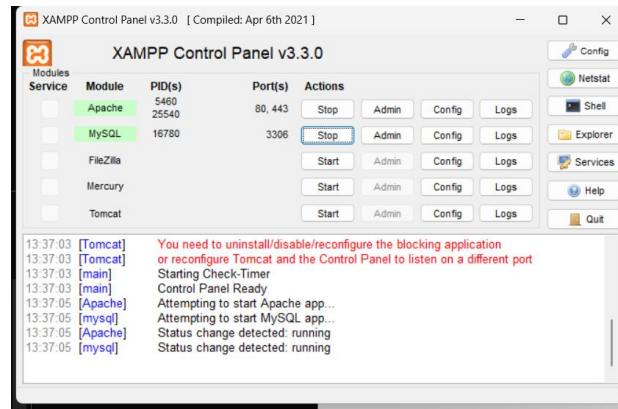
- `java.util.Optional`  
Kelas ini termasuk dalam Java Standard Library dan digunakan untuk menyatakan opsional nilai, yang sesuai dengan praktik baik dalam Java modern.
- `java.util.List`  
Kelas ini termasuk dalam Java Standard Library dan digunakan untuk merepresentasikan daftar objek.

#### 7. Java Persistence API (JPA) / Jakarta Persistence API

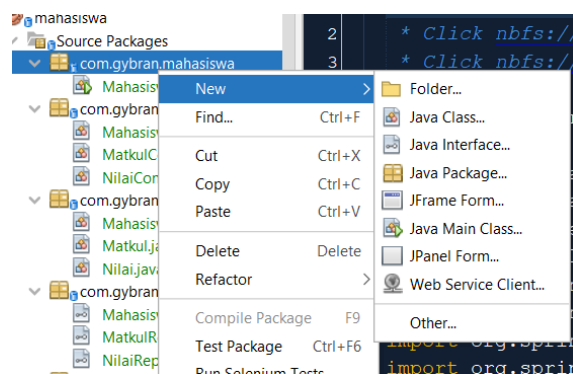
- `jakarta.persistence.Entity`  
Anotasi ini menandakan bahwa kelas tersebut adalah entitas JPA atau Jakarta Persistence API, yang dapat dipersistensi ke dalam database.
- `jakarta.persistence.GeneratedValue`  
Anotasi ini menandakan bahwa nilai dari suatu atribut (biasanya yang berfungsi sebagai kunci 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.

#### D. Langkah kerja

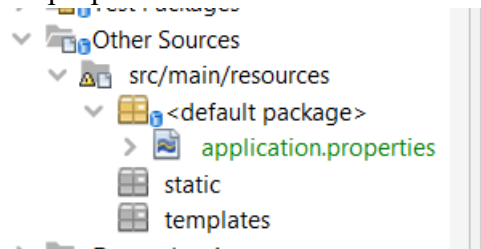
##### 1. Menhidupkan XAMPP



2. Membuat nama database dbmahasiswa pada mysql admin
3. Membuat new package entity, controller, repository, dan service dengan cara klik kanan pada “com.gybran.mahasiswa” kemudian klik new dan pilih java package

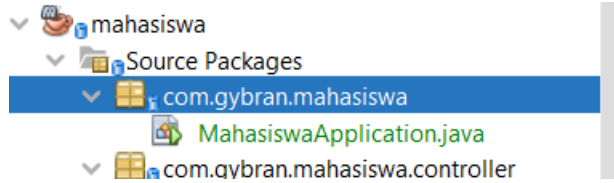


##### 4. Konfigurasi application properties



5. Membuat entity klik kanan pada package “com.gybran.mahasiswa.entity” kemudian klik new dan pilih java class
6. Membuat repository klik kanan pada package “com.gybran.mahasiswa.repository” kemudian klik new dan pilih java interface
7. Membuat Service klik kanan pada package “com.gybran.mahasiswa.service” kemudian klik new dan pilih java class
8. Membuat Controller klik kanan pada package “com.gybran.mahasiswa.controller” kemudian klik new dan pilih java class

9. Klik kanan pada MahasiswaApplication.java pilih run file maka otomatis framework akan running dan entity tadi akan tergenerate otomatis



