# BAB V JAVA SERVER FACES

## 5.1 Tujuan

1. Praktikan mengenal Framework JSF
2. Praktikan mengetahui penggunaan Java Server Faces.
3. Praktikan mampu membuat aplikasi CRUD sederhana dengan JSF
4. Praktikan dapat menggunakan Apache Tomcat *server* dalam pengembangan web menggunakan Bahasa pemrograman Java.
5. Praktikan dapat mengkoneksikan basis data MySQL dengan Apache Tomcat server.

## 5.2 Dasar Teori

### 5.2.1 NetBeans IDE

NetBeans adalah suatu serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Serambi Pada NetBeans, pengembangan suatu aplikasi dapat dilakukan dimulai dari setelan perangkat lunak modular bernama modules. Semula, aplikasi NetBeans IDE ini diperuntukkan bagi pengembangan dalam Java. Namun, aplikasi ini juga mendukung program-program pembuatan bahasa lain secara khusus seperti PHP, C/C++ dan HTML5. NetBeans adalah alat lintas serambi serta penerapannya dijalankan pada Microsoft Windows, Mac OS X, Linux, Solaris dan serambi-serambi lainnya yang mendukung JVM yang sepadan.

*(Sumber :* [*https://id.wikipedia.org/wiki/NetBeans*](https://id.wikipedia.org/wiki/NetBeans) *[diakses : 15 November 2019])*

### 5.2.2 Java Server Faces

JavaServer Faces (JSF) adalah spesifikasi Java untuk membangun antarmuka pengguna untuk aplikasi web. JSF merupakan bagian dari Java Platform, Enterprise Edition. JSF 2 menggunakan Facelets sebagai sistem templat defaultnya. Teknologi tampilan lainnya, seperti XUL, juga dapat digunakan. Sementara itu, JSF 1.x menggunakan JavaServer Pages (JSP) sebagai sistem templat defaultnya.

Javaserver Faces berdasarkan model perancangan antarmuka pengguna berbasis komponen, menggunakan berkas XML yang disebut templat view atau view Facelets. Permintaan (request) diproses oleh FacesServlet, yang memuat templat view yang sesuai, membangun tree komponen, memproses berbagai event, dan me-render respons (umumnya HTML) kepada klien. State dari komponen UI (dan beberapa objek lain) disimpan pada setiap akhir request (dinamakan stateSaving), dan dikembalikan pada saat pembuatan selanjutnya dari view tersebut. Ada beberapa jenis penyimpanan state, termasuk penyimpanan state client-side dan server-side.

*(Sumber :* [*https://id.wikipedia.org/wiki/JavaServer\_Faces*](https://id.wikipedia.org/wiki/JavaServer_Faces) *[diakses : 15 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 JavaServer 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 dikonfigurasi dengan mengedit file konfigurasi XML.

*(Sumber :* [*https://www.webhozz.com/blog/instalasi-tomcat-di-centos/*](https://www.webhozz.com/blog/instalasi-tomcat-di-centos/) *[diakses : 15 November 2019])*

### 5.2.4 Session Bean

Session bean adalah EJB yang digunakan untuk mengeksekusi proses. Isi dari Session Bean ini biassanya 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 Statefull 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/*](https://suhearie.wordpress.com/2008/08/26/java-enterprise-mulai-dari-mana-part-2-netbeans-glassfish/) *[diakses : 15 November 2019])*

### 5.2.5 MySQL Connector

Ketika membuat program aplikasi dengan bahasa pemrograman Java yang mengakses ke basis data (database), Anda dapat menggunakan berbagai jenis basis data. Salah satu basis data yang bisa Anda gunakan adalah MySQL. Agar program aplikasi Java dapat terkoneksi ke basis data MySQL, diperlukan JDBC Driver untuk MySQL. JDBC Driver dari MySQL disebut dengan MySQL Connector/J. Sebelum membuat program aplikasi basis data dengan Java, MySQL Connector/J harus sudah diinstal dengan benar di sistem komputer, bila belum, Anda dapat menggunakan petunjuk instalasi MySQL Connector/J di artikel ini. Bila Anda belum memiliki MySQL JDBC Driver (MySQL Connector/J), Anda dapat mengunduh (download) MySQL JDBC Driver dari website resmi MySQL untuk Connertor/J.

*(Sumber :* [*https://www.termasmedia.com/pemrograman/java/476-instalasi-mysql-connector-j-koneksi-program-java-ke-database-mysql.html*](https://www.termasmedia.com/pemrograman/java/476-instalasi-mysql-connector-j-koneksi-program-java-ke-database-mysql.html) *[diakses : 15 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 repositori pribadi dan gratis pada akun yang sama dan digunakan untuk proyek perangkat lunak sumber terbuka. Pada bulan April 2017, GitHub melaporkan bahwa mereka mempunyai lebih dari 20 juta pengguna dan lebih dari 57 juta repositori, menjadikannya layanan terbesar dari kode sumber di dunia.

*(Sumber :* [*https://id.wikipedia.org/wiki/GitHub*](https://id.wikipedia.org/wiki/GitHub) *[diakses : 15 November 2019])*

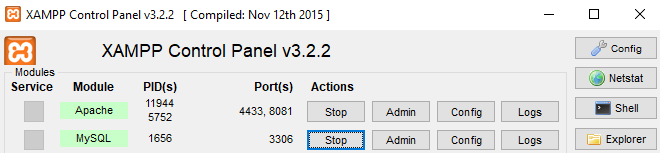
### 5.2.7 XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (localhost), yang terdiri atas program Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas, merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis. Untuk mendapatkanya dapat mendownload langsung dari web resminya.

*(Sumber :* [*https://id.wikipedia.org/wiki/XAMPP*](https://id.wikipedia.org/wiki/XAMPP) *[diakses : 15 November 2019])*

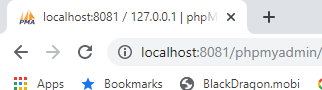
## 5.3 Langkah Kerja

1. Buka XAMPP, lalu Aktifkan Apache dan MySQL



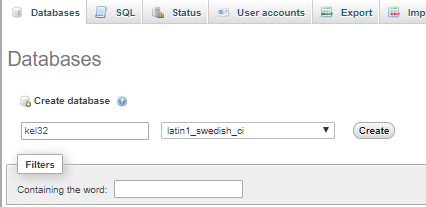
Gambar 5.1 Membuka XAMPP

1. Buka Browser lalu ketikkan : localhost/phpMyAdmin



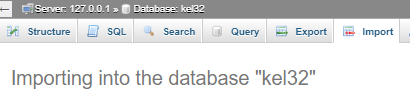
Gambar 5.2 Mengakses phpmyadmin

1. Buat database terlebih dahulu, database tersebut diberi nama kelxx



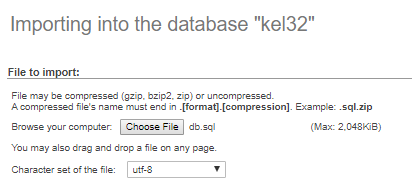
Gambar 5.3 Membuat Database

1. Kemudian, buka Tab impor



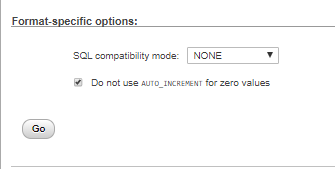
Gambar 5.4 Mengimport ke database baru

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



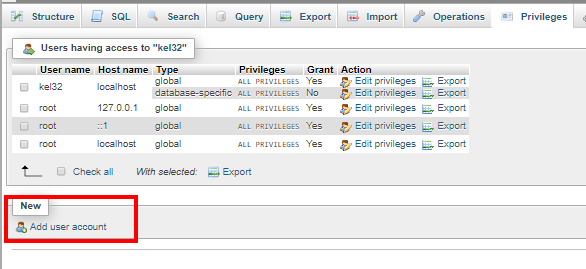
Gambar 5.5 Mengimport file sql

1. Dan kemudian pilih Go



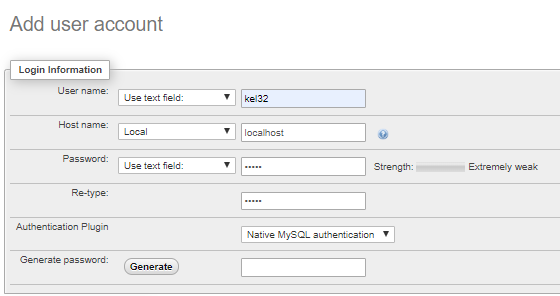
Gambar 5.6 Mengeksekusi file sql

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



Gambar 5.7 Membuat user baru terhadap database

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

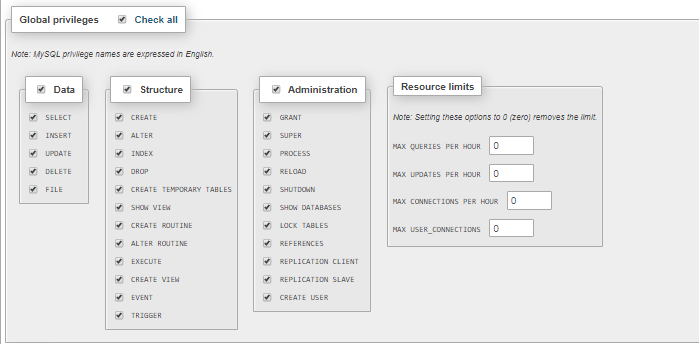


Gambar 5.8 Memasukan informasi login

Keterangan:

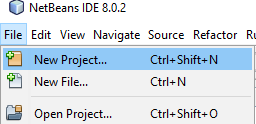
* Nama Pengguna : kelxx
* Nama Pemilik : localhost (form sampingnya harus lokal)
* Kata sandi : kelxx

