LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

MEMBUAT FUNGSI CRUD USER DENGAN DATABASE MYSQL

OLEH :

DEDE BINTANG GAFENDI

2411533010

DOSEN PENGAMPU:

NURFIAH, S.ST, M.Kom..,

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG 2025

PENDAHULUAN

XAMPP adalah paket software open-source yang terdiri dari Apache, MySQL, PHP, dan Perl, biasa digunakan sebagai development environment untuk pengembangan aplikasi web di localhost. Apache berfungsi sebagai web server, MySQL sebagai sistem manajemen basis data, sedangkan PHP sebagai bahasa pemrograman web.

MySQL merupakan RDBMS open-source yang menyimpan, mengelola, dan mengambil data dalam bentuk tabel. Untuk menghubungkan aplikasi Java dengan MySQL digunakan MySQL Connector/J, yang berfungsi membuka koneksi, mengirim query, menerima hasil, dan menutup koneksi.

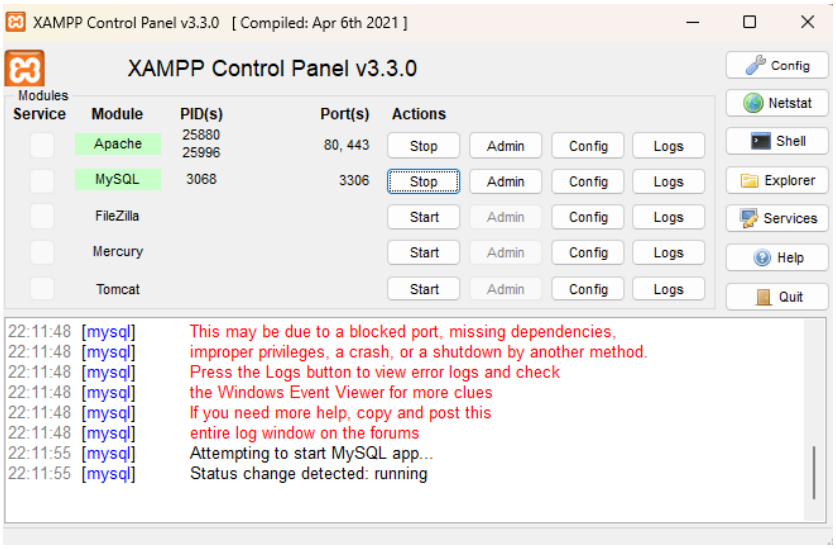
DAO (Data Access Object) adalah objek yang menyediakan antarmuka abstrak untuk operasi database seperti create, read, update, dan delete (CRUD). Penggunaan DAO bertujuan meningkatkan modularitas, reusabilitas, serta memisahkan logika akses data dengan logika bisnis. Dalam Java, interface mendefinisikan metode abstrak yang wajib diimplementasikan oleh kelas, sedangkan CRUD merupakan fungsi dasar aplikasi untuk mengelola data dalam database.

TUJUAN

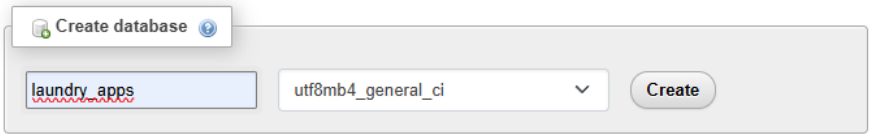
- Mahasiswa dapat merancang tabel **user** pada database MySQL.
- Mahasiswa dapat membangun koneksi antara aplikasi Java dan MySQL.
- Mahasiswa dapat membuat tampilan GUI untuk operasi CRUD pada data user.
- Mahasiswa dapat mendesain serta mengimplementasikan **interface** dalam Java.
- Mahasiswa dapat membuat dan menerapkan fungsi **DAO (Data Access Object)**.
- Mahasiswa dapat mengimplementasikan fungsi **CRUD** dengan konsep Pemrograman Berorientasi Objek

