

“LAPORAN PRAKTIKUM PEKAN 2”
PEMOGRAMAN BERORIENTASI OBJEK



NAMA : Reza Septian
NIM : 2311533008

DOSEN PENGAMPU :
Nurfiah, S.ST, M.Kom.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS

A. Tujuan

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

B. Alat

1. Computer / laptop yang telah terinstall JDK dan Eclipse
2. MySQL / XAMPP
3. MySQL connector atau Connector/J

C. Kajian teori

1. XAMPP

XAMPP adalah sebuah paket perangkat lunak open-source yang berisi berbagai komponen utama untuk pengembangan aplikasi berbasis web secara lokal (localhost). Komponen utama dalam XAMPP antara lain:

- 1) Apache HTTP Server:
 - Berfungsi sebagai web server yang menampilkan halaman web kepada pengguna melalui protokol HTTP.
 - Menjalankan dan mengelola permintaan HTTP dari browser ke aplikasi berbasis web.
- 2) MySQL:
 - Merupakan sistem manajemen basis data relasional (RDBMS) yang digunakan untuk mengelola dan menyimpan data dalam bentuk tabel.
 - MySQL mendukung penyimpanan, pengambilan, dan manipulasi data dengan menggunakan perintah SQL (Structured Query Language).
 - Karena bersifat open-source, MySQL menjadi pilihan populer dalam pengembangan aplikasi web.
- 3) PHP:
 - PHP adalah bahasa pemrograman server-side yang digunakan untuk membuat aplikasi web dinamis.
 - Berinteraksi dengan basis data, seperti MySQL, untuk memproses data dan menghasilkan konten yang disesuaikan untuk pengguna.
- 4) Perl:

- Sebagai bahasa pemrograman yang sering digunakan untuk tugas-tugas administratif dan manipulasi teks.

2. MySQL Connector/J

MySQL Connector/J adalah driver JDBC (Java Database Connectivity) yang digunakan untuk menghubungkan aplikasi berbasis Java dengan MySQL, yang memfasilitasi operasi CRUD pada database, yaitu:

- Create (Menyimpan Data)
- Read (Mengambil Data)
- Update (Mengubah Data)
- Delete (Menghapus Data)

Fungsi utama MySQL Connector/J:

- Membuka koneksi ke basis data MySQL.
- Mengirimkan perintah SQL dari aplikasi ke server MySQL.
- Menerima hasil dari server MySQL setelah eksekusi perintah.
- Menutup koneksi ke server MySQL setelah selesai melakukan operasi.

3. Data Access Object (DAO)

DAO (Data Access Object) adalah pola desain yang menyediakan antarmuka abstrak untuk mengakses dan memanipulasi data dari database. Keuntungan utama dari penggunaan DAO antara lain:

1. Modularitas:
 - Memisahkan logika akses data dari logika bisnis.
 - Meningkatkan keterbacaan dan pemeliharaan kode, sehingga lebih mudah diatur dan dikembangkan.
2. Reusabilitas:
 - DAO dapat digunakan kembali di berbagai bagian aplikasi, meminimalkan pengulangan kode.
3. Isolasi:
 - Perubahan pada logika akses data tidak mempengaruhi logika bisnis yang ada, karena DAO memberikan lapisan abstraksi.

4. Interface dalam Java

Dalam pemrograman Java, interface digunakan untuk mendefinisikan kumpulan metode abstrak yang harus diimplementasikan oleh kelas yang menggunakan interface tersebut.

Interface bermanfaat dalam penerapan pola desain DAO karena:

- Memberikan kontrak yang harus diikuti oleh setiap kelas yang mengimplementasikannya.
- Memastikan bahwa kelas yang berbeda dapat berinteraksi dengan konsistensi yang sama meskipun implementasi metode berbeda.

5. Operasi CRUD

CRUD adalah operasi dasar dalam pengelolaan data pada sebuah aplikasi, yang meliputi:

- Create: Menambahkan data baru ke dalam database.
- Read: Mengambil data dari database.
- Update: Mengubah data yang sudah ada dalam database.
- Delete: Menghapus data dari database.