Setelah itu pilih semua dan kirim



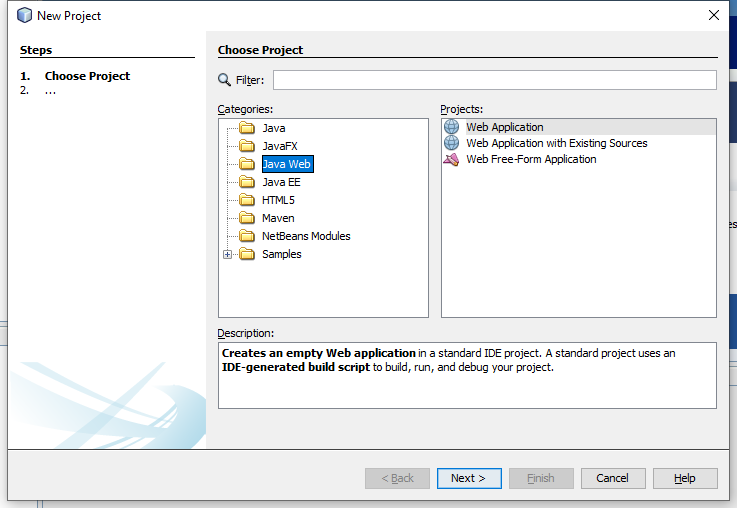
Gambar 5.9 Mengatur hak user

1. Kemudian, buka Netbeans kalian dan pastikan sudah terinstall Tomcat
2. Kemudian pilih File, lalu New Project



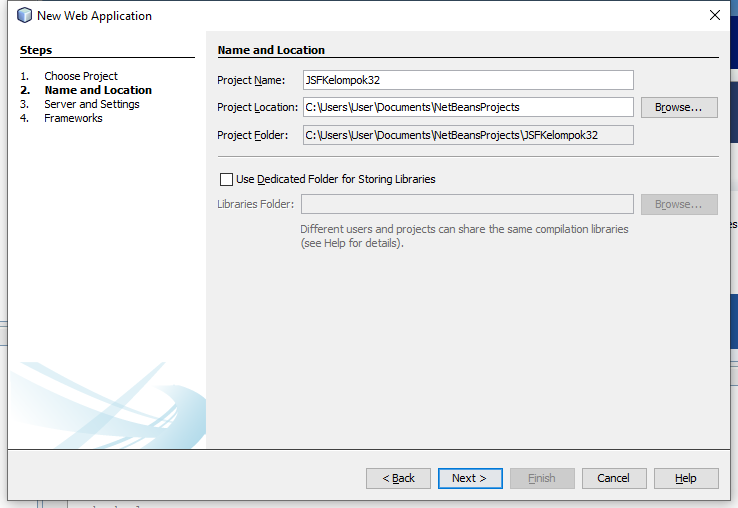
Gambar 5.10 Membuat project baru

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



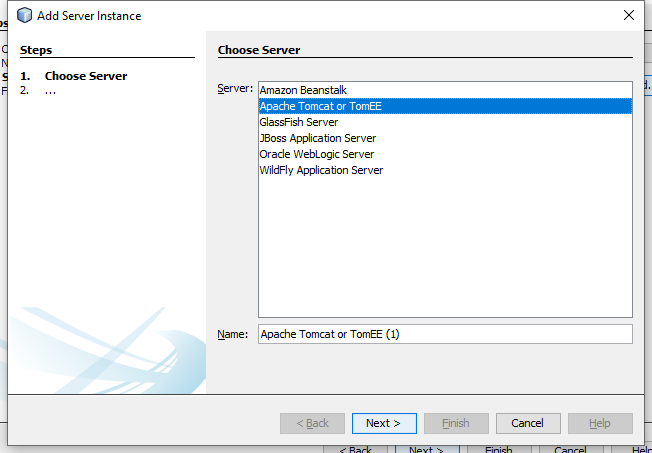
Gambar 5.11 Project Java Web Application

1. Lalu, beri nama ‘JSFKelompokXX’, [ Ganti XX dengan No. Kelompok kalian ], kemudian Next lagi



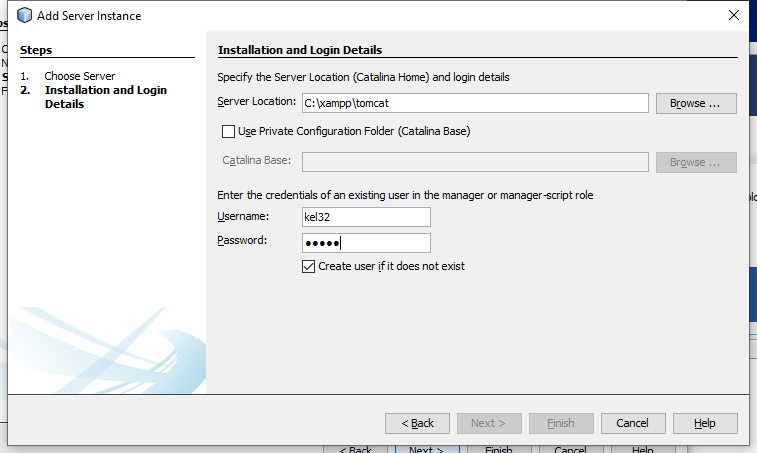
Gambar 5.12 Memberi nama project

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



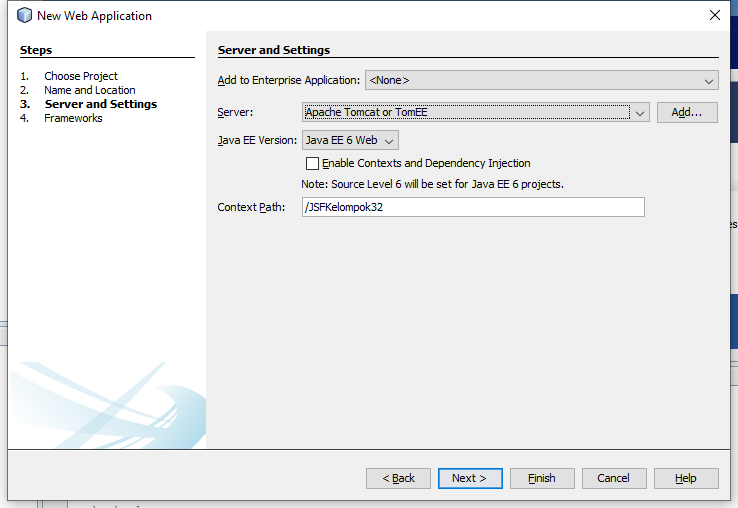
Gambar 5.13 Menambah instance server

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



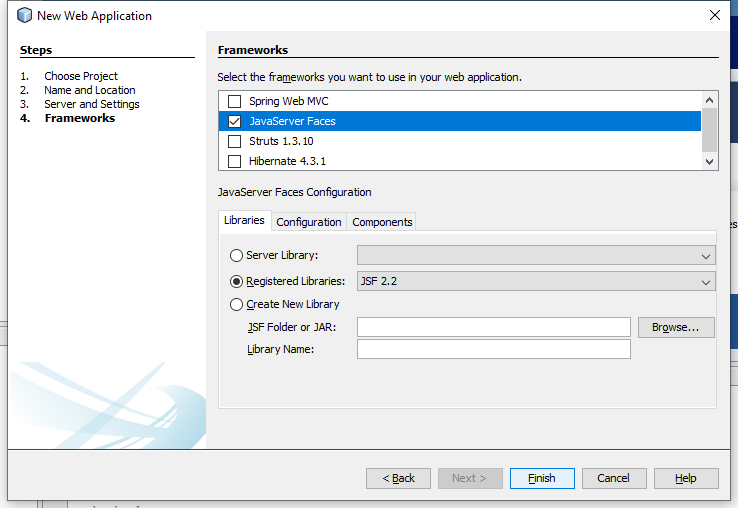
Gambar 5.14 Lokasi instalasi dan detail login server

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



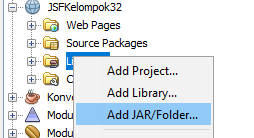
Gambar 5.15 Memilih Apache Tomcat sebagai server

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



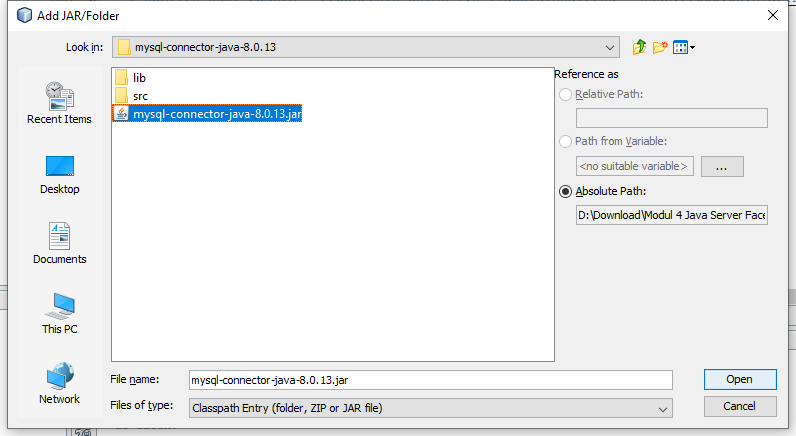
Gambar 5.16 Memilih JSF sebaagai *framework*

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



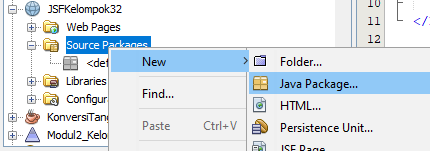
Gambar 5.17 Menambahkan file jar

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



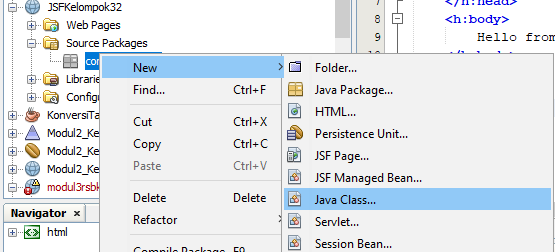
Gambar 5.18 Menambahkan connector

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



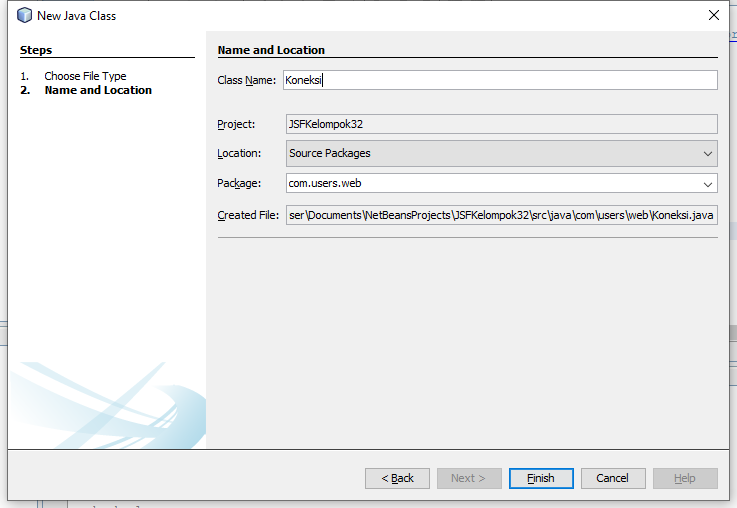
Gambar 5.19 Membuat Java Package baru

1. 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 5.20 Membuat Java Class baru

1. Kemudian berikana nama ‘Koneksi’, lalu tekan Finish

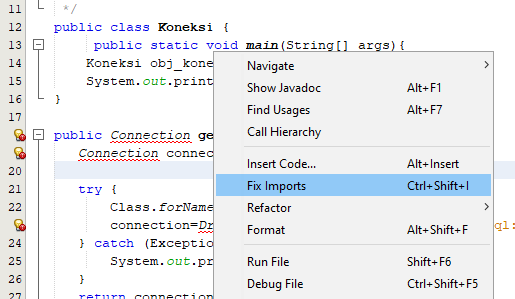


Gambar 5.21 Membuat class koneksi

1. Lalu, selanjutnya. Salin source code di bawah ke dalam file Koneksi.java. Tepatnya, di dalam perintah 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/rsbk\_kelxx", "kelxx", "kelxx");  } catch (Exception e) {  System.out.println(e);  }  return connection;  } |

