

## **BAB V**

### **JAVA SERVER FAGES**

#### **5.1 Tujuan**

1. Praktikan memahami dan mengetahui pembuatan/penggunaan *program* berbasis *Java Server Faces*
2. Praktikan memahami penggunaan *Tomcat Server*
3. Praktikan dapat mengetahui dan mengenal struktur *Framework* dari JSF
4. Praktikan memahami cara menghubungkan program pada java dengan *MySQL Connector*.
5. Praktikan mampu mengatasi masalah saat menjalankan program menggunakan JSF
6. *Praktikan dapat mengetahui perbedaan penggunaan xhtml dan html.*
7. Praktikan dapat membuat aplikasi CRUD sederhana dengan JSF
8. Praktikan mengetahui fungsi dari *session beans* dan *entity class* dari *managed bean*.

## 5.2 Dasar Teori

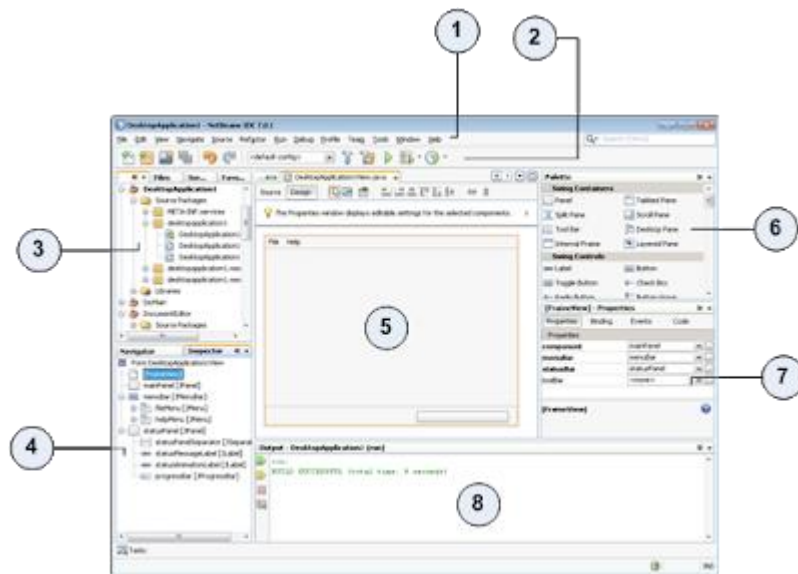
### 5.2.1 NetBeans IDE

Netbeans adalah sebuah aplikasi Integrated Development Environment (IDE) yang berbasiskan Java dari Sun Microsystems yang berjalan di atas swing. Swing merupakan sebuah teknologi Java untuk pengembangan aplikasi desktop yang dapat berjalan pada berbagai macam platform seperti windows, linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface* (GUI), suatu kode editor atau text, suatu compiler dan suatu debugger.

NetBeans juga dapat digunakan programmer untuk menulis, meng-compile, mencari kesalahan dan menyebarkan program NetBeans yang ditulis dalam bahasa pemrograman java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat professional desktop, *enterprise*, web, and *mobile applications* dengan Java language, C/C++, dan bahkan *dynamic languages* seperti PHP, JavaScript, Groovy, dan Ruby.

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama. Dan saat ini pun NetBeans memiliki 2 produk yaitu Platform NetBeans dan NetBeans IDE. Platform NetBeans merupakan framework yang dapat digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi desktop dan Platform NetBeans juga menawarkan layanan-layanan yang umum bagi aplikasi desktop, mengizinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi.

Lingkungan pengembangan yang terintegrasi pada IDE NetBeans, memudahkan pengguna untuk membuat beragam aplikasi yang mudah.



Gambar 5. 1 Tampilan Program Netbeans

- |                                   |  |
|-----------------------------------|--|
| 1. Menu Bar                       | 5. Jendela Utama                                 |
| 2. Toolbar                        | 6. Kontrol Komponen ( <i>Component Palette</i> ) |
| 3. Project Explorer               | 7. Jendela Properties                            |
| 4. Daftar Komponen yang digunakan | 8. Jendela Keluaran ( <i>debugging</i> )         |

NetBeans IDE 6.0 memperkenalkan dukungan untuk mengembangkan modul IDE dan aplikasi klien kaya berdasarkan platform NetBeans, Java Swing GUI builder (sebelumnya dikenal sebagai “Proyek Matisse”), meningkatkan CVS dukungan, WebLogic 9 dan JBoss 4 dukungan, dan perangkat tambahan banyak editor. NetBeans 6 tersedia dalam *repository* resmi dari distribusi Linux utama.

Selain itu, NetBeans Enterprise Pack mendukung pengembangan aplikasi Java EE 5 perusahaan, termasuk SOA alat desain visual, skema XML tools, web layanan (untuk BPEL), dan UML modeling. The NetBeans IDE Bundle for C/C++ supports C/C++ development. The NetBeans IDE Bundle untuk C / C + + mendukung C / C + + pembangunan.

(Sumber: <https://ilmunesia.com/pengertian-dan-sejarah-netbeans/> pada 12 November 2019)

### 5.2.2 Java Server Faces

JSF adalah sebuah framework yang berfungsi untuk membangun suatu *user interface* pada suatu aplikasi web. JSF ini dibangun berdasarkan konsep-konsep yang diperkenalkan oleh framework Struts, dan memiliki keuntungan berupa sebuah arsitektur yang benar-benar memisahkan antara bagian *business logic* dan bagian *standard komponen user interface* yang dibentuk dengan cara yang hampir sama dengan widget Swing.

#### Komponen-Komponen pada Aplikasi JSF

Sebuah aplikasi web memiliki berbagai komponen penting, seperti *UI Components*, *controller components*, dan *navigation components*. *UI components* berfungsi untuk berinteraksi antara user dengan aplikasi web, *controller components* berfungsi untuk memutuskan respons apa yang ingin ditampilkan oleh user, *navigation components* berfungsi untuk mengatur navigasi dari halaman ke halaman yang lain. semua komponen itu juga termasuk ada pada JSF diantaranya:

- UI Components
- Renderers
- Validator
- Managed Bean
- Converters
- Events and Listeners
- Messages
- Navigation

#### Mengenal Tag pada JSF Libraries

Terdapat 2 jenis tag pada teknologi JSF dan berfungsi untuk membuat sebuah model komponen yang berisi tag simpel dan dapat digunakan berulang, tag ini memberikan kemudahan untuk membuat UI yang kaya dan dinamis, berikut adalah 2 tag yang terdapat pada JSF:

- HTML Tag Library
- Core Tag Library

(Sumber: <http://blog.umy.ac.id/broadwell/2015/03/26/pengenalan-jsf-java-server-faces/> pada 12 November 2019)

### 5.2.3 Tomcat Server

Apache Tomcat adalah sebuah *web server open source* dan *servlet container* yang dikembangkan oleh Apache Software Foundation (ASF). Tomcat mengimplementasikan Java Servlet dan Java Server Pages (JSP) dari Oracle dan

menyediakan lingkungan server web HTTP “pure Java” untuk menjalankan kode Java. Apache Tomcat mencakup perangkat untuk konfigurasi dan manajemen, tetapi juga dapat di konfigurasi dengan mengedit file konfigurasi XML.

Apache Tomcat merupakan *web server open source* yang diperuntukkan Java Servlet, JavaServer Page, Java Expression Language, dan Java WebSocket. Semua dukungan teknologi tersebut diawasi oleh Java Community Process. Apache Tomcat dapat digunakan dalam berbagai aplikasi *web* yang mempunyai skala besar dan konkurensi banyak dari berbagai industri dan organisasi.

(Sumber: <https://www.webhozz.com/blog/instalasi-tomcat-di-centos/> pada 12 November 2019)

#### **5.2.4 Session Bean**

Session bean adalah EJB yang digunakan untuk mengeksekusi proses. Isi dari Session Bean ini biasanya berupa kata kerja (transfer, pay, calculate, updateData, dll). Stateless Session Bean (SLSB) adalah Session Bean yang tidak menyimpan state (keadaan) pada setiap kali eksekusi. Berbeda dengan State full Session Bean (SFSB) yang dapat menyimpan state. State ini dapat kita gunakan misalnya untuk menyimpan informasi user atau barang-barang yang sudah dibeli (pada kasus online shop).

(Sumber: <https://suhearie.wordpress.com/2008/08/26/java-enterprise-mulai-dari-mana-part-2-netbeans-glassfish/> pada 12 November 2019)

#### **5.2.5 MySQL Connector**

MySQL connector adalah driver khusus yang dibuat oleh MySQL sendiri yang menyediakan akses ke database MySQL. MySQL connectors juga mempunyai fungsi yang sama seperti ODBC, salah satunya adalah menghubungkan database dengan aplikasi program VB.net atau pemrograman lainnya.

(Sumber: <http://ilmukita.org/instal-mysql-connector/> pada 12 November 2019)

#### **5.2.6 Github**

GitHub adalah layanan penginangan web bersama untuk proyek pengembangan perangkat lunak yang menggunakan sistem pengontrol versi Git dan layanan hosting internet. Hal ini banyak digunakan untuk kode komputer. Ini

memberikan kontrol akses dan beberapa fitur kolaborasi seperti pelacakan bug, permintaan fitur, manajemen tugas, dan wiki untuk setiap proyek. GitHub menawarkan paket *repository* pribadi dan gratis pada akun yang sama dan digunakan untuk proyek perangkat lunak sumber terbuka

Github adalah sebuah website yang memberikan pelayanan untuk menyimpan repo anda secara gratis. Banyak perintah yang ada di git bash dan git gui bisa dilakukan melalui Github. Tidak hanya itu, Github juga memudahkan kolaborasi dalam suatu proyek dengan fitur-fitur tambahan seperti pull request, diskusi di patch, mengatur bugs, dan lain-lain.

Github juga menggabungkan elemen-elemen dari social network ke dalam sistemnya. Untuk urusan hosting Git dan alat kolaborasi, Github adalah website paling populer di dunia maya.

(Sumber: <https://codesaya.com/git/github/mengenal-github/unit/1/> pada 12 November 2019)

### **5.2.7 XAMPP**

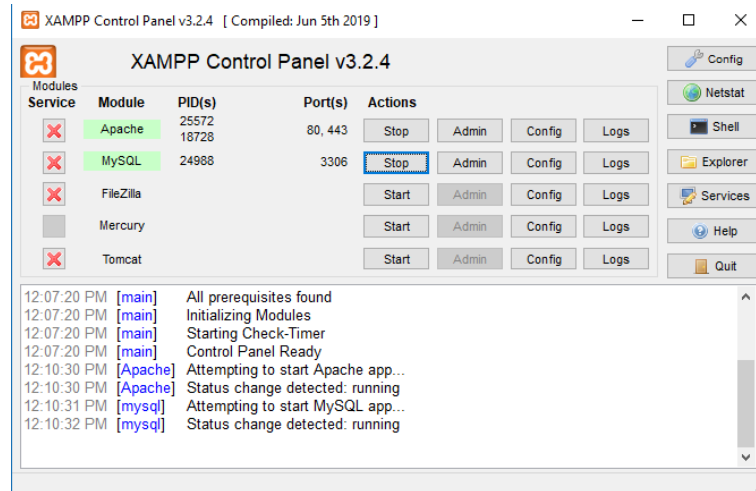
XAMPP merupakan software aplikasi *open source* dan gratis yang bisa diinstall pada berbagai sistem operasi seperti *Windows, Linux, dan Mac OS* yang memiliki fungsi untuk membuat server sendiri pada PC/ Laptop (istilah lainnya **Localhost**). Dengan menggunakan XAMPP kita bisa membuat web server sendiri pada komputer atau laptop yang kita gunakan untuk membuat sebuah aplikasi web.

Contohnya, untuk membuat website menggunakan wordpress terdapat dua pilihan yaitu secara online dan offline. Jika kita ingin membuatnya secara online maka kita harus terlebih dahulu memiliki domain dan hosting. Nah dengan menggunakan XAMPP kita bisa membuatnya secara offline.

(Sumber: <https://badoystudio.com/apa-itu-xampp/> pada 13 November 2019)

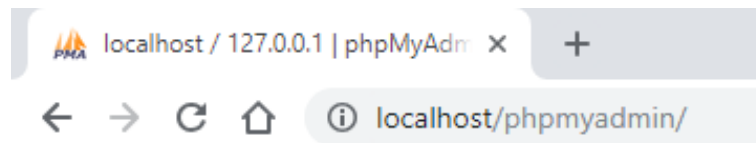
### 5.3 Langkah Percobaan

1. Buka XAMPP, lalu Aktifkan Apache dan MySQL



Gambar 5. 2 Program Control Panel

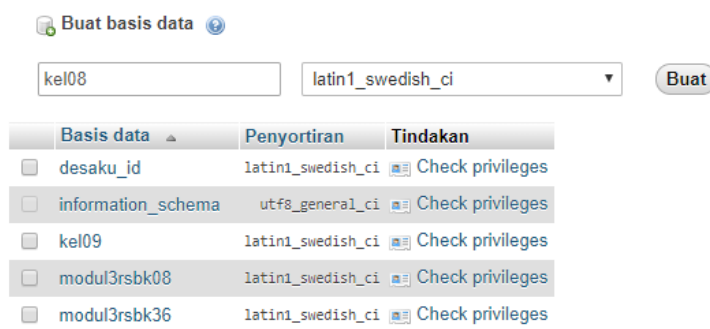
2. Buka Browser lalu ketikkan: localhost/phpMyAdmin



Gambar 5. 3 Panel localhost pada Browser

3. Buat database terlebih dahulu, database tersebut diberi nama kel08

#### Basis data



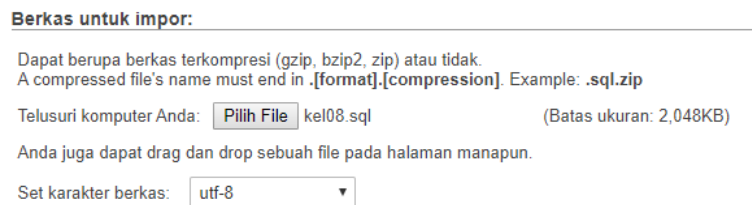
Gambar 5. 4 Membuat database kel08 pada MySQL Database

