DAFTAR PUSTAKA

Abdurahman, H., & Riswaya, A. R. (2014). Aplikasi Pinjaman Pembayaran Secara kredit Pada Bank Yudha Bakti. *Jurnal Computech & Bisnis.* 8, 61-69.

Alvin, & Gusrianty. (2019). Implementasi SMS Gatewaydan Application Programming Interface (API) pada Penjualan Mobil Tangki Berbasis Web. *Jurnal Mahasiswa Aplikasi Teknologi Komputer dan Informasi*, 1, 1-2.

Andoyo, A., & Sujarwadi, A. (2014). Sistem Informasi Berbasis Web Pada Desa Tersnomaju Kecamatan Negeri Katon Kab. Pesawaran. *Jurnal TAM (Technology Acceptance model)*, 3, 1-9.

Barri, M. W., Lumenta, A. S., & Wowor, A. (2015). Perancangan Aplikasi SMS Gateway Untuk Pembuatan Kartu Perpustakaan Di Fakultas Teknik Unsrat. *Jurnal Teknik Elektro dan Komputer*. 4, 23-28.

Fajar, R. (2016). *Mengenal Diagram UML (Unified Modeling Language)*. Diambil dari Codepolitan: https://www.codepolitan.com/mengenal-diagram-uml-unified-modeling-language. (Diakses Pada 2 Oktober 2021).

Firdaus, A., Widodo, S., Sutrisman, A., Nasution, S. G., & Mardiana, R. (2019). Rancang Bangun Sistem Informasi Perpustakaan Menggunakan Web Service Pada Jurusan Teknik Komputer Polsri. *Informanika*, 1, 1-9.

Jaya, T. S. (2018). Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis. *Jurnal Informatika: Jurnal Pengembangan IT*, 3, 45-48.

Jayanto, R. D. (2019). Rancang Bangun Sistem Monitoring Jaringan Menggunakan mikrotik Router OS. *Jatai (Jurnal Mahasiswa Teknik Informatika)*, 3, 391-395.

Jejaring. (2019). *Jenis-Jenis Diagram UML*. Diambil dari Jejaring: https://www.jejaring.web.id/jenis-jenis-diagram-uml/. (Diakses pada 2 Oktober 2021).

Nugroho, F. E. (2016). Perancangan Sistem Informasi Penjualan Onlone Studi Kasus Tokoku. *Jurnal Simetris*. 7, 717-724.

Riyadi, H. (2019). *Pengertian MySQL Beserta Fungsi dan Sejarah Terbentuknya MySQL Secara Lengkap*. Diambil Dari Nesabamedia: https://www.nesabamedia.com/pengertian-*MySQL*/. (Diakses Pada 02 Oktober 2021).

Simangunsong, A. (2018). Sistem Informasi Pengarsipan Dokumen Berbasis Web. *Jurnal Mantik Penusa*, 2, 11-19.

Syafnidawaty. (2020). *Blackbox Testing*. Diambil dari Universitas Raharja: https://raharja.ac.id/2020/10/20/black-box-testing/ (Diakses Pada 03 Oktober 2021).

Udaksana, A, P., & Kusaeri, W. R. (2018). 332 Rancang Bangun Aplikasi Digital School Dengan Java NetBeans IDE 8.1. *Jurnal Polban.* 332-336.

Wikana, A. N., Purwadi, J., & Resyandito. (2007). Implementasi *Remote Method Invocation* (RMI) Untuk Tes *Online* Interaktif Multiuser Pada Local Area Network (LAN). *Informatika: Jurnal Teknologi Komputer dan Informatika.* 3, 6-11.

Wiliani, N., & Zambi, S. (2017). Rancang Bangun Aplikasi Kasir Tiket Nonton Bola Bareng Pada X Kasir Di Suatu Lokasi X Dengan Visual Basic 2010 Dan *MySQL*. *Jurnal Rekayasa Informasi*. 6, 77-83.

Yasin K. (2019). *Pengertian MySQL, Fungsi dan Cara Kerjanya (Lengkap)*. Diambil dari Niagahoster: https://www.niagahoster.co.id/blog/*MySQL*-adalah/. (Diakses Pada 04 Oktober 2021).

MKS075 (2022). *Difference between RPC and RMI*. Diambil dari Geeksforgeeks: https://www.geeksforgeeks.org/difference-between-rpc-and-rmi/ (Diakses Pada 15 Agustus 2022).

Pyia (2019). *RMI (Remote Method Invocation) Dan RPC (Remote Procedure Call)*. Diambil dari Pyia: https://pyia.wordpress.com/2012/01/08/rmi-remote-method-invocation-dan-rpc-remote-procedure-call/ (Diakses Pada 15 Agustus 2022).

Aran (2010). *What is the difference between Java RMI and RPC?*. Diambil dari Stackoverflow: https://stackoverflow.com/questions/2728495/what-is-the-difference-between-java-rmi-and-rpc Diakses Pada 15 Agustus 2022).

Tutorialpoint (2010). *Java RMI - Introduction*. Diambil dari tutorialpoint: https://www.tutorialspoint.com/java_rmi/java_rmi_introduction.htm#:~:text=RMI%20stands%20for%20Remote%20Method,provided%20in%20the%20package%20java. Diakses Pada 20 Agustus 2022).

Geeksforgeeks (2022). *Remote Method Invocation in Java*. Diambil dari Geeksforgeeks:https://www.geeksforgeeks.org/remote-method-invocation-in-java/ Diakses Pada 20 Agustus 2022).

