LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: RPL 2
Kelas	: 4IA01
Praktikum ke-	: 6
Tanggal	: 18 November 2025
Materi	: AOP
NPM	: 50422657
Nama	: Harry Mardika
Ketua Asisten	: Satya Bara Justitia
Paraf Asisten	: -
Nama Asisten	: 1.
	2.
	3.
Jumlah Lembar	: 11 Lembar

LABORATORIUM INFORMATIKA

UNIVERSITAS GUNADARMA

2025

Soal

1. Jelaskan kode, langkah – langkah, dan output program yang telah dibuat!

Penjelasan

---

1. PENDAHULUAN

Proyek ini merupakan implementasi aplikasi desktop untuk manajemen data mahasiswa yang mengintegrasikan framework Spring Boot sebagai backend dengan Swing sebagai antarmuka pengguna (GUI). Aplikasi ini mendemonstrasikan penerapan pola arsitektur Model-View-Controller (MVC) dalam pengembangan aplikasi berbasis Java dengan memanfaatkan teknologi persistensi data melalui Spring Data JPA dan Hibernate. Tujuan utama dari praktikum ini adalah untuk memahami bagaimana mengintegrasikan multiple framework dalam satu aplikasi, mengelola dependensi melalui Maven, serta mengimplementasikan operasi CRUD (Create, Read, Update, Delete) pada data mahasiswa dengan interface pengguna yang user-friendly.

2. ARSITEKTUR DAN STRUKTUR PROYEK

2.1 Penjelasan Struktur Folder Proyek

Proyek ini mengorganisir kode berdasarkan pola MVC yang terpisah dalam berbagai package untuk memastikan pemisahan kepedulian (separation of concerns). Folder java menampung semua source code aplikasi yang diorganisir ke dalam beberapa subdirectory. Package com.mycompany.pertemuan6.model menyimpan kelas-kelas yang merepresentasikan struktur data aplikasi. Package com.mycompany.pertemuan6.repository menangani komunikasi dengan basis data melalui antarmuka penyimpanan data. Package com.mycompany.pertemuan6.service mengandung logika bisnis aplikasi yang menghubungkan layer repository dengan controller. Package com.mycompany.pertemuan6.controller bertugas sebagai penghubung antara view dan service untuk menangani instruksi dari antarmuka pengguna. Terakhir, package com.mycompany.pertemuan6.view mengandung komponen GUI yang membangun antarmuka visual untuk interaksi pengguna. Struktur ini memastikan bahwa setiap komponen memiliki tanggung jawab yang jelas dan mudah untuk dipelihara atau diperluas di masa depan.

2.2 Teknologi dan Dependensi yang Digunakan

Proyek ini menggunakan Spring Boot versi 3.3.3 sebagai framework utama yang menyediakan otomasi konfigurasi dan kemudahan dalam setup proyek. Spring Data JPA digunakan untuk abstraksi akses data dan menyederhanakan operasi terhadap basis data dengan menggunakan query methods. Hibernate berfungsi sebagai ORM (Object-Relational Mapping) yang menghubungkan antara object Java dan tabel-tabel di basis data. MySQL dipilih sebagai sistem manajemen basis data relasional dengan connector versi 8.0.33 untuk memastikan kompatibilitas dengan versi terbaru. Lombok adalah library yang mempermudah pembuatan kode dengan automatic generation dari getter, setter, constructor, dan method-method umum lainnya melalui anotasi. Terakhir, Swing digunakan sebagai framework untuk membangun GUI desktop yang responsif dan dapat dikustomisasi dengan mudah. Kombinasi teknologi ini menciptakan stack yang modern dan robust untuk pengembangan aplikasi desktop yang terintegrasi dengan basis data.

