

# **LAPORAN PRATIKUM**

## **“PEKAN 2”**

***Disusun Untuk Memenuhi Tugas Mata Kuliah PBO***

**DOSEN PENGAMPU:**

**Nurfiah, S.ST. M.Kom.**



**DISUSUN OLEH:**

**Karimah Irsyadiyah (2411533018)**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS ANDALAS  
T.A 2024/2025**

## Daftar Pustaka

BAB I PENDAHULUAN.....	3
1.1    Latar Belakang.....	3
1.2    Tujuan .....	3
1.3    Alat dan Bahan.....	4
BAB II PEMBAHASAN .....	5
2.1    Langkah-langkah Praktikum dan Pembahasan Program .....	5
BAB III PENUTUP .....	33
3.1    Kesimpulan .....	33

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Perkembangan teknologi informasi mendorong berbagai bidang usaha, termasuk usaha jasa, untuk beralih dari sistem manual ke sistem berbasis komputer. Salah satu bidang yang membutuhkan pengelolaan data secara cepat dan akurat adalah usaha laundry. Pada umumnya, laundry melayani banyak pelanggan dengan berbagai jenis layanan, sehingga diperlukan sistem yang dapat mencatat data pelanggan, data layanan, serta pengguna yang mengelola aplikasi.

Sistem informasi laundry berbasis Java dengan konsep **CRUD (Create, Read, Update, Delete)** merupakan solusi praktis untuk mempermudah pengelolaan data. Dengan adanya sistem ini, pengguna dapat menambahkan, menampilkan, memperbarui, dan menghapus data pelanggan maupun layanan dengan lebih terstruktur. Selain itu, penggunaan database MySQL memungkinkan data tersimpan secara permanen, aman, dan mudah diakses kembali.

Melalui praktikum ini, mahasiswa diajak untuk memahami bagaimana cara membangun aplikasi sederhana menggunakan bahasa pemrograman Java dan memanfaatkan JDBC untuk menghubungkan aplikasi dengan database. Pembelajaran ini tidak hanya melatih keterampilan teknis pemrograman, tetapi juga mengajarkan pentingnya desain sistem yang terorganisir, mulai dari struktur project, pembuatan model, repository, hingga implementasi antarmuka pengguna (GUI).

### 1.2 Tujuan

- Mempelajari implementasi koneksi database MySQL dengan Java menggunakan JDBC.
- Menerapkan konsep CRUD (Create, Read, Update, Delete) pada entitas User, Layanan, dan Pelanggan.
- Membuat antarmuka grafis (GUI) menggunakan Java Swing untuk mengelola data laundry.
- Membuat Login dan Menu Utama agar aplikasi dapat digunakan sesuai kebutuhan.

### 1.3 Alat dan Bahan

Alat dan bahan yang digunakan dalam praktikum ini adalah:

- **Perangkat Keras (Hardware):**
  - Laptop/PC dengan sistem operasi Windows/Linux/macOS.
  - RAM minimal 4 GB.
- **Perangkat Lunak (Software):**
  - Java Development Kit (JDK) versi 8 atau lebih baru.
  - IDE Eclipse untuk coding dan desain GUI.
  - Library bawaan Java Swing dan AWT untuk membangun GUI.

## BAB II

### PEMBAHASAN

#### 2.1 Langkah-langkah Praktikum dan Pembahasan Program

##### A. Membuat Database

Langkah pertama adalah membuat database laundry\_apps di MySQL. Database ini berfungsi sebagai tempat penyimpanan data. Di dalamnya dibuat tiga tabel utama yaitu user untuk menyimpan data pengguna, layanan untuk data jenis layanan laundry, serta pelanggan untuk data pelanggan. Struktur tabel ditentukan menggunakan perintah CREATE TABLE.

The image shows two parts of the MySQL environment. The top part displays two SQL queries in a command-line interface. The left query creates the 'layanan' table with columns: id (VARCHAR(50) PRIMARY KEY), nama\_layanan (VARCHAR(150) NOT NULL), and harga (DOUBLE NOT NULL). The right query creates the 'pelanggan' table with columns: id (VARCHAR(50) PRIMARY KEY), nama (VARCHAR(150) NOT NULL), alamat (TEXT), and no\_hp (VARCHAR(30)).

The bottom part is a screenshot of the phpMyAdmin web interface. The left sidebar shows a list of databases, with 'laundry\_apps' selected. The main panel shows the 'Structure' tab for the 'laundry\_apps' database. It lists three tables: 'layanan', 'pelanggan', and 'user'. Below the table list, there is a 'Create new table' dialog box with fields for 'Table name' and 'Number of columns' (set to 4), and a 'Create' button.

Table	Action	Rows	Type	Collation	Size	Overhead
layanan	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
pelanggan	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
user	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<b>3 tables</b>	<b>Sum</b>	<b>3</b>	<b>InnoDB</b>	<b>utf8mb4_general_ci</b>	<b>48.0 KiB</b>	<b>0 B</b>

```
CREATE DATABASE IF NOT EXISTS laundry_apps;
USE laundry_apps;

-- Tabel User
CREATE TABLE IF NOT EXISTS user (
  id VARCHAR(50) PRIMARY KEY,
  nama VARCHAR(100),
```

```

    username VARCHAR(100),
    password VARCHAR(100)
);

-- Tabel Layanan
CREATE TABLE IF NOT EXISTS layanan (
    id VARCHAR(50) PRIMARY KEY,
    nama_layanan VARCHAR(150),
    harga DOUBLE
);

-- Tabel Pelanggan
CREATE TABLE IF NOT EXISTS pelanggan (
    id VARCHAR(50) PRIMARY KEY,
    nama VARCHAR(150),
    alamat TEXT,
    no_hp VARCHAR(30)
);

