

**LAPORAN PRAKTIKUM
PEMOGRAMAN BERORIENTASI OBJEK**

PRAKTIKUM 3:

“Menggunakan Fungsi CRUD DAO pada GUI”



**Disusun oleh:
Putri Balqis Afradinata
2411531009**

**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS
SEPTEMBER
2025**

i. PENDAHULUAN

Pemrograman Berorientasi Objek (PBO) tidak hanya menekankan pada konsep kelas, objek, pewarisan, maupun enkapsulasi, tetapi juga bagaimana penerapannya dalam aplikasi nyata. Salah satu penerapan penting adalah pembuatan aplikasi berbasis GUI (Graphical User Interface) yang terhubung dengan database.

Aplikasi berbasis database membutuhkan operasi dasar seperti Create, Read, Update, dan Delete (CRUD) untuk mengelola data. Agar pengelolaan data menjadi lebih terstruktur, konsep DAO (Data Access Object) digunakan. DAO memisahkan kode yang berhubungan dengan database dari kode logika bisnis maupun tampilan.

Dalam praktikum ini, mahasiswa mengimplementasikan fungsi CRUD menggunakan DAO pada GUI dengan bahasa pemrograman Java, menggunakan Swing sebagai library GUI, serta MySQL sebagai database.

ii. TUJUAN

1. Menerapkan pola DAO pada aplikasi Java berbasis GUI.
2. Membuat fungsi CRUD (Create, Read, Update, Delete) pada data yang tersimpan di database.
3. Memisahkan kode program ke dalam Model, DAO/Repo, TableModel, dan Frame (UI) sesuai konsep MVC sederhana.
4. Membiasakan diri menghubungkan aplikasi Java dengan database MySQL menggunakan JDBC.

iii. LANGKAH-LANGKAH

1. Di praktikum sebelumnya kita sudah membuat fungsi UserDAO, selanjutnya kita akan menggunakan Fungsi CRUD DAO pada GUI
2. Buat method reset pada JFrame seperti kode program dibawah ini. Method digunakan untuk menghapus value inputan Ketika suatu proses berhasil dilakukan

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

3. Selanjutnya membuat instance pada UserFrame. Baris kode ini menyiapkan akses DAO (UserRepo), penampung data (List), dan penanda data (id) untuk keperluan CRUD pada tabel user.

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

4. Klik kanan pada tombol save → add event handlers → actionPerformed kemudian isi dengan kode program berikut

```
user user = new user();  
user.setName(name);  
user.setUsername(username);  
user.setPassword(password);  
  
usr.save(user);  
reset();  
loadTable();  
JOptionPane.showMessageDialog(null, "User berhasil disimpan.");
```

Kode tersebut berfungsi untuk menyimpan data user baru ke dalam database. Pertama dibuat objek user kemudian diisi dengan data dari input (name, username, password). Setelah itu objek dikirim ke `usr.save(user)` agar data benar-benar tersimpan. Metode `reset()` dipanggil untuk mengosongkan kembali field input, lalu `loadTable()` memperbarui tampilan tabel agar data baru muncul. Terakhir, ditampilkan pesan konfirmasi dengan `JOptionPane` bahwa user berhasil disimpan.

5. Buat method dengan nama `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);  
}
```

Method `loadTable()` digunakan untuk menampilkan data user ke dalam tabel. Pertama, data user diambil dari database melalui `usr.show()`. Data tersebut kemudian dimasukkan ke objek `TableUser` sebagai model tabel. Selanjutnya model ini dipasang ke `tableUsers` dengan `setModel(tu)`. Terakhir, bagian header tabel diaktifkan agar terlihat di tampilan GUI.

6. Memanggil method pada class main, sehingga Ketika pertama kali program

dijalankan maka loadTable akan dipanggil.

```
public static void main(String[] args) {  
    EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            try {  
                UserFrame frame = new UserFrame();  
                frame.setVisible(true);  
                frame.loadTable();  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
    });  
}
```

7. Klik kanan pada JTable → add event handler → mouse → mouseClicked, lalu isikan dengan kode di bawah:

```
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());
```

Kode program diatas berfungsi yaitu mengambil id user dan menyimpannya kedalam variable id kemudian mengambil data nama, username dan password dan ditampilkan kedalam form inputan.

8. Klik salah satu isi table maka akan secara otomatis tampil pada form inputan.

ID	Name	Username	Password
10	balqis	pedang	123456
11	putri	lirtau	654321

9. Klik kanan tombol update → add event handler → action → actionPerformed dan isikan dengan kode program berikut.