10. Api yang tadi di buat bisa di test pada postman

## E. Penjelasan Kode Program

### 1. MahasiswaApplication.java

```
package com.gybran.mahasiswa;

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

@SpringBootApplication
public class MahasiswaApplication {

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

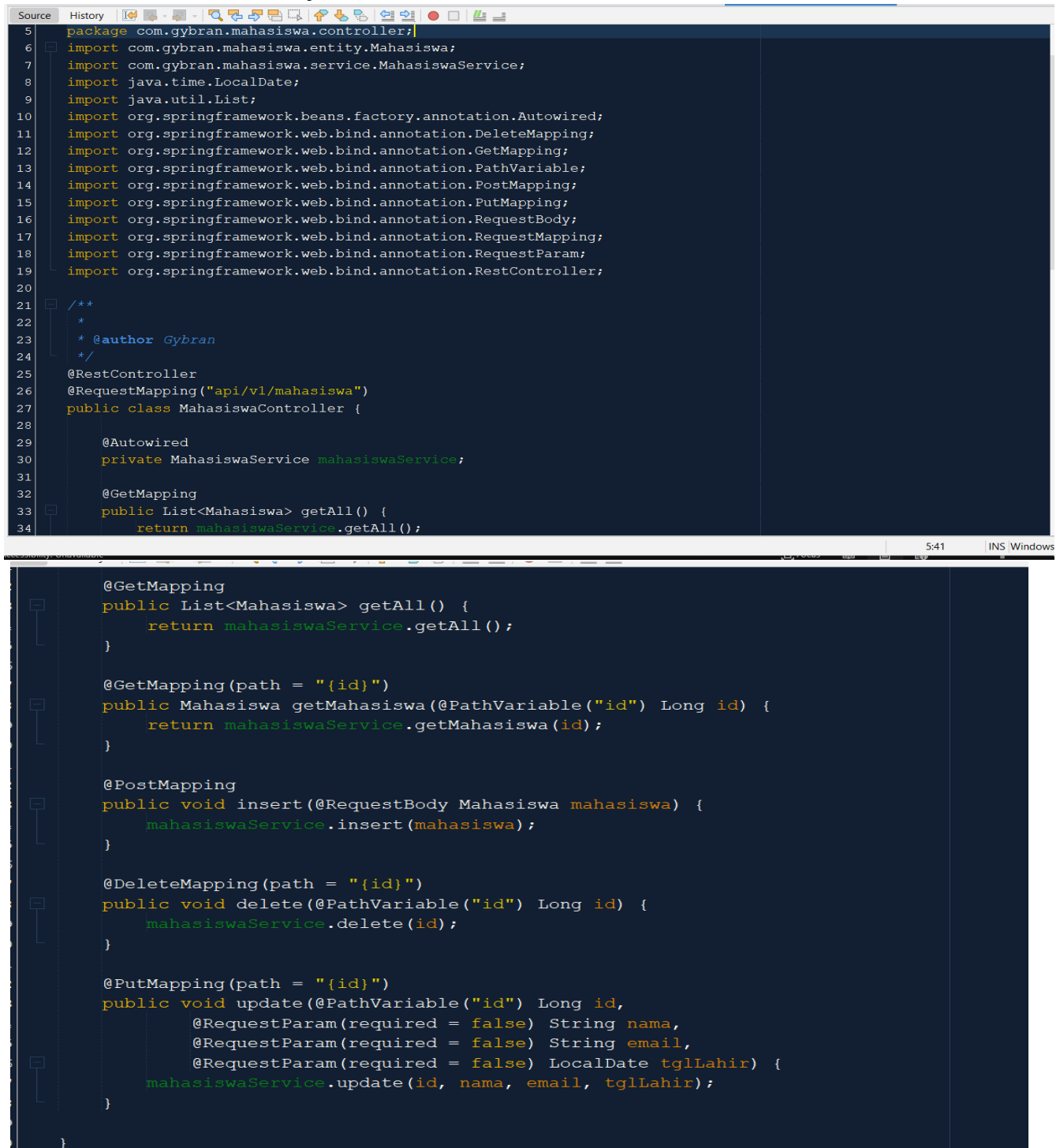
}
```

