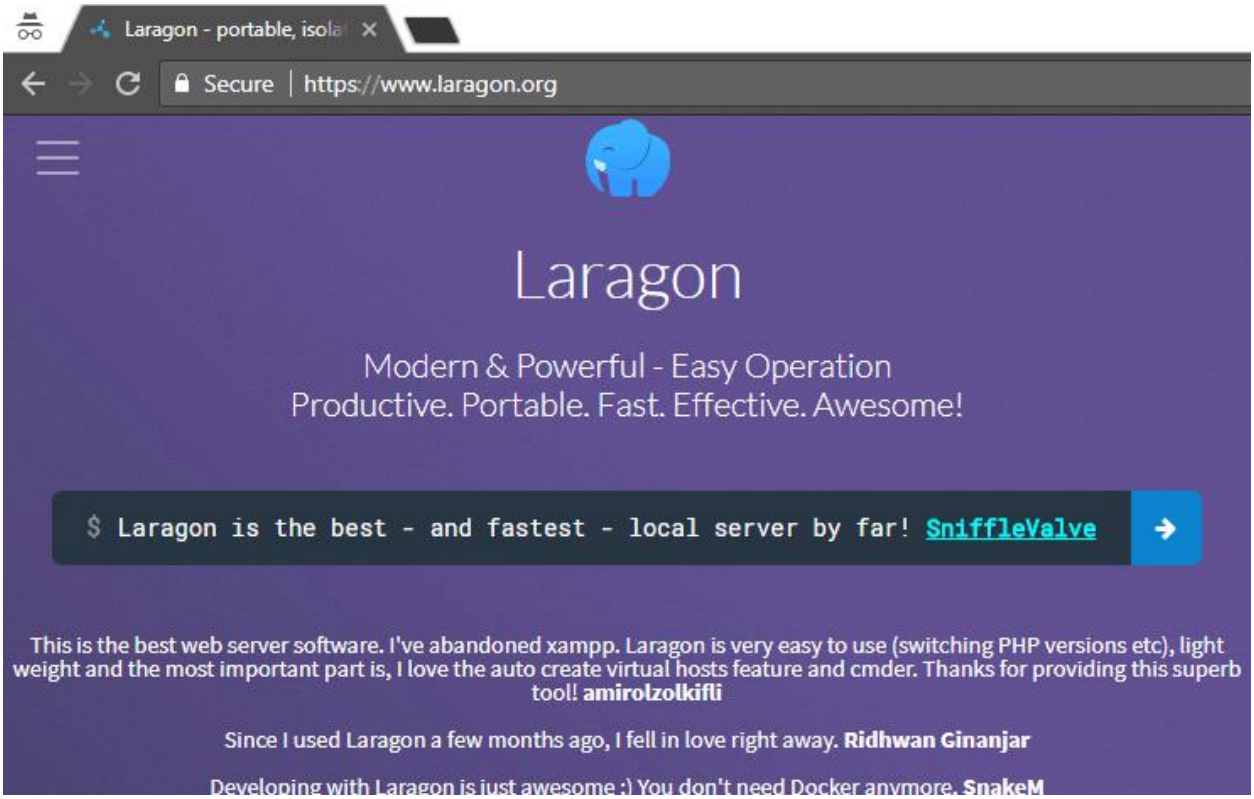
LARAVEL EBOOK BAHASA MELAYU
KHIRULNIZAM ABD RAHMAN
khirulnizam.com

Isi kandungan tutorial LARAVEL

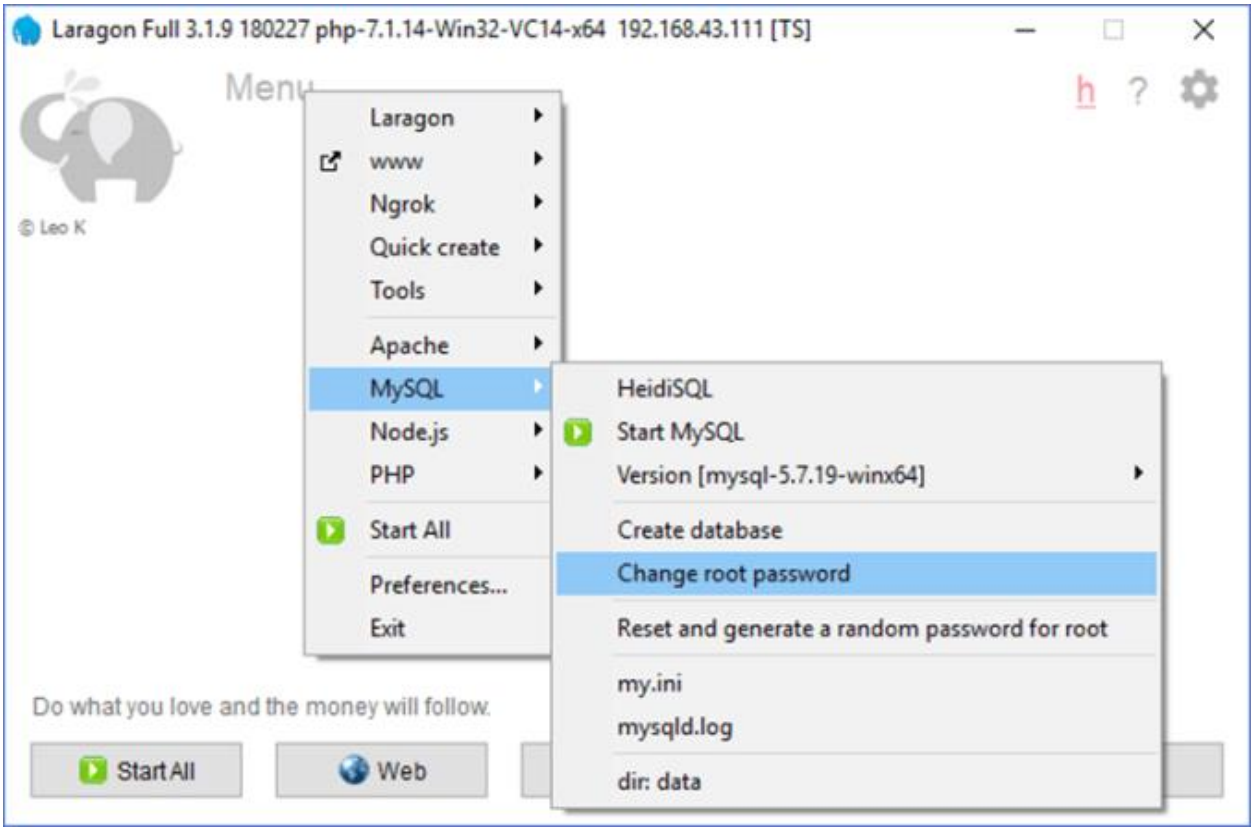
Tutorial 01 Pengenalan Laravel



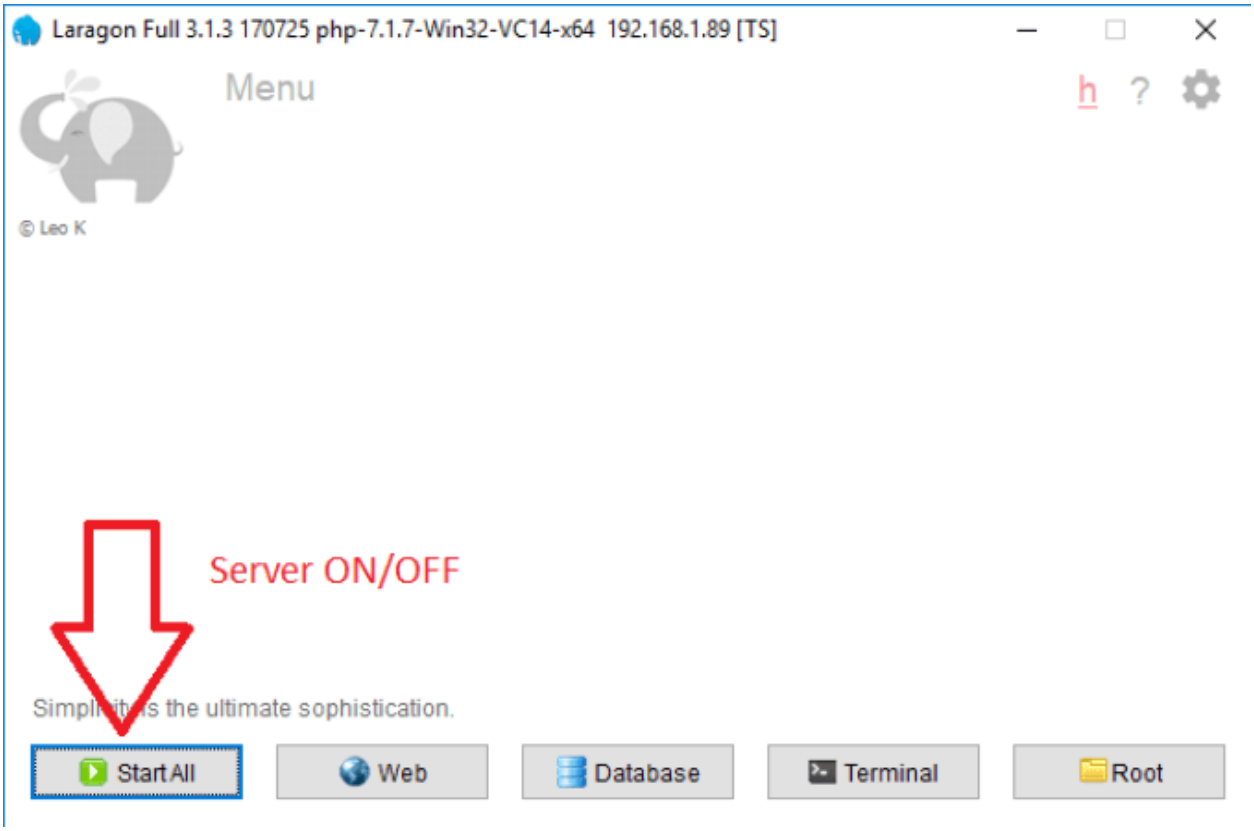
Tutorial Laravel bahasa Melayu ini dimulakan dengan pemasangan LARAGON. Laragon adalah satu perisian kompilasi yang telah disediakan semua keperluan perisian pembangunan projek Laravel.

Laragon (WAMP-Windows Apache, MySQL,PHP) terdiri daripada; Apache 2.4, Nginx 1.12, MySQL 5.7, PHP 7.1.7, Node.js 6.11, git, … **muatturun di sini Laragon – WAMP (112 MB)**

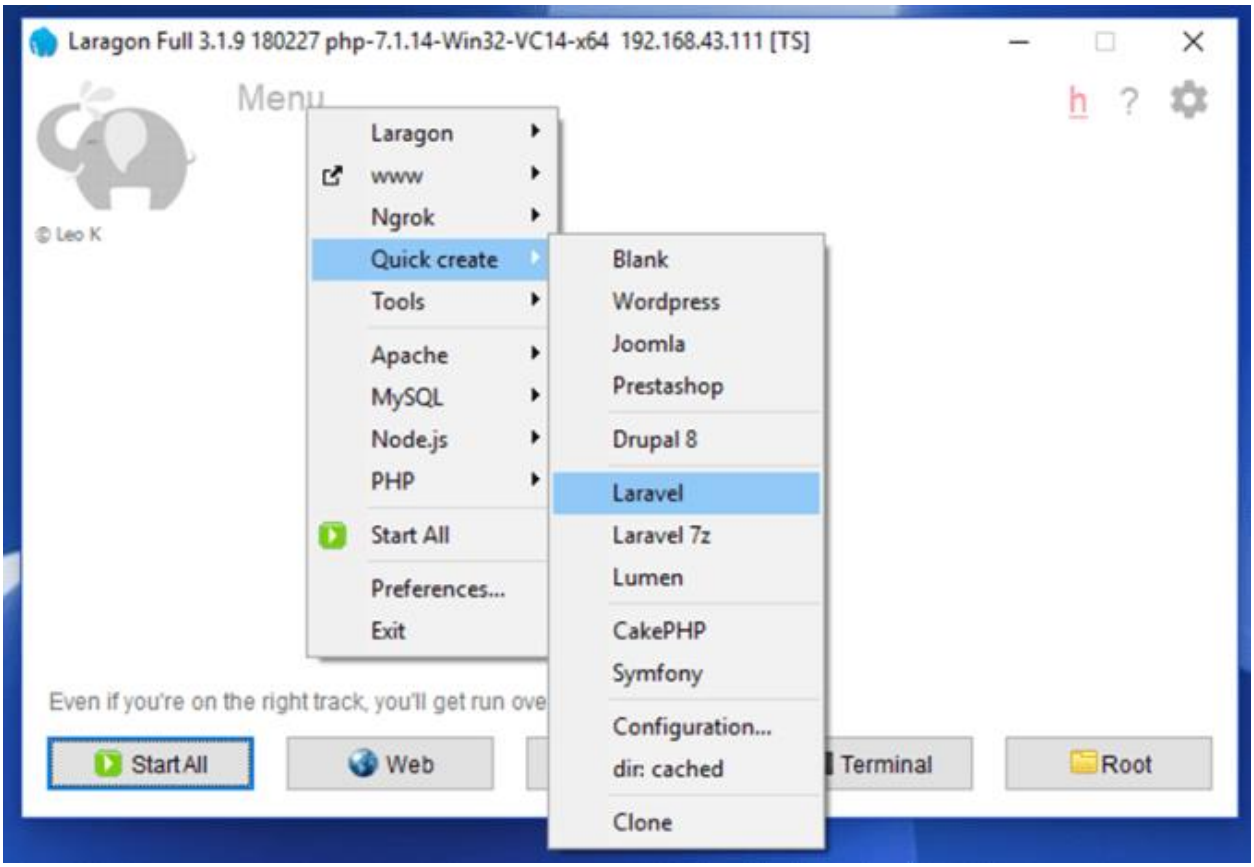
Selepas berjaya install Laragon, anda akan melihat paparan berikut. Pertamanya sila tukar kata laluan pengguna pangakalan data *root (root database username)* .



