LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Rekayasa Perangkat Lunak 2
Kelas	: 4IA07
Praktikum ke-	: 4
Tanggal	: 8 November 2025
Materi	: ORM
NPM	: 51422084
Nama	: Muhammad Izzuddin Almansyur
Ketua Asisten	: Haikal Abizar
Jumlah Lembar	: 8

LABORATORIUM INFORMATIKA

UNIVERSITAS GUNADARMA

2025

1. Jelaskan apa yang dimaksud dengan ORM dan keuntungannya!

   ORM (*Object Relational Mapping*) adalah teknik yang menghubungkan objek dalam pemrograman berorientasi objek dengan tabel di database relasional. Dengan ORM, developer tidak perlu menulis SQL secara manual karena operasi pada objek otomatis diterjemahkan menjadi query SQL. Keuntungannya antara lain mempermudah interaksi dengan database, meningkatkan produktivitas, menjaga konsistensi kode, membuat aplikasi lebih portabel antar database, serta memudahkan pemeliharaan karena struktur program lebih terorganisir.

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

   Langkah – langkah pembuatan program pada netbeans:

   1) Buka NetBeans → File → New Project → Maven → Java Application.
   2) Beri nama: rpl_pert4, GroupId: com.mycompany, Finish.
   3) Buka file pom.xml, dan tambahkan dependency
   4) Buat database di xampp
   5) Buat Java Package:

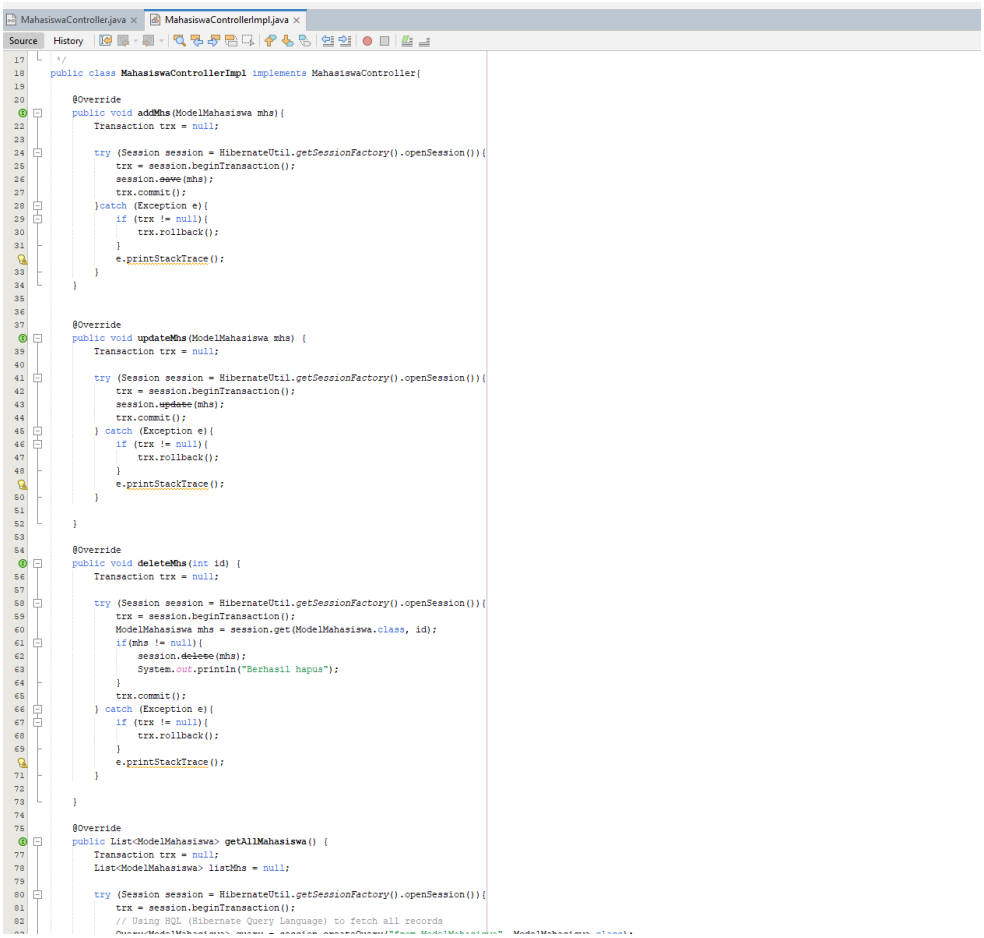
      Klik kanan → New → Java Package dan beri nama Model, Controller, dan View
   6) Buat class java di masing – masing package:
      - Controller = MahasiswaController, MahasiswaControllerImpl
      - Model = HibernateUtil, ModelMahasiswa, ModelTabelMahasiswa
      - View = MahasiswaView