```
user user = new user();
user.setId(id);
user.setName(name);
user.setUsername(username);
user.setPassword(password);

usr.update(user);
reset();
loadTable();
JOptionPane.showMessageDialog(null, "User berhasil diupdate.");
```

Potongan kode ini digunakan untuk update data user. Pertama dibuat objek user baru, lalu atributnya (id, nama, username, password) diisi dari input pengguna. Setelah itu dipanggil `usr.update(user)` untuk menyimpan perubahan ke database. Kemudian form direset, tabel diperbarui dengan `loadTable()`, dan ditampilkan pesan konfirmasi bahwa data berhasil diupdate.

10. Klik salah satu data pada JTable, lalu klik kanan tombol delete → add event handler → action → actionPerformed dan isikan dengan kode program berikut

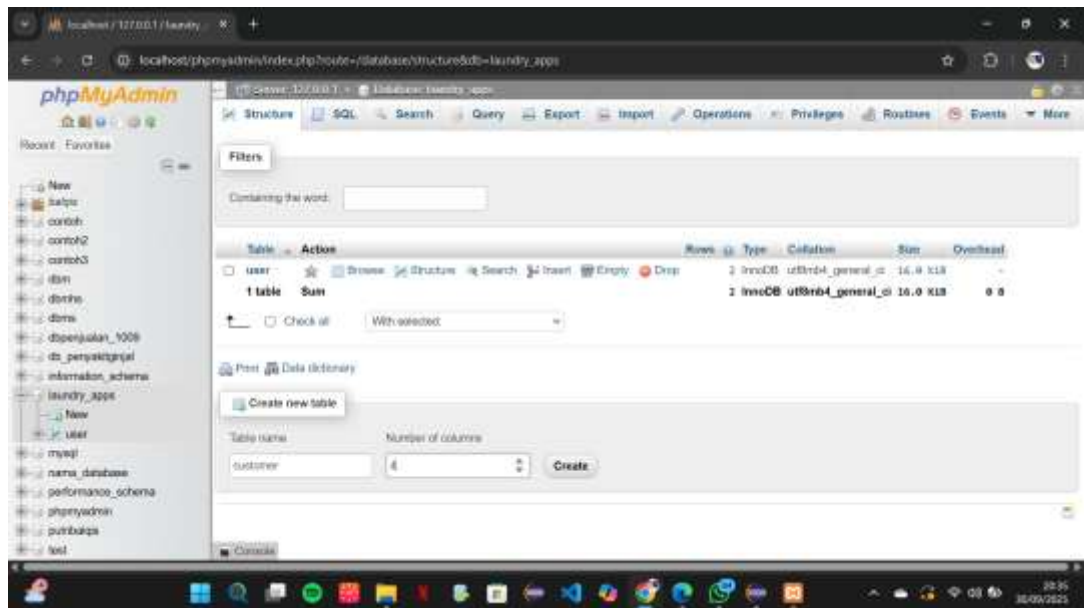
```
if(id != null) {
    usr.delete(id);
    reset();
    loadTable();
} else {
    JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
}
```

Potongan kode ini digunakan untuk menghapus data user. Jika id tidak kosong, maka data dengan id tersebut dihapus lewat `usr.delete(id)`, lalu form direset dan tabel diperbarui. Jika tidak ada data yang dipilih, ditampilkan pesan peringatan agar pengguna memilih data yang akan dihapus.

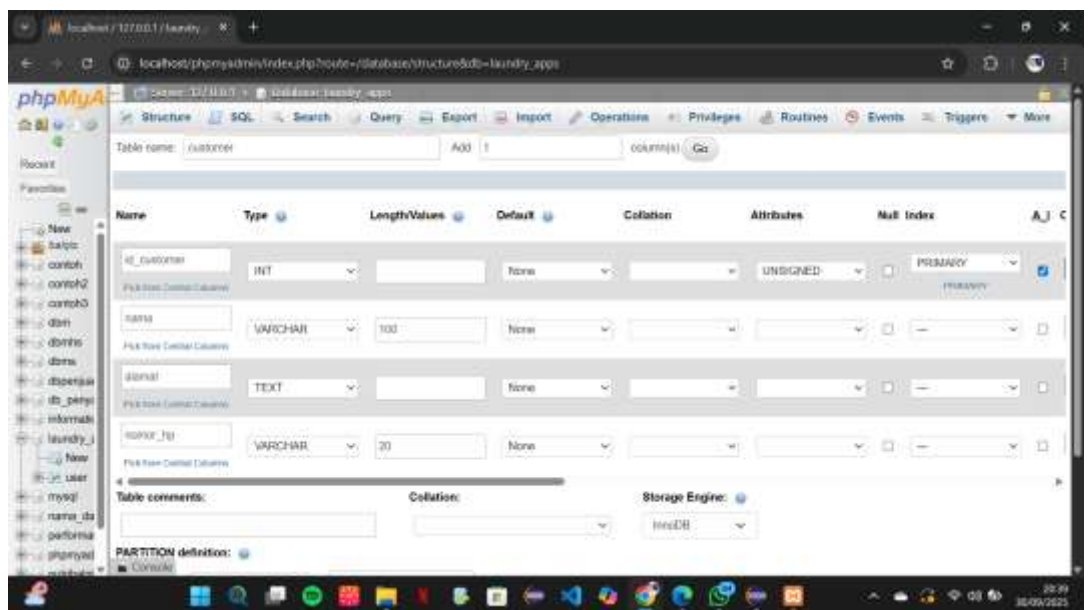
iv. TUGAS

➤ Fungsi CRUD Customer

1. Buka phpMyAdmin melalui XAMPP
2. Pilih database laundry_apps yang sudah dibuat sebelumnya, lalu buat table baru bernama “customer”



3. Lalu isi dengan ketentuan seperti di bawah ini



4. selanjutnya kita akan membuat class “CustomerDAO” pada package DAO