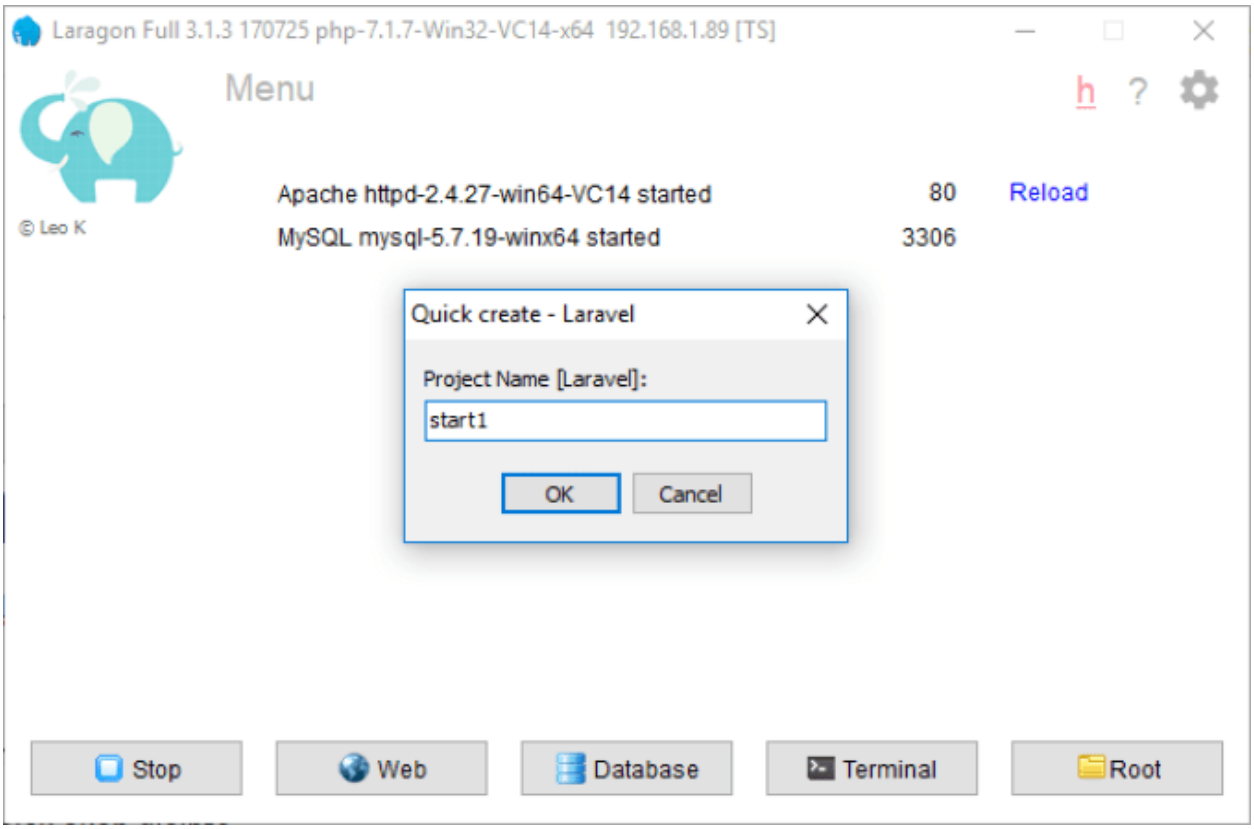
Kemudian ON web (*Apache*) dan server pangkalan data (*mysql*) dengan tekan pada butang “Start All”.



Untuk memulakan projek Laravel yang baharu, klik-kanan pada paparan Quick Create -> Laravel. Tindakan ini akan memuat-turun fail-fail template kod projek Laravel dari server repository Laravel. Sila pastikan Internet berfungsi dengan baik.



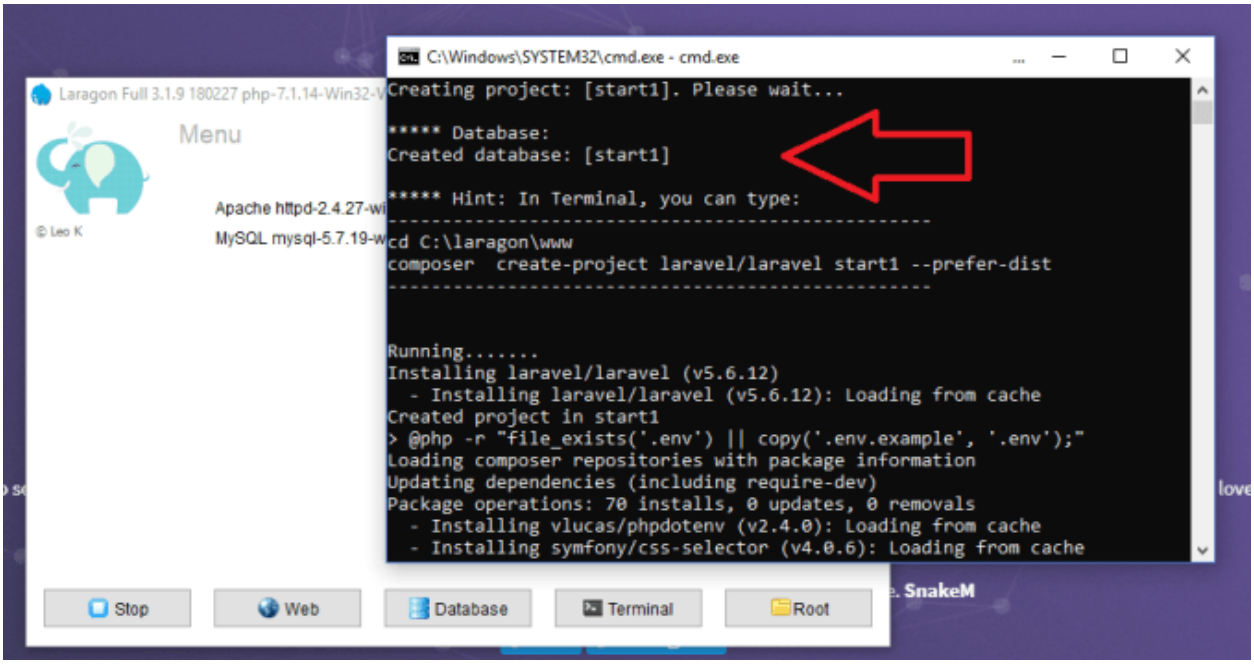
Masukkan nama projek Laravel; contoh ***start1***



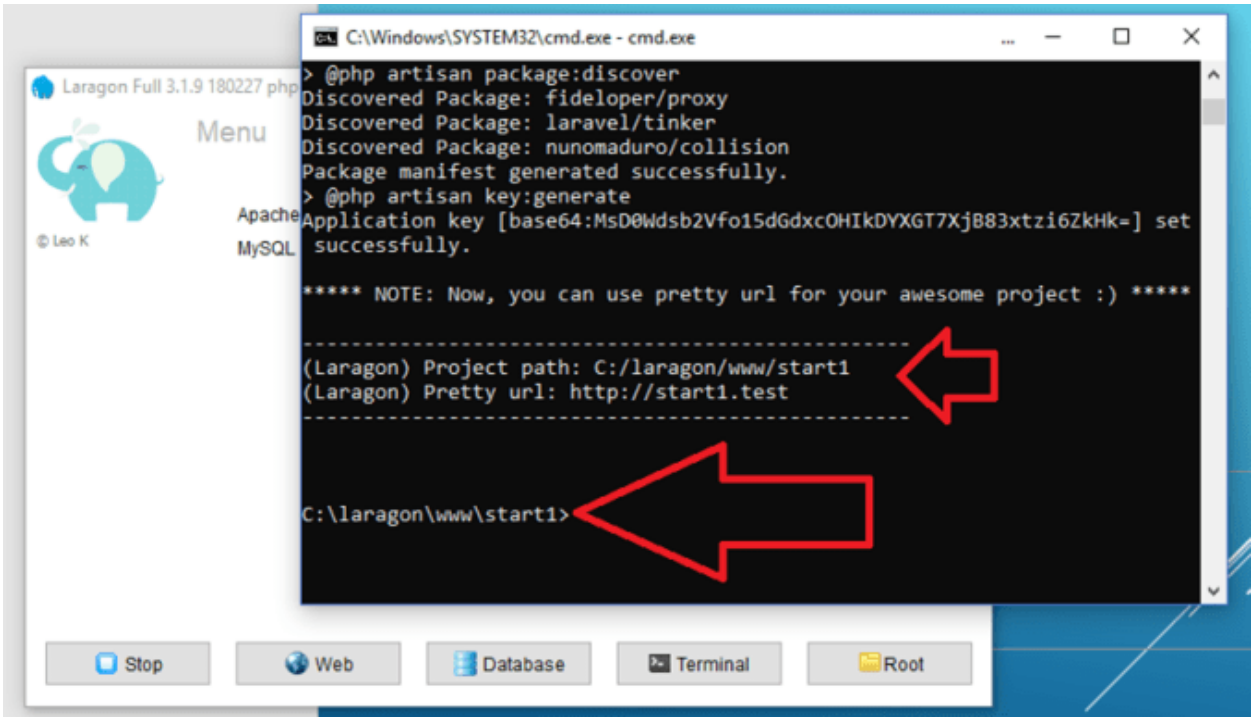
Paparan pada cmd.exe menunjukkan beberapa langkah untuk menjana projek baharu Laravel.

Pangkalan data projek akan dicipta

Fail-fail template projek Laravel akan dimuatturun dari server repository (sila bersabar mungkin akan mengambil masa tertakluk kepada kelajuan Internet atau rangkaian anda)

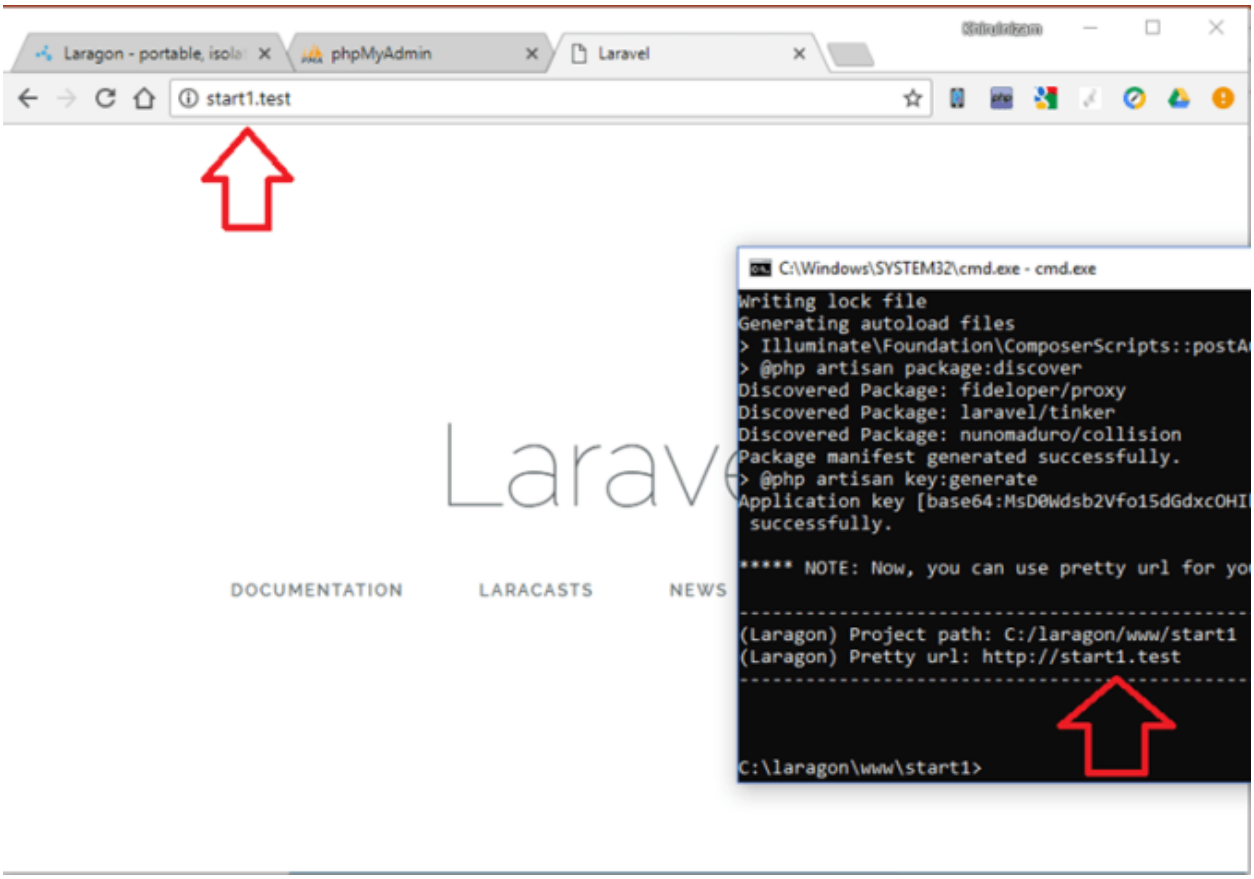


...rajah bersambung...

