LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete) USER DENGAN DATABASE MYSQL

Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU :

Nurfiah, S.ST., M.Kom

PROGRAM STUDI S1-INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

SEPTEMBER 2025

A. Pendahuluan

Praktikum ini dilakukan untuk membuat database aplikasi loundry dengan menerapkan CRUD (Create, Read, Update, Delete) dengan membuat database localhost mneggunakan XAMPP sebagai server, lalu dihubungkan ke java project loundry apps lewat MySQL Connector/J. Menerapkan konsep DAO (Data Access Object) dan Interface unutk logika akses data dan interface/GUI pada aplikasi dengan menerapkan konsep Pemrograman berorientasi objek pada java.

1. XAMPP
   Merupakan paket software teridir dari Apache HTTP Server, MySQL,PHP dan Perl yang bersifat open source, xampp biasanya digunakan sebagai deleveopment 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.

2. MySQL
   Merupakan sistem manajemen basis data relasional (RDBMS) open-source yang menggunakan Structured Query Language (SQL) sebagai bahasa untuk mengelola dan memanipulasi data dalam database. SQL adalah bahasa yang digunakan untuk melakukan berbagai operasi seperti menyimpan, mengambil, memperbarui, dan menghapus data dalam basis data. Sistem ini juga mendukung operasi CRUD (Create, Read, Update, Delete) untuk memanipulasi data dalam format table.

3. MySQL Connection/j
   Merupakan driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data. Berfungsi :
   - Membuka koneksi ke database MySQL
   - Mengirimkan permintaan SQL ke server MySQL
   - Menerima hasil dari permintaan SQL
   - Menutup koneksi ke database MySQL

4. DAO (Data Access Object)
   merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data(create), menghapus data (delete), mengubah data(update). Bertujuan unutk :
   - Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga
   - Meningkatkan reusabilitas yaitu DAO dapat digunakan Kembali
   - Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis.

5. Interface (Antarmuka)
   Mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya. Merupakan blue print dari class, berfungsi

menghubungkan antar objek. Interface bersifat abstrak, sehingga interface tidak bisa dibuat objek instance dengan kunci “new”. Interface menerapkan prinsip Abstraksi dan Enakpsulasi.

6. CRUD (Create, Read, Update, Delete)
   Merupakan fungsi dasar aplikasi yang berfungsi unutk membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

1. Mampu membuat table user pada database MySQL
2. Mampu membuat koneksi Java dengan database MySQL
3. Mampu membuat tampilan GUI CRUD user
4. Mampu membuat dan mengimplementasikan interface
5. Mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikan
6. Mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek.

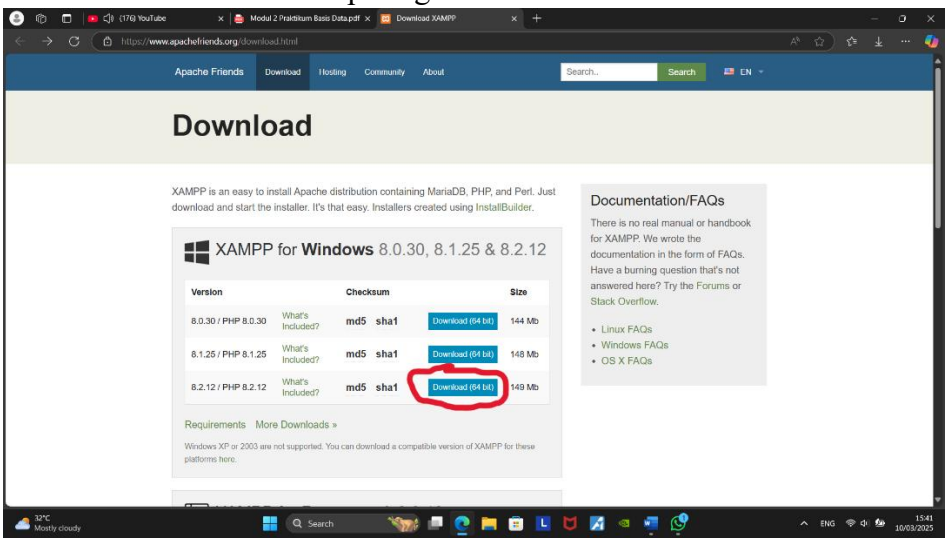
C. Langkah kerja praktikum

a. Alat

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integreted Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)
5. MySQL & XAMPP
6. MySQL connector atau Connector/J