Operasi CRUD sangat fundamental dalam aplikasi yang berhubungan dengan basis data, seperti aplikasi web yang menggunakan MySQL dan PHP. CRUD juga diterapkan melalui DAO untuk memastikan akses data yang efisien dan terstruktur.

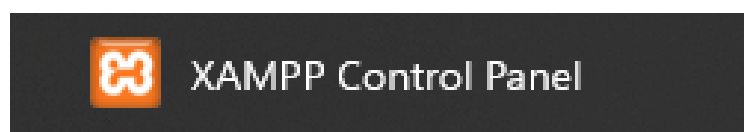
D. Langkah Kerja

1. Instal XAMPP

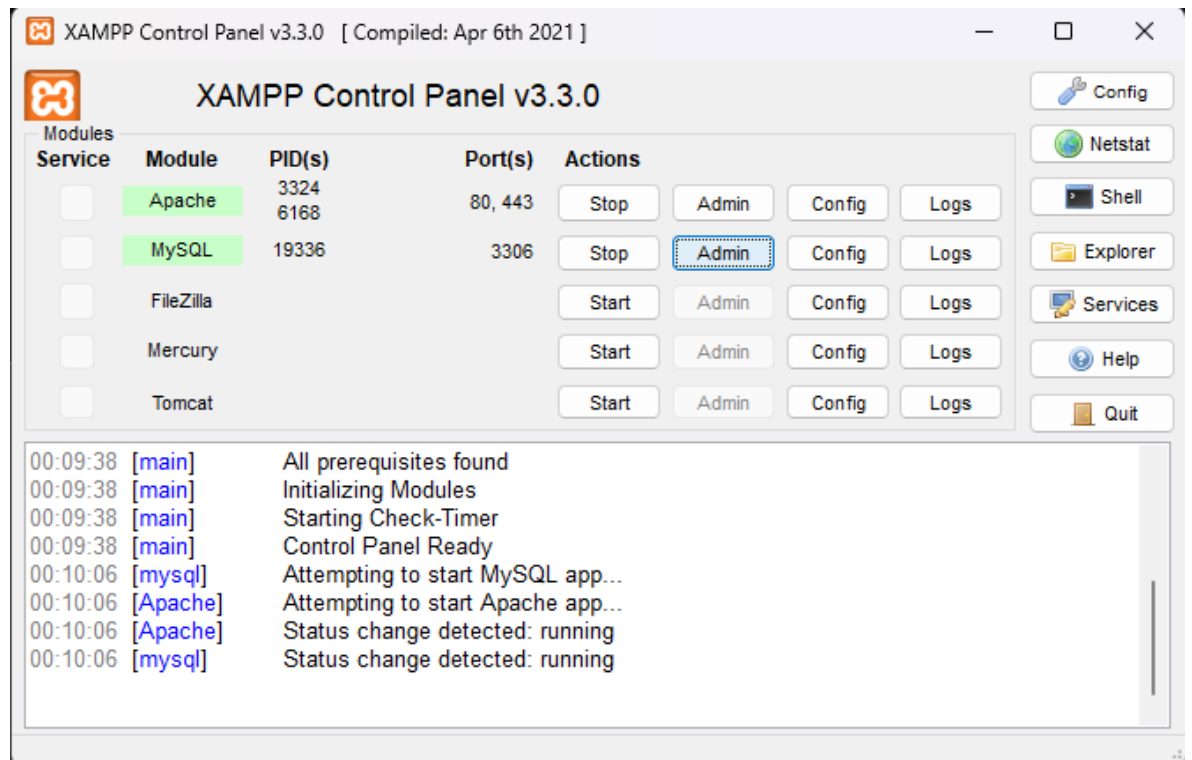
- Download XAMPP dari pada link berikut : <https://www.apachefriends.org/>