LAMPIRAN

"API ASISTEN"
Membuat API dari tabel asisten untuk digunakan dalam pemanggilan remote method

Project : rmi-asistensi-javafx-api
Package: rmi.asistensi.javafx.api.entity
Class: Asisten.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rmi.asistensi.javafx.api.entity;

import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;
import java.time.LocalDate;
import javafx.beans.property.LongProperty;
import javafx.beans.property.ObjectProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;

/**
 *
 * @author Wawan
 */
public class Asisten implements Externalizable {

    private final LongProperty id = new SimpleLongProperty();
    private final StringProperty nim = new SimpleStringProperty();
    private final StringProperty nama = new SimpleStringProperty();
    private final ObjectProperty<LocalDate> tglLahir = new SimpleObjectProperty<>();
    private final StringProperty jenisKelamin = new SimpleStringProperty();
    private final StringProperty alamat = new SimpleStringProperty();
    private final StringProperty email = new SimpleStringProperty();
    private final StringProperty fakultas = new SimpleStringProperty();
    private final StringProperty jurusan = new SimpleStringProperty();
    private final StringProperty programStudi = new SimpleStringProperty();
    private final StringProperty tahunAngkatan = new SimpleStringProperty();
    private final StringProperty nomorHp = new SimpleStringProperty();
    private final LongProperty idUser = new SimpleLongProperty();

    public long getIdUser() {
        return idUser.get();
    }
```

```
public void setIdUser(long value) {
    idUser.set(value);
}

public LongProperty idUserProperty() {
    return idUser;
}


// Properti Nomor HP
public String getNomorHp() {
    return nomorHp.get();
}

public void setNomorHp(String value) {
    nomorHp.set(value);
}

public StringProperty nomorHpProperty() {
    return nomorHp;
}

// Properti Tahun Angkatan
public String getTahunAngkatan() {
    return tahunAngkatan.get();
}

public void setTahunAngkatan(String value) {
    tahunAngkatan.set(value);
}

public StringProperty tahunAngkatanProperty() {
    return tahunAngkatan;
}

// Properti Pogram Studi
public String getProgramStudi() {
    return programStudi.get();
}

public void setProgramStudi(String value) {
    programStudi.set(value);
}

public StringProperty programStudiProperty() {
    return programStudi;
}

// Properti Jurusan
public String getJurusan() {
    return jurusan.get();
}
```

```
public void setJurusan(String value) {
    jurusan.set(value);
}

public StringProperty jurusanProperty() {
    return jurusan;
}

// Properti Fakultas
public String getFakultas() {
    return fakultas.get();
}

public void setFakultas(String value) {
    fakultas.set(value);
}

public StringProperty fakultasProperty() {
    return fakultas;
}

// Properti Email
public String getEmail() {
    return email.get();
}

public void setEmail(String value) {
    email.set(value);
}

public StringProperty emailProperty() {
    return email;
}

// Properti Alamat
public String getAlamat() {
    return alamat.get();
}

public void setAlamat(String value) {
    alamat.set(value);
}

public StringProperty alamatProperty() {
    return alamat;
}

// Properti Jenis Kelamin
public String getJenisKelamin() {
    return jenisKelamin.get();
}
```

```
public void setJenisKelamin(String value) {
    jenisKelamin.set(value);
}

public StringProperty jenisKelaminProperty() {
    return jenisKelamin;
}

// tgl lahir
public LocalDate getTglLahir() {
    return tglLahir.get();
}

public void setTglLahir(LocalDate value) {
    tglLahir.set(value);
}

public ObjectProperty tglLahirProperty() {
    return tglLahir;
}

// Properti Nama
public String getNama() {
    return nama.get();
}

public void setNama(String value) {
    nama.set(value);
}

public StringProperty namaProperty() {
    return nama;
}

// Properti Nim
public String getNim() {
    return nim.get();
}

public void setNim(String value) {
    nim.set(value);
}

public StringProperty nimProperty() {
    return nim;
}

// Properti ID
public long getId() {
    return id.get();
}

public void setId(long value) {
```

```
        id.set(value);
    }

    public LongProperty idProperty() {
        return id;
    }



    @Override
    public void writeExternal(ObjectOutput out) throws IOException {
        out.writeLong(getId());
        out.writeObject(getNim());
        out.writeObject(getNama());
        out.writeObject(getTglLahir());
        out.writeObject(getJenisKelamin());
        out.writeObject(getAlamat());
        out.writeObject(getEmail());
        out.writeObject(getFakultas());
        out.writeObject(getJurusan());
        out.writeObject(getProgramStudi());
        out.writeObject(getTahunAngkatan());
        out.writeObject(getNomorHp());
        out.writeLong(getIdUser());
    }

    @Override
    public void readExternal(ObjectInput in) throws IOException,
ClassNotFoundException {
        setId(in.readLong());
        setNim((String) in.readObject());
        setNama((String) in.readObject());
        setTglLahir((LocalDate) in.readObject());
        setJenisKelamin((String) in.readObject());
        setAlamat((String) in.readObject());
        setEmail((String) in.readObject());
        setFakultas((String) in.readObject());
        setJurusan((String) in.readObject());
        setProgramStudi((String) in.readObject());
        setTahunAngkatan((String) in.readObject());
        setNomorHp((String) in.readObject());
        setIdUser(in.readLong());