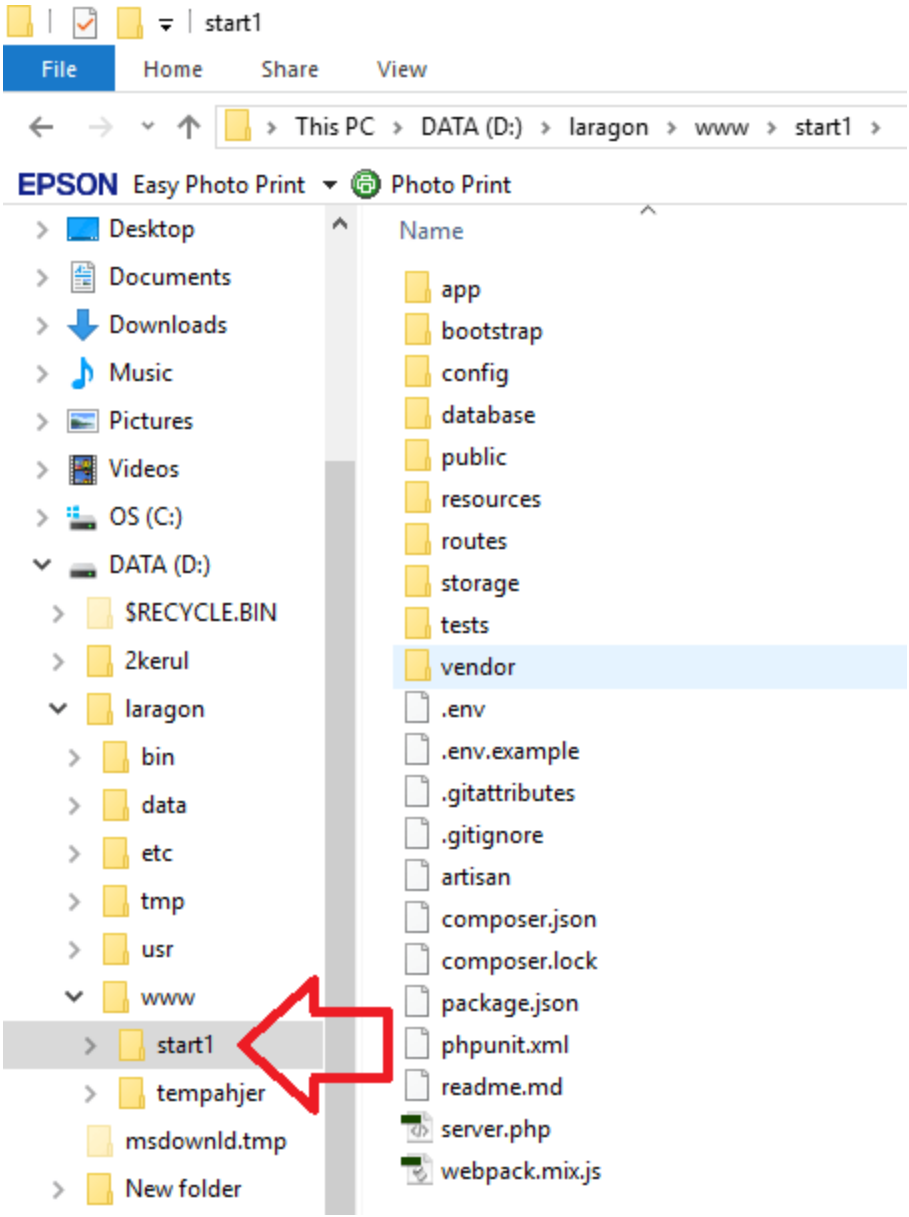


Uji server anda dengan menaip start1.test
Sepatutnya paparan teks Laravel seperti dibawah akan kelihatan. Pastikan paparan seperti berikut untuk meneruskan langkah seterusnya. Sekiranya

server tidak dalam keadaan baik, atau keluar paparan laman tidak dijumpai boleh gunakan arahan php artisan serve.



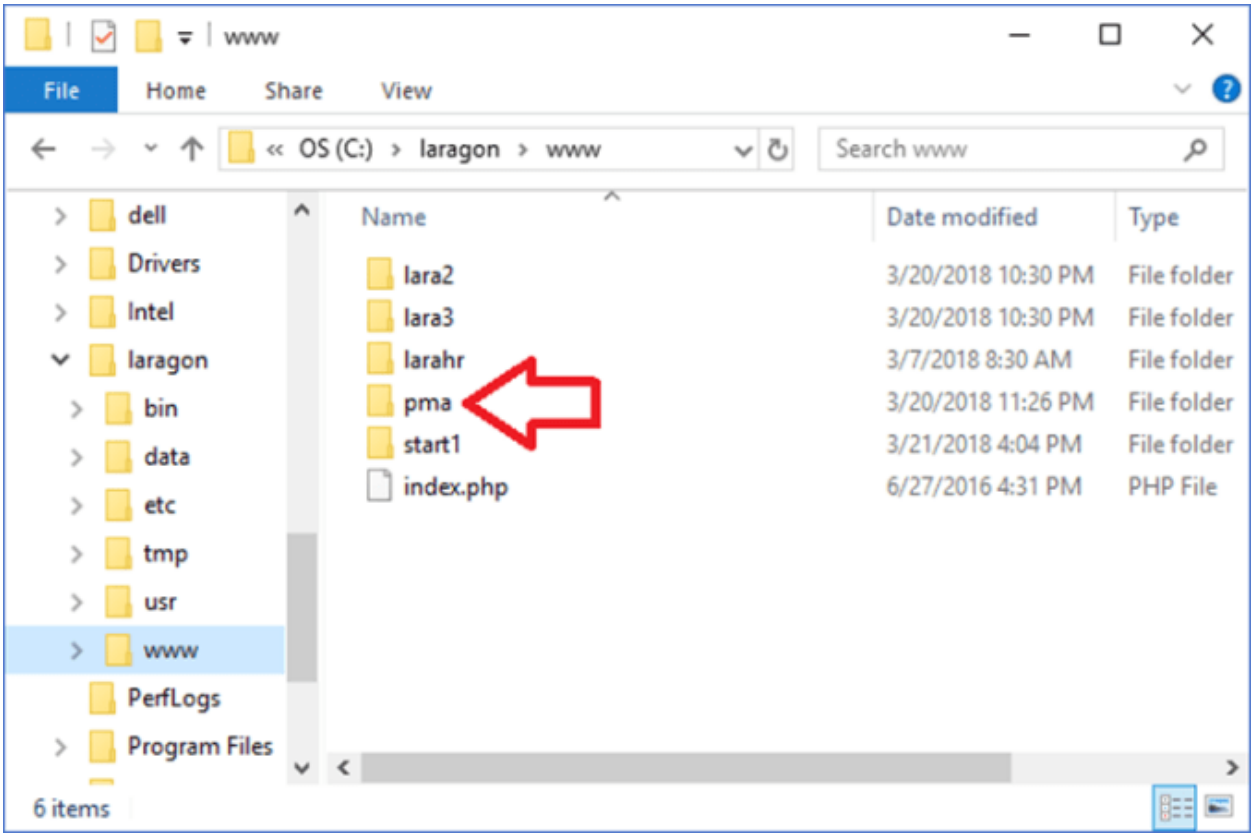
Berikut struktur fail projek Laravel yang akan diperolehi selepas selesai muatturun.



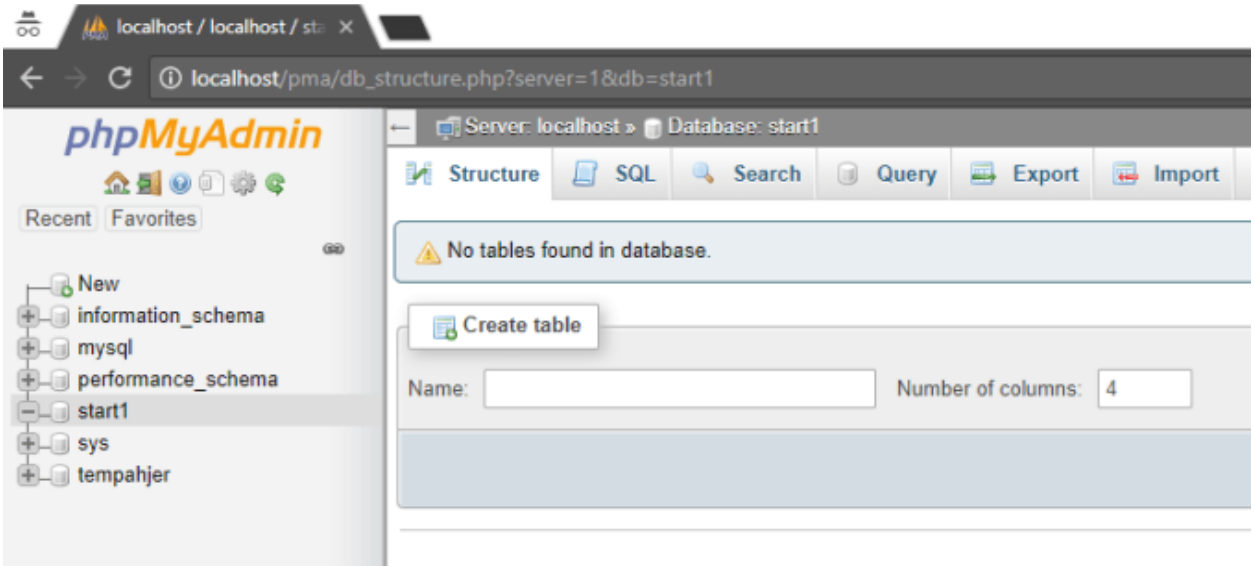
Penyelenggara Pangkalan data (*database admin*)

Proses seterusnya adalah untuk memasang database admin. Boleh pilih web phpmyadmin, atau GUI lain seperti SQLYOG atau NAVICAT dll. Di sini kami tunjukkan phpmyadmin (PMA).

Download PMA dari phpmyadmin.net dan extract file zip ke dalam folder laragon/www



PMA cuma diperlukan untuk semak pangkalan data / jadual (tables) berfungsi dengan baik. Setakat cipta projek baharu tadi, hanya pangkalan data yang disediakan. Jadual belum ada lagi. Bina jadual kita kena laksanakan proses *migrate* .

