

LAPORAN AKHIR
MATAKULIAH PRAKTIKUM PEMROGRAMAN FRAMEWORK
SISTEM INFORMASI AKADEMIK SMA 70 SAMARINDA



Disusun Oleh:
KELOMPOK 1
Heri Aditia / 1515015130
Fetrisye Delp Parenden / 1515015138
Ade Chrisvitandy / 1515015141
Indra Jambia / 1515015179

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI DAN KOMUNIKASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS MULAWARMAN
2017

KATA PENGANTAR

Puji Syukur kehadiran ALLAH SWT, karena atas perkenanNYA laporan akhir praktikum Framework dapat diselesaikan.

Makalah ini merupakan salah satu syarat untuk mengikuti Ujian Akhir guna untuk mendapatkan nilai yang baik di mata kuliah Framework. Selesaiannya penyusunan laporan akhir ini berkat bantuan dari berbagai pihak oleh karena itu, pada kesempatan ini kami sampaikan terimakasih kepada:

1. Bapak Hario Jati Setyadi M.Kom selaku dosen Framework yang telah meluangkan waktu, tenaga dan pikiran dalam pelaksanaan bimbingan, pengarahan, dorongan dalam rangka penyelesaian penyusunan laporan ini.
2. Yasmine Nabillah Sulaiman dan Rizky Ariesta selaku asisten laboratorium kelompok praktikum Kecerdasan Buatan
3. Rekan-rekan sesama mahasiswa yang telah memberikan masukan dan bantuan dalam pengerjaan dan laporan.

Semoga Allah SWT, memberikan balasan atas kebaikan yang telah diberikan kepada kami. kami menyadari masih banyak kekurangan dari laporan ini, baik dari materi maupun teknik penyajiannya, mengingat kurangnya pengetahuan dan pengalaman kami. oleh karena itu, kritik dan saran yang membangun sangat kami harapkan.

Samarinda, 26 Mei 2016

Penyusun

TAKARIR

<i>Database</i>	basis data
<i>Export</i>	Kirim ke luar
<i>Entity Relationship</i>	Relasi entitas
<i>Import</i>	Ambil dari luar
<i>Level</i>	tingkat
<i>Many to Many</i>	Banyak ke banyak
<i>One to One,</i>	Satu ke satu
<i>One to Many</i>	Satu ke banyak
<i>Project</i>	proyek
<i>request</i>	permintaan
<i>screenshot</i>	hasil tampilan
<i>source code</i>	kode program
<i>User</i>	pengguna
<i>view</i>	Lihat

DAFTAR ISI

KATA PENGANTAR	i
TAKARIR	ii
DAFTAR ISI	iii
DAFTAR GAMBAR	iv
BAB I PENDAHULUAN	1
1.1 Deskripsi Masalah	1
1.2 Rumusan Masalah	1
1.3 Batasan Masalah.....	2
1.4 Tujuan.....	2
BAB II PERANCANGAN.....	3
2.1 Analisis Program	3
2.2 <i>Entity Relationship</i> Diagram	3
2.3 Konsep/Materi Praktikum yang dipakai.....	4
BAB III HASIL DAN PEMBAHASAN.....	6
3.1 Implementasi Program	6
3.2 Source Code	9
BAB IV KESIMPULAN DAN SARAN	28
1.1 Kesimpulan.....	28
1.2 Saran.....	28
DAFTAR PUSTAKA	29

DAFTAR GAMBAR

GAMBAR 2.1 <i>Entity Relationship Diagram</i>
GAMBAR 3.1 Index

BAB I

PENDAHULUAN

1.1 Deskripsi Masalah

Bangsa Indonesia adalah salah satu negara yang memiliki jumlah penduduk yang tinggi. Dengan tingginya jumlah penduduk di Indonesia maka mempengaruhi beberapa aspek kehidupan di Indonesia, salah satunya adalah pendidikan. Untuk menjadikan pendidikan di Indonesia menjadi lebih baik, maka dibangunlah sekolah-sekolah yang diharapkan dapat memenuhi kebutuhan masyarakat Indonesia dalam menimba ilmu dan menjadikan bangsa Indonesia lebih baik.

Namun, pada kenyataannya jumlah sekolah yang ada dan jumlah murid yang terdaftar di sekolah tersebut memiliki perbandingan yang cukup jauh khususnya di daerah perkotaan. Dimana satu sekolah dapat memiliki ribuan murid. Banyaknya jumlah murid yang harus di data pihak sekolah khususnya pendataan nilai akademik murid membuat pihak sekolah kewalahan jika tidak ditangani dengan benar. Dan juga pada zaman ini, masih banyak orang tua yang tidak dapat melihat atau mengikuti perkembangan pendidikan anaknya disekolah.

Melihat permasalahan ini, maka kami mencoba membuat program yang dapat membantu pihak sekolah melakukan pengolahan data nilai murid dan dapat membantu pihak orang tua dapat melihat perkembangan nilai anak-anaknya dengan mudah.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah, maka penulis merumuskan permasalahan yang akan digunakan untuk menyelesaikan tugas akhir ini yaitu:

1. Bagaimana cara membantu pihak sekolah mendata data-data nilai murid pada sekolah tersebut dengan mudah dan cepat?
2. Bagaimana cara membantu pihak wali murid dalam mengawasi perkembangan akademik anaknya?