```
1 package DAO;
2 import java.util.List;
3 import model.customer;
4
5 public interface CustomerDAO {
6     void save(customer c);
7     List<customer> show();
8     void update(customer c);
9     void delete(String id); /
10 }
```

Kode di atas adalah interface CustomerDAO yang berisi kontrak fungsi CRUD

untuk entitas customer. Ada 4 method utama:

- save(customer c) → menyimpan data customer baru.
- show() → menampilkan daftar semua customer.
- update(customer c) → memperbarui data customer.
- delete(String id) → menghapus customer berdasarkan ID.

Interface ini memastikan setiap class yang mengimplementasikannya harus menyediakan implementasi lengkap untuk operasi CRUD tersebut.

5. lalu kita membuat class “CustomerRepo” dan membuat codingan seperti di bawah ini

```
1 package DAO;
2 import java.sql.*;
3 import java.util.*;
4 import javax.swing.JOptionPane;
5 import model.Customer;
6 import Config.Database;
7
8 public class CustomerRepo implements CustomerDAO {
9     private Connection conn = Database.Akses();
10
11     @Override
12     public void save(Customer c) {
13         String sql = "INSERT INTO customer (nama, alamat, nomor_hp) VALUES (?, ?, ?)";
14         try (PreparedStatement ps = conn.prepareStatement(sql)) {
15             ps.setString(1, c.getName());
16             ps.setString(2, c.getAlamat());
17             ps.setString(3, c.getNoHp());
18             ps.executeUpdate();
19             JOptionPane.showMessageDialog(null, "Customer berhasil ditambahkan!");
20         } catch (SQLException e) {
21             JOptionPane.showMessageDialog(null, "Gagal tambah customer: " + e.getMessage());
22         }
23     }
24
25     @Override
26     public List<Customer> show() {
27         List<Customer> list = new ArrayList<>();
28         String sql = "SELECT * FROM customer";
29         try (Statement st = conn.createStatement(); ResultSet rs = st.executeQuery(sql)) {
30             while (rs.next()) {
31                 Customer c = new Customer();
32                 c.setId(rs.getString("id_customer")); // id_customer di DB = field di kelas
33                 c.setName(rs.getString("nama"));
34                 c.setAlamat(rs.getString("alamat"));
35                 c.setNoHp(rs.getString("nomor_hp"));
36                 list.add(c);
37             }
38         } catch (SQLException e) {
39             JOptionPane.showMessageDialog(null, "Gagal load customer: " + e.getMessage());
40         }
41         return list;
42     }
43
44     @Override
45     public void update(Customer c) {
46         String sql = "UPDATE customer SET nama=?, alamat=?, nomor_hp=? WHERE id_customer=?";
47         try (PreparedStatement ps = conn.prepareStatement(sql)) {
48             ps.setString(1, c.getName());
49             ps.setString(2, c.getAlamat());
50             ps.setString(3, c.getNoHp());
51             ps.setString(4, c.getId()); // karena id di class String
52             ps.executeUpdate();
53             JOptionPane.showMessageDialog(null, "Customer berhasil diupdate!");
54         } catch (SQLException e) {
55             JOptionPane.showMessageDialog(null, "Gagal update customer: " + e.getMessage());
56         }
57     }
58
59     @Override
60     public void delete(String id) {
61         String sql = "DELETE FROM customer WHERE id_customer=?";
62         try (PreparedStatement ps = conn.prepareStatement(sql)) {
63             ps.setString(1, id);
64             ps.executeUpdate();
65             JOptionPane.showMessageDialog(null, "Customer berhasil dihapus!");
66         } catch (SQLException e) {
67             JOptionPane.showMessageDialog(null, "Gagal hapus customer: " + e.getMessage());
68         }
69     }
70 }
71
```

Kelas CustomerRepo berfungsi sebagai implementasi dari CustomerDAO yang

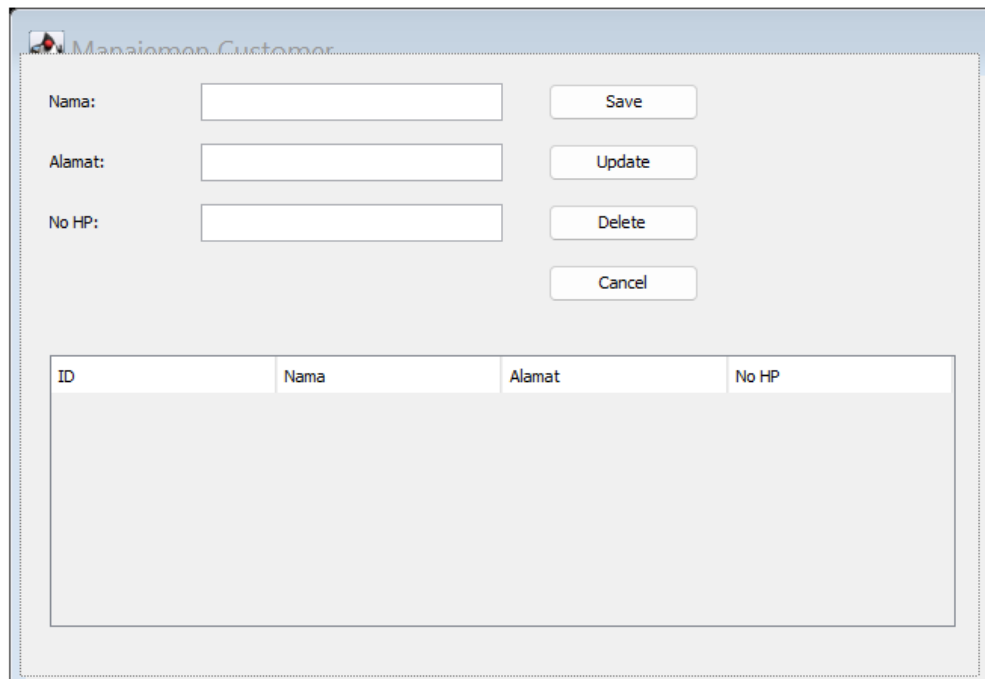
menangani logika akses database untuk tabel customer. Di dalamnya terdapat kode untuk menjalankan query SQL seperti menyimpan data baru (INSERT), menampilkan data (SELECT), mengubah data (UPDATE), dan menghapus data (DELETE). Jadi, kelas ini menjadi jembatan antara aplikasi (GUI) dengan database, agar data customer bisa diolah melalui fungsi CRUD.

6. Selanjutnya pada package Table kita tambahkan class “TableCustomer” seperti berikut