4. Kemudian, buka Tab impor



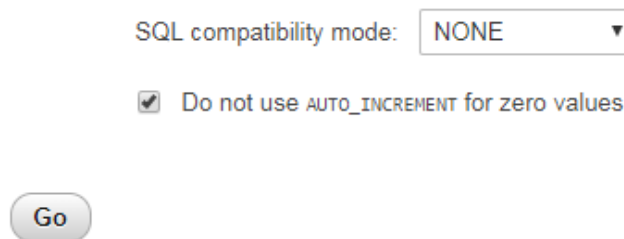
Gambar 5. 5 Import file kel08.sql pada database

5. Lalu, pilih Choose File dan masukkan Database yang sudah disediakan



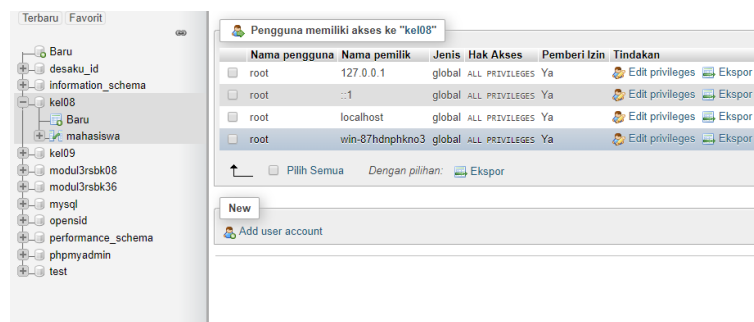
Gambar 5. 6 File kel08.sql telah di import pada database

6. Dan kemudian pilih Go



Gambar 5. 7 Klik Go untuk proses import database dilakukan

7. Setelah itu buka tab Hak Akses/ *privilege* dan pilih *add user account*



Gambar 5. 8 Add User Account pada *Privileges Database*



8. Setelah masuk ke halaman add user account, ikuti langkah seperti digambar

Gambar 5. 9 Isi file Nama pengguna dan kata sandi

Keterangan:

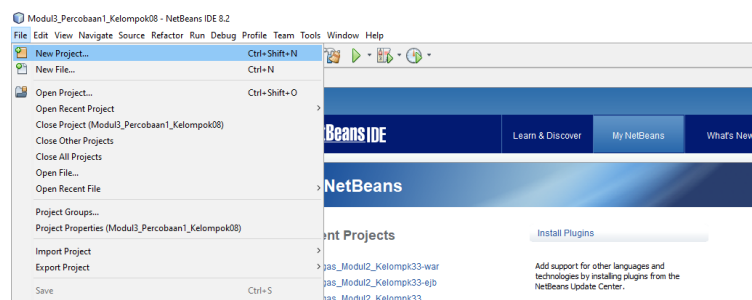
- Nama Pengguna : kel08
- Nama Pemilik : localhost (form sampingnya harus lokal)
- Kata sandi : kel08

Setelah itu pilih semua dan kirim

Gambar 5. 10 Cek List Pilih Semua pada hak akses global

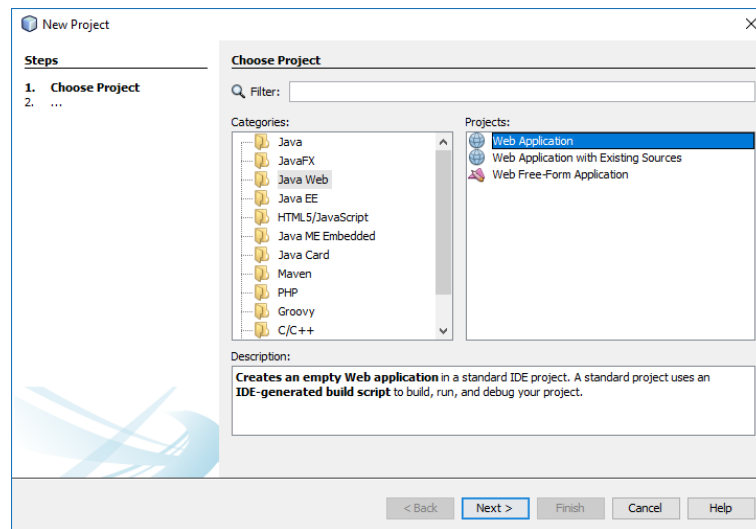
9. Kemudian, buka NetBeans kalian dan pastikan sudah terinstall Tomcat

10. Kemudian pilih File, lalu New Project



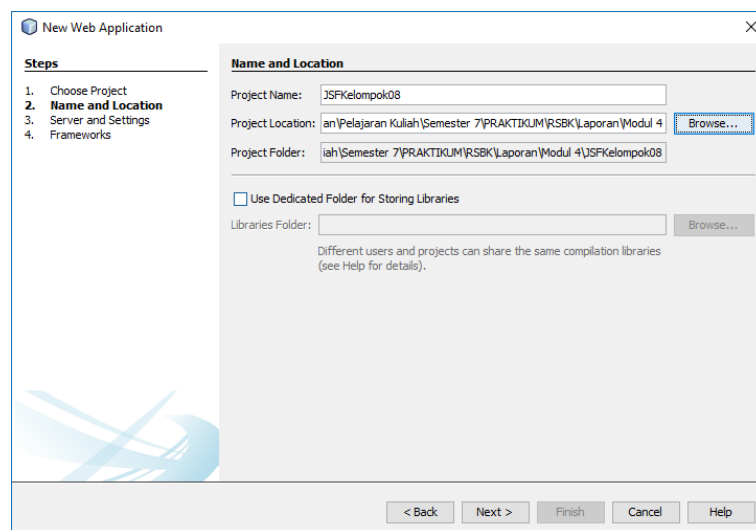
Gambar 5. 11 Create New Project

11. Lalu pilih Java Web, kemudian pilih Web Application, kemudian Next



Gambar 5. 12 Create Project Web Application

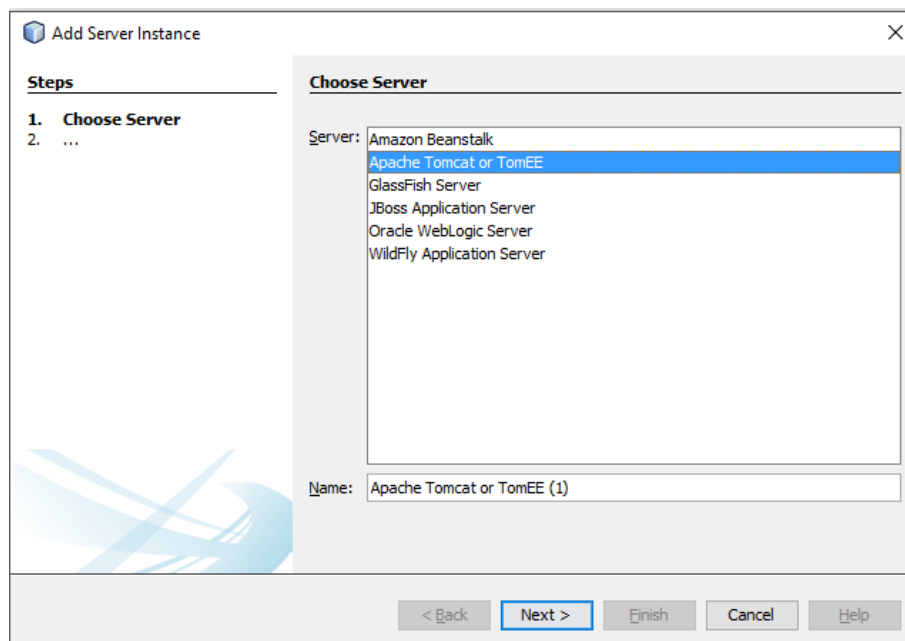
12. Lalu, beri nama 'JSFKelompok08', kemudian Next.



Gambar 5. 13 Isi Project Name dengan JSFKelompok08

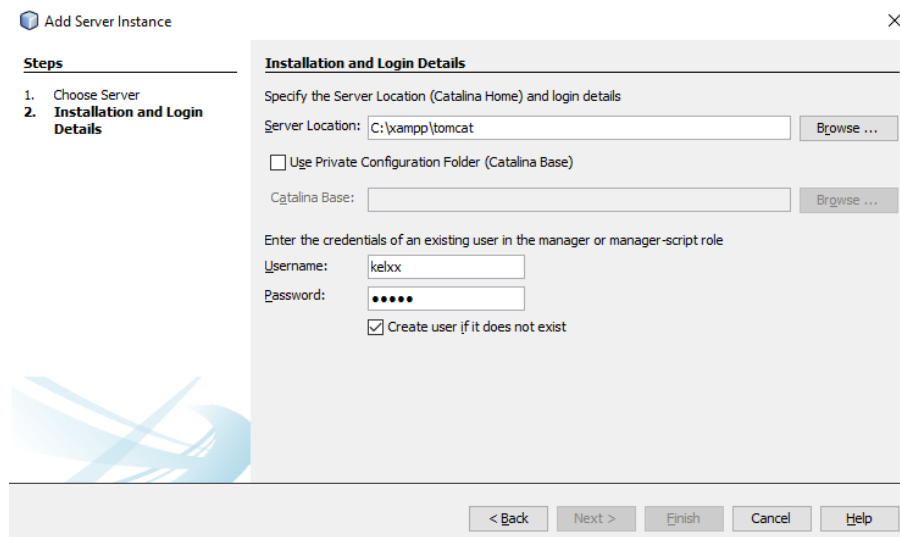
13. Kemudian untuk Server, pilih Apache Tomcat or TomEE, dan untuk Java EE Version pilih Java EE 6 Web, jika belum ada tekan tombol add.

14. Terus pilih Apache Tomcat or TomEE. next



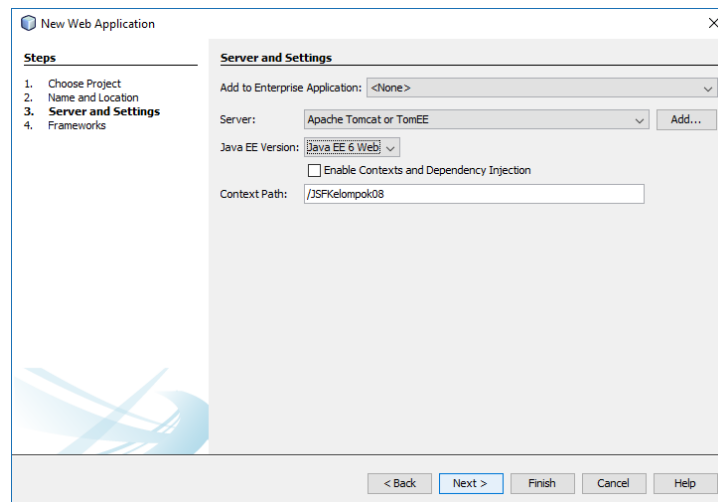
Gambar 5. 14 Jendela *Add Server Instance*

15. Maka akan ada form seperti gambar dibawah dan jangan lupa tekan tombol finish



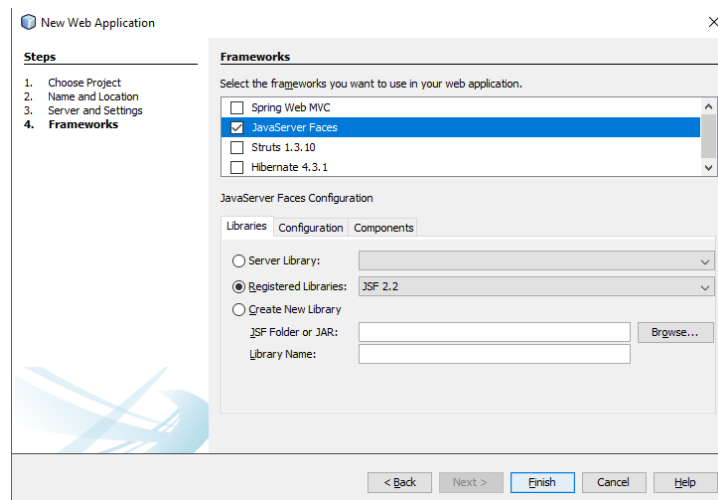
Gambar 5. 15 Jendela *Installation and Login Details*

16. Setelah berhasil menambahkan server Apache Tomcat or TomEE, tekan next



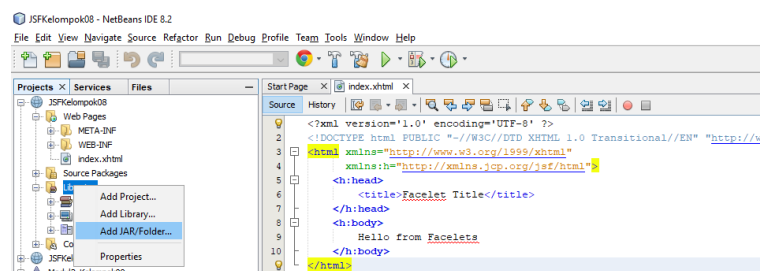
Gambar 5. 16 Jendela *Server and Settings*

