

# **LAPORAN PRAKTIKUM PEMROGRAMAN WEB**

## **PRAKTIKUM 14 – CRUD**



Disusun oleh:

Nama : Afif Imam Rahadi

Nim : L0122006

Kelas : A

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA**

**UNIVERSITAS SEBELAS MARET**

**2024**

## 1. Screenshot Source Code

### Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Student extends Model
{
    use HasFactory;
    protected $fillable =
    [
        'nama', 'nim', 'prodi', 'kelas', 'angkatan'
    ];
}
```

Code di atas adalah sebuah model di Laravel yang mewakili tabel students dalam database. Model ini terletak dalam namespace App\Models, yang menunjukkan bahwa model ini berada dalam direktori app/Models dari aplikasi Laravel. Dua kelas dari framework Laravel diimpor, yaitu HasFactory dan Model. HasFactory digunakan untuk mendukung pembuatan instance model menggunakan factory, yang berguna untuk pengujian dan pengisian data tiruan, sementara Model adalah kelas dasar yang menyediakan fungsionalitas ORM (Object-Relational Mapping) Eloquent di Laravel. Kelas Student didefinisikan dan mewarisi semua fungsionalitas dari kelas Model, menjadikannya model Eloquent yang dapat digunakan untuk berinteraksi dengan tabel students dalam database.

Model ini menggunakan trait HasFactory, memberikan kemampuan untuk menggunakan factory dalam pembuatan data tiruan atau pengujian. Atribut fillable didefinisikan sebagai array yang berisi daftar kolom-kolom yang diizinkan untuk pengisian massal (mass assignment), yaitu nama, nim, prodi, kelas, dan angkatan. Ini bertujuan untuk mencegah serangan yang dikenal sebagai Mass Assignment Vulnerability, dengan memastikan hanya kolom-kolom yang diizinkan saja yang dapat diisi melalui metode mass assignment. Secara keseluruhan, model Student ini berfungsi sebagai representasi dari tabel students dalam database, memungkinkan interaksi yang mudah dengan data mahasiswa melalui ORM Eloquent Laravel.

## Route

```
<?php

use App\Http\Controllers\StudentController;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
}) -> name('welcome');

Route::get('/students', [StudentController::class, 'index']) -> name('students');

Route::get('/students/form', function () {
    return view('student.form');
});

Route::post('/students/add', [StudentController::class, 'store']) -> name('add-students');
Route::get('/students/delete/{id}', [StudentController::class, 'destroy']);
Route::get('/students/edit/{student}', [StudentController::class, 'edit']);
Route::post('/students/update/{student}', [StudentController::class, 'update']) -> name('students.update');
```

Code ini adalah konfigurasi rute (routes) untuk aplikasi Laravel yang mendefinisikan bagaimana berbagai URL dalam aplikasi tersebut dipetakan ke controller dan view. Baris pertama menggunakan use untuk mengimpor StudentController dari App\Http\Controllers dan Route dari Illuminate\Support\Facades. Rute pertama menggunakan metode get untuk mendefinisikan rute dasar '/', yang mengembalikan view welcome saat diakses dan menamainya welcome. Rute kedua juga menggunakan metode get untuk mendefinisikan URL /students, yang memetakan ke metode index dari StudentController dan menamainya students.

Selanjutnya, rute dengan URL /students/form mengembalikan view student.form tanpa menggunakan controller. Rute post dengan URL /students/add memetakan ke metode store dari StudentController dan menamainya add-students, yang biasanya digunakan untuk menambahkan data baru ke dalam database. Rute get dengan URL /students/delete/{id} memetakan ke metode destroy dari StudentController yang digunakan untuk menghapus data berdasarkan id yang diberikan. Rute berikutnya adalah untuk URL /students/edit/{student}, yang memetakan ke metode edit dari StudentController untuk mengedit data berdasarkan entitas student yang diberikan. Terakhir, rute post dengan URL /students/update/{student} memetakan ke metode update dari StudentController dan menamainya students.update, yang biasanya digunakan untuk memperbarui data yang ada. Secara keseluruhan, rute-rute ini mengatur navigasi aplikasi, menghubungkan URL dengan tindakan yang sesuai di controller dan view, memungkinkan interaksi pengguna dengan data mahasiswa.

## Student Controller

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Student;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\DB;
8
9 class StudentController extends Controller
10 {
11     /**
12      * Display a listing of the resource.
13      */
14     public function index()
15     {
16         $students = DB::table('students')->get();
17         return view('student.index', ['students' => $students]);
18     }
19
20     /**
21      * Show the form for creating a new resource.
22      */
23     public function create()
24     {
25         //
26     }
27
28     /**
29      * Store a newly created resource in storage.
30      */
31     public function store(Request $request)
32     {
33         $request->validate([
34             'nama' => 'required',
35             'nim' => 'required',
36             'prodi' => 'required',
37             'kelas' => 'required',
38             'angkatan' => 'required',
39         ]);
40
41         $student = new Student;
42         // kiri objek model - kanan dari request
43         $student->nama = $request->nama;
44         $student->nim = $request->nim;
45         $student->prodi = $request->prodi;
46         $student->kelas = $request->kelas;
47         $student->angkatan = $request->angkatan;
48
49         $student->save();
50
51         return redirect()->route('students')->with('success', 'Data mahasiswa berhasil ditambahkan');
52     }
53
54     /**
55      * Display the specified resource.
56      */
57     public function show(Student $student)
58     {
59         //
60     }
61
62     /**
63      * Show the form for editing the specified resource.
64      */
65     public function edit(Student $student)
66     {
67         return view('student.edit', ['student' => $student]);
68     }
69
70     /**
71      * Update the specified resource in storage.
72      */
73     public function update(Request $request, Student $student)
74     {
75         $request->validate([
76             'nama' => 'required',
77             'nim' => 'required',
78             'prodi' => 'required',
79             'kelas' => 'required',
80             'angkatan' => 'required',
81         ]);
82
83         $student->update($request->all());
84         return redirect()->route('students')->with('success', 'Data berhasil diupdate');
85     }
86
87     /**
88      * Remove the specified resource from storage.
89      */
90     public function destroy($id)
91     {
92         $student = DB::table('students')->where('id', $id)->delete();
93         return redirect()->route('students');
94     }
95 }
```

