Laporan Praktikum Pekan 2

Pemrograman Berorientasi Objek: Membuat Fungsi CRUD (Create, Read, Update, Delete) User dengan Database MySQL

Disusun Oleh:
Nama: Fadil Insanus Siddik
NIM: 2411532013

Dosen Pengampu:
d

Program Studi Informatika
Fakultas Teknologi Informasi
Universitas Andalas

Padang, 2025

A.PENDAHULUAN

1. Tujuan Praktikum

   Tujuan dari praktikum ini yaitu, kita sebagai mahasiswa mampu membuat fungsi CRUD data user menggunakan database MySQL. Adapun poin-poin praktikum yaitu:

   - Kita sebagai mahasiswa mampu membuat table user pada database MySQL
   - Kita sebagai mahasiswa mampu membuat koneksi Java dengan database MySQL
   - Kita sebagai mahasiswa mampu membuat tampilan GUI CRUD user
   - Kita sebagai mahasiswa mampu membuat dan mengimplementasikan interface
   - Kita seabgai mahasiswa mampu membuat funngsi DAO (Data Access Object) dan mengimplementasikannya.
   - Kita sebagai mahasiswa mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek

2. Alat yang diperlukan

   Adapun alat yang diperlukan untuk melaksanakan praktikum ini yaitu:

   - Computer / laptop yang telah terinstall JDK dan Eclipse
   - MySQL / XAMPP
   - MySQL connector atau Connector /J

3. Teori

   XAMPP yaitu paket software yang terdiri dari Apache HTTP Server, MySQL, PHP dan Perl yang bersifat open source. XAMPP biasanya digunakan sebagai development environment dalam pengembangan aplikasi berbasis web secara localhost.

   Apache berfungsi sebagai web server yang digunakan untuk menjalankan halaman web, MySQL digunakan untuk manajemen basis data dalam melakukan manipulasi data, PHP digunakan sebagai Bahasa pemrograman untuk membuat aplikasi berbasis web.

   MySQL merupakan sebuah relational database management system (RDBMS) open-source yang digunakan dalam pengelolaan database suatu aplikasi, MySQL ini dapat digunakan untuk menyimpan, mengelola dan mengambil data dalam format table.

Logo MySQL

MySQL Connection/J adalah driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data. Beberapa fungsi MySQL connector yaitu:

- Membuka koneksi ke databse MySQL
- Mengirimkan permintaan SQL ke server MySQL
- Menerima hasil dari permintaan SQL
- Menutup koneksi ke database MySQL

DAO (Data Access Object) merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data (create), mengahpus data (delete), mengubah data (update). Tujuan penggunaan DAO yaitu:

- Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga memudahkan untuk dikelola
- Meningkatkan reusabilitas yaitu DAO dapat digunakan kembali
- Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis

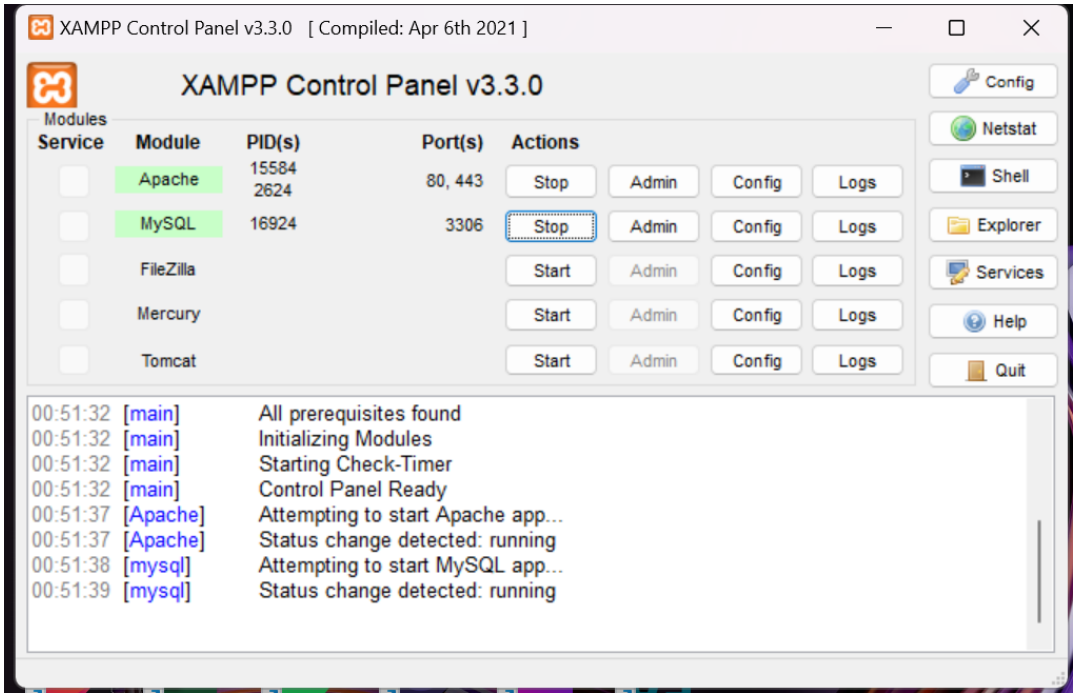
Interface dalam Bahasa Java yaitu, mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya.