17. Kemudian untuk Framework, centang JavaServer Faces, dan untuk pengaturan ikuti gambar di bawah. Lalu tekan Finish

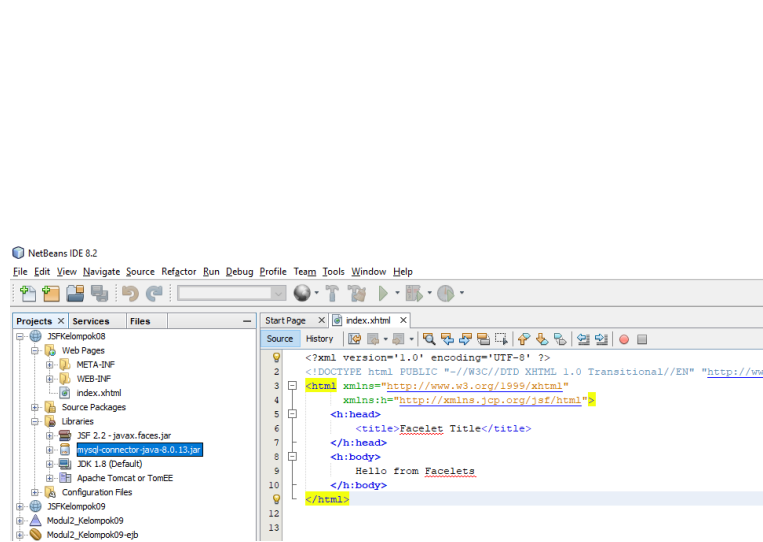


Gambar 5. 17 Jendela *Framework* dan *Cek List JavaServer Pages*

18. Selanjutnya, tambahkan MySQLConnector dengan cara klik kanan pada Folder Libraries dan pilih Add JAR/Folder

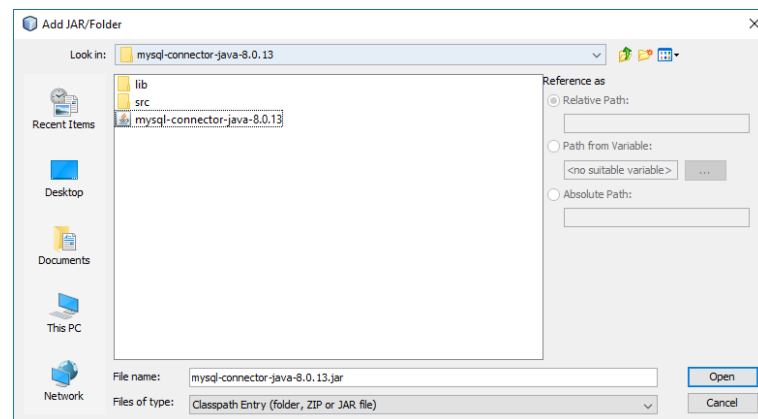


Gambar 5. 18 Cara menambahkan Add JAR/Folder pada project



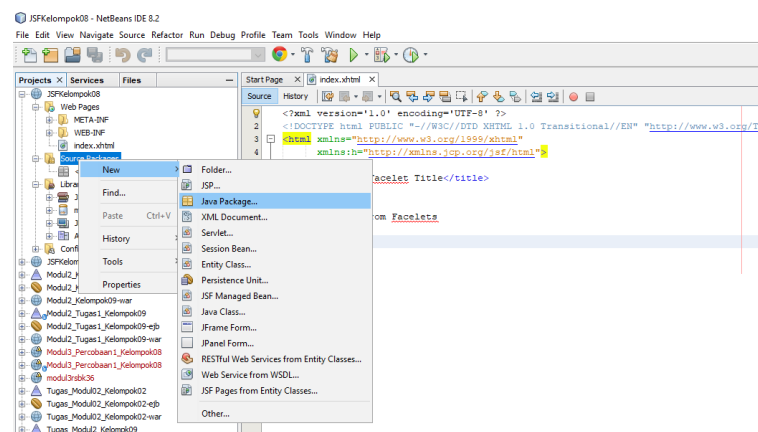
Gambar 5. 19 Kondisi ketika sudah ditambah MySQLConnector

19. Kemudian, cari lokasi dimana kalian menyimpan MySQL Connectornya, lalu tekan Open



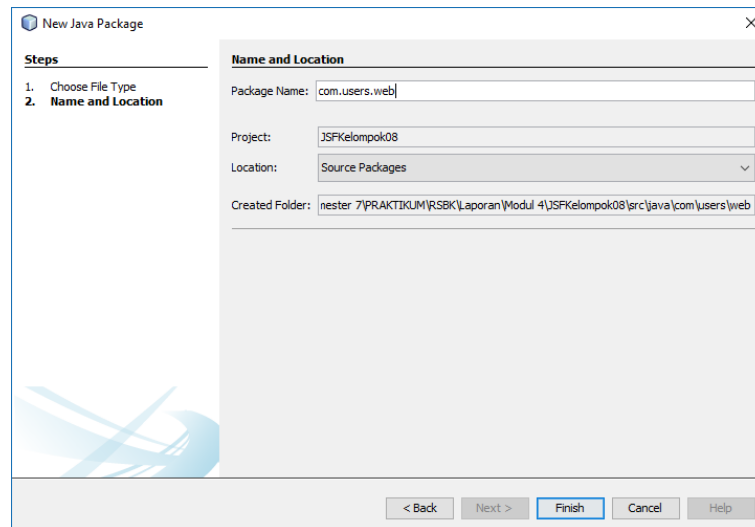
Gambar 5. 20 Lokasi MySQLConnector pada folder komputer

20. Kemudian, pada bagian Source Package, klik kanan dan pilih New, lalu pilih Java Package

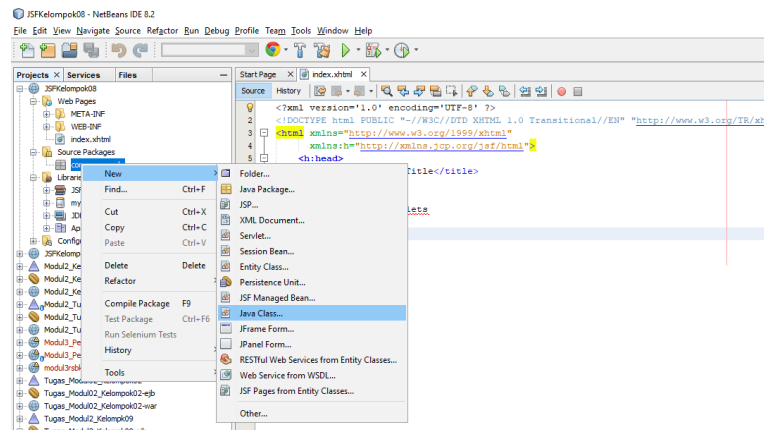


Gambar 5. 21 Cara membuat Java Package pada Project

21. Lalu beri nama 'com.users.web', kemudian klik Finish. Lalu pada Folder com.users.web, klik kanan lalu pilih New dan pilih Java Class

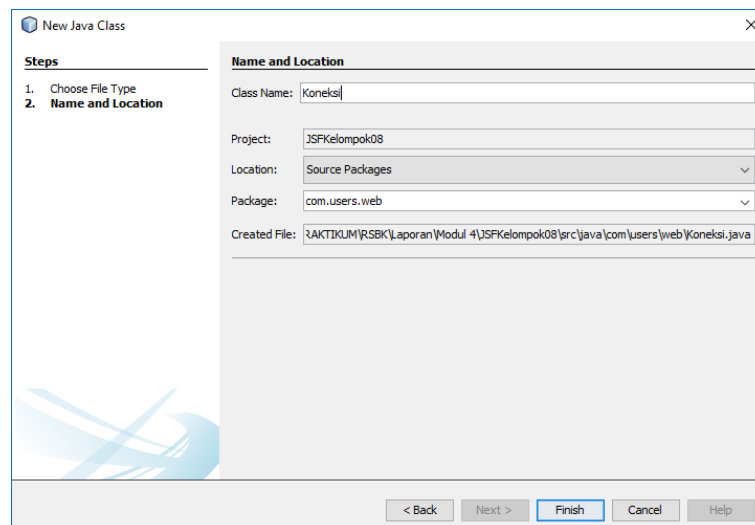


Gambar 5. 22 Memberi nama com.users.web pada Java package



Gambar 5. 23 Cara membuat Java Class pada folder com.users.web

22. Kemudian berikan nama 'Koneksi', lalu tekan Finish



Gambar 5. 24 Memberi Class Name Koneksi

23. Lalu, selanjutnya. Copy kodingan ‘Koneksi.txt’ yang sudah disediakan ke dalam file Koneksi.java. Tepatnya, di dalam perintah public class Koneksi. Dan berikut adalah sourcenya.

```
package com.users.web;

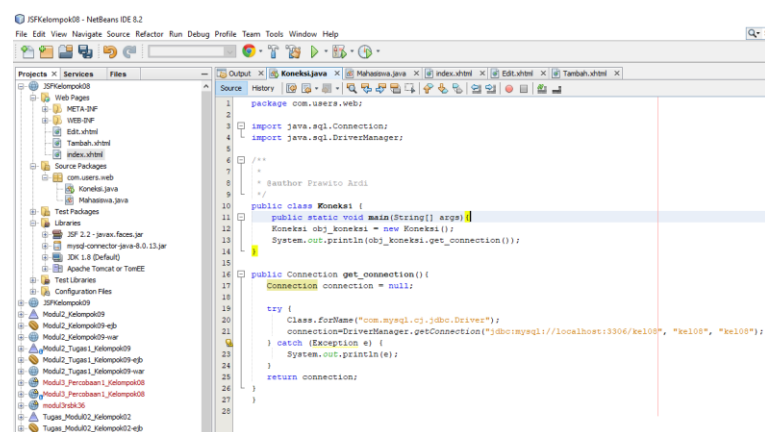
import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author Prawito Ardi
 */
public class Koneksi {
    public static void main(String[] args){
        Koneksi obj_koneksi = new Koneksi();
        System.out.println(obj_koneksi.get_connection());
    }

    public Connection get_connection(){
        Connection connection = null;

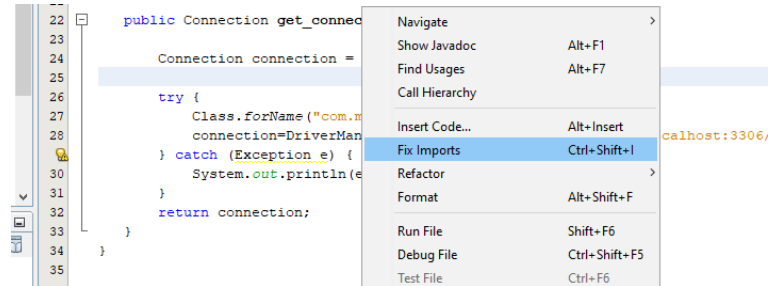
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/kel08", "kel08", "kel08");
        } catch (Exception e) {
            System.out.println(e);
        }
        return connection;
    }
}
```



Gambar 5. 25 Memberi *Source Code* pada java class Koneksi

24. Lalu apabila ditemukan Error ketika selesai mengcopy, maka dapat klik kanan pada area kerja, dan pilih Fix Import



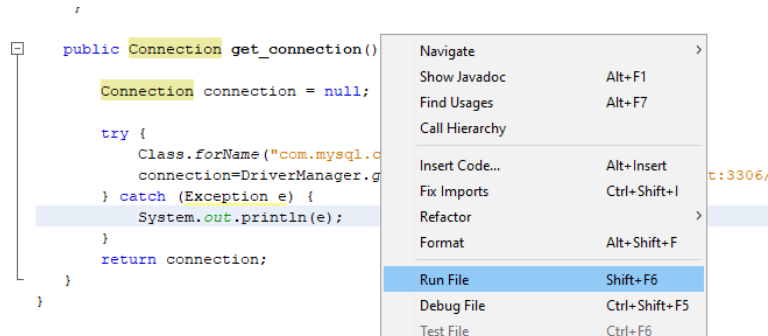
Gambar 5. 26 Klik Kanan Fix Import pada file untuk mengatasi error

25. Selanjutnya sesuaikan source code dengan SS-an yang ada diatas dan sesuaikan dengan user hak akses yang sudah di buat sebelumnya



Gambar 5. 27 Memberikan user hak akses agar dapat ter koneksi dengan database

26. Kemudian klik kanan pada Area kerja dan pilih Run File untuk memastikan bahwa berhasil konek ke Database



Gambar 5. 28 Run File untuk mengetest apakah sudah terhubung dengan database

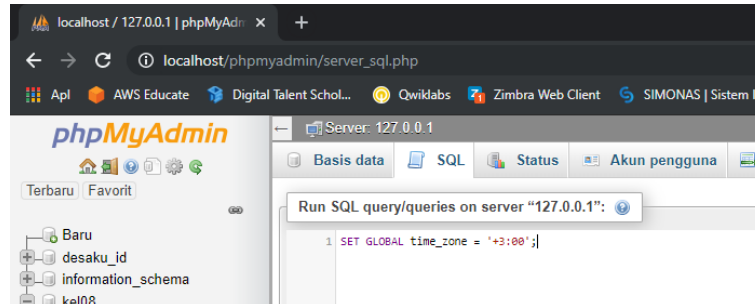
27. Apabila berhasil, maka akan muncul tulisan seperti berikut:



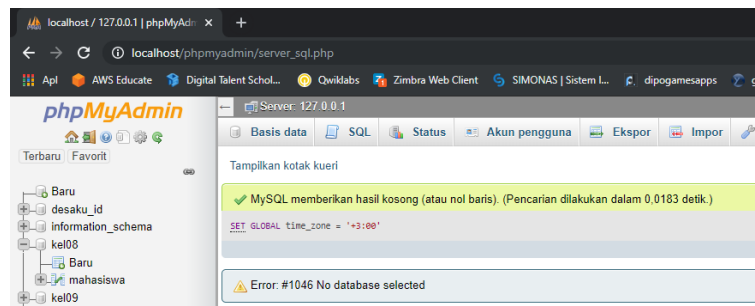
Gambar 5. 29 Kondisi ketika sudah terhubung dengan database



28. Jika terjadi suatu kesalahan / error (TIME ZONE ISSUE) dapat menggunakan perintah yang ada di file solve.txt dengan cara dipaste ke database masing2

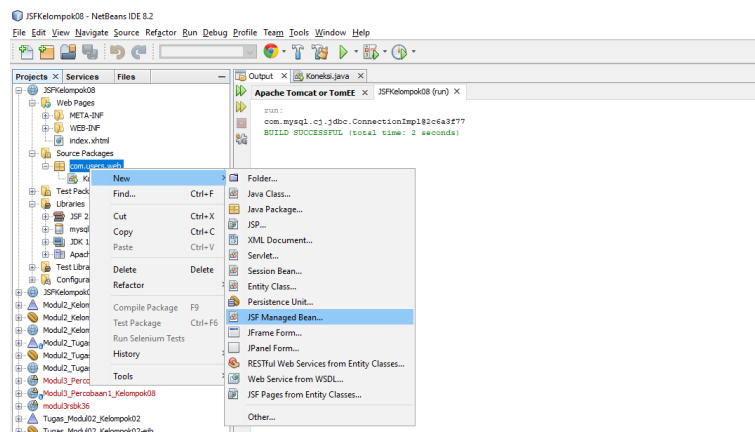


Gambar 5. 30 Memberikan kode pada SQL untuk mengatasi error Time Zone Issue



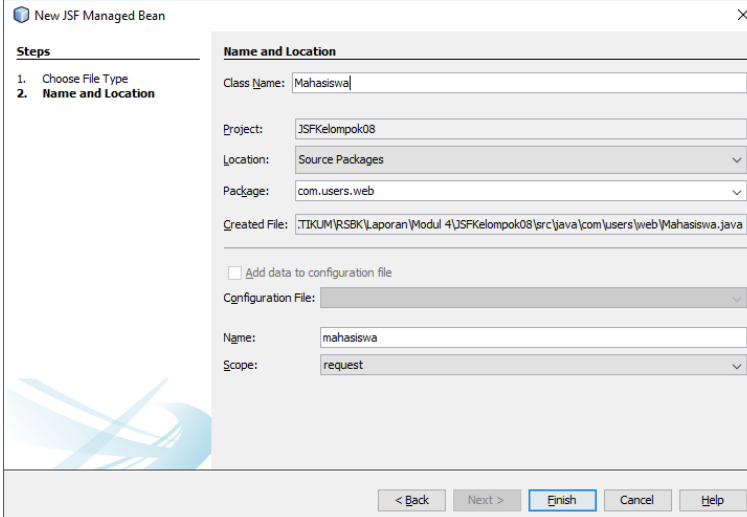
Gambar 5. 31 SQL berhasil diberikan

29. Atau dapat mengganti file mySql jdbc rar nya
30. Selanjutnya, klik kanan lagi pada;com.users.web;. Lalu pilih New, dan pilih JSF Managed Bean. Jika pilihan tidak tersedia klik pilihan “Other”



Gambar 5. 32 Langkah untuk membuat JSF Managed Bean pada project

31. Lalu beri nama Mahasiswa, dan pada bagian Scope. Ubah menjadi Session. Kemudian klik Finish



The screenshot shows the 'New JSF Managed Bean' dialog box. The 'Name and Location' tab is selected. The 'Class Name' field contains 'Mahasiswa'. The 'Project' is 'JSFKelompok08'. The 'Location' is 'Source Packages'. The 'Package' is 'com.users.web'. The 'Created File' path is '.TIKJUM\RSBK\Laporan\Modul 4\JSFKelompok08\src\java\com\users\web\Mahasiswa.java'. The 'Add data to configuration file' checkbox is unchecked. The 'Configuration File' is empty. The 'Name' field is 'mahasiswa'. The 'Scope' is 'request'. The 'Finish' button is highlighted.