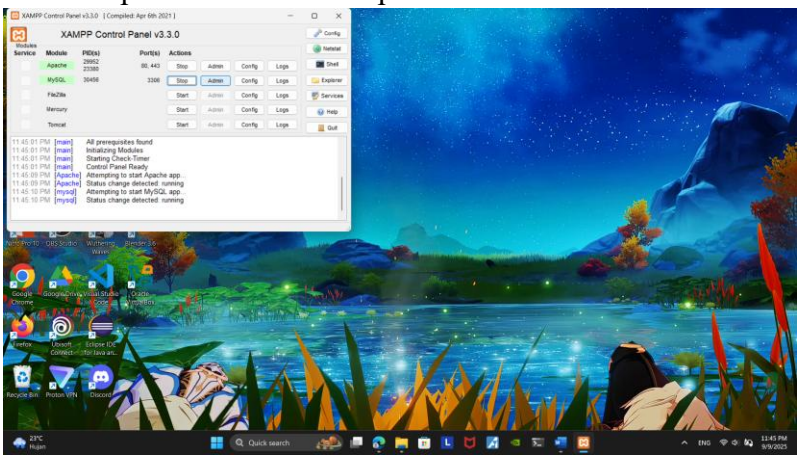
b. Instalasi XAMPP

1. Search di browser https://www.apachefriends.org/download.html, setelahnya akan muncul lawab web seperti gambar dibawah. Donwload versi terbarunya..

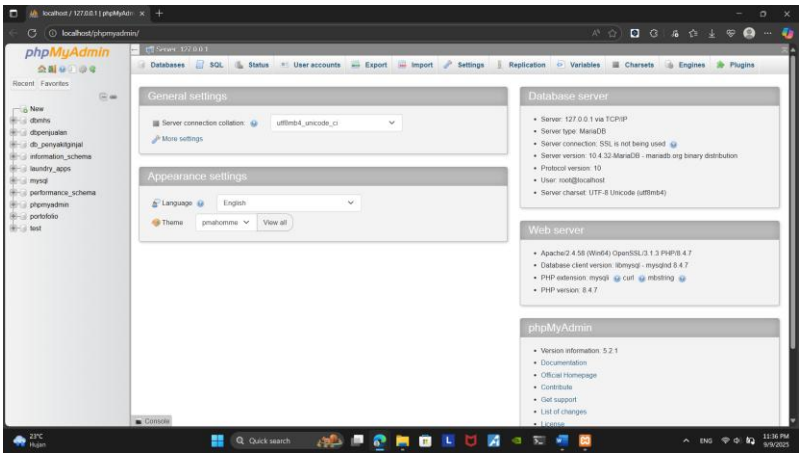


2. Setelah unduhan selesei, lakukan instalasi paket XAMPP dan pilih directory, ikuti sampai finish instalasi dan XAMPP bisa dijalankan.

3. Jalankan XAMPP, aktifkan Apache dan MySql dengan cara tekan tombol start. Kemudian tekan “admin” pada MySQL unutk membuka localhost database pada browser atau open browser lalu ketik ‘localhost/phpmyadmin’.