   Penjelasan Code:

   MahasiswaController.java

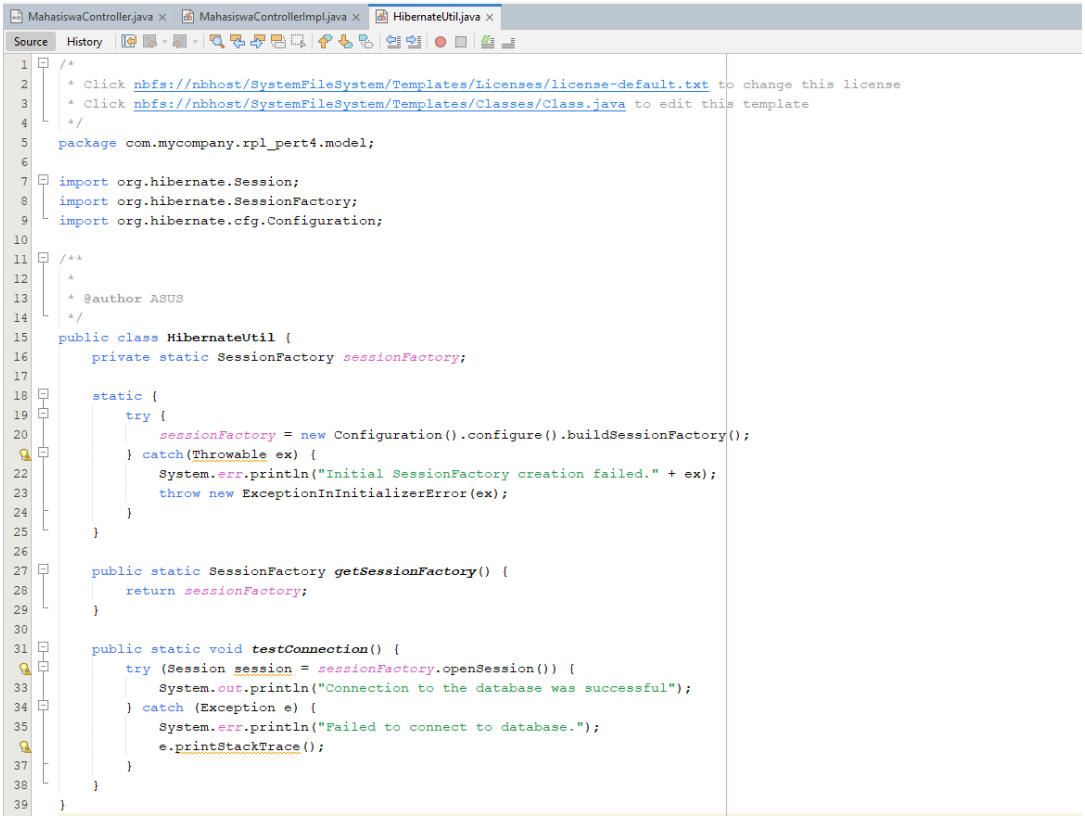
MahasiswaController adalah sebuah **interface** yang mendefinisikan kontrak dari operasi CRUD yang bisa dilakukan terhadap entitas mahasiswa. Interface ini berisi deklarasi metode seperti addMhs() untuk menambah data mahasiswa, getMhs() untuk mengambil data berdasarkan ID, updateMhs() untuk memperbarui data, deleteMhs() untuk menghapus data, dan getAllMahasiswa() untuk mengambil semua data mahasiswa dari tabel. Interface ini belum memiliki implementasi logika, hanya mendefinisikan struktur metode yang akan diimplementasikan oleh kelas lain.