3. PENJELASAN KODE SETIAP KOMPONEN

3.1 Model Layer (ModelMahasiswa.java)

Kelas ModelMahasiswa merupakan representasi entity dari tabel mahasiswa di basis data yang juga berfungsi sebagai POJO (Plain Old Java Object) untuk menampung data dalam aplikasi. Kelas ini dianotasi dengan @Entity untuk menandakan bahwa kelas ini adalah entity JPA yang akan dipetakan ke tabel di basis data. Setiap properti dalam kelas ini merepresentasikan satu kolom dalam tabel: properti id berfungsi sebagai kunci utama (primary key) dengan strategi auto-increment menggunakan anotasi @GeneratedValue(strategy = GenerationType.IDENTITY), properti npm menyimpan nomor pokok mahasiswa sebagai identitas unik mahasiswa, properti nama menyimpan nama lengkap mahasiswa, properti semester mencatat semester yang sedang ditempuh mahasiswa dengan tipe data integer, dan properti ipk menyimpan nilai indeks prestasi kumulatif dengan tipe data float. Kelas ini menggunakan Lombok annotations yaitu @Data untuk menggenerate getter, setter, equals(), hashCode(), dan toString(), @NoArgsConstructor untuk membuat constructor tanpa parameter, dan @AllArgsConstructor untuk membuat constructor dengan semua parameter. Selain itu, terdapat constructor tambahan yang menerima parameter tanpa id untuk memudahkan pembuatan objek baru sebelum disimpan ke basis data. Desain ini memastikan bahwa data mahasiswa dapat dengan mudah dienkapsulasi, ditransmisikan, dan dipersistensikan ke dalam basis data.

3.2 Model Layer (ModelTabelMahasiswa.java)

Kelas ModelTabelMahasiswa merupakan custom table model yang meng-extend AbstractTableModel dari Swing untuk mengadaptasi data List of ModelMahasiswa agar dapat ditampilkan di dalam JTable. Tujuan dari kelas ini adalah untuk membuat bridge antara struktur data Java List dengan model yang diharapkan oleh komponen JTable. Kelas ini memiliki field mahasiswaList yang menyimpan referensi ke list mahasiswa dan field array columnNames yang mendefinisikan nama-nama kolom yang akan ditampilkan yaitu ID, NPM, Nama, Semester, dan IPK. Implementasi method getRowCount() mengembalikan jumlah baris sesuai dengan ukuran list mahasiswa, sehingga JTable tahu berapa banyak data yang harus ditampilkan. Method getColumnCount() mengembalikan jumlah kolom yang sama dengan panjang array columnNames. Method getValueAt() adalah yang paling penting karena ia mengambil data dari object ModelMahasiswa pada baris tertentu dan mengembalikan nilai yang sesuai dengan kolom yang diminta melalui switch statement. Method getColumnName() mengembalikan nama kolom yang sesuai dengan indeks yang diminta untuk ditampilkan di header tabel. Method isCellEditable() di-override untuk mengembalikan false pada semua sel, berarti data di dalam tabel tidak dapat diedit secara langsung dari GUI. Method setMahasiswaList() memungkinkan update data dan memanggil fireTableDataChanged() untuk memberitahu JTable bahwa data telah berubah sehingga tabel akan merefresh tampilan. Implementasi kelas ini memastikan bahwa data dari basis data dapat ditampilkan dengan rapi dan terstruktur dalam format tabel.

3.3 Repository Layer (MahasiswaRepository.java)

Kelas MahasiswaRepository merupakan antarmuka (interface) yang di-extend dari JpaRepository<ModelMahasiswa, Integer> untuk menyediakan operasi database standar tanpa perlu menulis query SQL secara manual. Anotasi @Repository menandakan bahwa interface

ini adalah Spring component untuk layer akses data. Dengan meng-extend JpaRepository, interface ini secara otomatis mewarisi berbagai method standar seperti save() untuk menyimpan data baru atau mengupdate data yang sudah ada, findById() untuk mencari data berdasarkan id, deleteById() untuk menghapus data berdasarkan id, dan findAll() untuk mengambil semua data dari tabel. Spring akan secara otomatis mengimplementasikan interface ini saat runtime melalui mekanisme proxy, sehingga developer tidak perlu membuat implementasi konkret. Penggunaan JpaRepository ini sangat menyederhanakan kode akses database dan mengurangi boilerplate code yang sebelumnya harus ditulis secara manual. Desain ini juga memungkinkan untuk dengan mudah menambahkan custom query methods di masa depan jika diperlukan dengan hanya mendeklarasikan method signature tanpa implementasi.

