LAPORAN PRATIKUM OOP PEKAN 3
IMPLEMENTASI APLIKASI DESKTOP LAUNDRY DENGAN FUNGSI CRUD DAN DATABASE MYSQL

MATA KULIAH PEMROGRAMAN BERORIENTASI OBJEK (PBO)

DOSEN PENGAMPU : NURFIAH, S.ST, M.KOM

OLEH:

AUFAN TAUFIQURRAHMAN

NIM 2411532011

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

BAB 1

PENDAHULUAN

1.1.Latar Belakang

Praktikum awal telah berhasil meletakkan fondasi untuk aplikasi "Laundry Apps" dengan membangun model data dasar dan antarmuka login yang statis. Namun, fungsionalitas aplikasi tersebut masih sangat terbatas karena belum terhubung dengan sistem penyimpanan data permanen, sehingga semua data bersifat sementara dan hardcoded.

Laporan ini merangkum evolusi signifikan dari aplikasi tersebut, yaitu implementasi fungsionalitas penuh CRUD (Create, Read, Update, Delete) yang terhubung ke database MySQL. Transformasi ini mengubah aplikasi dari sekadar prototipe antarmuka menjadi sebuah sistem informasi desktop yang dinamis dan fungsional. Implementasi CRUD tidak hanya diterapkan pada data Pengguna (User), tetapi juga pada entitas bisnis inti lainnya, yaitu Pelanggan (Customer) dan Layanan (Service), sesuai dengan tugas latihan yang diberikan.

1.2.Tujuan Praktikum

Tujuan dari keseluruhan praktikum ini adalah agar mahasiswa mampu:

1. Merancang dan membangun koneksi yang stabil antara aplikasi Java dengan database MySQL menggunakan driver JDBC.
2. Menerapkan pola desain DAO (Data Access Object) untuk menciptakan arsitektur yang modular dan memisahkan logika akses data dari logika bisnis.
3. Merancang dan mengimplementasikan antarmuka pengguna (GUI) yang fungsional untuk melakukan operasi CRUD pada berbagai entitas data (User, Customer, dan Service).
4. Mengintegrasikan beberapa jendela (JFrame) menjadi satu alur aplikasi yang koheren, yang dapat diakses melalui menu utama.

BAB 2

DASAR TEORI

2.1. MySQL dan JDBC (MySQL Connector/J)

MySQL adalah sebuah Relational Database Management System (RDBMS) yang digunakan untuk menyimpan dan mengelola data secara terstruktur. Untuk menghubungkan aplikasi Java dengan database MySQL, diperlukan sebuah driver JDBC (Java Database Connectivity). MySQL Connector/J adalah implementasi driver resmi yang berfungsi sebagai jembatan, memungkinkan eksekusi kueri SQL dari dalam kode Java.

2.2. Pola Desain DAO (Data Access Object)

DAO adalah sebuah pola desain yang bertujuan memisahkan logika akses data dari logika aplikasi utama. Pola ini dicapai dengan mendefinisikan sebuah interface yang berisi kontrak operasi data (misalnya save, show), dan sebuah kelas repository yang mengimplementasikan interface tersebut dengan menuliskan kueri SQL yang spesifik. Keuntungannya adalah kode menjadi lebih rapi, mudah dikelola, dan fleksibel terhadap perubahan sumber data.

2.3. AbstractTableModel

Dalam Java Swing, AbstractTableModel adalah kelas abstrak yang berfungsi sebagai adapter antara sumber data (seperti List) dengan komponen visual JTable. Dengan meng-override beberapa metodenya, kita dapat secara efisien menampilkan, memperbarui, dan mengelola data yang muncul di dalam tabel pada GUI.