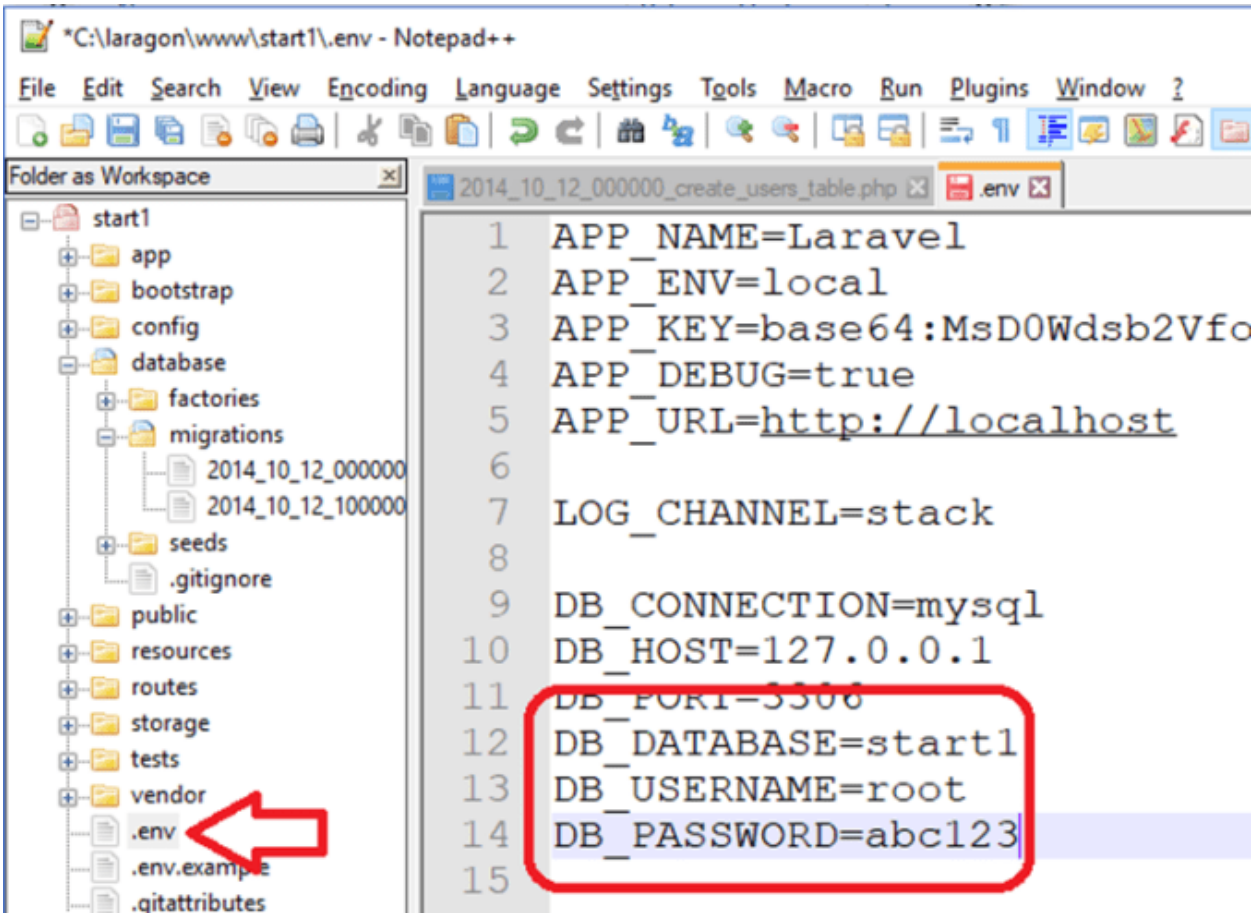


Latihan *User authentication*

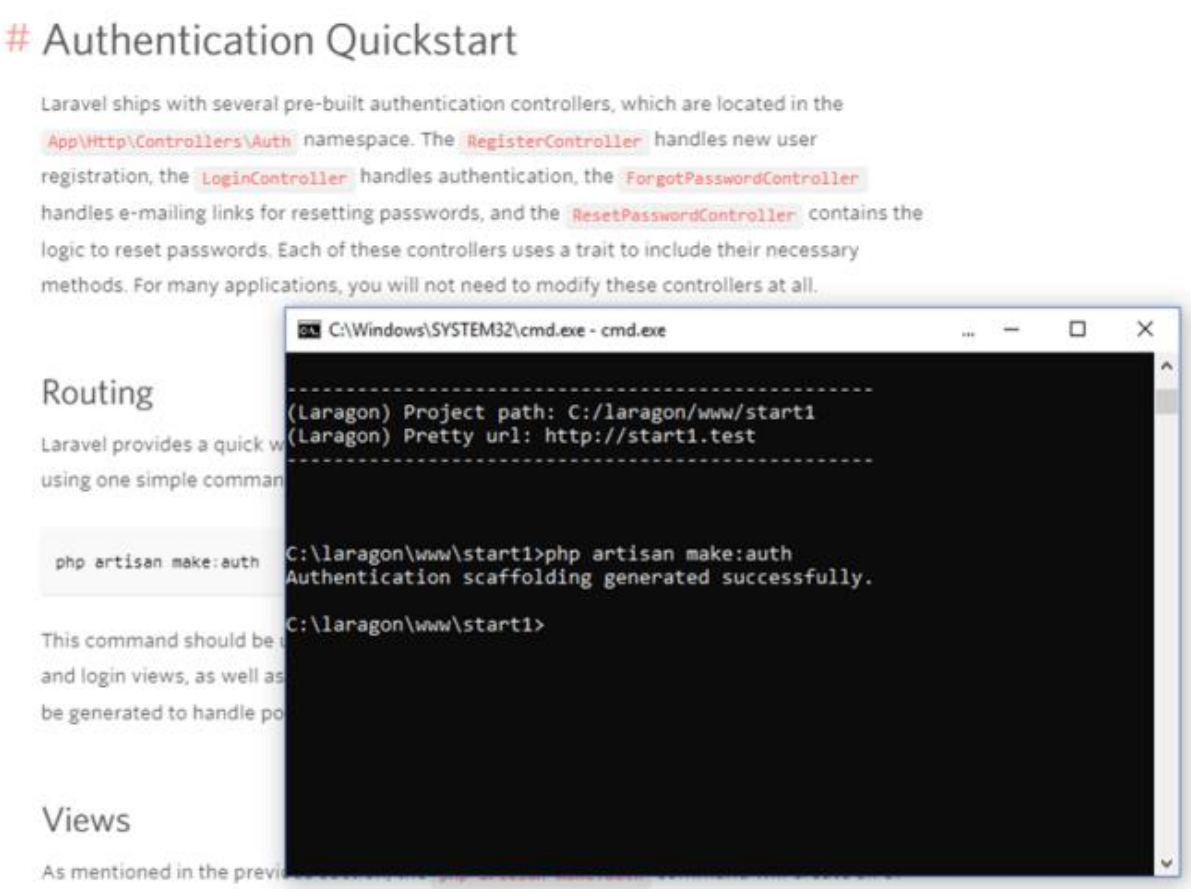
Pangkalan data telah tesedia. Sekarang kita akan menambah modul pengurusan pengguna (*user authentication*).

Untuk menyambungkan projek Laravel kepada pangkalan data yang tersedia perlu diubah pada tetapan pangkalan data dalam fail **env** .

Gunakan pengedit kod kegemaran anda; Notepad++, Atom, Sublime atau vscode, dll. Ubah tetapan seperti dalam server database anda. Tutorial ini menggunakan Notepad++ untuk menulis kod Laravel, PHP atau blade. Editor (IDE) terbaik untuk mengedit kod Laravel yang terbaik setakat ini adalah PHPStorm (berbayar), boleh juga cuba Eclipse for PHP/web.

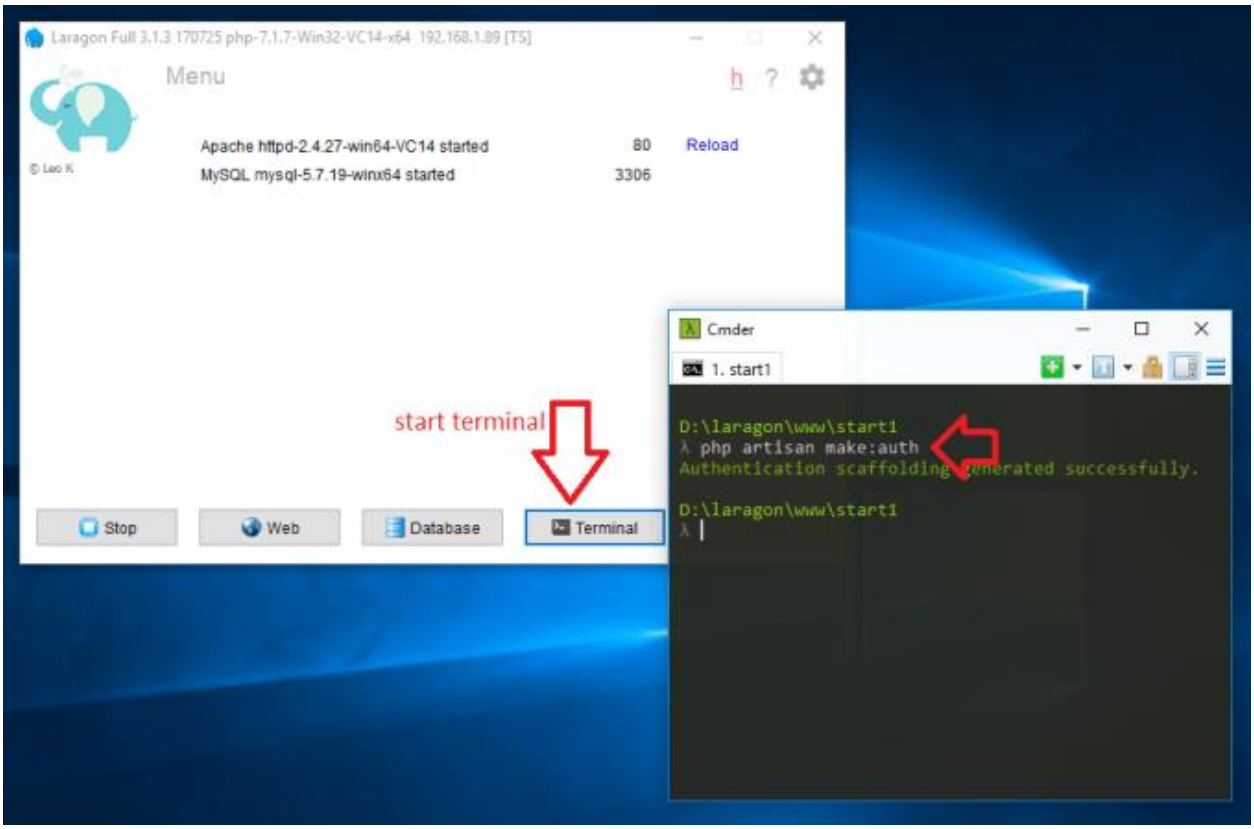


Untuk menambahkan modul pengguna (***auth***) dalam projek web Laravel.

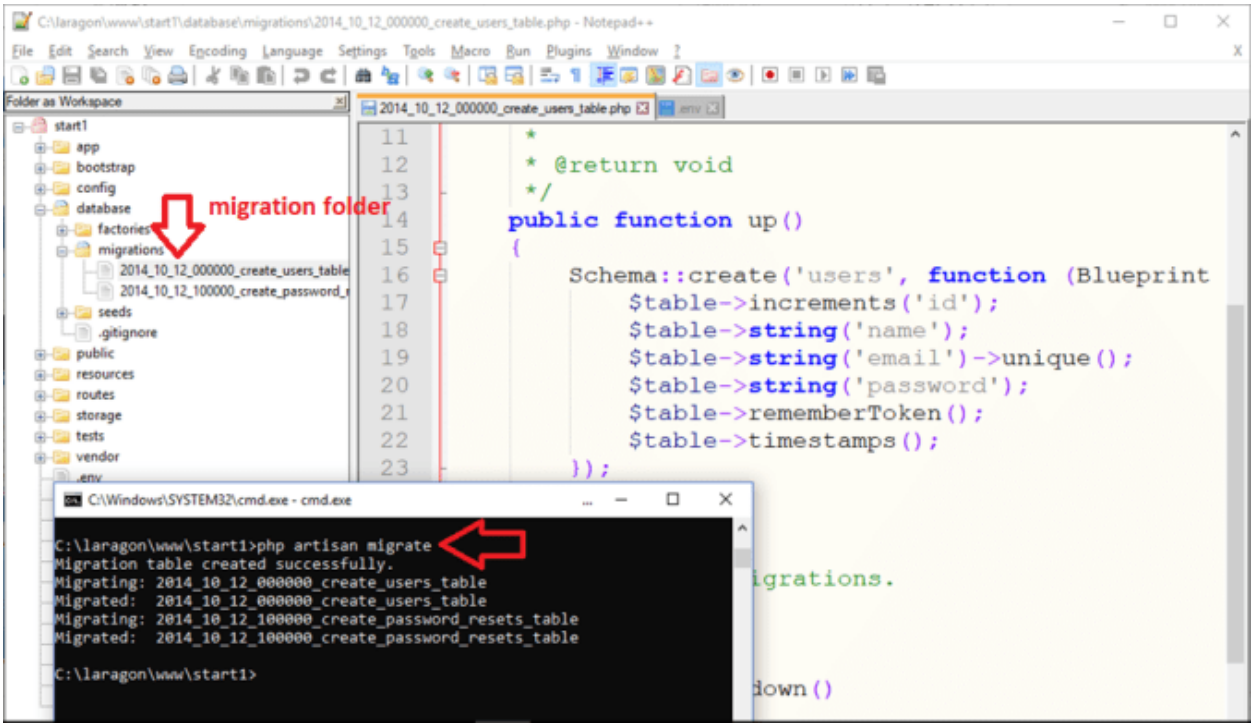


Laksanakan arahan berikut dalam terminal untuk menambah modul auth,

php artisan make:auth



Sekiranya kita nak membuat perbandingan nama dan katalaluan pengguna () dengan maklumat disimpan dalam pangkalan data, semestinya kita ada jadual untuk menyimpan maklumat tersebut. Kaedah Laravel menggalakkan kita isytihar struktur jadual di dalam folder ***migration*** . Edit fail create_users_table.php dalam folder migration. Tambah kod dalam fungsi ***up()*** seperti berikut.

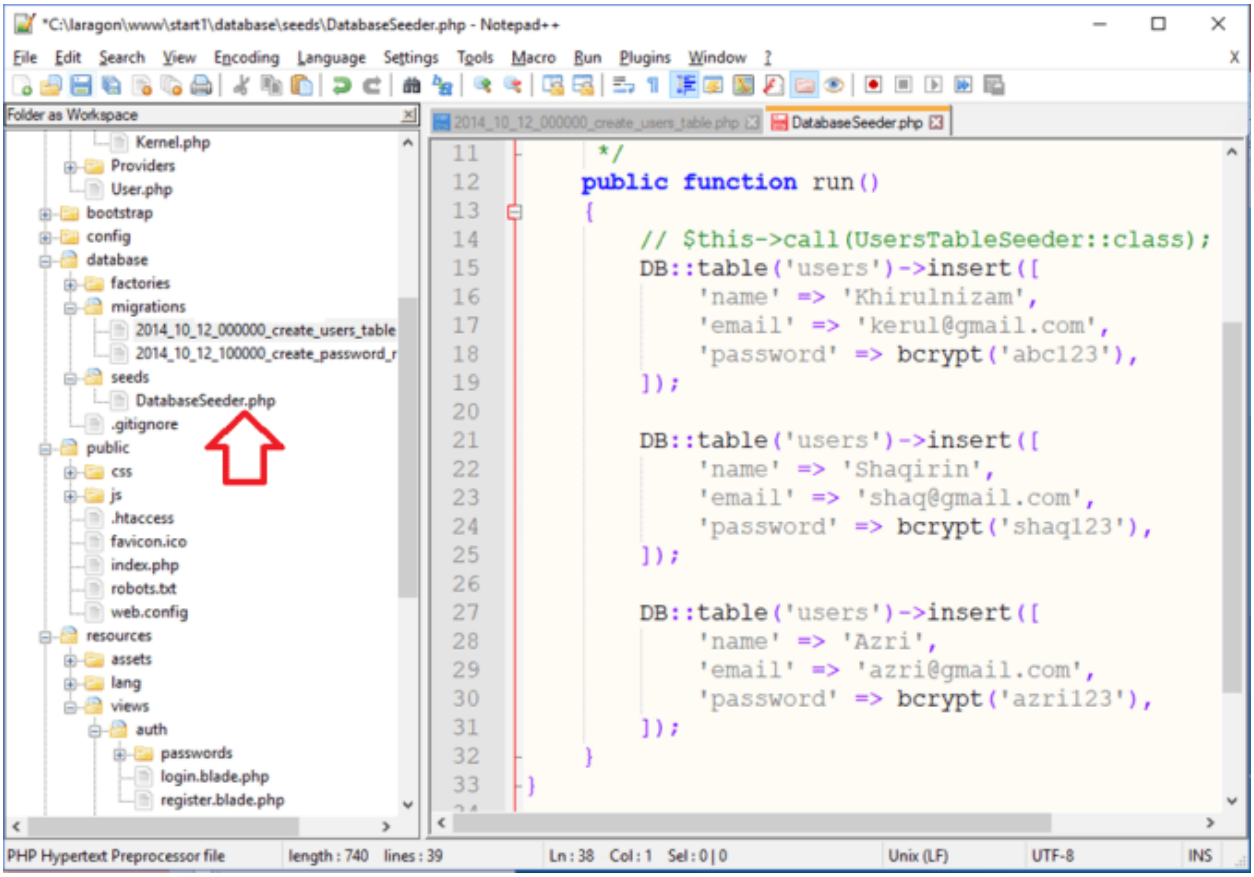


Selepas selesai kod pengisytiharan jadual dimasukkan, laksanakan arahan berikut untuk mencipta jadual ke dalam pangkalan data;

php artisan migrate

Rujuk dalam localhost/pma untuk melihat struktur jadual telah dicipta dalam pangkalan data.

