

LAPORAN PRAKTIKUM 3
PEMROGRAMAN BERORIENTASI OBJEK



Oleh:

Nama : Arkan Ubaidillah Warman

NIM : 2411537001

Dosen Pengampu : Nurfiah,S.ST,M.Kom

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERITAS ANDALAS
PADANG

A. PENDAHULUAN

1. Latar Belakang

Perkembangan teknologi informasi menuntut adanya sistem yang mampu mengelola data secara efisien, cepat, dan terstruktur. Salah satu kebutuhan utama dalam pembangunan aplikasi adalah manajemen data pengguna (*user management*). Proses manajemen data umumnya mencakup penambahan data baru, menampilkan data, memperbarui data, serta menghapus data. Empat proses dasar ini dikenal dengan istilah CRUD (*Create, Read, Update, Delete*).

Dalam praktik pengembangan aplikasi, integrasi antara bahasa pemrograman dan sistem manajemen basis data (DBMS) merupakan hal penting agar data dapat dikelola secara efektif. MySQL sebagai salah satu DBMS populer, sering digunakan karena bersifat open source, ringan, dan mendukung integrasi dengan berbagai bahasa pemrograman, termasuk Java.

Eclipse sebagai *Integrated Development Environment (IDE)* mendukung pengembangan aplikasi Java dengan mudah melalui fitur debugging, integrasi library, serta dukungan konektivitas database. Oleh karena itu, membuat fungsi CRUD dengan database MySQL di Eclipse menjadi salah satu dasar penting untuk memahami proses pengelolaan data secara terstruktur dalam aplikasi berbasis Java.

2. Tujuan Praktikum

- Mampu membuat tabel user pada database MYSQL
- Mampu membuat koneksi java dengan database MYSQL
- Mampu membuat tampilan GUI CRUD user
- Mampu membuat dan mengimplementasikan interface
- Mampu membuat fungsi DAO dan mengimplementasikannya
- Mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek

3. Landasan teori

- **CRUD (Create, Read, Update, Delete)**

CRUD merupakan empat operasi dasar dalam manajemen data:

- a. **Create:** menambahkan data baru ke dalam database.
- b. **Read:** menampilkan atau membaca data yang sudah ada.
- c. **Update:** memperbarui data tertentu sesuai kebutuhan.

d. **Delete:** menghapus data dari database.

- **Database MySQL**

MySQL adalah sistem manajemen basis data relasional (*Relational Database Management System/RDBMS*) yang menggunakan bahasa SQL (*Structured Query Language*). Kelebihan MySQL adalah bersifat open source, mendukung skala besar, cepat, serta kompatibel dengan berbagai bahasa pemrograman.

- **Java Database Connectivity (JDBC)**

JDBC merupakan API (Application Programming Interface) dalam Java yang digunakan untuk menghubungkan program dengan database. Dengan JDBC, aplikasi Java dapat mengirim query SQL ke database MySQL dan memproses hasilnya.

- **Eclipse IDE**

Eclipse adalah IDE populer untuk pengembangan aplikasi Java. Fasilitas yang disediakan, seperti integrasi library JDBC dan fitur debugging, memudahkan pengembang dalam membuat aplikasi berbasis database.

B. LANGKAH-LANGKAH

1. Buka phpMyAdmin lalu buat database untuk tabel layanan dan pelanggan.

- Tabel pelanggan



The screenshot shows the phpMyAdmin interface for a database named 'laundry_apps'. The 'Tabel: pelanggan' (Table: customer) is selected. The 'Struktur tabel' (Table structure) tab is active, displaying the table's schema. The table has four columns: 'id' (int(11), primary key, auto-increment), 'nama' (varchar(128)), 'alamat' (varchar(256)), and 'no_telp' (varchar(20)). Each column has a checkbox for selection and a 'Tindakan' (Action) column with 'Ubah' (Edit), 'Hapus' (Delete), and 'Lainnya' (More) options.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
<input type="checkbox"/>	1 id	int(11)			Tidak	Tidak ada		AUTO_INCREMENT	Ubah Hapus Lainnya
<input type="checkbox"/>	2 nama	varchar(128)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/>	3 alamat	varchar(256)	utf8mb4_general_ci		Ya	NULL			Ubah Hapus Lainnya
<input type="checkbox"/>	4 no_telp	varchar(20)	utf8mb4_general_ci		Ya	NULL			Ubah Hapus Lainnya