    }

//    @Override
//    public String toString() {
//        return this.getId() +
//                this.getNim() +
//                this.getNama() +
//                this.getJenisKelamin() +
//                this.getAlamat() +
//                this.getEmail() +
```

```
//              this.getFakultas() +
//              this.getJurusan() +
//              this.getProgramStudi() +
//              this.getTahunAngkatan() +
//              this.getNomorHp();
//    }
}
```

"API PRAKTIKAN"
Membuat API dari tabel asisten service yang akan di remote

Project : rmi-asistensi-javafx-api
Package: rmi.asistensi.javafx.api.service
Class : AsistenService.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rmi.asistensi.javafx.api.service;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
import rmi.asistensi.javafx.api.entity.Asisten;

/**
 *
 * @author Wawan
 */
public interface AsistenService extends Remote {

    // create
    Asisten insertByAsisten(Asisten asisten) throws RemoteException;
    Asisten insertAccessCode(Asisten asisten) throws RemoteException;

    // read
    Asisten getById(Long id) throws RemoteException;

    // read all
    List<Asisten> getAllByAsisten() throws RemoteException;
    List<Asisten> getAllByAsisten(String value) throws RemoteException;
    List<Asisten> getAllAccessCode() throws RemoteException;

    // update
    void updateByAsisten(Asisten asisten) throws RemoteException;
    void updateEmailByAsisten(Asisten asisten) throws RemoteException;

    // delete
    void deleteByAsisten(Long id) throws RemoteException;
    void deleteAsistenByIdUser(Long id_user) throws RemoteException;
    void deleteAccessCode(Long id) throws RemoteException;
```

```
    // get data
    List<String> getAsistenByValue(String value) throws RemoteException;
    List<String> getAsistenByNama() throws RemoteException;
    List<String> getAsistenByNama(String nama_asisten) throws RemoteException;
    String getAsistenId(String value) throws RemoteException;
    String getAsistenIdByNama(String value) throws RemoteException;
    String getAsistenNamaById(Long id_asisten) throws RemoteException;
    List<String> getAccessCodeByValue(String value) throws RemoteException;

}
```

"PROGRAM SERVER"
Membuat server dari aplikasi asistensi

Project : rmi-asistensi-javafx-server
Package: rmi.asistensi.javafx.server
Class: Main.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rmi.asistensi.javafx.server;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
import rmi.asistensi.javafx.server.service.AsistenScheduleServiceImpl;
import rmi.asistensi.javafx.server.service.AsistenServiceImpl;
import rmi.asistensi.javafx.server.service.AsistensiServiceImpl;
import rmi.asistensi.javafx.server.service.PraktikanServiceImpl;
import rmi.asistensi.javafx.server.service.UserServiceImpl;
import rmi.asistensi.javafx.server.utilities.DatabaseConnection;

/**
 *
 * @author Wawan
 */
public class Main extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        DatabaseConnection.getConnection();
```

```
        Scanner input = new Scanner(System.in);

        System.out.println("Setting the server");
        System.out.println("==================");
        System.out.print("Input hostname IP address : ");
        String hostname = input.nextLine();
        System.out.print("Input port : ");
        int port = input.nextInt();
        System.out.println("==================\n");

        // You can setting the hostname IP Address on your source code
        System.setProperty("java.rmi.server.hostname", hostname);
        Registry registry = LocateRegistry.createRegistry(port);

        AsistenServiceImpl asistenServiceImpl = new AsistenServiceImpl();
        PraktikanServiceImpl praktikanServiceImpl = new PraktikanServiceImpl();
        UserServiceImpl userServiceImpl = new UserServiceImpl();
        AsistenScheduleServiceImpl asistenScheduleServiceImpl = new
AsistenScheduleServiceImpl();
        AsistensiServiceImpl asistensiServiceImpl = new AsistensiServiceImpl();

        // I think the problem may come from this command line,
        // Cara 1 :
//        AsistenService asistenService = (AsistenService)
UnicastRemoteObject.exportObject(asistenServiceImpl, 0);
//        PraktikanService praktikanService = (PraktikanService)
UnicastRemoteObject.exportObject(praktikanServiceImpl, 0);

//        registry.bind("service", asistenService);
//        registry.bind("service2", praktikanService);

        // Cara 2 :
        // So, if u get eror use this method, the UnicastRemoteObject class
        registry.bind("service", asistenServiceImpl);
        registry.bind("service2", praktikanServiceImpl);
        registry.bind("service3", userServiceImpl);
        registry.bind("service4", asistenScheduleServiceImpl);
        registry.bind("service5", asistensiServiceImpl);

        // ubah jadi :
        // registry.bind("service", personServiceImpl); (bisatong rebind)
        System.out.println("Server is Running");
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

“ASISTEN SERVICE IMPLEMENTATION”
Mengimplementasi service dari API asistensiService

Project : rmi-asistensi-javafx-server
Package: rmi.asistensi.javafx.server.service
Class: AsistenServiceImpl.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rmi.asistensi.javafx.server.service;

import java.rmi.RemoteException;
import static java.rmi.server.RemoteServer.getClientHost;
import java.rmi.server.ServerNotActiveException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import rmi.asistensi.javafx.api.entity.Asistensi;
import rmi.asistensi.javafx.api.service.AsistensiService;
import rmi.asistensi.javafx.server.utilities.DatabaseConnection;