Code ini adalah definisi dari StudentController di Laravel yang mengatur logika aplikasi terkait operasi CRUD (Create, Read, Update, Delete) untuk model Student. Controller ini berada di dalam namespace App\Http\Controllers yang menunjukkan bahwa file ini berada di direktori app/Http/Controllers. Beberapa kelas diimpor di bagian awal, termasuk Student dari App\Models, Request dari Illuminate\Http\Request, dan DB dari Illuminate\Support\Facades.

Metode index dalam controller ini mengambil semua data dari tabel students menggunakan query builder `DB::table('students')->get()`, dan mengirimkannya ke view `student.index`. Metode `create` didefinisikan tetapi kosong, menunjukkan bahwa form untuk membuat resource baru belum diimplementasikan. Metode `store` menerima Request dari form, memvalidasi input untuk memastikan semua kolom wajib diisi (nama, nim, prodi, kelas, dan angkatan), kemudian membuat instance baru dari `Student` dan menyimpan data yang valid ke dalam database. Setelah data berhasil disimpan, pengguna akan dialihkan ke route `students` dengan pesan sukses.

Metode `show` didefinisikan tetapi kosong, biasanya digunakan untuk menampilkan detail dari satu resource `Student`. Metode `edit` menerima instance `Student` sebagai parameter, mengembalikan view `student.edit` dengan data mahasiswa yang akan diedit. Metode `update` menerima Request dan instance `Student`, memvalidasi input, kemudian memperbarui data mahasiswa yang ada dengan data baru dari request. Setelah berhasil memperbarui data, pengguna akan dialihkan kembali ke route `students` dengan pesan sukses.

Metode `destroy` menerima parameter `$id`, menghapus data mahasiswa dari tabel `students` berdasarkan id yang diberikan, kemudian mengarahkan pengguna kembali ke route `students`. Metode ini menggunakan query builder untuk operasi penghapusan. Secara keseluruhan, controller ini menangani semua operasi dasar CRUD untuk model `Student`, memastikan data dapat dibuat, dibaca, diperbarui, dan dihapus sesuai kebutuhan aplikasi.

