LAPORAN PRAKTIKUM PEKAN 3 PEMOGRAMAN BERORIENTASI OBJEK



NAMA : FIKHRI HANIF

NIM : 2311533007

DOSEN PENGAMPU :

Nurfiah, S.ST, M.Kom.

DEPARTEMEN INFORMATIKA FAKULTAS TEKNOLOGI INFORMASI UNIVERSITAS ANDALAS

PADANG

1. **Tujuan** 
   1. Mahasiswa mampu membuat table user pada database MySQL
   2. Mahasiswa mampu membuat koneksi Java dengan database MySQL
   3. Mahasiswa mampu membuat tampilan GUI CRUD user
   4. Mahasiswa mampu membuat dan mengimplementasikan interface
   5. Mahasiswa mampu membuat fungsi DAO (Data Access Object) dan

mengimplementasikannya.

* 1. Mahasiswa mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek

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

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

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

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

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

Fungsi utama MySQL Connector/J:

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

1. Data Access Object (DAO)

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

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

Dalam pemrograman Java, interface digunakan untuk mendefinisikan kumpulan metode abstrak yang harus diimplementasikan oleh kelas yang menggunakan interface tersebut. Interface bermanfaat dalam penerapan pola desain DAO karena:

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

1. Operasi CRUD

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

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

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

1. **Langkah Kerja**
   1. Costumer
   2. Membuat Costumer Model

Buat class Costumer pada package model dengan berisikan codingan berikut:

package model;

public class Costumer {

String id, nama, alamat, 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) {

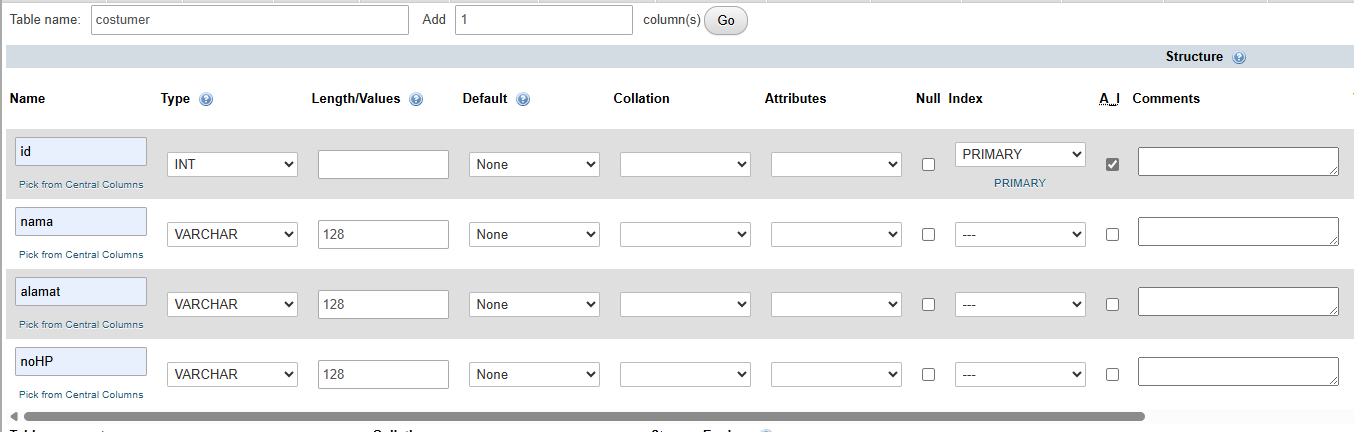
this.nohp = nohp;

}

}

* 1. Membuat Database pada XAMPP

Pada XAMPP, buat database baru dengan nama costumer, yang berisikan seperti gambar berikut :



* 1. Membuat Costumer DAO

Buat class interface baru pada package DAO dengan nama CostumerDAO dengan berisikan codingan berikut :

package DAO;

import java.util.List;

import model.Costumer;

public interface CostumerDAO {

void save(Costumer costumer);

public List<Costumer> show();

public void delete (String id);

public void update (Costumer costumer);