```
1 package table;
2
3 import javax.swing.table.AbstractTableModel;
4 import java.util.List;
5 import model.customer;
6
7 public class TableCustomer extends AbstractTableModel {
8     private List<Customer> list;
9
10    public TableCustomer(List<Customer> list) {
11        this.list = list;
12    }
13
14    @Override
15    public int getRowCount() {
16        return list.size();
17    }
18
19    @Override
20    public int getColumnCount() {
21        return 4; // id, name, alamat, noHp
22    }
23
24    @Override
25    public String getColumnName(int column) {
26        switch (column) {
27            case 0: return "ID";
28            case 1: return "Nama";
29            case 2: return "Alamat";
30            case 3: return "No HP";
31            default: return null;
32        }
33    }
34
35    @Override
36    public Object getValueAt(int rowIndex, int columnIndex) {
37        Customer c = list.get(rowIndex);
38        switch (columnIndex) {
39            case 0: return c.getId();
40            case 1: return c.getName();
41            case 2: return c.getAlamat();
42            case 3: return c.getNoHp();
43            default: return null;
44        }
45    }
46 }
47
```

Kelas TableCustomer berfungsi untuk mengatur tampilan data customer pada JTable di GUI. Kelas ini biasanya meng-extend AbstractTableModel dan menampung daftar objek customer, lalu menentukan kolom, baris, serta nilai apa yang akan ditampilkan di tabel. Singkatnya, TableCustomer adalah adapter yang menghubungkan data customer dari database agar bisa ditampilkan rapi dalam bentuk tabel di aplikasi.

7. Selanjutnya kita membuat class baru bernama “CustomerFrame” pada package ui untuk membuat desainnya



Manajemen Customer

Nama: Save

Alamat: Update

No HP: Delete

Cancel

ID	Nama	Alamat	No HP
----	------	--------	-------

```
67
68     repo = new CustomerRepo();
69     loadTable();
70
71     // Event Tombol Save
72     btnSave.addActionListener(new java.awt.event.ActionListener() {
73         public void actionPerformed(java.awt.event.ActionEvent e) {
74             customer c = new customer();
75             c.setNama(txtNama.getText());
76             c.setAlamat(txtAlamat.getText());
77             c.setNoHp(txtNoHp.getText());
78             repo.save(c);
79             loadTable();
80             reset();
81         }
82     });
83
84     // Event Tombol Update
85     btnUpdate.addActionListener(new java.awt.event.ActionListener() {
86         public void actionPerformed(java.awt.event.ActionEvent e) {
87             if (selectedId != null) {
88                 customer c = new customer();
89                 c.setId(selectedId);
90                 c.setNama(txtNama.getText());
91                 c.setAlamat(txtAlamat.getText());
92                 c.setNoHp(txtNoHp.getText());
93                 repo.update(c);
94                 loadTable();
95                 reset();
96             }
97         }
98     });
```

```

100 // Tombol Delete
101 btnDelete.addActionListener(new java.awt.event.ActionListener() {
102     public void actionPerformed(java.awt.event.ActionEvent e) {
103         if (selectedId != null) {
104             repo.delete(selectedId);
105             loadTable();
106             reset();
107         }
108     }
109 });
110
111 // Tombol Cancel
112 btnCancel.addActionListener(new java.awt.event.ActionListener() {
113     public void actionPerformed(java.awt.event.ActionEvent e) {
114         reset();
115     }
116 });
117
118 // Event Ulang Tabel
119 table.getSelectionModel().addListSelectionListener(new javax.swing.event.ListSelectionListener() {
120     public void valueChanged(javax.swing.event.ListSelectionEvent e) {
121         int row = table.getSelectedRow();
122         if (row != -1) {
123             selectedId = (String) table.getValueAt(row, 0);
124             txtNama.setText((String) table.getValueAt(row, 1));
125             txtAlamat.setText((String) table.getValueAt(row, 2));
126             txtNoHp.setText((String) table.getValueAt(row, 3));
127         }
128     }
129 });

```

