LAPORAN PRAKTIKUM
PEMOGRAMAN BERORIENTASI OBJEK

PRAKTIKUM 2

Membuat Fungsi CRUD (Create,Read,Update Delete) USER Dengan Database MYSQL

Disusun oleh:

Renaldi Alwean Saputra

2411532003

Dosen Pengampu :

Nurfiah, S.ST, M.Kom

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

SEPTEMBER

2025

A. Pendahuluan

Pemrograman Berorientasi Objek (PBO) merupakan salah satu paradigma pemrograman yang banyak digunakan dalam pengembangan aplikasi modern. Dalam praktiknya, PBO sangat mendukung pembuatan aplikasi yang modular, mudah dikelola, dan dapat digunakan kembali. Salah satu contoh penerapannya adalah dalam pengembangan aplikasi berbasis database.

CRUD (*Create, Read, Update, Delete*) merupakan operasi dasar yang wajib ada pada sebuah aplikasi untuk mengelola data. Dengan adanya fungsi CRUD, pengguna dapat menambahkan data baru, menampilkan data yang sudah ada, memperbarui data, maupun menghapus data. Pada praktikum ini, mahasiswa diminta untuk membuat fungsi CRUD data user menggunakan bahasa pemrograman Java dengan database MySQL, serta mengimplementasikan konsep PBO melalui penggunaan class, interface, dan DAO (*Data Access Object*).

B. Tujuan

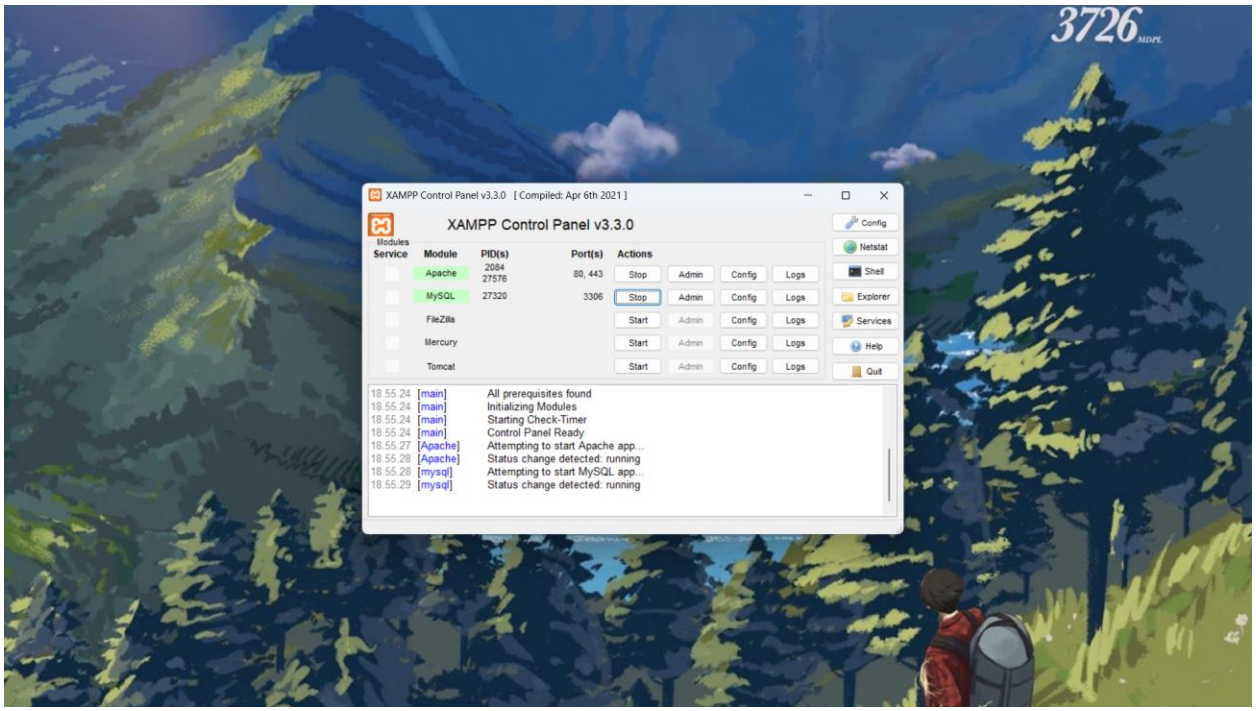
Tujuan praktikum ini yaitu :