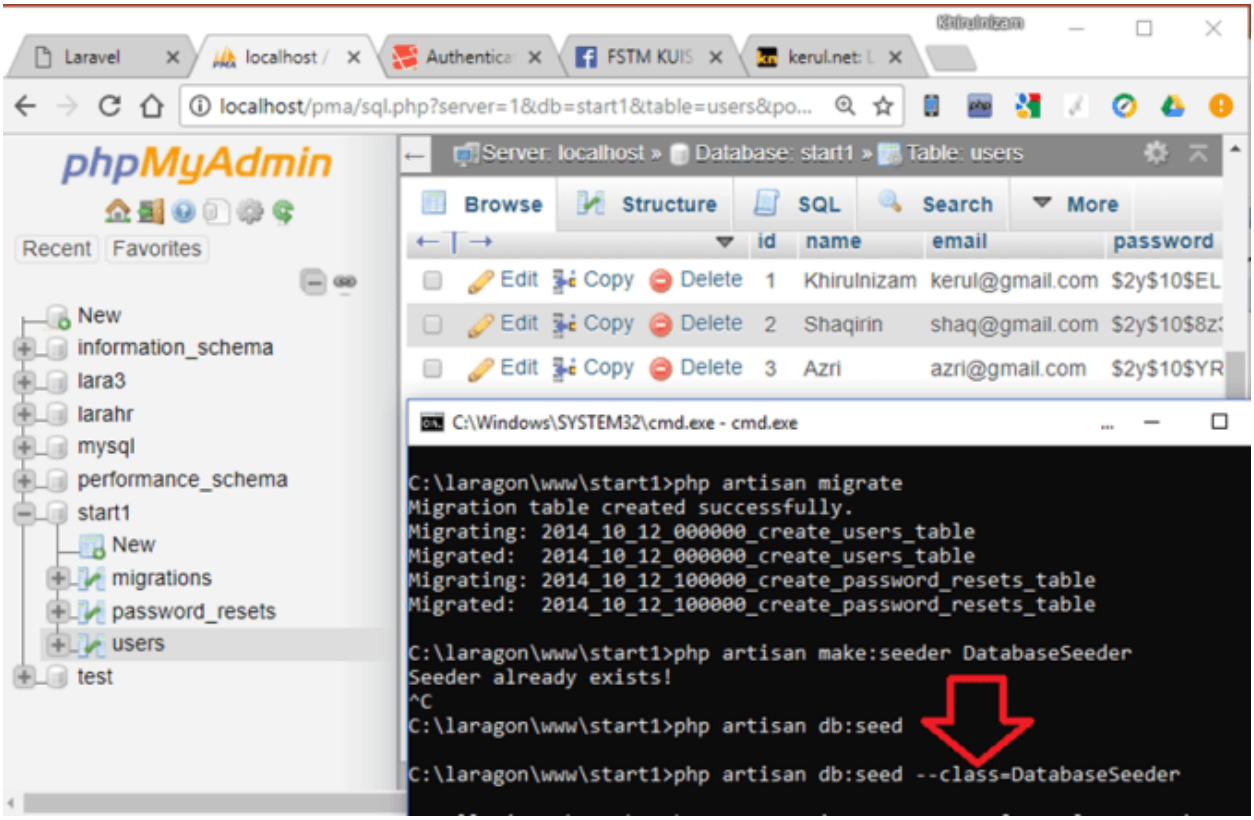
Seterusnya dalam DatabaseSeeder.php kita akan masukkan beberapa contoh rekod. Ada banyak cara untuk jana data contoh (dummy data). Laravel juga menyediakan penjana data (*data factory*). Akan dibincangkan dalam tutorial akan datang.



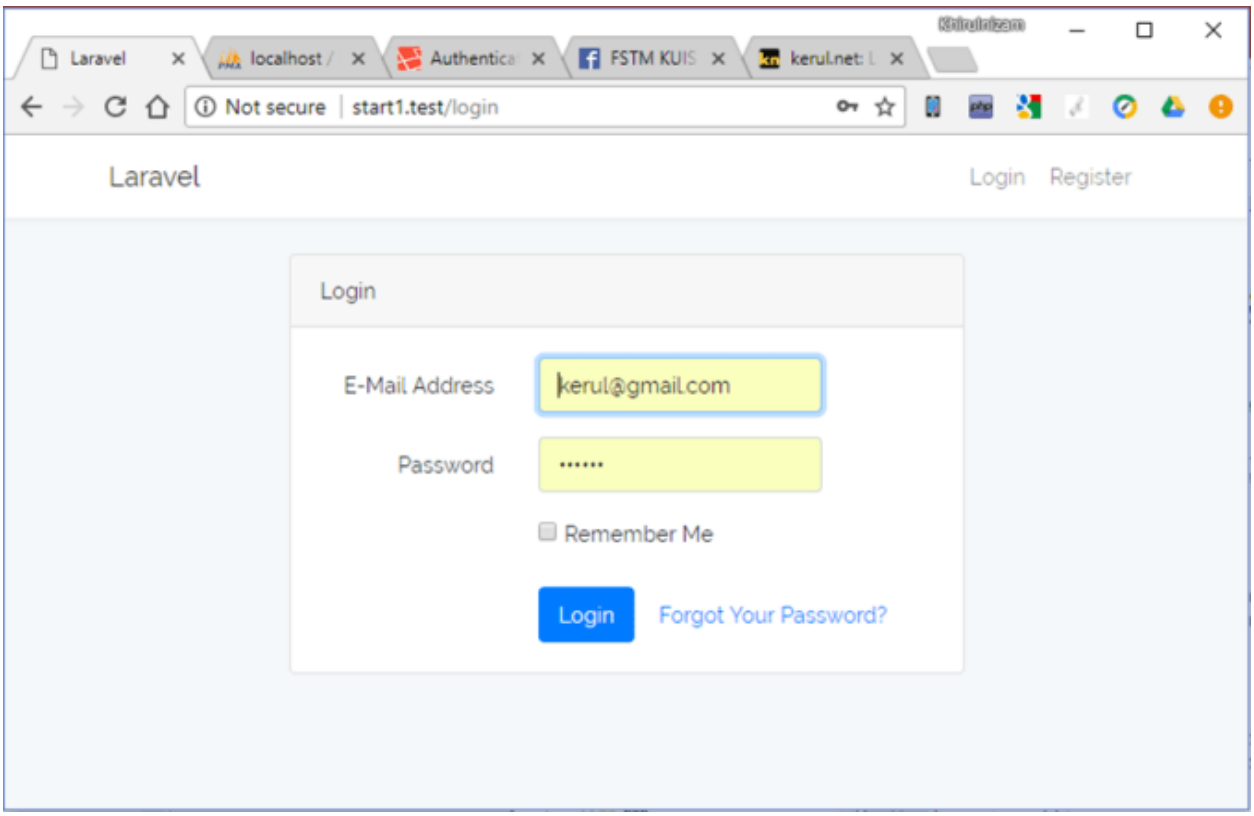
Data yang telah disediakan dalam DatabaseSeeder perlu dimasukkan ke dalam jadual (database). Boleh dilakukan dengan melaksana arahan berikut pada terminal;

php artisan db:seed

Dan hasilnya sila lihat dalam PMA jadual users telah diisi dengan data contoh.



Arahan pada make:auth tadi juga menjana antaramuka login/logout pengguna. Cuma perlu papar dalam browser, start1.test/login dan anda akan mendapat paparan di bawah yang berfungi untuk login/logout dan daftar pengguna. Selamat mencuba dan berjaya.

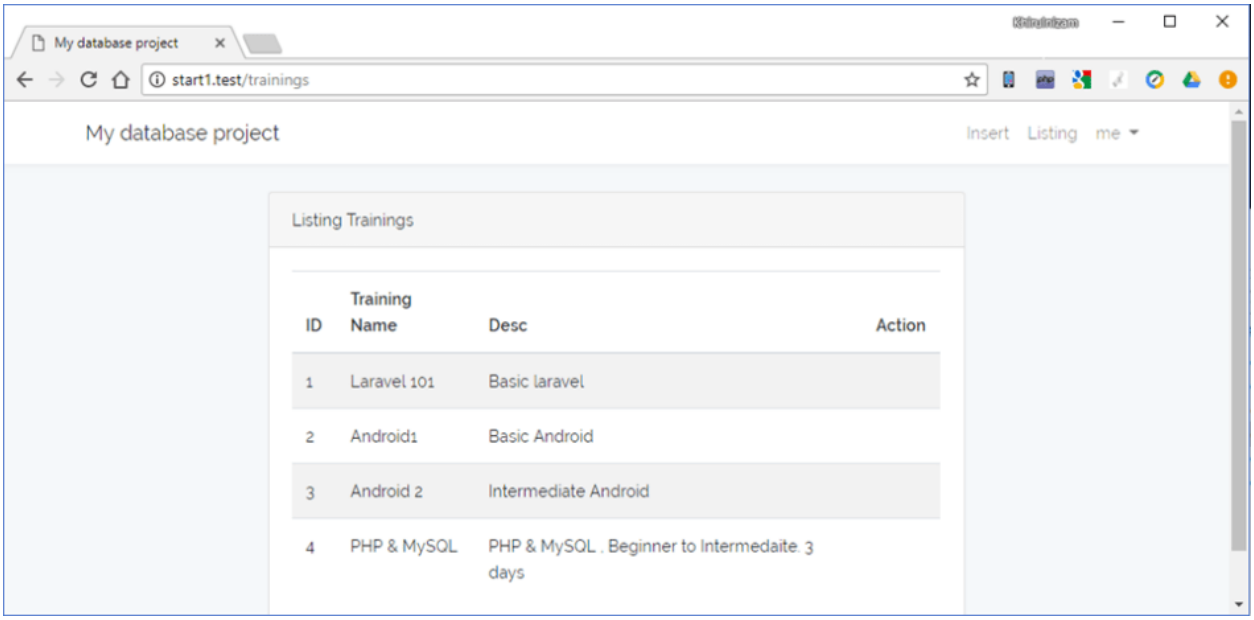


Selesai tutorial pertama Laravel. Komen/cadangan/bantuan sila email khirulnizam{at}gmail.com.
Bersambung seterusnya nanti. Insya-Allah...

Kod Sumber dalam GITHUB – https://github.com/khirulnizam/start1

(Slide for English version is available at http://bit.ly/laravelfstm1)

Tutorial 02 CRUD Pengenalan Laravel



Tutorial Laravel bahasa Melayu ini disambung dengan pemasangan LARAGON. Laragon adalah satu perisian kompilasi yang telah disediakan semua keperluan perisian pembangunan projek Laravel.

Rangka tutorial

- Membina MODEL (jadual pangkalan data /db)
- Membina VIEW (antaramuka pengguna)
- CONTROLLER & route
- Masukkan rekod baharu
- Senaraikan semua rekod
- Kemaskini rekod

Kod Sumber dalam GITHUB – https://github.com/khirulnizam/start1
Sebarang applikasi web yang menjana laman web dinamik memerlukan pangkalan data. Kaedah penyambungan pangkalan data telah dibincangkan dalam tutorial Laravel 1. Kali ini kita cuma perlu menambah jadual (*table*) dalam DB yang sedia ada.

MEMBINA MODEL (jadual pangkalan data)

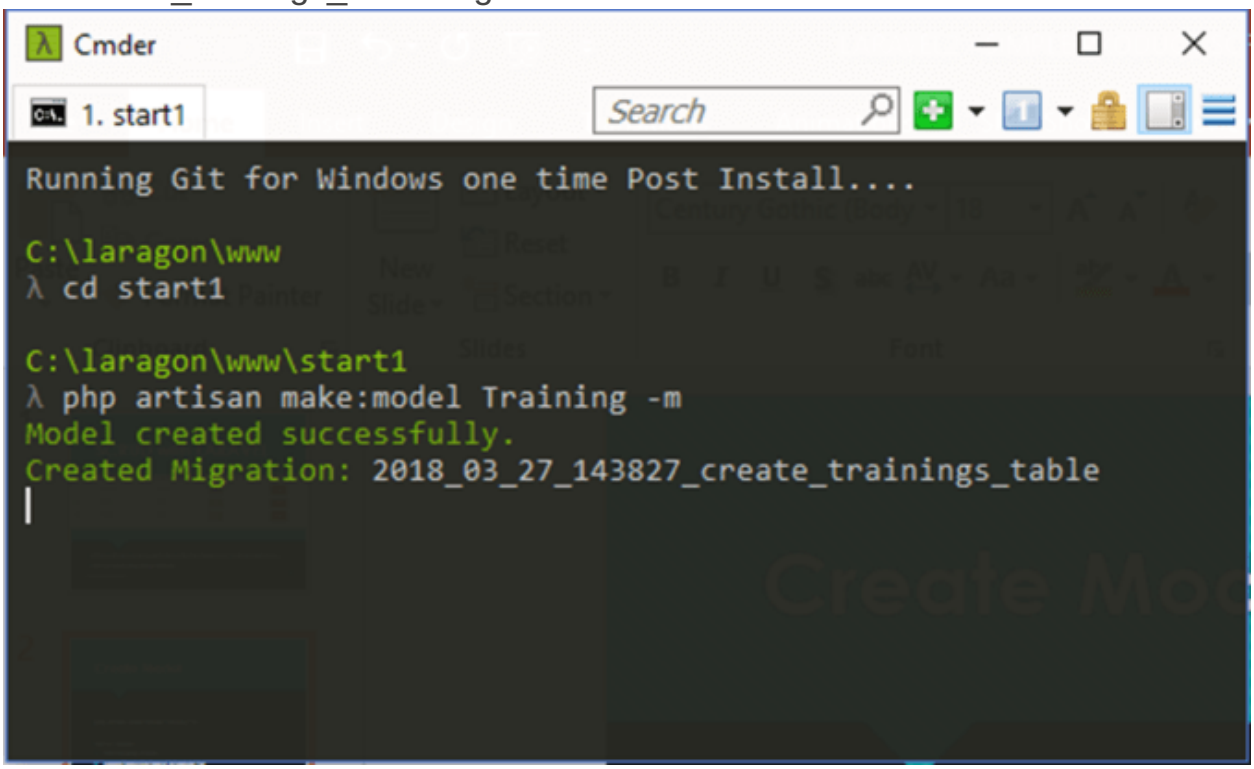
Laksanakan arahan berikut dalam *command shell* Laragon.

```
php artisan make:model Training –m
```