MahasiswaControllerImpl.java



Selanjutnya, MahasiswaControllerImpl adalah kelas yang **mengimplementasikan interface** tersebut dan berisi logika CRUD menggunakan Hibernate. Pada metode addMhs(), objek ModelMahasiswa yang berisi data baru disimpan ke database dengan menggunakan session Hibernate dan transaksi (Transaction). Jika proses berhasil, transaksi akan di-commit, namun jika terjadi kesalahan, transaksi akan di-rollback agar database tetap konsisten. Metode updateMhs() dan deleteMhs() bekerja dengan cara yang mirip, hanya berbeda pada fungsi yang dipanggil (session.update() dan session.delete()). Metode getAllMahasiswa() menggunakan **HQL (Hibernate Query Language)** untuk mengambil seluruh data mahasiswa dari tabel mahasiswa, kemudian mengembalikannya dalam bentuk List<ModelMahasiswa>. Dengan demikian, kelas ini berfungsi sebagai penghubung antara model data dan tampilan antarmuka pengguna.

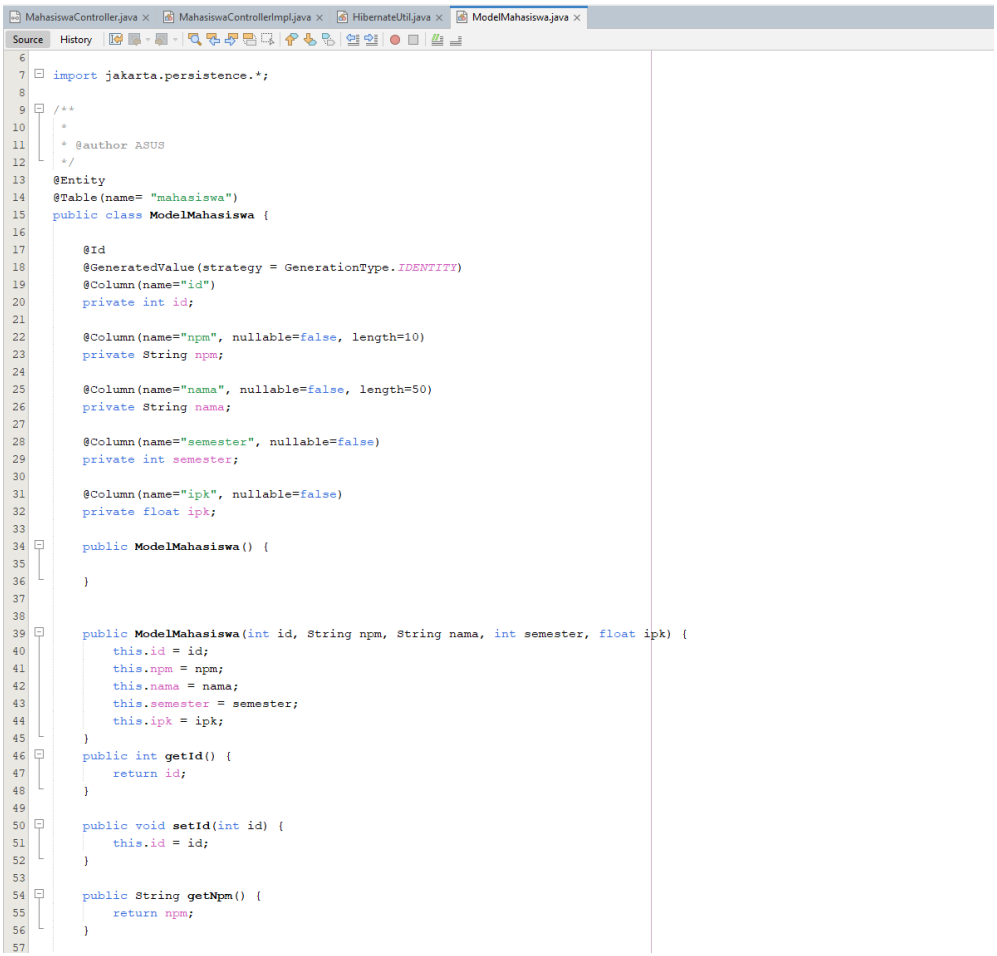
HibernateUtil.java



```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.rpl_pert4.model;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

/**
 *
 * @author ASUS
 */
public class HibernateUtil {
    private static SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch(Throwable ex) {
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void testConnection() {
        try (Session session = sessionFactory.openSession()) {
            System.out.println("Connection to the database was successful");
        } catch (Exception e) {
            System.err.println("Failed to connect to database.");
            e.printStackTrace();
        }
    }
}
```