```

## B. Struktur Project

Struktur project diatur agar lebih terorganisir dengan membagi kode ke dalam beberapa package: config untuk koneksi database, DAO untuk interface data access object, model untuk entitas dan form, table untuk model tabel, dan ui untuk tampilan login serta menu utama. Pembagian ini memudahkan pemeliharaan kode

```

src
config      : Database.java
DAO         : UserDAO, UserRepo, LayananDAO, LayananRepo, PelangganDAO,
PelangganRepo
model       : User, UserFrame, Layanan, LayananFrame, Pelanggan,
PelangganFrame
table       : TableUsers, TableLayanan, TablePelanggan
ui          : LoginFrame, MainFrame

```

## C. Membuat Koneksi Database

File Database.java berfungsi membuat koneksi ke MySQL menggunakan JDBC. Method koneksi() menghubungkan aplikasi dengan database laundry\_apps. Jika koneksi berhasil, aplikasi dapat menjalankan query; jika gagal, akan muncul pesan error.

### 1. File: config/Database.java

```

package config;

import java.sql.Connection;
import java.sql.DriverManager;

```

```
import javax.swing.JOptionPane;

public class Database {
    public static Connection koneksi() {
        Connection conn = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            String url =
"jdbc:mysql://localhost:3306/laundry_apps?useSSL=false&serverTimezone=UTC";
            String user = "root";
            String pass = ""; // ganti jika pakai password
            conn = DriverManager.getConnection(url, user, pass);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Koneksi gagal: " +
e.getMessage());
        }
        return conn;
    }
}
```

#### D. Membuat Model

Model berisi kelas representasi data:

- User.java menyimpan atribut pengguna (id, nama, username, password).
- Layanan.java menyimpan atribut layanan (id, nama layanan, harga).
- Pelanggan.java menyimpan atribut pelanggan (id, nama, alamat, no HP).

Model ini digunakan sebagai blueprint data yang akan diproses.

##### 1. User.java

```
package model;

public class User {
    String id, nama, username, password;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }
}
```

```

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public static boolean login(String username, String password) {
        boolean isLoggin = false;
        User user = new User();
        user.setId("1");
        user.setNama("ima");
        user.setUsername("ima");
        user.setPassword("12345");

        if (user.getUsername().equalsIgnoreCase(username)
            && user.getPassword().equalsIgnoreCase(password)) {
            isLoggin = true;
        } else {
            isLoggin = false;
        }
        return isLoggin;
    }
}

```

## 2. Layanan.java

```

package model;

public class Layanan {
    private String id;
    private String namaLayanan;
    private double harga;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getNamaLayanan() { return namaLayanan; }
    public void setNamaLayanan(String namaLayanan) { this.namaLayanan =
namaLayanan; }

    public double getHarga() { return harga; }
    public void setHarga(double harga) { this.harga = harga; }
}

```

## 3. Pelanggan.java



```

package model;

public class Pelanggan {
    private String id;
    private String nama;
    private String alamat;
    private String noHp;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getNama() { return nama; }
    public void setNama(String nama) { this.nama = nama; }

    public String getAlamat() { return alamat; }
    public void setAlamat(String alamat) { this.alamat = alamat; }

    public String getNoHp() { return noHp; }
    public void setNoHp(String noHp) { this.noHp = noHp; }
}

```

## E. Membuat DAO

DAO (Data Access Object) adalah interface untuk operasi CRUD. UserDAO, LayananDAO, dan PelangganDAO berisi deklarasi method seperti save, update, delete, dan show. DAO memastikan konsistensi penggunaan method dalam mengakses database.

### 1. DAO/UserDAO.java

```

package DAO;

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

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

```

### 2. PelangganDAO.java

```

package DAO;

import model.Pelanggan;
import java.util.List;

public interface PelangganDAO {
    void save(Pelanggan p);
    void update(Pelanggan p);
    void delete(String id);
    List<Pelanggan> show();
}

```

### 3. LayananDAO.java

```

package DAO;

import model.Layanan;
import java.util.List;

public interface LayananDAO {
    void save(Layanan l);
    void update(Layanan l);
    void delete(String id);
    List<Layanan> show();
}

```

## F. Membuat Repo

DAO (Data Access Object) adalah interface untuk operasi CRUD. UserDAO, LayananDAO, dan PelangganDAO berisi deklarasi method seperti save, update, delete, dan show. DAO memastikan konsistensi penggunaan method dalam mengakses database.

### 1. UserRepo.java

```

package DAO;

import java.sql.*;
import java.util.*;
import model.User;
import config.Database;

public class UserRepo implements UserDAO {

    private Connection connection;
    private final String insert = "INSERT INTO user (name, username, password) VALUES (?, ?, ?)";
    private final String select = "SELECT * FROM user";
    private final String delete = "DELETE FROM user WHERE id=?";
    private final String update = "UPDATE user SET name=?, username=?, password=? WHERE id=?";

    public UserRepo() {
        this.connection = Database.koneksi();
    }

    @Override
    public void save(User user) {
        try {
            PreparedStatement 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();
        }
    }

    @Override
    public List<User> show() {

```

```

        List<User> ls = new ArrayList<>();
        try {
            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) {
            e.printStackTrace();
        }
        return ls;
    }

    @Override
    public void update(User user) {
        try {
            PreparedStatement 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();
        }
    }

    @Override
    public void delete(String id) {
        try {
            PreparedStatement st = connection.prepareStatement(delete);
            st.setString(1, id);
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

## 2. PelangganRepo.java