1.3 Batasan Masalah

Sistem yang akan diimplementasikan memiliki beberapa batasan masalah sebagai berikut:

1. Hak akses Guru bentrok dengan hak akses Admin, jadi Guru tidak bisa melakukan CRUD.
2. Wali murid tidak bisa melakukan CRUD.

2.4 Tujuan

Berdasarkan rumusan masalah yang dipaparkan diatas, maka program ini mempunyai tujuan, yaitu:

1. Memudahkan pihak sekolah dalam mendata nilai murid pada sekolah tersebut dengan mudah dan cepat.
2. Memudahkan pihak wali murid dalam mengawasi perkembangan akademik anaknya.

BAB II

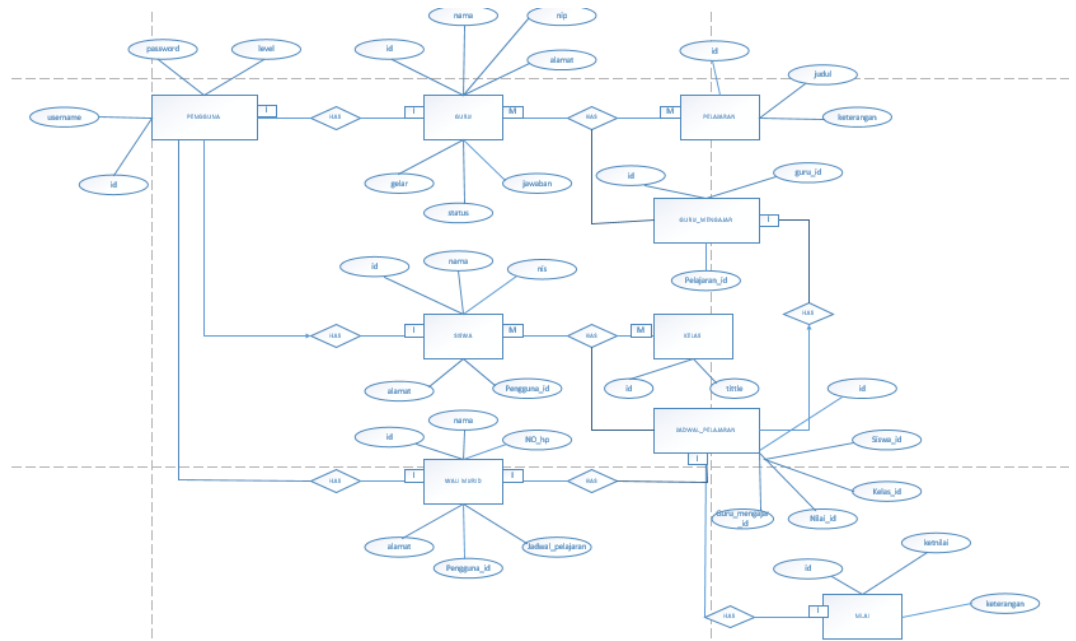
PERANCANGAN

2.1 Analisis Program

Program kami dibuat untuk memudahkan guru dan admin dalam mengelola data-data murid dan membantu wali murid dalam mengawasi perkembangan akademis anak mereka. Maka program kami memiliki 4 *level* user yaitu admin, guru, murid dan wali murid yang dapat menggunakan program kami.

Untuk menggunakan program ini, pertama-tama user harus login untuk mendapatkan hak akses mereka sesuai level mereka. Dimana untuk level admin, diberikan hak akses untuk mengakses semua data. Lalu guru mempunyai hak akses untuk mengakses semua data kecuali data pengguna dan data kelas karena yang dapat mengatur pengguna dan kelas hanya admin. Kemudian murid mempunyai hak akses untuk melihat nilai, jadwal pelajaran, dan guru mengajar. Sedangkan wali murid hanya diberikan hak akses untuk melihat nilai murid.

2.2 Entity Relationship Diagram



Gambar 2.1 Entity Relationship Diagram

2.3 Konsep/Materi Praktikum yang dipakai

2.3.1 Migration

Migration adalah sebuah fasilitas di Laravel yang digunakan untuk mempermudah kita ketika ada perubahan dalam database. Schema Builder digunakan untuk membuat sebuah skema *database*. Dengan menggunakan migration dan schema builder kita tidak perlu repot-repot membuka phpmyadmin, ataupun aplikasi *Sql* lainnya untuk membuat database. Dengan migrations dan schema builder juga akan lebih mudah ketika kita membuat *project* besar dan dengan developer yang berbeda-beda jadinya si developer ini tidak perlu *import* atau *export* database tapi langsung saja menggunakan migration.

