# LAPORAN AKHIR

**MATAKULIAH PRAKTIKUM PEMROGRAMAN FRAMEWORK SISTEM INFORMASI BIMBEL PRIMA ALPHA**



**Disusun Oleh:**

**KELOMPOK 8**

**IQBAL REFLIANSYAH / 1315015002 ROZY RIVALDI / 1515015007**

**PROGRAM STUDI TEKNIK INFORMATIKA JURUSAN TEKNOLOGI INFORMASI DAN KOMUNIKASI**

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS MULAWARMAN**

**2017**

# KATA PENGANTAR

Puji Syukur kehadirat ALLAH SWT, karena atas perkenanNYA laporan akhir praktikum Pemrograman Web dapat diselesaikan.

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

1. Bapak Hario Jati Setyadi selaku dosen Framework yang telah meluangkan waktu, tenaga dan pikiran dalam pelaksanaan bimbingan, pengarahan, dorongan dalam rangka penyelesaian penyusunan laporan ini.
2. Teman-teman mahasiswa Fakultas Teknologi Informasi dan Komunikasi.

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 menbangun sangat kami harapkan.

Samarinda, 19 Mei 2016

Penyusun

# TAKARIR

*database* basis data

*input* masukan

*output* keluaran

*programming project* proyek pemrograman

*screenshot* hasil tampilan

*source code* kode program

*user* pengguna

DAFTAR ISI

[KATA PENGANTAR ii](#_TOC_250016)

[TAKARIR iii](#_TOC_250015)

[DAFTAR ISI iv](#_TOC_250014)

[DAFTAR GAMBAR v](#_TOC_250013)

[BAB I PENDAHULUAN 1](#_TOC_250012)

* 1. [Deskripsi Masalah 1](#_TOC_250011)
  2. [Rumusan Masalah 2](#_TOC_250010)
  3. [Batasan Masalah 2](#_TOC_250009)
  4. [Tujuan 2](#_TOC_250008)

[BAB II PERANCANGAN 3](#_TOC_250007)

* 1. [Analisis Program 3](#_TOC_250006)
  2. Entity Relationship Diagram 3
  3. [Konsep/Materi Praktikum yang dipakai 4](#_TOC_250005)
     1. Blade 4
     2. Controller 4
     3. [Eloquent Relationship 5](#_TOC_250004)

BAB III HASIL DAN PEMBAHASAN 7

* 1. [Implementasi Program 7](#_TOC_250003)
  2. [Source Code 13](#_TOC_250002)

[BAB IV KESIMPULAN DAN SARAN ............................................................................](#_TOC_250001)

* 1. Kesimpulan 47
  2. Saran 47

[DAFTAR PUSTAKA 48](#_TOC_250000)

# DAFTAR GAMBAR

GAMBAR 2.1 *Entity Relationship Diagram* 3

GAMABR 3.1 *Implementasi gambar* 6

GAMBAR 3.1.1 *Home Login* 6

GAMBAR 3.1.9 *Login* 11

# BAB I PENDAHULUAN

## Deskripsi Masalah

Program dibuat agar memudahkan info bimbel kepada siswa dan tutor.

Agar siswa dan tutor dapat melihat jadwal menambah jadwal bimbel masing2

## Rumusan Masalah

Program kami memiliki perumusan masalah, yaitu bagaimana suatu program mampu membantu memudahkan masalah dalam sistem pendaftran bimbel

## Batasan Masalah

Batasan masalah kami disini adalah :

* + 1. Hanya cuma pihak admin yang bisa mengakses program

## Tujuan

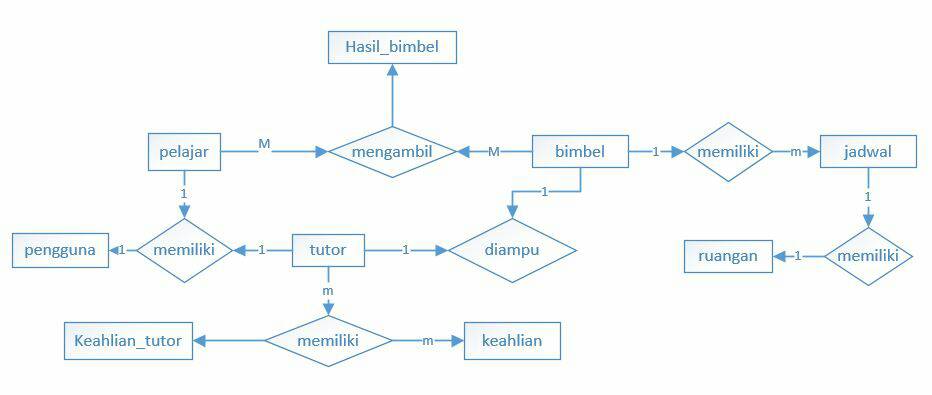
Tujuan dan manfaat dari program ini adalah memudahkan perusahaan pororoland untuk melakukan pendataan mengenai informasi status dari tipe rumah, mengenai status transaksi pembayaran , data para pelanggan, dan data pengguna. Selain itu untuk memudahkan pengguna dalam mengetahui informasi dari setiap perumahan yang sudah di beli.