/**
 *
 * @author Wawan
 */
public class AsistensiServiceImpl extends UnicastRemoteObject implements
AsistensiService{

    public AsistensiServiceImpl() throws RemoteException {
    }

    @Override
    public Asistensi insertByAsistensi(Asistensi asistensi) throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request
insertByAsistensi(Asistensi asistensi) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;
        String sql = "insert into asistensi(id, lab, praktikum, asistensi_ke, tgl_asistensi,
jadwal, pesan, summary, link_zoom, status, id_praktikan, id_asisten) values
(null,?,?,?,?,?,?,?,?,?,?,?)";
```

```
      try {
         statement = DatabaseConnection.getConnection().prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
         statement.setString(1, asistensi.getLab());
         statement.setString(2, asistensi.getPraktikum());
         statement.setString(3, asistensi.getAsistensiKe());
         statement.setDate(4, Date.valueOf(asistensi.getTglAsistensi().toString()));
         statement.setString(5, asistensi.getJadwal());
         statement.setString(6, asistensi.getPesan());
         statement.setString(7, asistensi.getSummary());
         statement.setString(8, asistensi.getLinkZoom());
         statement.setString(9, asistensi.getStatus());
         statement.setLong(10, asistensi.getIdPraktikan());
         statement.setLong(11, asistensi.getIdAsisten());

         statement.executeUpdate();

         ResultSet result = statement.getGeneratedKeys();
         if(result.next())
         {
            asistensi.setId(result.getLong(1));
         }

         result.close();
         System.out.println("[Successfull]");

         return asistensi;

      } catch (SQLException ex)
      {
         System.out.println("[Failed]");
         ex.printStackTrace();
         return null;
      } finally {
         if(statement != null){
            try {
               statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
         }
         try {
             // remember to close the connection to the database
             DatabaseConnection.getConnection().close();
          } catch (SQLException ex) {
            ex.printStackTrace();
          }
      }
   }

   @Override
   public Asistensi getById(Long id) throws RemoteException {
      try {
```

```
            System.out.print("\nClient " + getClientHost() + "request getById(Long id)
method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "select * from asistensi where id = ?";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, id);

            ResultSet result = statement.executeQuery();

            Asistensi asistensi = null;
            if(result.next()){
                asistensi = new Asistensi();

                asistensi.setId(result.getLong("id"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
                asistensi.setJadwal(result.getString("jadwal"));
                asistensi.setPesan(result.getString("pesan"));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setLinkZoom(result.getString("link_zoom"));
                asistensi.setStatus(result.getString("status"));
                asistensi.setIdPraktikan(result.getLong("id_praktikan"));
                asistensi.setIdAsisten(result.getLong("id_asisten"));

            }

            result.close();
            System.out.println("[successfull]");
            return asistensi;

        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        }finally{
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
```

```
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public List<Asistensi> getAllByAsistensi() throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request getAllByAsistensi()
method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select * from asistensi";

        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);

            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setJadwal(result.getString("jadwal"));
                asistensi.setPesan(result.getString("pesan"));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setLinkZoom(result.getString("link_zoom"));
                asistensi.setStatus(result.getString("status"));
                asistensi.setIdPraktikan(result.getLong("id_praktikan"));
                asistensi.setIdAsisten(result.getLong("id_asisten"));

                list.add(asistensi);
            }
            result.close();
            System.out.println("[successfull]");
            return list;

        } catch (SQLException ex) {
```

```
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }

    @Override
    public List<Asistensi> getAllByAsistensi(Long value) throws RemoteException {

       // tampung parameter
       Long val = value;

        try {
            System.out.print("\nClient " + getClientHost() + "request
getAllByAsistensi(String Value) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "select * from asistensi where id_asisten = ?";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, val);

            ResultSet result = statement.executeQuery();

            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));
```

```
asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
            asistensi.setJadwal(result.getString("jadwal"));
            asistensi.setPesan(result.getString("pesan"));
            asistensi.setSummary(result.getString("summary"));
            asistensi.setLinkZoom(result.getString("link_zoom"));
            asistensi.setStatus(result.getString("status"));
            asistensi.setIdPraktikan(result.getLong("id_praktikan"));
            asistensi.setIdAsisten(result.getLong("id_asisten"));

            list.add(asistensi);
          }
          result.close();
          System.out.println("[successfull]");
          return list;
        } catch (SQLException ex) {
          System.out.println("[failed]");
          ex.printStackTrace();
          return null;
        } finally {
          if(statement != null){
            try {
              statement.close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
          }
          try {
            // remember to close the connection to the database
            DatabaseConnection.getConnection().close();
          } catch (SQLException ex) {
            ex.printStackTrace();
          }
        }
    }

    @Override
    public List<Asistensi> getAllJoinTable() throws RemoteException {
        // tampung parameter

        try {
          System.out.print("\nClient " + getClientHost() + "request getAllJoinTable()
method...");
        } catch (ServerNotActiveException ex) {
          ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
            + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
```

```
ON asistensi.id_praktikan = praktikan.id"
            + " JOIN asisten ON asistensi.id_asisten = asisten.id ORDER BY
asistensi.tgl_asistensi DESC";

      try {
         statement = DatabaseConnection.getConnection().createStatement();

         ResultSet result = statement.executeQuery(sql);