}

* 1. Membuat Costumer Repo

Buat class CostmerRepo pada package DAO dengan berisikan codingan berikut :

package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import confg.Database;

import model.Costumer;

public class CostumerRepo implements CostumerDAO{

private Connection connection;

final String insert = "INSERT INTO costumer (nama, alamat, nohp) VALUES (?,?,?);";

final String select = "SELECT \* FROM costumer;" ;

final String delete = "DELETE FROM costumer WHERE id = ?;";

final String update = "UPDATE costumer SET nama=?, alamat=?, nohp=? WHERE id=?;";

public CostumerRepo() {

connection = Database.*koneksi*();

}

*@Override*

public void save(Costumer costumer) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(insert);

st.setString(1, costumer.getNama());

st.setString(2, costumer.getAlamat());

st.setString(3, costumer.getNohp());

st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}finally {

try {

st.close();

}catch(SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public List<Costumer> show() {

// **TODO** Auto-generated method stub

List<Costumer> ls = null;

try {

ls = new ArrayList<Costumer>();

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery(select);

while(rs.next()) {

Costumer costumer = new Costumer();

costumer.setId(rs.getString("id"));

costumer.setNama(rs.getString("nama"));

costumer.setAlamat(rs.getString("alamat"));

costumer.setNohp(rs.getString("nohp"));

ls.add(costumer);

}

}catch(SQLException e) {

Logger.*getLogger*(CostumerDAO.class.getName()).log(Level.***SEVERE***, null, e);

}

return ls;

}

*@Override*

public void delete(String id) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(delete);

st.setString(1, id);

st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}finally {

try {

st.close();

}catch(SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public void update(Costumer costumer) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(update);

st.setString(1, costumer.getNama());

st.setString(2, costumer.getAlamat());

st.setString(3, costumer.getNohp());

st.setString(4, costumer.getId());

st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}finally {

try {

st.close();

}catch(SQLException e){

e.printStackTrace();

}

}

}

}

* 1. Membuat Table Costumer

Buat class TableCostumer pada Package Table dengan berisikan codingan berikut :

package table;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import model.Costumer;

public class TableCostumer extends AbstractTableModel {

List<Costumer> ls;

private String[] columnNames = {"ID", "Nama", "Alamat", "Nohp"};

public TableCostumer(List<Costumer> ls) {

this.ls = ls;

}

*@Override*

public int getRowCount() {

return ls.size();

}

*@Override*

public int getColumnCount() {

return 4;

}

*@Override*

public String getColumnName(int colomn) {

return columnNames[colomn];

}

*@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).getNohp();

default:

return null;

}

}

}

* 1. Membuat JFrame Costumer

Buat JFrame dengan nama CostumerFrame pada Package ui dengan berisikan codingan berikut :

package ui;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import DAO.CostumerRepo;

import DAO.UserRepo;

import model.Costumer;

import model.User;

import table.TableCostumer;

import table.TableUser;

import javax.swing.JScrollPane;

import java.awt.FlowLayout;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import java.util.List;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JTable;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class CostumerFrame extends JFrame {

private static final long ***serialVersionUID*** = 1L;

private JPanel contentPane;

private JTextField txtNama;

private JTextField txtAlamat;

private JTextField txtNohp;

private JTable tableCostumer;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.*invokeLater*(new Runnable() {

public void run() {

try {

CostumerFrame frame = new CostumerFrame();

frame.setVisible(true);

frame.loadTable();

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

CostumerRepo cst = new CostumerRepo();

List<Costumer> ls;

public String id;

/\*\*

\* Create the frame.

\*/

public CostumerFrame() {

setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

setBounds(100, 100, 586, 600);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(10, 11, 541, 260);

contentPane.add(panel);

panel.setLayout(null);

JLabel lblNama = new JLabel("Nama Pelanggan");

lblNama.setBounds(26, 25, 121, 25);

lblNama.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

panel.add(lblNama);

txtNama = new JTextField();

txtNama.setBounds(165, 31, 326, 19);

panel.add(txtNama);

txtNama.setColumns(10);

JLabel lblAlamat = new JLabel("Alamat");

lblAlamat.setBounds(89, 74, 58, 25);

lblAlamat.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

panel.add(lblAlamat);

txtAlamat = new JTextField();

txtAlamat.setBounds(165, 80, 326, 19);

txtAlamat.setColumns(10);

panel.add(txtAlamat);