Kelas HibernateUtil berfungsi untuk **mengatur koneksi ke database** dengan memanfaatkan konfigurasi Hibernate. Di dalam kelas ini terdapat SessionFactory, yaitu objek utama yang digunakan Hibernate untuk membuat sesi (session) koneksi ke database. Di dalam blok static initializer, konfigurasi Hibernate dijalankan melalui new Configuration().configure().buildSessionFactory(). Metode testConnection() digunakan untuk menguji apakah koneksi ke database berhasil dengan mencoba membuka sesi dan menampilkan pesan sukses di konsol.

ModelMahasiswa.java



```java
import jakarta.persistence.*;

/**
 *
 * @author ASUS
 */
@Entity
@Table(name= "mahasiswa")
public class ModelMahasiswa {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="npm", nullable=false, length=10)
    private String npm;

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

    @Column(name="semester", nullable=false)
    private int semester;

    @Column(name="ipk", nullable=false)
    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;
    }

```

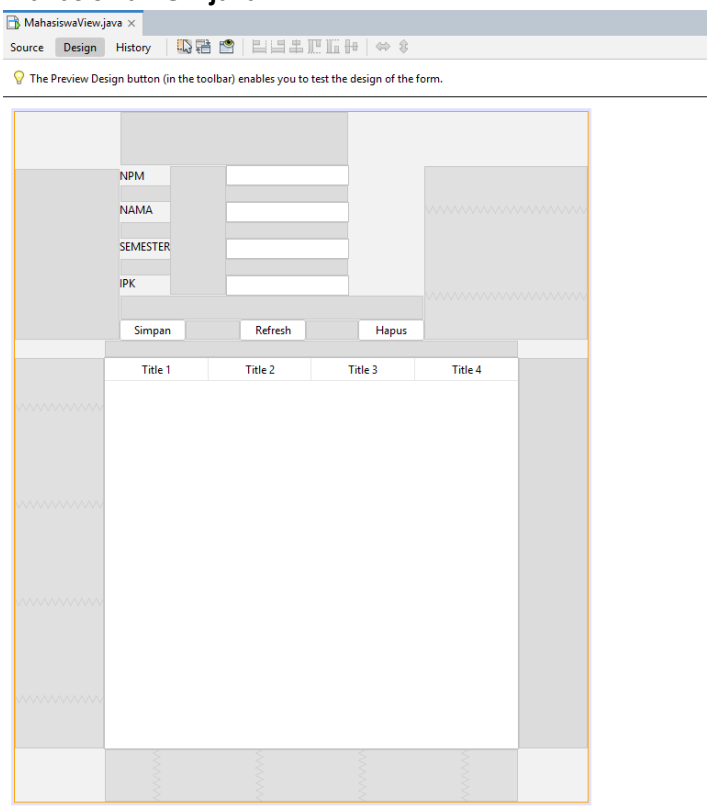
Kelas ModelMahasiswa merupakan **entity class** yang memetakan data di database ke dalam bentuk objek Java menggunakan anotasi **Jakarta Persistence (JPA)**. Kelas ini diberi anotasi @Entity dan @Table(name="mahasiswa") untuk menunjukkan bahwa kelas ini merepresentasikan tabel bernama “mahasiswa” di database. Setiap atribut di kelas ini seperti id, npm, nama, semester, dan ipk diberi anotasi @Column untuk menyesuaikan nama kolom di tabel serta pengaturan seperti panjang karakter dan sifat nullable. Atribut id memiliki anotasi @Id dan @GeneratedValue(strategy = GenerationType.IDENTITY) yang berarti kolom tersebut adalah primary key dan nilainya akan diisi otomatis oleh database. Kelas ini juga memiliki konstruktor kosong (wajib untuk entity Hibernate) dan konstruktor dengan parameter, serta getter dan setter untuk mengakses setiap atribut.