## Migration

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('students', function (Blueprint $table) {
            $table->id();
            $table->string('nama');
            $table->string('nim');
            $table->string('prodi');
            $table->char('kelas');
            $table->integer('angkatan');
            $table->timestamps();
        });
    }
}
```

Kode ini adalah definisi migrasi database di Laravel yang digunakan untuk membuat dan menghapus tabel `students`. Migrasi ini terletak dalam namespace `Illuminate\Database\Migrations` dan menggunakan beberapa kelas dari Laravel termasuk `Blueprint` dan `Schema`. Kode ini didefinisikan dalam sebuah anonymous class yang mewarisi `Migration`.

Metode up didefinisikan untuk menjalankan migrasi, di mana Schema::create digunakan untuk membuat tabel students. Di dalam fungsi ini, sebuah instance Blueprint diberikan sebagai parameter untuk mendefinisikan struktur tabel. Tabel students memiliki beberapa kolom: id yang merupakan primary key dengan auto-increment, nama dan nim yang keduanya adalah string, prodi yang juga merupakan string, kelas yang merupakan karakter (char), angkatan yang merupakan integer, dan dua kolom timestamp (created\_at dan updated\_at) yang secara otomatis dikelola oleh Laravel. Metode down didefinisikan untuk membalikkan migrasi, yang dalam hal ini menghapus tabel students jika ada. Metode ini menggunakan Schema::dropIfExists untuk memastikan bahwa tabel students dihapus dengan aman.

## Welcome Views

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>welcome</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QwTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRj" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLEsaAA55NDzOxhy9GkcIdslK1eN7" crossorigin="anonymous"></script>
</head>
<body>
  <div class="container centered">
    <h1>Selamat Datang Dalam Web Pendataan Mahasiswa</h1>
    <div class="mt-4">
      <button class="btn btn-primary mx-2">
        <a href="/students" class="text-light" style="text-decoration: none;">List Daftar Mahasiswa</a>
      </button>
      <button class="btn btn-primary mx-2">
        <a href="/students/form" class="text-light" style="text-decoration: none;">Tambah Data Mahasiswa</a>
      </button>
    </div>
  </div>
</body>
</html>
```

Pada bagian welcome ini, merupakan teks html biasa yang menggunakan bootstrap sebagai bantuan styling. Pada halaman ini hanya sebagai halaman utama sebelum user dapat melihat dan mengisi data mahasiswa. Di halaman ini, terdapat 2 button yang mengarah ke halaman yang berbeda.

### Student/index view

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tampil Data</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXhK" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9FVDZLEAAas95o6VqLkewpz/rNBZ0IpYhtLOszAPnCQHdQ" crossorigin="anonymous">
  <style type="text/css">
    .container
    {
      width: 100%;
      margin: auto;
    }
    body
    {
      margin: 8px;
      padding: 10px;
    }

    .tambah-data{
      width: 130px;
      margin-left: 187px;
    }
  </style>
</head>
<body>
  <div class="container">

    <h2 class="my-4" style="text-align:center;">Data Mahasiswa</h2>
    <table class="table table-striped table-bordered">
      <tr bgcolor = "beige" align="center">
        <th>No</th>
        <th>Nama</th>
        <th>NIM</th>
        <th>Poli</th>
        <th>Kelask</th>
        <th>Angkatan</th>
        <th>Aksi</th>
      </tr>