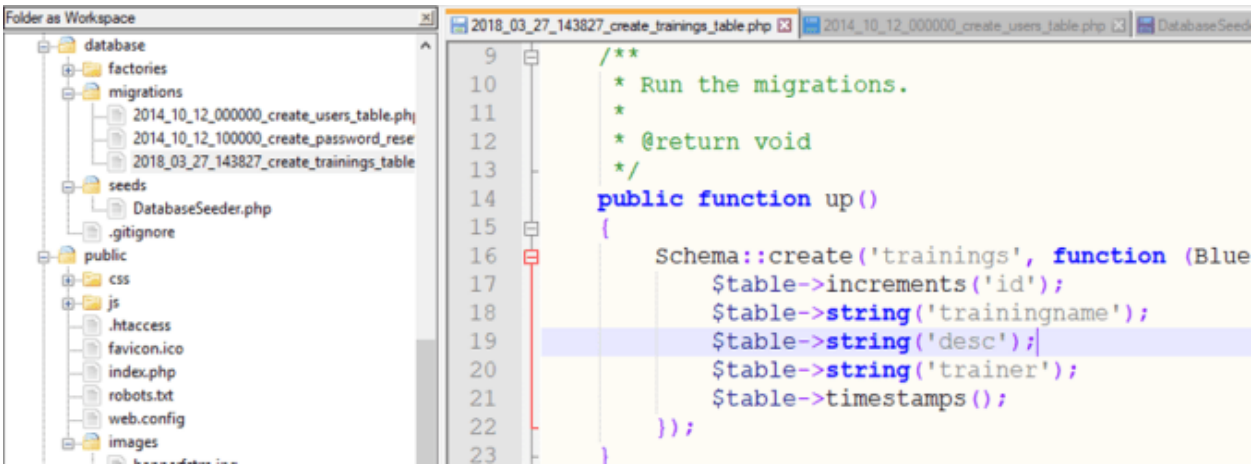
Arahan ini akan menyediakan dua item berikut dalam folde projek Laravel;

Fail model Training.php

Fail create_trainings_table migration

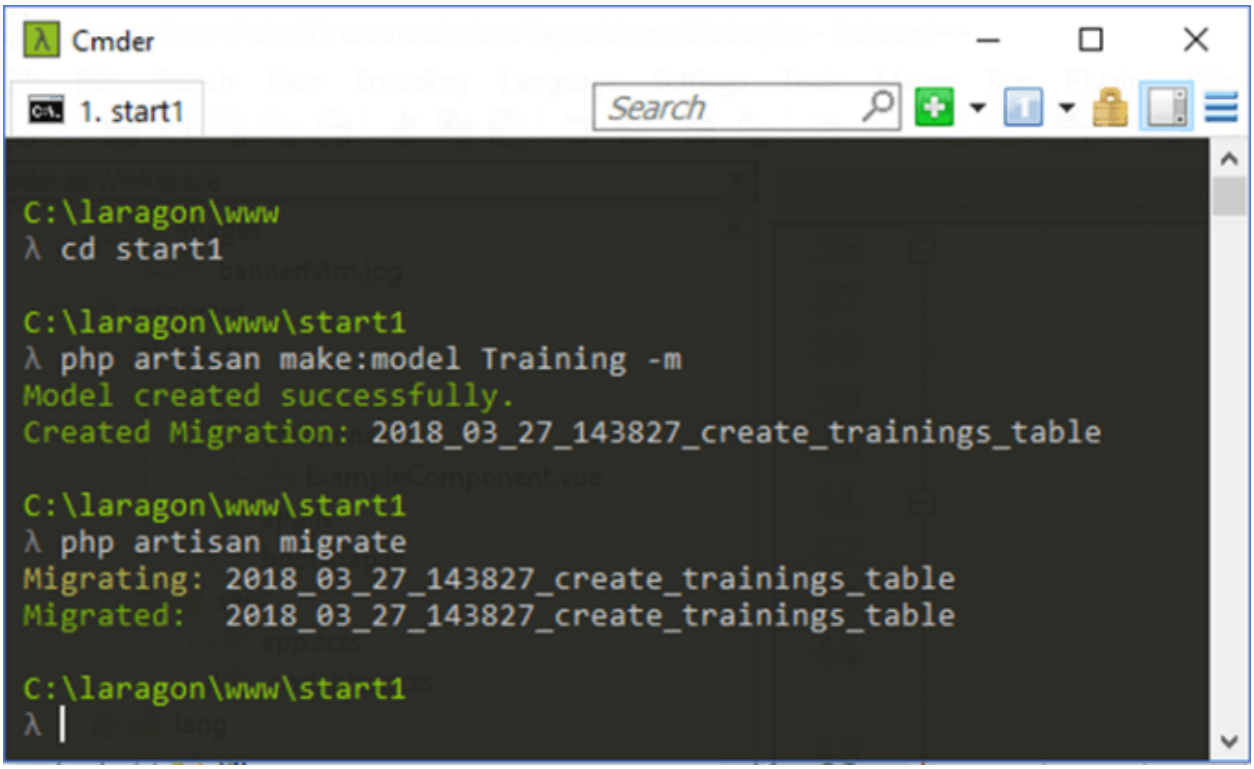


Dalam fail *create_trainings_table* , tulis kod berikut;



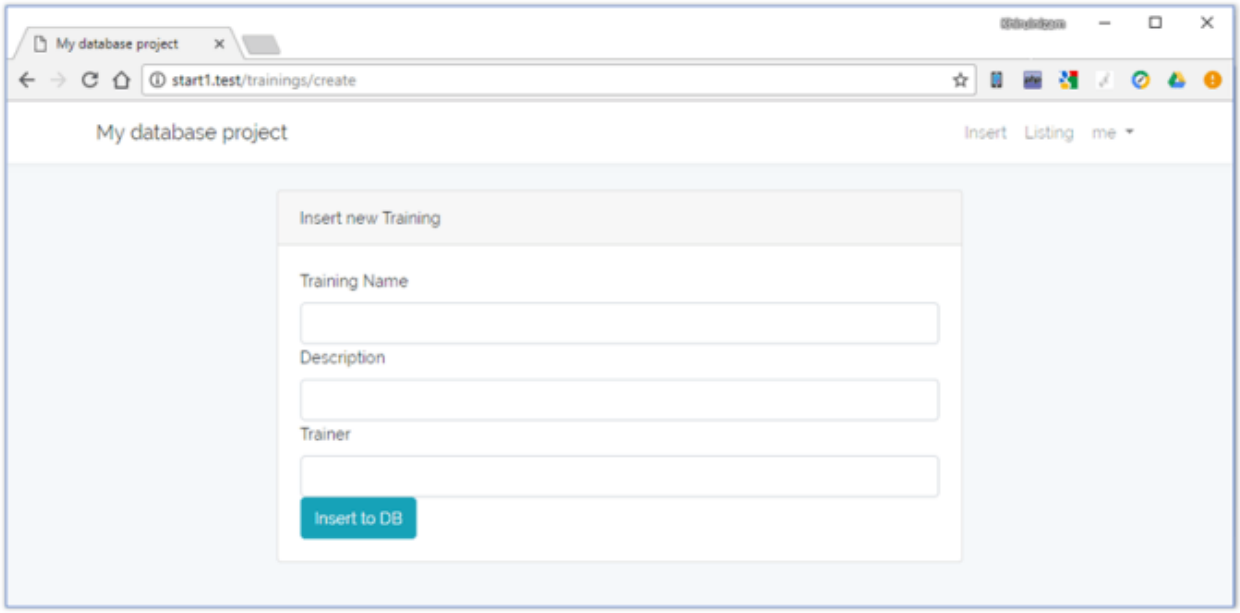
Kembali ke *command shell* Laragon dan laksnakan arahan *migrate.* Arahan migrate akan membina jadual baharu dengan struktur seperti dalam *create_trainings_table.*

php artisan migrate



BINA ANTARAMUKA (Views) untuk masukkan rekod baharu

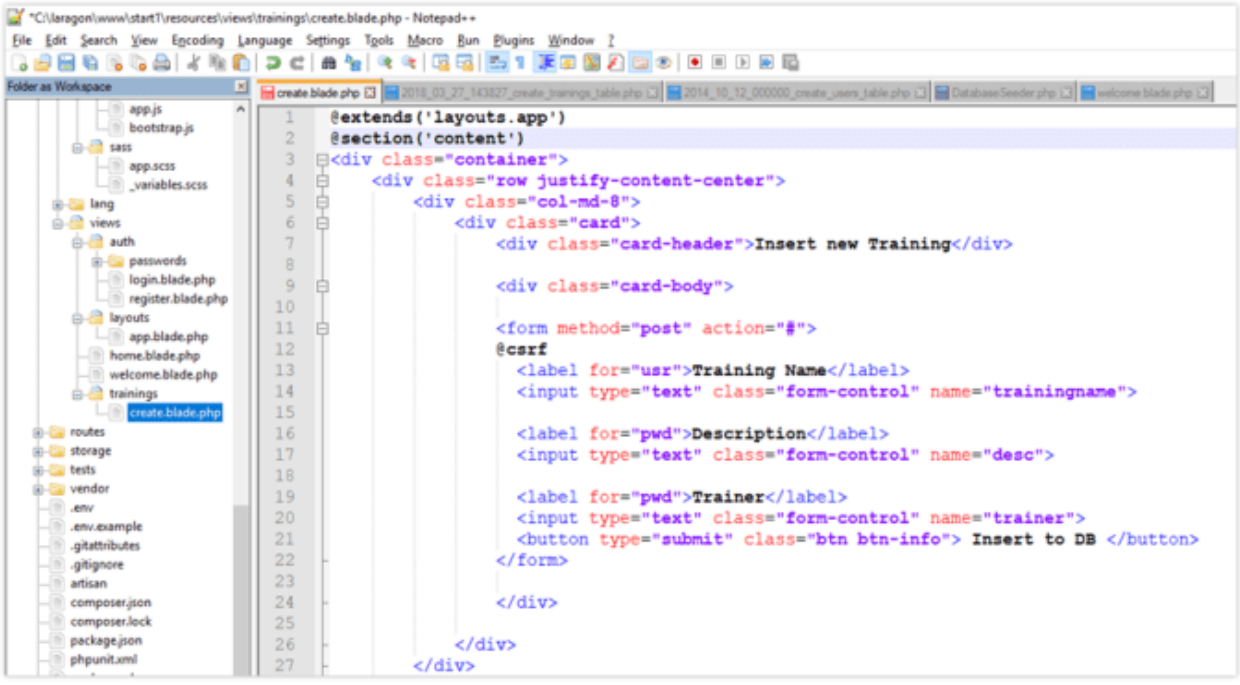
Rajah di bawah adalah antaramuka pengguna yang akan dibina untuk menerima input rekod yang baharu untuk disimpan dalam jadual yang baharau iaitu *Training* .



Proses untuk membuat fail antaramuka tersebut;

Buat satu folder untuk antaramuka ***trainings*** dalam folder *resources >> views*

Tambah satu fail *blade* baharu dengan nama ***create.blade.php***



```
@extends('layouts.app')
@section('content')


< div class = "container" >


< div class = "row justify-content-center" >
< div class = "col-md-8" >
< div class = "card" >
< div class = "card-header" >Insert new Training</ div >
< div class = "card-body" >
< form method = "post" action = "{{url('trainings')}}" >
@csrf
< label for = "usr" >Training Name</ label >
< input type = "text" class = "form-control" name =
"trainingname" >

< label for = "pwd" >Description</ label >
< input type = "text" class = "form-control" name =
"desc" >

< label for = "pwd" >Trainer</ label >
< input type = "text" class = "form-control" name =
"trainer" >
< button type = "submit" class = "btn btn-info" > Insert
to DB </ button >
</ form >



</ div >



</ div >


</ div >
```

```
</ div >

</ div >

@endsection
```

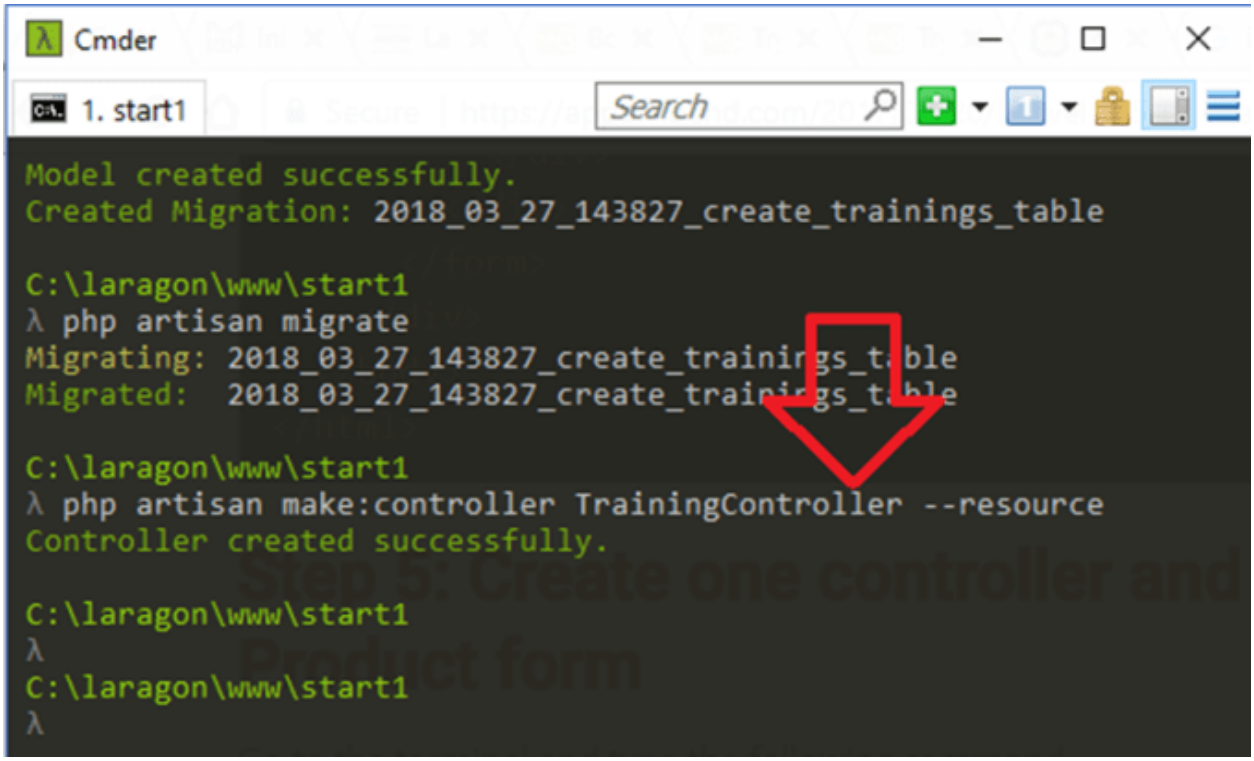
CONTROLLER & ROUTE

Seterusnya hubungkan antaramuka (*create.blade.php*) dengan pangkalan data, kita memerlukan Controller untuk melaksanakan aliran logik dan proses. Manakala *route* diperlukan sebagai laluan (*path*) kepada antaramuka/proses dalam *Controller* .