- Mahasiswa mampu membuat tabel **user** pada database MySQL.
- Mahasiswa mampu membuat koneksi antara aplikasi Java dan database MySQL.
- Mahasiswa mampu merancang tampilan GUI untuk fungsi CRUD user.
- Mahasiswa mampu membuat dan mengimplementasikan interface dalam Java.
- Mahasiswa mampu membuat fungsi **DAO** (Data Access Object) untuk mengakses database.
- Mahasiswa mampu mengimplementasikan operasi CRUD (*Create, Read, Update, Delete*) dengan konsep Pemrograman Berorientasi Objek.

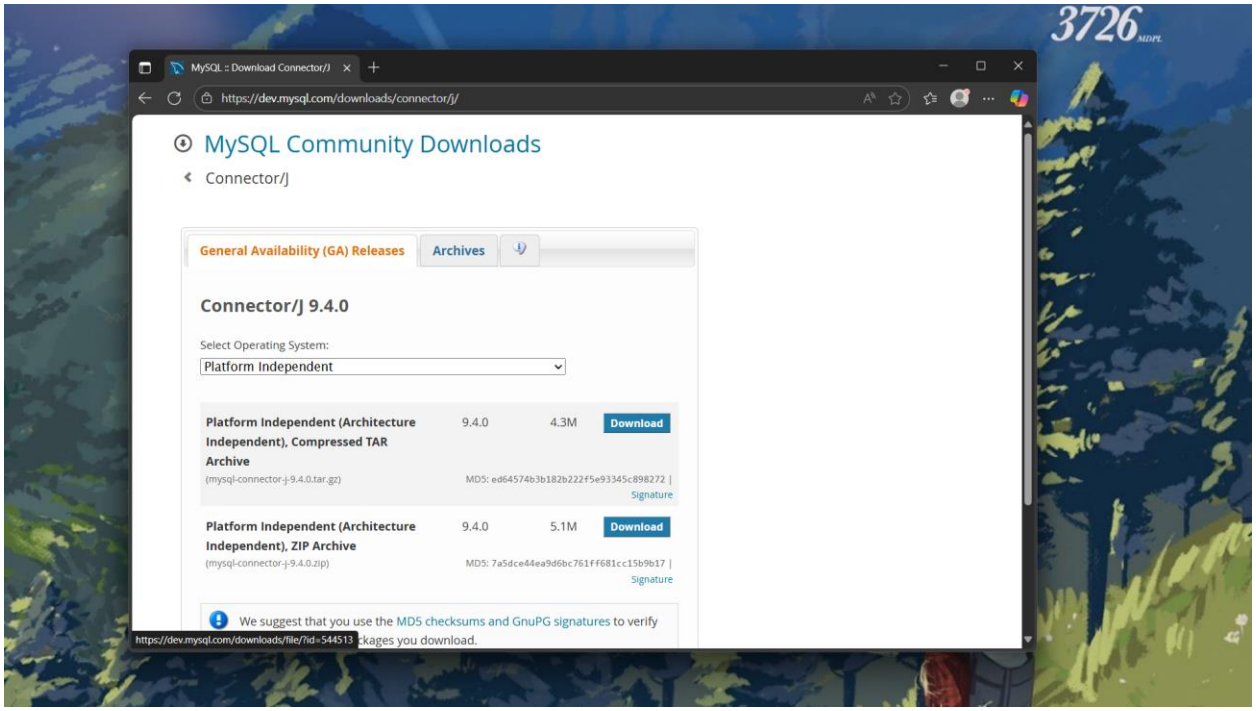
C. Langkah kerja

1. Install dan Jalankan XAMPP

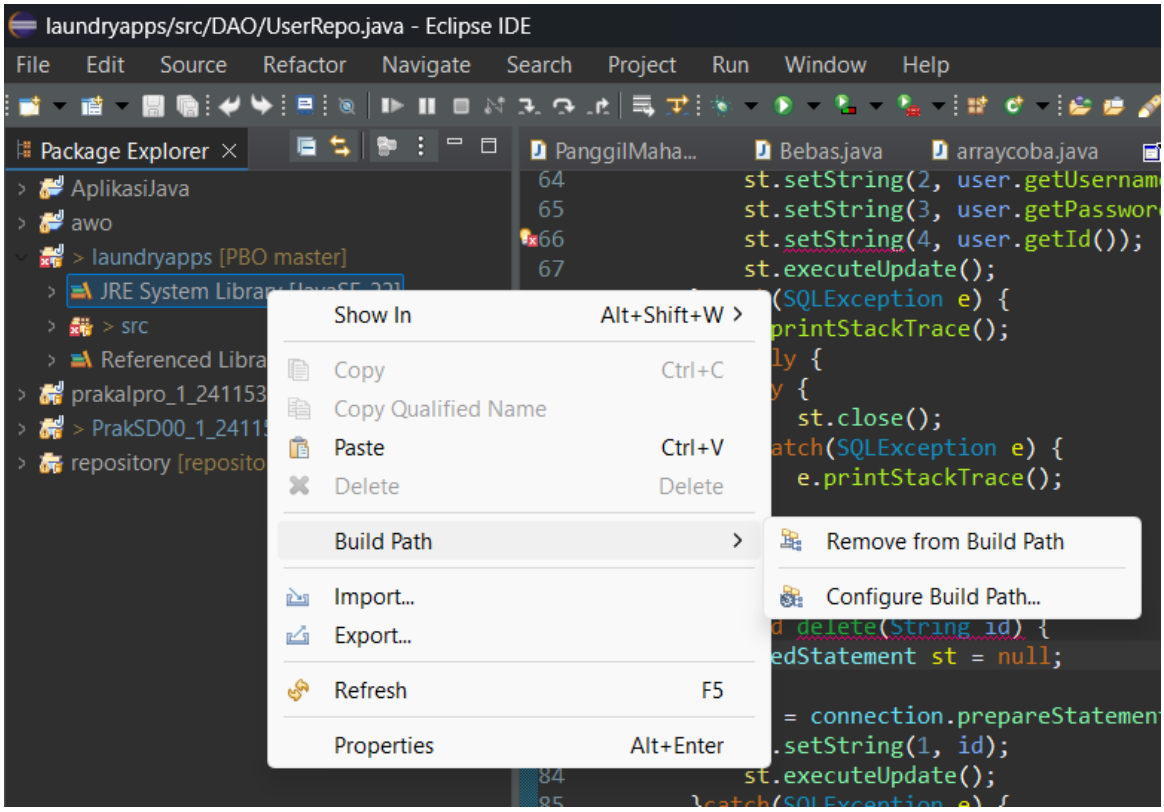
- Download dan Install XAMPP , jika sudah Run aplikasi nya dan jalankan bagian Apache dan MySQL