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



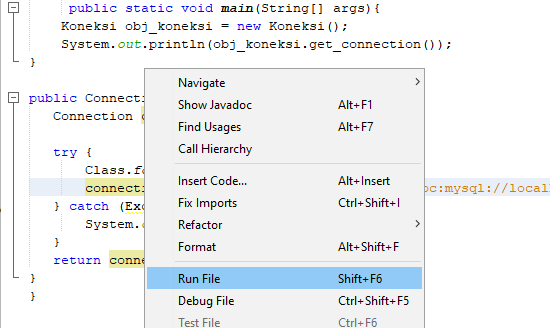
Gambar 5.22 *Fix import*

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



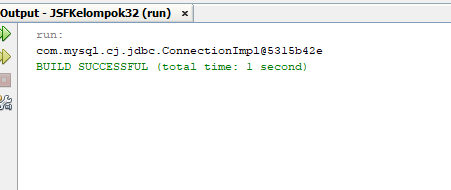
Gambar 5.23 Mengubah informasi login database

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



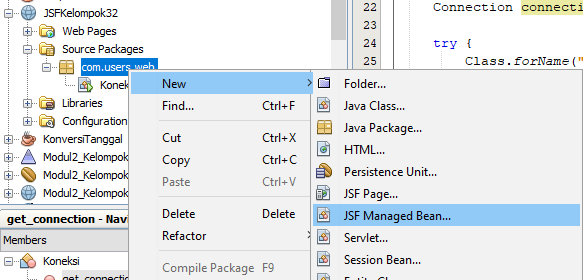
Gambar 5.24 Coba run file koneksi

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



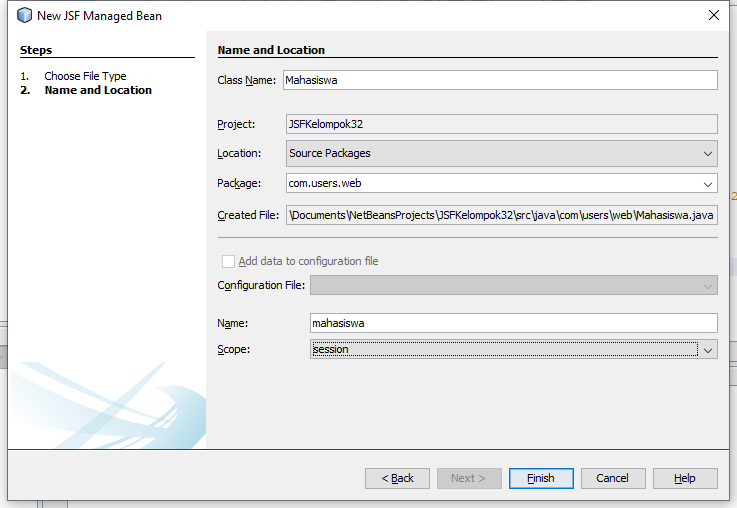
Gambar 5.25 Koneksi berhasil dibuat

1. Selanjutnya, klik kanan lagi pada ;com.users.web;. Lalu pilih New, dan pilih JSF Managed Bean. Jika pilihan tidak tersedia klik pilihan “Other”



Gambar 5.26 Membuat JSF Managed Bean baru

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

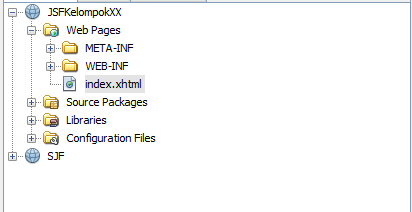


Gambar 5.27 Membuat JSF Managed Bean Mahasiswa

1. Selanjutnya, pada Area Kerja ‘Mahasiswa.java’ yang terletak dibawah tulisan 'package com.users.web', isi dengan source code di bawah ini.

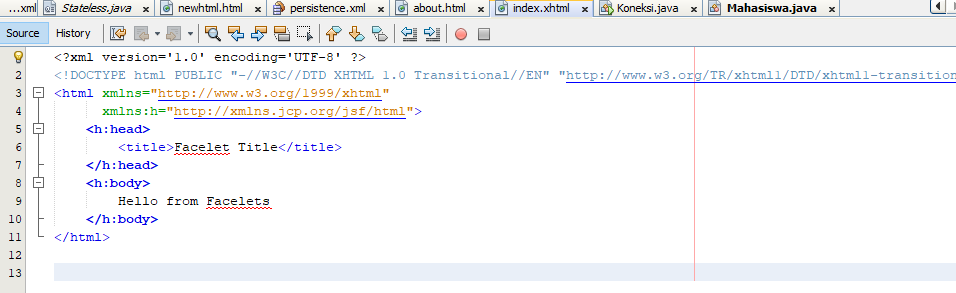
|  |
| --- |
| import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.Map;  import javax.faces.bean.ManagedBean;  import javax.faces.bean.RequestScoped;  import javax.faces.context.FacesContext;  /\*\*  \*  \* @author Hakushuu  \*/  @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() {}    } |

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



Gambar 5.28 File index.xhtml

1. Lalu akan tampil, tampilan seperti berikut

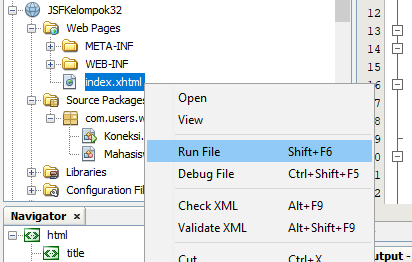


Gambar 5.29 isi file index.xhtml

1. Lalu salin source code di bawah ke dalam file ‘index.xhtml’ tersebut

|  |
| --- |
| <?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>  <h:form>  <h:form>  Tambahkan Data Mahasiswa &nbsp;  <h:commandButton value="Tambahkan" action="Tambah.xhtml"></h:commandButton>  </h:form>  </h:form>  </center>  </h:body>  </html> |

1. Kemudian klik kanan pada file ‘Index.xhtml’ lalu pilih Run File

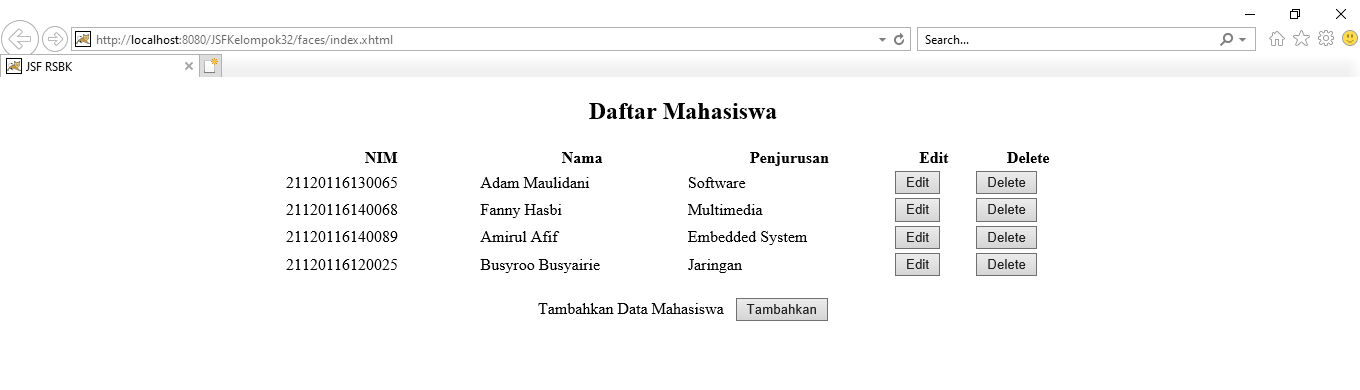


Gambar 5.30 Mencoba run file index.xhtml

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

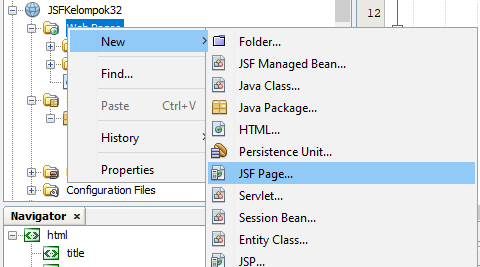
|  |
| --- |
| Run cmd as administrator  netstat -ano | findstr :8080  taskkill /PID 3748 /F  # 3748 adalah process id |

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



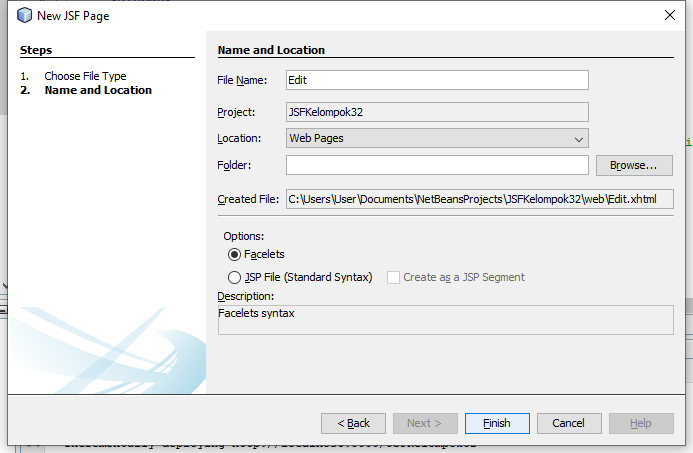
Gambar 5.31 Hasil run file index.xhtml

1. Pada tampilan tersebut, button Edit dan Tambahkan, belum dapat digunakan, maka kita lanjut ke langkah selanjutnya
2. Kembali lagi ke Netbeans kalian, lalu klik kanan pada Folder Web Pages, lalu pilih New, dan pilih JSF Page