```

132 // Muat ulang data tabel
133 private void loadTable() {
134     List<customer> list = repo.show();
135     TableCustomer model = new TableCustomer(list);
136     table.setModel(model);
137 }
138
139 // Reset input
140 private void reset() {
141     txtNama.setText("");
142     txtAlamat.setText("");
143     txtNoHp.setText("");
144     selectedId = null;
145 }
146
147 // Untuk testing langsung jalankan CustomerFrame
148 public static void main(String[] args) {
149     SwingUtilities.invokeLater(new Runnable() {
150         public void run() {
151             new CustomerFrame().setVisible(true);
152         }
153     });
154 }
155

```

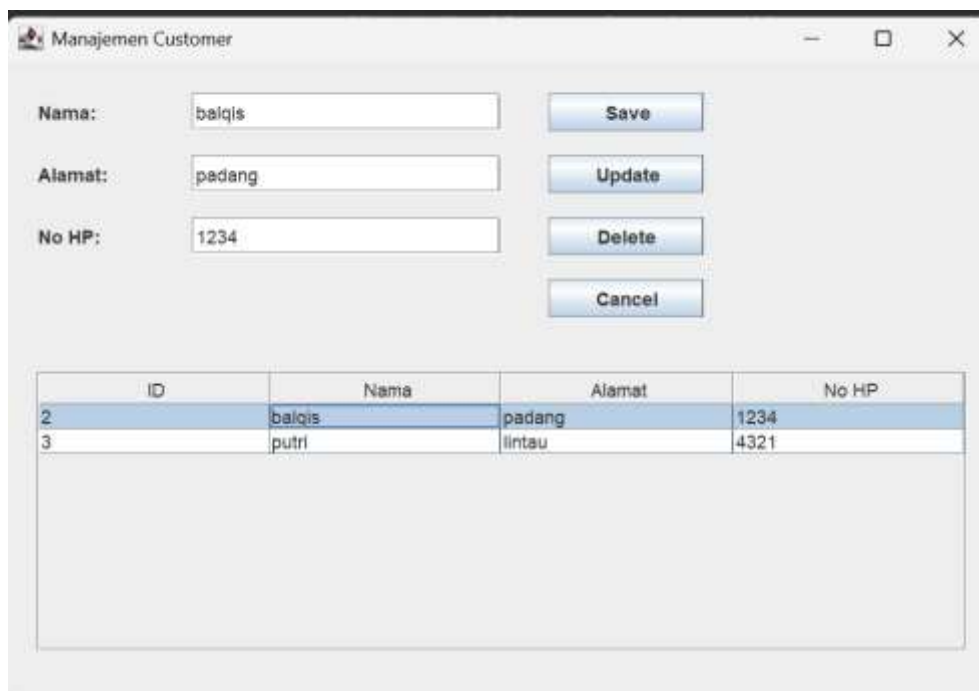
Kelas CustomerFrame adalah bagian GUI (Graphical User Interface) yang digunakan untuk mengelola data customer melalui tampilan form dan tabel. Di dalamnya terdapat beberapa komponen seperti text field untuk mengisi nama, alamat, dan nomor HP, serta tombol-tombol aksi (Save, Update, Delete, Cancel) yang masing-masing terhubung dengan fungsi CRUD.