LANGKAH KERJA

1. Unduh dan instal XAMPP pada komputer/laptop.
2. Jalankan XAMPP lalu aktifkan modul *Apache* dan *MySQL*.

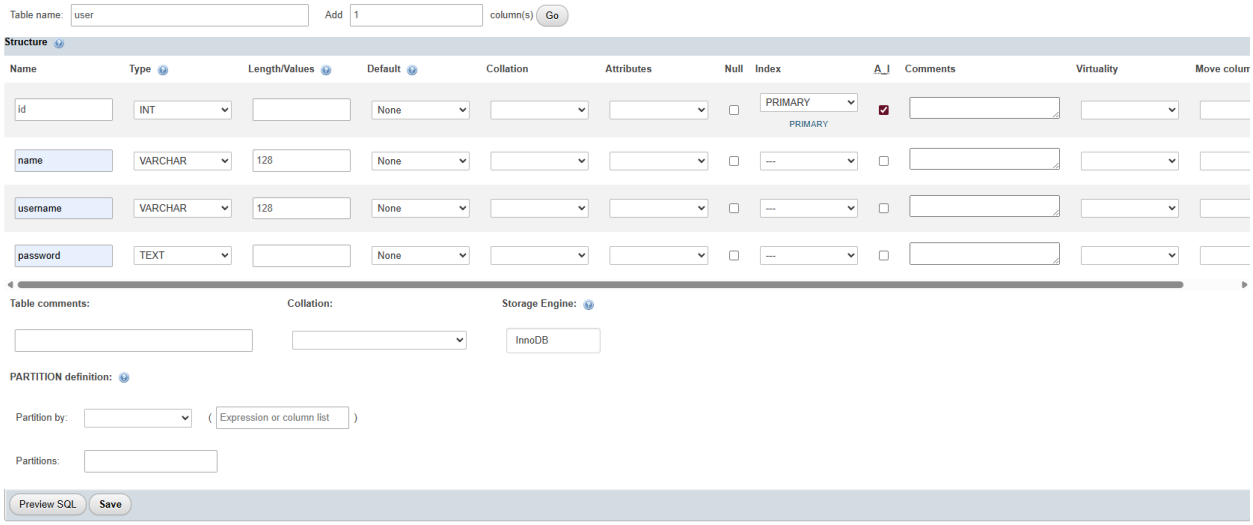


3. Buat database baru dengan nama laundry_apps.

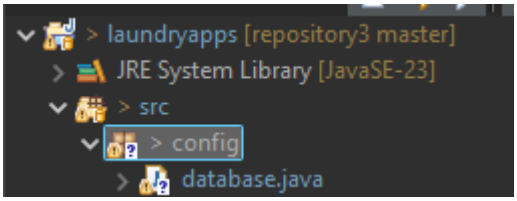


4. Di dalam database tersebut, buat tabel dengan nama user.

5. Isi tabel dengan beberapa data awal untuk pengujian

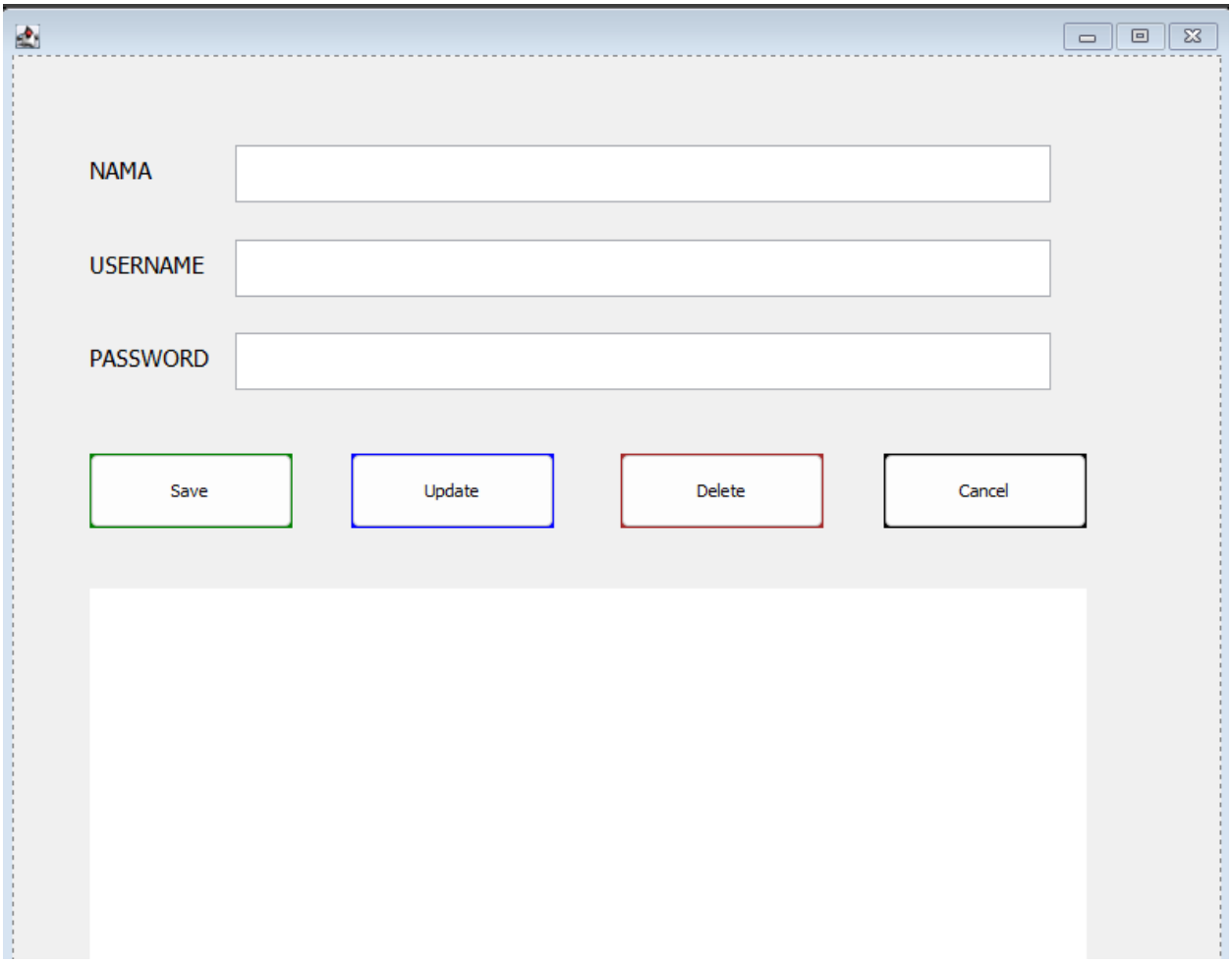


6. Buat package baru dengan nama config. Tambahkan class Database untuk menyimpan konfigurasi koneksi. kemudian konfigurasi sesuai dengan kode program

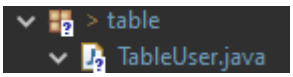



```java
package config;

import java.sql.*;

public class database {

    Connection conn;
    public static Connection koneksi()  {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps" ,
                    "root", "");
            return conn;
        }   catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
            return null;
        }
    }

}
```

7. Buat file baru dengan JFrame pada package ui dengan nama UserFrame untuk menampilkan antarmuka CRUD.



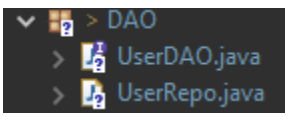
8. Buat package baru bernama table. Tambahkan file TableUser berisi kode program untuk mengatur model data tabel.



```java
package table;

import java.util.List;
import javax.swing.table.AbstractTableModel;
import model.User;

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;
        }
    }
}
```

9. Buat package baru bernama DAO. Tambahkan interface UserDAO yang mendefinisikan method save, show, update, dan delete. Interface ini bersifat kontrak, sehingga setiap class yang menggunakannya wajib mengimplementasikan method-method tersebut.

10. Buat class UserRepo pada package DAO.



11. Gunakan keyword implements untuk mengimplementasikan interface UserDAO. Lakukan inisialisasi Connection, buat konstruktor, serta definisikan query SQL. Implementasikan method:

```
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();
    }
```

Membuat method save, isikan dengan kode program

```
@Override
public void save(User user) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(insert);
        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();
        }
    }
}
```

Membuat method show untuk mengambil data dari database

```
@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;
}
```

Membuat method update yang digunakan untuk mengubah data

```java
@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();
        }
    }
}
```

Membuat method delete yang digunakan untuk menghapus data

```java
@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();
        }
    }
}
```