3.4 Service Layer (MahasiswaService.java)

Kelas MahasiswaService berfungsi sebagai layer logika bisnis yang menengahi antara controller dan repository untuk menangani semua operasi yang berhubungan dengan data mahasiswa. Anotasi @Service menandakan bahwa kelas ini adalah Spring component yang bertanggung jawab untuk implementasi logika bisnis. Kelas ini memiliki field mahasiswaRepository yang di-inject oleh Spring menggunakan anotasi @Autowired untuk memberikan akses ke operasi database. Method addMhs() menerima object ModelMahasiswa dan menyimpannya ke basis data melalui repository. Method getMhs() mencari dan mengembalikan object mahasiswa berdasarkan id tertentu, dengan mengembalikan null jika tidak ditemukan. Method updateMhs() mengupdate data mahasiswa yang sudah ada dengan cara menyimpan kembali object yang sudah dimodifikasi. Method deleteMhs() menghapus data mahasiswa dari basis data berdasarkan id. Method getAllMahasiswa() mengambil dan mengembalikan seluruh list mahasiswa dari basis data. Semua method di dalam service ini merupakan wrapper thin terhadap method repository, yang berarti logika bisnis di sini minimal dan terutama berfungsi untuk orchestration. Jika di masa depan diperlukan validasi data, penghitungan, atau operasi kompleks lainnya, kode tersebut dapat ditambahkan di layer ini tanpa perlu mengubah controller atau view. Struktur ini memastikan bahwa logika bisnis terisolasi dan mudah untuk ditest dan dipelihara.

3.5 Controller Layer (MahasiswaController.java)

Kelas MahasiswaController merupakan penghubung utama antara view dan service yang mengelola permintaan dari GUI dan mengarahkannya ke service yang sesuai untuk diproses. Anotasi @Component menandakan bahwa kelas ini adalah Spring managed bean yang akan di-instantiate dan dikelola oleh Spring container. Field mahasiswaService di-inject menggunakan @Autowired untuk memberikan akses ke layer service. Method addMahasiswa() menerima object ModelMahasiswa dari view dan mengirimkannya ke service untuk disimpan ke basis data. Method getMahasiswa() mencari data mahasiswa berdasarkan id yang diberikan dan mengembalikan object tersebut ke view. Method updateMahasiswa() menerima object mahasiswa yang sudah dimodifikasi dan mengirimkannya ke service untuk diupdate di basis data. Method deleteMahasiswa() menerima id mahasiswa dan mengirimkan perintah penghapusan ke service. Method getAllMahasiswa() mengambil seluruh list mahasiswa dari service untuk ditampilkan di view. Semua method di controller ini tidak memiliki decorators seperti @RequestMapping atau @GetMapping karena controller ini bukan REST controller tetapi component biasa yang digunakan oleh GUI. Method-method di

controller ini dirancang untuk menerima dan mengembalikan object Java langsung, bukan JSON atau String HTTP response. Peran controller adalah untuk decouple view dari service sehingga view tidak perlu mengetahui detail implementasi service.

3.6 View Layer (MahasiswaView.java)

Kelas MahasiswaView merupakan implementasi GUI utama yang meng-extend JFrame untuk membuat jendela aplikasi desktop. Anotasi @Component menandakan bahwa class ini dikelola oleh Spring container. Constructor kelas ini menerima MahasiswaController yang di-inject melalui parameter constructor dengan anotasi @Autowired, memungkinkan access ke semua fungsi bisnis. Dalam constructor, method initCustomComponents() dipanggil untuk membangun semua komponen UI, diikuti dengan loadMahasiswaTable() untuk mengisi tabel dengan data awal dari basis data. Method initCustomComponents() adalah method yang paling kompleks dalam kelas ini karena bertanggung jawab untuk membuat seluruh antarmuka pengguna. Pertama, method ini mendefinisikan berbagai font dan warna yang digunakan di seluruh aplikasi untuk memastikan konsistensi visual. Kemudian, method ini membuat custom JPanel dengan gradient background yang memberikan tampilan modern pada aplikasi. Panel utama menggunakan BorderLayout untuk mengorganisir komponen ke dalam berbagai region. Panel form di sebelah utara menampilkan label dan text field untuk input data mahasiswa yang terdiri dari NPM, Nama, Semester, dan IPK. Panel form menggunakan GridBagLayout untuk positioning yang lebih fleksibel dan responsif. Di bawah field input terdapat tiga tombol: tombol Simpan untuk menambahkan data baru, tombol Refresh untuk merefresh tampilan tabel, dan tombol Buang untuk menghapus data yang sudah ada. Panel tabel di bagian tengah menampilkan JTable yang menampilkan semua data mahasiswa yang diambil dari basis data dengan menggunakan ModelTabelMahasiswa sebagai model tabel. Setiap tombol memiliki event listener yang didefinisikan menggunakan anonymous inner class yang meng-implement ActionListener. Event listener untuk tombol Simpan mengambil nilai dari semua text field, membuat object ModelMahasiswa baru, mengirimkannya ke controller untuk disimpan, dan kemudian merefresh tabel serta mengosongkan field input. Event listener untuk tombol Refresh hanya memanggil method loadMahasiswaTable() untuk merefresh data tabel dari basis data. Event listener untuk tombol Buang menampilkan dialog box yang meminta user untuk memasukkan id mahasiswa yang ingin dihapus, dan setelah id divalidasi sebagai integer, data tersebut dihapus melalui controller dan tabel direfresh. Method loadMahasiswaTable() mengambil semua data mahasiswa dari controller, membuat instance baru dari ModelTabelMahasiswa dengan data tersebut, dan mengset model tabel pada JTable. Method styleButton() adalah utility method untuk mengaplikasikan styling konsisten pada tombol-tombol dan menambahkan efek hover sederhana ketika mouse memasuki area tombol. Terakhir, beberapa getter method disediakan untuk memberikan akses ke text field kepada event listener dan komponen lainnya. Desain view ini memastikan bahwa antarmuka pengguna intuitif, responsif, dan memberikan feedback yang jelas kepada user tentang operasi yang sedang dilakukan.