Gambar 5. 33 Memberi Class Name Mahasiswa untuk JSF Managed Bean

32. Selanjutnya, pada Area Kerja 'Mahasiswa.java' yang terletak dibawah tulisan 'package com.users.web', isi dengan Kodingan yang telah disediakan pada 'Mahasiswa.txt'. Dan berikut adalah sourcenya.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Map;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

/**
 *
 * @author
 */
@ManagedBean
@RequestScoped
public class Mahasiswa {

    /**
     * Creates a new instance of Mahasiswa
     */

    private String NIM;
    public void setNIM(String NIM) {
        this.NIM = NIM;
    }
    public String getNIM() {
        return NIM;
    }
}
```

```

    }

    private String NAMA;
    public void setNAMA(String NAMA) {
        this.NAMA = NAMA;
    }
    public String getNAMA() {
        return NAMA;
    }

    private String PENJURUSAN;
    public void setPENJURUSAN(String PENJURUSAN) {
        this.PENJURUSAN = PENJURUSAN;
    }
    public String getPENJURUSAN() {
        return PENJURUSAN;
    }

    private Map<String, Object> sessionMap =
FacesContext.getCurrentInstance().getExternalContext().getSe
ssionMap();

    public String Edit_Mahasiswa(){
        FacesContext fc = FacesContext.getCurrentInstance();
        Map<String, String> params =
fc.getExternalContext().getRequestParameterMap();
        String Field_NIM = params.get("action");
        try {
            Koneksi obj_koneksi = new Koneksi();
            Connection connection =
obj_koneksi.get_connection();
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery("select * from
mahasiswa where NIM="+Field_NIM);
            Mahasiswa obj_Mahasiswa = new Mahasiswa();
            rs.next();
            obj_Mahasiswa.setNIM(rs.getString("NIM"));
            obj_Mahasiswa.setNAMA(rs.getString("Nama"));

obj_Mahasiswa.setPENJURUSAN(rs.getString("Penjurusan"));
            sessionMap.put("EditMahasiswa", obj_Mahasiswa);
        } catch (Exception e) {
            System.out.println(e);
        }
        return "/Edit.xhtml?faces-redirect=true";
    }

    public String Delete_Mahasiswa(){
        FacesContext fc = FacesContext.getCurrentInstance();
        Map<String, String> params =
fc.getExternalContext().getRequestParameterMap();
        String Field_NIM = params.get("action");
        try {
            Koneksi obj_koneksi = new Koneksi();
            Connection connection =
obj_koneksi.get_connection();

```

```

        PreparedStatement ps =
connection.prepareStatement("delete from mahasiswa where
NIM=?");
        ps.setString(1, Field_NIM);
        System.out.println(ps);
        ps.executeUpdate();
    } catch (Exception e) {
        System.out.println(e);
    }
    return "/index.xhtml?faces-redirect=true";
}

public String Update_Mahasiswa(){
    FacesContext fc = FacesContext.getCurrentInstance();
    Map<String,String> params =
fc.getExternalContext().getRequestParameterMap();
    String Update_NIM= params.get("Update_NIM");

    try {
        Koneksi obj_koneksi = new Koneksi();
        Connection connection =
obj_koneksi.get_connection();
        PreparedStatement ps =
connection.prepareStatement("update mahasiswa set NIM=?,
Nama=?, Penjurusan=? where NIM=?");
        ps.setString(1, NIM);
        ps.setString(2, NAMA);
        ps.setString(3, PENJURUSAN);
        ps.setString(4, Update_NIM);
        System.out.println(ps);
        ps.executeUpdate();
    } catch (Exception e) {
        System.out.println(e);
    }

    return "/index.xhtml?faces-redirect=true";
}

public ArrayList getGet_all_mahasiswa() throws
Exception{
    ArrayList list_of_mahasiswa=new ArrayList();
    Connection connection=null;
    try {
        Koneksi obj_koneksi = new Koneksi();
        connection = obj_koneksi.get_connection();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery("Select * from
mahasiswa");
        while(rs.next()){
            Mahasiswa obj_Mahasiswa = new Mahasiswa();
            obj_Mahasiswa.setNIM(rs.getString("NIM"));
            obj_Mahasiswa.setNAMA(rs.getString("Nama"));

obj_Mahasiswa.setPENJURUSAN(rs.getString("Penjurusan"));
            list_of_mahasiswa.add(obj_Mahasiswa);
        }
    }
}

```

```

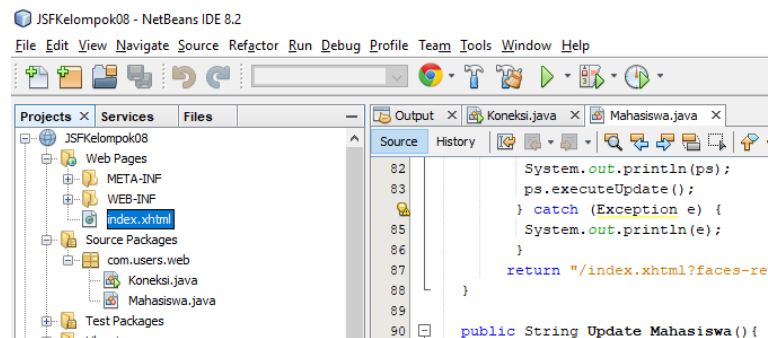
    } catch (Exception e) {
        System.out.println(e);
    } finally {
        if (connection != null) {
            connection.close();
        }
    }
    return list_of_mahasiswa;
}

public String Tambah_Mahasiswa() {
    try {
        Connection connection = null;
        Koneksi obj_koneksi = new Koneksi();
        connection = obj_koneksi.get_connection();
        PreparedStatement
ps = connection.prepareStatement("insert into mahasiswa (NIM,
Nama, Jurusan)
value ('"+NIM+"', '"+NAMA+"', '"+PENJURUSAN+"')");
        ps.executeUpdate();
    } catch (Exception e) {
        System.out.println(e);
    }
    return "/index.xhtml?faces-redirect=true";
}

public Mahasiswa() {}
}

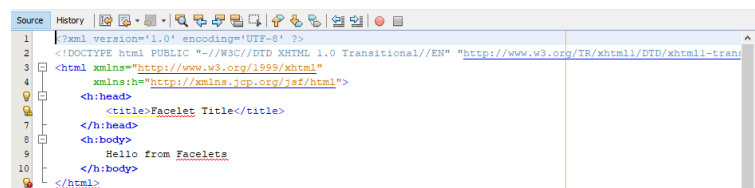
```

33. Selanjutnya, Expand bagian Web Pages, lalu kalian akan melihat file 'index.xhtml'. Kemudian klik 2 kali pada File tersebut



Gambar 5. 34 File Index.xhtml pada folder Web pages

34. Lalu akan tampil, tampilan seperti berikut



Gambar 5. 35 Tampilan file index.xhtml

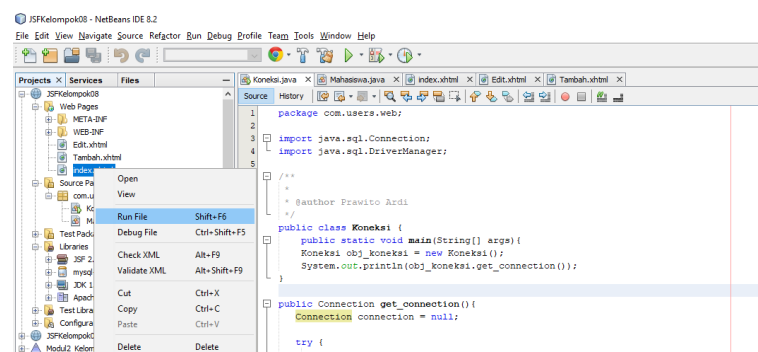
35. Lalu copy kodingan yang ada pada file 'Index.txt' ke dalam file 'index.xhtml' tersebut. Berikut adalah sourcenya.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>JSF RSBK</title>
  </h:head>
  <h:body>

    <center>
      <h2>Daftar Mahasiswa</h2>
      <h:form>
        <h:dataTable
value="#{mahasiswa.get_all_mahasiswa}" var="maha"
width="800px">
          <h:outputText value="#{mahasiswa.NAMA}"/>
          <h:column>
            <f:facet name="header">NIM</f:facet>
            <h:outputText
value="#{maha.NIM}"></h:outputText>
          </h:column>
          <h:column>
            <f:facet name="header">Nama</f:facet>
            <h:outputText
value="#{maha.NAMA}"></h:outputText>
          </h:column>
          <h:column>
            <f:facet
name="header">Penjurusan</f:facet>
            <h:outputText
value="#{maha.PENJURUSAN}"></h:outputText>
          </h:column>
          <h:column>
            <f:facet name="header">Edit</f:facet>
            <h:commandButton value="Edit"
action="#{mahasiswa.Edit_Mahasiswa}">
              <f:param name="action"
value="#{maha.NIM}" />
            </h:commandButton>
          </h:column>
          <h:column>
            <f:facet name="header">Delete</f:facet>
            <h:commandButton value="Delete"
action="#{mahasiswa.Delete_Mahasiswa}">
              <f:param name="action"
value="#{maha.NIM}" />
            </h:commandButton>
          </h:column>
        </h:dataTable>
        <br></br>
      </h:form>
    </center>
  </h:body>
</html>
```

[illegible]

36. Kemudian klik kanan pada file 'Index.xhtml' lalu pilih Run File

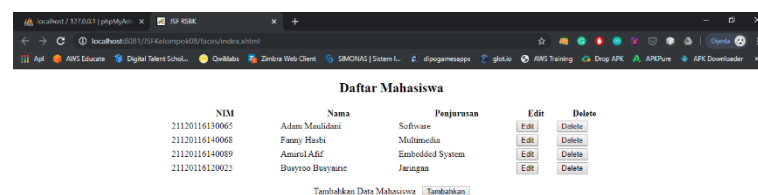


Gambar 5. 36 Run file index.xhtml

Jika terjadi suatu kesalahan / error “port is used” gunakan perintah

```
Run cmd as administrator
netstat -ano | findstr :8080
taskkill /PID 3748 /F
# 3748 adalah process id
```

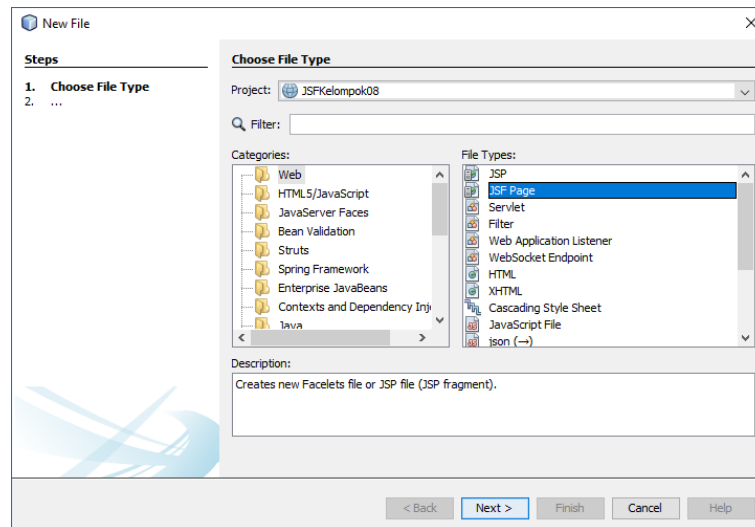
37. Maka, akan muncul tampilan sebagai berikut pada browser kalian:



Gambar 5. 37 Tampilan project pada web browser

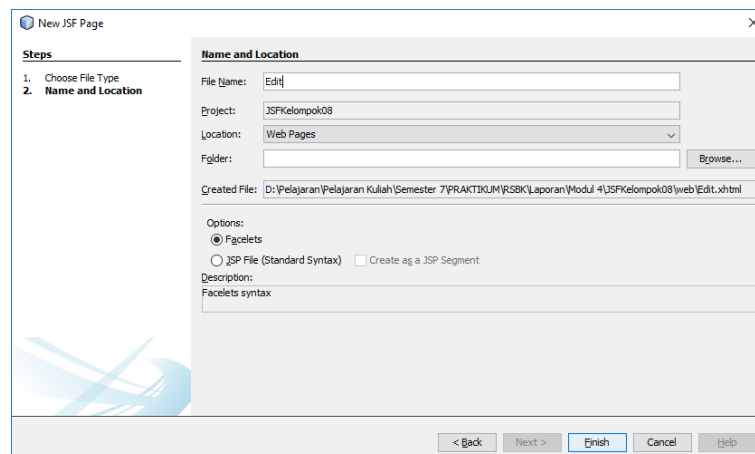
38. Pada tampilan tersebut, button Edit dan Tambahkan, belum dapat digunakan, maka kita lanjut ke langkah selanjutnya