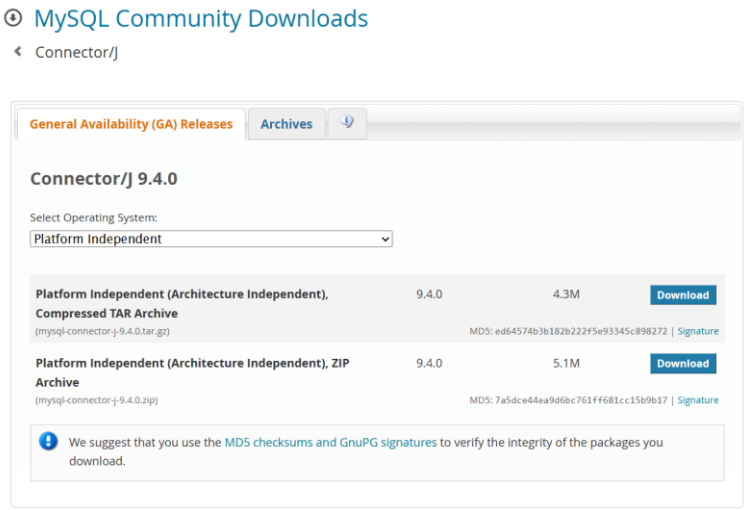


4. Berhasil memulai server database secara localhost.

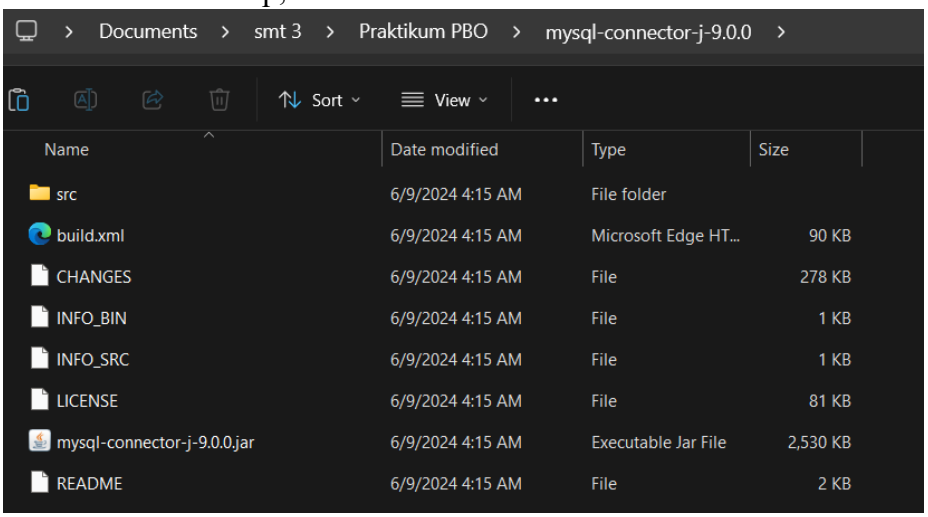


c. Menambahkan MySQL Connector

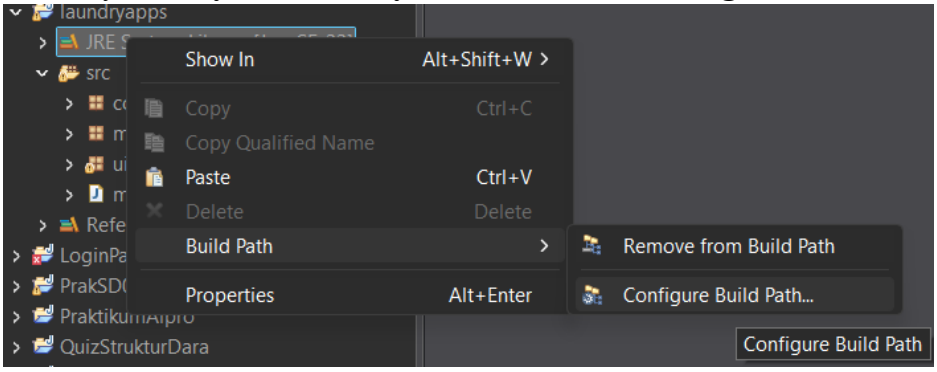
1. Donwload MySQL connection pada link berikut https://dev.mysql.com/downloads/connector/j/, pilih platform independent, lalu donwload ZIP archive. Diperlukan agar aplikasi java dapat terhubung ke database MySQL menggunakan driver MySQL Connection.



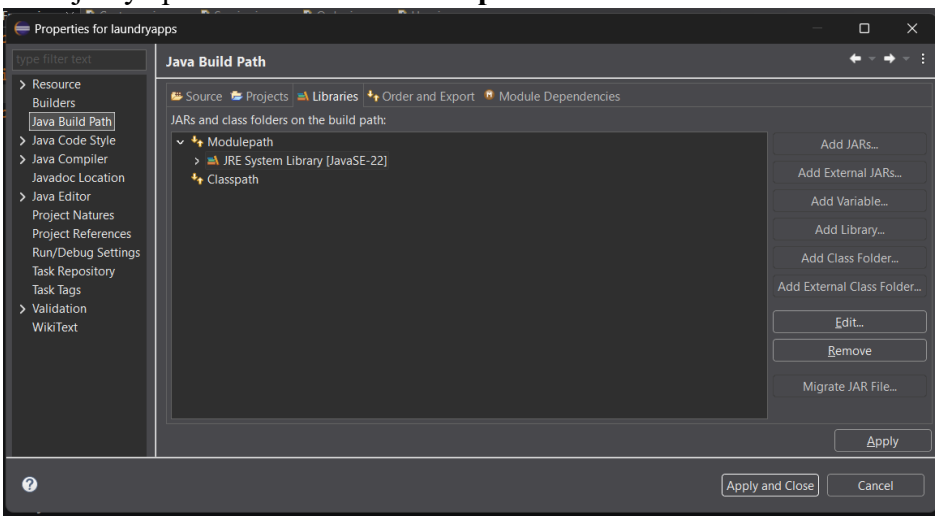
Setelah download zip, ektrak file terlebih dahulu.