3.7 Main Application Class (MahasiswaApp.java)

Kelas MahasiswaApp merupakan entry point utama dari aplikasi yang di-anotasi dengan @SpringBootApplication untuk menandakan bahwa ini adalah aplikasi Spring Boot. Anotasi @SpringBootApplication merupakan kombinasi dari @SpringBootConfiguration, @EnableAutoConfiguration, dan @ComponentScan. Kelas ini meng-implement interface

ApplicationRunner untuk menjalankan custom logic setelah Spring context fully initialized. Field mahasiswaService di-inject menggunakan @Autowired meskipun tidak digunakan secara langsung dalam konteks ini, tetapi menunjukkan bagaimana dependency injection bekerja. Method main() adalah entry point yang pertama kali dijalankan ketika aplikasi distart. Di awal method ini, property java.awt.headless diset menjadi false untuk memastikan bahwa Swing GUI dapat ditampilkan tanpa masalah. Kemudian, SpringApplication.run() dipanggil untuk start Spring container dan menginisialisasi semua Spring beans termasuk controller, service, dan repository. Method ini mengembalikan ApplicationContext yang berisi semua Spring beans yang ter-instantiate. Setelah Spring context ready, MahasiswaController diambil dari context menggunakan method getBean() dan kemudian digunakan untuk membuat instance MahasiswaView. Instance MahasiswaView ini kemudian dibuat visible untuk ditampilkan kepada user. Method run() dari interface ApplicationRunner di-override tetapi dibiarkan kosong karena semua initialization logic sudah dilakukan di method main(). Struktur ini memastikan bahwa Spring container diinisialisasi terlebih dahulu sebelum GUI ditampilkan, sehingga semua beans sudah siap digunakan ketika aplikasi mulai berinteraksi dengan user.

4. LANGKAH-LANGKAH EKSEKUSI PROGRAM

4.1 Persiapan Awal

Sebelum menjalankan aplikasi, developer harus memastikan bahwa environment sudah tersiapkan dengan baik. Pertama, Java Development Kit (JDK) versi 21 atau lebih tinggi harus sudah terinstall di sistem karena aplikasi ini dikonfigurasi untuk menggunakan Java 21 sebagai target compilation. Kedua, MySQL server harus berjalan dan sudah siap menerima koneksi. Ketiga, database untuk aplikasi harus sudah dibuat terlebih dahulu dengan nama yang sesuai, dan tabel model_mahasiswa akan dibuat secara otomatis oleh Hibernate melalui property spring.jpa.hibernate.ddl-auto=create-drop atau sesuai konfigurasi yang ada. Keempat, file application.properties di folder resources harus dikonfigurasi dengan benar dengan informasi koneksi database seperti URL, username, dan password. Kelima, Maven harus sudah terinstall atau bisa menggunakan Maven wrapper yang mungkin sudah disertakan dalam proyek.