Saat pengguna menekan tombol Save, data dari form akan dibuat sebagai objek customer lalu disimpan ke database melalui CustomerRepo. Tombol Update memungkinkan pengguna memperbarui data customer yang sudah ada, sedangkan tombol Delete digunakan untuk menghapus data berdasarkan ID yang dipilih di tabel. Tombol Cancel berfungsi mengosongkan form input

(reset).

Selain itu, CustomerFrame juga memiliki JTable yang menampilkan seluruh daftar customer menggunakan TableCustomer. Event klik pada tabel akan mengisi kembali data ke form agar bisa diedit atau dihapus. Dengan demikian, CustomerFrame menjadi antarmuka utama yang menghubungkan pengguna dengan fungsi CRUD customer secara interaktif dan mudah digunakan.

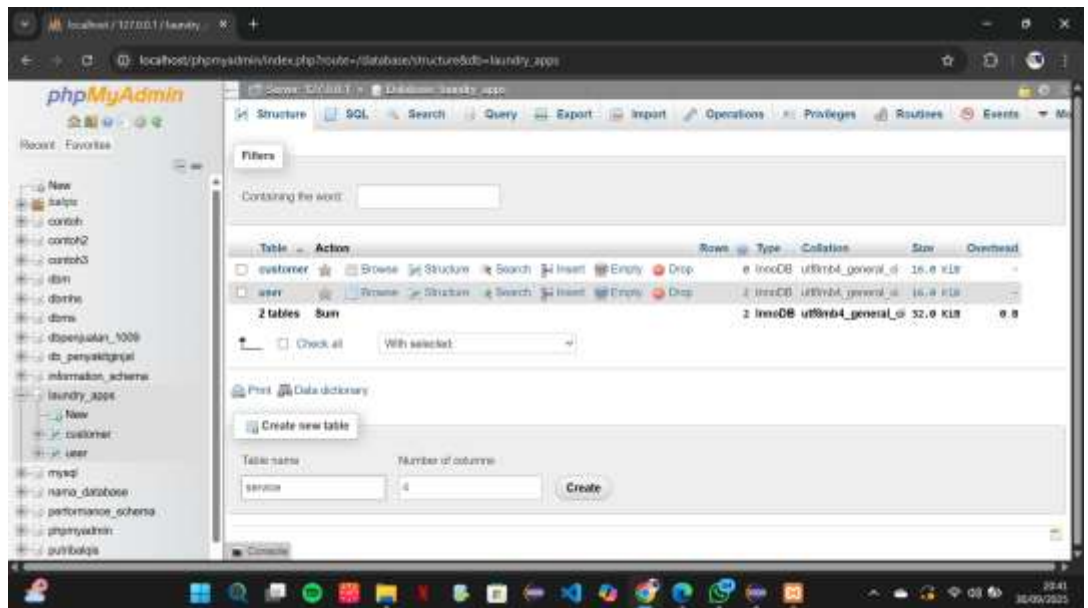
8. Maka outputnya akan seperti berikut



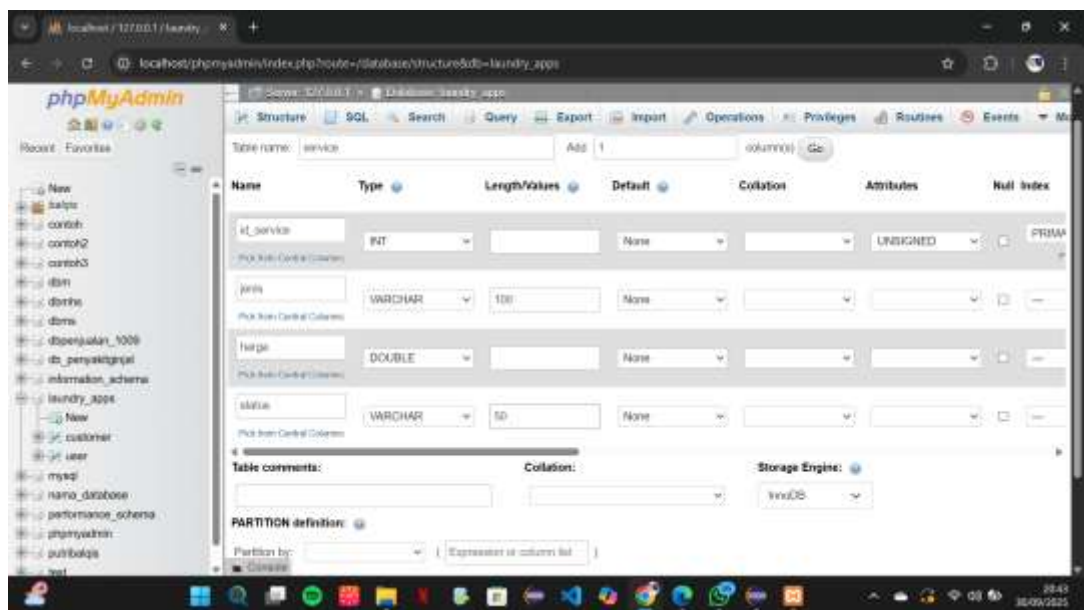
ID	Nama	Alamat	No HP
2	balqis	padang	1234
3	putri	lintau	4321

➤ **Fungsi CRUD Service**

1. Buka phpMyAdmin melalui XAMPP
2. Pilih database laundry_apps yang sudah dibuat sebelumnya, lalu buat table baru bernama “service”



3. Lalu isi dengan ketentuan seperti di bawah ini



4. selanjutnya kita akan membuat class “ServiceDAO” pada package DAO