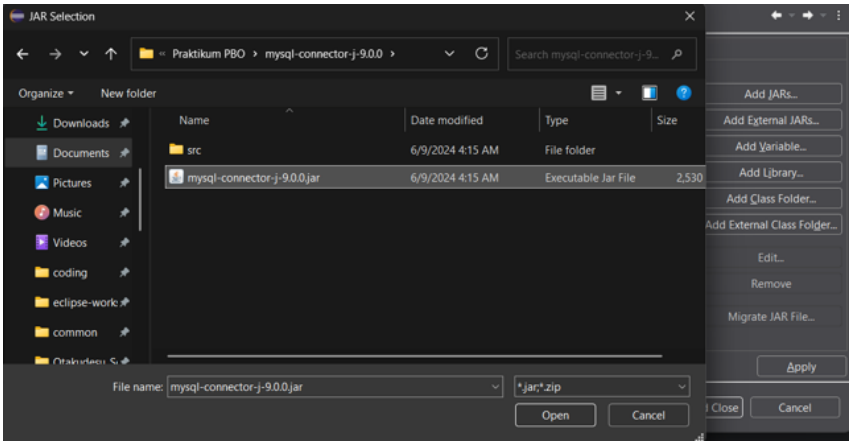
2. Tambahkan MySQL Connector kedalam project dengan cara klik kanan directory **JRE System Library → Built Path → Configure Build Path**.



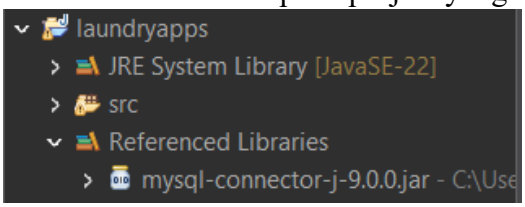
3. Selanjutnya pilih **Libraries → Classpath.**



Pada option menu kanan klik “Add External JARs”, lalu pilih file dengan extension jar, lalu apply dan close.

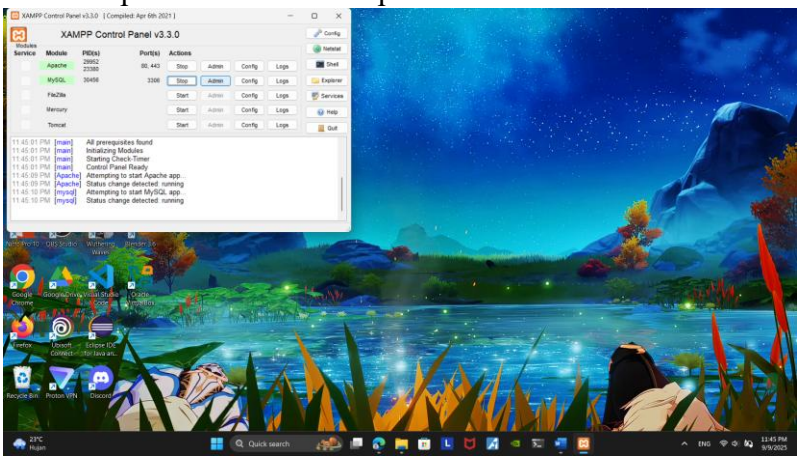


4. Setelah berhasil menambahkan MySQL Connector, dapat terlihat ada folder Referenced Libraries pada project yang berisi MySQL Connector.