- Tabel layanan

Server: 127.0.0.1 > Database: laundry_apps > Tabel: layanan

Jelajahi Struktur SQL Cari Tambahkan Ekspor Impor Hak Akses Operasi Pelacakan Trigger

Struktur tabel Tampilan hubungan

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
<input type="checkbox"/>	1 id	int(11)			Tidak	Tidak ada		AUTO_INCREMENT	Ubah Hapus Lainnya
<input type="checkbox"/>	2 namaLayanan	varchar(128)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/>	3 harga	decimal(10,2)			Tidak	Tidak ada			Ubah Hapus Lainnya

☐ Pilih Semua Dengan pilihan: Jelajahi Ubah Hapus Utama Unik Indeks Spasial Teks penuh

Remove from central columns

- Selanjutnya buka package model pada project laundry_apps lalu buat class baru dengan nama Layanan.

```

1 package model;
2
3 public class Layanan {
4     private int id;
5     private String namaLayanan;
6     private double harga;
7
8
9     public int getId() { return id; }
10    public void setId(int id) { this.id = id; }
11
12    public String getNamaLayanan() { return namaLayanan; }
13    public void setNamaLayanan(String namaLayanan) { this.namaLayanan = namaLayanan; }
14
15    public double getHarga() { return harga; }
16    public void setHarga(double harga) { this.harga = harga; }
17 }
18

```

- Pada package model buat class baru dengan nama Pelanggan.

```

1 package model;
2
3 public class Pelanggan {
4     private int id;
5     private String nama;
6     private String alamat;
7     private String noTelp;
8
9     // Getter & Setter
10    public int getId() { return id; }
11    public void setId(int id) { this.id = id; }
12
13    public String getNama() { return nama; }
14    public void setNama(String nama) { this.nama = nama; }
15
16    public String getAlamat() { return alamat; }
17    public void setAlamat(String alamat) { this.alamat = alamat; }
18
19    public String getNoTelp() { return noTelp; }
20    public void setNoTelp(String noTelp) { this.noTelp = noTelp; }
21 }
22

```

- Selanjutnya pada package DAO buat interface baru dengan nama LayananDAO.

```

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

```

5. Pada package DAO buat interface baru dengan nama PelangganDAO.

```

1 package DAO;
2
3 import java.util.List;
4 import model.Pelanggan;
5
6 public interface PelangganDAO {
7     void save(Pelanggan pelanggan);
8     List<Pelanggan> show();
9     void update(Pelanggan pelanggan);
10    void delete(int id);
11 }
12
13

```

6. Lalu pada package DAO buat class dengan nama LayananRepo

```

1 package DAO;
2
3 import config.Database;
4 import model.Layanan;
5 import java.sql.*;
6 import java.util.*;
7
8 public class LayananRepo implements LayananDAO {
9     Connection conn;
10
11     public LayananRepo() {
12         conn = Database.koneksi();
13     }
14
15     @Override
16     public void save(Layanan layanan) {
17         try {
18             String sql = "INSERT INTO layanan (nama_layanan, harga) VALUES (?, ?)";
19             PreparedStatement ps = conn.prepareStatement(sql);
20             ps.setString(1, layanan.getNamaLayanan());
21             ps.setDouble(2, layanan.getHarga());
22             ps.executeUpdate();
23         } catch (Exception e) {
24             e.printStackTrace();
25         }
26     }
27
28     @Override
29     public List<Layanan> show() {
30         List<Layanan> list = new ArrayList<>();
31         try {
32             String sql = "SELECT * FROM layanan";
33             Statement st = conn.createStatement();
34             ResultSet rs = st.executeQuery(sql);
35             while (rs.next()) {
36                 Layanan l = new Layanan();
37                 l.setId(rs.getInt("id"));
38                 l.setNamaLayanan(rs.getString("nama_layanan"));
39                 l.setHarga(rs.getDouble("harga"));
40                 list.add(l);
41             }
42         } catch (Exception e) {
43             e.printStackTrace();
44         }
45         return list;
46     }
47
48     @Override
49     public void update(Layanan layanan) {
50         try {
51             String sql = "UPDATE layanan SET nama_layanan=?, harga=? WHERE id=?";
52             PreparedStatement ps = conn.prepareStatement(sql);
53             ps.setString(1, layanan.getNamaLayanan());
54             ps.setDouble(2, layanan.getHarga());
55