2.3.2 Routing

Routing adalah proses dimana suatu item dapat sampai ke tujuan dari satu lokasi ke lokasi lain. Dalam hal framework Laravel, item yang dimaksud adalah halaman website. Para developer Laravel dapat menentukan sendiri halaman yang akan muncul pada saat dikunjungi oleh User. Misalnya User mengunjungi halaman dashboard, maka kita dapat menentukan tampilan apa yang akan muncul, apakah itu hanya berupa tulisan, berupa halaman controller, berupa halaman view, maupun halaman error. Route dapat handle semua perintah yang telah dideklarasikan oleh kita.

2.3.3 Model, Mass Assignment & Controller

Model digunakan untuk query atau proses mengambil data dari database. Mass assignment adalah fungsi yang digunakan untuk menyimpan data ke dalam database. Controller adalah file yang nantinya akan handle semua logika dari program, dari controller nantinya akan menyuruh model untuk mengambil data

dari database, dan juga meminta view untuk menampilkan data yang telah di olah kepada user.

2.3.4 MVC & Blade

Aplikasi Laravel mengikuti arsitektur design pattern MVC yang tradisional, di mana menggunakan :

- Controller untuk handle *request* user dan mengambil data dengan memanfaatkan Model,
- Model untuk berinteraksi dengan database dan mengambil informasi dari objek,
- View untuk render halaman.

Blade adalah templating engine yang merupakan bawaan dari Laravel, tidak seperti templating engine dari Framework lain yang juga populer, Blade tidak memaksakan pengunanya menggunakan kode plain PHP pada View. Semua view Blade akan compile menjadi code plain PHP dan di chace-kan sampai terjadi modifikasi pada file Blade, dan artinya Blade tidak menambah overhead pada aplikasi yang di buat.

2.3.5 Eloquent: Relationship

Tabel database sering berhubungan satu sama lain. Fungsi ini berguna untuk membuat pengelolaan dan bekerja dengan relasi menjadi mudah, dan mendukung beberapa jenis relasi yaitu *One to One*, *One to Many*, dan *Many to Many*

2.3.6 Eloquent: Relationship Reborn

BAB III

HASIL DAN PEMBAHASAN

3.1 Implementasi Program

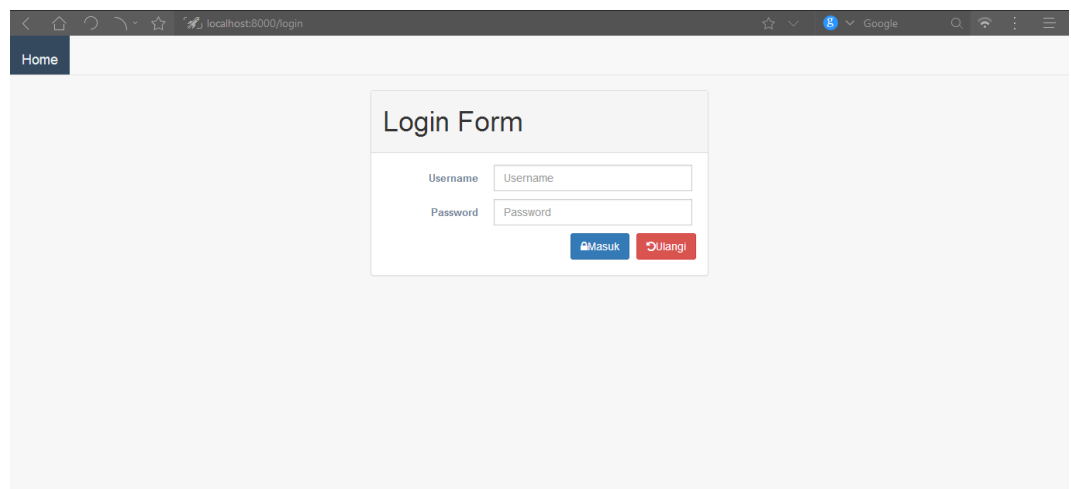
3.1.1 Tampilan Awal



Gambar 3.1 Tampilan Awal

Gambar diatas adalah tampilan awal program dimana terdapat profil tentang SMA 70 Samarinda dan terdapat menu-menu lain. Namun, user harus login terlebih dahulu untuk mengakses.