4.2 Build Proyek

Setelah persiapan awal selesai, proyek harus di-build untuk mengkompilasi seluruh source code menjadi bytecode yang dapat dijalankan di JVM. Build proyek dapat dilakukan dengan membuka command prompt atau terminal di direktori root proyek dan menjalankan perintah Maven untuk compile dan package. Maven akan download semua dependensi yang diperlukan dari repository Maven central berdasarkan konfigurasi yang ada di file pom.xml. Selama proses download ini, semua dependencies seperti Spring Boot, Hibernate, MySQL connector, dan Lombok akan didownload dan di-cache di local repository. Setelah semua dependencies berhasil didownload, Maven akan melakukan phase compile untuk mengkompilasi semua source code Java menjadi class files. Lombok akan di-trigger selama proses kompilasi untuk men-generate method-method yang dianotasikan pada class-class model. Setelah kompilasi berhasil, Maven akan melakukan phase package untuk membuat JAR file yang berisi semua compiled classes dan resources. File JAR ini akan disimpan di folder target dengan nama sesuai artifactId dan version yang dikonfigurasi di pom.xml.

4.3 Konfigurasi Database

Konfigurasi database sangat penting untuk memastikan aplikasi dapat terhubung dan berinteraksi dengan database secara benar. File application.properties yang terletak di resources harus dikonfigurasi dengan parameter koneksi yang sesuai. Parameter spring.datasource.url harus diset dengan URL koneksi MySQL yang menunjuk ke database yang sudah dibuat sebelumnya, biasanya dalam format jdbc:mysql://localhost:3306/nama_database. Parameter spring.datasource.username harus diset dengan username MySQL yang mempunyai privilege untuk mengakses database, umumnya adalah root. Parameter spring.datasource.password harus diset dengan password untuk username tersebut. Parameter spring.jpa.hibernate.ddl-auto harus diset dengan nilai yang menentukan bagaimana Hibernate menangani schema database, nilai create akan membuat tabel baru setiap kali aplikasi distart dan menghapus yang lama, nilai update akan mengupdate schema yang ada atau membuat yang baru jika tidak ada. Parameter spring.jpa.show-sql dapat diset ke true untuk menampilkan SQL queries yang dijalankan Hibernate di console untuk keperluan debugging.

4.4 Menjalankan Aplikasi

Setelah build dan konfigurasi selesai, aplikasi dapat dijalankan dengan berbagai cara. Cara pertama adalah dengan menjalankan perintah Maven run di terminal yang akan mengkompilasi dan menjalankan aplikasi dalam satu perintah. Cara kedua adalah dengan menjalankan JAR file yang sudah di-build di folder target menggunakan perintah Java. Cara ketiga adalah dengan menjalankan aplikasi langsung dari IDE seperti IntelliJ IDEA atau Eclipse dengan mengclick tombol run. Ketika aplikasi dijalankan, Spring Boot akan melakukan auto-configuration berdasarkan classpath dan properties file yang ada. Spring akan membuat connection pool ke database MySQL menggunakan HikariCP yang merupakan default connection pool di Spring Boot 3. Spring akan juga membuat semua beans yang didefinisikan dengan anotasi Spring seperti services, controllers, dan repositories. Setelah Spring context fully initialized, method run() dari interface ApplicationRunner akan dipanggil, yang dalam hal ini tidak melakukan apa-apa. Kemudian, method main() akan menampilkan GUI window kepada user dengan title "Manajemen Data Mahasiswa (Versi Spring)".

5. OUTPUT DAN FUNGSI PROGRAM

5.1 Antarmuka Pengguna

Ketika aplikasi dijalankan, window utama akan ditampilkan dengan tampilan yang profesional dan user-friendly. Window ini memiliki dua bagian utama yang terletak di atas dan di tengah. Bagian atas berisi form input data untuk menambahkan mahasiswa baru dengan lima komponen: label NPM dan text field untuk input NPM, label Nama dan text field untuk input nama lengkap mahasiswa, label Semester dan text field untuk input semester yang sedang ditempuh, dan label IPK dengan text field untuk input nilai IPK. Semua text field memiliki lebar yang sama dan tersusun dalam grid layout yang rapi untuk memberikan tampilan yang seragam dan profesional. Di bawah form input terdapat tiga tombol berwarna berbeda yang berfungsi untuk operasi berbeda: tombol Simpan berwarna biru untuk menyimpan data mahasiswa baru, tombol Refresh berwarna cyan untuk merefresh tampilan tabel, dan tombol Buang berwarna merah untuk menghapus data mahasiswa. Bagian tengah window