```

```

55         ps.setInt(3, layanan.getId());
56         ps.executeUpdate();
57     } catch (Exception e) {
58         e.printStackTrace();
59     }
60 }
61
62 @Override
63 public void delete(int id) {
64     try {
65         String sql = "DELETE FROM layanan WHERE id=?";
66         PreparedStatement ps = conn.prepareStatement(sql);
67         ps.setInt(1, id);
68         ps.executeUpdate();
69     } catch (Exception e) {
70         e.printStackTrace();
71     }
72 }
73 }

```

7. Lalu pada package DAO buat class baru untuk pelanggan dengan nama PelangganRepo.

```

1 package DAO;
2
3 import config.Database;
4 import model.Pelanggan;
5 import java.sql.*;
6 import java.util.*;
7
8 public class PelangganRepo implements PelangganDAO {
9     Connection conn;
10
11     public PelangganRepo() {
12         conn = Database.koneksi();
13     }
14
15     @Override
16     public void save(Pelanggan pelanggan) {
17         try {
18             String sql = "INSERT INTO pelanggan (nama, alamat, no_telp) VALUES (?, ?, ?)";
19             PreparedStatement ps = conn.prepareStatement(sql);
20             ps.setString(1, pelanggan.getNama());
21             ps.setString(2, pelanggan.getAlamat());
22             ps.setString(3, pelanggan.getNoTelp());
23             ps.executeUpdate();
24         } catch (Exception e) {
25             e.printStackTrace();
26         }
27     }
28
29     @Override
30     public List<Pelanggan> show() {
31         List<Pelanggan> list = new ArrayList<>();
32         try {
33             String sql = "SELECT * FROM pelanggan";
34             Statement st = conn.createStatement();
35             ResultSet rs = st.executeQuery(sql);
36             while (rs.next()) {
37                 Pelanggan p = new Pelanggan();
38                 p.setId(rs.getInt("id"));
39                 p.setNama(rs.getString("nama"));
40                 p.setAlamat(rs.getString("alamat"));
41                 p.setNoTelp(rs.getString("no_telp"));
42                 list.add(p);
43             }
44         } catch (Exception e) {
45             e.printStackTrace();
46         }
47         return list;
48     }
49
50     @Override
51     public void update(Pelanggan pelanggan) {
52         try {
53             String sql = "UPDATE pelanggan SET nama=?, alamat=?, no_telp=? WHERE id=?";
54             PreparedStatement ps = conn.prepareStatement(sql);
55             ps.setString(1, pelanggan.getNama());
56             ps.setString(2, pelanggan.getAlamat());
57             ps.setString(3, pelanggan.getNoTelp());
58             ps.setInt(4, pelanggan.getId());
59             ps.executeUpdate();
60         } catch (Exception e) {
61             e.printStackTrace();
62         }
63     }
64
65     @Override
66     public void delete(int id) {
67         try {
68             String sql = "DELETE FROM pelanggan WHERE id=?";
69             PreparedStatement ps = conn.prepareStatement(sql);
70             ps.setInt(1, id);
71             ps.executeUpdate();
72         } catch (Exception e) {
73             e.printStackTrace();
74         }
75     }
76 }

```

8. Selanjutnya buat class TableLayanan pada package table.

```

1 package table;
2
3 import model.Layanan;
4 import javax.swing.table.AbstractTableModel;
5 import java.util.List;
6
7 public class TableLayanan extends AbstractTableModel {
8     private List<Layanan> list;
9     private String[] column = {"ID", "Nama Layanan", "Harga"};
10
11     public TableLayanan(List<Layanan> list) {
12         this.list = list;
13     }
14
15     @Override
16     public int getRowCount() { return list.size(); }
17
18     @Override
19     public int getColumnCount() { return column.length; }
20
21     @Override
22     public String getColumnName(int col) { return column[col]; }
23
24     @Override
25     public Object getValueAt(int row, int col) {
26         Layanan l = list.get(row);
27         switch (col) {
28             case 0: return l.getId();
29             case 1: return l.getNamaLayanan();
30             case 2: return l.getHarga();
31             default: return null;
32         }
33     }
34 }