         List<Asistensi> list = new ArrayList<Asistensi>();

         while(result.next()){
            Asistensi asistensi = new Asistensi();
            asistensi.setId(result.getLong("id"));
            asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
            asistensi.setNamaAsisten(result.getString("nama_asisten"));
            asistensi.setLab(result.getString("lab"));
            asistensi.setPraktikum(result.getString("praktikum"));
            asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
            asistensi.setSummary(result.getString("summary"));
            asistensi.setJadwal(result.getString("jadwal"));
            asistensi.setPesan(result.getString("pesan"));
            asistensi.setLinkZoom(result.getString("link_zoom"));
            asistensi.setStatus(result.getString("status"));
            asistensi.setIdPraktikan(result.getLong("id_praktikan"));
            asistensi.setIdAsisten(result.getLong("id_asisten"));

            list.add(asistensi);
         }
         result.close();
         System.out.println("[successfull]");
         return list;
      } catch (SQLException ex) {
         System.out.println("[failed]");
         ex.printStackTrace();
         return null;
      } finally {
         if(statement != null){
            try {
               statement.close();
            } catch (SQLException ex) {
               ex.printStackTrace();
            }
         }
         try {
            // remember to close the connection to the database
            DatabaseConnection.getConnection().close();
         } catch (SQLException ex) {
           ex.printStackTrace();
         }
      }
```

```
    }

    @Override
    public List<Asistensi> getAllJoinTable(int value) throws RemoteException {
      // tampung parameter
      int val = value;

       try {
          System.out.print("\nClient " + getClientHost() + "request getAllJoinTable(Long
Value) method...");
       } catch (ServerNotActiveException ex) {
          ex.printStackTrace();
       }

       PreparedStatement statement = null;

       String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
             + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
             + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.asistensi_ke = ?";

       try {
          statement = DatabaseConnection.getConnection().prepareStatement(sql);

          statement.setInt(1, val);

          ResultSet result = statement.executeQuery();

          List<Asistensi> list = new ArrayList<Asistensi>();

          while(result.next()){
             Asistensi asistensi = new Asistensi();
             asistensi.setId(result.getLong("id"));
             asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
             asistensi.setNamaAsisten(result.getString("nama_asisten"));
             asistensi.setLab(result.getString("lab"));
             asistensi.setPraktikum(result.getString("praktikum"));
             asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
             asistensi.setSummary(result.getString("summary"));
             asistensi.setJadwal(result.getString("jadwal"));
             asistensi.setPesan(result.getString("pesan"));
             asistensi.setLinkZoom(result.getString("link_zoom"));
             asistensi.setStatus(result.getString("status"));
             asistensi.setIdPraktikan(result.getLong("id_praktikan"));
             asistensi.setIdAsisten(result.getLong("id_asisten"));

             list.add(asistensi);
```

```
            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }

    @Override
    public List<Asistensi> getAllAsistenJoinTable(Long id_asisten) throws
RemoteException {
        // tampung parameter

        try {
            System.out.print("\nClient " + getClientHost() + "request
getAllAsistenJoinTable(Long id_asisten) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
                + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
                + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.id_asisten = ? ORDER BY asistensi.id DESC";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, id_asisten);
```

```
            ResultSet result = statement.executeQuery();
            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
                asistensi.setNamaAsisten(result.getString("nama_asisten"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
                asistensi.setJadwal(result.getString("jadwal"));
                asistensi.setPesan(result.getString("pesan"));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setLinkZoom(result.getString("link_zoom"));
                asistensi.setStatus(result.getString("status"));
                asistensi.setIdPraktikan(result.getLong("id_praktikan"));
                asistensi.setIdAsisten(result.getLong("id_asisten"));

                list.add(asistensi);
            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }

    @Override
    public List<Asistensi> getAllAsistenJoinTable(Long id_asisten, String status) throws
RemoteException {
        // tampung parameter

        try {
```

```
            System.out.print("\nClient " + getClientHost() + "request
getAllAsistenJoinTable(Long id_asisten, String status) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
                + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
                + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.id_asisten = ? AND asistensi.status = '"+status+"' ORDER BY asistensi.id
DESC";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, id_asisten);

            ResultSet result = statement.executeQuery();
            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
                asistensi.setNamaAsisten(result.getString("nama_asisten"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
                asistensi.setJadwal(result.getString("jadwal"));
                asistensi.setPesan(result.getString("pesan"));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setLinkZoom(result.getString("link_zoom"));
                asistensi.setStatus(result.getString("status"));
                asistensi.setIdPraktikan(result.getLong("id_praktikan"));
                asistensi.setIdAsisten(result.getLong("id_asisten"));

                list.add(asistensi);
            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
```

```
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }

    @Override
    public List<Asistensi> getAllAsistenJoinTable(int value, Long id_asisten) throws
RemoteException {
        // tampung parameter

        try {
            System.out.print("\nClient " + getClientHost() + "request
getAllAsistenJoinTable(int value, Long Value) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
                + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
                + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.asistensi_ke = ? AND asistensi.id_asisten = ? ORDER BY asistensi.id DESC";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setInt(1, value);
            statement.setLong(2, id_asisten);

            ResultSet result = statement.executeQuery();
            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
```

```
            asistensi.setNamaAsisten(result.getString("nama_asisten"));
            asistensi.setLab(result.getString("lab"));
            asistensi.setPraktikum(result.getString("praktikum"));
            asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
            asistensi.setJadwal(result.getString("jadwal"));
            asistensi.setPesan(result.getString("pesan"));
            asistensi.setSummary(result.getString("summary"));
            asistensi.setLinkZoom(result.getString("link_zoom"));
            asistensi.setStatus(result.getString("status"));
            asistensi.setIdPraktikan(result.getLong("id_praktikan"));
            asistensi.setIdAsisten(result.getLong("id_asisten"));

            list.add(asistensi);
          }
          result.close();
          System.out.println("[successfull]");
          return list;
        } catch (SQLException ex) {
          System.out.println("[failed]");
          ex.printStackTrace();
          return null;
        } finally {
          if(statement != null){
            try {
              statement.close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
          }
          try {
            // remember to close the connection to the database
            DatabaseConnection.getConnection().close();
          } catch (SQLException ex) {
            ex.printStackTrace();
          }
        }
    }