- Kemudian install XAMPP

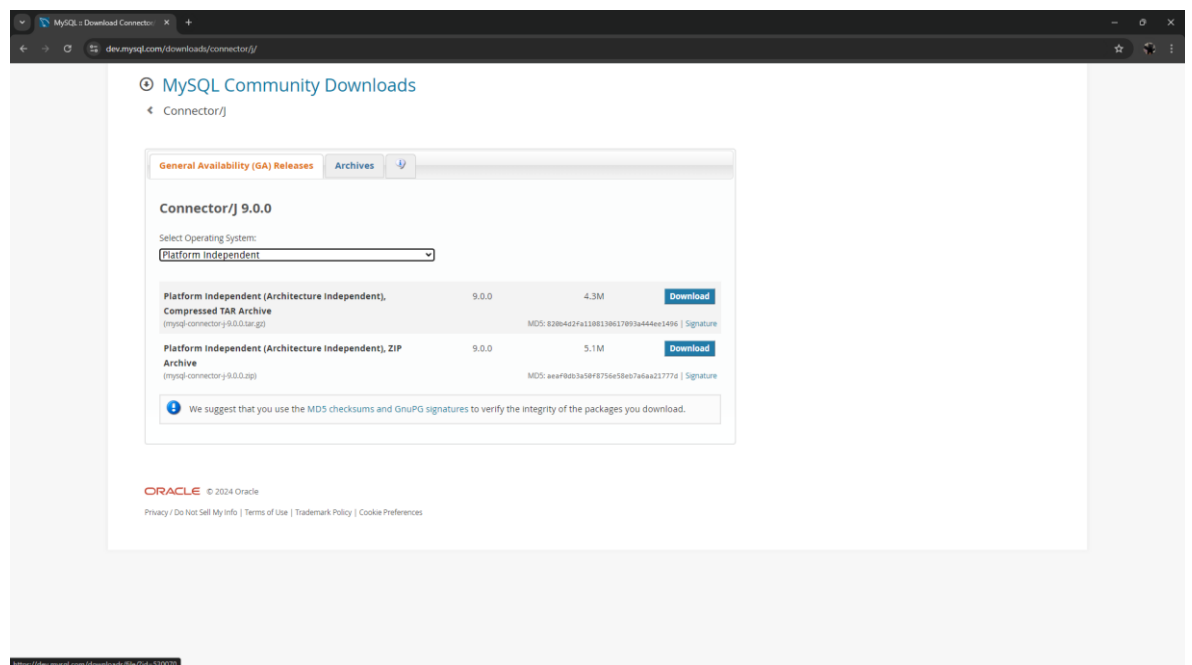


- Jalankan XAMPP dan aktifkan Apache dan MySQL

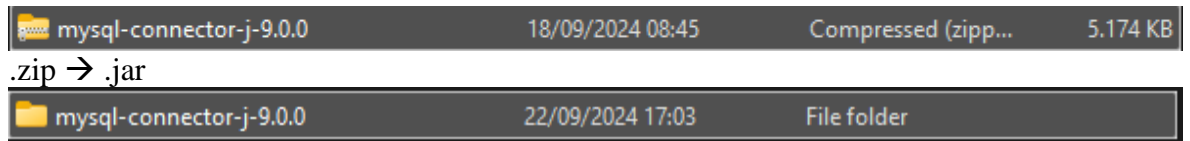


2. Menambahkan MySQL Connector

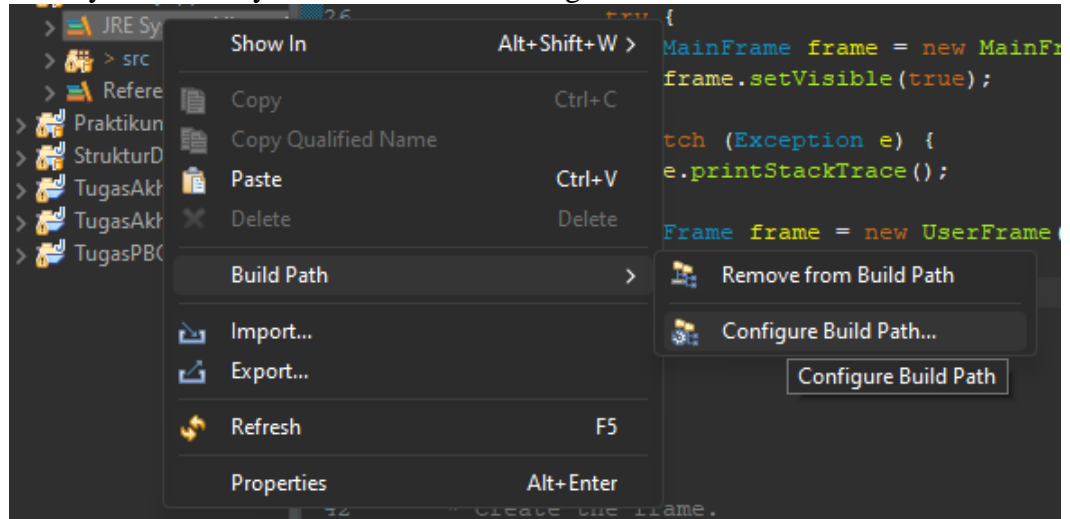
- Download MySQL connection pada link berikut : <https://dev.mysql.com/downloads/connector/j/>
- Pilih Operating System : Platform Independent
- Pilih file yang berekstensi .zip



