

LAPORAN PRAKTIKUM  
PEMOGRAMAN BERORIENTASI OBJEK



OLEH:  
TIARA AZIZAH  
(2411533001)

DOSEN PENGAMPU:

FAKULTAS TEKNOLOGI INFORMASI  
DEPARTEMEN INFORMATIKA  
UNIVERSITAS ANDALAS

2025

## A. Pendahuluan

Database adalah kumpulan data yang terorganisasi dan terstruktur yang memiliki hubungan satu sama lain. Data-data ini disimpan secara elektronik dalam sistem komputer untuk memudahkan pengelolaannya.

XAMPP adalah paket software yang bersifat open source. XAMPP digunakan untuk menjalankan halaman web, MySQL, web yang bisa manajemen basis data sehingga data bisa dimanipulasi.

MySQL merupakan relational database management system (RDBMS) open source. MySQL bisa digunakan untuk memanipulasi data seperti mengelola dan mengambil data dalam bentuk format table.

MySQL Connection/j adalah driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL, sehingga bisa berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data melalui java.

DAO(Data Acces Object) adalah objek yang menyediakan abstrak interface terhadap beberapa method yang berhubungan dengan database, seperti mengambil data(read), menyimpan data(create), menghapus (delete), mengubah (update)

Interface dalam bahasa java berarti mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan digunakan.

CRUD(Create, Read, Update, Delete) adalah fungsi dasar yang ada pada sebuah fungsi di database.

## B. Tujuan

Pratikum ini bertujuan untuk membantu mahasiswa agar mampu membuat fungsi CRUD data user menggunakan database MySQL melalui Java.

## C. Langkah-Langkah Pratikum

### 1. Menambahkan MySQL Connector

- Langkah pertama adalah menambahkan MySql Connector ke dalam project Java. Klik directory JRE System Library, Built Path, Configure Build Path. Setelah itu pilih libraries dan classpath.
- Untuk menambahkan MySQL Connector, caranya klik Add External JARs dan pilih file yang sudah di download. Jika sudah, pilih Apply and Close.

### 2. Membuat Database

- Gunakan XAMPP, MySQL, untuk membuat database dengan nama laundry\_apps.
- Create sebuah table dengan nama table user. Isikan tipe data sesuai kebutuhan, kemudian klik save, agar table tersimpan.

### 3. Membuat Koneksi Database

- Selanjutnya, membuat koneksi ke database.
- Buat package baru dengan nama config, yang akan berisikan konfigurasi aplikasi yang akan dibuat, seperti konfigurasi database.

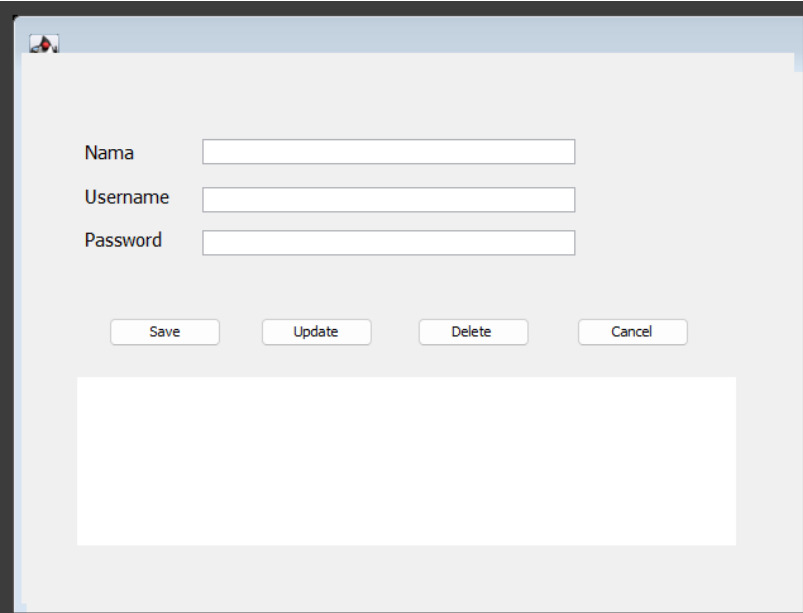
- c. Buat class baru dengan nama database, untuk mengkonfigurasi buat kode program di bawah ini. Program di bawah ini menggunakan blok kode try-catch, berfungsi untuk menguji blok kode yang salah

```
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;
        }
    }
}
```