      @foreach ($students as $index => $student)
      <tr align=center bgcolor = #FFDAB9>
        <th>{{ $index + 1 }}</th>
        <td>{{ $student->nama }}</td>
```

Ini adalah halaman utama dari students. Halaman utama ini akan menampilkan data mahasiswa yang berbentuk tabel. Di dalam tabel ini terdapat data-data yang telah diinputkan oleh user. Pada setiap data terdapat aksi yang dapat dilakukan oleh user, yaitu edit dan delete. Oleh karena itu, user dapat mengatur data sesuai yang diinginkan.

## Student/form View

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QMTKZyypPjEJ5v5WU90F0Rpok6YctnYmDr5pNlyT2bRjXh"
  <title style="text-align: center;">Form Data Mahasiswa</title>
</head>
<body>
  <h1 class="my-4 mx-3">Form Data Mahasiswa</h1>
  <div class="container w-75 m-4 mx-5" >
    <form method="POST" action={{ route('add-students')}}>
      @csrf
      <div>
        <div>
          <label class="input-group-text w-25">Nama</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="nama" size="30" placeholder="Masukkan nama anda" >
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">NIM</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="nim" size="30" placeholder="Masukkan nim anda"></input>
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Prodi</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="prodi" size="30" placeholder="Masukkan prodi anda">
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Kelas</label>
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

Ini adalah code halaman dari form data mahasiswa. Di sini user dapat mengisi data sesuai dengan tempatnya. Terdapat tag form agar lebih terstruktur untuk user input. Setiap form sudah memiliki name masing-masing yang nantinya akan diproses untuk dimasukkan ke dalam database.

### Student/edit view

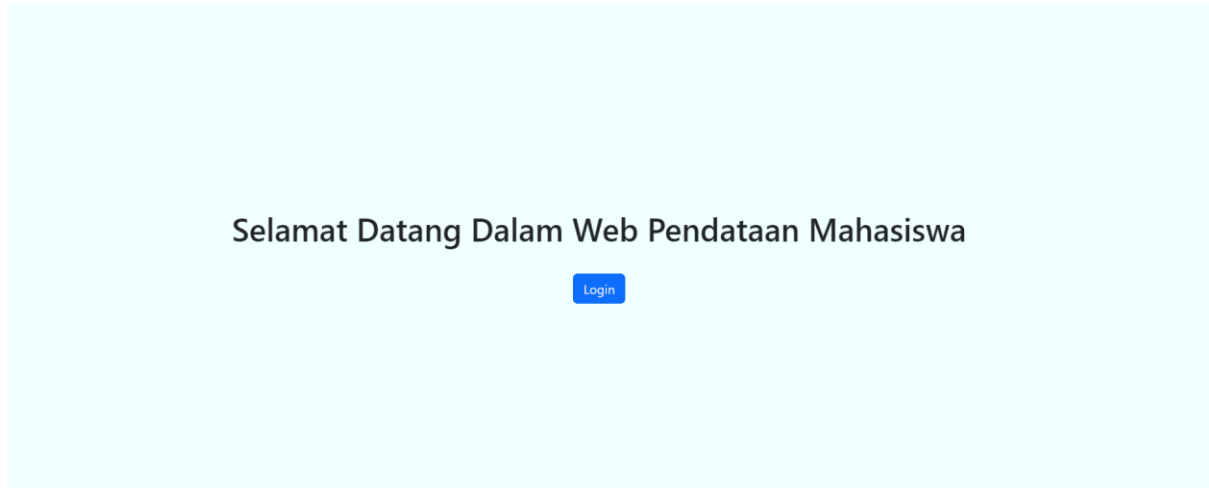
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QMTKZyypPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh4" crossorigin="anonymous">
  <title style="text-align: center;">Edit Form Data Mahasiswa</title>
</head>
<body>
  <h1 class="my-4 mx-3">Edit Form Data Mahasiswa</h1>
  <div id="mainContainer" class="container w-75 m-4 mx-5">
    <form method="POST" action={{ route('students.update', $student -> id )}}>
      @csrf
      <div>
        <div>
          <label class="input-group-text w-25">Nama</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="nama" value="{{ $student->nama }}" size="30" >
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">NIM</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="nim" value="{{ $student->nim }}" size="30"></input>
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Prodi</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="prodi" value="{{ $student->prodi }}" size="30">
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Kelas</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="kelas" value="{{ $student->kelas }}" size="30">
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

Tidak berbeda jauh dengan views form. Yang membedakannya adalah pada halaman edit ini, pada bagian form sudah memiliki value masing-masing yang diambil dari databasenya. Sehingga saat user ingin melakukan edit, sudah ada data sebelumnya sehingga tidak bingung untuk mengganti yang bagian mana.