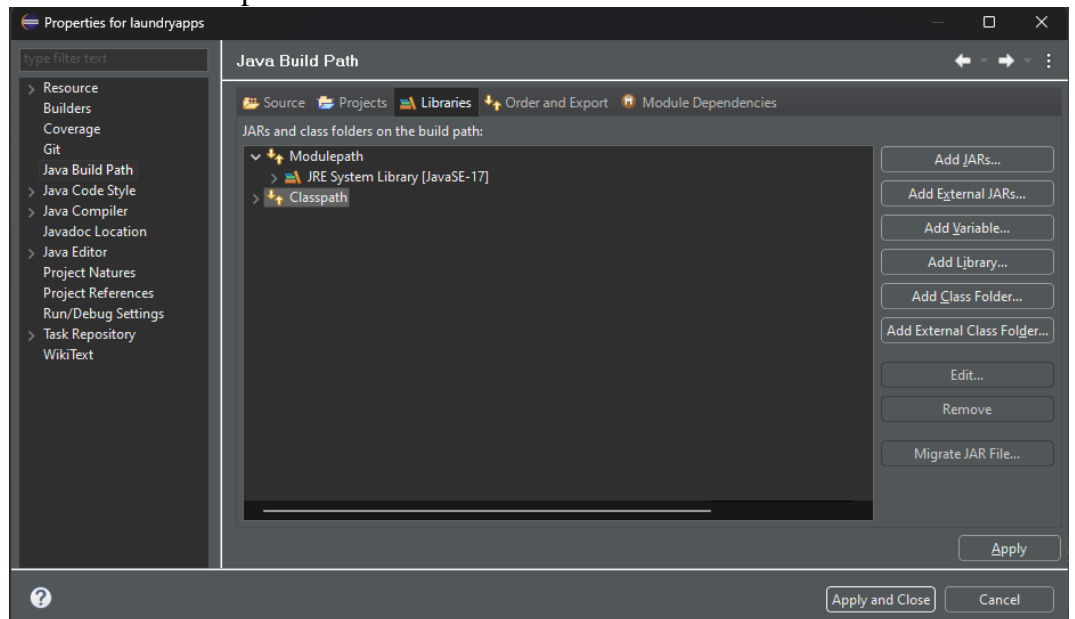
- Extract file .zip tersebut hingga menjadi .jar