39. Kembali lagi ke Netbeans kalian, lalu klik kanan pada Folder Web Pages, lalu pilih New, dan pilih JSF Page

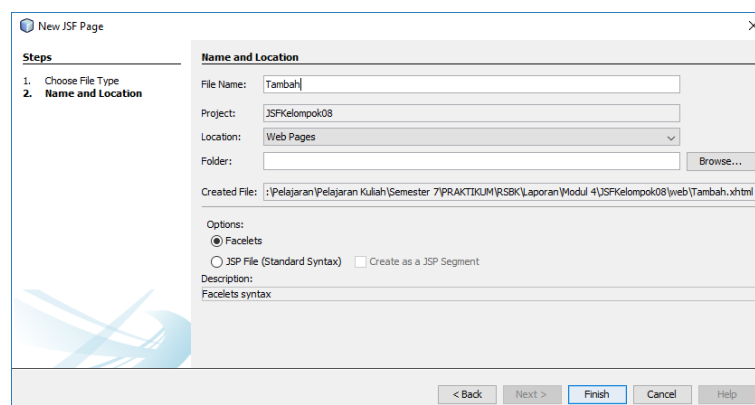


Gambar 5. 38 Langkah membuat file JSF Page pada project

40. Buat 2 buah File JSF Page, dengan nama 'Edit' dan 'Tambah'



Gambar 5. 39 Membuat file JSF Page file name Edit



Gambar 5. 40 Membuat file JSF Page file name Tambah



41. Kemudian copy kodingan yang ada pada file 'Edit.txt' ke dalam file 'Edit.xhtml'.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head>
<title>JSF RSBK</title>
</h:head>
<h:body>
<center>
<h:form>
<h1>Edit Data Mahasiswa</h1>

<table>
<tr>
<td>NIM :</td>
<td><h:inputText value="#{EditMahasiswa.NIM}"
></h:inputText></td>
</tr>
<tr>
<td>Nama Mahasiswa :</td>
<td><h:inputText
value="#{EditMahasiswa.NAMA}" ></h:inputText></td>
</tr>
<tr>
<td>Penjurusan :</td>
<td><h:inputText
value="#{EditMahasiswa.PENJURUSAN}"></h:inputText></td>
</tr>
</table>
<br></br>
<h:commandButton value="Update"
action="#{EditMahasiswa.Update_Mahasiswa}">
<f:param name="Update_NIM" value="#{EditMahasiswa.NIM}"
/></h:commandButton>
<br></br>
<br></br>
</h:form>
<h:form>
<h:commandButton value="Kembali"
action="index.xhtml"></h:commandButton>
</h:form>

</center>
</h:body>
</html>
```

42. Dan juga copy kodingan yang ada pada file 'Tambah.txt' ke dalam file 'Tambah.xhtml'.

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>JSF RSBK</title>
  </h:head>
  <h:body>
    <center>
      <h2>Tambah Data Mahasiswa</h2>

      <h:form>
        <table>
          <tr>
            <td>NIM :</td>
            <td><h:inputText          value="#{mahasiswa.NIM}"
></h:inputText></td>
          </tr>
          <tr>
            <td>Nama Mahasiswa :</td>
            <td><h:inputText          value="#{mahasiswa.NAMA}"
></h:inputText></td>
          </tr>
          <tr>
            <td>Penjurusan :</td>
            <td><h:inputText
value="#{mahasiswa.PENJURUSAN}"></h:inputText></td>
          </tr>
        </table>
        <br></br>
        <h:commandButton          value="Tambahkan"
action="#{mahasiswa.Tambah_Mahasiswa}"></h:commandButton>
        <br></br>
        <br></br>
      </h:form>
      <h:form>
        <h:commandButton          value="Kembali"
action="index.xhtml"></h:commandButton>
      </h:form>
    </center>

  </h:body>
</html>

```

43. Kemudian, coba kalian lakukan lagi Run File 'index.xhtml'. Dan, coba tekan button Edit atau Tambahkan. Maka button tersebut kini dapat digunakan

## 5.4 Hasil dan Analisa Pembahasan

Pada Praktikum Java Server Pages ini adalah membuat sebuah program sederhana yang berbasis JSF file, dimana pada file ini akan membuat suatu tampilan dengan format xhtml. Pada project ini menggunakan database MySQL yang digunakan untuk menyimpan data yang di dihubungkan dengan program yang dibuat menggunakan class Koneksi.java. Kemudian terdapat 2 fungsi yang dapat dilakukan pada program ini yaitu Edit, yang digunakan untuk mengubah data yang ada, dan fungsi Tambah yang digunakan untuk menambah 1 data baru. Dan berikut adalah hasil percobaan.

### a. Class Koneksi.java

```
package com.users.web;

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

/**
 *
 * @author Prawito
 */
public class Koneksi {
    public static void main(String[] args){
        Koneksi obj_koneksi = new Koneksi();
        System.out.println(obj_koneksi.get_connection());
    }

    public Connection get_connection(){
        Connection connection = null;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/kel08", "kel08", "kel08");
        } catch (Exception e) {
            System.out.println(e);
        }
        return connection;
    }
}
```

File Koneksi.java ini digunakan untuk menghubungkan database MySQL pada program, dimana terdapat fungsi crud data pada program yang datanya akan disimpan database. Dan terdapat syntax("jdbc:mysql://localhost:3306/kel08", "kel08", "kel08") yang harus disesuaikan dengan database

yang akan dihubungkan serta username dan password yang digunakan dalam database ini.

b. Class Mahasiswa

Kemudian pada *project* ini terdapat pula *file java managed bean* yaitu mahasiswa.java yang digunakan sebagai *class* yang terdiri dari beberapa method untuk melakukan fungsi standar aplikasi yaitu CRUD (*Create Delete dan Update*).

Pada file class mahasiswa.java ini berisi source yang berhubungan dengan fungsi mulai dari tambah, delete, update, hingga edit. Terdapat setter dan getter yang digunakan untuk menerima dan mengembalikan data Nama, NIM, Jurusan yang ada pada program ketika memberikan input atau menerima inputan.

Pada method Edit\_Mahasiswa terdapat Query (`"select * from mahasiswa where NIM="+Field_NIM);` yaitu mengambil data mahasiswa yang akan di edit berdasarkan NIM yang digunakan. Terdapat beberapa objek `obj_Mahasiswa.setNIM (rs.getString("NIM"));` dan beberapa method lainnya sesuai dengan data yang digunakan akan mengambil data yang di inputkan yang kemudian akan masuk dalam database sesuai dengan data baru yang di inputkan yang kemudian akan ditampilkan pada halaman Edit.xhtml.

Selanjutnya method Delete\_Mahasiswa yang digunakan untuk menghapus data berdasarkan NIM dengan penggunaan query (`"delete from mahasiswa where NIM=?"`). Fungsi ini kemudian akan terhubung melalui syntax `ps.executeUpdate();` yang nantinya data ini akan hilang dari database serta akan return untuk kembali ke halaman index.xhtml dengan kondisi data yang telah terhapus.

Method Update\_Mahasiswa ini digunakan untuk merubah data yang sudah ada dengan menggunakan query (`"update mahasiswa set NIM=?, Nama=?, Jurusan=? where NIM=?"`), dengan mengganti data yang ada kemudian method ini akan mengeksekusi data yang telah diinputkan untuk ditampilkan pada halaman index.xhtml.

Kemudian method Tambah\_Mahasiswa, dimana method ini berfungsi ketika button tambah di klik setelah memberikan inputan pada text field yang ada yang kemudian method ini akan menjalankan query (`"insert into mahasiswa (NIM,`

Nama, Penjurusan) value('"+NIM+"', '"+NAMA+"', '"+PENJURUSAN+"'))")

yaitu memberikan data berupa NIM, NAMA dan PENJURUSAN pada database table mahasiswa. Dan berikut adalah source codenya.

```
package com.users.web;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Map;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
/**
 *
 * @author Prawito
 */
@ManagedBean
@RequestScoped
public class Mahasiswa {
    /**
     * Creates a new instance of Mahasiswa
     */
    private String NIM;
    public void setNIM(String NIM) {
        this.NIM = NIM;
    }
    public String getNIM() {
        return NIM;
    }
    private String NAMA;
    public void setNAMA(String NAMA) {
        this.NAMA = NAMA;
    }
    public String getNAMA() {
        return NAMA;
    }
    private String PENJURUSAN;
    public void setPENJURUSAN(String PENJURUSAN) {
        this.PENJURUSAN = PENJURUSAN;
    }
    public String getPENJURUSAN() {
        return PENJURUSAN;
    }
    private Map<String, Object> sessionMap =
FacesContext.getCurrentInstance().getExternalContext().getSessionMap();
    public String Edit_Mahasiswa(){
        FacesContext fc = FacesContext.getCurrentInstance();
        Map<String, String> params =
fc.getExternalContext().getRequestParameterMap();
        String Field_NIM = params.get("action");
        try {
            Koneksi obj_koneksi = new Koneksi();
```

```

        Connection connection = obj_koneksi.get_connection();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery("select * from mahasiswa
where NIM="+Field_NIM);
        Mahasiswa obj_Mahasiswa = new Mahasiswa();
        rs.next();
        obj_Mahasiswa.setNIM(rs.getString("NIM"));
        obj_Mahasiswa.setNAMA(rs.getString("Nama"));

obj_Mahasiswa.setPENJURUSAN(rs.getString("Penjurusan"));
        sessionMap.put("EditMahasiswa", obj_Mahasiswa);
    } catch (Exception e) {
        System.out.println(e);
    }
    return "/Edit.xhtml?faces-redirect=true";    }

    public String Delete_Mahasiswa() {
        FacesContext fc = FacesContext.getCurrentInstance();
        Map<String,String> params =
fc.getExternalContext().getRequestParameterMap();
        String Field_NIM = params.get("action");
        try {
            Koneksi obj_koneksi = new Koneksi();
            Connection connection = obj_koneksi.get_connection();
            PreparedStatement ps =
connection.prepareStatement("delete from mahasiswa where NIM=?");
            ps.setString(1, Field_NIM);
            System.out.println(ps);
            ps.executeUpdate();
        } catch (Exception e) {
            System.out.println(e);
        }
        return "/index.xhtml?faces-redirect=true";
    }

    public String Update_Mahasiswa() {
        FacesContext fc = FacesContext.getCurrentInstance();
        Map<String,String> params =
fc.getExternalContext().getRequestParameterMap();
        String Update_NIM= params.get("Update_NIM");
        try {
            Koneksi obj_koneksi = new Koneksi();
            Connection connection = obj_koneksi.get_connection();
            PreparedStatement ps =
connection.prepareStatement("update mahasiswa set NIM=?, Nama=?,
Penjurusan=? where NIM=?");
            ps.setString(1, NIM);
            ps.setString(2, NAMA);
            ps.setString(3, PENJURUSAN);
            ps.setString(4, Update_NIM);
            System.out.println(ps);
            ps.executeUpdate();
        } catch (Exception e) {
            System.out.println(e);
        }
        return "/index.xhtml?faces-redirect=true";
    }

    public ArrayList getGet_all_mahasiswa() throws Exception{

```

```

        ArrayList list_of_mahasiswa=new ArrayList();
        Connection connection=null;
        try {
            Koneksi obj_koneksi = new Koneksi();
            connection = obj_koneksi.get_connection();
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery("Select * from
mahasiswa");
            while(rs.next()){
                Mahasiswa obj_Mahasiswa = new Mahasiswa();
                obj_Mahasiswa.setNIM(rs.getString("NIM"));
                obj_Mahasiswa.setNAMA(rs.getString("Nama"));

obj_Mahasiswa.setPENJURUSAN(rs.getString("Penjurusan"));
                list_of_mahasiswa.add(obj_Mahasiswa);
            }
        } catch (Exception e) {
            System.out.println(e);
        }finally{
            if(connection!=null){
                connection.close();
            }
        }
        return list_of_mahasiswa;
    }
    public String Tambah_Mahasiswa() {
        try {
            Connection connection=null;
            Koneksi obj_koneksi = new Koneksi();
            connection = obj_koneksi.get_connection();
            PreparedStatement
ps=connection.prepareStatement("insert into mahasiswa(NIM, Nama,
Penjurusan) value('"+NIM+"', '"+NAMA+"', '"+PENJURUSAN+"')");
            ps.executeUpdate();
        } catch (Exception e) {
            System.out.println(e);
        }
        return "/index.xhtml?faces-redirect=true";
    }
    public Mahasiswa() {}
}

```

### c. JSF Managed Bean

Selain itu untuk membuat suatu tampilan maka diperlukan dengan adanya file JSF Page dalam hal ini terdapat 3 file JSF *Page* yaitu *index.xhtml*, *edit.xhtml* dan *tambah.xhtml*.

#### 1. Edit.xhtml

*File edit.xhtml* memuat suatu tabel yang akan menampilkan data mahasiswa dan juga terdapat beberapa *button* dengan fungsi untuk tambah, *update* dan *delete*.

Terdapat perintah `<td><h:inputText value="#"#{EditMahasiswa.NIM}"></h:inputText></td>` digunakan untuk membuat suatu form yang berisi value dari data sebelumnya yang kemudian akan di edit sesuai dengan value yang baru atau sesuai dengan parameter yang akan ditampilkan. Kemudian terdapat `<h:commandButton value="Update" action="#"#{EditMahasiswa.Update_Mahasiswa}">` digunakan untuk button yang nantinya akan mengeksekusi method dari `Update_Mahasiswa` yang mana pada *method* tersebut berisi *query select from mahasiswa* sehingga data yang akan ditampilkan sesuai dengan data yang ada pada tabel mahasiswa di *database*.