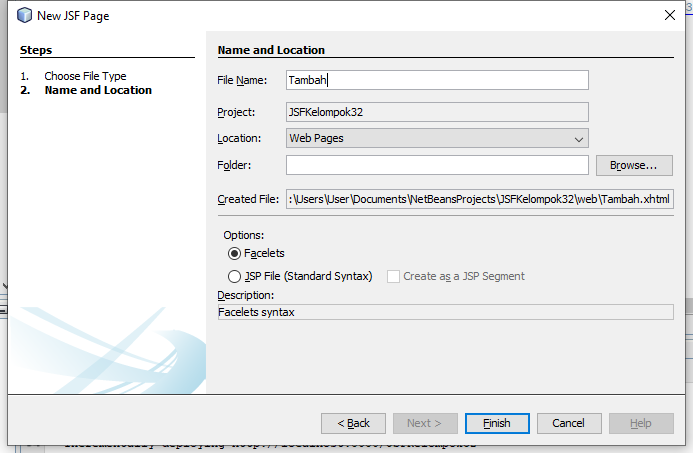


Gambar 5.32 Membuat JSF Page baru

1. Buat 2 buah File JSF Page, dengan nama ‘Edit’ dan ‘Tambah’



Gambar 5.33 Membuat JSF Page Edit



Gambar 5.34 Membuat JSF Page Tambah

1. Kemudian salin source code di bawah ke dalam file ‘Edit.xhtml.

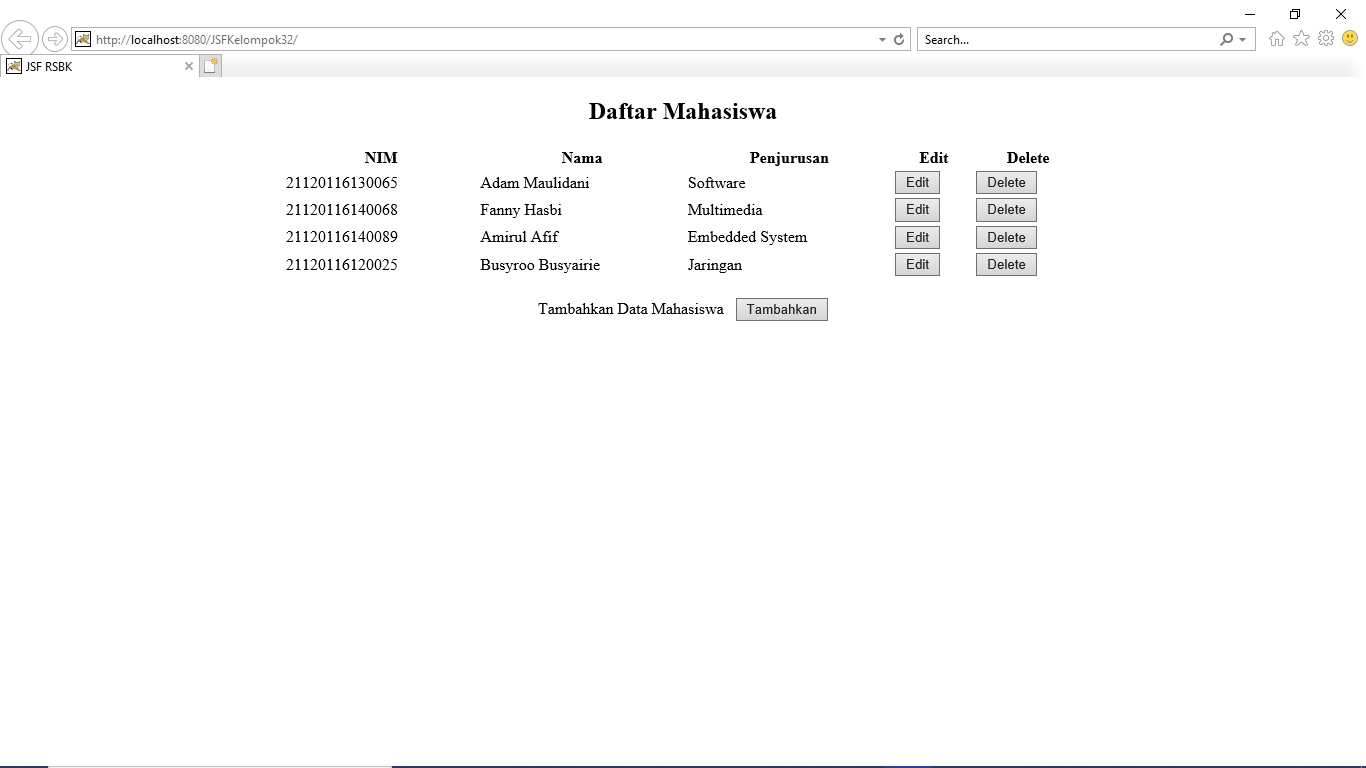
|  |
| --- |
| <?xml version='1.0' encoding='UTF-8' ?>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:f="http://java.sun.com/jsf/core"  xmlns:h="http://xmlns.jcp.org/jsf/html">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

1. Dan juga salin source code berikut 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> |

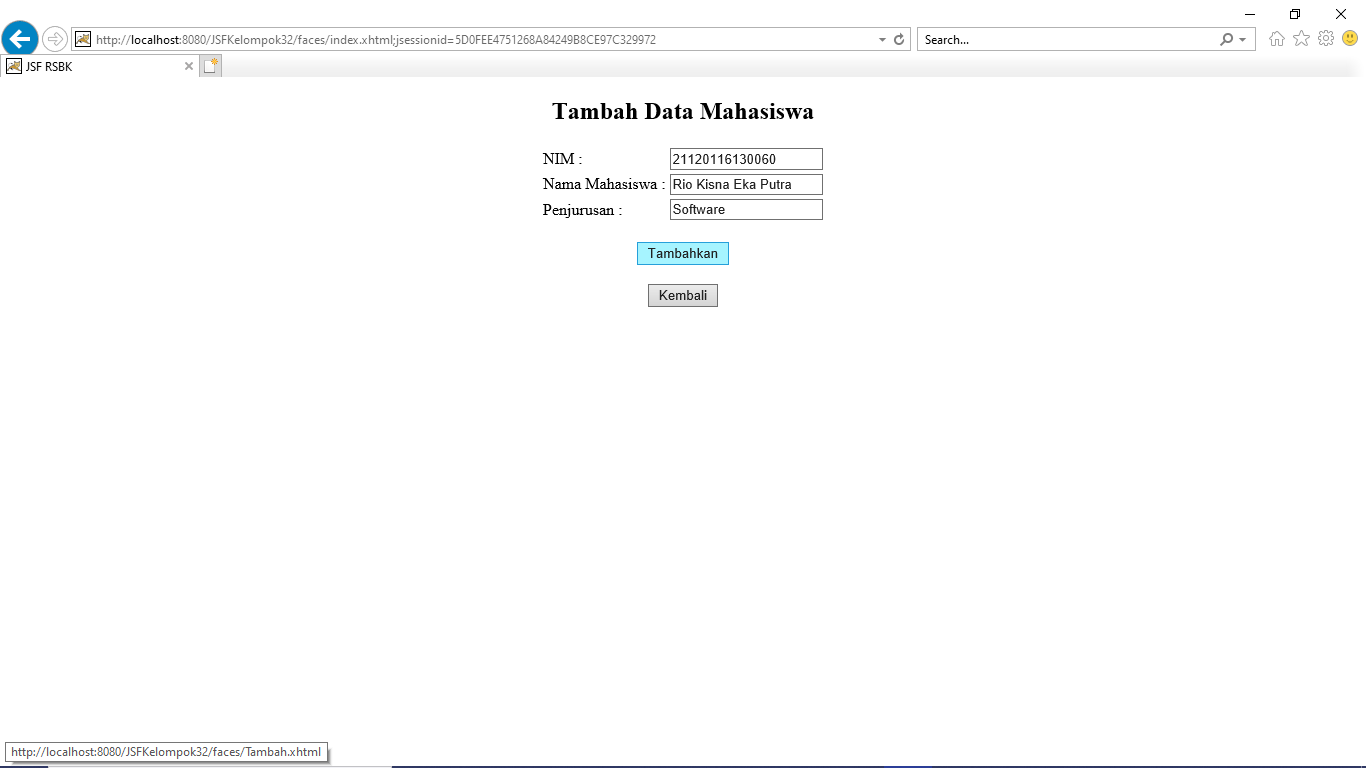
1. Kemudian, coba kalian lakukan lagi Run File ‘index.xhtml’ . Dan, coba tekan button Edit atau Tambahkan. Maka button tersebut kini dapat digunakan

## 5.4 Hasil dan Analisa



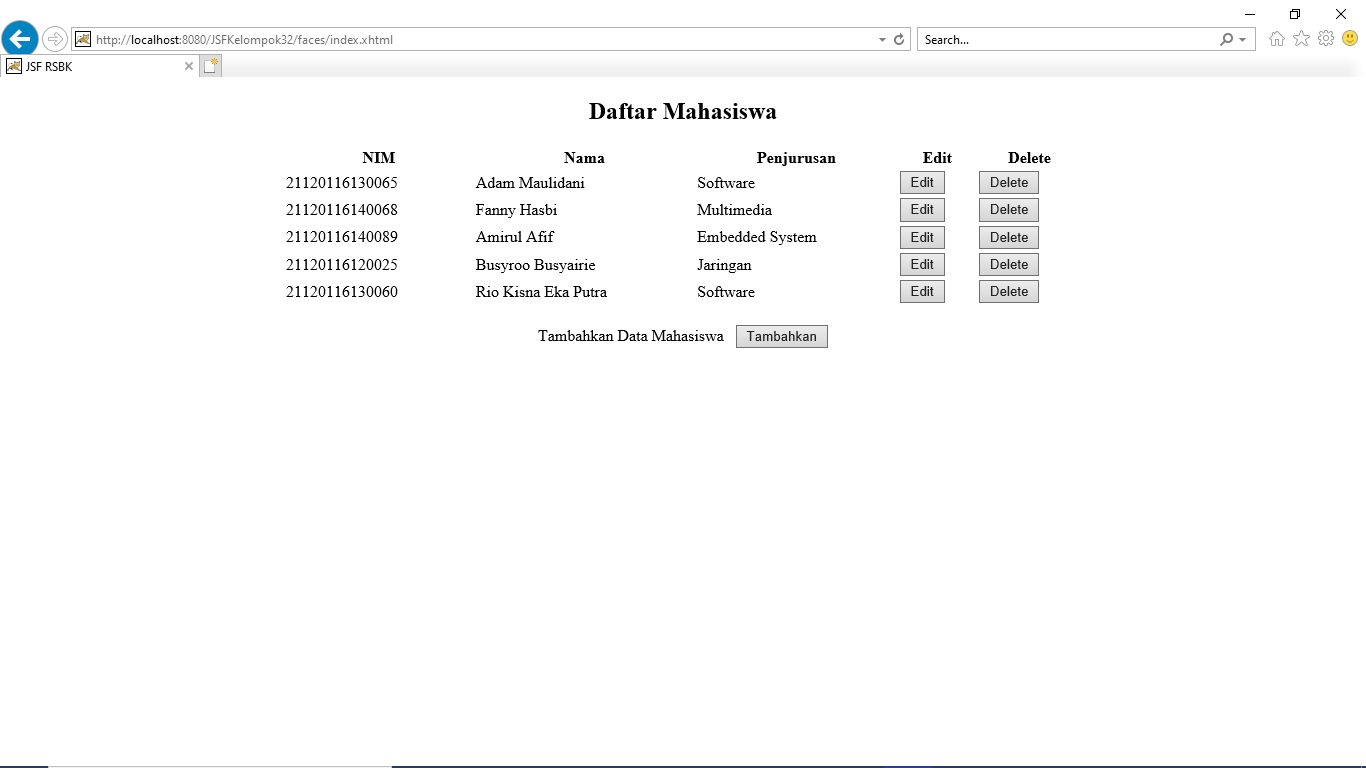
Gambar 5.35 Hasil praktikum halaman index

Hasil praktikum pada halaman index akan seperti itu. Sebuah tabel di tampilkan berdasarkan data yang diambil dari basis data melalui fungsi get\_all\_mahasiswa(). Kemudian juga ada button Edit, delete, dan tambahkan data. Ketika button edit atau delete di tekan, maka akan mengambil dan mengirim sebuah parameter NIM, untuk di eksekusi lebih lanjut pada fungsi lainnya.