```

package DAO;

import config.Database;
import model.Pelanggan;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PelangganRepo implements PelangganDAO {
    private Connection conn;

    public PelangganRepo() {
        conn = Database.koneksi();
    }
}

```

```

    }

    @Override
    public void save(Pelanggan p) {
        String sql = "INSERT INTO pelanggan(id, nama, alamat, no_hp)
VALUES(?,?,?,?)";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            if (p.getId() == null)
                p.setId(String.valueOf(System.currentTimeMillis()));
            ps.setString(1, p.getId());
            ps.setString(2, p.getNama());
            ps.setString(3, p.getAlamat());
            ps.setString(4, p.getNoHp());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void update(Pelanggan p) {
        String sql = "UPDATE pelanggan SET nama = ?, alamat = ?, no_hp = ? WHERE
id = ?";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, p.getNama());
            ps.setString(2, p.getAlamat());
            ps.setString(3, p.getNoHp());
            ps.setString(4, p.getId());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void delete(String id) {
        String sql = "DELETE FROM pelanggan WHERE id = ?";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public List<Pelanggan> show() {
        List<Pelanggan> list = new ArrayList<>();
        String sql = "SELECT * FROM pelanggan";
        try (Statement st = conn.createStatement(); ResultSet rs =
st.executeQuery(sql)) {
            while (rs.next()) {
                Pelanggan p = new Pelanggan();
                p.setId(rs.getString("id"));
                p.setNama(rs.getString("nama"));
                p.setAlamat(rs.getString("alamat"));
                p.setNoHp(rs.getString("no_hp"));
                list.add(p);
            }
        }
    }

```

```

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }
}

```

### 3. LayananRepo.java

```

package DAO;

import config.Database;
import model.Layanan;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class LayananRepo implements LayananDAO {
    private Connection conn;

    public LayananRepo() {
        conn = Database.koneksi();
    }

    @Override
    public void save(Layanan l) {
        String sql = "INSERT INTO layanan(id, nama_layanan, harga) VALUES(?,?,?)";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            if (l.getId() == null)
                l.setId(String.valueOf(System.currentTimeMillis()));
            ps.setString(1, l.getId());
            ps.setString(2, l.getNamaLayanan());
            ps.setDouble(3, l.getHarga());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void update(Layanan l) {
        String sql = "UPDATE layanan SET nama_layanan = ?, harga = ? WHERE id = ?";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, l.getNamaLayanan());
            ps.setDouble(2, l.getHarga());
            ps.setString(3, l.getId());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void delete(String id) {
        String sql = "DELETE FROM layanan WHERE id = ?";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, id);
            ps.executeUpdate();
        }
    }
}

```

```

    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@Override
public List<Layanan> show() {
    List<Layanan> list = new ArrayList<>();
    String sql = "SELECT * FROM layanan";
    try (Statement st = conn.createStatement(); ResultSet rs =
st.executeQuery(sql)) {
        while (rs.next()) {
            Layanan l = new Layanan();
            l.setId(rs.getString("id"));
            l.setNamaLayanan(rs.getString("nama_layanan"));
            l.setHarga(rs.getDouble("harga"));
            list.add(l);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
}

```

## G. Membuat Table Model

Table model mengatur bagaimana data ditampilkan pada JTable. TableUsers, TablePelanggan, dan TableLayanan menampilkan data dari list model masing-masing ke tabel. Dengan ini, data yang ada di database dapat ditampilkan dalam bentuk tabel interaktif.

### 1. TableUsers.java

```

package table;

import javax.swing.table.AbstractTableModel;
import java.util.List;
import model.User; //

public class TableUsers extends AbstractTableModel {

    private List<User> ls;
    private String[] columnNames = {"ID", "Name", "Username", "Password"};

    public TableUsers(List<User> 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) {
    User u = ls.get(rowIndex);
    switch (columnIndex) {
        case 0: return u.getId();
        case 1: return u.getNama();
        case 2: return u.getUsername();
        case 3: return u.getPassword();
        default: return null;
    }
}

public void setList(List<User> ls) {
    this.ls = ls;
    fireTableDataChanged();
}

public User getUserAt(int rowIndex) {
    return ls.get(rowIndex);
}
}

```

## 2. Table Pelanggan.java

```

package table;

import model.Pelanggan;

import javax.swing.table.AbstractTableModel;
import java.util.List;

public class TablePelanggan extends AbstractTableModel {
    private List<Pelanggan> list;
    private String[] columns = {"ID", "Nama", "Alamat", "No HP"};

    public TablePelanggan(List<Pelanggan> list) {
        this.list = list;
    }

    @Override
    public int getRowCount() {
        return list == null ? 0 : list.size();
    }

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

```

```

    }

    @Override
    public String getColumnName(int column) {
        return columns[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Pelanggan p = list.get(rowIndex);
        switch (columnIndex) {
            case 0: return p.getId();
            case 1: return p.getNama();
            case 2: return p.getAlamat();
            case 3: return p.getNoHp();
            default: return null;
        }
    }
}

```

### 3. Table Layanan

```

package table;

import model.Layanan;

import javax.swing.table.AbstractTableModel;
import java.util.List;

public class TableLayanan extends AbstractTableModel {
    private List<Layanan> list;
    private String[] columns = {"ID", "Nama Layanan", "Harga"};

    public TableLayanan(List<Layanan> list) {
        this.list = list;
    }

    @Override
    public int getRowCount() {
        return list == null ? 0 : list.size();
    }