# BAB II PERANCANGAN

## Analisis Program

Program ini di buat untuk memudahkan Agar siswa dan tutor dapat melihat jadwal menambah jadwal bimbel masing2

* 1. ***Entity Relationship Diagram***



**Gambar 2.1** *Entity RelationshipDiagram*

## Konsep/Materi Praktikum yang dipakai

* + 1. **Controller**

Controller adalah file yang nantinya akan menghandle 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. Umumnya pada suatu program yang modular (terpisah-pisah) setiap modul akan memiliki 1paket MVC (Model View Controller) di mana di laravel juga menerapkan arsitektur MCV namum pada laravel standarnya tidak di buat dalam 1 modul, melainkan terpisah, Model di simpan di direktori app/, Controller di simpan pada folder app/Http/Controllers/ dan view di simpan pada resources/views.

* + 1. **Blade**

### Blade adalah templating engine yang merupakan bawaan dari laravel, tidak seperti templating engine dari framework lain yang juga popular, Blade tidak memaksakan penggunanya menggunakan kode *plain* PHP pada view. Semua view Blade akan di *compile* menjadi *code plain* PHP dan di *chace*-kan sampai terjadi modifikasi pada file blade, dan artinya Blade tidak menambah *overhead* pada applikasi yang di buat. File Blade view harus di menggunakan ekstensi .blade.php dan umumnya di simpan pada resources/views directory.

Setelah tadi membuat controller dan model, maka selanjutnya yang di buat adalah view dari pengguna yang berfungsi sebagai perantara antara *system* dan pengguna. Langkah paling awal adalah membuat sebuah master template yang mana akan berfungsi untuk menyimpan semua header dan footer dari halaman yang selalu sama pada setiap halaman yang di tampilkan pada user, buka kembali text editor dan buat satu file baru pada folder resources/views/ dan berikan berikan nama master.blade.php

## Eloquent Relationship

Eloquent: Relationship adalah tabel database sering berhubungan satu sama lain. Sebagai contoh, sebuah posting blog mungkin memiliki banyak komentar, atau orderan dapat berhubungan dengan pengguna yang memesan orderan itu. Eloquent membuat pengelolaan dan bekerja dengan relationship menjadi mudah, dan mendukung beberapa jenis relationship:

* + - * *One to One*

Relasi one to one adalah relasi yang sangat dasar. Misalnya, model Pengguna mungkin berrelasi dengan satu Model Mahasiswa. Untuk menentukan hubungan ini, kita membuat satu fungsi mahasiswa() pada model User. Fugnsi mahasiswa() harus mempunyai nilai return dari fungis hasOne().

* + - * *One to Many*

Relasi One-to-Many digunakan untuk mendefinisikan hubungan di mana satu model memiliki banyak jumlah model lain. Sebagai contoh, Seorang Dosen mungkin akan memiliki banyak jadwal matakuliah yang di ajar. Seperti semua hubungan eloquent lainnya, hubungan one-to-many didefinisikan dengan membuat fungsi hasMany().

* + - * *Many to Many*

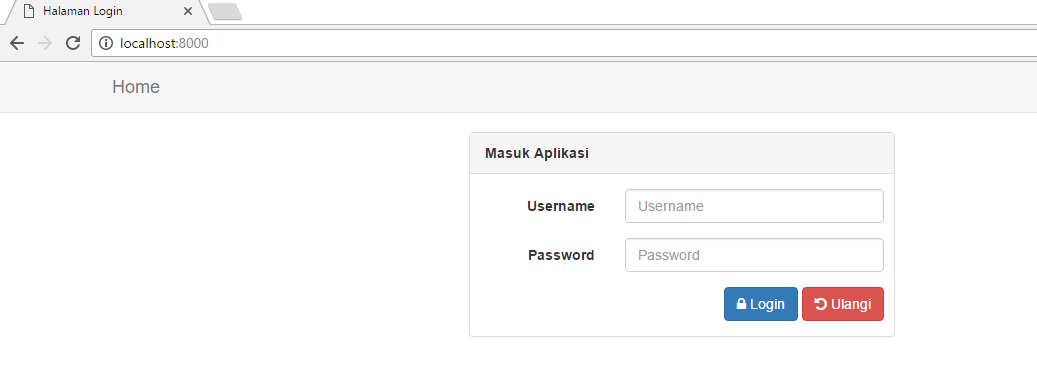
Relasi many-to-many yang sedikit lebih rumit daripada hubungan hasOne dan hasMany. Contoh hubungan tersebut adalah pengguna dengan banyak peran, di mana peran juga dimiliki oleh pengguna lain. Misalnya, banyak pengguna dapat memiliki peran "Admin". Untuk menentukan hubungan ini, tiga tabel database yang diperlukan: pengguna, peran, dan pengguna\_peran. Tabel pengguna\_peran berasal dari urutan abjad dari nama-nama model yang terkait, dan berisi pengguna\_id dan kolom peran\_id. Relasi many-to-many didefinisikan dengan menulis sebuah metode yang memanggil fungis belongsToMany dari kelas eloquent.