JLabel lblNohp = new JLabel("No. HP");

lblNohp.setBounds(89, 122, 51, 25);

lblNohp.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

panel.add(lblNohp);

txtNohp = new JTextField();

txtNohp.setBounds(165, 128, 326, 19);

txtNohp.setColumns(10);

panel.add(txtNohp);

JButton btnSave = new JButton("Save");

btnSave.setBounds(89, 193, 85, 40);

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Costumer costumer = new Costumer();

costumer.setNama(txtNama.getText());

costumer.setAlamat(txtAlamat.getText());

costumer.setNohp(txtNohp.getText());

cst.save(costumer);

reset();

loadTable();

}

});

btnSave.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

panel.add(btnSave);

JButton btnUpdate = new JButton("Update");

btnUpdate.setBounds(184, 193, 85, 40);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(id != null) {

Costumer costumer = new Costumer();

costumer.setNama(txtNama.getText());

costumer.setAlamat(txtAlamat.getText());

costumer.setNohp(txtNohp.getText());

costumer.setId(id);

cst.update(costumer);

reset();

loadTable();

}

else {

JOptionPane.*showMessageDialog*(null, "Silahkan pilih data yang akan dihapus");

}

}

});

btnUpdate.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

panel.add(btnUpdate);

JButton btnDelete = new JButton("Delete");

btnDelete.setBounds(279, 193, 85, 40);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(id != null) {

cst.delete(id);

reset();

loadTable();

}else {

JOptionPane.*showMessageDialog*(null, "Silahkan pilih data yang akan dihapus");

}

}

});

btnDelete.setFont(new Font("HP Simplified Jpan Light", Font.***PLAIN***, 16));

panel.add(btnDelete);

JButton btnCancel = new JButton("Cancel");

btnCancel.setBounds(374, 193, 85, 40);

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

MainFrame main = new MainFrame();

main.setVisible(true);

dispose();

}

});

btnCancel.setFont(new Font("HP Simplified Jpan Light", Font.***PLAIN***, 16));

panel.add(btnCancel);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 280, 541, 273);

contentPane.add(scrollPane);

tableCostumer = new JTable();

tableCostumer.addMouseListener(new MouseAdapter() {

*@Override*

public void mouseClicked(MouseEvent e) {

id = tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 0).toString();

txtNama.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 1).toString());

txtAlamat.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 2).toString());

txtNohp.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 3).toString());

}

});

scrollPane.setViewportView(tableCostumer);

}

public void loadTable() {

ls = cst.show();

TableCostumer tc = new TableCostumer(ls);

tableCostumer.setModel(tc);

tableCostumer.getTableHeader().setVisible(true);

}

public void reset() {

txtNama.setText("");

txtAlamat.setText("");

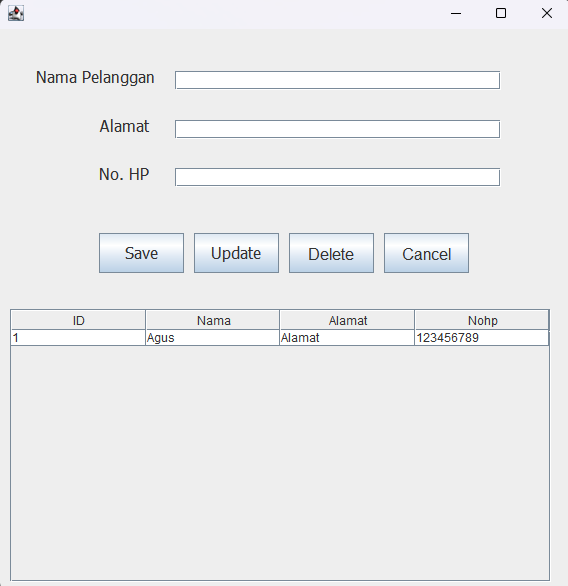
txtNohp.setText("");

}

}

* 1. Hasil

Setelah semuanya ditambahkan, kemudian jalankan CostumerFrame, maka data yang sudah kita inputkan akan masuk ke database localhost pad XAMPP dengan tampilan seperti berikut :



* 1. Service
  2. Membuat Service Model

Buat class Service pada package model dengan berisikan codingan berikut:

package model;

public class Service {

String id, jenis, status;

int harga;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getJenis() {

return jenis;