## 2. Screenshot Ouput

### Home



Ini adalah halaman home. Halaman home sangat sederhana yang berisi tulisan selamat datang. Di bawahnya, terdapat button login. Jika button ini di klik, maka program akan mengarahkannya ke halaman login. Pada halaman utama, hanya ada button login yang berarti semua user untuk melihat data dan melakukan CRUD harus melakukan login/register.

### Login

**Login**

Username

|

Password

Login

Didn't Have An Account? [Register](#)

Ini adalah halaman login, halaman login terdiri dari username dan password. Pada bagian username terdapat autofocus yang membuat user langsung dapat mengisi field username secara langsung.

## Register

### Register

Name
Username
Email address
Password
Register

Already Have An Account? [Login](#)

Ini adalah halaman register yang terdiri dari 4 inputan berupa nama, username, email, dan password. Pada bagian ini, user akan melakukan register agar masuk ke dalam database.

## Index

### Data Mahasiswa

No	Nama	NIM	Prodi	Kelas	Angkatan	Aksi
Tambah Data	Logout					

Ini adalah halaman utama dari data mahasiswa. Sama seperti sebelumnya yang menampilkan data nama, nim, prodi, kelas, angkatan, dan terdapat aksi untuk melakukan edit dan delete. Pada bagian bawah terdapat button untuk melakukan penambahan data dan juga melakukan logout. Jika button logout dipencet, maka akan masuk ke halaman awal yang berupa judul.

## Student/Form

### Form Data Mahasiswa    Form Data Mahasiswa

Nama	Afif Imam Rahadi
NIM	L0122006
Prodi	Informatika
Kelas	A
Angkatan	2022
Tambah Data	

Nama	Masukkan nama anda
NIM	Masukkan nim anda
Prodi	Masukkan prodi anda
Kelas	Masukkan kelas anda
Angkatan	Masukkan angkatan anda
Tambah Data	

Ini adalah halaman form. Halaman form ini dapat digunakan saat user memencet tombol tambah data. Di dalam form ini, terdapat baris-baris kolom yang bisa diisi. Pada gambar kanan, terdapat form awal sebelum diisi dan di dalamnya sudah terdapat placeholder yang dapat membantu user dalam mengisi data. Dan sebelah kiri, merupakan hasil setelah user menuliskan datanya. Di bawahnya terdapat button untuk tambah data.

## Student/edit

### Edit Form Data Mahasiswa

Nama	Afif Imam Rahadi
NIM	L0122006
Prodi	Informatika
Kelas	A
Angkatan	2022
Edit Data	

Bentuk dari edit ini sama seperti form pengisian data. Pada saat user memencet tombol edit. Maka tampilan awal akan seperti di atas yaitu sesuai dengan data yang diinputkan. Jadi, jika user ingin mengubah salah satu data, user hanya tinggal menggantikan bagian data yang ingin diganti.

## Tampilan setelah ditambah data

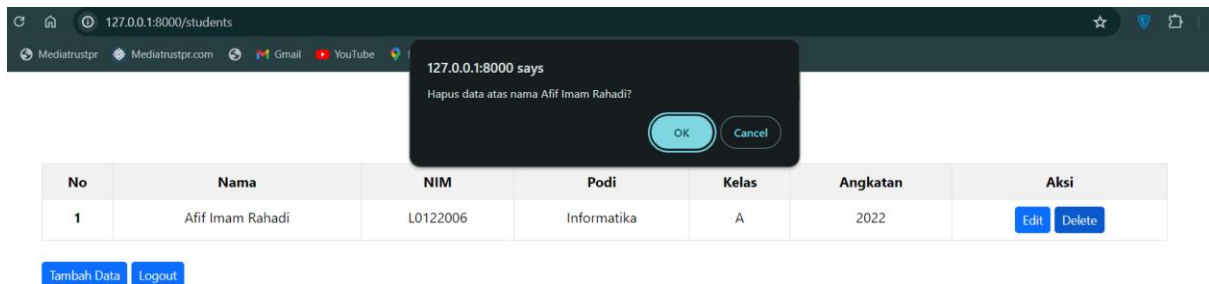
### Data Mahasiswa

No	Nama	NIM	Podi	Kelas	Angkatan	Aksi
1	Afif Imam Rahadi	L0122006	Informatika	A	2022	<a href="#">Edit</a> <a href="#">Delete</a>

[Tambah Data](#) [Logout](#)

Ini adalah tampilan setelah user menambahkan data. Tampilan berupa table yang berisi data-data yang diinputkan oleh user. Dari data ini, user dapat edit ataupun delete.

## Delete Data



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/students". A confirmation dialog box is overlaid on the page, asking "Hapus data atas nama Afif Imam Rahadi?" with "OK" and "Cancel" buttons. Below the dialog is a table with student data.

No	Nama	NIM	Podi	Kelas	Angkatan	Aksi
1	Afif Imam Rahadi	L0122006	Informatika	A	2022	<a href="#">Edit</a> <a href="#">Delete</a>

[Tambah Data](#) [Logout](#)

Ini adalah tampilan saat ingiin menghapus data. Pada saat ingin menghapus, terdapat alert yang diikuti oleh pesan nama data yang ingin dihapus. Untuk menghapus, user hanya tinggal klik OK.