CRUD (Create, Read, Update, Delete) merupakan fungsi dasar atau umum yang ada pada sebuah aplikasi yang mana fungsi ini dapat membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

4. Langkah-langkah pengerjaan

Install XAMPP

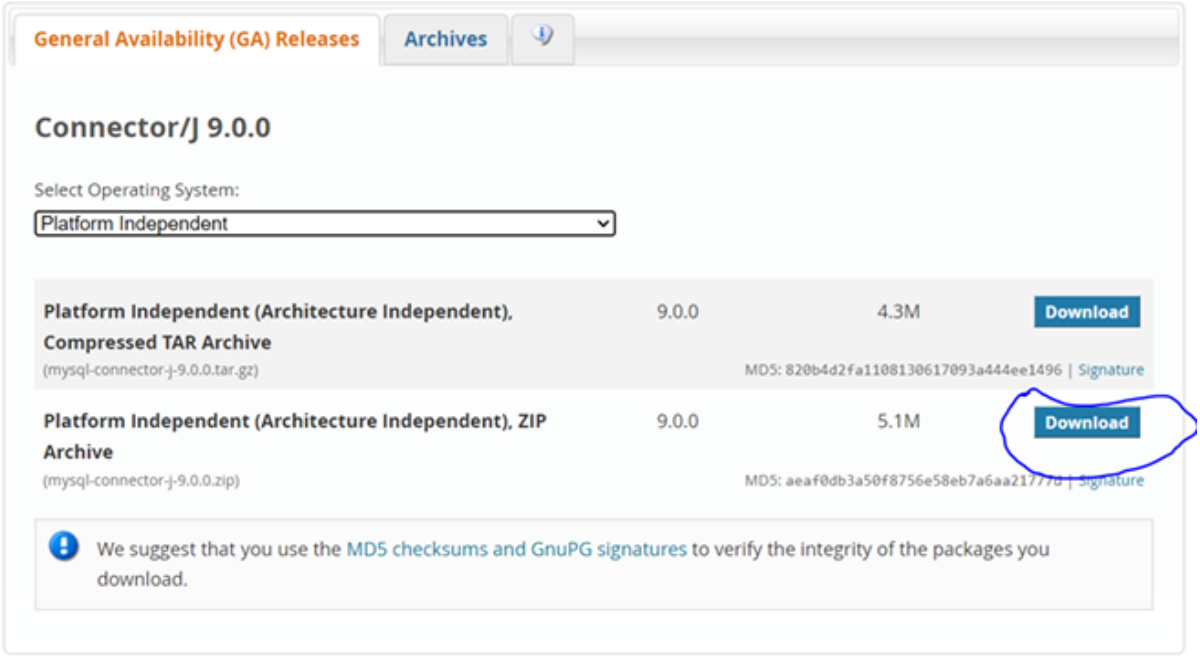
- Download XAMPP dari link berikut: https://www.apachefriends.org/
- Setelah didownload, install XAMPP pada computer / laptop masing-masing
- Jalankan XAMPP dan aktifkan Apache dan MySQL