}

public void setJenis(String jenis) {

this.jenis = jenis;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public int getHarga() {

return harga;

}

public void setHarga(int harga) {

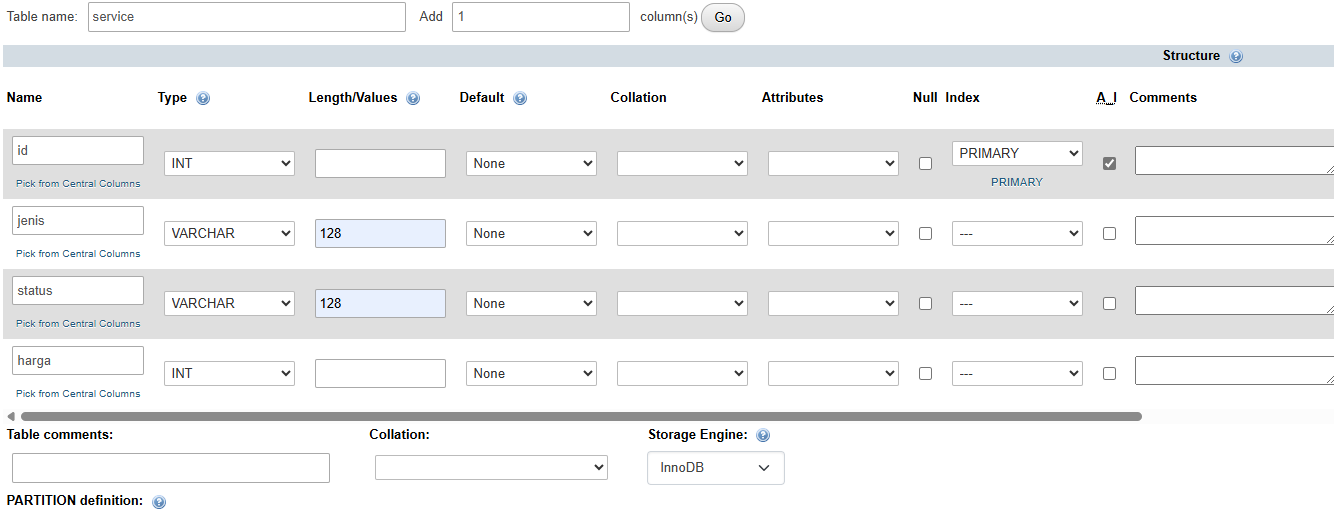
this.harga = harga;

}

}

* 1. Membuat Database pada XAMPP

Pada XAMPP, buat database baru dengan nama costumer, yang berisikan seperti gambar berikut :



* 1. Membuaat Service DAO

Buat class interface baru pada package DAO dengan nama ServiceDAO dengan berisikan codingan berikut :

package DAO;

import java.util.List;

import model.Service;

public interface ServiceDAO {

void save(Service service);

public List<Service> show();

public void delete (String id);

public void update (Service service);

}

* 1. Membuat Service Repo

Buat class ServiceRepo pada package DAO dengan berisikan codingan berikut :

package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import confg.Database;

import model.Service;

public class ServiceRepo implements ServiceDAO{

private Connection connection;

final String insert = "INSERT INTO service (jenis, status, harga) VALUES (?,?,?);";

final String select = "SELECT \* FROM service;" ;

final String delete = "DELETE FROM service WHERE id = ?;";

final String update = "UPDATE service SET jenis = ?, status = ?, harga = ? WHERE id = ?;";

public ServiceRepo() {

connection = Database.*koneksi*();

}

*@Override*

public void save(Service service) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(insert);

st.setString(1, service.getJenis());

st.setString(2, service.getStatus());

st.setInt(3, service.getHarga());

st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}finally {

try {

st.close();

}catch(SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public List<Service> show() {

// **TODO** Auto-generated method stub

List<Service> ls = null;

try {

ls = new ArrayList<Service>();

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery(select);

while(rs.next()) {

Service service = new Service();

service.setId(rs.getString("id"));

service.setJenis(rs.getString("jenis"));

service.setStatus(rs.getString("status"));

service.setHarga(rs.getInt("harga"));

ls.add(service);

}

}catch(SQLException e) {

Logger.*getLogger*(ServiceDAO.class.getName()).log(Level.***SEVERE***, null, e);