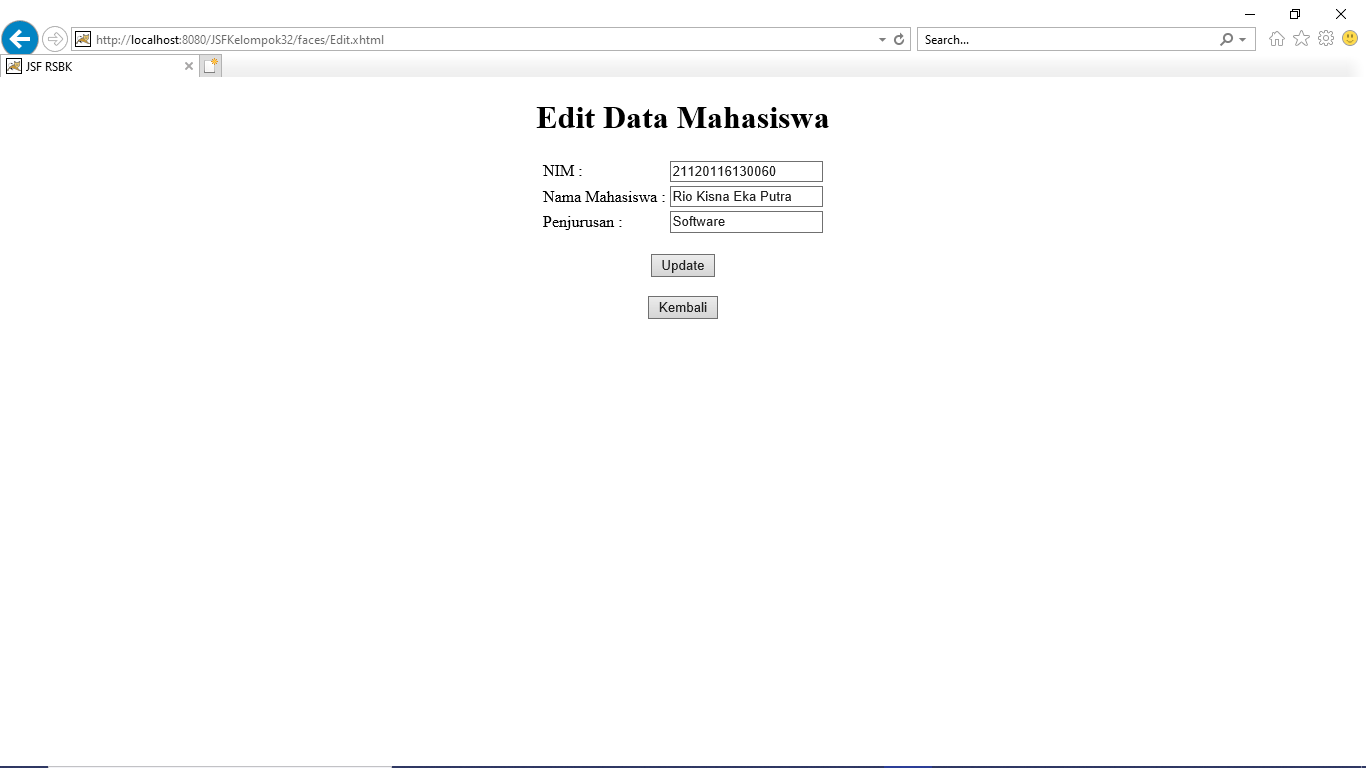
Gambar 5.36 Hasil praktikum tambah data

Pada halaman tambah data, terdapat textBox yang dapat diisi dengan nilai untuk memasukan informasi. Informasi tersebut nantinya akan di baca dan di kirimkan ke fungsi Tambah\_mahasiswa() untuk di eksekusi menggunakan query ke database.



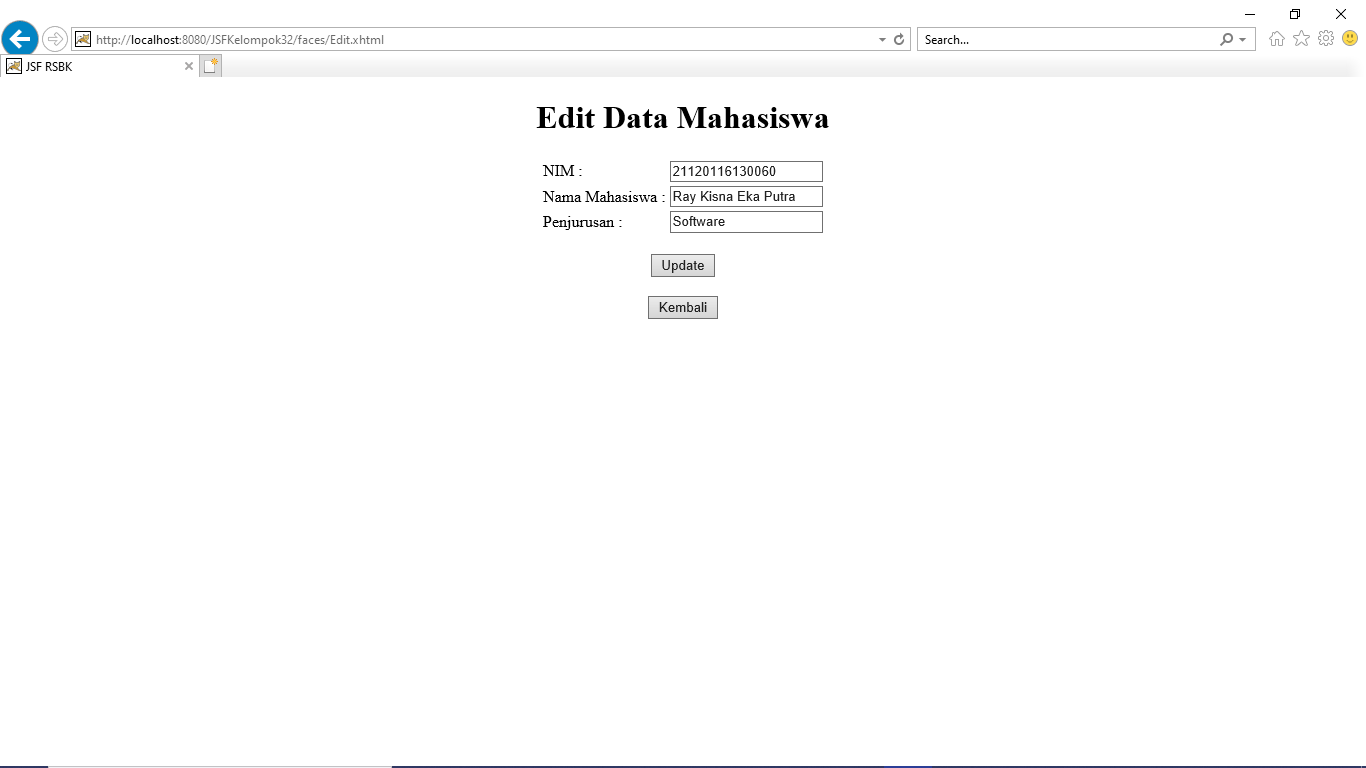
Gambar 5.37 Hasil praktikum berhasil menambah data

Ketika data ditambahkan, maka akan terdapat data baru sesuai data yang ditambahkan sebelumnya.



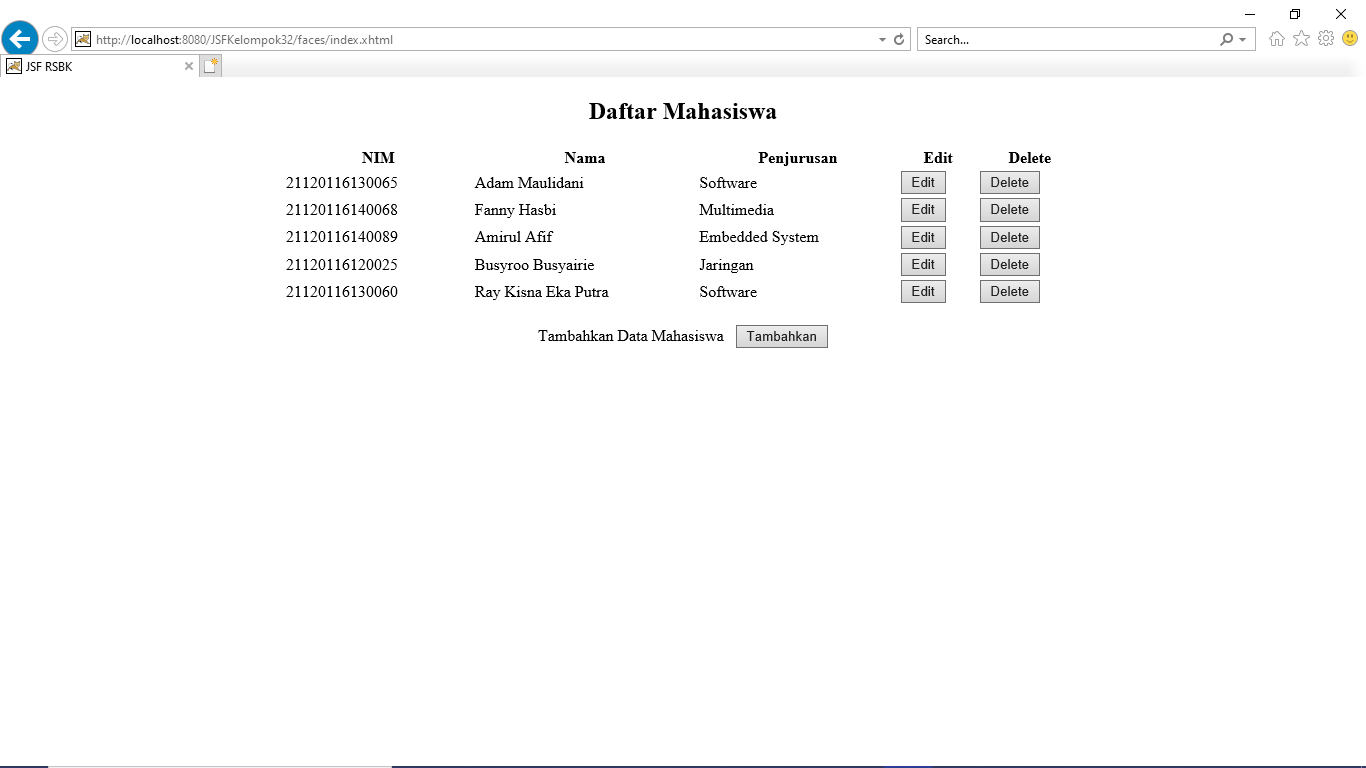
Gambar 5.38 Hasil praktikum edit mahasiswa

Untuk halaman edit data diakses dengan menekan button edit pada halaman index. Pada button tersebut, Facelet akan menyimpan nilai pada parameter berupa objek mahasiswa. Nim tersebut akan di gunakan untuk mengakses data pada tabel mahasiswa kemudian menulikannya ke textBox yang sudah di buat. Fungsi yang di eksekusi ketika membuka halaman ini adalah fungsi Edit\_Mahasiswa().