- package com.gybran.mahasiswa; kelas ini berada dalam package dengan nama "com.gybran.mahasiswa".
- import org.springframework.boot.SpringApplication; Mengimpor kelas SpringApplication dari paket org.springframework.boot.
- import org.springframework.boot.autoconfigure.SpringBootApplication; Mengimpor anotasi @SpringBootApplication dari paket org.springframework.boot.autoconfigure.
- - Anotasi @SpringBootApplication menggabungkan tiga anotasi utama, yakni @Configuration, @EnableAutoConfiguration, dan @ComponentScan. Dengan demikian, anotasi ini menunjukkan bahwa kelas tersebut berperan sebagai kelas konfigurasi dalam lingkup Spring Boot, mengaktifkan konfigurasi otomatis Spring Boot, dan melakukan pemindaian komponen untuk mengidentifikasi serta mendaftarkan komponen Spring dalam paket tersebut dan paket-paket yang terletak di bawahnya. Kelas ini merupakan kelas utama dari aplikasi Spring Boot. Nama kelas ini mengikuti konvensi dengan menambahkan kata "Application" pada nama proyek atau aplikasi.
- Metode main Metode ini merupakan titik masuk utama untuk menjalankan aplikasi. Dalam metode ini, SpringApplication.run(...) digunakan untuk memulai dan menjalankan aplikasi Spring Boot. Parameter pertama adalah kelas

utama (MahasiswaApplication.class), dan parameter kedua adalah argumen baris perintah (args).

- Dengan adanya anotasi `@SpringBootApplication`, Spring Boot secara otomatis akan mengkonfigurasi dan memulai aplikasi.

## 2. MahasiswaController.java



```
5 package com.gybran.mahasiswa.controller;
6 import com.gybran.mahasiswa.entity.Mahasiswa;
7 import com.gybran.mahasiswa.service.MahasiswaService;
8 import java.time.LocalDate;
9 import java.util.List;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.web.bind.annotation.DeleteMapping;
12 import org.springframework.web.bind.annotation.GetMapping;
13 import org.springframework.web.bind.annotation.PathVariable;
14 import org.springframework.web.bind.annotation.PostMapping;
15 import org.springframework.web.bind.annotation.PutMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RequestParam;
19 import org.springframework.web.bind.annotation.RestController;
20
21 /**
22  *
23  * @author Gybran
24  */
25 @RestController
26 @RequestMapping("api/v1/mahasiswa")
27 public class MahasiswaController {
28
29     @Autowired
30     private MahasiswaService mahasiswaService;
31
32     @GetMapping
33     public List<Mahasiswa> getAll() {
34         return mahasiswaService.getAll();
35
36     @GetMapping(path = "{id}")
37     public Mahasiswa getMahasiswa(@PathVariable("id") Long id) {
38         return mahasiswaService.getMahasiswa(id);
39     }
40
41     @PostMapping
42     public void insert(@RequestBody Mahasiswa mahasiswa) {
43         mahasiswaService.insert(mahasiswa);
44     }
45
46     @DeleteMapping(path = "{id}")
47     public void delete(@PathVariable("id") Long id) {
48         mahasiswaService.delete(id);
49     }
50
51     @PutMapping(path = "{id}")
52     public void update(@PathVariable("id") Long id,
53         @RequestParam(required = false) String nama,
54         @RequestParam(required = false) String email,
55         @RequestParam(required = false) LocalDate tglLahir) {
56         mahasiswaService.update(id, nama, email, tglLahir);
57     }
58 }
```

- `@RestController`; memberitahukan ke Spring bahwa class `MahasiswaApiController` merupakan sebuah controller.
- `@Autowired`; digunakan untuk menginstance dari interface `MahasiswaRepository`
- `@GetMapping("/api/mahasiswa")`; menggunakan method GET untuk mendapatkan data mahasiswa, `http://localhost:9001/api/mahasiswa`
- `@ResponseBody`; untuk memberikan response ke pemanggil menggunakan format JSON. Class `Pageable` berarti data yang disajikan bisa disajikan per halaman(page).
- `@PathVariable`; mekanisme yang digunakan untuk request menggunakan path. Misalkan `http://localhost:9001/api/mahasiswa/075410099`
- `@PostMapping("/api/mahasiswa")`; berarti method POST digunakan untuk menyimpan data/insert.
- `@RequestBody @Valid Mahasiswa mahasiswa`; berarti ketika akan melewatkan data pada sebuah body, `@Valid` untuk memvalidasi data ketika melakukan request. Misalkan sebuah nama tidak boleh null atau nama tidak boleh kosong.