ModelTabelMahasiswa.java

```java
public class ModelTabelMahasiswa extends AbstractTableModel {

    private List<ModelMahasiswa> mahasiswaList;
    private String[] columnNames = {"ID", "NPM", "Nama", "Semester", "IPK"};

    public ModelTabelMahasiswa(List<ModelMahasiswa> mahasiswaList) {
        this.mahasiswaList = mahasiswaList;
    }
    @Override
    public int getRowCount() {
        return mahasiswaList.size(); // Jumlah baris sesuai dengan jumlah data mahasiswa
    }

    @Override
    public int getColumnCount() {
        return columnNames.length; // Jumlah kolom sesuai dengan jumlah elemen dalam columnNames
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        ModelMahasiswa mahasiswa = mahasiswaList.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return mahasiswa.getId();
            case 1:
                return mahasiswa.getNpm();
            case 2:
                return mahasiswa.getNama();
            case 3:
                return mahasiswa.getSemester();
            case 4:
                return mahasiswa.getIpk();
            default:
                return null;
        }
    }

    @Override
    public String getColumnName(int column) {
        return columnNames[column]; // Mengatur nama kolom
    }

    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return false; // Semua sel tidak dapat diedit
    }

    // Method untuk menambahkan atau memodifikasi data, jika dibutuhkan
    public void setMahasiswaList(List<ModelMahasiswa> mahasiswaList) {
        this.mahasiswaList = mahasiswaList;
        fireTableDataChanged(); // Memberitahu JTable bahwa data telah berubah
    }
```

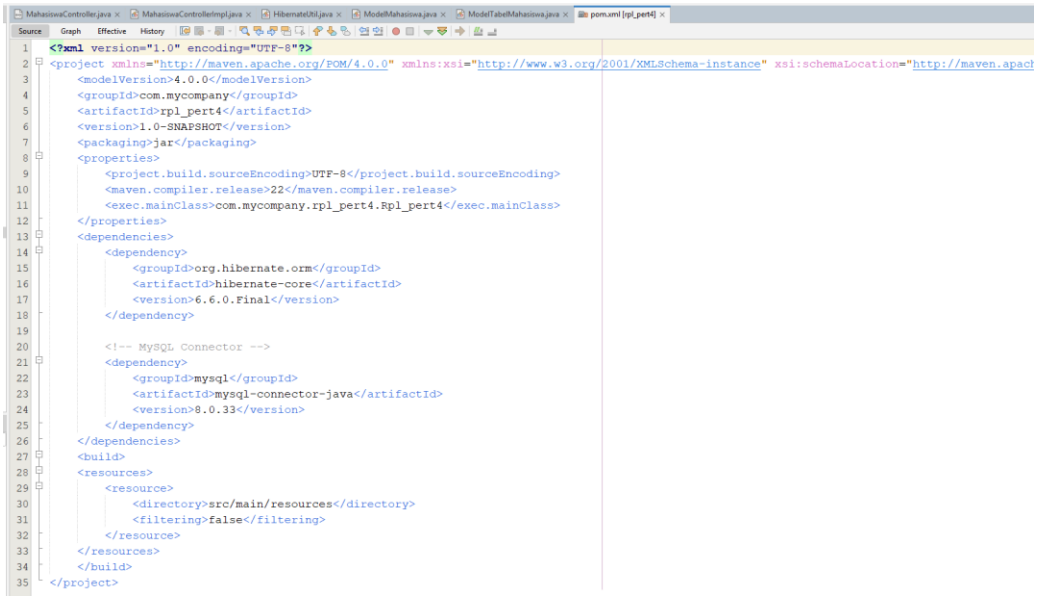
Kelas ModelTabelMahasiswa berfungsi untuk menampilkan data mahasiswa dalam **JTable** pada antarmuka grafis. Kelas ini meng-extend AbstractTableModel, sehingga dapat digunakan untuk mengatur bagaimana data ditampilkan dalam bentuk tabel di Swing. Di dalamnya terdapat List<ModelMahasiswa> yang menyimpan data mahasiswa, serta array columnNames untuk mendefinisikan nama kolom tabel. Metode getRowCount() dan getColumnCount() mengatur jumlah baris dan kolom tabel, sedangkan getValueAt() digunakan untuk menentukan data apa yang muncul pada setiap sel tabel. Dengan cara ini, tabel pada GUI akan menampilkan data yang diambil dari database melalui Hibernate.