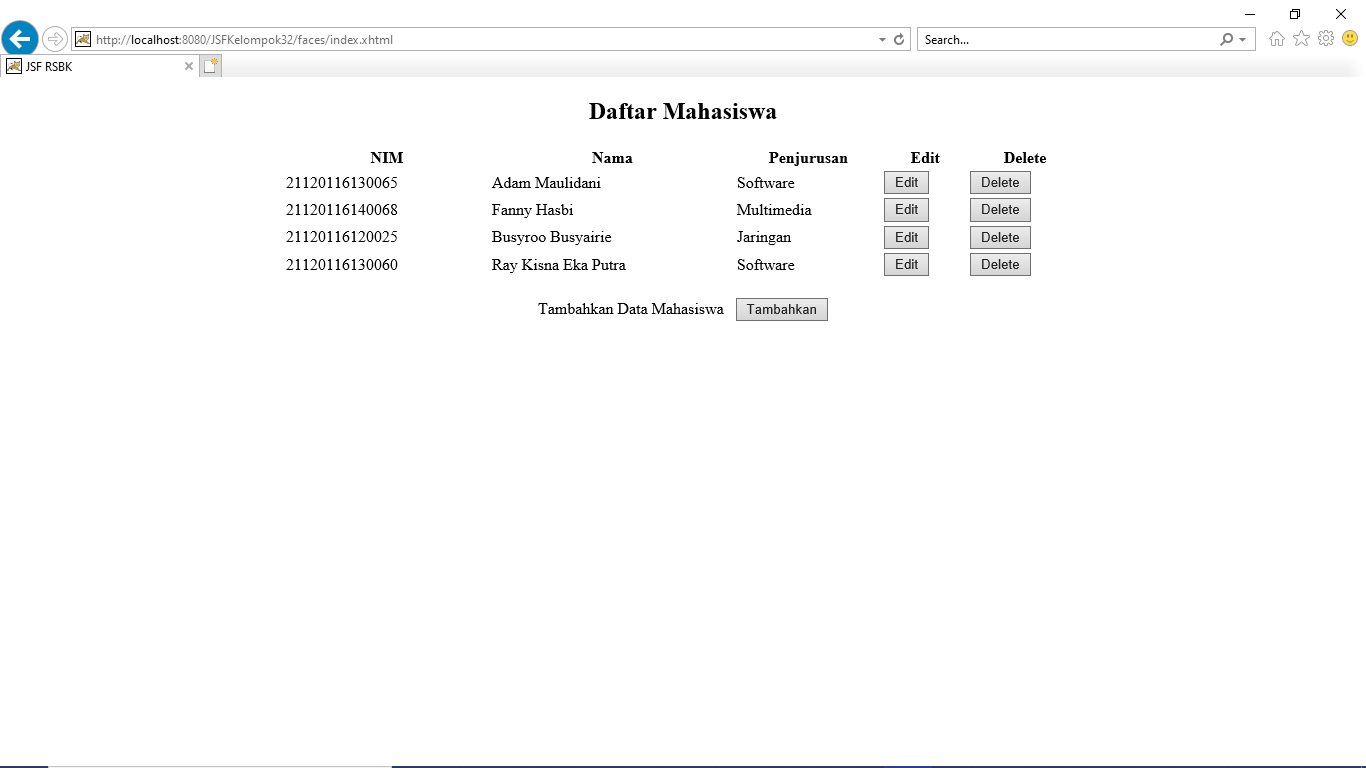
Gambar 5.39 Hasil praktikum halaman edit mahasiswa

Ketika button update di tekan, maka program akan menjalankan fungsi Update\_Mahasiswa(). Fungsi tersebut berguna untuk mengupdate pada basis data sesuai dengan data yang di ubah. Kemudian dengan menggunakan nilai NIM yang juga disimpan sebagai parameter, maka data sesuai dengan NIM lama akan diubah dengan data yang baru.



Gambar 5.40 Hasil praktikum edit mahasiswa berhasil

Terlihat bahwa pada nama berubah.



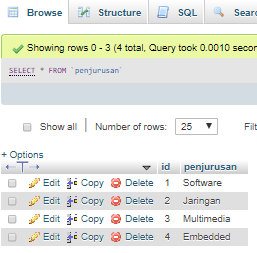
Gambar 5.41 Hasil praktikum delete mahasiswa berhasil

Kemudian untuk fungsi delete, sama dengan fungsi edit dengan mengambil parameter NIM, kemudian menjalankan fungsi Delete\_Mahasiswa() untuk mengeksekusi query delete berdasarkan parameter NIM yang tersimpan. Sehingga data tersebut akan langsung terhapus

## 5.5 Tugas dan Pembahasan

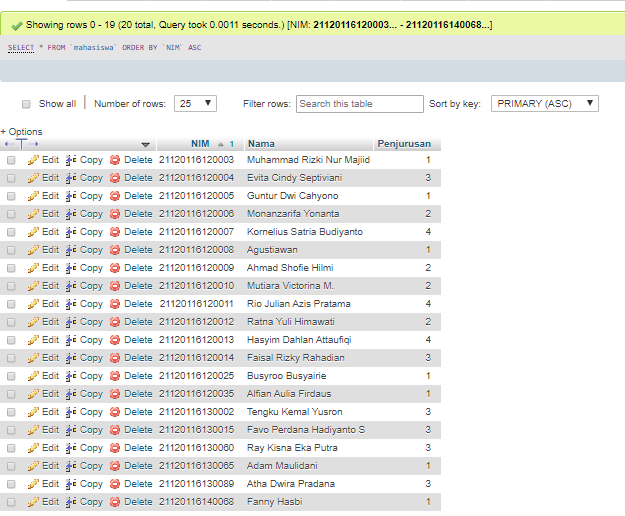
Tugas diminta untuk menambahkan tabel baru yaitu tabel penjurusan, setelah itu menghubungkannya dengan tabel mahasiswa. Ketika melakukan tambah data atau edit data dan salah memasukan penjurusan, maka akan menampilkan pesan error.

Pertama kali yang dilakukan adalah menambahkan tabel baru, yaitu tabel penjurusan. Berikut ini adalah tabel penjurusan yang kami buat



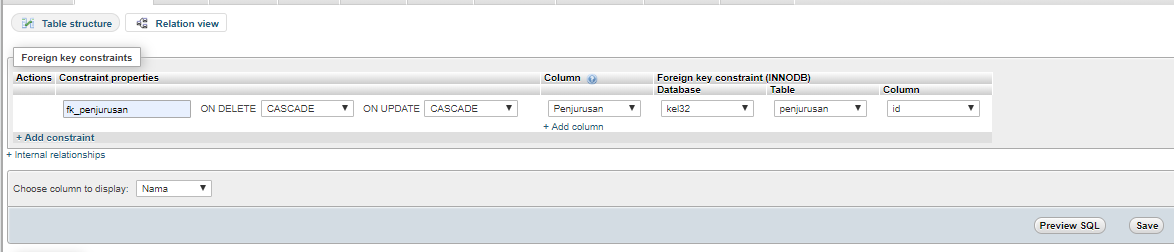
Gambar 5.42 Tabel penjurusan

Karena primary key yang di gunakan pada tabel tersebut adalah ID, untuk membuatnya dapat dihubungkan dengan tabel mahasiswa, perlu dilakukan pengubahan struktur tabel mahasiswa dimana penjurusan pada tabel itu dijadikan sebagai id nya.



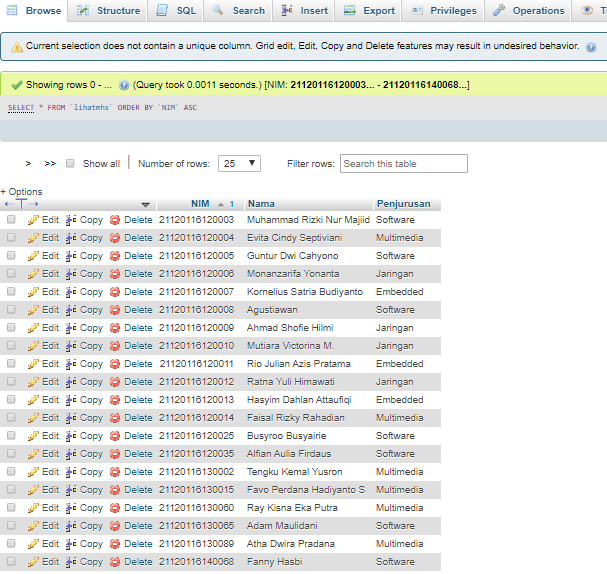
Gambar 5.43 Tabel mahasiswa baru

Setelah itu melakukan constraint antara tabel mahasiswa dengan tabel penjurusan



Gambar 5.44 Constraint foreign key

Setelah di constraint agar data yang ditampilkan pada tabel di index.xhtml tetap seperti semula, perlu dibuat sebuah view untuk menghasilkan tampilan hasil join kedua tabel, kemudian mengubah semua select mahasiswa menjadi select view tersebut. Berikut ini adalah view yang dimaksud