}

return ls;

}

*@Override*

public void delete(String id) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(delete);

st.setString(1, id);

st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}finally {

try {

st.close();

}catch(SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public void update(Service service) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(update);

st.setString(1, service.getJenis());

st.setString(2, service.getStatus());

st.setInt(3, service.getHarga());

st.setString(4, service.getId());

st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}finally {

try {

st.close();

}catch(SQLException e){

e.printStackTrace();

}

}

}

}

* 1. Membuat Table Service

Buat class TableService pada Package Table dengan berisikan codingan berikut :

package table;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import model.Service;

public class TableService extends AbstractTableModel{

List<Service> ls;

private String[] columnNames = {"ID", "Jenis", "Status", "Harga"};

public TableService(List<Service> ls) {

this.ls = ls;

}

*@Override*

public int getRowCount() {

// **TODO** Auto-generated method stub

return ls.size();

}

*@Override*

public int getColumnCount() {

// **TODO** Auto-generated method stub

return 4;

}

*@Override*

public Object getValueAt(int rowIndex, int columnIndex) {

// **TODO** Auto-generated method stub

switch (columnIndex) {

case 0:

return ls.get(rowIndex).getId();

case 1:

return ls.get(rowIndex).getJenis();

case 2:

return ls.get(rowIndex).getStatus();

case 3:

return ls.get(rowIndex).getHarga();

default:

return null;

}

}

}

* 1. Membuat JFrame Service

Buat JFrame dengan nama CostumerFrame pada Package ui dengan berisikan codingan berikut :

package ui;

import java.awt.EventQueue;

import java.awt.Font;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import DAO.ServiceRepo;

import DAO.UserRepo;

import model.Service;

import model.User;

import table.TableService;

import table.TableUser;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JTable;

import javax.swing.JScrollPane;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class ServiceFrame extends JFrame {

private static final long ***serialVersionUID*** = 1L;

private JPanel contentPane;

private JTextField txtJenis;

private JTextField txtStatus;

private JTextField txtHarga;

private JTable tableService;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.*invokeLater*(new Runnable() {

public void run() {

try {

ServiceFrame frame = new ServiceFrame();

frame.setVisible(true);

frame.loadTable();

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public void reset() {

txtJenis.setText("");

txtStatus.setText("");

txtHarga.setText("");

}

ServiceRepo srv = new ServiceRepo();

List<Service> ls;

public String id;

public void loadTable() {

ls = srv.show();

TableService ts = new TableService(ls);

tableService.setModel(ts);

tableService.getTableHeader().setVisible(true);

}

public ServiceFrame() {

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 569, 600);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(10, 10, 530, 260);

contentPane.add(panel);

panel.setLayout(null);

JLabel lblJenis = new JLabel("Jenis");

lblJenis.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

lblJenis.setBounds(70, 25, 45, 25);

panel.add(lblJenis);

txtJenis = new JTextField();

txtJenis.setBounds(165, 31, 303, 19);

panel.add(txtJenis);

txtJenis.setColumns(10);

JLabel lblStatus = new JLabel("Status");

lblStatus.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

lblStatus.setBounds(70, 75, 76, 25);

panel.add(lblStatus);

txtStatus = new JTextField();

txtStatus.setColumns(10);

txtStatus.setBounds(165, 81, 303, 19);

panel.add(txtStatus);

JLabel lblHarga = new JLabel("Harga");

lblHarga.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

lblHarga.setBounds(70, 122, 76, 25);

panel.add(lblHarga);

txtHarga = new JTextField();

txtHarga.setColumns(10);

txtHarga.setBounds(165, 128, 303, 19);

panel.add(txtHarga);

JButton btnSave = new JButton("Save");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Service service = new Service();

service.setJenis(txtJenis.getText());

service.setStatus(txtStatus.getText());

service.setHarga(Integer.*parseInt*(txtHarga.getText()));

srv.save(service);

reset();

loadTable();

}

});

btnSave.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

btnSave.setBounds(92, 193, 85, 40);

panel.add(btnSave);

JButton btnUpdate = new JButton("Update");

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(id != null) {

Service service = new Service();

service.setJenis(txtJenis.getText());

service.setStatus(txtStatus.getText());

service.setHarga(Integer.*parseInt*(txtHarga.getText()));

service.setId(id);

srv.update(service);

reset();

loadTable();