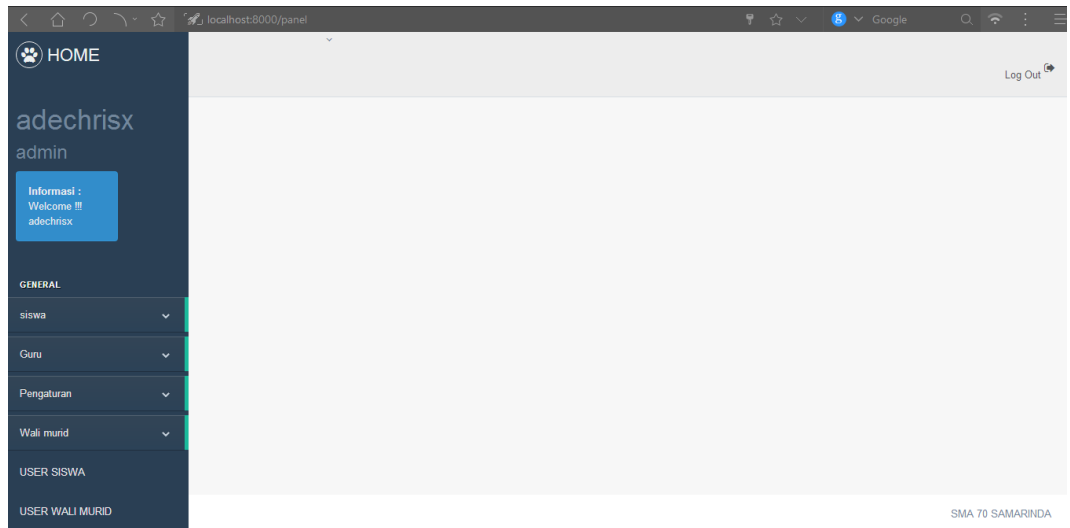
3.1.2 Form Login



Gambar 3.2 Form Login

Gambar diatas tampilan form login program, terdapat field username dan password yang harus diisi user untuk dapat login.

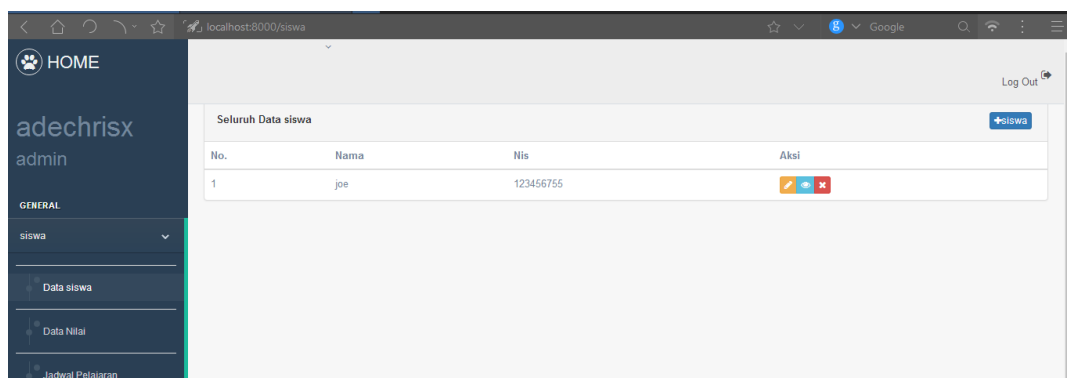
3.1.3 Panel Admin



Gambar 3.3 Panel Admin

Gambar diatas adalah tampilan panel untuk admin jika sebelumnya user login dengan level admin. Seperti yang terlihat pada gambar, admin memiliki semua hak akses.

3.1.4 Menu Siswa

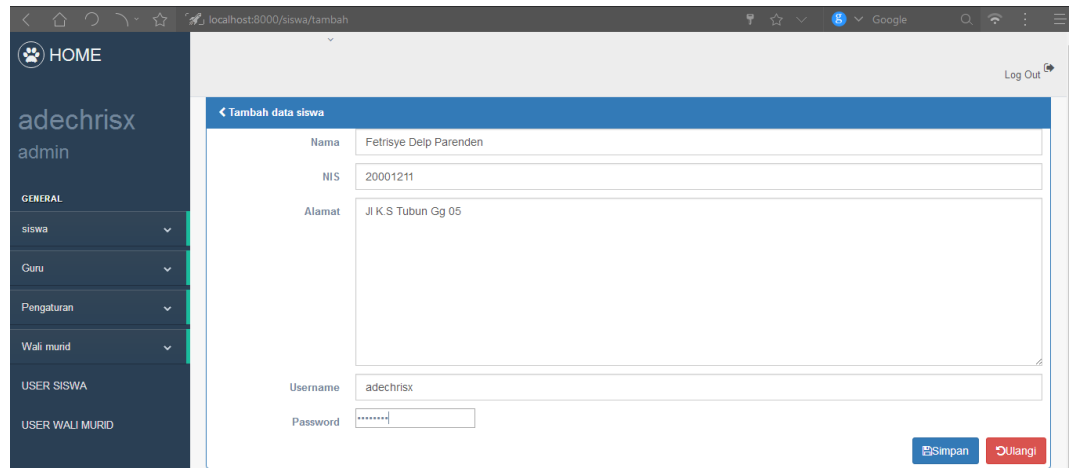


Gambar 3.4 Menu Siswa

Admin memiliki semua hak akses, salah satunya adalah admin dapat mengelola data siswa. Seperti pada gambar, gambar diatas adalah tampilan

ketika admin melihat data siswa. Pada tabel, terdapat pilihan yang dapat admin gunakan untuk mengelola data siswa yaitu tambah, ubah, lihat dan hapus.