BAB 3
IMPLEMENTASI DAN PEMBAHASAN

3.1. Arsitektur Aplikasi: Koneksi dan Pola DAO

Fondasi dari aplikasi dinamis ini adalah kelas src.confg.Database.java yang mengelola koneksi JDBC. Di atasnya, arsitektur DAO diterapkan secara sistematis. Untuk setiap entitas, dibuat tiga komponen utama:

1. Interface DAO (misal, UserDAO.java): Mendefinisikan metode CRUD.

```java
package src.DAO;

import java.util.*;
import src.model.User;

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

2. Kelas Repository (misal, UserRepo.java): Mengimplementasikan interface DAO dengan kueri SQL.

```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();
    }

    @Override
    public void save(User user) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(parameterIndex:1, user.getNama());
            st.setString(parameterIndex:2, user.getUsername());
            st.setString(parameterIndex:3, user.getPassword());
            st.executeUpdate();
```

3. Kelas TableModel (misal, TableUser.java): Menjadi jembatan antara data dari repository ke JTable.

```java
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;
        }
    }
}
```

3.2. Implementasi CRUD Pengguna (User)

Sebuah jendela baru, UserFrame.java, dibuat untuk mengelola data pengguna. Jendela ini berisi formulir input dan sebuah JTable.

- Create: Tombol "Save" akan mengambil data dari formulir, membuat objek User, dan menyimpannya ke database melalui userRepo.save(user).

```
btnSave = new JButton(text:"Save");
btnSave.setBounds(x:120, y:150, width:80, height:30);
panel.add(btnSave);
```

- Read: Saat jendela dibuka, metode loadTable() memanggil userRepo.show() untuk mengambil semua data pengguna dan menampilkannya di JTable menggunakan TableUser.

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

- Update & Delete: Ketika sebuah baris di tabel diklik, datanya akan muncul di formulir. Tombol "Update" dan "Delete" kemudian menggunakan id dari baris yang dipilih untuk melakukan operasi melalui userRepo.update(user) atau userRepo.delete(id).

```
btnUpdate = new JButton(text:"Update");
btnUpdate.setBounds(x:210, y:150, width:80, height:30);
panel.add(btnUpdate);

btnDelete = new JButton(text:"Delete");
btnDelete.setBounds(x:300, y:150, width:80, height:30);
panel.add(btnDelete);
```

3.3. Implementasi CRUD Pelanggan (Customer)

Mengikuti pola yang sama, CustomerFrame.java dibuat untuk mengelola data pelanggan. Kelas ini terhubung dengan CustomerRepo.java dan TableCustomer.java. Formulir disesuaikan untuk input nama, alamat, dan telepon. Logika di balik tombol-tombol CRUD sama persis dengan pada UserFrame, menunjukkan keberhasilan penerapan pola desain yang reusable.

```java
public class CustomerRepo implements CustomerDAO {
    private Connection connection;

    final String insert = "INSERT INTO customer (nama, alamat, telepon) VALUES (?,?,?);";
    final String select = "SELECT * FROM customer;";
    final String update = "UPDATE customer SET nama=?, alamat=?, telepon=? WHERE id=?;";
    final String delete = "DELETE FROM customer WHERE id=?;";

    public CustomerRepo() {
        connection = Database.koneksi();
    }

    @Override
    public void save(Customer customer) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(parameterIndex:1, customer.getNama());
            st.setString(parameterIndex:2, customer.getAlamat());
            st.setString(parameterIndex:3, customer.getTelepon());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
```

```java
public class TableCustomer extends AbstractTableModel {
    List<Customer> ls;
    private String[] columnNames = {"ID", "Nama Pelanggan", "Alamat", "No. Telepon"};

    public TableCustomer(List<Customer> 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).getAlamat();
            case 3:
                return ls.get(rowIndex).getTelepon();
            default:
                return null;
        }
    }
}
```

3.4. Implementasi CRUD Layanan (Service)

Fungsionalitas untuk mengelola layanan diimplementasikan dalam ServiceFrame.java, yang terhubung dengan ServiceRepo.java dan TableService.java. Antarmuka ini memungkinkan admin atau kasir untuk menambah layanan baru (misal,

"Cuci Ekspress"), mengubah harga, atau mengubah status ketersediaannya.