Menambahkan MySQL Connector

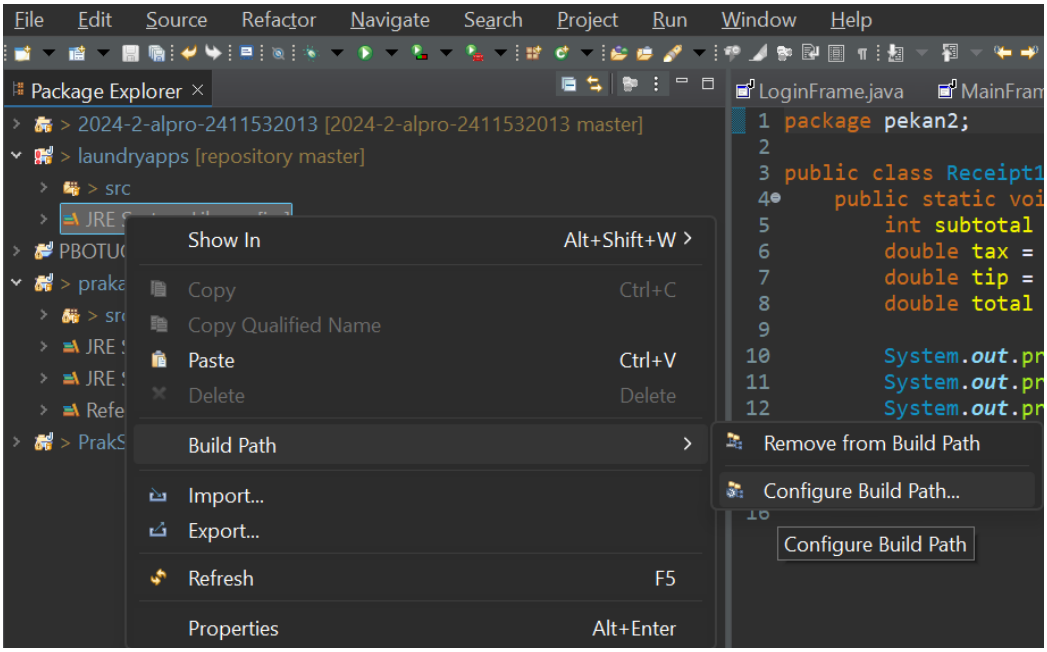
Aplikasi java agar dapat terhubung dengan database MySQL membutuhkan sebuah driver yaitu MySQL Connection, berikut langkah-langkah membuat koneksi Database MySQL:

- Download MySQL connection pada link berikut https://dev.mysql.com/downloads/connector/j/

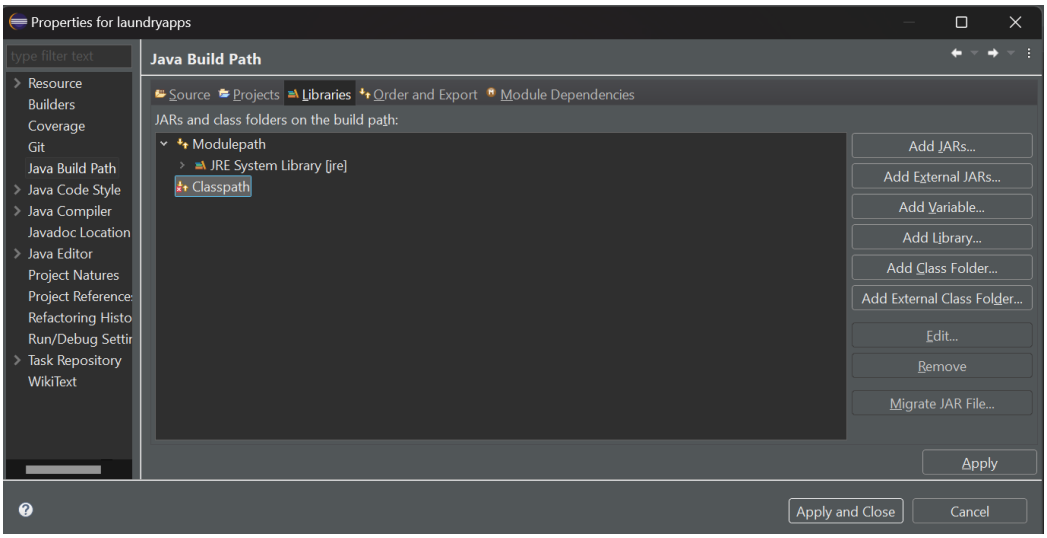


Pilih file yang berekstensi .zip

- Menambahkan MySQL Connector ke dalam project dengan cara klik kaan directory **JRE System Library → Built Path → Configure Build Path**