Pada *file edit.xhtml* terdapat *textfield* yang digunakan untuk *inputan* teks dengan beberapa variabel sesuai dengan kolom yang ada pada *database*. Adanya perintah `<td><h:inputText value="#"#{EditMahasiswa.NIM}"></h:inputText></td>` berfungsi untuk memanggil *value* dari variabel yang akan *diedit*, pada perintah tersebut berarti input teks tersebut akan diisi dengan NIM yang akan *diedit*, serta method `Edit_Mahasiswa` akan dipanggil yang akan menjalankan fungsi *query select* dengan diikuti *where NIM*. *Button* `<h:commandButton value="Update" action="#"#{EditMahasiswa.Update_Mahasiswa}">` akan melakukan suatu *trigger* untuk menjalankan fungsi dari *method* `Update_Mahasiswa` yang ada pada *class Mahasiswa.java* yang mana pada *method* tersebut terdapat *query* untuk melakukan *update* pada *database*. Dalam melakukan *update* diperlukan dengan adanya suatu parameter yang *unique* atau tidak dapat sama nilainya jadi yang akan dijadikan sebagai parameter untuk melakukan *update* yaitu pada NIM.

## 2. Index.xhtml

*File index.xhtml* memuat suatu tabel yang akan menampilkan data mahasiswa dan juga terdapat beberapa *button* dengan fungsi untuk tambah, *update* dan *delete*. Terdapat perintah `<h:dataTable value="#"#{mahasiswa.get_all_mahasiswa}" var="maha" width="800px">` digunakan untuk membuat suatu tabel yang mana *value* pada tabel tersebut akan diisi dengan nilai dari *class mahasiswa* dan menjalankan *method* `get_all_mahasiswa`, pada *method* tersebut berisi *query select*

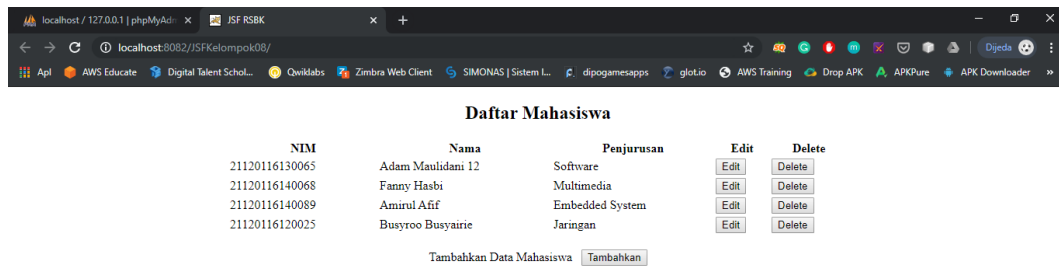


from mahasiswa sehingga data yang akan ditampilkan sesuai dengan data yang ada pada tabel mahasiswa di *database*. Data pada kolom akan ditampilkan dengan cara `<h:outputText value="#{mahasiswa.NAMA}"/>` sesuai dengan parameter yang akan ditampilkan. Pada file `index.xhtml` terdapat syntax untuk fungsi button tambah dan edit, serta delete, dimana `<h:commandButton value="Tambahkan" action="Tambah.xhtml"></h:commandButton>` ini adalah button edit yang akan menjalankan method `Edit_Mahasiswa` pada class `mahasiswa.java` dan syntax `<h:commandButton value="Delete" action="#{mahasiswa.Delete_Mahasiswa}">` digunakan ketika button delete di klik yang akan menjalankan method `Delete_Mahasiswa` pada class `mahasiswa.java` juga. Terakhir adalah button edit yang menggunakan syntax `<h:commandButton value="Edit" action="#{mahasiswa.Edit_Mahasiswa}">` yang menjalankan fungsi atau method `Edit_Mahasiswa` pada class `mahasiswa` ketika button ini digunakan.

### 3. Tambah.xhtml

Selain itu terdapat juga *file JSF page* yang digunakan untuk melakukan tambah data yaitu `tambah.xhtml` pada *file* tersebut terdapat parameter data yang akan dijadikan sebagai inputan. Adanya *input* teks digunakan sebagai *inputan* data yang akan dimasukkan kedalam *database* `<td><h:inputText value="#{mahasiswa.NIM}" ></h:inputText></td>` potongan source code ini berfungsi untuk menambahkan data pada variabel `NIM` yang sudah di integrasikan dengan *database*. Kemudian terdapat juga perintah `<h:commandButton value="Tambahkan" action="#{mahasiswa.Tambah_Mahasiswa}"></h:commandButton>` untuk membuat aksi pada *button* tambah, sehingga ketika *button* ini di klik maka akan menjalankan *method* `Tambah_Mahasiswa` pada *class* `mahasiswa` yang didalam *method* tersebut terdapat *query* untuk *insert* ke *database*.

Untuk memastikan apakah *file – file* yang dibuat sudah benar maka dapat dilakukan dengan menjalankan `file index.xhtml`, ketika *file index.xhtml* dijalankan maka akan menampilkan data tabel mahasiswa dengan beberapa *button* yang akan menjalankan fungsi *CRUD*. Berikut adalah tampilan dari halaman `index.xhtml`.

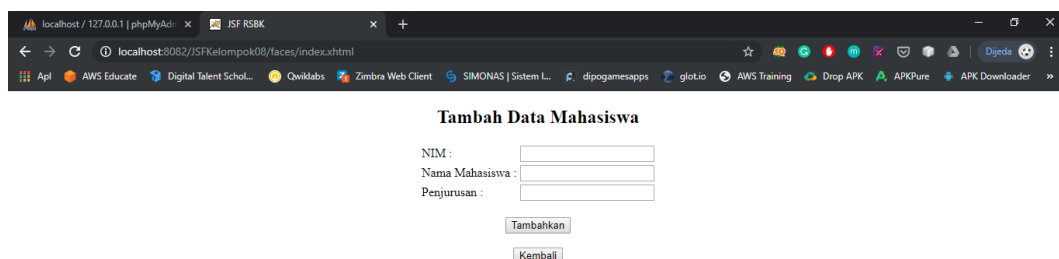


Gambar 5. 41 Tampilan Halaman Index.xhtml

Dalam halaman index.xhtml ini terdapat beberapa button dan juga tampilan data mulai dari NIM, Nama, Penjurusan dan button edit dan tambah serta button delete. Button Tambahkan ini digunakan untuk menambahkan data dengan memberikan input pada form yang tersedia, kemudian button edit digunakan untuk mengganti value yang sudah ada sebelumnya serta button delete yang digunakan untuk menghapus data.

a. Button Tambahkan

Ketika ingin menambahkan daftar mahasiswa maka dapat dilakukan dengan cara menekan pada *button* Tambahkan maka akan muncul halaman dengan 3 *input* teks yaitu NIM, Nama dan Penjurusan serta button Tambahkan dan Kembali. Kemudian ketika *button* Tambahkan ditekan maka akan menjalankan fungsi method dari Tambah\_Mahasiswa dari class mahasiswa.java yang berisi *query insert* ke *database*.



Gambar 5. 42 Tampilan Halaman Tambah Data

**Tambah Data Mahasiswa**

NIM :

Nama Mahasiswa :

Penjurusan :

Gambar 5. 43 Tampilan Halaman Tambah Data Isi Data Form

**Daftar Mahasiswa**

NIM	Nama	Penjurusan	Edit	Delete
21120116130065	Adam Maulidani 12	Software	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
21120116140068	Fanny Hasbi	Multimedia	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
21120116140089	Amirul Atif	Embedded System	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
21120116120025	Busyroo Busyairie	Jaringan	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
21120116120019	Prawito	Software	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

Tambahkan Data Mahasiswa

Gambar 5. 44 Tampilan Hasil Tambah Data Berhasil

Dan ini adalah tampilan dimana data yang di tambahkan berhasil disimpan dalam database dan ditampilkan pada halaman index.xhtmlml setelah button Tambahkan di klik dengan kondisi form yang ada telah di isi dengan nilai.

#### b. Button Edit

Selanjutnya ketika ingin melakukan *edit* data maka dapat dilakukan dengan menekan tombol *edit* pada bagian kolom edit data yang ingin diubah. Kemudian ketika sudah ditekan tombol *edit* tersebut maka akan berpindah halaman ke *form edit* data. Terdapat form NIM, Nama Mahasiswa dan Penjurusan yang akan diubah (secara otomatis data akan terisikan pada ke tiga input teks sesuai dengan data yang akan di *edit*). *Button* tersebut akan menjalankan method *Edit\_Mahasiswa* yang ada pada class mahasiswa.java.

**Edit Data Mahasiswa**

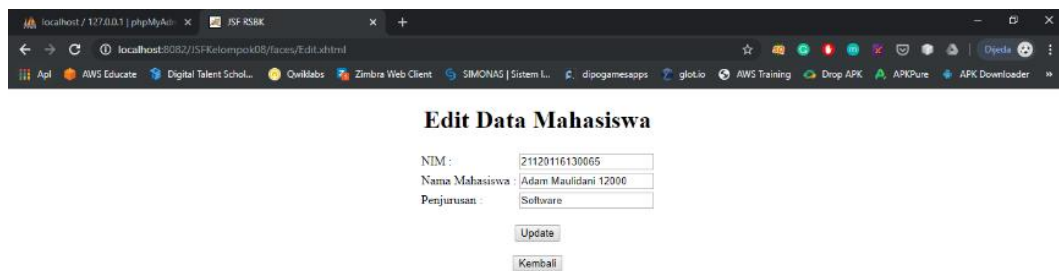
NIM :

Nama Mahasiswa :

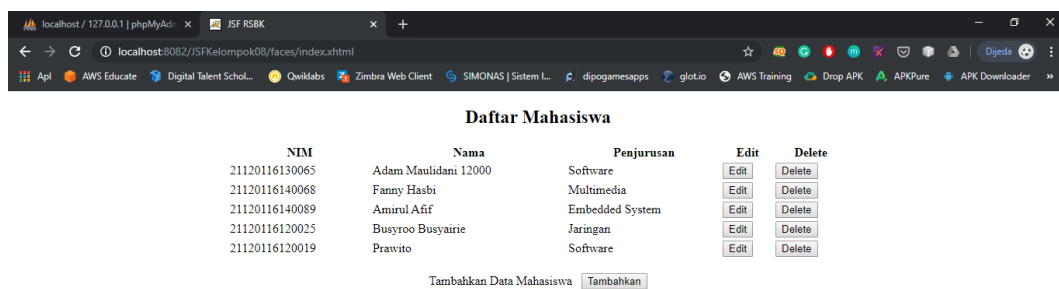
Penjurusan :

Gambar 5. 45 Tampilan Halaman Edit Data

Dan ini adalah kondisi dimana ketika button edit diklik maka data akan muncul pada form ini berdasarkan NIM, dan terdapat 3 form yang sudah terisi dengan data. Kemudian mencoba untuk mengganti value atau data dari Nama Mahasiswa, setelah itu button Update digunakan untuk mengeksekusi value data yang ada pada form dengan menjalankan method Update\_Mahasiswa pada class mahasiswa.java.



Gambar 5. 46 Tampilan antarmuka halaman *edit*

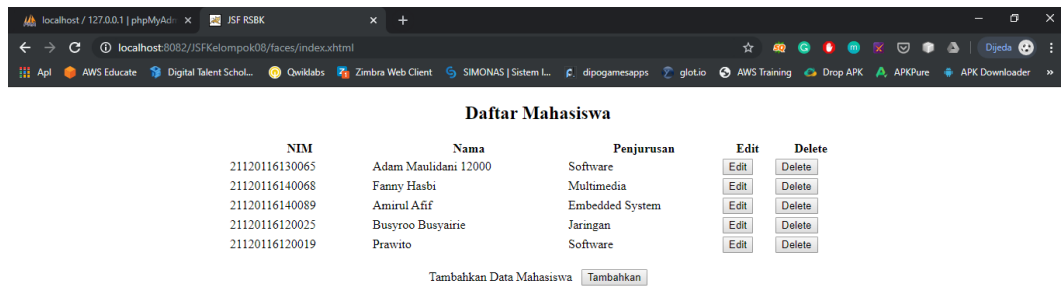


Gambar 5. 47 Halaman Berhasil Edit Data

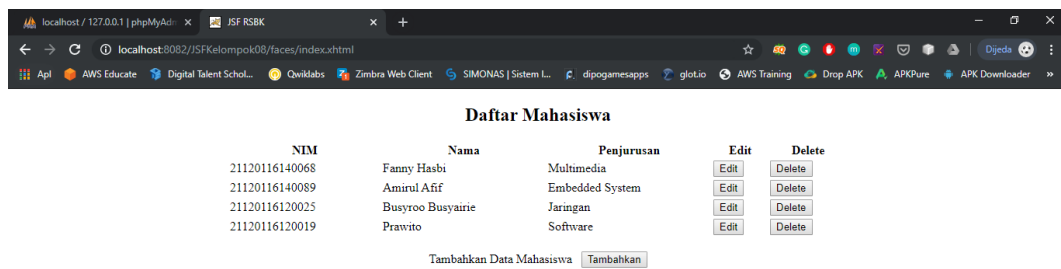
Selanjutnya setelah menambahkan data dan di klik *button update* maka data akan diubah dan simpan ke database dan muncul pada tabel di halaman *index*.

### c. Button Delete

Delete dapat dilakukan dengan menekan *button* yang ada pada bagian kolom *delete* pada tabel. Ketika *button* tersebut ditekan maka akan menjalankan fungsi *query delete* pada *class* mahasiswa.java method Delete\_Mahasiswa, parameter yang dijadikan untuk proses *delete* adalah NIM dan ini adalah syntaxnya ("delete from mahasiswa where NIM=?". Setelah *delete* dilakukan maka data akan berubah dan disimpan kedalam database serta data yang tertampil pada index.html akan berubah. Dan berikut adalah fungsi button delete.



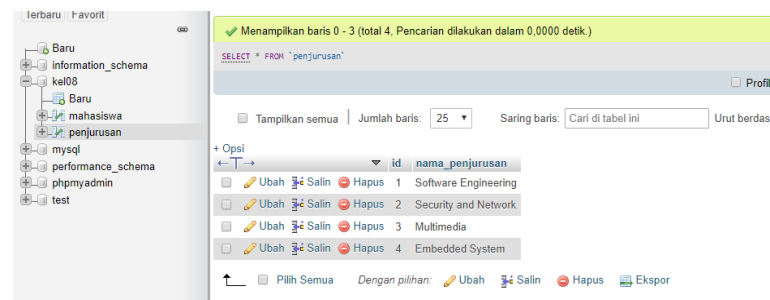
Gambar 5. 48 Tampilan Halaman Index



Gambar 5. 49 Tampilan Halaman Berhasil Hapus Data

## 5.5 Tugas

Pada praktikum ini, ditugaskan untuk membuat atau menambahkan table penjurusan pada database yang digunakan pada pembuatan project sebelumnya. Tabel penjurusan ini memiliki atribut berupa id dan nama penjurusan yang digunakan untuk menyimpan data penjurusan dan memberikan value jika misal penjurusan Software Engineering memiliki id 1 dan seterusnya hingga penjurusan 4. Dan berikut adalah table penjurusan dan yang ditambahkan pada database.