```java
package src.DAO;

import java.util.List;
import src.model.Service;

public interface ServiceDAO {
    void save(Service service);
    public List<Service> show();
    public void update(Service service);
    public void delete(String id);
}
```

```java
public class TableService extends AbstractTableModel {
    List<Service> ls;
    private String[] columnNames = {"ID", "Jenis Layanan", "Harga", "Status"};

    public TableService(List<Service> ls) {
        this.ls = ls;
    }

    @Override
    public int getRowCount() {
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @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).getJenis();
            case 2:
                return ls.get(rowIndex).getHarga();
            case 3:
                return ls.get(rowIndex).getStatus();
            default:
                return null;
        }
```

3.5. Integrasi dengan Menu Utama (MainFrame.java)

Sebagai langkah akhir, semua jendela CRUD yang baru diintegrasikan ke dalam menu utama. ActionListener ditambahkan pada tombol "PENGGUNA", "PELANGGAN", dan "LAYANAN" di dalam MainFrame.java. Ketika salah satu tombol ini diklik, ia akan membuat instance dari frame yang sesuai (UserFrame, CustomerFrame, atau

ServiceFrame), menampilkannya, dan langsung memanggil metode loadTable() untuk memuat data. Perilaku tombol tutup (X) pada setiap frame sekunder diatur ke DISPOSE_ON_CLOSE agar tidak menghentikan aplikasi utama saat ditutup.

```
JButton btnPengguna = new JButton(text:"PENGGUNA");
btnPengguna.setBounds(x:50, y:140, width:150, height:40);
panel.add(btnPengguna);

btnPengguna.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        UserFrame userFrame = new UserFrame();
        userFrame.setVisible(b:true);
        userFrame.loadTable();
    }
});
```

```
JButton btnPelanggan = new JButton(text:"PELANGGAN");
btnPelanggan.setBounds(x:390, y:80, width:150, height:40);
panel.add(btnPelanggan);

btnPelanggan.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        CustomerFrame customerFrame = new CustomerFrame();
        customerFrame.setVisible(b:true);
        customerFrame.loadTable();
    }
});
```

```
JButton btnLayanan = new JButton(text:"LAYANAN");
btnLayanan.setBounds(x:220, y:80, width:150, height:40);
panel.add(btnLayanan);

btnLayanan.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ServiceFrame serviceFrame = new ServiceFrame();
        serviceFrame.setVisible(b:true);
        serviceFrame.loadTable();
    }
});
```

BAB 4

PENYELESAIAN LATIHAN/TUGAS

Tugas latihan yang diberikan dalam modul praktikum adalah mengimplementasikan fungsionalitas CRUD untuk data Layanan (Service) dan Pelanggan (Customer).

Tugas ini telah berhasil diselesaikan sepenuhnya. Implementasi dari kedua fungsionalitas tersebut telah diuraikan secara rinci pada Bab 3, sub-bab 3.3 dan 3.4. Proses pengerjaannya dilakukan dengan mereplikasi arsitektur dan pola yang sebelumnya telah berhasil diterapkan pada CRUD User, membuktikan pemahaman yang baik terhadap konsep DAO, JDBC, dan

BAB 5

KESIMPULAN

Praktikum ini telah berhasil mentransformasi aplikasi "Laundry Apps" dari sebuah prototipe antarmuka statis menjadi sebuah aplikasi desktop berbasis data yang fungsional dan dinamis. Seluruh tujuan praktikum telah tercapai dengan sukses.

Aplikasi kini mampu melakukan operasi CRUD penuh untuk entitas-entitas bisnis inti—User, Customer, dan Service—dengan data yang disimpan secara permanen di database MySQL. Penerapan pola desain DAO telah menciptakan sebuah arsitektur yang kokoh, modular, dan mudah untuk dikembangkan lebih lanjut. Integrasi semua jendela manajemen data ke dalam satu menu utama juga telah membentuk alur aplikasi yang logis dan mudah digunakan.