    @Override
    public List<Asistensi> getAllPraktikanJoinTable(Long id_praktikan) throws
RemoteException {
      // tampung parameter
//       System.out.println(result.getString("nama_praktikan"));

       try {
          System.out.print("\nClient " + getClientHost() + "request
getAllPraktikanJoinTable(Long Value) method...");
       } catch (ServerNotActiveException ex) {
          ex.printStackTrace();
       }
```

```
        PreparedStatement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
                + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
                + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.id_praktikan = ? ORDER BY asistensi.id DESC";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, id_praktikan);

            ResultSet result = statement.executeQuery();
            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
                asistensi.setNamaAsisten(result.getString("nama_asisten"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
                asistensi.setJadwal(result.getString("jadwal"));
                asistensi.setPesan(result.getString("pesan"));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setLinkZoom(result.getString("link_zoom"));
                asistensi.setStatus(result.getString("status"));
                asistensi.setIdPraktikan(result.getLong("id_praktikan"));
                asistensi.setIdAsisten(result.getLong("id_asisten"));

                list.add(asistensi);
            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
```

```
        }
        try {
            // remember to close the connection to the database
            DatabaseConnection.getConnection().close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}


@Override
public List<Asistensi> getAllPraktikanJoinTable(int value, Long id_praktikan) throws
RemoteException {
    // tampung parameter

    try {
        System.out.print("\nClient " + getClientHost() + "request
getAllPraktikanJoinTable(int value, Long Value) method...");
    } catch (ServerNotActiveException ex) {
        ex.printStackTrace();
    }

    PreparedStatement statement = null;

    String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
            + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
            + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.asistensi_ke = ? AND asistensi.id_praktikan = ? ORDER BY asistensi.id
DESC";

    try {
        statement = DatabaseConnection.getConnection().prepareStatement(sql);

        statement.setInt(1, value);
        statement.setLong(2, id_praktikan);

        ResultSet result = statement.executeQuery();
        List<Asistensi> list = new ArrayList<Asistensi>();

        while(result.next()){
            Asistensi asistensi = new Asistensi();
            asistensi.setId(result.getLong("id"));
            asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
            asistensi.setNamaAsisten(result.getString("nama_asisten"));
            asistensi.setLab(result.getString("lab"));
            asistensi.setPraktikum(result.getString("praktikum"));
            asistensi.setAsistensiKe(result.getString("asistensi_ke"));
```

```
asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
            asistensi.setJadwal(result.getString("jadwal"));
            asistensi.setPesan(result.getString("pesan"));
            asistensi.setSummary(result.getString("summary"));
            asistensi.setLinkZoom(result.getString("link_zoom"));
            asistensi.setStatus(result.getString("status"));
            asistensi.setIdPraktikan(result.getLong("id_praktikan"));
            asistensi.setIdAsisten(result.getLong("id_asisten"));

            list.add(asistensi);
          }
          result.close();
          System.out.println("[successfull]");
          return list;
        } catch (SQLException ex) {
          System.out.println("[failed]");
          ex.printStackTrace();
          return null;
        } finally {
          if(statement != null){
            try {
              statement.close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
          }
          try {
            // remember to close the connection to the database
            DatabaseConnection.getConnection().close();
          } catch (SQLException ex) {
            ex.printStackTrace();
          }
        }
    }


    @Override
    public void updateByAsistensi(Asistensi asistensi) throws RemoteException {
        try {
          System.out.print("\nClient " + getClientHost() + "request
updateByAsistensi(Asistensi asistensi) method...");
        } catch (ServerNotActiveException ex) {
          ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "update asistensi set lab = ?"
              +", praktikum = ?, asistensi_ke = ?"
              +", tgl_asistensi = ?, jadwal = ?, pesan = ?, summary = ?, link_zoom = ?"
              +", status = ?, id_praktikan = ?"
              +", id_asisten = ?"
```

```
            +" where id = ?";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);
            statement.setString(1, asistensi.getLab());
            statement.setString(2, asistensi.getPraktikum());
            statement.setString(3, asistensi.getAsistensiKe());
            statement.setDate(4, Date.valueOf(asistensi.getTglAsistensi().toString()));
            statement.setString(5, asistensi.getJadwal());
            statement.setString(6, asistensi.getPesan());
            statement.setString(7, asistensi.getSummary());
            statement.setString(8, asistensi.getLinkZoom());
            statement.setString(9, asistensi.getStatus());
            statement.setLong(10, asistensi.getIdPraktikan());
            statement.setLong(11, asistensi.getIdAsisten());

            statement.setLong(12, asistensi.getId());

            statement.executeUpdate();

            System.out.println("[successfull]");

        } catch (SQLException ex) {
            System.out.println("[Failed]");
            ex.printStackTrace();
        } finally {
            if(statement != null)
            {
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public void deleteByAsistensi(Long id) throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request deleteByAsistensi(Long
id) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;
```