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

    @Override
    public String getColumnName(int column) {
        return columns[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Layanan l = list.get(rowIndex);
        switch (columnIndex) {
            case 0: return l.getId();
            case 1: return l.getNamaLayanan();
        }
    }
}

```



```

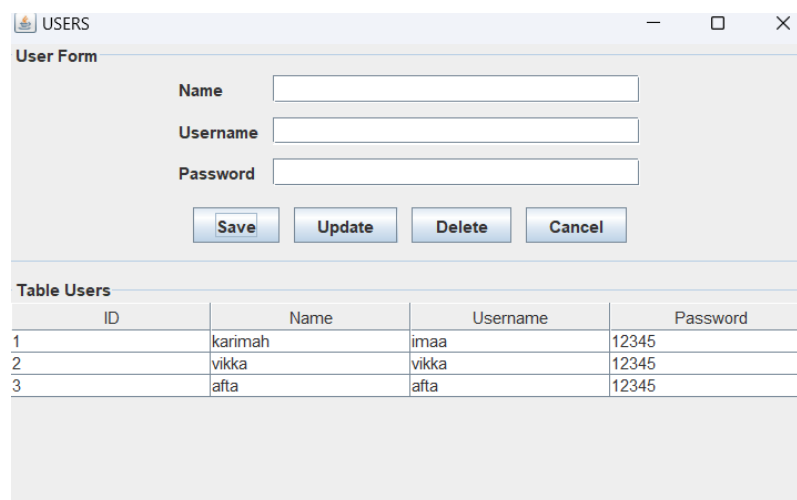
        case 2: return l.getHarga();
        default: return null;
    }
}
}

```

## H. Membuat Frame CRUD

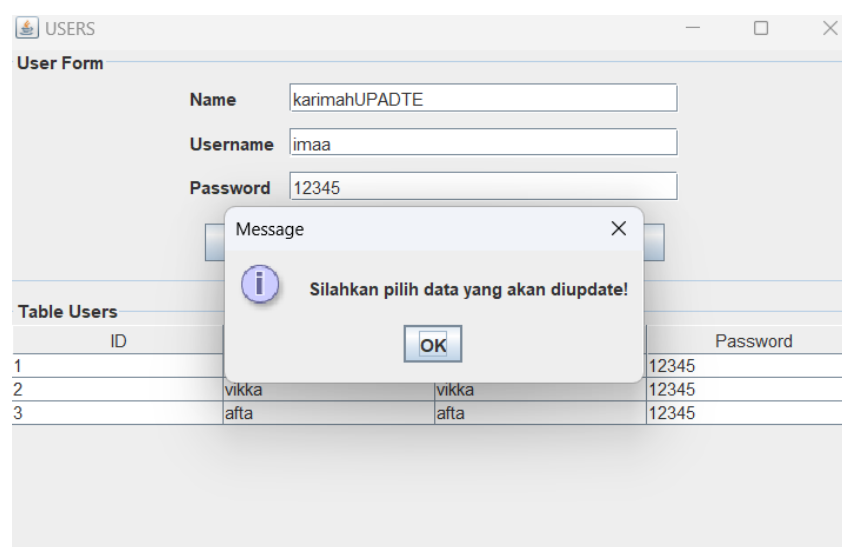
Frame CRUD adalah form GUI untuk mengelola data. UserFrame digunakan untuk mengelola data user. PelangganFrame digunakan untuk data pelanggan. LayananFrame digunakan untuk data layanan laundry. Setiap frame memiliki input field, tombol (Save, Update, Delete, Cancel), serta tabel. Tombol dihubungkan dengan method CRUD di repository, sehingga data langsung tersimpan atau berubah di database.

### 1. UserFrame.java



The screenshot shows a window titled "USERS" with a "User Form" section. It contains three input fields: "Name", "Username", and "Password". Below the fields are four buttons: "Save", "Update", "Delete", and "Cancel". At the bottom, there is a table titled "Table Users" with the following data:

ID	Name	Username	Password
1	karimah	imaa	12345
2	vikka	vikka	12345
3	afta	afta	12345



The screenshot shows the same "User Form" window, but with a "Message" dialog box overlaying the "Table Users". The dialog box contains an information icon and the text: "Silahkan pilih data yang akan diupdate!". There is an "OK" button in the dialog box. The input fields in the background are filled with the values: Name: "karimahUPADTE", Username: "imaa", and Password: "12345".

USERS

User Form

Name

Username

Password

Save Update Delete Cancel

Table Users

ID	Name	Username	Password
1	karimahUPADTE	imaa	12345
2	vikka	vikka	12345
3	afta	afta	12345

USERS

User Form

Name

Username

Password

Save Update Delete Cancel

Message

**Silahkan pilih data yang akan dihapus!**

OK

Table Users

ID	Name	Username	Password
1	karimahUPADTE	imaa	12345
2	vikka	vikka	12345
3	afta	afta	12345

USERS

User Form

Name

Username

Password

Save Update Delete Cancel

Table Users

ID	Name	Username	Password
2	vikka	vikka	12345
3	afta	afta	12345

**User Form**

Name:

Username:

Password:

**Table Users**

ID	Name	Username	Password
2	vikka	vikka	12345
3	afta	afta	12345

*Cancel berfungsi*

```
package model;

import DAO.UserRepo;
import table.TableUsers;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class UserFrame extends JFrame {

    // Deklarasi komponen
    private JTextField txtName;
    private JTextField txtUsername;
    private JTextField txtPassword;
    private JButton btnSave, btnUpdate, btnDelete, btnCancel;
    private JTable tableUsers;