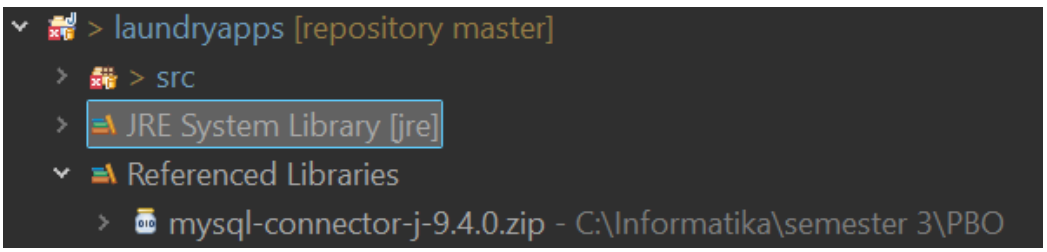


Selanjutnya pilih **Libraries → Classpath**



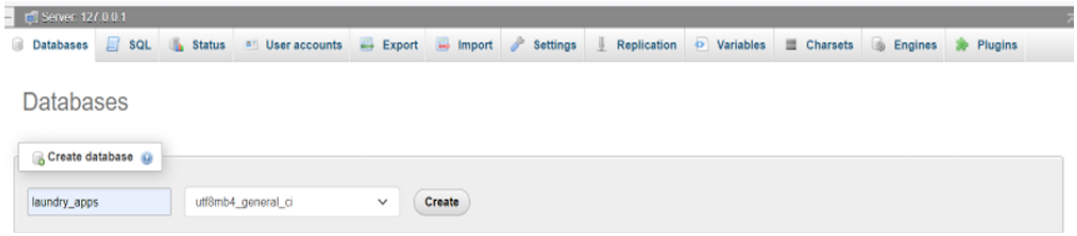
Tambahkan file MySQL Connector dengan cara klik **Add External JARs** dan pilih file yang telah didownload dan pilih **Appli and Close.**

Jika berhasil menambahkan MySQL Connector, maka akan generate folder Referenced Libraries pada project yang berisi MySQL Connector.

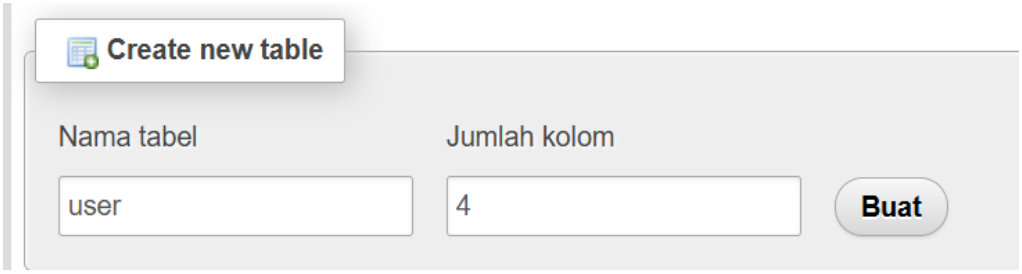


Membuat Database dan Table User

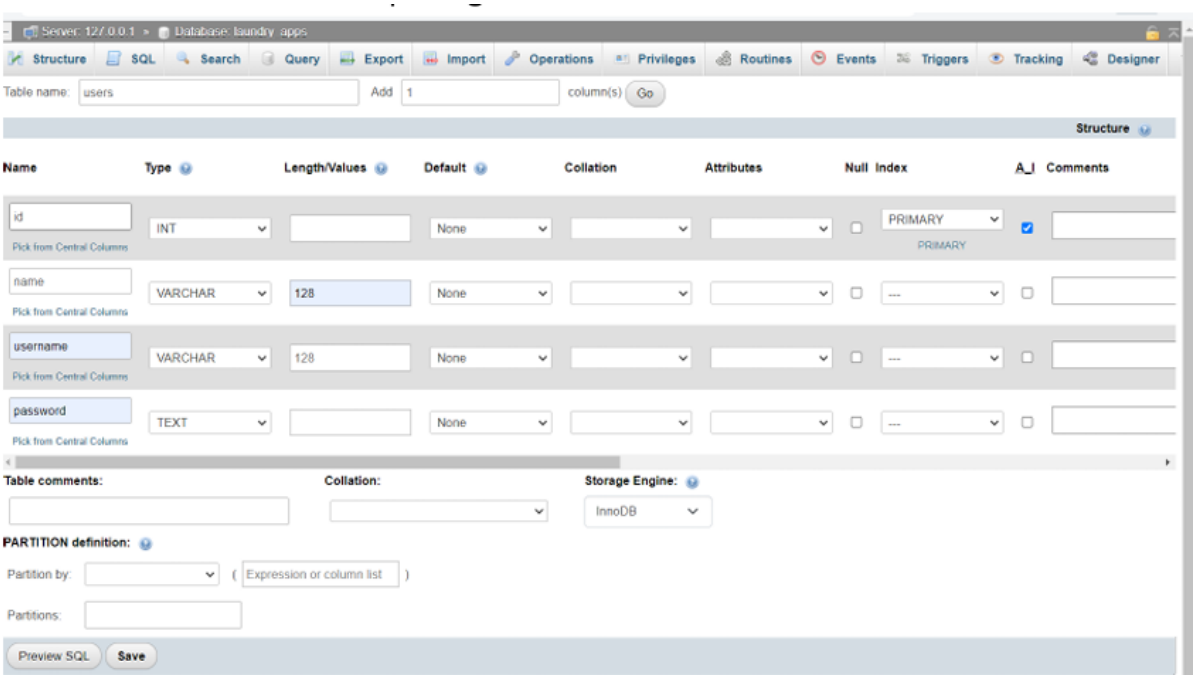
- Buka http://localhost/phpmyadmin
- Klik new dan buat database dengan nama laundry_apps