Gambar 5.45 tabel view

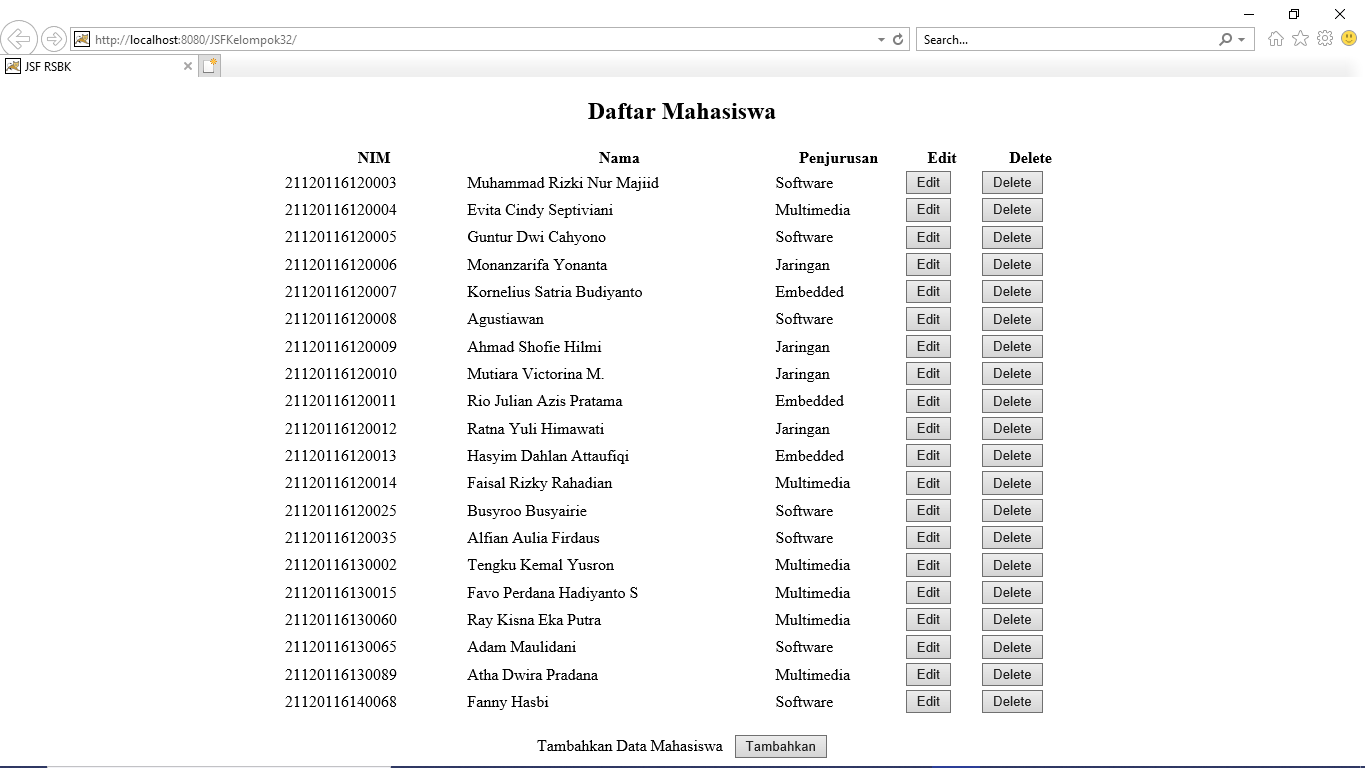
Basis data telah di siapkan, selanjutnya adalah memberi pengkondisian baru pada source code Mahasiswa.java untuk proses penambahan data dan pengubahan data. Pengkondisian tersebut digunakan untuk mengecek apakan penjurusan yang di maksud sudah ada pada tabel penjurusan. Ketika ada maka data berhasil ditambahkan, namun ketika tidak ada maka data tidak jadi di tambahkan dan akan menampilkan pesan error.

Source code pada fungsi tambah Mahasiswa

|  |
| --- |
| public String Tambah\_Mahasiswa(){  try {  Connection connection=null;  Koneksi obj\_koneksi = new Koneksi();  connection = obj\_koneksi.get\_connection();  Statement st = connection.createStatement();  ResultSet rs = st.executeQuery("Select \* from penjurusan where penjurusan = '"+PENJURUSAN+"'");  String id=null;  while(rs.next()){  id = rs.getString("id");  }  if (id!=null){  PreparedStatement ps=connection.prepareStatement("insert into mahasiswa(NIM, Nama, Penjurusan) value('"+NIM+"','"+NAMA+"','"+id+"')");  ps.executeUpdate();  }  else {  FacesMessage message = new FacesMessage("Ayolah.. gak ada penjurusan macam itu");  FacesContext context = FacesContext.getCurrentInstance();  context.addMessage(myButton.getClientId(context), message);  return null;  }  } catch (Exception e) {  System.out.println(e);  }  return "/index.xhtml?faces-redirect=true";  } |

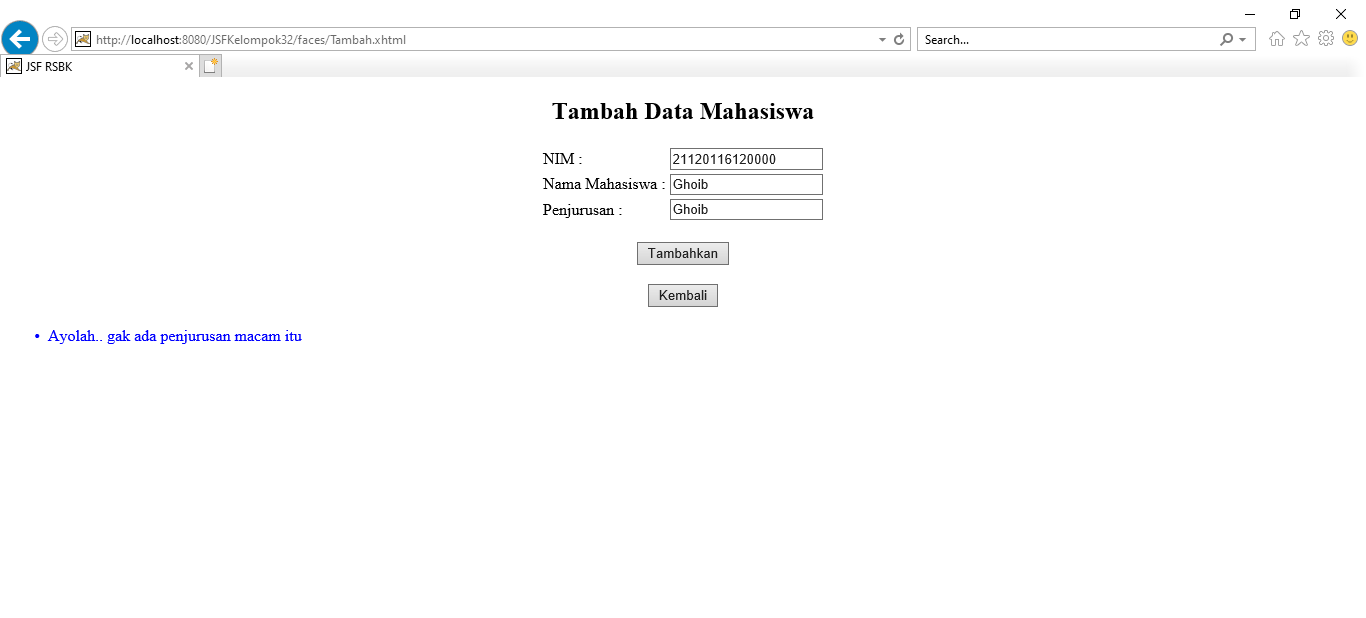
Source code pada fungsi Update\_Mahasiswa()

|  |
| --- |
| public String Update\_Mahasiswa(){  FacesContext fc = FacesContext.getCurrentInstance();  Map<String,String> params = fc.getExternalContext().getRequestParameterMap();  String Update\_NIM= params.get("Update\_NIM");  try {  Connection connection=null;  Koneksi obj\_koneksi = new Koneksi();  connection = obj\_koneksi.get\_connection();  Statement st = connection.createStatement();  ResultSet rs = st.executeQuery("Select \* from penjurusan where penjurusan = '"+PENJURUSAN+"'");  String id2=null;  while(rs.next()){  id2 = rs.getString("id");  }  if (id2!=null){  PreparedStatement ps = connection.prepareStatement("update mahasiswa set NIM=?, Nama=?, Penjurusan=? where NIM=?");  ps.setString(1, NIM);  ps.setString(2, NAMA);  ps.setString(3, id2);  ps.setString(4, Update\_NIM);  ps.executeUpdate();  System.out.println(id2);  }  else {  // FacesMessage message2 = new FacesMessage("Ayolah.. gak ada penjurusan macam itu");  // FacesContext context2 = FacesContext.getCurrentInstance();  // context2.addMessage(myButton2.getClientId(context2), message2);  return "/ErrorEdit.xhtml?faces-redirect=true";  }  } catch (Exception e) {  System.out.println(e);  }  return "/index.xhtml?faces-redirect=true";  } |



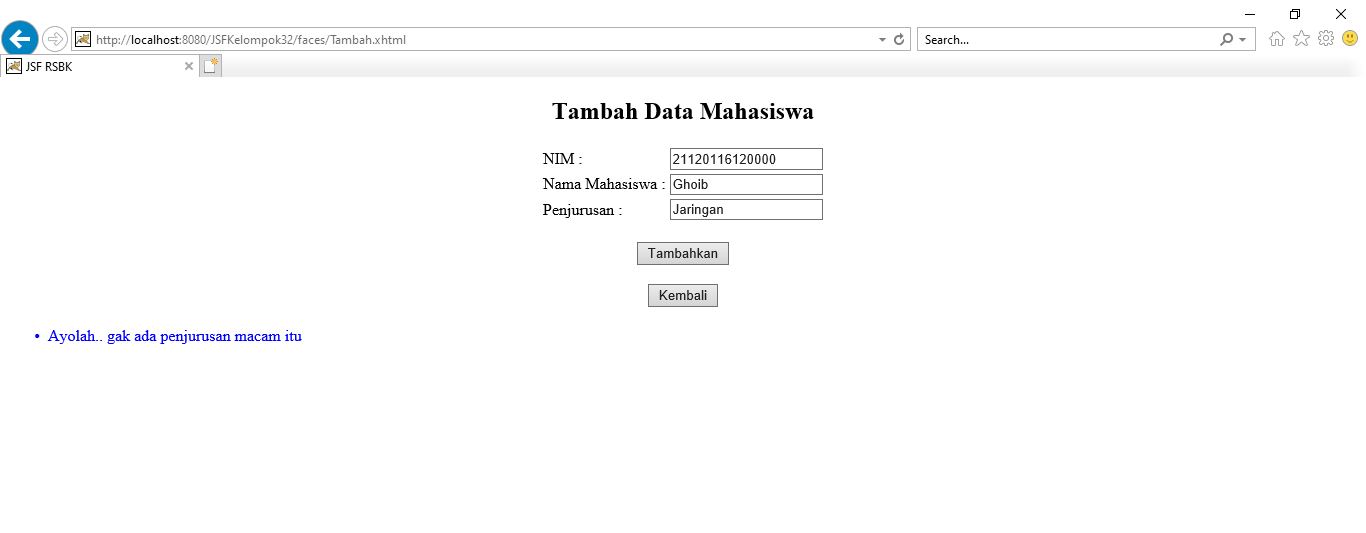
Gambar 5.46 Hasil tugas halaman index

Berikut ini adalah 20 lebih daftar mahasiswa yang sudah di tambahkan dengan fitur tambah sesuai tugas.