3.1.5 Tambah Murid

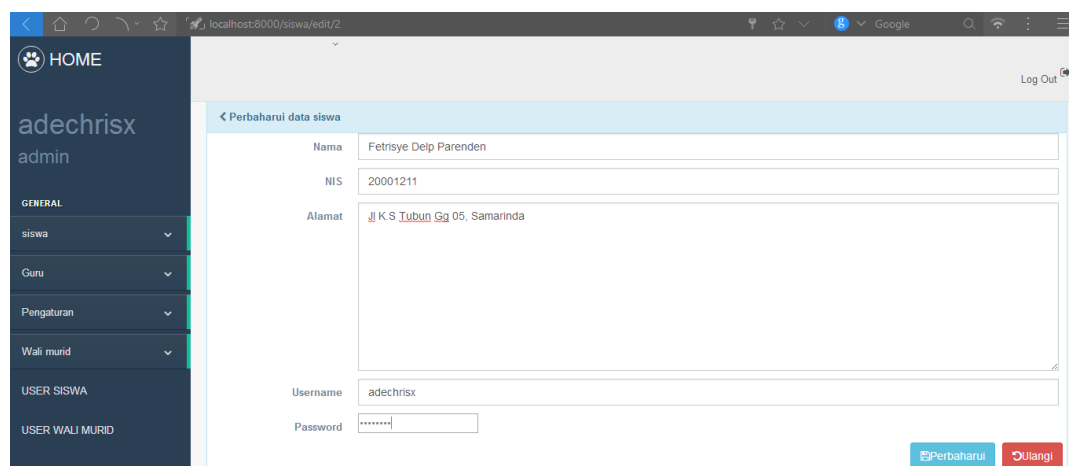


The screenshot shows a web application interface for adding a new student. On the left is a dark blue sidebar with a 'HOME' button and a user profile for 'adechrisx admin'. Below this is a 'GENERAL' section with a list of menu items: 'siswa', 'Guru', 'Pengaturan', 'Wali murid', 'USER SISWA', and 'USER WALI MURID'. The 'siswa' item is currently selected. The main content area has a title 'Tambah data siswa' and a 'Log Out' link. The form contains the following fields: 'Nama' (Fetrisye Delp Parenden), 'NIS' (20001211), 'Alamat' (Jl K.S Tubun Gg 05), 'Username' (adechrisx), and 'Password' (masked with dots). At the bottom right of the form are two buttons: 'Simpan' (Save) and 'Ulangi' (Retry).

Gambar 3.5 Tambah Murid

Gambar diatas adalah tampilan form tambah ketika admin ingin menambahkan siswa baru. Terdapat field Nama, NIS, Alamat, Username dan Password yang harus diisi oleh admin.

3.1.6 Edit Murid

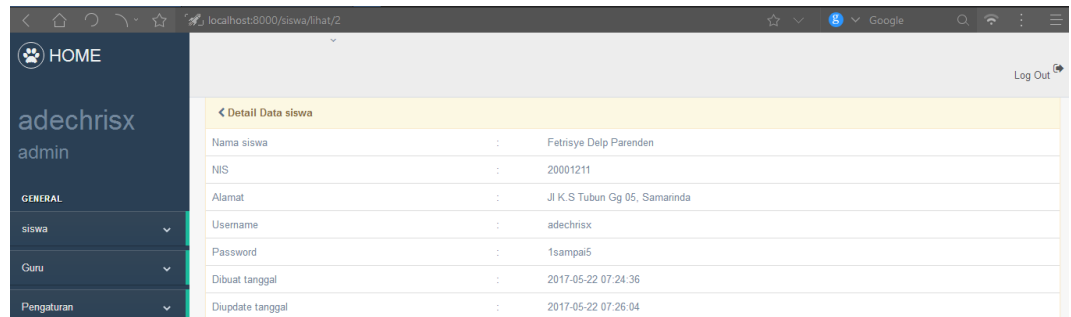


The screenshot shows the same web application interface but for editing an existing student. The title of the form is 'Perbaharui data siswa'. The 'Alamat' field now contains the text 'Jl K.S Tubun Gg 05, Samarinda'. The 'Username' field is 'adechrisx' and the 'Password' field is masked. The 'Simpan' button has been replaced with a 'Perbaharui' (Update) button, and the 'Ulangi' button remains. The sidebar and user profile are identical to the previous screenshot.

Gambar 3.6 Edit Murid

Gambar diatas merupakan tampilan form untuk mengubah data siswa. Sama seperti pada saat menambahkan murid baru, pada saat mengubah juga diharuskan mengisi username dan password.

3.1.7 Lihat Siswa

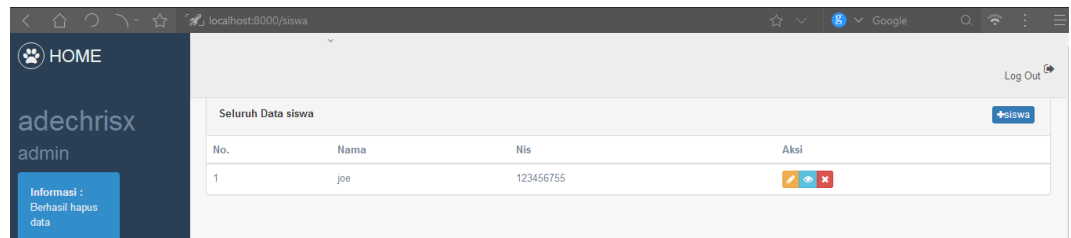





< Detail Data siswa	
Nama siswa	: Fetrisye Delp Parenden
NIS	: 20001211
Alamat	: Jl K.S Tubun Gg 05, Samarinda
Username	: adechrisx
Password	: 1sampai5
Dibat tanggal	: 2017-05-22 07:24:36
Diupdate tanggal	: 2017-05-22 07:26:04

Gambar 3.7 Lihat Siswa

Gambar diatas adalah tampilan data siswa yang barusaja admin ubah. Pada aksi lihat, admin dapat melihat detail data siswa disertai dengan waktu pembuatan dan pembaruan.