    // DAO & Data
    UserRepo usr = new UserRepo();
    List<User> ls;
    public String id;

    public UserFrame() {
        initComponents();
        loadTable(); // langsung tampilkan data saat frame dibuka
        initActions(); // event handler tombol
    }

    private void initComponents() {
        setTitle("USERS");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Panel utama
        JPanel panelMain = new JPanel(new BorderLayout(10, 10));
```

```

getContentPane().add(panelMain);

// Panel input data
JPanel panelInput = new JPanel(new GridBagLayout());
panelInput.setBorder(BorderFactory.createTitledBorder("User Form"));

GridBagConstraints gbc;

// Label Name
gbc = new GridBagConstraints();
gbc.gridx = 0; gbc.gridy = 0;
gbc.insets = new Insets(5, 5, 5, 5);
gbc.anchor = GridBagConstraints.WEST;
panelInput.add(new JLabel("Name"), gbc);

// TextField Name
gbc = new GridBagConstraints();
gbc.gridx = 1; gbc.gridy = 0;
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;
txtName = new JTextField(20);
panelInput.add(txtName, gbc);

// Label Username
gbc = new GridBagConstraints();
gbc.gridx = 0; gbc.gridy = 1;
gbc.insets = new Insets(5, 5, 5, 5);
gbc.anchor = GridBagConstraints.WEST;
panelInput.add(new JLabel("Username"), gbc);

// TextField Username
gbc = new GridBagConstraints();
gbc.gridx = 1; gbc.gridy = 1;
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;
txtUsername = new JTextField(20);
panelInput.add(txtUsername, gbc);

// Label Password
gbc = new GridBagConstraints();
gbc.gridx = 0; gbc.gridy = 2;
gbc.insets = new Insets(5, 5, 5, 5);
gbc.anchor = GridBagConstraints.WEST;
panelInput.add(new JLabel("Password"), gbc);

// TextField Password
gbc = new GridBagConstraints();
gbc.gridx = 1; gbc.gridy = 2;
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;
txtPassword = new JTextField(20);
panelInput.add(txtPassword, gbc);

// Panel tombol
JPanel panelButton = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
5));

btnSave = new JButton("Save");
btnUpdate = new JButton("Update");
btnDelete = new JButton("Delete");

```

```

        btnCancel = new JButton("Cancel");
        panelButton.add(btnSave);
        panelButton.add(btnUpdate);
        panelButton.add(btnDelete);
        panelButton.add(btnCancel);

        gbc = new GridBagConstraints();
        gbc.gridx = 0; gbc.gridy = 3;
        gbc.gridwidth = 2;
        gbc.insets = new Insets(5, 5, 5, 5);
        panelInput.add(panelButton, gbc);

        panelMain.add(panelInput, BorderLayout.NORTH);

        // Tabel users
        tableUsers = new JTable();
        JScrollPane scrollPane = new JScrollPane(tableUsers);
        scrollPane.setBorder(BorderFactory.createTitledBorder("Table Users"));
        panelMain.add(scrollPane, BorderLayout.CENTER);
    }

    // Method untuk clear input
    public void reset() {
        txtName.setText("");
        txtUsername.setText("");
        txtPassword.setText("");
        id = null;
    }

    // Method untuk tampilkan data di JTable
    public void loadTable() {
        ls = usr.show();
        TableUsers tu = new TableUsers(ls);
        tableUsers.setModel(tu);
        tableUsers.getTableHeader().setVisible(true);
    }

    // Event handler tombol & tabel
    private void initActions() {

        // Tombol SAVE
        btnSave.addActionListener(e -> {
            User user = new User();
            user.setNama(txtName.getText());
            user.setUsername(txtUsername.getText());
            user.setPassword(txtPassword.getText());

            usr.save(user);
            reset();
            loadTable();
        });

        // Tombol UPDATE
        btnUpdate.addActionListener(e -> {
            if (id != null) {
                User user = new User();
                user.setId(id);
                user.setNama(txtName.getText());
                user.setUsername(txtUsername.getText());
            }
        });
    }

```

```

        user.setPassword(txtPassword.getText());

        usr.update(user);
        reset();
        loadTable();
    } else {
        JOptionPane.showMessageDialog(this, "Silahkan pilih data yang
akan diupdate!");
    }
});

// Tombol DELETE
btnDelete.addActionListener(e -> {
    if (id != null) {
        usr.delete(id);
        reset();
        loadTable();
    } else {
        JOptionPane.showMessageDialog(this, "Silahkan pilih data yang
akan dihapus!");
    }
});

// Tombol CANCEL
btnCancel.addActionListener(e -> reset());

// Klik tabel isi form
tableUsers.addMouseListener(new java.awt.event.MouseAdapter() {
    @Override
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        int row = tableUsers.getSelectedRow();
        if (row >= 0) {
            id = tableUsers.getValueAt(row, 0).toString();
            txtName.setText(tableUsers.getValueAt(row, 1).toString());
            txtUsername.setText(tableUsers.getValueAt(row,
2).toString());
            txtPassword.setText(tableUsers.getValueAt(row,
3).toString());
        }
    }
});
}