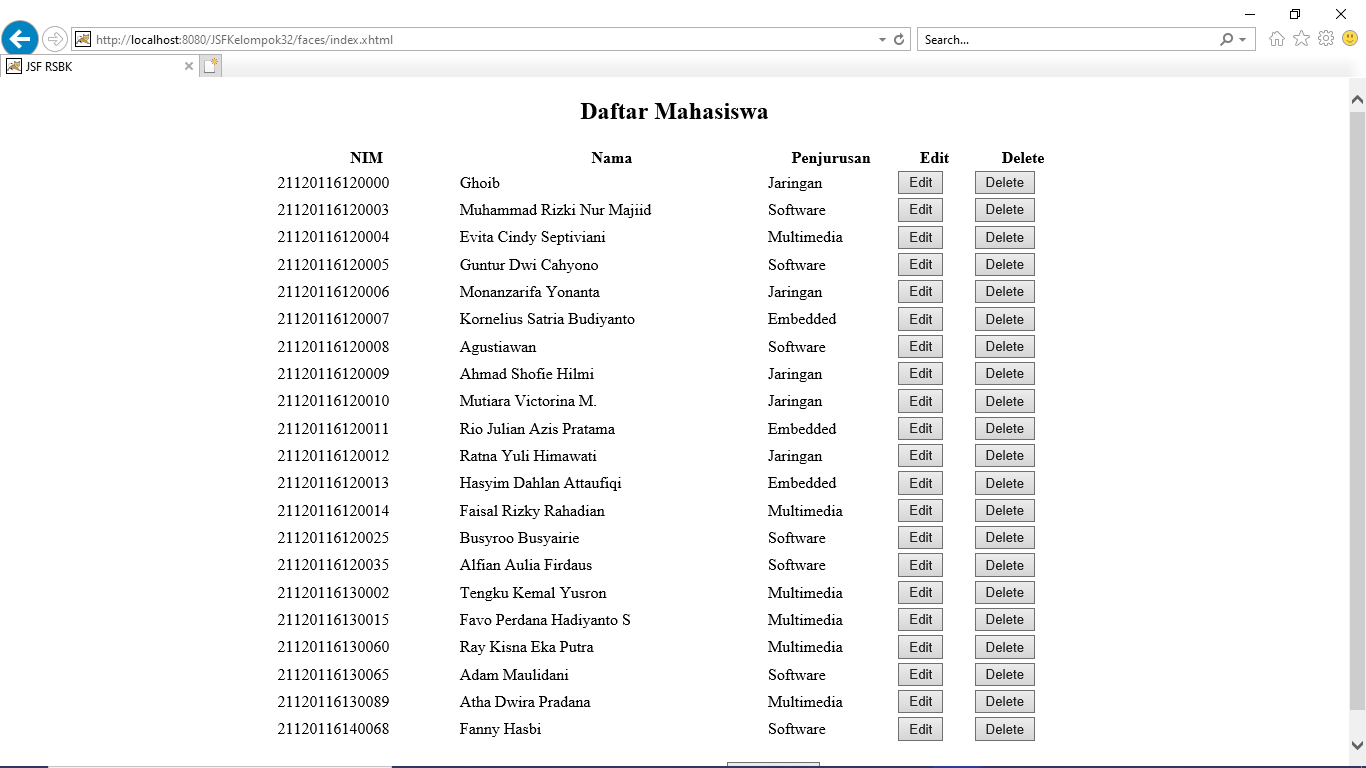
Gambar 5.47 Hasil tugas pesan tambah mahasiswa

Tampilan diatas adalah peringatan ketika menambahkan mahasiswa dengan penjurusan yang salah, maka akan keluar peringatan seperti gambar di atas.



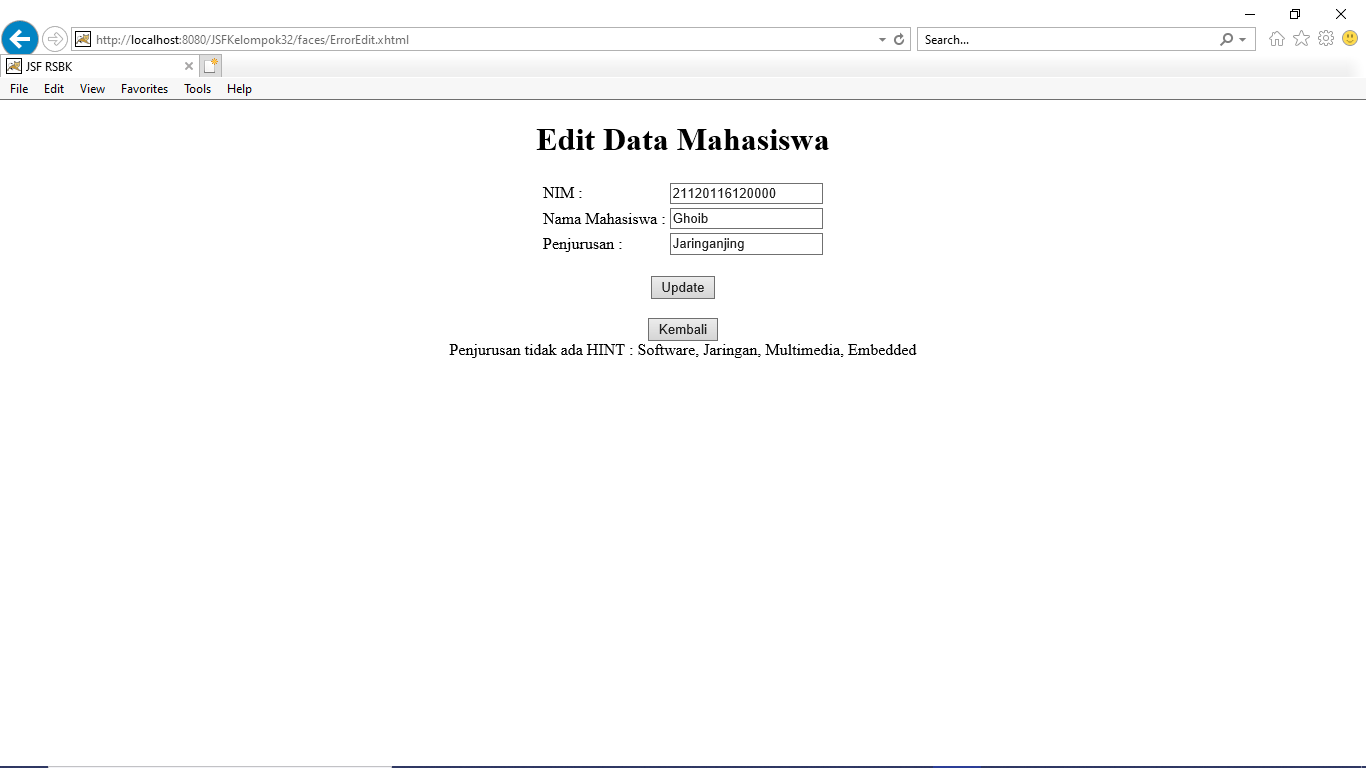
Gambar 5.48 Hasil tugas tambah mahasiswa berhasil

Namun jika menambahkan dengan penjurusan yang benar, maka data akan langsung tertambah ke basis data.



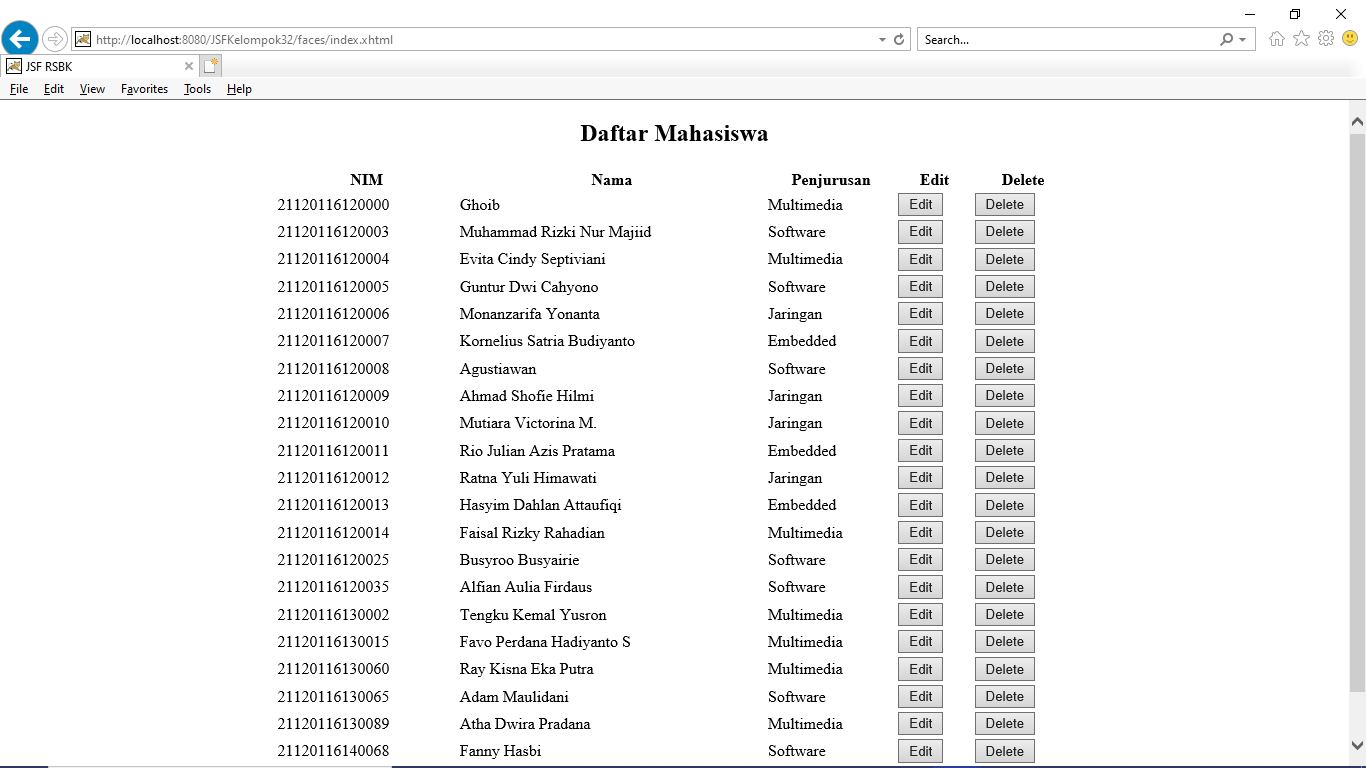
Gambar 5.49 Hasil tugas data tertambah

Data dengan nama Ghoib sudah di tambahkan dan akan langsung di alihkan ke halaman index, sedangkan jika salah maka akan tetap di halaman tambah.



Gambar 5.50 Hasil tugas pesan error edit mahasiswa

Ini adalah screenshot ketika mengubah data mahasiswa dengan penjurusan yang salah, maka akan di tampilkan pesan dengan hint di bagian bawah.



Gambar 5.51 Hasil tugas edit data berhasil

Jika data di edit dengan data yang benar, maka data dapat di ubah dan dialihkan ke halaman index.

## 5.6 Kesimpulan

1. *Connector* MySQL digunakan untuk membangun suatu koneksi antara program dengan *database* MySQL.
2. Untuk menambahkan *connector* MySQL dilakukan dengan cara menambahkannya pada *library* program.
3. Pada JSF xhtml untuk menampilkan *table* dilakukan dengan cara <dataTable></dataTable>.
4. Bootstrap CSS dapat diimplementasikan pada program dengan cara menambahkannya pada *folder* *resource* kemudian untuk memanggilnya dengan cara <h:openStylesheet>.
5. File JSF *Managed Bean* berguna sebagai *back-end* yang mana pada *file* ini terdapat *method* untuk menjalankan *query* CRUD.
6. Untuk menampilkan suatu *join* dapat dilakukan dengan cara membuat *method* dengan *query* *join*.
7. Fungsi dari required = true agar input teks tidak boleh dibiarkan kosong.
8. Untuk menambahkan pesan peringatan ketika *input* teks kosong yaitu dapat dilakukan dengan fungsi requiredMessage =” “.