3.1.8 Hapus Siswa



Seluruh Data siswa			
No.	Nama	Nis	Aksi
1	joe	123456755	  

Gambar 3.8 Hapus Siswa

Gambar diatas merupakan tampilan ketika salah satu data siswa yaitu siswa dengan nama ‘Fetrisye Delp Parenden’ dihapus.

3.2 Source Code

3.2.1 Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
```

```

use Illuminate\Database\Migrations\Migration;

class BuatTableGuru extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('guru', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nama',50);
            $table->string('nip',18);
            $table->text('alamat');
            $table->string('gelar');
            $table->string('status');
            $table->string('jabatan');
            $table->integer('pengguna_id',false,true)->references('id')-
>on('pengguna')->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()

```

```

    {
        Schema::drop('guru');
    }
}

```

3.2.2 Model

```

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class guru extends Model
{
    protected $table = 'guru';
    protected $fillable =
['nama','nip','alamat','gelar','status','jabatan','pengguna_id'];

    public function pengguna()
    {
        return $this->belongsTo(Pengguna::class);
    }
    public function getUsernameAttribute(){
        return $this->pengguna->username;
    }

    public function guru_mengajar()
    {
        return $this->hasOne(Guru_Mengajar::class);
    }
}

```



```

public function listGuruDanNip(){
    $out = [];
    foreach ($this->all() as $gru) {
        $out[$gru->id] = "{$gru->nama} ({$gru->nip})";
    }
    return $out;
}
}

```

3.2.3 Controller

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;
use App\pengguna;
use App\guru;

class guruController extends Controller
{
    protected $informasi = 'Gagal melakukan aksi';
    public function awal()
    {
        $semuaGuru=guru::all();
        return view('guru.awal', compact('semuaGuru'));
    }
    public function tambah()
    {

```

```

        return view('guru.tambah');
    }

    public function simpan(Request $input)
    {
        $this->validate($input,[
            'username'=>'required |min:6',
            'password'=>'required |min:6',
            'nama'=>'required',
            'nip'=>'required|max:12',
            'alamat'=>'required',
            'gelar'=>'required',
            'status'=>'required',
            'jabatan'=>'required',

        ]);

        $pengguna = new Pengguna($input-
>only('username','password','level'));
        if ($pengguna->save()){
            $guru = new guru;
            $guru -> nama = $input->nama;
            $guru -> nip = $input->nip;
            $guru -> alamat = $input->alamat;
            $guru -> gelar = $input->gelar;
            $guru -> status = $input->status;
            $guru -> jabatan = $input->jabatan;
            if ($pengguna->guru()->save($guru))
                $this->informasi='Berhasil simpan data';
        }

        return redirect('guru')->with(['informasi'=>$this->informasi]);
    }

    public function edit($id){

```

```

        $guru = guru::find($id);
        return view('guru.edit')->with(array('guru'=>$guru));
    }

    public function lihat($id)
    {
        $guru = guru::find($id);
        return view('guru.lihat')->with(array('guru'=>$guru));
    }

    public function update($id, Request $input){
        $guru = guru::find($id);
        $guru -> nama = $input->nama;
        $guru -> nip = $input->nip;
        $guru -> alamat = $input->alamat;
        $guru -> gelar = $input->gelar;
        $guru -> status = $input->status;
        $guru -> jabatan = $input->jabatan;
        $guru->save();

        if (!is_null($input->username)){
            $pengguna=$guru->pengguna->fill($input-
>only('username'));
            if (!empty($input->password))
                $pengguna->password=$input->password;
            if ($pengguna->save()) $this->informasi='Berhasil simpan
data';
        }else {
            $this->informasi='Gagal simpan data';
        }
        return redirect('guru')->with(['informasi'=>$this->informasi]);
    }

    public function hapus($id){

```

```

        $guru = guru::find($id);
        if ($guru->pengguna()->delete()){
            if ($guru->delete())$this->informasi='Berhasil hapus data';
        }
        return redirect('guru')->with(['informasi'=>$this->informasi]);
    }
}

```

3.2.4 Auth Controller

```

<?php

namespace App\Http\Controllers\Auth;

use App\pengguna;
use Validator;
use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ThrottlesLogins;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

class AuthController extends Controller
{
    /**
     |-----
     | Registration & Login Controller
     |-----
     |
     | This controller handles the registration of new users, as well as
the
     | authentication of existing users. By default, this controller uses
     | a simple trait to add these behaviors. Why don't you explore it?