#### 4. Tampilan CRUD User

- a. Buat kelas baru di package ui
- b. Desain frame seperti gambar di bawah ini. Elemen yang digunakan adalah JTextField, JButton, dan JTable.



## 5. Tabel Model

- Langkah selanjutnya membuat table model dengan cara menambahkan package baru dengan nama table.
- Di dalam package table isikan class bernama TableUser. Kelas ini akan berfungsi sebagai model tabel pengguna.
- Setelah membuat class baru, isikan program yang bisa menghubungkan data user yang disimpan di List<User> dengan visual tabel di package UI, seperti program di bawah ini. Kode program ini juga akan mengatur visual dan ukuran pada table yang akan ditampilkan.

```
package table;

import java.util.List;

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() {
        return ls.size();
    }
    @Override
    public int getColumnCount() {
        return 4;
    }
    @Override
    public String getColumnName(int column) {
        return columnNames [column];
    }
    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        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;
        }
    }
}
```

## 6. Membuat Fungsi DAO

- Buat package baru dengan nama DAO dan tambahkan class baru bernama UserDAO.
- Isi class UserDAO dengan interface yang berfungsi sebagai pendefinisian metode yang harus diimplementasikan oleh class yang menggunakannya.

```
package DAO;

import java.util.List;

import model.User;

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

- Selanjutnya tambahkan class baru di dalam package, dengan nama UserRepo. Untuk program yang akan diisi di dalam class ini adalah kode program yang akan mengimplementasikan program dari class UserDAO. UserRepo akan mengelola semua program CRUD yang akan digunakan.

```

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 ="DELET from user where id=?;";
    final String update ="UPDATE user SET name=?, username=?, password=? WHERE id=?;";
}

```

Program ini adalah variabel konstan yang menyimpan semua perintah SQL.

```

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

```

Kode ini adalah konstruktor yang digunakan dalam class UserRepo. Berfungsi untuk memanggil objek database.koneksi dan akan disimpan di variabel connection.

```

@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();
        }
    }
}

```

Method ini akan mengambil objek User sebagai parameter dan menyimpan data yang akan diinputkan dengan menggunakan perintah insert. Blok try-catch-finally digunakan untuk menangani kesalahan dan memastikan resource ditutup.

```

5 * @Override
6 public List<User> show(){
7     List<User> ls=null;
8     try {
9         ls = new ArrayList<User>();
10        Statement st = connection.createStatement();
11        ResultSet rs =st.executeQuery(select);
12        while(rs.next()) {
13            User user = new User();
14            user.setId(rs.getString("id"));
15            user.setNama(rs.getString("name"));
16            user.setUsername(rs.getString("username"));
17            user.setPassword(rs.getString("password"));
18            ls.add(user);
19        };
20    }
21 }

```

Method show akan menampung sebuah arraylist dari objek user. Perintah ini akan dijalankan menggunakan select untuk menampilkan semua data dari tabel pengguna.

```

@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();
        }
    }
}

```

Method update akan mengambil objek yang diperbarui dari User. PreparedStatement yang digunakan adalah update, mengatur data baru diinputkan dengan ID sebagai nilai unik untuk menentukan baris nilai yang akan diperbarui.

```

    }
    @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();
            }
        }
    }
}

```

Method delete akan menghapus data dengan perintah PreparedStatementnya adalah delete. Id digunakan sebagai nilai unik yang akan menentukan baris data yang akan dihapus.