- Buat table user dengan cara klik database laundry_apps dan buat table dengan nama user



Klik create maka akan muncul seperti gambar di bawah ini.



Isi seperti gambar di atas dan klik save

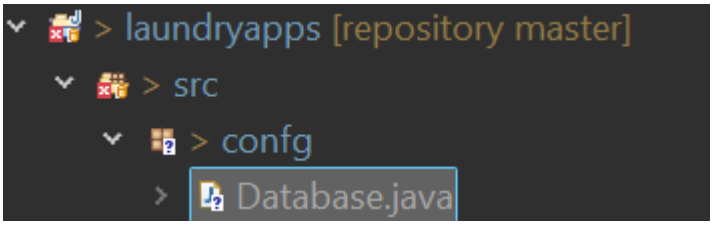
- Cara lain membuat table user pada database laundry_apps, klik SQL pada phpMyAdmin dan ketikkan SQL seperti ambar di bawah ini

```
CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `username` varchar(64) NOT NULL,
  `password` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Membuat koneksi ke Database MySQL

Setelah berhasil menambahkan MySQL Conector maka dapat embuat koneksi ke daabase MySQL, berikut langkah-langkahnya.

- Buat package baru dengan nama config, package ini yang akan digunaka untuk membuat konfigurasi aplikasi yang akan dibuat, termasuk dengan konfigurasi database



- Buat class baru dengan nama Database, kemudian konfigurasi dengan kode program berikut

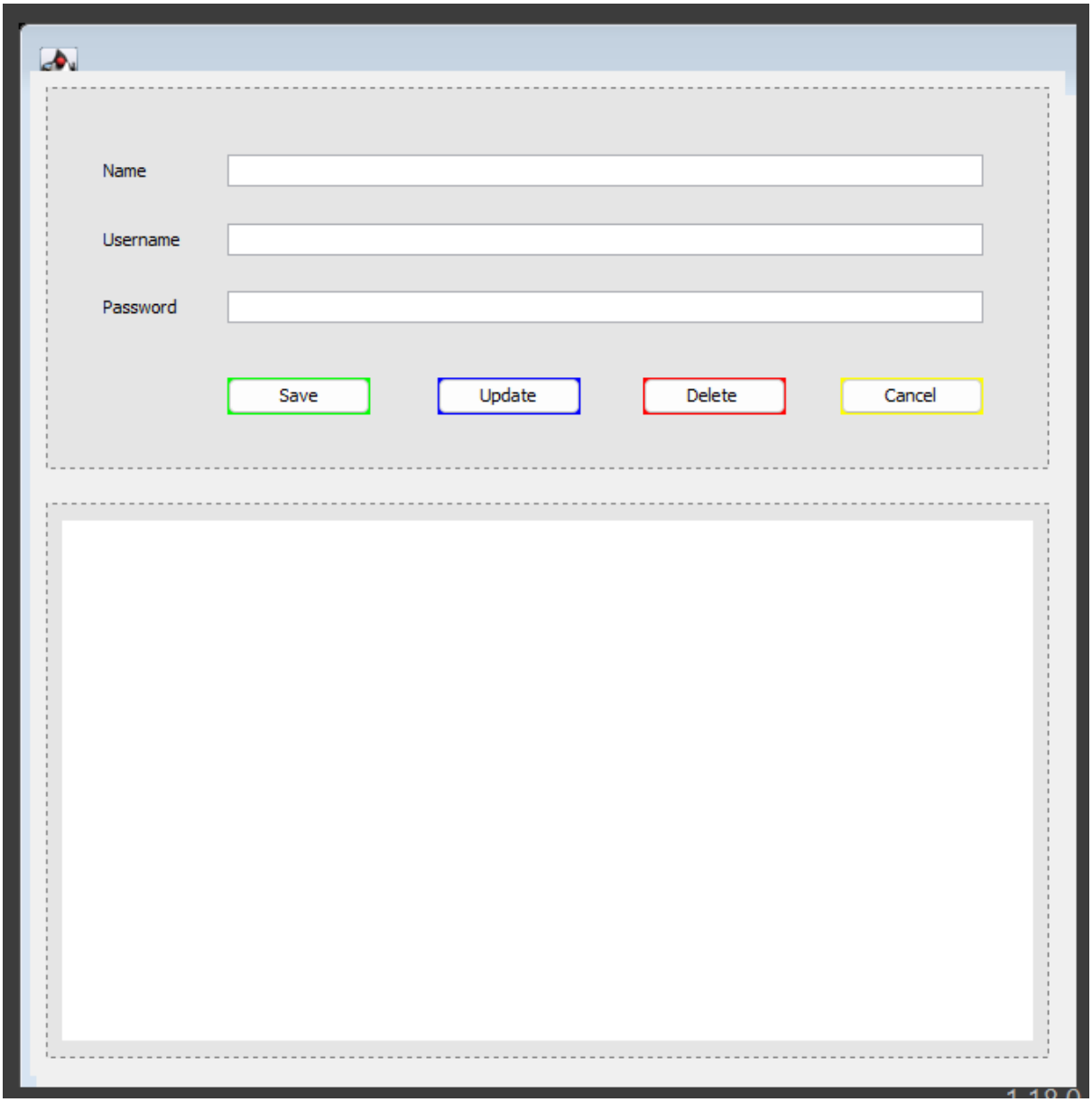
```
1  package confg;
2
3  import java.sql.*;
5
6  public class Database {
7      Connection conn;
8      public static Connection koneksi() {
9          try {
10             Class.forName("com.mysql.cj.jdbc.Driver");
11             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps","root","");
12         return conn;
13     }catch(Exception e) {
14         JOptionPane.showMessageDialog(null,  e);
15         return null;
16         }
17     }
18
19
20 }
```

Penejelasan:

- Import java.sql.* digunakan untuk import seluruh fungsi-fungsi SQL
- Line 8 membuka method Connection dengan nama koneksi, yangmana method ini akan digunakan untuk membuka koneksi ke database
- Line 10-13 membuat koneksi ke database, jika koneksi berhasil maka akan mengemablikan nilai Connection
- Lie 15-16 jika koneksi gagal maka akan ditampilkan pesan error menggunakan JOptionPane.