// Main method
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        UserFrame frame = new UserFrame();
        frame.setVisible(true);
    });
}
}

```

## 2. PelangganFrame.java

PELANGGAN

**Form Pelanggan**

Nama

Alamat

No HP

Save Update Delete Cancel

**Data Pelanggan**

ID	Nama	Alamat	No HP
1757869243631	ima	andalas	085174120405
1757869254000	vikka	andalas	0225158445
1757869264433	afta	andalas	028154854555

PELANGGAN

**Form Pelanggan**

Nama

Alamat

No HP

Save Update Delete Cancel

**Data Pelanggan**

ID	Nama	Alamat	No HP
1757869243631	ima	andalas	085174120405
1757869254000	vikka	andalas	0225158445
1757869264433	afta	andalas	028154854555

PELANGGAN

Form Pelanggan

Nama

Alamat

No HP

Save Update Delete Cancel

Data Pelanggan

ID	Nama	Alamat	No HP
1757869243631	IMA UPDATE	andalas	085174120405
1757869254000	vikka	andalas	0225158445
1757869264433	afta	andalas	028154854555

PELANGGAN

Form Pelanggan

Nama

Alamat

No HP

Save Update Delete Cancel

Data Pelanggan

ID	Nama	Alamat	No HP
1757869243631	IMA UP		085174120405
1757869254000	vikka		0225158445
1757869264433	afta		028154854555

Konfirmasi

?

Yakin akan menghapus?

Yes No



PELANGGAN

Form Pelanggan

Nama

Alamat

No HP

Save Update Delete Cancel

Data Pelanggan

ID	Nama	Alamat	No HP
1757869254000	vikka	andalas	0225158445
1757869264433	afta	andalas	028154854555

```
package model;

import DAO.PelangganRepo;
import table.TablePelanggan;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class PelangganFrame extends JFrame {
    private JTextField txtNama, txtAlamat, txtNoHp;
    private JButton btnSave, btnUpdate, btnDelete, btnCancel;
    private.JTable tablePelanggan;

    private PelangganRepo repo = new PelangganRepo();
    private List<Pelanggan> ls;
    private String id;

    public PelangganFrame() {
        initComponents();
        loadTable();
        initActions();
    }

    private void initComponents() {
        setTitle("PELANGGAN");
        setSize(700, 450);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel main = new JPanel(new BorderLayout(10,10));
        getContentPane().add(main);

        JPanel input = new JPanel(new GridBagLayout());
        input.setBorder(BorderFactory.createTitledBorder("Form Pelanggan"));
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5,5,5,5);

        // Nama
```

```

        gbc.gridx = 0; gbc.gridy = 0; gbc.anchor = GridBagConstraints.WEST;
        input.add(new JLabel("Nama"), gbc);
        gbc.gridx = 1; gbc.fill = GridBagConstraints.HORIZONTAL;
        txtNama = new JTextField(25);
        input.add(txtNama, gbc);

        // Alamat
        gbc.gridx = 0; gbc.gridy = 1; gbc.fill = GridBagConstraints.NONE;
        input.add(new JLabel("Alamat"), gbc);
        gbc.gridx = 1; gbc.fill = GridBagConstraints.HORIZONTAL;
        txtAlamat = new JTextField(25);
        input.add(txtAlamat, gbc);

        // No HP
        gbc.gridx = 0; gbc.gridy = 2; gbc.fill = GridBagConstraints.NONE;
        input.add(new JLabel("No HP"), gbc);
        gbc.gridx = 1; gbc.fill = GridBagConstraints.HORIZONTAL;
        txtNoHp = new JTextField(25);
        input.add(txtNoHp, gbc);

        // Buttons
        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 5));
        btnSave = new JButton("Save");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnCancel = new JButton("Cancel");
        btnPanel.add(btnSave); btnPanel.add(btnUpdate); btnPanel.add(btnDelete);
        btnPanel.add(btnCancel);

        gbc.gridx = 0; gbc.gridy = 3; gbc.gridwidth = 2;
        input.add(btnPanel, gbc);

        main.add(input, BorderLayout.NORTH);

        tablePelanggan = new JTable();
        JScrollPane sp = new JScrollPane(tablePelanggan);
        sp.setBorder(BorderFactory.createTitledBorder("Data Pelanggan"));
        main.add(sp, BorderLayout.CENTER);
    }

    private void reset() {
        txtNama.setText("");
        txtAlamat.setText("");
        txtNoHp.setText("");
        id = null;
    }

    private void loadTable() {
        ls = repo.show();
        TablePelanggan tp = new TablePelanggan(ls);
        tablePelanggan.setModel(tp);
        tablePelanggan.getTableHeader().setVisible(true);
    }

    private void initActions() {
        btnSave.addActionListener(e -> {
            Pelanggan p = new Pelanggan();
            p.setId(String.valueOf(System.currentTimeMillis()));
            p.setNama(txtNama.getText());

```

```

        p.setAlamat(txtAlamat.getText());
        p.setNoHp(txtNoHp.getText());

        repo.save(p);
        reset();
        loadTable();
    });

    btnUpdate.addActionListener(e -> {
        if (id != null) {
            Pelanggan p = new Pelanggan();
            p.setId(id);
            p.setNama(txtNama.getText());
            p.setAlamat(txtAlamat.getText());
            p.setNoHp(txtNoHp.getText());

            repo.update(p);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(this, "Silahkan pilih data yang akan
diupdate!");
        }
    });

    btnDelete.addActionListener(e -> {
        if (id != null) {
            int confirm = JOptionPane.showConfirmDialog(this, "Yakin akan
menghapus?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
            if (confirm == JOptionPane.YES_OPTION) {
                repo.delete(id);
                reset();
                loadTable();
            }
        } else {
            JOptionPane.showMessageDialog(this, "Silahkan pilih data yang akan
dihapus!");
        }
    });

    btnCancel.addActionListener(e -> reset());