Gambar 5. 50 Tampilan Database Table Penjurusan

Selanjutnya, dalam program sederhana ini menggunakan 1 table lagi yaitu mahasiswa, dimana pada table mahasiswa memiliki 3 atribut yaitu NIM sebagai primary key, Nama, dan id\_penjurusan sebagai foreign key dengan table penjurusan yang sebelumnya kita buat, jadi ketika mengimpitkan angka 1 – 4 pada form penjurusan di tampilan program, maka yang akan tampil adalah nama jurusan yang sesuai dengan id yang telah ditentukan. Dan berikut adalah table mahasiswa.



Gambar 5. 51 Tampilan Database Table Mahasiswa

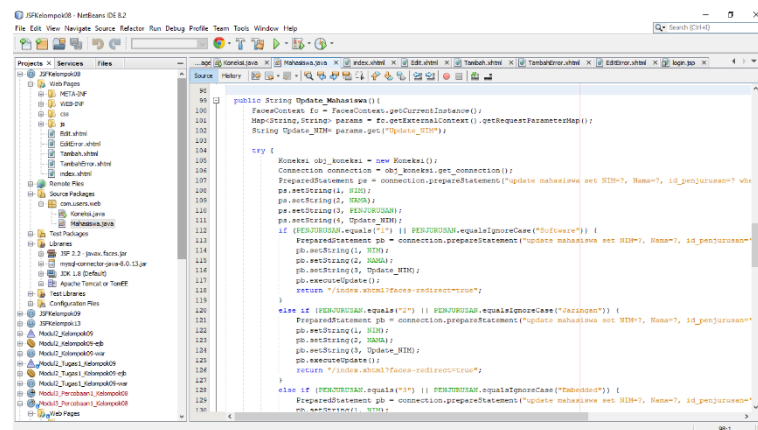
Pada tugas ini terdapat beberapa kriteria, dimana ketika nilai atau value data yang ada di penjurusan dan dimasukan pada form tambah atau update data maka akan menampilkan pesan pada tampilan yaitu Data Penjurusan Tidak Ditemukan. Dan untuk mempercantik tampilan, digunakan beberapa file css dan beberapa

atribut yang ada pada internet, dan pada tampilan atau program ini digunakan tema dari Tokopedia, mulai dari warna dan beberapa atribut gambar lainnya.

Untuk css yang digunakan yaitu bootstrap yaitu dengan cara mendownload file bootstrap kemudian membuat folder dengan nama resource, setelah itu letakkan file bootstrap tersebut kedalam folder resource. Terdapat 5 file JSF Page dan 2 file JSF managed bean. Pada dasarnya sama seperti ketika praktikum untuk menjalankan fungsi – fungsi CRUD terdapat pada file JSF managed bean selain itu pada file tersebut juga terdapat setter dan getter, beberapa parameter variabel yang akan diolah sesuai dengan method yang ada. Dan berikut adalah hasil dari percobaan.

#### 1. Managed Bean Mahasiswa.java

Pada file mahasiswa.java ini terdapat beberapa tambahan atau perubahan yang dilakukan mulai dari perubahan atribut nama\_penjurusan menjadi id\_penjurusan, kemudian terjadi banyak perubahan adalah pada method Update\_Mahasiswa, dimana pada method ini ditambahkan beberapa pengkondisian ketika input yang diberikan pada form penjurusan bernilai 1 hingga 4 dan kondisi ketika diisi nilai selain angka tersebut. Dan berikut adalah sreenshoot source codenya.



Gambar 5. 52 Source Code class mahasiswa.java

Kemudian terjadi perubahan atau penambahan pada method Tambah\_Mahasiswa, dimana pada method ini juga diberikan pengkondisian ketika form penjurusan itu diisi dengan nilai 1 hingga 4 dan kondisi ketika form tersebut diisi dengan nilai selain dari pada nilai 1 hingga 4. Contoh pengkondisian pada method Tambah\_Mahasiswa.

```

if (PENJURUSAN.equals("1")
PENJURUSAN.equalsIgnoreCase("Security and Network")) {
    PreparedStatement pl =
connection.prepareStatement("insert into mahasiswa (NIM, Nama,
id_penjurusan) value('\" + NIM + '\", '\" + NAMA + '\", '1')");
    pl.executeUpdate();
    return "/index.xhtml?faces-redirect=true";
}

```

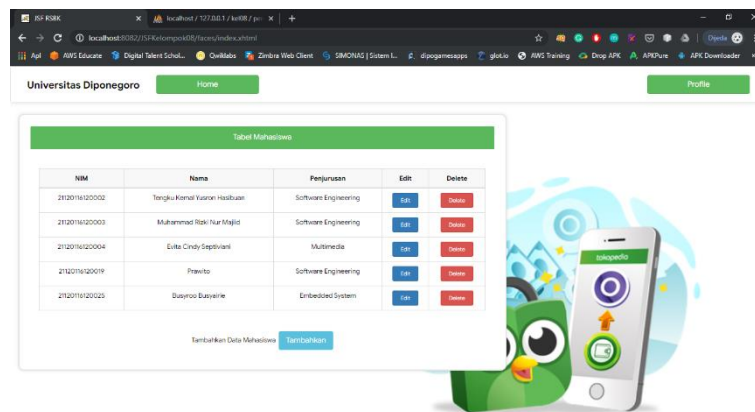
Dimana pada source code tersebut, berisi apabila input adalah nilai 1 maka pada tampilan data mahasiswa kolom penjurusan adalah Security dan Network dan data ini akan masuk pada database tabel mahasiswa yang kemudian akan me return tampilan dari index.xhtml.

## 2. Managed Bean koneksi.java

File Koneksi.java ini digunakan untuk menghubungkan database MySQL pada program, dimana terdapat fungsi crud data pada program yang datanya akan disimpan database. Dan terdapat syntax ("jdbc:mysql://localhost:3306/ke108", "ke108", "ke108") yang harus disesuaikan dengan database yang akan dihubungkan serta username dan password yang digunakan dalam database ini.

## 3. Index.xhtml

Untuk file JSF *index.xhtml* yaitu merupakan tampilan awal ketika aplikasi ini dijalankan, pada tampilan awal ini terdapat bebrapa *button* yang digunakan untuk fungsi crud dan beberapa button lainnya serta data dari table mahasiswa mulai dari NIM, Nama dan Penjurusan. Berikut adalah tampilannya.



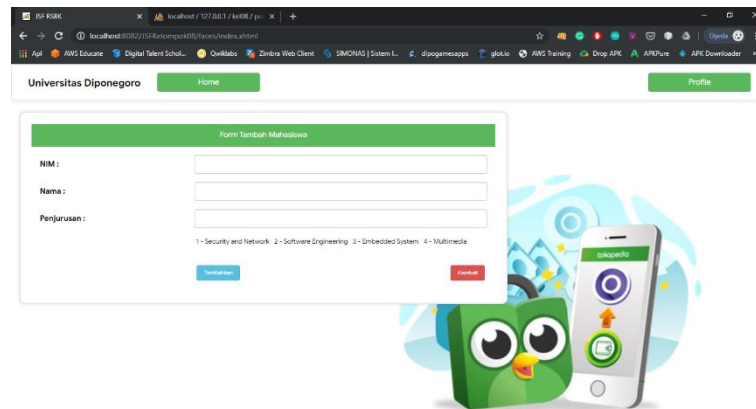
Gambar 5. 53 Tampilan Halaman Index.xhtml



Kemudian button Tambahkan ini digunakan untuk menambahkan data, dimana ketika button ini di klik akan menampilkan halaman dengan form berupa NIM, Nama dan Id\_Penjurusan. Selanjutnya button edit digunakan untuk mengubah data yang telah ada, ketika button edit ini di klik maka akan menampilkan halaman dengan form yang telah terisi data sebelumnya dan hanya perlu mengganti data jika ingin terjadi perubahan pada data, serta button delete digunakan untuk menghapus data.

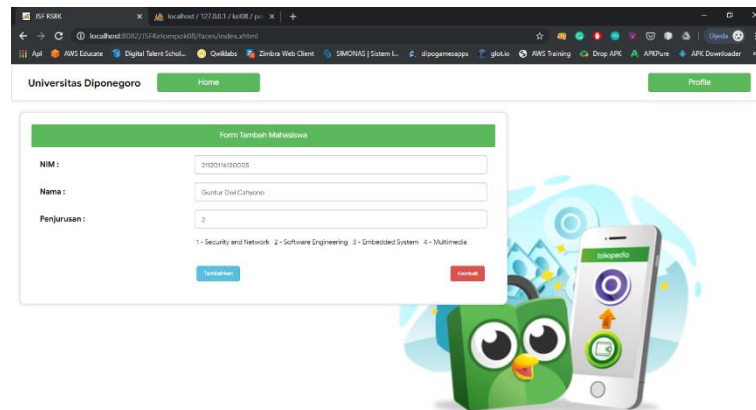
#### 4. Tambah.xhtml

Pada halaman tambah.xhtml ini terdapat beberapa form yang digunakan untuk data yang akan diisikan, dimana terdapat form NIM, Nama dan Jurusan yang diisi dengan id jurusan sesuai dengan keterangan yang ada yaitu 1 untuk Security dan Network dan seterusnya. Kemudian terdapat button Tambahkan yang digunakan untuk mengeksekusi data yang ada pada form agar tersimpan pada database dan ditampilkan pada halaman index. Serta button Kembali yang digunakan untuk kembali ke halaman sebelumnya.



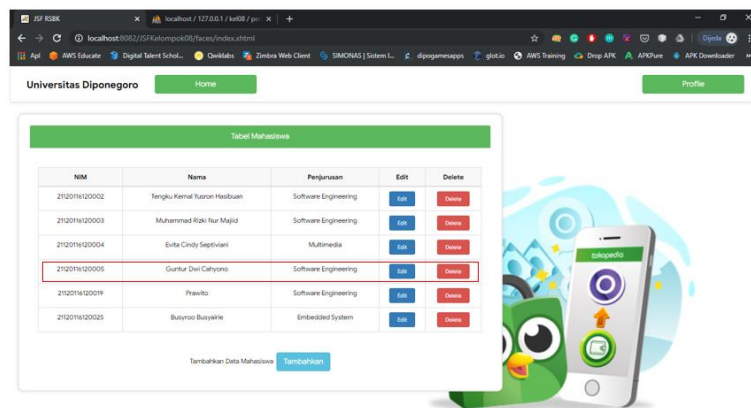
Gambar 5. 54 Halaman Tampilan Tambah Data

Dan ini adalah form kosong ketika button tambahkan pada halaman index.xhtml di klik, terdapat form NIM, Nama dan Jurusan.



Gambar 5. 55 Halaman Tampilan Proses Tambah Data

Selanjutnya adalah mencoba untuk memberikan input atau value berupa NIM : 21120116120005, Nama : “Guntur Dwi Cahyono”, dan Jurusan : 2 (Software Engineering)



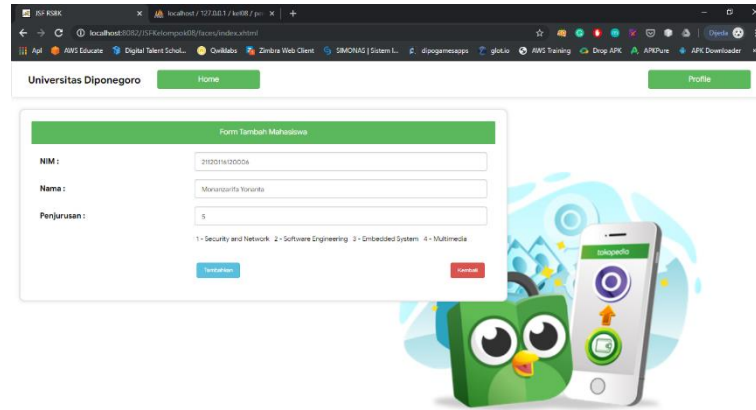
Gambar 5. 56 Halaman Tampilan Berhasil Tambah Data

Dan ini adalah hasil dari tambah data, dimana terdapat atau tertampil data yang baru saja ditambahkan pada form/halaman tambah.xhtml.

## 5. TambahError.xhtmll

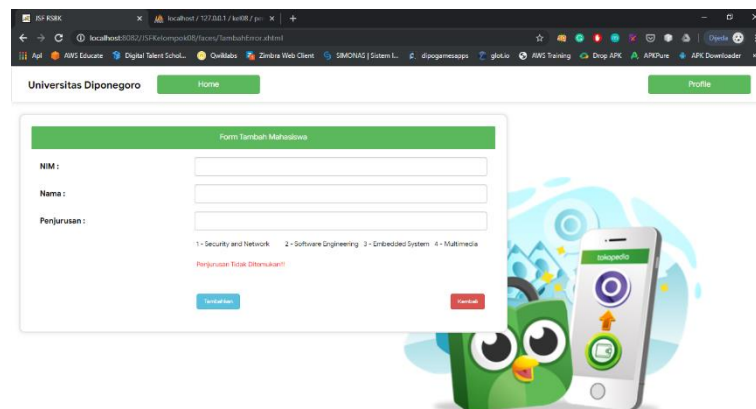
Pada file ini hanya digunakan untuk memberikan keterangan apabila pada form jurusan diisi dengan nilai bukan angka 1 hingga 4. Secara fungsional source code halaman ini sama dengan tambah.xhtmll hanya saja ditambahkan beberapa source code keterangan untuk menampilkan text “ Jurusan Tidak Ditemukan ”. Jadi ketika input diberikan 1 hingga 4 maka method Tambah\_Mahasiswa pada class mahasiswa akan mengeksekusi pengkondisiannya dan me return halaman index.xhtmll dengan data telah masuk ke database, kemudian apabila input bukan 1

hingga 5 maka pengkondisian yang di eksekusi akan me return halaman TambahError.xhtml. Dan berikut adalah hasilnya.