```

9. Selanjutnya pada package table juga, buat class untuk TablePelanggan.

```

1 package table;
2
3 import model.Pelanggan;
4 import javax.swing.table.AbstractTableModel;
5 import java.util.List;
6
7 public class TablePelanggan extends AbstractTableModel {
8     private List<Pelanggan> list;
9     private String[] column = {"ID", "Nama", "Alamat", "No Telp"};
10
11     public TablePelanggan(List<Pelanggan> list) {
12         this.list = list;
13     }
14
15     @Override
16     public int getRowCount() { return list.size(); }
17
18     @Override
19     public int getColumnCount() { return column.length; }
20
21     @Override
22     public String getColumnName(int col) { return column[col]; }
23
24     @Override
25     public Object getValueAt(int row, int col) {
26         Pelanggan p = list.get(row);
27         switch (col) {
28             case 0: return p.getId();
29             case 1: return p.getNama();
30             case 2: return p.getAlamat();
31             case 3: return p.getNoTelp();
32             default: return null;
33         }
34     }
35 }

```

10.Selanjutnya Untuk tampilan GUI nya silakan menggunakan ini.

11.Buat pada package ui JFrame baru dengan nama LayananFrame

ID	Nama Layanan	Harga
1	tidur	10000.0

12. Pada tombol save buat add action listener lalu isi dengan kode berikut

```
btnSave.addActionListener(e -> {
    Layanan l = new Layanan();
    l.setNamaLayanan(txtNamaLayanan.getText());
    l.setHarga(Double.parseDouble(txtHarga.getText()));
    layananDAO.save(l);
    reset();
    loadTable();
});
```

13. Pada update kodenya sebagai berikut

```
btnUpdate.addActionListener(e -> {
    if (selectedId != -1) {
        Layanan l = new Layanan();
        l.setId(selectedId);
        l.setNamaLayanan(txtNamaLayanan.getText());
        l.setHarga(Double.parseDouble(txtHarga.getText()));
        layananDAO.update(l);
        reset();
        loadTable();
    }
});
```

14. Dan pada delete kode nya sebagai berikut

```
btnDelete.addActionListener(e -> {
    if (selectedId != -1) {
        layananDAO.delete(selectedId);
        reset();
        loadTable();
    }
});
```


15. Lalu pada tabel click kanan, lalu pilih mouseClicked

16. Lalu isi dengan kode berikut

```
tableLayanan.addMouseListener(new MouseAdapter() {  
    public void mouseClicked(MouseEvent e) {  
        int row = tableLayanan.getSelectedRow();  
        selectedId = (int) tableLayanan.getValueAt(row, 0);  
        txtNamaLayanan.setText(tableLayanan.getValueAt(row, 1).toString());  
        txtHarga.setText(tableLayanan.getValueAt(row, 2).toString());  
    }  
});
```

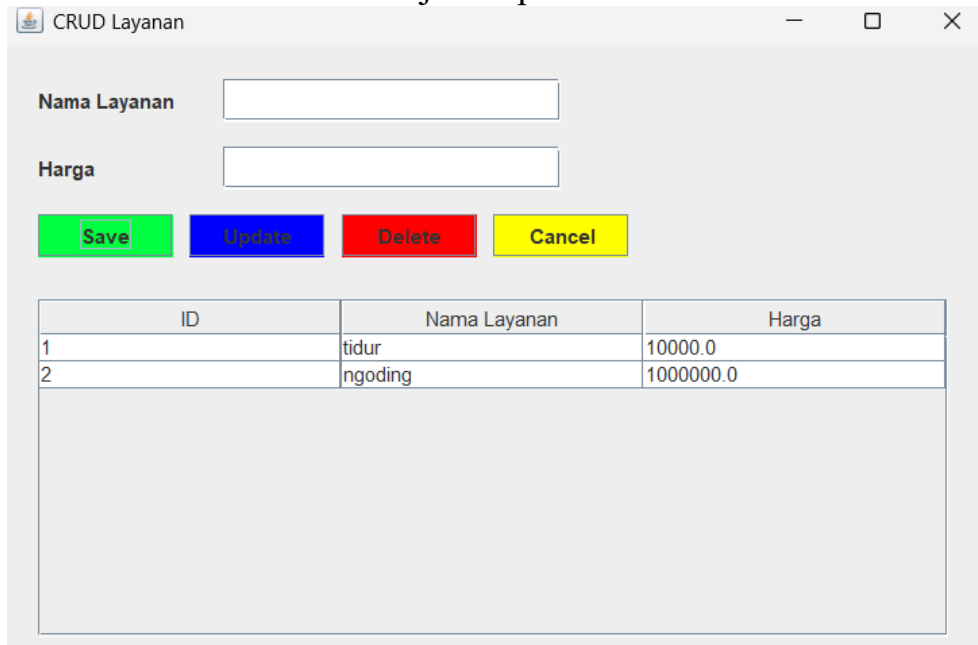
17. Selanjutnya buat method loadTable

```
private void loadTable() {  
    List<Layanan> list = layananDAO.show();  
    TableLayanan tbl = new TableLayanan(list);  
    tableLayanan.setModel(tbl);  
}
```