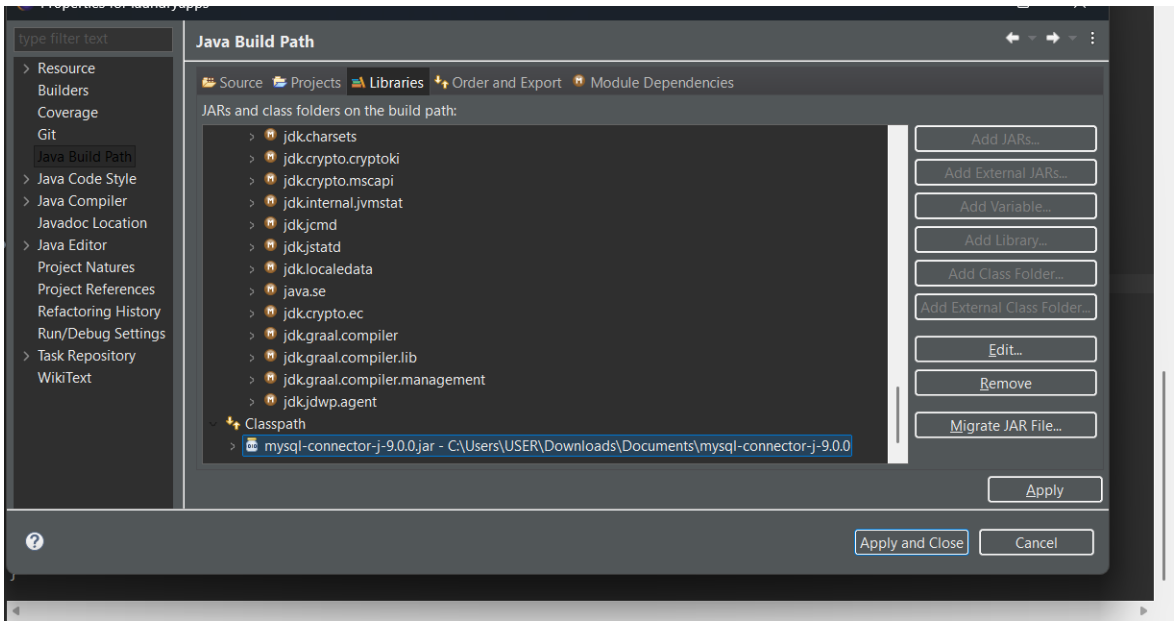


Jika sudah seperti gambar diatas , Langkah selanjutnya Adalah download MySql connector nya dari link (MySQL :: Download Connector/J) lalu download yang file bawah



Lalu tambahkan ke java eclipse dengan tahap seperti berikut : Menambahkan MySQL Connector kedalam project dengan cara klik kanan directory JRE System Library → Built Path → Configure Build Path

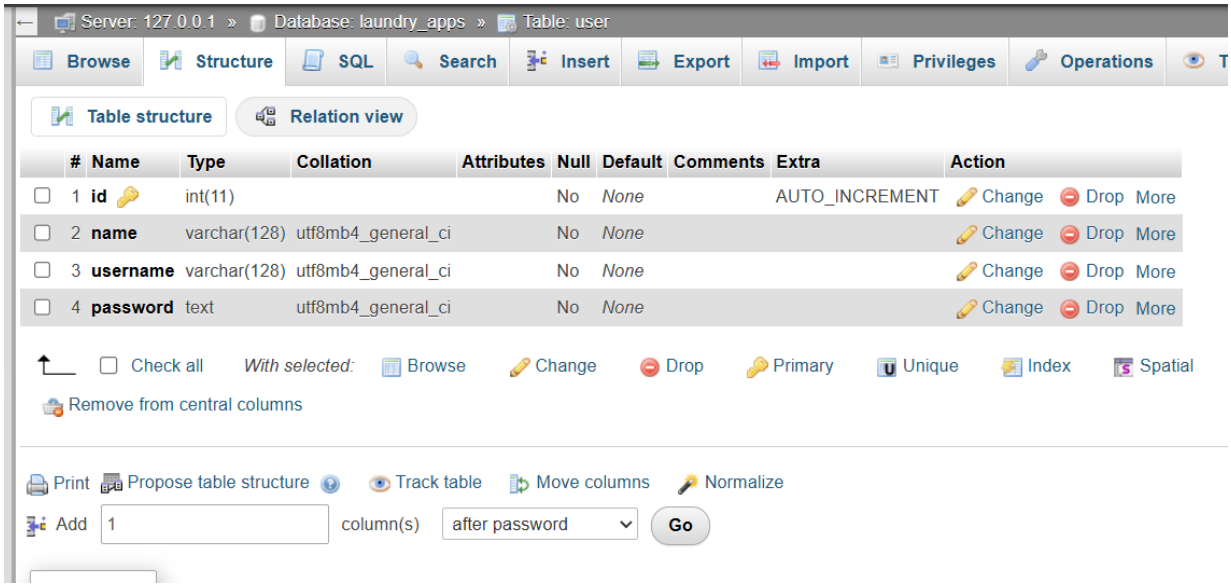




Nah jika sudah seperti ini maka mysql connector sudah terhubung dengan project java

2. Membuat Table dan Data baru

- Selanjutnya buka : http://localhost/phpmyadmin
- Buatlah tabel seperti di bawah