```

```

|
*/

use AuthenticatesAndRegistersUsers, ThrottlesLogins;

/**
 * Where to redirect users after login / registration.
 *
 * @var string
 */
protected $redirectTo = '/';

/**
 * Create a new authentication controller instance.
 *
 * @return void
 */
public function __construct()
{
    $this->middleware($this->guestMiddleware(), ['except' =>
'logout']);
}

/**
 * Get a validator for an incoming registration request.
 *
 * @param array $data
 * @return \Illuminate\Contracts\Validation\Validator
 */
protected function validator(array $data)
{

```

```

return Validator::make($data, [
    'name' => 'required|max:255',
    'email' => 'required|email|max:255|unique:users',
    'password' => 'required|min:6|confirmed',
]);
}

/**
 * Create a new user instance after a valid registration.
 *
 * @param array $data
 * @return User
 */
protected function create(array $data)
{
    return User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => bcrypt($data['password']),
    ]);
}
}

```

3.2.5 Middleware

```
<?php
```

```

namespace App\Http\Middleware;

use Closure;

class AuthentifikasiGuru
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    private $auth;

    public function __construct()
    {
        # code...

        $this->auth = app('auth');
    }

    public function handle($request, Closure $next)
    {
        if ($this->auth->check()) {
            if(Auth::user()->role != 'guru'){
                return abort(403, 'Unauthorized action.');
            }
            else {
                return redirect('/');
            }
            return $next($request);
        }

        return redirect('login')->withErrors('Silahkan Login terlebih

```

```

dahulu');
    }
}

```

3.2.6 View

3.2.6.1 Awal

```

@extends('saya')
@section('container')
<div class="panel panel-default">
<div class="panel-heading">
<strong>Seluruh Data guru</strong>
<a href="{{url('guru/tambah')}}" class="btn btn-xs btn-
primary pull-right">
<i class="fa fa-plus"></i>guru</a>
<div class="clearfix"></div>
</div>
<table class="table">
<thead>
<tr>
<th>No.</th>
<th>Nama</th>
<th>NIP</th>
{ { -- <th>Alamat</th> -- } }
{ { -- <th>Pengguna_id</th> -- } }
<th>Aksi</th>
</tr>
</thead>
<tbody>
<?php $x=1;?>
@foreach ($semuaGuru as $guru)

```



```

<tr>
  <td>{{ $x++ }}</td>
  <td>{{ $guru->nama or 'nama kosong' }}</td>
  <td>{{ $guru->nip or 'NIP kosong' }}</td>
  {{-- <td>{{ $guru->alamat or 'Alamat
kosong' }}</td> --}}
  {{-- <td>{{ $guru->pengguna_id or 'Pengguna_id
kosong' }}</td> --}}
  <td>
    <div class="btn-group" role="group">
      <a href="{{ url('guru/edit/'.$guru->id) }}"
class="btn btn-warning btn-xs" data-toggle="tooltip" data-
placement="top" title="ubah">
        <i class="fa fa-pencil"></i>
      </a>
      <a href="{{ url('guru/lihat/'.$guru->id) }}"
class="btn btn-info btn-xs" data-toggle="tooltip" data-
placement="top" title="lihat">
        <i class="fa fa-eye"></i>
      </a>
      <a href="{{ url('guru/hapus/'.$guru->id) }}"
class="btn btn-danger btn-xs" data-toggle="tooltip" data-
placement="top" title="Hapus">
        <i class="fa fa-remove"></i>
      </a>
    </div>
  </td>

</tr>
@endforeach
</tbody>

```

```
</table>
</div>
@stop
```

3.2.6.2 Edit

```
@extends('saya')
@section('container')
<div class="panel panel-info">
<div class="panel-heading">

<strong><a href="{ { url('dosen') } }">
  <i class="fa text-default fa-chevron-left"></i>
</a>Perbaharui data guru</strong>
</div>
{!!          Form::model($guru,['url'=>'guru/edit/'.$guru-
>id,'class'=>'form-horizontal']) !!}
@include('guru.Form')
<div style="width:100%;text-align:right;">
<button    class="btn    btn-info"><i    class="fa    fa-
save"></i>Perbaharui</button>
<button type="reset" class="btn btn-danger"><i class="fa
fa-undo"></i>Ulangi</button>
</div>
{!! Form::close() !!}

</div>
@stop
```

3.2.6.3 Form

```

<div class="form-group">
  <label class="col-sm-2 control-label">Nama</label>
  <div class="col-sm-10">
    {!!          Form::text('nama',null,['class'=>'form-
control','placeholder'=>"Nama"]) !!}
  </div>
</div>

<div class="form-group">
<label class="col-sm-2 control-label">NIP</label>
  <div class="col-sm-10">
    {!!          Form::text('nip',null,['class'=>'form-
control','placeholder'=>"NIP"]) !!}
  </div>
</div>

<div class="form-group">
<label class="col-sm-2 control-label">Alamat</label>
  <div class="col-sm-10">
    {!!          Form::textarea('alamat',null,['class'=>'form-
control','placeholder'=>"Alamat"]) !!}
  </div>
</div>

<div class="form-group">
<label class="col-sm-2 control-label">Alamat</label>
  <div class="col-sm-10">
    {!!          Form::textarea('gelar',null,['class'=>'form-
control','placeholder'=>"gelar"]) !!}
  </div>
</div>

<div class="form-group">
<label class="col-sm-2 control-label">Alamat</label>
  <div class="col-sm-10">