Membuat Tampilan CRUD User

- Buat file baru menggunakan JFrame pada package ui dengan nama Userframe seperti berikut



Keterangan:

Component	Variable	Keterangan
JTextFiel	txtName	Name
JTextFiel	txtUsername	Username
JTextFiel	txtPassword	Password
JButton	btnSave	Save
JButton	btnUpdate	Update
JButton	btnDelete	Delete
JButton	btnCancel	Cancel
JTable	tableUsers	Table Users

Membuat Table Model

Table model user ini berguna untuk mengambil data dari database dan ditampilkan ke dalam table

- Buat package baru dengan nama **table**
- Buat file baru di dalam package table dengan nama **TableUSer**, kemudian isikan dengan kode program berikut

```java
package table;

import java.util.*;
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;
        }
    }
}
```

Membuat Fungsi DAO

- Buat package baru dengan nama DAO
- Buat class interface baru dengan nama **UserDAO**, kemudian isikan dengan kode program berikut.

```java
package DAO;

import model.User;
import java.util.List;

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

Terdapat method **save, show, delete** dan **update**. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya

Mengunakan Fungsi DAO

- Buat class baru pada package DAO dengan nama UserRepo yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat
- Implementasikan UserDAO dengan kata kunci implements

```java
public  class UserRepo implements UserDAO {
```

- Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulas database

```java
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 berikut

```java
@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

```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;
}
```

- 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();
        }
    }
}}
```