Jika sudah Langkah selanjutnya Adalah membuat koneksi ke Database MYSQL

- Buat Package baru Namanya Config dan buat class baru dengan nama Database.java

```java
package config;

import java.sql.*;

public class Database {
    public static Connection koneksi() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/laundry_apps?useSSL=false&serverTimezone=UTC",
                "root",
                ""
            );
            return conn;
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Koneksi Gagal: " + e.getMessage());
            return null;
        }
    }
}
```

Penjelasan :

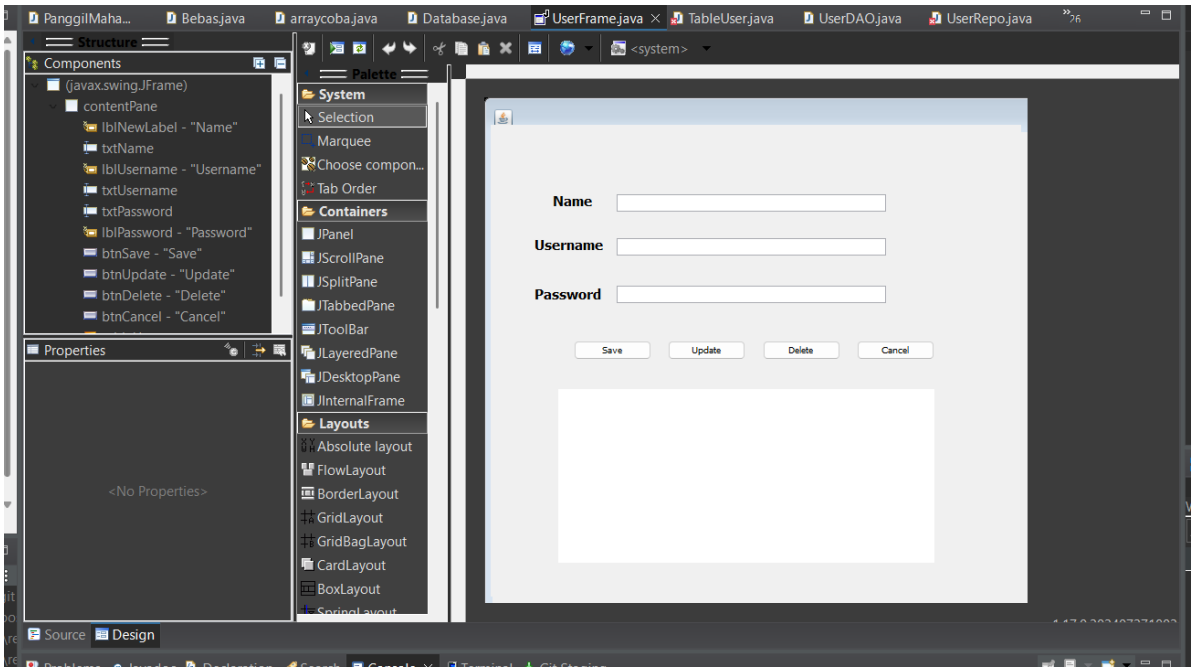
- Import java.sql.* digunakan untuk import seluruh fungsi-fungsi SQL
- Line 8 membuka method Connection dengan nama koneksi, yang mana method ini akan digunakan untuk membuka koneksi ke database
- Line 10-13 membuat koneksi database, jika koneksi berhasil maka akan mengembalikkan nilai Connection

- Line 15-16 jika koneksi gagal maka akan ditampilkan pesan error menggunakan JOptionPane.