Gambar 5. 57 Halaman Tampilan Tambah Data

Mencoba untuk menambahkan data dengan nilai penjurusan adalah 5, serta NIM: 21120116120006 dan Nama: “Monanzarifa Yonanta”.



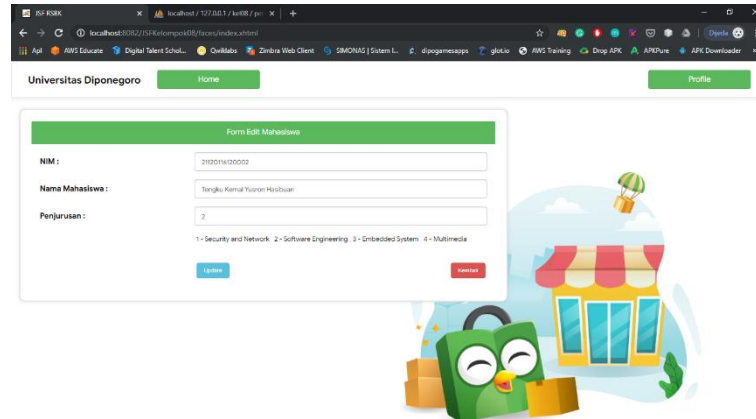
Gambar 5. 58 Halaman Tampilan Data Tidak Berhasil Ditambah

Pada gambar menampilkan text keterangan berupa “Penjurusan Tidak Ditemukan” serta halaman yang ditampilkan adalah TambahError.xhtml.

## 6. Edit.xhtmll

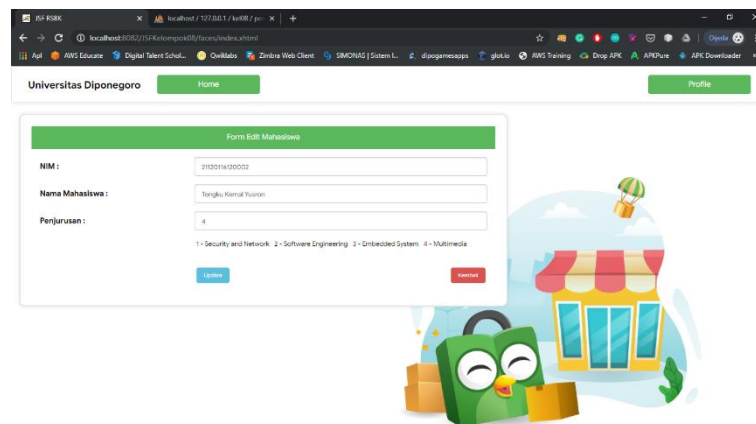
Pada halaman edit.xhtmll ini terdapat beberapa form yang digunakan untuk menampilkan data mulai dari NIM, Nama dan Penjurusan. Pada halaman ini form sudah terisi dengan data dan hal yang harus dilakukan adalah merubah data jika ingin terjadi perubahan pada database atau tabel data. Kemudian terdapat button Update yang digunakan untuk mengeksekusi data yang ada pada form agar

tersimpan pada database dan ditampilkan pada halaman index. Serta button Kembali yang digunakan untuk kembali ke halaman sebelumnya.



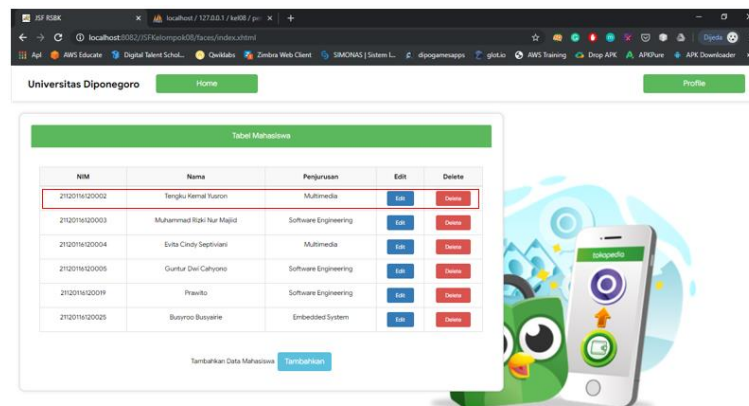
Gambar 5. 59 Halaman Tampilan Edit Data

Pada gambar ini ditampilkan data yang akan dirubah, dan secara otomatis form terisi data ketika button edit pada halaman index.xhtml di klik, data yang ditampilkan ini diambil berdasarkan NIM yang digunakan yang sesuai dengan method Edit\_Mahasiswa pada class mahasiswa.java



Gambar 5. 60 Halaman Tampilan Proses Edit Data

Kemudian mencoba untuk mengganti nilai yang sebelumnya dengan nilai yang "baru mulai dari NIM : 21120116120002, Nama: “Tengku Kemal Yusron”, dan Jurusan : 4.

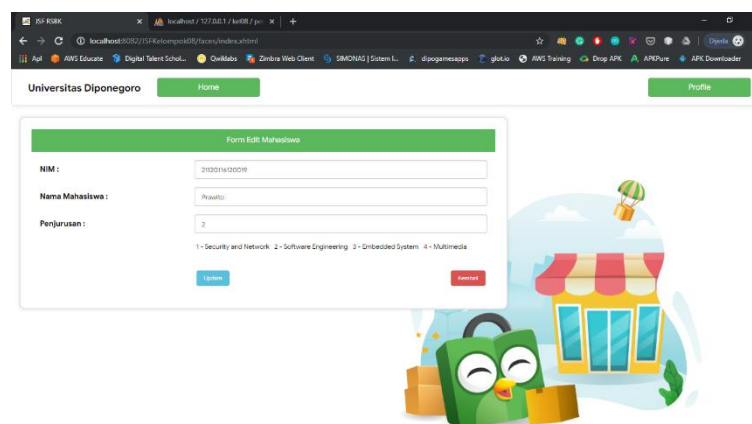


Gambar 5. 61 Halaman Tampilan Data Berhasil di Edit

Dan berikut ditampilkan data di halaman index.xhtml, dan terjadi perubahan pada data yang di edit dari NIM 21120116120002.

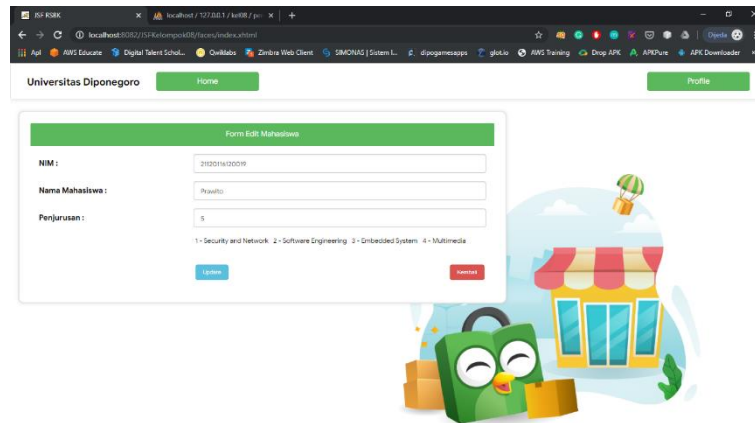
## 7. EditError.xhtmll

Pada file ini hanya digunakan untuk memberikan keterangan apabila pada form penjurusan diisi dengan nilai bukan angka 1 hingga 4. Secara fungsional source code halaman ini sama dengan edit.xhtml hanya saja ditambahkan beberapa source code keterangan untuk menampilkan text “ Penjurusan Tidak Ditemukan ”. Jadi ketika input diberikan 1 hingga 4 maka method Update\_Mahasiswa pada class mahasiswa akan mengeksekusi pengkondisiannya dan me return halaman index.xhtml dengan data telah masuk ke database, kemudian apabila input bukan 1 hingga 5 maka pengkondisian yang di eksekusi akan me return halaman EditError.xhtml. (Sama dengan file TambahError.xhtmll). Dan berikut adalah hasilnya.



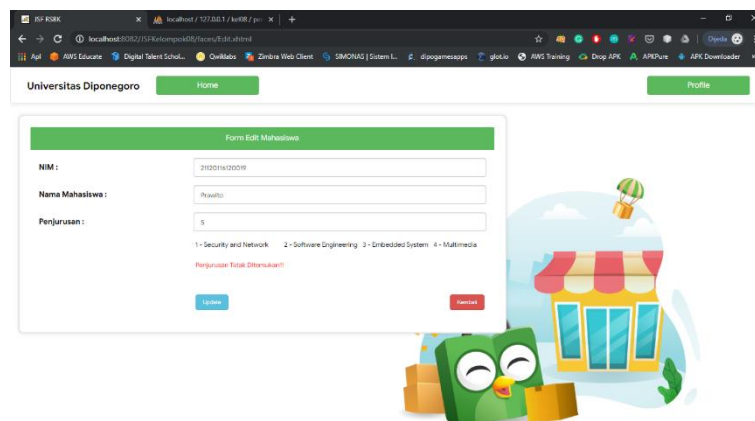
Gambar 5. 62 Halaman Tampilan Edit Data

Menampilkan data dari NIM 21120116120019, Nama “Prawito” dan Penjurusan 2. Kemudian pada gambar dibawah ini adalah perubahan nilai yang dilakukan pada form penjurusan menjadi 5 yang dimana sebelumnya adalah 2.



The screenshot shows a web browser window with the URL `localhost:3000/131/0019/16120019/edit`. The page is titled "Universitas Diponegoro" and has a "Home" button. The main content is a form titled "Form Edit Mahasiswa". The form contains three input fields: "NIM:" with the value "21120116120019", "Nama Mahasiswa:" with the value "Prawito", and "Penjurusan:" with the value "5". Below the "Penjurusan:" field, there is a list of majors: "1 - Security and Network", "2 - Software Engineering", "3 - Embedded System", and "4 - Multimedia". A red error message "Penjurusan Tidak Ditemukan!!" is displayed below the list. A green "Update" button is at the bottom left, and a red "Cancel" button is at the bottom right. A cartoon penguin character is visible in the bottom right corner.

Gambar 5. 63 Halaman Tampilan Proses Edit Data value penjurusan 5



The screenshot shows the same web browser window as the previous one. The "Form Edit Mahasiswa" form is displayed. The "NIM:" field contains "21120116120019", the "Nama Mahasiswa:" field contains "Prawito", and the "Penjurusan:" field contains "5". Below the "Penjurusan:" field, the same list of majors is shown. A red error message "Penjurusan Tidak Ditemukan!!" is displayed below the list. A green "Update" button is at the bottom left, and a red "Cancel" button is at the bottom right. A cartoon penguin character is visible in the bottom right corner.

Gambar 5. 64 Hasil Tampilan Proses Edit Data tidak Berhasil

Dan ini adalah gambar dengan di tampilkannya keterangan Penjurusan Tidak Ditemukan, karena input atau nilai pada form penjurusan adalah 5. Dan berikut adalah tampilan dari 20 data mahasiswa.

Universitas Diponegoro
Home
Profile

Tabel Mahasiswa

NIM	Name	Penjurusan	Edit	Delete
210201620002	Tengku Kemal Yusrin	Multimedia	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620003	Muhammad Rizki Nur Majid	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620004	Erika Cindy Septiarni	Multimedia	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620005	Gurkita Dwi Cahyono	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620007	Kurniasa Satrio Budiyanto	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620008	Agustianan	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620009	Ahmad Shofie Hibi	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620010	Mutiara Victoria M.	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620011	Rio Julian Azis Pratama	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620012	Rahma Nuli Hinaewati	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620013	Nasyim Dahlan Attasulq	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620014	Faisal Rizky Rahadian	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620015	Fauzan Pendera Hedyanto Sutopo	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620009	Ahmad Shofie Hibi	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620010	Mutiara Victoria M.	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620011	Rio Julian Azis Pratama	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620012	Rahma Nuli Hinaewati	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620013	Nasyim Dahlan Attasulq	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620014	Faisal Rizky Rahadian	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620015	Fauzan Pendera Hedyanto Sutopo	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620016	Amara Rani Rochita	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620017	Septi Nurma Afiani	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620018	Ami Syarifuddin Yahya	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620019	Prascho	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620020	Umaran Rishi Muli	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620021	Fidiana Yunka Sari	Software Engineering	<a href="#">Edit</a>	<a href="#">Delete</a>
210201620025	Butyasa Butyasa	Embedded System	<a href="#">Edit</a>	<a href="#">Delete</a>

Tambahkan Data Mahasiswa
[Tambahkan](#)

Gambar 5. 65 Halaman Tampilan index dengan 20 Data

Link Github : <https://github.com/Prawito-17/PraktikumRSBKKel08.git>

Nama File : MODUL 4 JAVA SERVER FACES

## 5.6 Tugas

1. Untuk menambahkan *connector* MySQL dilakukan dengan cara menambahkannya pada library program.
2. Dalam membuat aplikasi web, JSF menyimpan beberapa state termasuk state client-side dan server-side sehingga pengembangannya lebih cepat dari framework lain
3. JSF mendukung event UI dan penanganannya sangat mudah untuk dipahami, serta terdapat beberapa perbedaan dalam penggunaan xhtml dan html dalam penggunaan css dll.
4. Bootstrap css dapat diimplementasikan pada *program* dengan cara menambahkannya pada *folder resource* kemudian untuk memanggilnya dengan cara `<h:openStylesheet>`. Serta dengan menggunakan CSS, tampilan dapat mendesain warna background, jenis tampilan, border, text area, button, dll.
5. File JSF *managed bean* berguna sebagai *backend* yang mana pada *file* ini terdapat *method* untuk menjalankan *query* CRUD. Dan JSF mengambil manfaat dari komponen pihak ketiga tentunya dapat meminimalkan biaya menulis ulang kode program sehingga efisien waktu
6. Entity menjelaskan aturan-aturan dan penempatan kelas entity secara rinci sehingga dapat memodelkan informasi yang berumur relative. Dan Session bean mempresentasikan proses hanya berdasarkan permintaan client dan dapat berinteraksi dengan resource yang lain seperti entity bean, dan session bean yang lain.
7. Webpages yang dibuat digunakan untuk menampilkan tampilan yang sesuai dengan yang dibuat. Serta pada JSF xhtml untuk menampilkan *table* dilakukan dengan cara `<dataTable></dataTable>`.
8. *Connector* MySQL digunakan untuk membangun suatu koneksi antara program dengan *database* MySQL.