Untuk menambah Controller dalam modul Training ini, laksanakan arahan *make:controller* pada *Command Shell* .

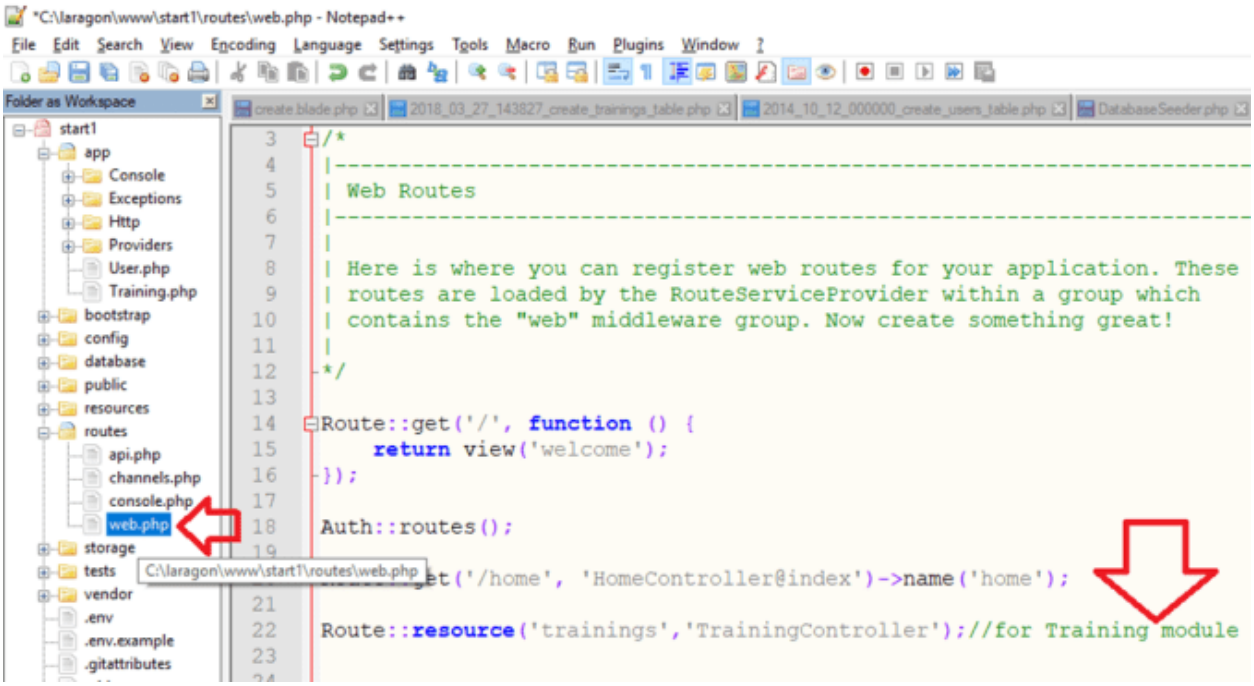
```
php artisan make:controller TrainingController --resources
```

Arahan ini akan mecipta fail *TrainingController.php* . –resouces bermaksud semua path yang mungkin terlibat untuk fungsi-fungsi dalam modul Training akan diwujudkan sama.



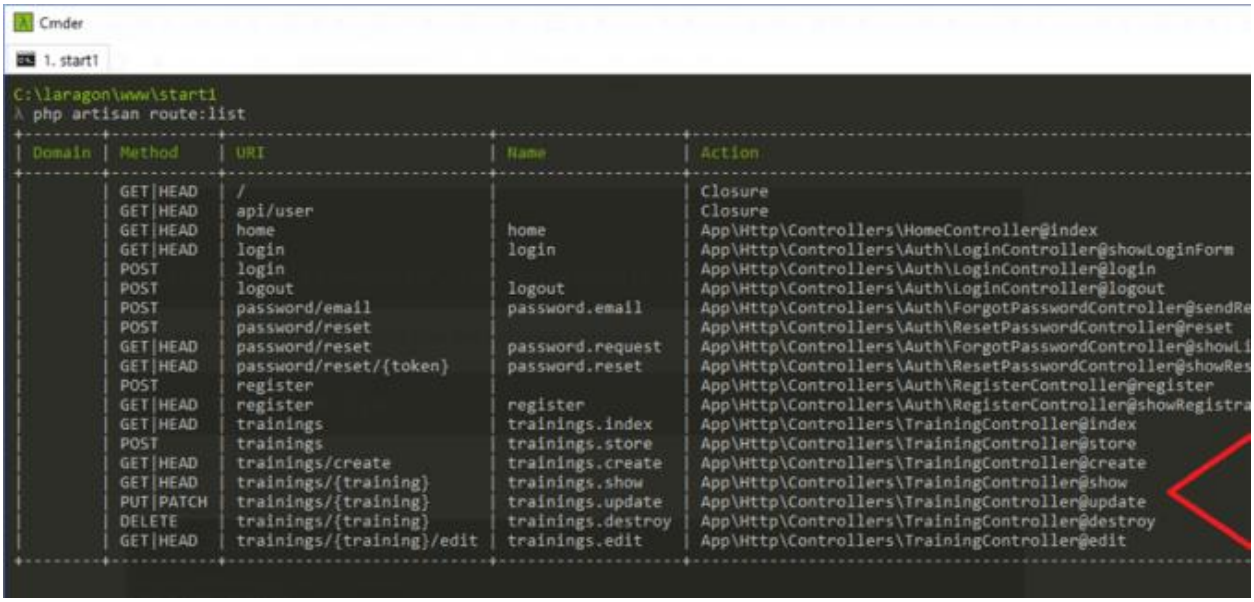
Dalam fail ***routes/web.php*** tuliskan baris kod di bawah seperti ditunjuk anak panah.

Route::resource('trainings','TrainingController');



Untuk menyenaraikan path kepada fungsi yang disediakan, laksanakan arahan route:list.

```
php artisan route:list
```



Domain	Method	URI	Name	Action
	GET|HEAD	/		Closure
	GET|HEAD	api/user		Closure
	GET|HEAD	home	home	App\Http\Controllers\HomeController@index
	GET|HEAD	login	login	App\Http\Controllers\Auth\LoginController@showLoginForm
	POST	login		App\Http\Controllers\Auth\LoginController@login
	POST	logout	logout	App\Http\Controllers\Auth\LoginController@logout
	POST	password/email	password.email	App\Http\Controllers\Auth\ForgotPasswordController@sendRe
	POST	password/reset		App\Http\Controllers\Auth\ResetPasswordController@reset
	GET|HEAD	password/reset	password.request	App\Http\Controllers\Auth\ForgotPasswordController@showLi
	GET|HEAD	password/reset/{token}	password.reset	App\Http\Controllers\Auth\ResetPasswordController@showRes
	POST	register		App\Http\Controllers\Auth\RegisterController@register
	GET|HEAD	register	register	App\Http\Controllers\Auth\RegisterController@showRegistra
	GET|HEAD	trainings	trainings.index	App\Http\Controllers\TrainingController@index
	POST	trainings	trainings.store	App\Http\Controllers\TrainingController@store
	GET|HEAD	trainings/create	trainings.create	App\Http\Controllers\TrainingController@create
	GET|HEAD	trainings/{training}	trainings.show	App\Http\Controllers\TrainingController@show
	PUT|PATCH	trainings/{training}	trainings.update	App\Http\Controllers\TrainingController@update
	DELETE	trainings/{training}	trainings.destroy	App\Http\Controllers\TrainingController@destroy
	GET|HEAD	trainings/{training}/edit	trainings.edit	App\Http\Controllers\TrainingController@edit

Perhatikan path yang disediakan disertakan ‘s’ dihujung (training **s**). Path yang automatik disertakan adalah;

trainings.index (fungsi senaraikan rekod),

training.store (fungsi simpan rekod),

trainings.create (fungsi simpan rekod baharu),

trainings.update (fungsi perbaharui rekod sedia ada),

trainings.destroy (fungsi memadam rekod),

trainings.edit (fungsi borang kemaskini rekod).
Seterusnya, akan ditunjukkan bagaimana nak paparkan borang kemasukan rekod baharu yang telah disediakan kod dalam fail *create.blade.php* .

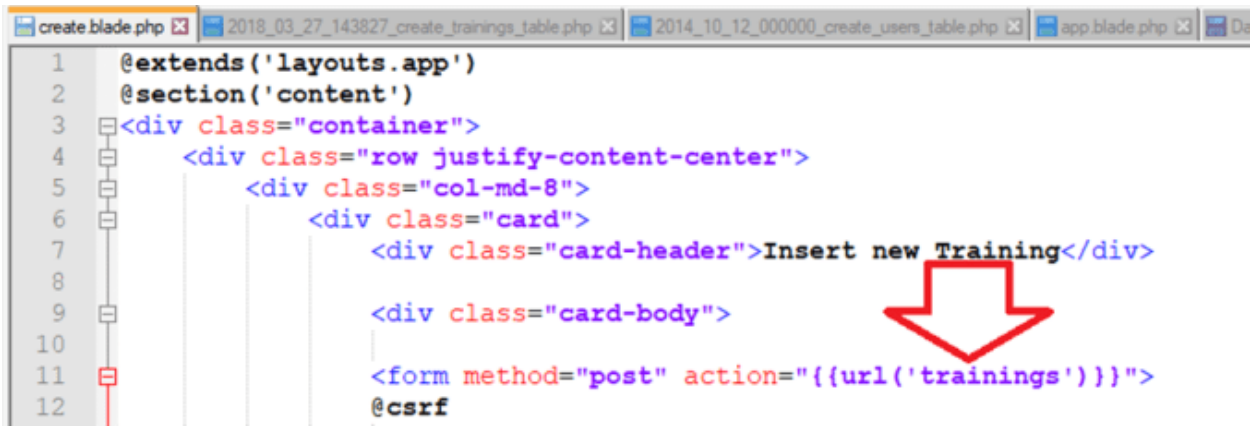
Buka fail TrainingController.php dalam folder *app/Http/Controllers/* dan tambahkan baris kod berikut ke dalam *function create()* .

```
TrainingController.php | Handler.php | app.php | auth.php | home.blade.php | app.blade.php | web.php

        \Illuminate\Http\Response
23      */
24      //TrainingController.php
25     public function create()
26     {
27         //display form
28         return view (
               'trainings.create');
29     }
```

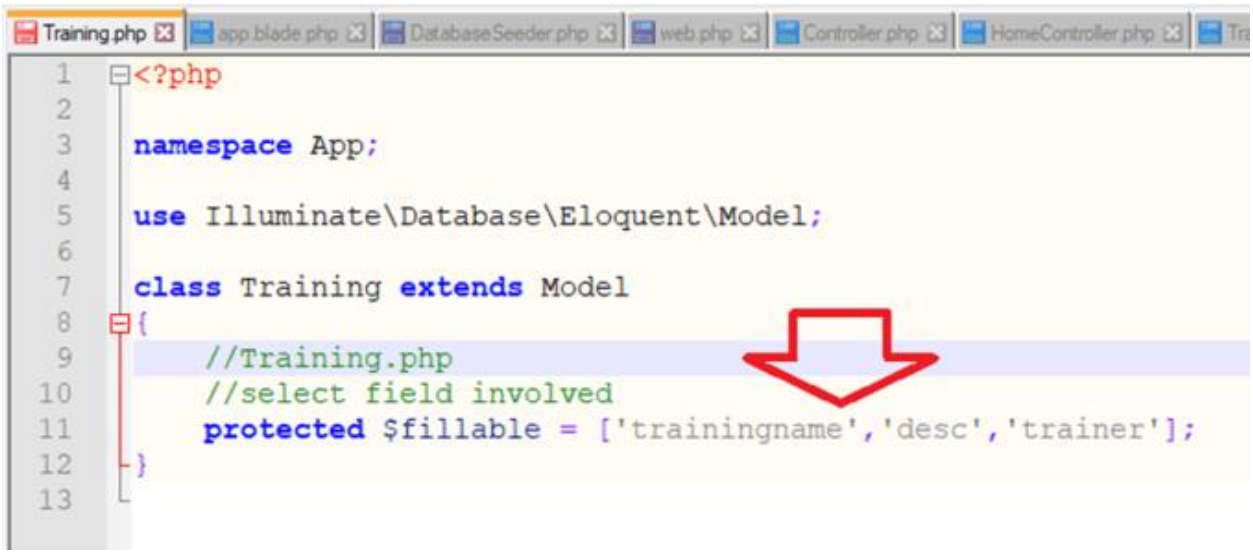
Untuk menyediakan butang/link kepada borang masuk rekod baharu, tambah kod berikut ke fail template, *resources/views/layouts/app.blade.php* .

```
app.blade.php | DatabaseSeeder.php | web.php | Controller.php | HomeController.php | TrainingController.php | home.blade.php

38                  </ul>
39
40                  <!-- Right Side Of Navbar -->
41                  <ul class="navbar-nav ml-auto">
42                      <!-- Authentication Links -->
43                      @guest
44                          <li><a class="nav-link" href="{{ route('login') }}">{{ __('Login') }}
                            </a></li>
45                          <li><a class="nav-link" href="{{ route('register') }}">{{
                            __('Register') }}</a></li>
46                      @else
47                          <li><a class="nav-link" href="{{ url('trainings/create') }}">Insert
                            </a></li>
48                          <li><a class="nav-link" href="#">Listing</a></li>
49                          <li class="nav-item dropdown">
50                              <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#"
                                role="button" data-toggle="dropdown" aria-haspopup="true"
                                aria-expanded="false" v-pre>
51                                  {{ Auth::user()->name }} <span class="caret"></span>
52                              </a>
53
54                              <div class="dropdown-menu" aria-labelledby="navbarDropdown">
```

Dalam form action pula, hantar rekod baharu ke link berikut untuk proses penyimpanan data ke pangkalan data (db).