```
        String sql = "delete from asistensi where id = ?";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, id);

            statement.executeUpdate();

            System.out.println("[successfull]");

        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public List<String> getDataByValue(String value, String value2) throws
RemoteException {
        String val = value;
        try {
            System.out.print("\nClient " + getClientHost() + "request getDataByValue(String
value, String value2) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select asistensi_ke from asistensi where asistensi_ke = '"+val+"'";

        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);

             List<String> list = new ArrayList<>();
```

```
            while(result.next()){
                list.add(result.getString("asistensi_ke"));

            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public List<String> getDataJoinTable() throws RemoteException {
//        String val = value;
        try {
            System.out.print("\nClient " + getClientHost() + "request getDataJoinable()
method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
                + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status from asistensi JOIN praktikan ON asistensi.id_praktikan = praktikan.id"
                + " JOIN asisten ON asistensi.id_asisten = asisten.id";

        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);
```

```
            List<String> list = new ArrayList<>();


            while(result.next()){
                list.add(result.getString("id"));
                list.add(result.getString("nama_praktikan"));
                list.add(result.getString("nama_asisten"));
                list.add(result.getString("lab"));
                list.add(result.getString("praktikum"));
                list.add(result.getString("asistensi_ke"));
                list.add(result.getString("tgl_asistensi"));
                list.add(result.getString("jadwal"));
                list.add(result.getString("pesan"));
                list.add(result.getString("summary"));
                list.add(result.getString("link_zoom"));
                list.add(result.getString("status"));
            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public List<String> getDataLab() throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request getDataLab()
method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select lab from lab";
```

```
        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);

             List<String> list = new ArrayList<>();

            while(result.next()){
                list.add(result.getString("lab"));

            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public List<String> getDataPraktikum(Long id_lab) throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request getDataPraktikum(Long
id_lab) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select praktikum from praktikum where id_lab = "+id_lab;

        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);
```

```
            List<String> list = new ArrayList<>();

            while(result.next()){
                list.add(result.getString("praktikum"));

            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    @Override
    public Long getLabId(String nama_lab) throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request getLabId(String
nama_lab) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select id from lab where lab = '"+nama_lab+"'";

        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);

            Long hasil = null;

            while(result.next()){
                hasil = result.getLong("id");
```

```
            }
            result.close();
            System.out.println("[successfull]");
            return hasil;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }

    @Override
    public Long getAsistensiByValue(String lab, String praktikum, int asistensi_ke, Long
id_praktikan) throws RemoteException {
        try {
            System.out.print("\nClient " + getClientHost() + "request
getAsistensiByValue(String lab, String praktikum, int asistensi_ke, Long id_praktikan)
method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        Statement statement = null;

        String sql = "select asistensi_ke from asistensi where lab = '"+lab+"' AND praktikum
= '"+praktikum+"' AND asistensi_ke = "+asistensi_ke+" AND id_praktikan =
"+id_praktikan;

        try {
            statement = DatabaseConnection.getConnection().createStatement();

            ResultSet result = statement.executeQuery(sql);

            Long hasil = null;

            while(result.next()){
                hasil = result.getLong("asistensi_ke");
```

```
            }
            result.close();
            System.out.println("[successfull]");
            return hasil;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }

    @Override
    public List<Asistensi> getAllPraktikanJoinTableLastRow(Long id_praktikan) throws
RemoteException {
         // tampung parameter
//       System.out.println(result.getString("nama_praktikan"));

        try {
            System.out.print("\nClient " + getClientHost() + "request
getAllPraktikanJoinTableLastRow(Long id_praktikan) method...");
        } catch (ServerNotActiveException ex) {
            ex.printStackTrace();
        }

        PreparedStatement statement = null;

        String sql = "select asistensi.id, praktikan.nama AS nama_praktikan, asisten.nama
AS nama_asisten, asistensi.lab, asistensi.praktikum, asistensi.asistensi_ke,
asistensi.tgl_asistensi"
                + ", asistensi.jadwal, asistensi.pesan, asistensi.summary, asistensi.link_zoom,
asistensi.status, asistensi.id_praktikan, asistensi.id_asisten from asistensi JOIN praktikan
ON asistensi.id_praktikan = praktikan.id"
                + " JOIN asisten ON asistensi.id_asisten = asisten.id WHERE
asistensi.id_praktikan = ? ORDER BY asistensi.id DESC LIMIT 1";

        try {
            statement = DatabaseConnection.getConnection().prepareStatement(sql);

            statement.setLong(1, id_praktikan);
```

```
            ResultSet result = statement.executeQuery();
            List<Asistensi> list = new ArrayList<Asistensi>();

            while(result.next()){
                Asistensi asistensi = new Asistensi();
                asistensi.setId(result.getLong("id"));
                asistensi.setNamaPraktikan(result.getString("nama_praktikan"));
                asistensi.setNamaAsisten(result.getString("nama_asisten"));
                asistensi.setLab(result.getString("lab"));
                asistensi.setPraktikum(result.getString("praktikum"));
                asistensi.setAsistensiKe(result.getString("asistensi_ke"));

asistensi.setTglAsistensi(LocalDate.parse(result.getDate("tgl_asistensi").toString()));
                asistensi.setJadwal(result.getString("jadwal"));
                asistensi.setPesan(result.getString("pesan"));
                asistensi.setSummary(result.getString("summary"));
                asistensi.setLinkZoom(result.getString("link_zoom"));
                asistensi.setStatus(result.getString("status"));
                asistensi.setIdPraktikan(result.getLong("id_praktikan"));
                asistensi.setIdAsisten(result.getLong("id_asisten"));