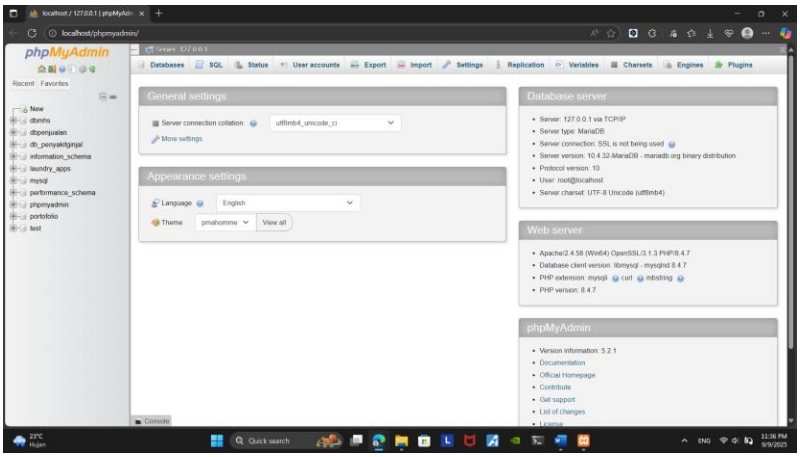


d. Membuat Database dan Table User

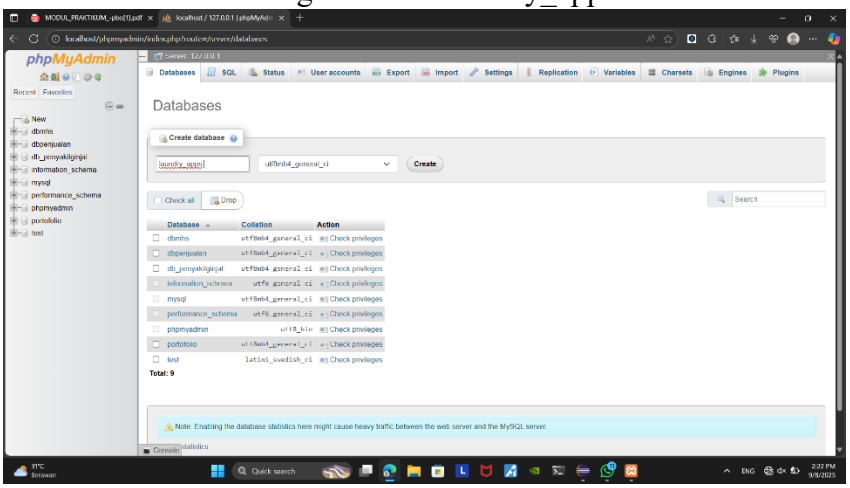
1. Jalankan XAMPP, aktifkan Apache dan MySql dengan cara tekan tombol start. Kemudian tekan “admin” pada MySQL unutk membuka localhost database pada browser atau open browser lalu ketik ‘localhost/phpmyadmin’.



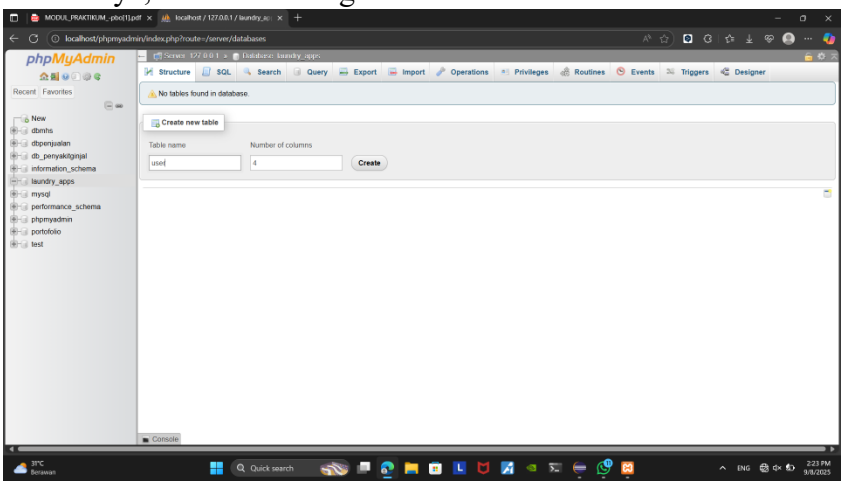
2. Berhasil memulai server database secara localhost.



3. Buat new database dengan nama “laundry_apps”.



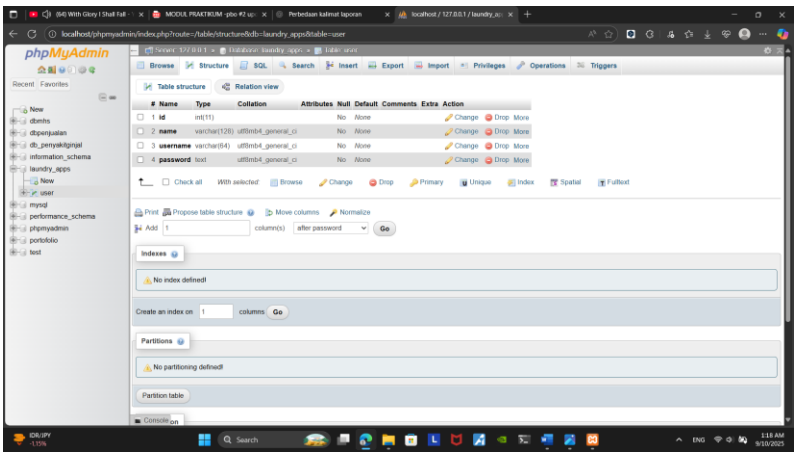
4. Setelahnya, buat table dengan nama “user”.