menampilkan tabel data mahasiswa yang berisi lima kolom: kolom ID untuk menampilkan id unik mahasiswa yang di-generate secara otomatis oleh database, kolom NPM untuk menampilkan nomor pokok mahasiswa, kolom Nama untuk menampilkan nama lengkap mahasiswa, kolom Semester untuk menampilkan semester yang sedang ditempuh, dan kolom IPK untuk menampilkan nilai indeks prestasi kumulatif. Tabel ini dilengkapi dengan scrollbar horizontal dan vertikal untuk menampilkan lebih banyak data jika diperlukan. Background window memiliki gradient color yang halus dari biru muda ke putih yang memberikan tampilan modern dan tidak membosankan.

5.2 Fungsi Tambah Data (Create)

Fungsi tambah data memungkinkan user untuk menambahkan record mahasiswa baru ke dalam database. User dapat memasukkan data melalui text field di form input dengan memasukkan NPM, nama, semester, dan IPK mahasiswa. Setelah semua field diisi dengan data yang valid, user dapat mengklik tombol Simpan untuk menyimpan data. Event listener dari tombol Simpan akan membaca nilai dari semua text field, melakukan konversi tipe data yang diperlukan yaitu semester dikonversi menjadi integer dan IPK dikonversi menjadi float, kemudian membuat object ModelMahasiswa baru dengan data tersebut. Object mahasiswa ini kemudian dilewatkan ke controller melalui method addMahasiswa(). Controller akan meneruskan object ini ke service, yang kemudian akan menyimpannya ke database melalui repository. Setelah data berhasil disimpan, tabel akan otomatis di-refresh untuk menampilkan data mahasiswa yang baru ditambahkan. Console juga akan menampilkan debug output berupa nilai IPK, nama, semester, dan NPM yang baru ditambahkan untuk memverifikasi bahwa data telah diproses dengan benar. Semua text field kemudian akan dikosongkan untuk memudahkan input data berikutnya.

5.3 Fungsi Tampil Data (Read)

Fungsi tampil data menampilkan semua data mahasiswa yang tersimpan di database dalam bentuk tabel yang terstruktur. Ketika aplikasi pertama kali dijalankan, method loadMahasiswaTable() akan langsung dipanggil untuk memuat data awal. Method ini akan memanggil controller untuk mengambil semua data mahasiswa dari database melalui method getAllMahasiswa(). Data yang dikembalikan berupa List of ModelMahasiswa akan dimasukkan ke dalam instance baru dari ModelTabelMahasiswa yang berfungsi sebagai adapter untuk mengkonversi List data menjadi format yang dapat ditampilkan di JTable. Model tabel ini kemudian di-assign ke JTable menggunakan method setModel(). JTable akan secara otomatis menampilkan data dalam bentuk baris dan kolom sesuai dengan struktur yang didefinisikan di ModelTabelMahasiswa. Setiap baris mewakili satu record mahasiswa, dan setiap kolom menampilkan field yang berbeda dari mahasiswa seperti ID, NPM, Nama, Semester, dan IPK. Jika user menambahkan data baru atau menghapus data, tabel akan di-refresh secara otomatis untuk menampilkan perubahan terbaru. User juga dapat mengklik tombol Refresh secara manual untuk memastikan data yang ditampilkan adalah data paling terkini dari database.

5.4 Fungsi Perbarui Data (Update)

Fungsi update atau perbarui data memungkinkan user untuk mengubah data mahasiswa yang sudah ada di database. Meskipun dalam implementasi saat ini tidak ada UI button khusus untuk update, fungsi ini tersedia melalui service dan controller. Untuk melakukan update, user dapat

menggunakan pendekatan alternatif dengan memanfaatkan method updateMahasiswa() di controller yang dapat dipanggil secara langsung jika UI diperluas di masa depan. Proses update melibatkan pengambilan object mahasiswa yang ingin diupdate, memodifikasi field-field yang diperlukan, dan kemudian mengirimkan object yang sudah dimodifikasi ke controller. Controller akan meneruskan object ini ke service, yang kemudian akan memanggil method save() di repository. Karena object ini sudah memiliki ID yang sama dengan record di database, Hibernate akan mengenali ini sebagai update operation dan tidak akan membuat record baru melainkan mengupdate record yang sudah ada dengan nilai-nilai baru. Setelah operasi update berhasil, perubahan data akan segera terlihat ketika tabel di-refresh.