    tablePelanggan.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            int row = tablePelanggan.getSelectedRow();
            if (row >= 0) {
                id = tablePelanggan.getValueAt(row, 0).toString();
                txtNama.setText(tablePelanggan.getValueAt(row, 1).toString());
                txtAlamat.setText(tablePelanggan.getValueAt(row,
2).toString());
                txtNoHp.setText(tablePelanggan.getValueAt(row, 3).toString());
            }
        }
    });
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new PelangganFrame().setVisible(true));
}
}

```

### 3. LayananFrame.java

LAYANAN

Form Layanan

Nama Layanan

Harga

Data Layanan

ID	Nama Layanan	Harga
1757869394522	WEBSITE	500000.0
1757869414672	MOBILE	8000000.0
1757869425507	DESIGN FEED	500000.0

LAYANAN

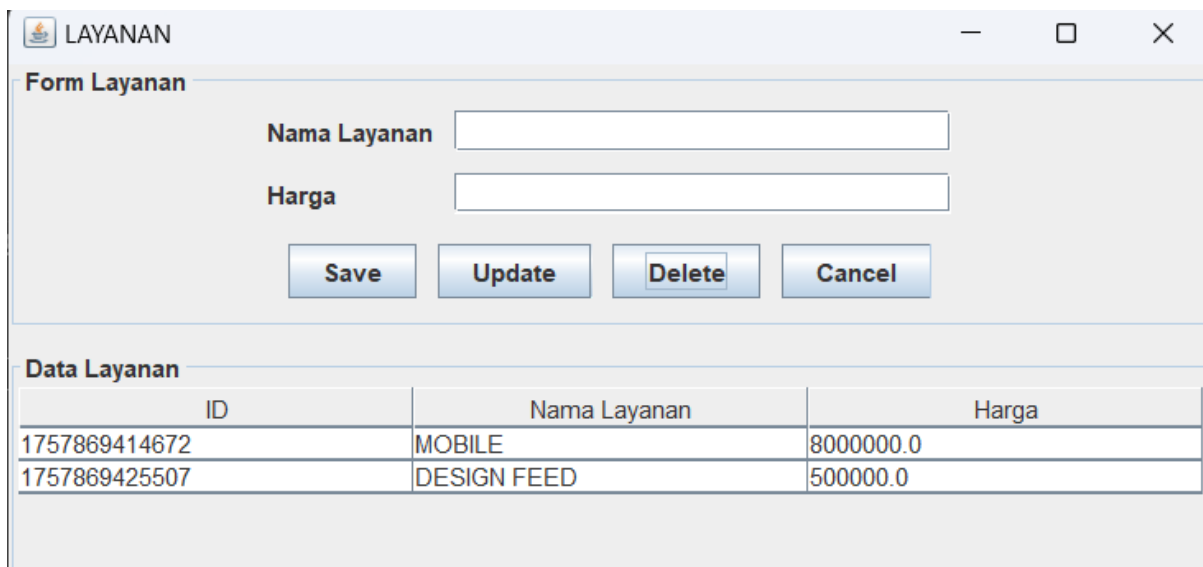
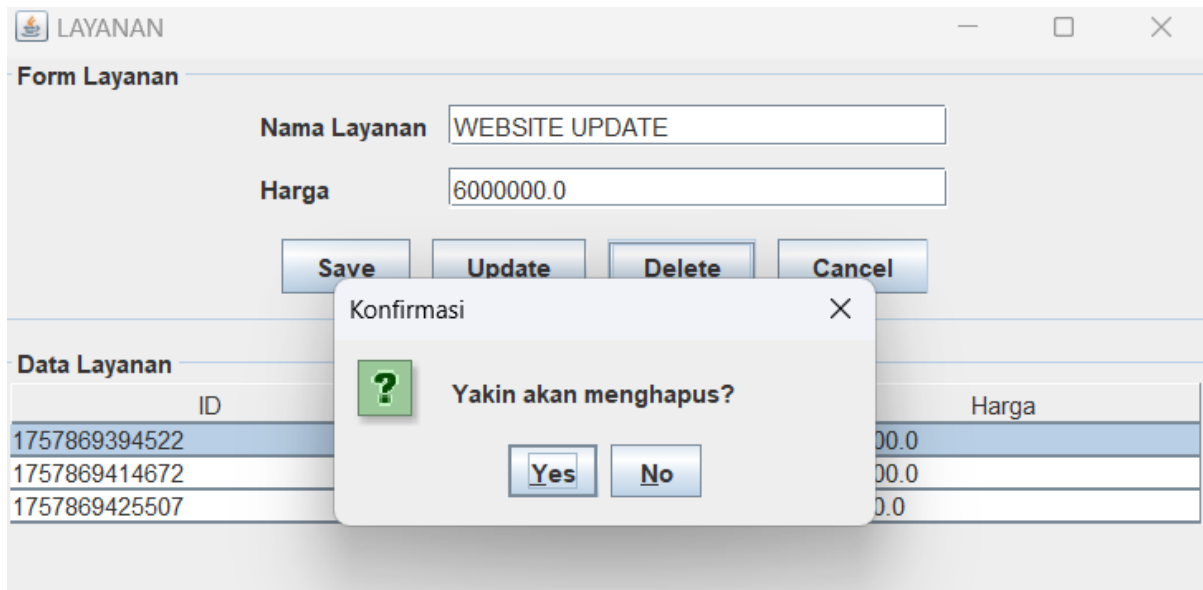
Form Layanan

Nama Layanan

Harga

Data Layanan

ID	Nama Layanan	Harga
1757869394522	WEBSITE UPDATE	6000000.0
1757869414672	MOBILE	8000000.0
1757869425507	DESIGN FEED	500000.0