```

1 package DAO;
2
3 import java.util.List;
4 import model.service;
5
6 public interface ServiceDAO {
7     void save(service s);
8     List<service> show();
9     void update(service s);
10    void delete(String id);
11 }
12

```

Interface ServiceDAO ini mendefinisikan kontrak untuk operasi CRUD pada entitas service.

- save(service s) → menyimpan data layanan baru ke database.
- show() → menampilkan daftar semua layanan dalam bentuk list.
- update(service s) → memperbarui data layanan yang sudah ada.
- delete(String id) → menghapus layanan berdasarkan ID.

Jadi, interface ini memastikan bahwa setiap class yang mengimplementasikannya (misalnya ServiceRepo) harus menyediakan implementasi lengkap untuk pengelolaan data layanan di database.

5. Lalu kita membuat class “ServiceRepo” dan membuat codingan seperti di bawah ini

```
1 package DAO;
2
3 import java.sql.*;
4 import java.util.*;
5 import javax.swing.JOptionPane;
6
7 import config.Database;
8 import model.service;
9
10 public class ServiceRepo implements ServiceDAO {
11     private Connection conn = Database.getConnection();
12
13     @Override
14     public void save(service s) {
15         String sql = "INSERT INTO service(jenis, harga, status) VALUES(?, ?, ?)";
16         try (PreparedStatement ps = conn.prepareStatement(sql)) {
17             ps.setString(1, s.getJenis());
18             ps.setDouble(2, s.getHarga());
19             ps.setString(3, s.getStatus());
20             ps.executeUpdate();
21             JOptionPane.showMessageDialog(null, "Service berhasil ditambahkan!");
22         } catch (SQLException e) {
23             JOptionPane.showMessageDialog(null, "Gagal tambah service: " + e.getMessage());
24         }
25     }
26
27     @Override
28     public List<service> show() {
29         List<service> list = new ArrayList<>();
30         String sql = "SELECT * FROM service";
31         try (Statement st = conn.createStatement(); ResultSet rs = st.executeQuery(sql)) {
32             while (rs.next()) {
33                 service s = new service();
34                 s.setId(rs.getString("id_service"));
35                 s.setJenis(rs.getString("jenis"));
36                 s.setHarga(rs.getDouble("harga"));
37                 list.add(s);
38             }
39         } catch (SQLException e) {
40             JOptionPane.showMessageDialog(null, "Gagal load service: " + e.getMessage());
41         }
42         return list;
43     }
44
45     @Override
46     public void update(service s) {
47         String sql = "UPDATE service SET jenis=?, harga=?, status=? WHERE id_service=?";
48         try (PreparedStatement ps = conn.prepareStatement(sql)) {
49             ps.setString(1, s.getJenis());
50             ps.setDouble(2, s.getHarga());
51             ps.setString(3, s.getStatus());
52             ps.setString(4, s.getId());
53             ps.executeUpdate();
54             JOptionPane.showMessageDialog(null, "Service berhasil diupdate!");
55         } catch (SQLException e) {
56             JOptionPane.showMessageDialog(null, "Gagal update service: " + e.getMessage());
57         }
58     }
59
60     @Override
61     public void delete(String id) {
62         String sql = "DELETE FROM service WHERE id_service=?";
63         try (PreparedStatement ps = conn.prepareStatement(sql)) {
64             ps.setString(1, id);
65             ps.executeUpdate();
66             JOptionPane.showMessageDialog(null, "Service berhasil dihapus!");
67         } catch (SQLException e) {
68             JOptionPane.showMessageDialog(null, "Gagal hapus service: " + e.getMessage());
69         }
70     }
71 }
72 }
```

Kelas ServiceRepo adalah implementasi dari interface ServiceDAO yang berfungsi untuk menghubungkan aplikasi dengan database pada tabel service. Di dalamnya terdapat kode SQL untuk menjalankan operasi CRUD:

- save(service s) → menjalankan query INSERT untuk menambahkan data layanan baru (jenis, harga, status).
- show() → menjalankan query SELECT untuk mengambil seluruh data layanan dari tabel service dan mengubahnya menjadi list objek service.
- update(service s) → menjalankan query UPDATE untuk memperbarui data layanan berdasarkan ID.
- delete(String id) → menjalankan query DELETE untuk menghapus layanan sesuai ID.

Singkatnya, ServiceRepo bertugas sebagai jembatan antara GUI (seperti ServiceFrame) dengan database, sehingga semua interaksi data layanan dapat dikelola secara terstruktur dan konsisten.

6. Selanjutnya pada package Table kita tambahkan class “TableService” seperti berikut

```
1 package Table;
2
3 import javax.swing.table.AbstractTableModel;
4 import java.util.List;
5 import model.service;
6
7 public class TableService extends AbstractTableModel {
8     private List<service> list;
9
10    public TableService(List<service> list) {
11        this.list = list;
12    }
13
14    @Override
15    public int getRowCount() {
16        return list.size();
17    }
18
19    @Override
20    public int getColumnCount() {
21        return 4; // id, jenis, harga, status
22    }
23
24    @Override
25    public String getColumnName(int column) {
26        switch (column) {
27            case 0: return "ID";
28            case 1: return "Jenis";
29            case 2: return "Harga";
30            case 3: return "Status";
31            default: return null;
32        }
33    }
34
35    @Override
36    public Object getValueAt(int rowIndex, int columnIndex) {
37        service s = list.get(rowIndex);
38        switch (columnIndex) {
39            case 0: return s.getId();
40            case 1: return s.getJenis();
41            case 2: return s.getHarga();
42            case 3: return s.getStatus();
43            default: return null;
44        }
45    }
46 }
```

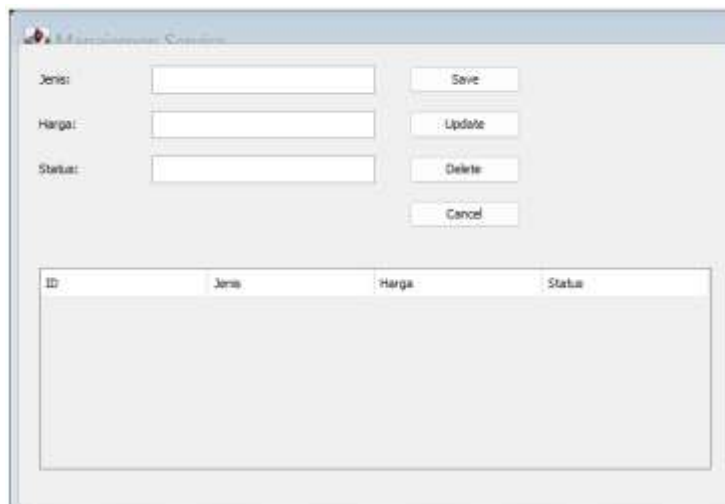

Kelas TableService dipakai untuk menampilkan data layanan (service) di dalam JTable pada GUI. Kelas ini biasanya meng-extend AbstractTableModel dan berisi daftar objek service.

Fungsinya adalah:

- Menentukan berapa jumlah baris dan kolom yang ditampilkan di tabel.
- Mengatur nama kolom (misalnya: ID, Jenis, Harga, Status).
- Mengambil nilai dari objek service untuk ditampilkan di setiap sel tabel.

Dengan begitu, TableService bertindak sebagai adapter antara data layanan dari database (List<service>) dengan tampilan tabel di aplikasi. Hasilnya, pengguna bisa melihat data layanan laundry dalam bentuk tabel yang rapi dan mudah dibaca.

7. Selanjutnya kita membuat class baru bernama “ServiceFrame” pada package ui untuk membuat desainnya