5. Klik SQL pada tab diatas, lalu masukan SQL Query berikut unutk membuat field/colum dengan tipe datanya.

```
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;
```

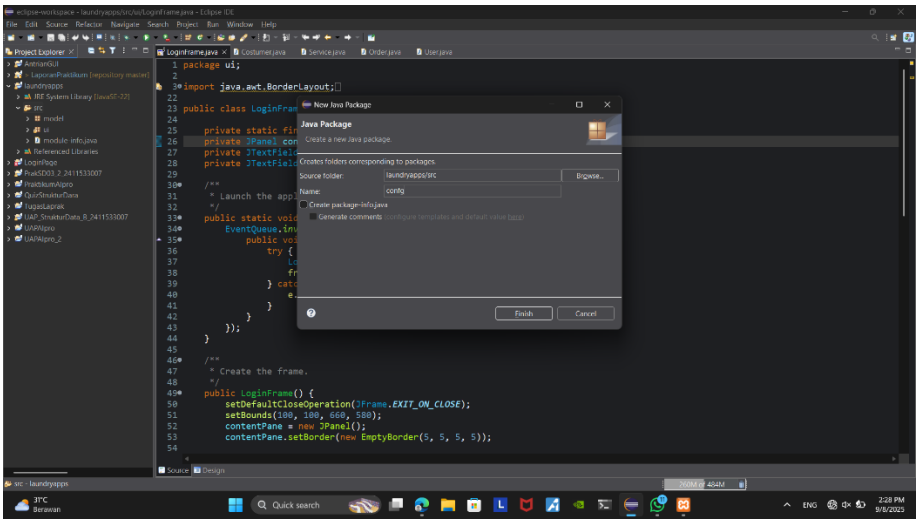
6. Database secara localhost server XAMPP berhasil dibuat.



Penjelasan : Membuat database localhost menggunakna XAMPP unutk membuat tabel user yang nantinya menyimpan data berupa id, nama, username dan password untuk login ke dalam aplikasi loundry pada interface/halaman login.

e. Membuat Koneksi ke Database MySQL

1. Setelah berhasil include MySQL connector ke dalam project loundry apps, agar dapat terhubung dengan database MySQL.
2. Buat package baru dengan nama config didalam package src, digunakan unutk membuat konfigurasi aplikasi/logika program/method unutk terhubung ke database.



3. Di dalam package confg, buat new class “Database”, lalu tuliskan kode program berikut.

*Database.java

```
1  package confg;
2
3  import java.sql.*;
4  import javax.swing.JOptionPane;
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
12                     ("jdbc:mysql://localhost/laundry_apps", "root", "");
13             System.out.println("Berhasil koneksi");
14             return conn;
15             }catch(Exception e) {
16             JOptionPane.showMessageDialog(null, e);
17             return null;
18         }
19     }
20
21     public static void main (String[] args) {
22         Database.koneksi();
23     }
24 }
```

Class ini berfungsi unutk membuka koneksi ke database MySQL, menangkap jika terjadi error, sehingga program dapat berjalan dengan baik.

4. Penjelasan kode :

*Database.java

```
1 package confg;
2
3 import java.sql.*;
4 import javax.swing.JOptionPane;
```

Baris 1, file berada pada package confg.
Baris 3, import dari java library unutk akses semua class dari java.sql
Baris 4, import javax librabry class JoptionPane unutk fitur pada GUI yang memungkikan pop up notifikasi pada gui

```
6 public class Database {
7     Connection conn;
```

Class bernama Database bertipe public yang dapat diakses dari package lain. Deklarasi variabel "conn" dengan tipe (interface) "connection", berfungsi untuk menghubungkan koneksi ke database. Connection merupakan interface dari package java.sql

```
8     public static Connection koneksi() {
```

Membuat method bernama "koneksi" dengan modifier public static, jadi dapat digunakan tanpa membuat objek database dan bisa diakses dari luar package. Method ini berfungsi untuk membuat koneksi ke database MySQL.

```
9         try {
```

Try statement unutk menangkap jika terjadi eror yang nantinya ditangani oleh catch statement.

```
10            Class.forName("com.mysql.cj.jdbc.Driver");
```

Memanggil driver MySQL JDBC, untuk memberitahu JVM unutk memuat class driver MySQL dapat digunakan.

```
11            Connection conn = DriverManager.getConnection
12                    ("jdbc:mysql://localhost/laundry_apps", "root", "");
```

Memuat koneksi ke database dengan dengan mengembalikan objek connection.

```
13            System.out.println("Berhasil koneksi");
14            return conn;
```