Buatlah codingan seperti diatas , lalu jalankan untuk tes koneksi apakah sudah terhubung atau belum

3. Membuat Tampilan CRUD User

- Buat file baru JFrame pada package ui dengan nama user frame dengan desain seperti dibawah



4. Membuat Table Model

- Buat package baru dengan nama Table
- Buat file baru didalam package Table dengan nama TableUser, kemudian isi dengan kode program berikut:

```java
public class TableUser extends AbstractTableModel {
    List<User> ls;
    private String[] columnNames = {"ID", "Name", "Username","Password"};
    public TableUser(List<User> ls) {
        this.ls = ls;
    }

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return 4;
    }

    @Override
    public String getColumnName(int column) {
        // TODO Auto-generated method stub
        return columnNames[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        switch (columnIndex) {
        case 0:
            return ls.get(rowIndex).getId();
        case 1:
            return ls.get(rowIndex).getNama();
        case 2:
            return ls.get(rowIndex).getUsername();
        case 3:
            return ls.get(rowIndex).getPassword();
        default:
            return null;
        }
    }
```

5. Membuat Fungsi DAO

- Buat Package baru dengan nama UserDAO kemudian isi dengan kode berikut

PanggilMaha... | Bebas.java | arraycoba.java | Teguh.java | Database.java | UserFrame.java | TableUs

```java
package DAO;

import java.util.List;

public interface UserDAO {
    public void save(User user);
    public List<User> show();
    public void update(User user);
    public void delete(int id);
}
```

- Buat class baru lagi dengan nama UserRepo lalu masukkan kode berikut

PanggilMaha... Bebas.java arraycoba.java Teguh.java Database.java UserFrame.java TableUser.java UserDAO.java UserRepo.java

```java
package DAO;

import config.Database;

public class UserRepo implements UserDAO {
    private Connection connection;
    final String insert = "INSERT INTO user (name, username, password) VALUES (?,?,?);";
    final String select = "SELECT * FROM user;";
    final String delete = "DELETE FROM user WHERE id=?;";
    final String update = "UPDATE user SET name=?, username=?, password=? WHERE id=?;";

    public UserRepo() {
        connection = Database.koneksi();
    }

    @Override
    public void save(User user) {
        PreparedStatement st = null;
        try {
            st.setString(1, user.getNama());
            st.setString(2, user.getUsername());
            st.setString(3, user.getPassword());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
        }
    @Override
    public List<User> show() {
        List<User> ls=null;
        try {
            ls = new ArrayList<User>();
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery(select);
            while(rs.next()) {
```

```java
    @Override
    public List<User> show() {
        List<User> ls=null;
        try {
            ls = new ArrayList<User>();
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery(select);
            while(rs.next()) {
                User user = new User();
                user.setId(rs.getString("id"));
                user.setNama(rs.getString("name"));
                user.setUsername(rs.getString("username"));
                user.setPassword(rs.getString("password"));
                ls.add(user);
            }
        }catch(SQLException e) {
            Logger.getLogger(UserDAO.class.getName()).log(Level.SEVERE, null, e);
        }
        return ls;
    }
    @Override
    public void update(User user) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(update);
            st.setString(1, user.getNama());
            st.setString(2, user.getUsername());
            st.setString(3, user.getPassword());
            st.setString(4, user.getId());
            st.executeUpdate();
        }catch(SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
    }
    @Override
```

```
        return rs;
    }
    @Override
    public void update(User user) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(update);
            st.setString(1, user.getNama());
            st.setString(2, user.getUsername());
            st.setString(3, user.getPassword());
            st.setString(4, user.getId());
            st.executeUpdate();
        }catch(SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
    }
    @Override
    public void delete(String id) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(delete);
            st.setString(1, id);
            st.executeUpdate();
        }catch(SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```

Semua kode meliputi instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulas database,serta method save , update dan delete

D. Kesimpulan

Fungsi CRUD merupakan komponen penting dalam pengembangan aplikasi berbasis data, karena memungkinkan pengelolaan data secara lengkap mulai dari menambah, membaca, memperbarui, hingga menghapus data.

Koneksi aplikasi Java dengan MySQL dapat dilakukan menggunakan **MySQL Connector (JDBC)** sehingga aplikasi mampu berkomunikasi dengan database.

Penerapan **DAO** membantu memisahkan logika bisnis dengan logika akses data, sehingga aplikasi menjadi lebih modular, mudah dipelihara, dan dapat digunakan kembali.

Dengan menggabungkan GUI (Swing), DAO, dan database, mahasiswa dapat membuat aplikasi sederhana berbasis Java yang interaktif serta memiliki kemampuan mengelola data user.