```

```

        {!!          Form::textarea('status',null,['class'=>'form-
control','placeholder'=>"status"]) !!}

    </div>
</div>

    <div class="form-group">
<label class="col-sm-2 control-label">Alamat</label>
    <div class="col-sm-10">
        {!!          Form::textarea('jabatan',null,['class'=>'form-
control','placeholder'=>" jabatan "]) !!}

    </div>
</div>

    {{ -- .....
<label class="col-sm-2 control-label">Pengguna_Id</label>
    <div class="col-sm-10">
        {!!          Form::text('pengguna_id',null,['class'=>'form-
control','placeholder'=>"Pengguna_Id"]) !!}

    </div>
</div> --}}

    <div class="form-group">
<label class="col-sm-2 control-label">Username</label>
    {!! form::hidden('level','guru')!!}
    <div class="col-sm-10">
        {!!          Form::text('username',null,['class'=>'form-
control','placeholder'=>"username"]) !!}

    </div>
</div>

    <div class="form-group">
<label class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
        {!!          Form::text('password',null,['class'=>'form-
control','placeholder'=>"password"]) !!}

```

```
</div>
</div>
```

3.2.6.4 Lihat

```
@extends('saya')
@section('container')
<div class="panel panel-warning">
    <div class="panel-heading">
        <strong><a href="{{ url('guru') }}"><i
style="color:#8a6d3b" class="fa text-default fa-chevron-
left"></i></a>Detail Data guru
        </strong>
    </div>
    <table class="table">
        <tr>
            <td>nama</td>
            <td>:</td>
            <td>{{ $guru->nama }}</td>
        </tr>
        <tr>
            <td>nip</td>
            <td>:</td>
            <td>{{ $guru->nip }}</td>
        </tr>
        <tr>
            <td>alamat</td>
            <td>:</td>
            <td>{{ $guru->alamat }}</td>
        </tr>
        <tr>
```

```

<td>Username</td>
<td>:</td>
<td>{{ $guru->pengguna->username }}</td>
</tr>
<tr>
<td>Password</td>
<td>:</td>
<td>{{ $guru->pengguna->password }}</td>
</tr>
<tr>
<tr>
<td class="col-xs-4">Dibuat tanggal</td>
<td class="col-xs-1">:</td>
<td>{{ $guru->created_at }}</td>
</tr>
</tr>
<tr>
<td class="col-xs-4">Diperbaharui tanggal</td>
<td class="col-xs-1">:</td>
<td>{{ $guru->updated_at }}</td>
</tr>
</table>
</div>
@stop

```

3.2.6.5 Tambah

```

@extends('saya')
@section('container')
<div class="panel panel-primary">
<div class="panel-heading">

<strong><a href="{{ url('dosen') }}">
    <i style="color:white;" class="fa text-default fa-chevron-
left"></i>
</a>Tambah data guru</strong>
</div>
{!!      Form::open(['url'=>'guru/simpan','class'=>'form-
horizontal']) !!}
@include('guru.Form')
<div style="width:100%;text-align:right;">
<button class="btn btn-primary"><i class="fa fa-
save"></i>Simpan</button>
<button type="reset" class="btn btn-danger"><i class="fa
fa-undo"></i>Ulangi</button>
</div>
{!! Form::close() !!}

</div>
@stop

```

3.2.7 Routes

```

Route::get('guru/lihat/{guru}', 'guruController@lihat');
Route::post('guru/simpan','guruController@simpan');
Route::get('guru/edit/{guru}', 'guruController@edit');
Route::post('guru/edit/{guru}','guruController@update');
Route::get('guru/hapus/{guru}','guruController@hapus');

```

```
Route::get('guru/tambah', 'guruController@tambah');  
Route::get('guru', 'guruController@awal');
```


BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Program ini dirancang dengan tujuan membantu pekerjaan pihak sekolah yaitu admin atau guru untuk mengelola data atau nilai para siswa di sekolah tersebut.

4.2 Saran

Diharapkan dengan adanya program ini, tujuan yang diharapkan dari programmer dapat tercapai yaitu dapat meringankan pekerjaan pihak sekolah dalam mengelola data murid dan mempermudah wali murid dalam mengawasi akademis anak mereka.

DAFTAR PUSTAKA

<https://gilacoding.com/read/belajar-migration-dan-schema-laravel>(21 Mei 2017)

<https://www.babastudio.com/blog/tutorial-laravel-routing-model-view-controller>(22 Mei 2017)

<http://seputarpemrograman.com/memahami-dasar-controller-dan-view-di-framework-laravel.html>(22 Mei 2017)

<http://www.dumetschool.com/blog/Belajar-Laravel-Tentang-Eloquent-dan-Tinker>(20 Mei 2017)

<https://idcloudhost.com/pengertian-dan-keunggulan-framework-laravel/>(21 Mei 2017)