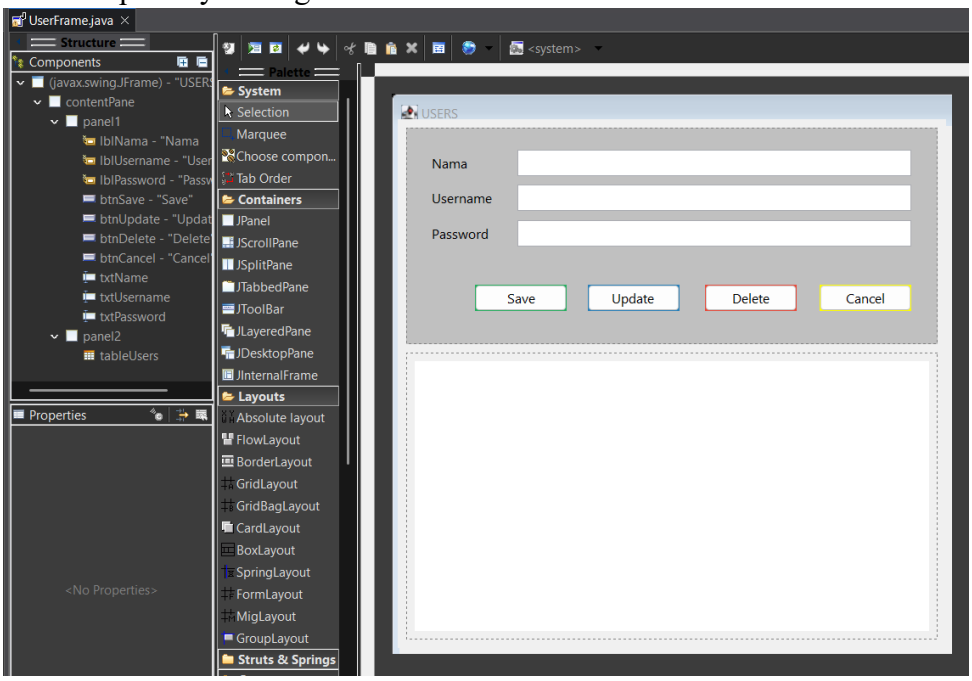
Mengembalikan nilai coon jika proses tidak terkendala error.

```
15            }catch(Exception e) {
16            JOptionPane.showMessageDialog(null, e);
17            return null;
18        }
19    }
```

Catch untuk menangani error (exception handling), jika error terjadi membuat program berhenti, blok catch akan dijalankan.

f. Jframe CRUD User

1. Buat new Jframe pada package ui dengan nama “UserFrame”.
2. Buat tampilannya sebagai berikut

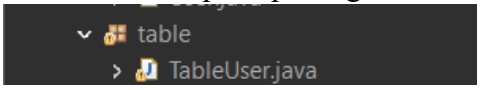


3. Penjelasan : Menggunakan compunent yaitu JtextField (Name, Useranme, Password), Jbutton (Save, Update, Delete, Cancel), dan Jtabel yang berfugnsi menampilkan data dalam bentk baris dan kolom dari database.

g. Tabel Model

Berfungsi unutk mengambil data dari database dan ditampilkan ke dalam table.

1. Buat package baru dengan nama “table”
2. Buat new class pada package tabel dengan nama “TableUser”



3. Tulis kodingan berikut

```
TableUser.java
 8      List<User> ls;
 9      private String[] columnNames = {"ID", "Name", "Username", "Password"};
10
11      public TableUser(List<User> ls) {
12          this.ls = ls;
13      }
14
15      @Override
16      public int getRowCount() {
17          // TODO Auto-generated method stub
18          return ls.size();
19      }
20
21      @Override
22      public int getColumnCount() {
23          // TODO Auto-generated method stub
24          return 4;
25      }
```

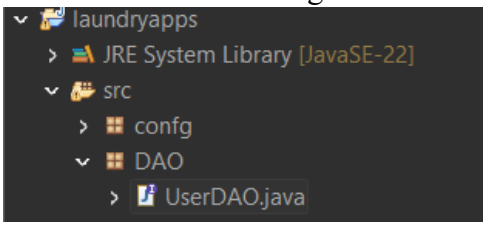
```java
    @Override
    public String getColumnName(int column) {
        // TODO Auto-generated method stub
        return columnNames[column];
    }

    @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;
        }
    }
}
```

4. Penjelasan : Berfungsi sebagai jembatan antara data “user” dengan import dari class user unutk field/attirbut yang telah dibuat sebelumnya dan tampilan “Jtable”. Menyimpan data mentah dalam tabel, menyimpan data dalam bentuk List<User>, kemudian memberikan informasi kepada component Jtable berupa ditampilkan nya data (baris, kolom, field name, is tiap sel). Sehingga Jtable otomatis menampilkan data “user” dengan format tabel rapi, berdasarkan getter disediakan oleh class user.