}

else {

JOptionPane.*showMessageDialog*(null, "Silahkan pilih data yang akan dihapus");

}

}

});

btnUpdate.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

btnUpdate.setBounds(187, 193, 85, 40);

panel.add(btnUpdate);

JButton btnDelete = new JButton("Delete");

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(id != null) {

srv.delete(id);

reset();

loadTable();

}else {

JOptionPane.*showMessageDialog*(null, "Silahkan pilih data yang akan dihapus");

}

}

});

btnDelete.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

btnDelete.setBounds(288, 193, 85, 40);

panel.add(btnDelete);

JButton btnCancel = new JButton("Cancel");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

MainFrame main = new MainFrame();

main.setVisible(true);

dispose();

}

});

btnCancel.setFont(new Font("Tahoma", Font.***PLAIN***, 16));

btnCancel.setBounds(383, 193, 85, 40);

panel.add(btnCancel);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 280, 530, 273);

contentPane.add(scrollPane);

tableService = new JTable();

tableService.addMouseListener(new MouseAdapter() {

*@Override*

public void mouseClicked(MouseEvent e) {

id = tableService.getValueAt(tableService.getSelectedRow(), 0).toString();

txtJenis.setText(tableService.getValueAt(tableService.getSelectedRow(), 1).toString());

txtStatus.setText(tableService.getValueAt(tableService.getSelectedRow(), 2).toString());

txtHarga.setText(tableService.getValueAt(tableService.getSelectedRow(), 3).toString());

}

});

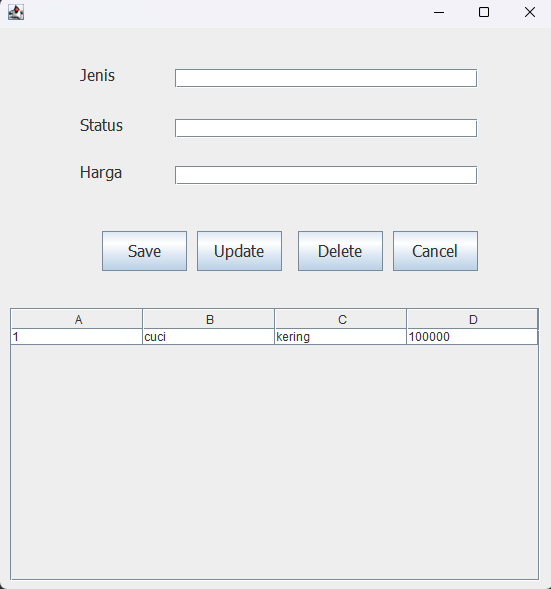
scrollPane.setViewportView(tableService);

}

}

* 1. Hasil

Setelah semuanya ditambahkan, kemudian jalankan ServiceFrame, maka data yang sudah kita inputkan akan masuk ke database localhost pad XAMPP dengan tampilan seperti berikut :



1. Kesimpulan

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

Pada prosesnya, saya belajar membuat tabel pengguna pada MySQL dan menghubungkannya dengan aplikasi Java melalui MySQL Connector. Dengan menggunakan konsep Pemrograman Berorientasi Objek (OOP), saya mengimplementasikan interface dan DAO (Data Access Object) yang bertujuan untuk memisahkan logika bisnis dari logika akses data. DAO memfasilitasi pengelolaan data dengan lebih modular, terstruktur, dan dapat digunakan kembali.

Selain itu, saya juga mengembangkan antarmuka grafis (GUI) yang memudahkan interaksi pengguna untuk mengelola data, mulai dari menambah, mengedit, menghapus, hingga membaca data dari database. Praktikum ini memberikan pemahaman menyeluruh tentang siklus pengelolaan data dalam aplikasi dan memperkuat kemampuan saya dalam mengembangkan aplikasi CRUD berbasis OOP dengan konsep yang terstruktur dan mudah dipelihara.

Kesimpulannya, praktikum ini memberikan landasan yang kuat dalam pengembangan aplikasi database berbasis Java, dan keterampilan yang diperoleh dapat diaplikasikan lebih lanjut pada pengembangan sistem yang lebih kompleks.