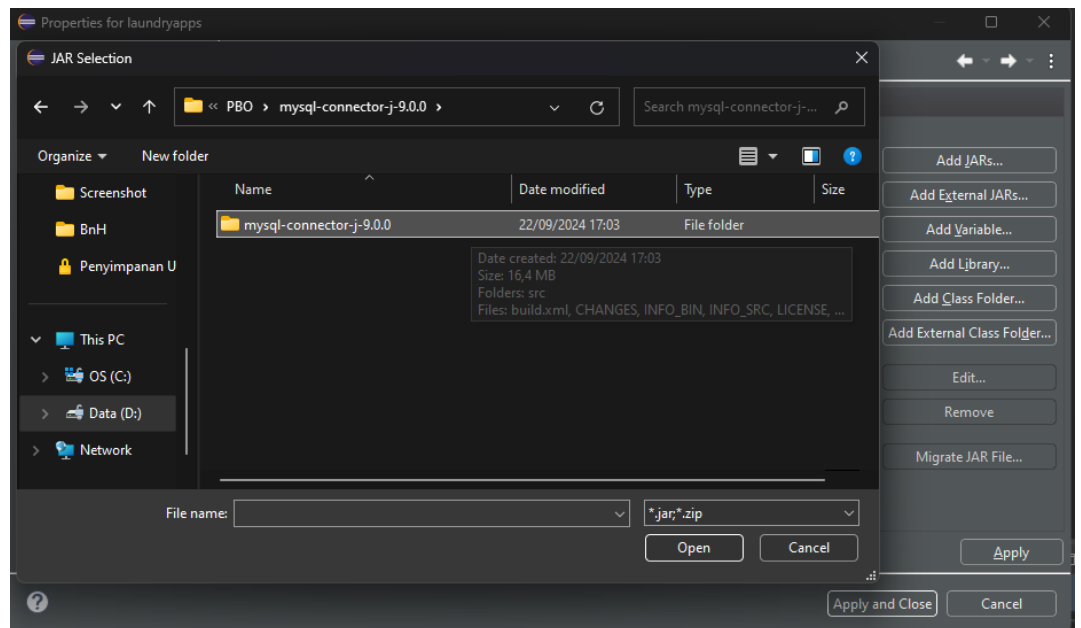
- Tambahkan MySQL Connector kedalam project dengan cara
 - JRE System Library → Built Path → Configure Build Path