18. Lalu method reset.

```
private void reset() {  
    txtNamaLayanan.setText("");  
    txtHarga.setText("");  
    selectedId = -1;  
}
```

19. Setelah dirun kan akan menjadi sepeerti ini.



ID	Nama Layanan	Harga
1	tidur	10000.0
2	ngoding	1000000.0

20. Untuk melihat apakah sudah terhubung dengan sql bisa dilihat pada phpMyAdmin dibagian tabel layanan.

☐ Tampilkan semua

Jumlah baris: 25

Saring baris:

Sort b

Extra options

←

T

→

▼

id

namalayanan

harga

☐

Ubah

Salin

Hapus

1

tidur

10000.00

☐

Ubah

Salin

Hapus

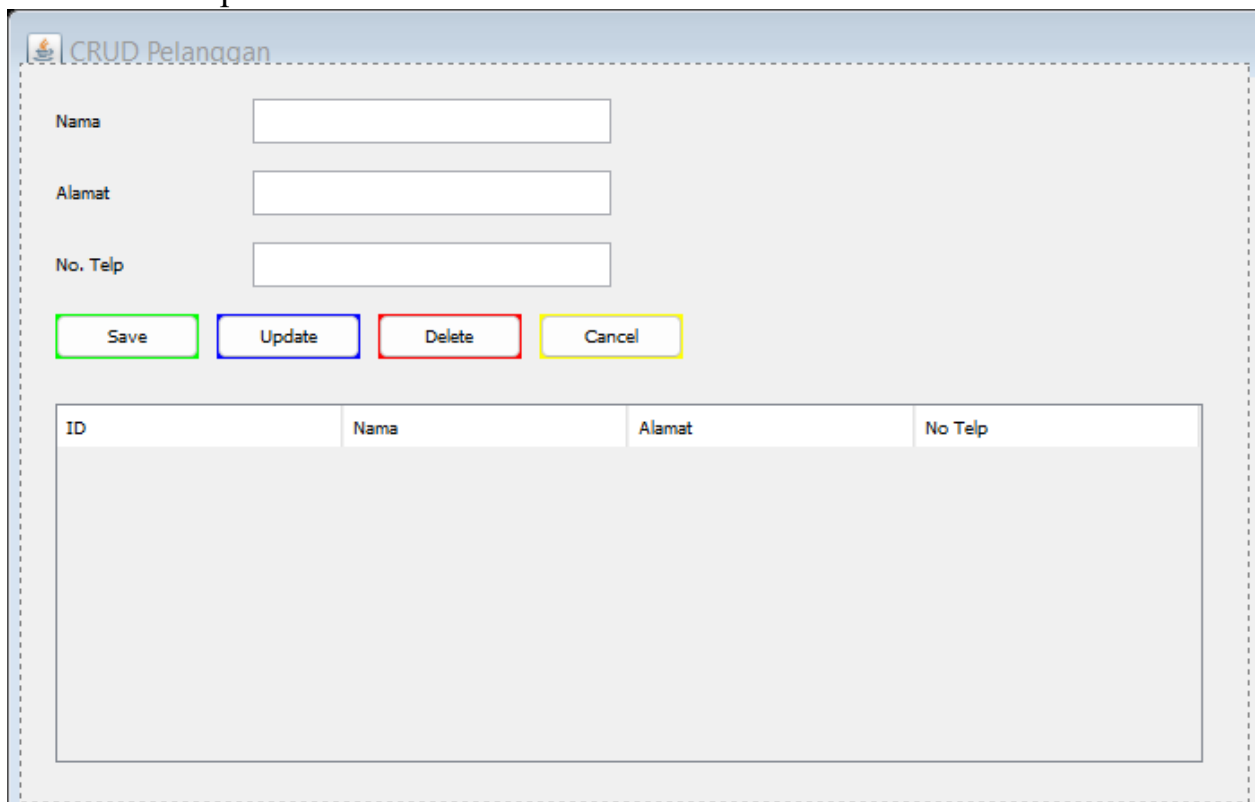
2

ngoding

1000000.00

21. Selanjutnya untuk pelanggan buat JFrame baru pada package ui dengan nama PelangganFrame.

22. Lalu desain seperti ini.



The screenshot shows a Java Swing window titled "CRUD Pelanggan". It contains three text input fields labeled "Nama", "Alamat", and "No. Telp". Below these fields are four buttons: "Save" (green border), "Update" (blue border), "Delete" (red border), and "Cancel" (yellow border). At the bottom of the window is a table with the following structure:

ID	Nama	Alamat	No Telp
----	------	--------	---------

23. Pada button save click kanan lalu add action listener lalu isi kodenya seperti berikut

```

btnSave.addActionListener(e -> {
    Pelanggan p = new Pelanggan();
    p.setNama(txtNama.getText());
    p.setAlamat(txtAlamat.getText());
    p.setNoTelp(txtNoTelp.getText());
    pelangganDAO.save(p);
    reset();
    loadTable();
});

```

24.Selanjutnya pada delete isi kode seperti berikut.

```

btnDelete.addActionListener(e -> {
    if (selectedId != -1) {
        pelangganDAO.delete(selectedId);
        reset();
        loadTable();
    }
});

```

25.Lalu pada cancel cukup isi dengan e -> reset()

26.Pada tabel click kanan lalu pilih mouse> addMouseClicked.