**BAB III**

**HASIL DAN PEMBAHASAN**

## 3.1 Implementasi Program

* + 1. **Home Login**



Pada halaman ini pengguna akan melakukan login sebagai admin dengan memasukkan username dan password.

**Gambar 3.1.1 Tampilan Login**

## 3.2 Source Code

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class bimbel extends Model

{

protected $table = 'bimbel';

protected $fillable = ['mata\_pelajaran','keterangan'];

public function tutor\_bimbel()

{

return $this->hasOne(tutor\_bimbel::class);

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class jadwal extends Model

{

protected $table = 'jadwal';

protected $fillable = ['id\_pelajar','id\_ruangan','id\_tutor\_bimbel'];

protected $guarded = ['id'];

public function pelajar(){

return $this->belongsTo(pelajar::class,'id\_pelajar');

}

public function tutor\_bimbel(){

return $this->belongsTo(tutor\_bimbel::class,'id\_tutor\_bimbel');

}

public function ruangan(){

return $this->belongsTo(ruangan::class,'id\_ruangan');

}

public function getNamaTutorAttribute(){

return $this->tutor\_bimbel->tutor->nama;

}

public function getNipTutorAttribute(){

return $this->tutor\_bimbel->tutor->nip;

}

public function getttrbmblAttribute(){

return $this->tutor\_bimbel->matakuliah->title;

}

public function getNamapljrAttribute(){

return $this->pelajar->nama;

}

public function getNimAttribute(){

return $this->pelajar->nim;

}

public function getKoderuanganAttribute(){

return $this->ruangan->title;

}

public function listTutorBimbelDanPelajarDanRuangan()

{

$out = [];

foreach ($this->all() as $jdwl) {

$out[$jdwl->id] = "{$jdwl->tutor\_bimbel->tutor->nip} {$jdwl->tutor\_bimbel->tutor->nama} {$jdwl->pelajar->nama} (ruangan {$jdwl->ruangan->kode\_ruangan})";

}

return $out;

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class keahlian extends Model

{

protected $table = 'keahlian';

protected $fillable = ['keahlian'];

public function keahlian\_tutor()

{

return $this->belongsToMany(keahlian\_tutor::class);

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class pelajar extends Model

{

protected $table = 'pelajar';

protected $fillable = ['nim','nama','alamat','pengguna\_id'];

public function pengguna()

{

return $this->belongsTo(pengguna::class);

}

public function jadwal()

{

return $this->hasMany(jadwal::class,'id\_pelajar');

}

public function getNama\_PenggunaAttribute()

{

return $this->pengguna->nama\_pengguna;

}

public function listpelajarDanNim()

{

$out = [];

foreach ($this->all() as $pljr)

{

$out[$pljr->id] = "{$pljr->nim} ({$pljr->nama})";

}

return $out;

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

// use Illuminate\Conracts\Auth\Authenticatable;

class Pengguna extends Model //implements Authenticatable

{

//use \Illuminate\Auth\Authenticatable;

protected $table = 'pengguna';

protected $fillable = ['nama\_pengguna','password'];

public function pelajar()

{

return $this->hasOne(pelajar::class);

}

public function tutor()

{

return $this->hasOne(tutor::class);

}

public function peran()

{

return $this->belongToMany(peran::class);

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class peran extends Model

{

protected $table = 'peran';

public function Pengguna()

{

return $this->belonsToMany(Pengguna::class);

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class ruangan extends Model

{

protected $table = 'ruangan';

protected $fillable = ['kode\_ruangan'];

public function jadwal()

{

return $this->hasMany(jadwal::class);

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class tutor extends Model

{

protected $table = 'tutor';

protected $fillable = ['nip','nama','email','alamat','pengguna\_id'];

public function pengguna(){

return $this->belongsTo(pengguna::class);

}

public function tutor\_bimbel()

{

return $this->hasMany(tutor\_bimbel::class,'id\_tutor','id');

}

public function getNama\_PenggunaAttribute()