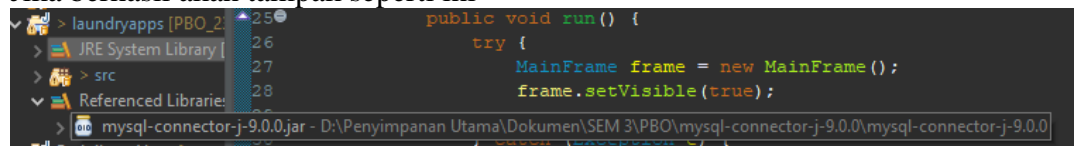
- Libraries → Classpath



- Add External JARs → Pilih file yang didownload → Apply and Close

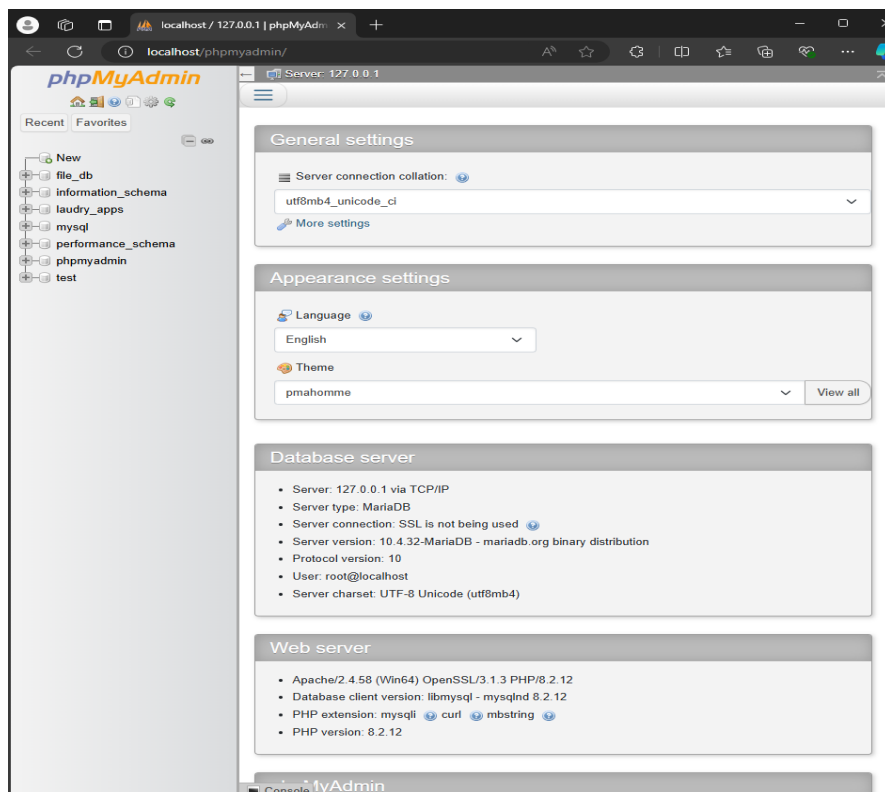


➤ Jika berhasil akan tampak seperti ini

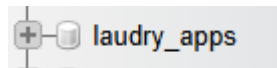


3. Membuat Database dan Table User

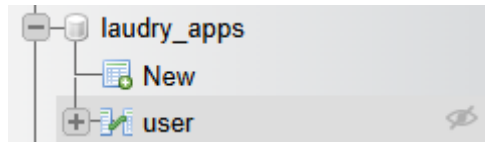
- Pada XAMPP yang sudah berjalan MySQL dan Apache nya, tekan “Admin” pada MySQL



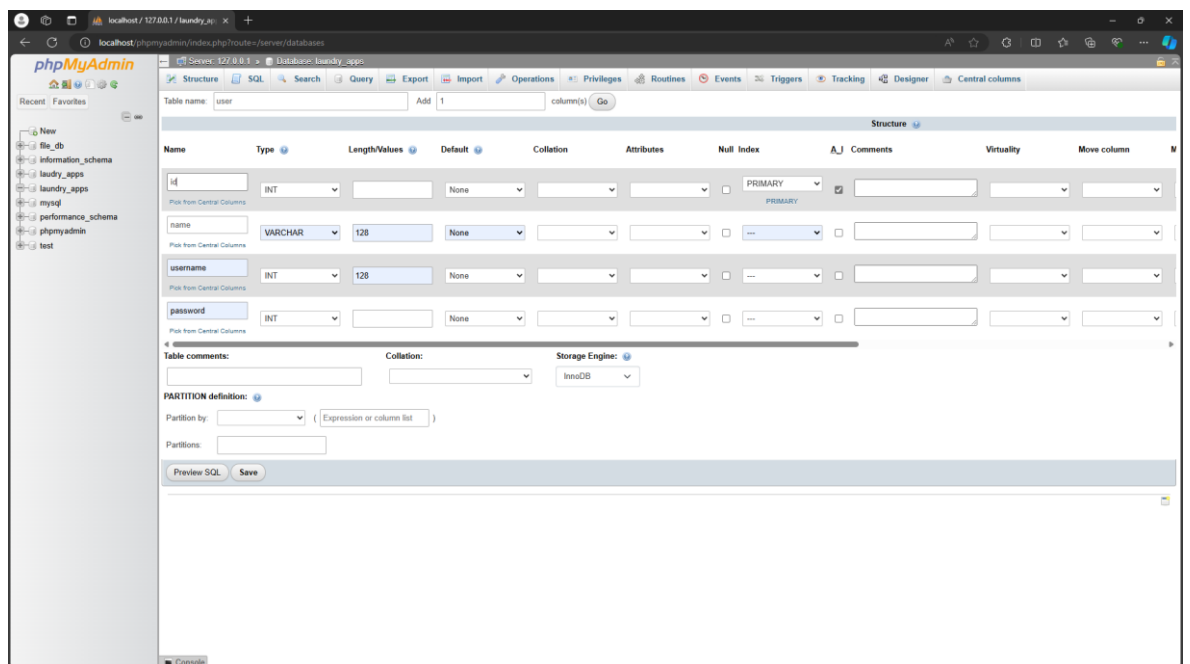
- Kemudian klik new dan buat database baru dengan nama laundry_apps



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

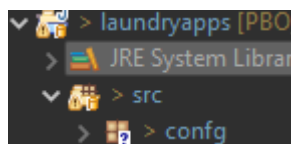


- Klik create akan muncul seperti gambar (Isikan seperti gambar dibawah)

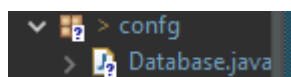


4. Membuat Koneksi ke Database MySQL

- Buat package baru dengan nama **config** yang berfungsi sebagai konfigurasi aplikasi yang akan dibuat termasuk database



- Buat class baru dengan nama **Database**, dengan isi codingan seperti berikut




```

1 package config;
2
3 import java.sql.*;
4
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             e.printStackTrace();
15             JOptionPane.showMessageDialog(null, e);
16             return null;
17         }
18     }
19 }
20