h. Fungsi DAO (Data Access Object)

1. Buat new package dengan nama “DAO”
2. Buat class interface dengan nama “UserDAO”



3. Tuliskan kodingan berikut

UserDAO.java

```java
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);
}
```

4. Penjelasan : Merupakan class interface untuk operasi CRUD saat membuat objek User. Mengimport class User sehingga menghubungkan logika data dari class user dengan akses data DAO. Implementasi interface UserDAO unutk mengatur cara user dikelola dalam database. TableUser menggunakan hasil method show() unutk menampilkan data tersebut di GUI Table.

i. Menggunakna Fungsi DAO

1. Buat new class pada package DAO dengan nama "UserRepo" unutk mengiplementasikan Interface DAO yang telah dibuat.
2. Implementasikan UserDao dengan kata kunci implements dari class UserDAO

```
1 package DAO;
2
3 import java.lang.System.Logger.Level;
4 import java.sql.*;
5 import java.util.*;
6 import java.util.logging.Logger;
7
8
9 import DAO.UserDao;
0 import confg.Database;
1 import model.User;
2
3 public class UserRepo implements UserDao {
```

3. Buat instasiai Connection dengan membuat construktr dan membuat String unutk melakukan manipulasi database.

```
15     private Connection connection;
16     final String insert = "INSERT INTO user (name, username, password) VALUES (?,?,?);";
17     final String select = "SELECT * FROM user;";
18     final String delete = "DELETE FROM user WHERE id=?;";
19     final String update = "UPDATE user SET name=?, username=?, password=? WHERE id=?;";
21         public UserRepo() {
22             connection = Database.koneksi();
23         }
```

4. Buat method save() unutk menyimpan data user saat tombol di Jframe user ditekan. Menggunakan preparedStatement dengan query insert.

```
26     public void save(User user) {
27         PreparedStatement st = null;
28         try {
29             st = connection.prepareStatement(insert);
30             st.setString(1, user.getNama());
31             st.setString(2, user.getUsername());
32             st.setString(3, user.getPassword());
33             st.executeUpdate();
34
35         }catch(SQLException e) {
36             e.printStackTrace();
37         }finally {
38             try {
39                 st.close();
40             }catch(SQLException e) {
41                 e.printStackTrace();
42             }
43         }
44     }
```

5. Buat method show() unutk mengambil data semua user dari Database. Deklarasi interface statement lalu menjalankan query select, hasil query tersimpan pada ResultSet dibaca tipa bars dengan while loop. Setiap baris dirubah jadi objek user (attribut pada class user). Menggunaka try and catch untuk menangkap error.

```
46    @Override
47    public List<User> show() {
48        List<User> ls = null;
49        try {
50            ls = new ArrayList<User>();
51            Statement st = connection.createStatement();
52            ResultSet rs = st.executeQuery(select);
53            while(rs.next()) {
54                User user = new User();
55                user.setId(rs.getString("id"));
56                user.setNama(rs.getString("name"));
57                user.setUsername(rs.getString("username"));
58                user.setPassword(rs.getString("password"));
59                ls.add(user);
60            }
61
62        }catch(SQLException e) {
63            Logger.getLogger(UserDao.class.getName()).log(Level.SEVERE, null, e);
64        }
65        return ls;
66    }
```

6. Buat method update() untuk mengubah/memperbarui data yang tersimpan pada database. Menggunakan preparedStatement dengaan query update. Mengambil nilai dari parameter class user.

```
68    @Override
69    public void update(User user) {
70        PreparedStatement st = null;
71        try {
72            st = connection.prepareStatement(update);
73            st.setString(1, user.getNama());
74            st.setString(2, user.getUsername());
75            st.setString(3, user.getPassword());
76            st.setString(4, user.getId());
77            st.executeUpdate();
78        } catch(SQLException e) {
79            e.printStackTrace();
80        } finally {
81            try {
82                st.close();
83            } catch(SQLException e) {
84                e.printStackTrace();
85            }
86        }
87    }
```

7. Buat method delete() unutk menghapus data tersimpan pada database berdasarkan “id” user. Membuat preparedStatment dengan query “delete”.

```
89•     @Override
90      public void delete(String id) {
91          PreparedStatement st = null;
92          try {
93              st = connection.prepareStatement(delete);
94              st.setString(1, id);
95              st.executeUpdate();
96
97          } catch(SQLException e) {
98              e.printStackTrace();
99          } finally {
00              try {
01                  st.close();
02              } catch(SQLException e) {
03                  e.printStackTrace();
04              }
05          }
06      }
07
08  }
```