5.5 Fungsi Hapus Data (Delete)

Fungsi hapus data memungkinkan user untuk menghapus record mahasiswa dari database berdasarkan ID-nya. User dapat mengklik tombol Buang (Delete) untuk memulai proses penghapusan. Event listener dari tombol ini akan menampilkan dialog box kepada user yang meminta untuk memasukkan ID mahasiswa yang ingin dihapus. Dialog box ini berisi label untuk menjelaskan maksud dialog, text field untuk input ID, dan tombol OK dan Cancel untuk konfirmasi atau pembatalan. User harus memasukkan ID dalam bentuk integer yang valid. Setelah user mengklik tombol OK, program akan mencoba untuk mengkonversi input text menjadi integer. Jika konversi berhasil, ID tersebut akan dilewatkan ke controller melalui method deleteMahasiswa(). Controller akan meneruskan ID ini ke service, yang kemudian akan memanggil method deleteById() di repository untuk menghapus record dengan ID tersebut dari database. Jika penghapusan berhasil, dialog box sukses akan ditampilkan kepada user, dan tabel akan di-refresh untuk menampilkan bahwa data telah dihapus. Jika user memasukkan nilai yang bukan integer, program akan menangkap exception NumberFormatException dan menampilkan dialog error message kepada user. Jika user mengklik tombol Cancel di dialog input ID, operasi penghapusan akan dibatalkan dan tidak ada yang dihapus.

6. INTEGRASI SPRING BOOT DENGAN SWING

6.1 Dependency Injection dan Spring Container

Aplikasi ini mendemonstrasikan integrasi yang elegan antara Spring Framework dengan Swing GUI. Spring container mengelola semua beans aplikasi termasuk service, controller, dan repository. MahasiswaView juga dimanaatkan oleh Spring container melalui anotasi @Component, yang berarti Spring akan membuat instance dari view dan melakukan dependency injection secara otomatis. Ketika Spring container di-initialize di method main() melalui SpringApplication.run(), semua beans akan di-create dan di-inject dengan dependencies mereka. MahasiswaController akan di-create dengan MahasiswaService di-inject melalui anotasi @Autowired. MahasiswaService akan di-create dengan MahasiswaRepository di-inject. MahasiswaView akan di-create dengan constructor-based injection menerima MahasiswaController yang sudah di-create sebelumnya. Pattern ini memastikan bahwa semua komponen aplikasi ter-couple dengan longgar dan mudah untuk ditest serta dipelihara. Spring container menangani lifecycle dari semua beans dan memastikan bahwa resources di-allocate dan di-deallocate dengan benar.

6.2 Database Connectivity dan ORM

Spring Data JPA menyediakan abstraksi yang powerful untuk berinteraksi dengan database tanpa perlu menulis SQL queries secara manual. Hibernate, sebagai ORM implementation yang digunakan di belakang Spring Data JPA, secara otomatis memetakan Java objects ke tabel database dan sebaliknya. Ketika aplikasi start, Spring akan membaca property file dan setup datasource connection yang menghubungkan aplikasi ke MySQL database. Entity class ModelMahasiswa di-anotasi dengan @Entity dan akan di-scan oleh Hibernate untuk dibuat menjadi tabel di database secara otomatis sesuai dengan property spring.jpa.hibernate.ddl-auto. Setiap field di entity class akan menjadi kolom di tabel, dan anotasi-anotasi seperti @Id dan @GeneratedValue akan dikonfigurasi untuk primary key dan auto-increment. Ketika aplikasi melakukan operasi CRUD melalui repository interface, Hibernate akan secara otomatis membuat SQL queries yang sesuai dan menjalankannya terhadap database. Hasil dari query akan otomatis dikonversi menjadi Java objects sesuai dengan entity class definition. Integrasi ini membuat kode aplikasi bebas dari SQL boilerplate dan fokus pada business logic.