```
package model;

import DAO.LayananRepo;
import table.TableLayanan;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class LayananFrame extends JFrame {
    private JTextField txtNama, txtHarga;
    private JButton btnSave, btnUpdate, btnDelete, btnCancel;
    private.JTable tableLayanan;

    private LayananRepo repo = new LayananRepo();
    private List<Layanan> ls;
    private String id;

    public LayananFrame() {
        initComponents();
    }
}
```

```

        loadTable();
        initActions();
    }

    private void initComponents() {
        setTitle("LAYANAN");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel main = new JPanel(new BorderLayout(10,10));
        getContentPane().add(main);

        JPanel input = new JPanel(new GridBagLayout());
        input.setBorder(BorderFactory.createTitledBorder("Form Layanan"));
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5,5,5,5);

        // Nama
        gbc.gridx = 0; gbc.gridy = 0; gbc.anchor = GridBagConstraints.WEST;
        input.add(new JLabel("Nama Layanan"), gbc);
        gbc.gridx = 1; gbc.fill = GridBagConstraints.HORIZONTAL;
        txtNama = new JTextField(20);
        input.add(txtNama, gbc);

        // Harga
        gbc.gridx = 0; gbc.gridy = 1; gbc.fill = GridBagConstraints.NONE;
        input.add(new JLabel("Harga"), gbc);
        gbc.gridx = 1; gbc.fill = GridBagConstraints.HORIZONTAL;
        txtHarga = new JTextField(20);
        input.add(txtHarga, gbc);

        // Buttons
        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 5));
        btnSave = new JButton("Save");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnCancel = new JButton("Cancel");
        btnPanel.add(btnSave); btnPanel.add(btnUpdate); btnPanel.add(btnDelete);
        btnPanel.add(btnCancel);

        gbc.gridx = 0; gbc.gridy = 2; gbc.gridwidth = 2;
        input.add(btnPanel, gbc);

        main.add(input, BorderLayout.NORTH);

        // Table
        tableLayanan = new JTable();
        JScrollPane sp = new JScrollPane(tableLayanan);
        sp.setBorder(BorderFactory.createTitledBorder("Data Layanan"));
        main.add(sp, BorderLayout.CENTER);
    }

    private void reset() {
        txtNama.setText("");
        txtHarga.setText("");
        id = null;
    }

```

```

private void loadTable() {
    ls = repo.show();
    TableLayanan tu = new TableLayanan(ls);
    tableLayanan.setModel(tu);
    tableLayanan.getTableHeader().setVisible(true);
}

private void initActions() {
    btnSave.addActionListener(e -> {
        try {
            Layanan l = new Layanan();
            l.setId(String.valueOf(System.currentTimeMillis()));
            l.setNamaLayanan(txtNama.getText());
            l.setHarga(Double.parseDouble(txtHarga.getText()));

            repo.save(l);
            reset();
            loadTable();
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Harga harus angka.");
        }
    });

    btnUpdate.addActionListener(e -> {
        if (id != null) {
            try {
                Layanan l = new Layanan();
                l.setId(id);
                l.setNamaLayanan(txtNama.getText());
                l.setHarga(Double.parseDouble(txtHarga.getText()));

                repo.update(l);
                reset();
                loadTable();
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Harga harus angka.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Silahkan pilih data yang akan diupdate!");
        }
    });

    btnDelete.addActionListener(e -> {
        if (id != null) {
            int confirm = JOptionPane.showConfirmDialog(this, "Yakin akan menghapus?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
            if (confirm == JOptionPane.YES_OPTION) {
                repo.delete(id);
                reset();
                loadTable();
            }
        } else {
            JOptionPane.showMessageDialog(this, "Silahkan pilih data yang akan dihapus!");
        }
    });

    btnCancel.addActionListener(e -> reset());
}

```

```

        tableLayanan.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                int row = tableLayanan.getSelectedRow();
                if (row >= 0) {
                    id = tableLayanan.getValueAt(row, 0).toString();
                    txtNama.setText(tableLayanan.getValueAt(row, 1).toString());
                    txtHarga.setText(tableLayanan.getValueAt(row, 2).toString());
                }
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new LayananFrame().setVisible(true);
        });
    }
}

```

## I. Membuat Login & Main Menu

### 1. File: ui/LoginFrame.java

Memverifikasi user (username & password). Jika benar, buka MainFrame.

### 2. File: ui/MainFrame.java

Menampilkan tombol User, Layanan, Pelanggan. Tombol akan membuka frame sesuai.





## **BAB III**

### **PENUTUP**

#### **3.1 Kesimpulan**

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa implementasi konsep CRUD (Create, Read, Update, Delete) pada aplikasi Laundry Apps menggunakan Java Swing dan MySQL dapat berjalan dengan baik. Penerapan pola desain DAO (Data Access Object) serta pemisahan antara model, repository, table model, dan tampilan membuat struktur program lebih rapi, mudah dipahami, serta fleksibel untuk dikembangkan. Dengan adanya antarmuka berbasis GUI, pengguna dapat melakukan pengelolaan data layanan, pelanggan, dan pengguna dengan lebih mudah dan interaktif. Selain itu, penggunaan JDBC memungkinkan integrasi langsung antara aplikasi Java dengan database MySQL sehingga data yang disimpan, diubah, maupun dihapus dapat diperbarui secara real time. Praktikum ini menunjukkan bahwa Java dapat digunakan untuk membangun aplikasi sederhana yang terhubung dengan database dan memiliki fitur login, menu utama, serta form pengolahan data secara lengkap.