```

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

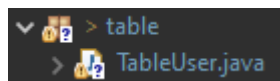
5. Membuat Tampilan CRUD User

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

- Ubahlah Variable pada JFrame menjadi :
 - txtName → Name
 - txtUser → Username
 - txtPassword → Password
 - btnSave → Save
 - btnUpdate → Update
 - btnDelete → Delete
 - btnCancel → Cancel
 - tableUsers → Table Users

6. Membuat Table Model

- Buat package baru dengan nama **table**
- Buat class baru pada **table** dengan nama **TableUser**



- Isi dengan codingan berikut :

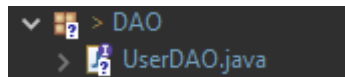
```

1 package table;
2
3 import java.util.List;
4
5
6 public class TableUser extends AbstractTableModel {
7     List<User> ls;
8     private String[] columnNames = {"ID", "Name", "Username", "Password"};
9     public TableUser(List<User> ls) {
10         this.ls = ls;
11     }
12
13     @Override
14     public int getRowCount() {
15         // TODO Auto-generated method stub
16         return ls.size();
17     }
18
19     @Override
20     public int getColumnCount() {
21         // TODO Auto-generated method stub
22         return 4;
23     }
24
25     @Override
26     public String getColumnName(int column) {
27         // TODO Auto-generated method stub
28         return columnNames[column];
29     }
30
31     @Override
32     public Object getValueAt(int rowIndex, int columnIndex) {
33         // TODO Auto-generated method stub
34         switch (columnIndex) {
35             case 0:
36                 return ls.get(rowIndex).getId();
37             case 1:
38                 return ls.get(rowIndex).getName();
39             case 2:
40                 return ls.get(rowIndex).getUsername();
41             case 3:
42                 return ls.get(rowIndex).getPassword();
43             default:
44                 return null;
45         }
46     }
47 }

```

7. Membuat Fungsi DAO

- Buat package baru dengan nama **DAO**
- Buat class interface baru dengan nama **UserDAO**



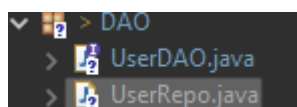
- Isi dengan codingan berikut :

```
1 package DAO;
2
3 import java.util.List;
4
5
6
7 public interface UserDAO {
8     void save(User user);
9     public List<User> show();
10    public void delete(String id);
11    public void update(User user);
12
13 }
14
```

- Catatan :
 - Method save, show, delete, dan update digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya

8. Menggunakan Fungsi DAO

- Buat class baru pada package **DAO** dengan nama **UserRepo** yang akan digunakan untuk mengimplementasikan **DAO** yang telah dibuat



- Implementasikan **UserDAO** dengan kata kunci **implements**

```
1 package DAO;
2
3 import java.sql.Connection;
4
5
6
7 public class UserRepo implements UserDAO{
8
9
10
11
12
13
14
15
16
17
```

- Membuat instansi Connection, membuat constructor dan membuat String untuk melakukan manipulasi database

```

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** kemudian isikan dengan codingan berikut :

```

@Override
public void save(User user) {
    // TODO Auto-generated method stub
    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** kemudian isikan dengan codingan berikut :

```

@Override
public List<User> show() {
    // TODO Auto-generated method stub
    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** kemudian isikan dengan codingan berikut :

```
@Override
public void update(User user) {
    // TODO Auto-generated method stub

    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** kemudian isikan dengan codingan berikut :

```
@Override
public void delete(String id) {
    // TODO Auto-generated method stub
    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();
        }
    }
}
```

9. Menggunakan Fungsi CRUD DAO pada GUI

- Buat method reset (menghapus value inputan) pada JFrame UserFrame seperti program dibawah ini

```
public void reset() {
    txtName.setText("");
    txtUsername.setText("");
    txtPassword.setText("");
}
```

- Membuat instance pada UserFrame

```
UserRepo usr = new UserRepo();
List<User> ls;
public String id;
```

10. CREATE USER

- Pada JFrame UserFrame, klik kanan pada tombol save → add event handlres → actionPerformed atau klik 2 kali pada button save kemudian isi dengan program berikut

```
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setPassword(txtPassword.getText());
        usr.save(user);
        reset();
        loadTable();
    }
});
btnSave.setBounds(81, 141, 76, 23);
contentPane.add(btnSave);
```