MahasiswaView.java



Pada package **view**, terdapat kelas MahasiswaView yang merupakan **tampilan utama aplikasi berbasis Swing**. Kelas ini berperan sebagai antarmuka pengguna yang memungkinkan input data mahasiswa serta menampilkan data dari database dalam bentuk tabel.
Di dalam konstruktor MahasiswaView(), program melakukan inisialisasi komponen GUI melalui initComponents(), membuat objek controller (MahasiswaControllerImpl), menguji koneksi ke database (HibernateUtil.testConnection()), lalu memanggil loadMahasiswaTable() untuk menampilkan data awal di tabel.
Metode loadMahasiswaTable() akan mengambil semua data mahasiswa dari database melalui controller.getAllMahasiswa(), lalu menampilkannya ke tabel menggunakan ModelTabelMahasiswa.
Ketika pengguna menekan tombol **Simpan**, event btnSimpanActionPerformed() akan dijalankan. Metode ini mengambil input dari text field (npm, nama, semester, ipk), membuat objek ModelMahasiswa baru, lalu memanggil controller.addMhs(mahasiswa) untuk menyimpan data ke database. Setelah itu, tabel diperbarui agar menampilkan data terbaru. Tombol **Refresh** menjalankan loadMahasiswaTable() untuk memuat ulang data dari database. Sedangkan tombol **Hapus** membuka dialog input menggunakan JOptionPane untuk meminta ID mahasiswa yang ingin dihapus, kemudian memanggil controller.deleteMhs(id) untuk menghapus data tersebut dari database. Setelah operasi berhasil, program menampilkan pesan konfirmasi kepada pengguna.

Pom.xml



```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-instance" xsi:schemaLocation="http://maven.apach
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>rpl_pert4</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.release>22</maven.compiler.release>
        <exec.mainClass>com.mycompany.rpl_pert4.Rpl_pert4</exec.mainClass>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>6.6.0.Final</version>
        </dependency>

        <!-- MySQL Connector -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>
    </dependencies>
    <build>
    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>false</filtering>
        </resource>
    </resources>
    </build>
</project>
```

Terakhir, file **pom.xml** merupakan konfigurasi Maven yang berisi informasi proyek dan dependensi yang digunakan. Di dalamnya terdapat dependensi untuk hibernate-core versi 6.6.0.Final dan mysql-connector-java versi 8.0.33. Dependensi ini memungkinkan proyek untuk menggunakan Hibernate ORM dan melakukan koneksi ke database MySQL. Selain itu, file ini juga berisi informasi build, seperti encoding UTF-8, versi Java (22), dan main class (com.mycompany.rpl_pert4.Rpl_pert4).

KESIMPULAN:

Secara keseluruhan, proyek ini merupakan implementasi **arsitektur MVC (Model-View-Controller)** pada aplikasi desktop Java. Bagian **Model** mengatur struktur data dan koneksi database melalui Hibernate, **Controller** menangani logika bisnis dan operasi CRUD, sementara **View** berfungsi sebagai antarmuka interaktif bagi pengguna. Dengan pembagian ini, kode menjadi lebih terstruktur, mudah dipelihara, dan sesuai dengan prinsip pemrograman berorientasi objek.