                list.add(asistensi);
            }
            result.close();
            System.out.println("[successfull]");
            return list;
        } catch (SQLException ex) {
            System.out.println("[failed]");
            ex.printStackTrace();
            return null;
        } finally {
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            try {
                // remember to close the connection to the database
                DatabaseConnection.getConnection().close();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }
    }
}
```

“DATABASE CONNECTION”

Membuat database connection agar dapat terhubung ke database

Project : rmi-asistensi-javafx-server

Package: rmi.asistensi.javafx.server.utilities
Class: DatabaseConnection.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rmi.asistensi.javafx.server.utilities;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Wawan
 */
public class DatabaseConnection {
   private static Connection connection;

   public static Connection getConnection() throws SQLException {
      if (connection == null || connection.isClosed())
       {
          try
          {
             DriverManager.registerDriver(new com.mysql.jdbc.Driver());
             connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rmi_asistensi","root","");
          }
          catch (SQLException ex)
          {
             Logger.getLogger(DatabaseConnection.class.getName()).log(Level.SEVERE,
null, ex);
          }
       }
       return connection;
   }
}
```

"PROGRAM CLIENT"

Membuat main program client agar dapat melakukan remote method ke server

Project : rmi-asistensi-javafx-client
Package: rmi.asistensi.javafx.client
Class: Main.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package rmi.asistensi.javafx.client;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import rmi.asistensi.javafx.api.service.AsistenScheduleService;
import rmi.asistensi.javafx.api.service.AsistenService;
import rmi.asistensi.javafx.api.service.AsistensiService;
import rmi.asistensi.javafx.api.service.PraktikanService;
import rmi.asistensi.javafx.api.service.UserService;

/**
 *
 * @author Wawan
 */
public class Main extends Application {

    private AsistenService asistenService;
    private PraktikanService praktikanService;
    private UserService userService;
    private AsistenScheduleService asistenScheduleService;
    private AsistensiService asistensiService;

    private double yOffset = 0;
    private double xOffset = 0;

    @Override
    public void start(Stage stage) throws Exception {
        Scanner input = new Scanner(System.in);

        try{
            System.out.println("Connect to server");
            System.out.println("================");
            System.out.print("Input Server IP Adress : ");
            String ip = input.nextLine();
            System.out.print("Input Server port : ");
            int port = input.nextInt();
            System.out.println("================");

            Registry registry = LocateRegistry.getRegistry(ip, port);
```

```
    asistenService = (AsistenService) registry.lookup("service");
    praktikanService = (PraktikanService) registry.lookup("service2");
    userService = (UserService) registry.lookup("service3");
    asistenScheduleService = (AsistenScheduleService) registry.lookup("service4");
    asistensiService = (AsistensiService) registry.lookup("service5");

    FXMLLoader loader = new FXMLLoader(getClass().getResource("login.fxml"));

    Parent root = loader.load();
    Scene scene = new Scene(root);

    stage.initStyle(StageStyle.DECORATED.UNDECORATED);

    root.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event){
            xOffset = event.getSceneX();
            yOffset = event.getSceneY();
        }

    });

    root.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event){
            stage.setX(event.getScreenX() - xOffset);
            stage.setY(event.getScreenY() - yOffset);
        }

    });

    scene.getStylesheets().add(getClass().getResource("style.css").toExternalForm());

    LoginController loginController = loader.getController();


    loginController.setMain(this);

    stage.setScene(scene);

    stage.setTitle("Asistensi Apps - Login Form");

    stage.show();

    root.requestFocus();

}catch(Exception ex) {
    System.out.println("-----------------------------------------------");
    System.out.println("Gagal, Maaf IP Adress atau Port tidak valid !");
    System.out.println("-----------------------------------------------");
    System.exit(0);
}
```

```
//          if(userService.getUserByValue(username, password) != null &&
!userService.getUserByValue(username, password).isEmpty()) {
//
//              if(!userService.getUserByValue(username, password).contains("1")) {
//                  FXMLLoader loader = new
FXMLLoader(getClass().getResource("form.fxml"));
//
//                  Parent root = loader.load();
//                  Scene scene = new Scene(root);
//
scene.getStylesheets().add(getClass().getResource("style.css").toExternalForm());
//
//                  FormController controller = loader.getController();
//
//
//                  controller.setMain(this, username, password);
//
//                  stage.setScene(scene);
//                  // supaya diperbesar
//                  stage.setResizable(false);
//
//                  stage.setTitle("Database Admin RMI");
//
//                  stage.show();
//
//                  root.requestFocus();
//
//                  kondisi = false;
//              }else{
//                  System.out.println("\nUsername atau Password tidak valid !!!\n");
//                  //Platform.exit();
//              }
//          }else if(username.equals("EXIT") && password.equals("EXIT")){
//              Platform.exit();
//              kondisi = false;
//          }else{
//              System.out.println("\nUsername atau Password tidak ditemukan !!!\n");
//          }

    }

    public AsistenService getAsistenService(){
        return asistenService;
    }

    public PraktikanService getPraktikanService(){
        return praktikanService;
    }

    public UserService getUserService(){
        return userService;
    }
```

```
    public AsistenScheduleService getAsistenScheduleService(){
        return asistenScheduleService;
    }

    public AsistensiService getAsistensiService(){
        return asistensiService;
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }



}
```