11. READ USER

- Buat method dengan nam loadTable() kemudian isikan dengan kode program berikut

```
public void loadTable() {
    ls = usr.show();
    TableUser tu = new TableUser(ls);
    tableUsers.setModel(tu);
    tableUsers.getTableHeader().setVisible(true);
}
```

- Panggil method pada JFrame MainFrame sehingga pertama kali program dijalankan maka loadTable akan dipanggil

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                MainFrame frame = new MainFrame();
                frame.setVisible(true);

            } catch (Exception e) {
                e.printStackTrace();
            }
            UserFrame frame = new UserFrame();
            frame.setVisible(true);
            frame.loadTable();
        }
    });
}
```

12. UPDATE USER

- Klik kanan pada.JTable → add event handler → mouse → mouseClicked
- Isikan dengan codingan berikut :

```
tableUsers = new JTable();
tableUsers.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableUsers.getValueAt(tableUsers.getSelectedRow(), 0).toString();
        txtName.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 1).toString());
        txtUsername.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 2).toString());
        txtPassword.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 3).toString());
    }
});
tableUsers.setBounds(39, 175, 376, 205);
contentPane.add(tableUsers);
```

- Penjelasan :
 - Kode diatas berfungsi mengambil id user dan menyimpannya kedalam variable id kemudian mengambil data nama, username dan password kemudian ditamirkan kedalam form inputan
- Klik salah satu isi table maka akan secara otomatis tampil pada form inputan
- Klik kanan tombol update → add event handler → action → actionPerformed atau bisa juga pada button update di klik 2 kali dan isikan codingan berikut

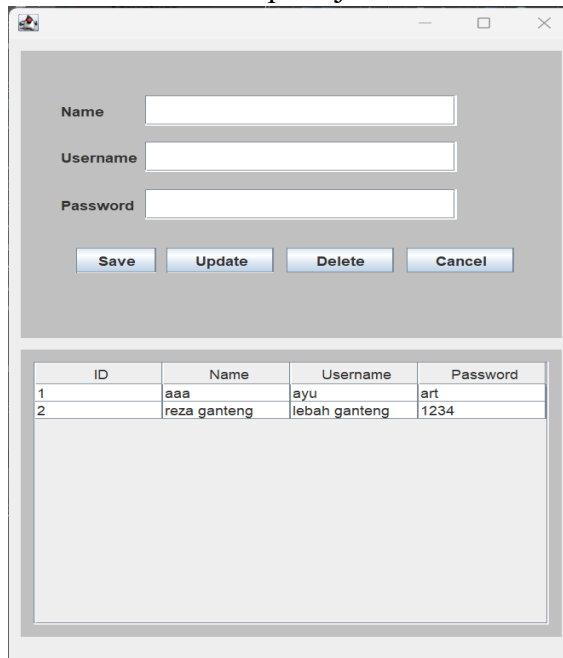
```

JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setPassword(txtPassword.getText());
        user.setId(id);
        usr.update(user);
        reset();
        loadTable();
    }
});
btnUpdate.setBounds(167, 141, 76, 23);
contentPane.add(btnUpdate);

```

13. DELETE USER

- Klik salah satu data pada jTable



ID	Name	Username	Password
1	aaa	ayu	art
2	reza ganteng	lebah ganteng	1234

- Klik kanan tombol delete → add event handler → action → actionPerformed atau bisa juga pada button update di klik 2 kali dan isikan codingan berikut

```

JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            usr.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
        }
    }
});
btnDelete.setBounds(253, 141, 76, 23);
contentPane.add(btnDelete);

```


E. Kesimpulan

Melalui praktikum ini, saya telah mempelajari dan mengimplementasikan konsep dasar dalam pembuatan aplikasi berbasis database dengan menerapkan fungsi CRUD (Create, Read, Update, Delete) untuk pengelolaan data pengguna menggunakan MySQL. Tujuan utama dari praktikum ini adalah agar saya mampu membuat, menghubungkan, dan memanipulasi data dalam sebuah sistem berbasis database dengan menggunakan Java sebagai bahasa pemrograman dan MySQL sebagai sistem manajemen basis data.