### 3. Mahasiswa.java

```

package com.gybran.mahasiswa.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import java.time.LocalDate;

/**
 *
 * @author Gybran
 */
@Entity
@Table
public class Mahasiswa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String nama;
    private String email;
    private LocalDate tglLahir;

    public Mahasiswa() {
    }
}

```



```

public Mahasiswa() {
}

public Mahasiswa(long id, String nama, String email, LocalDate tglLahir) {
    this.id = id;
    this.nama = nama;
    this.email = email;
    this.tglLahir = tglLahir;
}

public long getId() {
    return id;
}

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

public String getNama() {
    return nama;
}

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

public String getEmail() {
    return email;
}

```

```

public void setEmail(String email) {
    this.email = email;
}

public LocalDate getTglLahir() {
    return tglLahir;
}

public void setTglLahir(LocalDate tglLahir) {
    this.tglLahir = tglLahir;
}

@Override
public String toString() {
    return "Mahasiswa{" + "id=" + id + ", nama=" + nama + ", email=" + email + ", tglLahir=" + tglLahir + '}';
}

```

- Import statements Mengimpor anotasi dan kelas-kelas dari paket jakarta.persistence yang digunakan untuk mendefinisikan entitas dan propertinya.
- @Entity Menandakan bahwa kelas ini adalah entitas JPA atau Jakarta Persistence API, yang dapat dipersistensi ke dalam database.
- @Table Anotasi ini dapat digunakan untuk menyesuaikan konfigurasi tabel database yang sesuai dengan entitas.
- @Id Menandakan bahwa properti di bawahnya (id) adalah identitas utama (primary key) dari entitas.

- @GeneratedValue(strategy = GenerationType.IDENTITY) Menentukan bahwa nilai identitas utama akan dihasilkan secara otomatis oleh database dan mengikuti strategi identitas.
- private Long id; Properti yang mewakili identitas utama (primary key).
- private String nama; Properti yang mewakili nama mahasiswa.
- private String email; Properti yang mewakili alamat email mahasiswa.
- private String tgllahir; Properti yang mewakili tanggal lahir mahasiswa.
- Konstruktor default (public Mahasiswa() {}) dan konstruktor parameterized (public Mahasiswa(Long id, String nama, String email) {}) untuk inisialisasi objek.
- Metode-metode setter (setId, setName, setEmail) dan getter (getId, getName, getEmail) untuk mengakses dan mengubah nilai properti.
- Override metode toString untuk memberikan representasi string dari objek Mahasiswa. Digunakan untuk debugging atau keperluan log.

#### 4. MahasiswaRepository.java

```

package com.gybran.mahasiswa.repository;

import com.gybran.mahasiswa.entity.Mahasiswa;
import java.util.List;
import java.util.Optional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 *
 * @author Gybran
 */
@Repository
public interface MahasiswaRepository extends JpaRepository<Mahasiswa, Long>{

    public Optional<Mahasiswa> findMahasiswaByEmail(String email);

}

```

- Import statements Mengimpor beberapa kelas yang diperlukan, termasuk Optional dari java.util dan beberapa kelas dari Spring Data JPA.
- @Repository Anotasi yang menandakan bahwa kelas ini adalah komponen Spring dan akan diidentifikasi sebagai bean yang dapat dielusidasi secara otomatis oleh Spring Framework.
- public interface MahasiswaRepository extends JpaRepository<Mahasiswa, Long> antarmuka yang menggambarkan repositori untuk entitas Mahasiswa. Antarmuka ini menetapkan bahwa entitas adalah Mahasiswa dan kunci utamanya adalah Long.

- public Optional<Mahasiswa> findMahasiswaByEmail(String email) menetapkan metode khusus untuk mencari entitas Mahasiswa berdasarkan alamat email. Mengembalikan objek