27.Lalu isi dengan kode berikut

```

tablePelanggan.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        int row = tablePelanggan.getSelectedRow();
        selectedId = (int) tablePelanggan.getValueAt(row, 0);
        txtNama.setText(tablePelanggan.getValueAt(row, 1).toString());
        txtAlamat.setText(tablePelanggan.getValueAt(row, 2).toString());
        txtNoTelp.setText(tablePelanggan.getValueAt(row, 3).toString());
    }
});

```

28.Buat method loadTable

```

private void loadTable() {
    List<Pelanggan> list = pelangganDAO.show();
    TablePelanggan tbl = new TablePelanggan(list);
    tablePelanggan.setModel(tbl);
}

```

29.Lalu method reset

```

private void reset() {
    txtNama.setText("");
    txtAlamat.setText("");
    txtNoTelp.setText("");
    selectedId = -1;
}

```

30. Lalu hasilnya akan seperti ini.

CRUD Pelanggan

Nama

Alamat

No. Telp

ID	Nama	Alamat	No Telp
1	Arkan	Jl.ranah	087842184809

31. Pada database sql sudah terhubung bisa dilihat di phpMyAdmin

☐ Tampilkan semua | Jumlah baris: 25 ▼ | Saring baris:

Extra options

← T → ▼ id nama alamat no_telp

☐  Ubah  Salin  Hapus 1 Arkan Jl.ranah 087842184809

↑ ☐ Pilih Semua Dengan pilihan:  Ubah  Salin  Hapus  Ekspor

☐ Tampilkan semua | Jumlah baris: 25 ▼ | Saring baris:

C. KESIMPULAN

Setiap fungsi CRUD diimplementasikan secara fungsional:

- a. Create: Menambahkan data user baru ke tabel user dengan memanfaatkan PreparedStatement.
- b. Read: Menampilkan seluruh data user dari database ke dalam tabel GUI menggunakan ResultSet.
- c. Update: Memperbarui informasi user berdasarkan ID yang dipilih.
- d. Delete: Menghapus data user secara spesifik dengan parameter ID.

Koneksi database berhasil dilakukan melalui driver MySQL Connector/J.