{

return $this->pengguna->nama\_pengguna;

}

public function listTutorDanNip()

{

$out = [];

foreach ($this->all() as $ttr)

{

$out[$ttr->id] = "{$ttr->nip} ({$ttr->nama})";

}

return $out;

}

public function listTutorDanBimbel()

{

$out = [];

foreach ($this->all() as $ttrbmbl) {

$out[$ttrbmbl->id] = "{$ttrbmbl->tutor->nip} {$ttrbmbl->tutor->nama} (bimbel {$ttrbmbl->bimbel->mata\_pelajaran})";

}

return $out;

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class tutor\_bimbel extends Model

{

protected $table = 'tutor\_bimbel';

protected $fillable = ['id\_tutor','id\_bimbel'];

// protected $guarded = ['id'];

public function tutor(){

return $this->belongsTo(tutor::class,'id\_tutor');

}

public function jadwal(){

return $this->hasMany(jadwal::class,'id\_tutor\_bimbel');

}

public function bimbel(){

return $this->belongsTo(bimbel::class,'id\_bimbel');

}

public function getNamatutorAttribute(){

return $this->tutor->nama;

}

public function getNiptutorAttribute(){

return $this->tutor->nip;

}

public function getmata\_pelajaranAttribute(){

return $this->bimbel->mata\_pelajaran;

}

public function listTutorDanBimbel()

{

$out = [];

foreach ($this->all() as $ttrbmbl)

{

$out[$ttrbmbl->id] = "{$ttrbmbl->tutor->nama} (bimbel{$ttrbmbl->bimbel->mata\_pelajaran})";

}

return $out;

}

//

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class tutor\_bimbel extends Model

{

protected $table = 'tutor\_bimbel';

protected $fillable = ['id\_tutor','id\_bimbel'];

// protected $guarded = ['id'];

public function tutor(){

return $this->belongsTo(tutor::class,'id\_tutor');

}

public function jadwal(){

return $this->hasMany(jadwal::class,'id\_tutor\_bimbel');

}

public function bimbel(){

return $this->belongsTo(bimbel::class,'id\_bimbel');

}

public function getNamatutorAttribute(){

return $this->tutor->nama;

}

public function getNiptutorAttribute(){

return $this->tutor->nip;

}

public function getmata\_pelajaranAttribute(){

return $this->bimbel->mata\_pelajaran;

}

public function listTutorDanBimbel()

{

$out = [];

foreach ($this->all() as $ttrbmbl)

{

$out[$ttrbmbl->id] = "{$ttrbmbl->tutor->nama} (bimbel{$ttrbmbl->bimbel->mata\_pelajaran})";

}

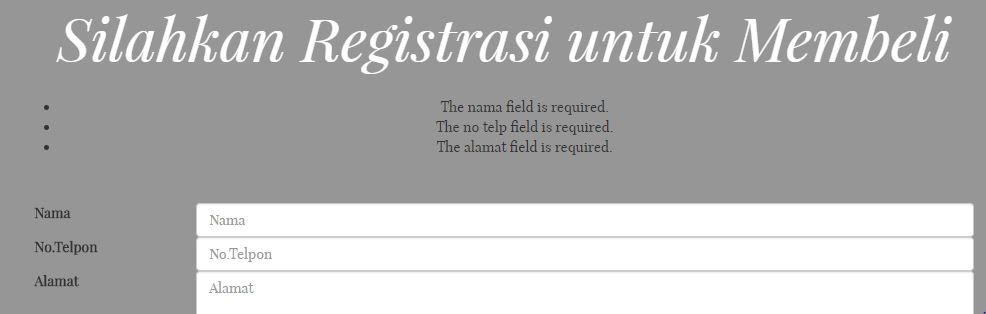
return $out;

}

//

}

**Pertanyaan Dan Jawaban**

1. Error di pendaftaran pembeli di keluarkan !
2. Ditombol aksi tambah data angsuran !
3. Nomor telpon di kasih batasan !





# BAB IV KESIMPULAN DAN SARAN

## Kesimpulan

Dari penjelasan program yang ada maka kesimpulan dari adanya program ini adalah agar dapat mempermudah seseorang untuk melakukan pendataan sistem informasi terhadap para customer yang ingin membeli rumah .

## Saran

Diharapkan dengan adanya framework sistem informasi perumahan pororoland berbasis Web dapat mempersingkat waktu dan mempermudah admin-admin dalam melakukannya .

# DAFTAR PUSTAKA

**htt**[**ps://www**](http://www.scribd.com/doc/100184351/makalah-perumahan-dan-)**.scr**[**ibd.com/doc/100184351/makalah-perumahan-dan-**](http://www.scribd.com/doc/100184351/makalah-perumahan-dan-) **permukiman**