## 5. MahasiswaService.java

```
    */
    package com.gybran.mahasiswa.service;

    import com.gybran.mahasiswa.entity.Mahasiswa;
    import com.gybran.mahasiswa.repository.MahasiswaRepository;
    import jakarta.transaction.Transactional;
    import java.time.LocalDate;
    import java.util.List;
    import java.util.Objects;
    import java.util.Optional;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Service;

    /**
     *
     * @author Gybran
     */
    @Service
    public class MahasiswaService {

        @Autowired
        private MahasiswaRepository mahasiswaRepository;

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

        public void insert(Mahasiswa mahasiswa) {
            Optional<Mahasiswa> mahasiswaOptional = mahasiswaRepository.findMahasiswaByEmail(mahasiswa.getEmail());

            if (mahasiswaOptional.isPresent()) {
                throw new IllegalStateException("Email Sudah Ada");
            }

            mahasiswaRepository.save(mahasiswa);
        }

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

        public void delete(Long id) {
            boolean ada = mahasiswaRepository.existsById(id);
            if (!ada) {
                throw new IllegalStateException("Mahasiswa ini tidak ada");
            }
            mahasiswaRepository.deleteById(id);
        }
    }
```

```

@Transactional
public void update(Long id, String nama, String email, LocalDate tglLahir) {
    Mahasiswa mahasiswa = mahasiswaRepository.findById(id)
        .orElseThrow(() -> new IllegalStateException("Mahasiswa tidak ada"));

    if (nama != null && nama.length() > 0 && !Objects.equals(mahasiswa.getNama(), nama)) {
        mahasiswa.setNama(nama);
    }

    if (email != null && email.length() > 0 && !Objects.equals(mahasiswa.getEmail(), email)) {
        mahasiswa.setEmail(email);
    }

    if (tglLahir != null && !Objects.equals(mahasiswa.getTglLahir(), tglLahir)) {
        mahasiswa.setTglLahir(tglLahir);
    }
}
}

```

- Import statements mengimpor beberapa kelas yang diperlukan, termasuk List, Objects, Optional, dan anotasi @Autowired, @Service, dan @Transactional dari Spring Framework.
- @Service Anotasi yang menandakan bahwa kelas ini adalah komponen layanan Spring, yang akan dielusidasi secara otomatis dan dapat digunakan dalam konteks aplikasi Spring.
- public class MahasiswaService Deklarasi kelas MahasiswaService.
- @Autowired private MahasiswaRepository mahasiswaRepository;  
Menggunakan fitur Dependency Injection Spring untuk menyuntikkan instance MahasiswaRepository ke dalam kelas ini.
- public List<Mahasiswa> getAll() { ... } untuk mendapatkan daftar semua mahasiswa.
- public Mahasiswa getMahasiswa(Long id) { ... } untuk mendapatkan informasi mahasiswa berdasarkan ID.
- public void insert(Mahasiswa mahasiswa) { ... } untuk menambahkan mahasiswa baru.
- public void delete(Long id) { ... } untuk menghapus mahasiswa berdasarkan ID.
- @Transactional public void update(Long id, String nama, String email, LocalDate tglLahir) { ... } untuk memperbarui informasi mahasiswa berdasarkan ID. Anotasi @Transactional menandakan bahwa metode ini akan dijalankan dalam sebuah transaksi.
- Dalam metode insert, dilihat apakah email siswa sudah ada; jika sudah ada, berikan pengecualian.
- Dalam metode delete, dilihat apakah siswa dengan ID yang diberikan ada; jika tidak ada, berikan pengecualian.
- Untuk menghindari masalah konsistensi data, metode update menggunakan transaksi. Dilakukan pemeriksaan untuk memastikan bahwa nama dan email

yang diupdate benar, dan bahwa email yang baru dikirim tidak bertentangan dengan email mahasiswa lain.

## 6. application.properties

```
server.port=9001
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/dbmahasiswa
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql: true
```

- server.port=9001 Menentukan port tempat aplikasi Spring Boot akan dijalankan. Dalam kasus ini, aplikasi akan dijalankan pada port 9001.
- spring.jpa.hibernate.ddl-auto=update; memastikan struktur yang terdapat pada entitas sama dengan struktur di database, jika beda akan ada error.
- spring.jpa.show-sql=true; untuk menampilkan query ketika terjadi operasi di dalam database
- spring.jpa.properties.hibernate.format\_sql=true; agar format query yang tampil di output lebih terformat.
- server.port=8081; port yang digunakan oleh tomcat, defaultnya adalah 8080.
- spring.jackson.serialization.indent\_output=true; digunakan untuk merapikan tampilan output JSON
- spring.jpa.show-sql: true Menentukan apakah Hibernate akan menampilkan pernyataan SQL yang dihasilkan di konsol atau tidak. Dengan nilai true, pernyataan SQL akan ditampilkan.