```

67
68 // Tombol Save
69 btnSave.addActionListener(new java.awt.event.ActionListener() {
70     public void actionPerformed(java.awt.event.ActionEvent e) {
71         service s = new service();
72         s.setJenis(txtJenis.getText());
73         s.setHarga(Double.parseDouble(txtHarga.getText()));
74         s.setStatus(txtStatus.getText());
75         repo.save(s);
76         loadTable();
77         reset();
78     }
79 });
80
81 // Tombol Update
82 btnUpdate.addActionListener(new java.awt.event.ActionListener() {
83     public void actionPerformed(java.awt.event.ActionEvent e) {
84         if (selectedId != null) {
85             service s = new service();
86             s.setId(selectedId);
87             s.setJenis(txtJenis.getText());
88             s.setHarga(Double.parseDouble(txtHarga.getText()));
89             s.setStatus(txtStatus.getText());
90             repo.update(s);
91             loadTable();
92             reset();
93         }
94     }
95 });
96
97 // Tombol Delete
98 btnDelete.addActionListener(new java.awt.event.ActionListener() {
99     public void actionPerformed(java.awt.event.ActionEvent e) {
100         if (selectedId != null) {

```



```

110     btnCancel.addActionListener(new java.awt.event.ActionListener() {
111         public void actionPerformed(java.awt.event.ActionEvent e) {
112             reset();
113         }
114     });
115
116     // Event klik tabel
117     table.getSelectionModel().addListSelectionListener(new javax.swing.event.ListSelectionListener() {
118         public void valueChanged(javax.swing.event.ListSelectionEvent e) {
119             int row = table.getSelectedRow();
120             if (row != -1) {
121                 selectedId = Integer.parseInt(table.getValueAt(row, 0).toString());
122                 txtJenis.setText(table.getValueAt(row, 1).toString());
123                 txtHarga.setText(table.getValueAt(row, 2).toString());
124                 txtStatus.setText(table.getValueAt(row, 3).toString());
125             }
126         }
127     });
128
129     private void loadTable() {
130         List<Service> list = repo.show();
131         TableService model = new TableService(list);
132         table.setModel(model);
133     }
134
135     private void reset() {
136         txtJenis.setText("");
137         txtHarga.setText("");
138         txtStatus.setText("");
139         selectedId = null;
140     }
141
142     public static void main(String[] args) {
143         SwingUtilities.invokeLater(new Runnable() {
144             public void run() {
145                 new ServiceFrame().setVisible(true);
146             }
147         });
148     }
149 }
150

```

Kelas ServiceFrame adalah GUI untuk mengelola data layanan. Di dalamnya ada form input (jenis, harga, status), tombol CRUD (Save, Update, Delete, Cancel), dan tabel (JTable) yang menampilkan data layanan melalui TableService. Frame ini memudahkan pengguna menambah, mengedit, menghapus, serta melihat layanan dengan menghubungkan langsung ke database lewat ServiceRepo.

8. Maka outputnya akan seperti berikut

ID	Jenis	Harga	Status
1	cuci	9000.0	aktif
2	cuci setrika	7000.0	aktif

v. KESIMPULAN

Pada praktikum ini berhasil dibuat aplikasi sederhana dengan konsep CRUD (Create, Read, Update, Delete) menggunakan pola DAO yang dihubungkan dengan GUI berbasis Java Swing. Setiap entitas seperti User, Customer, dan Service dapat dikelola melalui frame masing-masing dengan bantuan kelas Repository dan TableModel sebagai jembatan ke database. Dengan penerapan ini, data dapat diolah secara lebih terstruktur, mudah digunakan oleh pengguna, serta memperlihatkan bagaimana integrasi antara Java dan MySQL bekerja dalam pengembangan aplikasi.