Seterusnya dalam fail *app/Training.php* , isikan senarai lajur jadual yang terlibat. Rujuk rajah di bawah dan tambah kod berikut;



Sekiranya kita rujuk kepada *route:list* , *path* 'trainings' akan menuju kepada fungsi *store* (simpan) sekiranya data dalam bentuk *post* .

Jadi seterusnya, tambah kod berikut dalam function store(), fail *app/Http/Controllers/TrainingController.php.* Kod di bawah merupakan proses simpan rekod dalam pangkalan data.

Masukkan baris berikut di bahagian atas TrainingController.php;

```
use App\Training; //include the namespace of Training.php
```

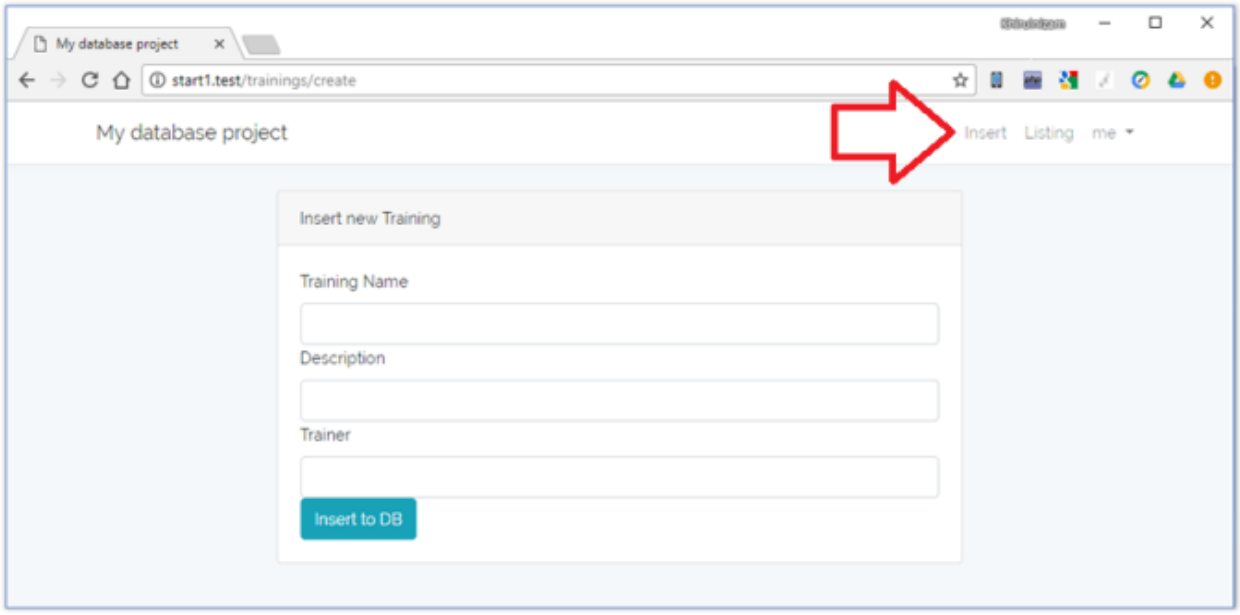
Baris 51: merupakan arahan *redirect* yang akan memaparkan semula borang, dengan membawa mesej rekod berjaya disimpan.

```php
    */

    //// TrainingController.php
    public function store(Request $request)
    {
        //insert new record process is here
        $training = $this->validate(request(), [
          'trainingname' => 'required',
          'desc' => 'required',
          'trainer' => 'required'
        ]);

        Training::create($training);

        return back()->with('success', 'Training has been added');
    }
```

Seterusnya dalam fail *app/Training.php* , isikan senarai lajur jadual yang terlibat. Rujuk rajah di bawah dan tambah kod berikut;

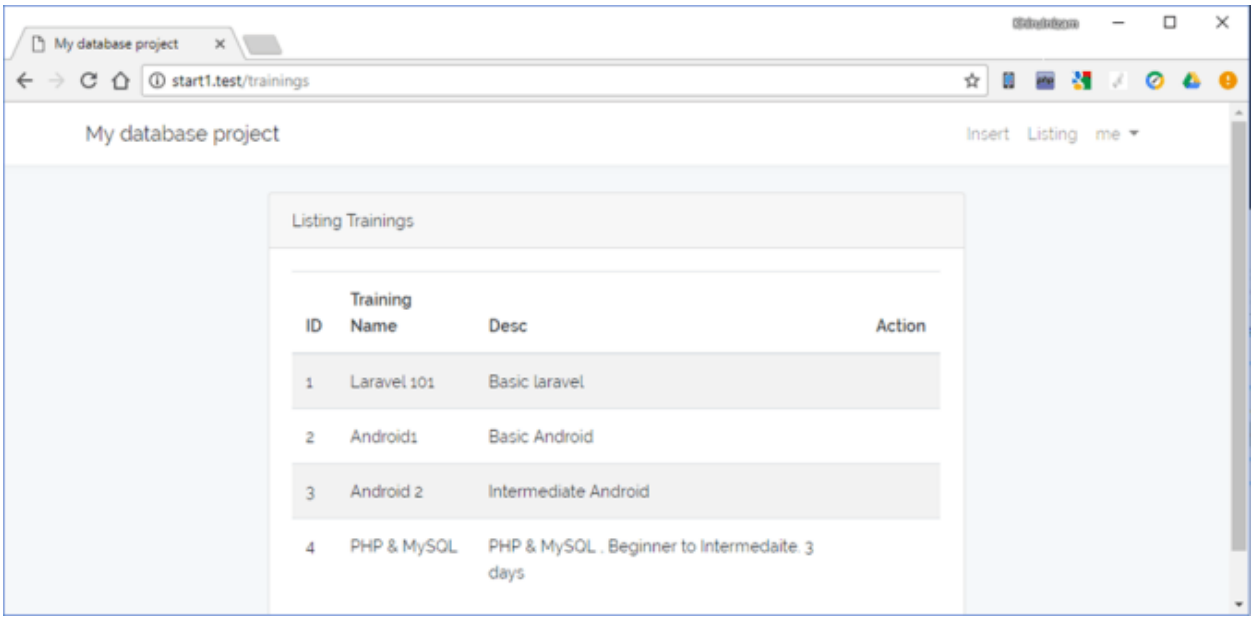
```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Training extends Model
{
    //Training.php
    //select field involved
    protected $fillable = ['trainingname','desc','trainer'];
}
```

Alhamdulillah, selesai fungsi masukkan rekod baharu. Bagaimana nak uji? Klik pada link Insert ditunjukkan anak panah.



SENARAIKAN REKOD

Sambungan latihan, menyenaraikan semua rekod dalam jadual. Kita akan menghasilkan antaramuka seperti di bawah.



Berdasarkan *route:list* , kita boleh buat penyenaraian dalam function index( ), fail *app/Http/Controllers/TrainingController.php*

Tulis kod berikut dalam function index( );

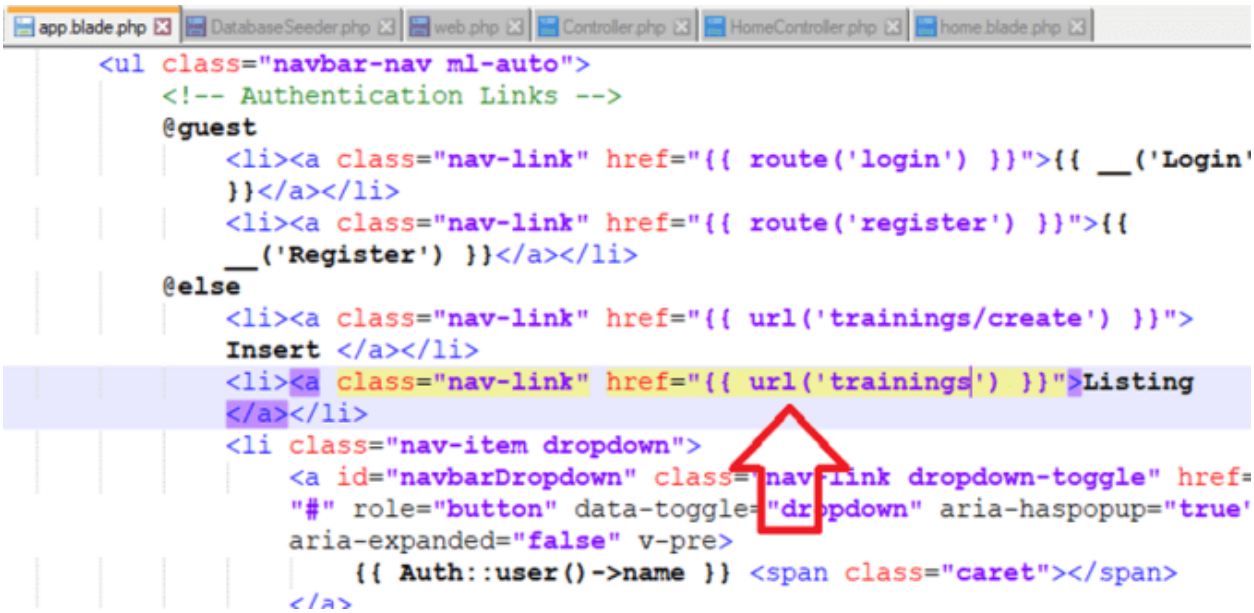
Antaramuka yang melibatkan pembentukan jadual dinamik beserta senarai rekod boleh dibina dalam fail *resources/views/trainings/index.blade.php* . Sekiranya fail blade in belum wujud, sila *create new file* .


```
@extends('layouts.app')
@section('content')

< div class = "container" >

< div class = "row justify-content-center" >

< div class = "col-md-8" >
```

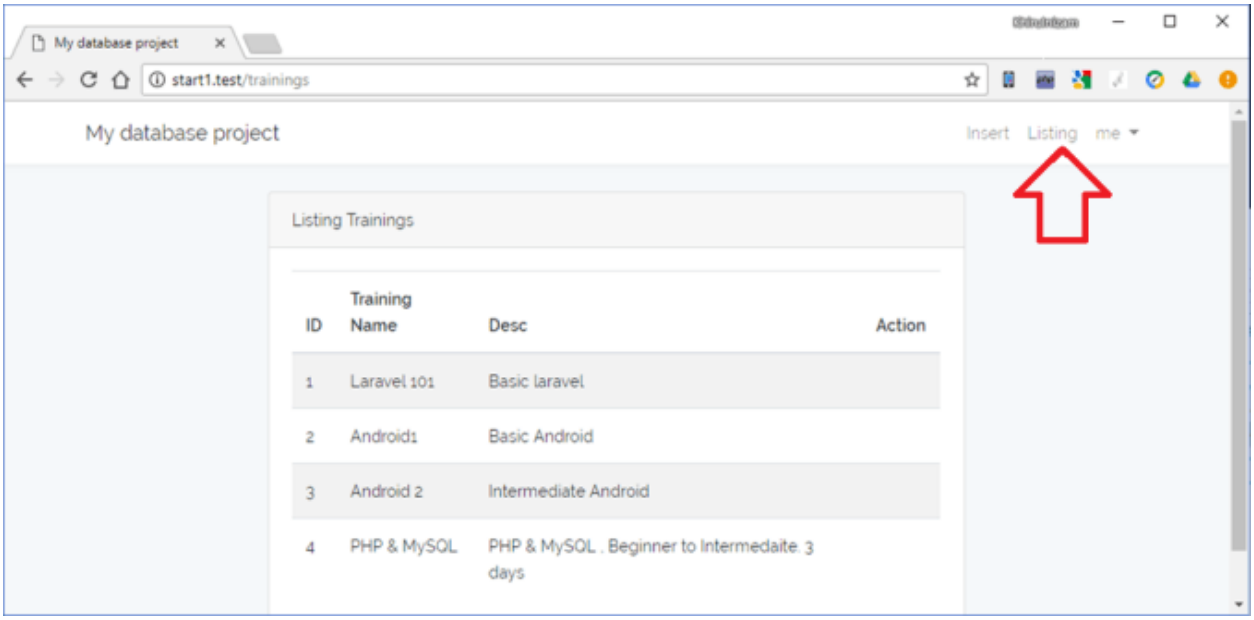
```
< div class = "card" >

< div class = "card-header" >Listing Trainings</ div >


< div class = "card-body" >

< table class = "table table-striped" >

< thead >

< tr >

< th >ID</ th >


< th >Training Name</ th >


< th >Desc</ th >


< th >Action</ th >

</ tr >

</ thead >


< tbody >
@foreach($trainings as $training)

< tr >

< td >{{$training['id']}}</ td >


< td >{{$training['trainingname']}}</ td >


< td >{{$training['desc']}}</ td >
```

```
< td ></ td >

</ tr >

@endforeach

< tbody >
</ table >


</ div >

</ div >

</ div >

</ div >

</ div >

@endsection
```

Tambahkan link kepada antaramuka ini, buka fail resources/views/layouts/app.blade.php .



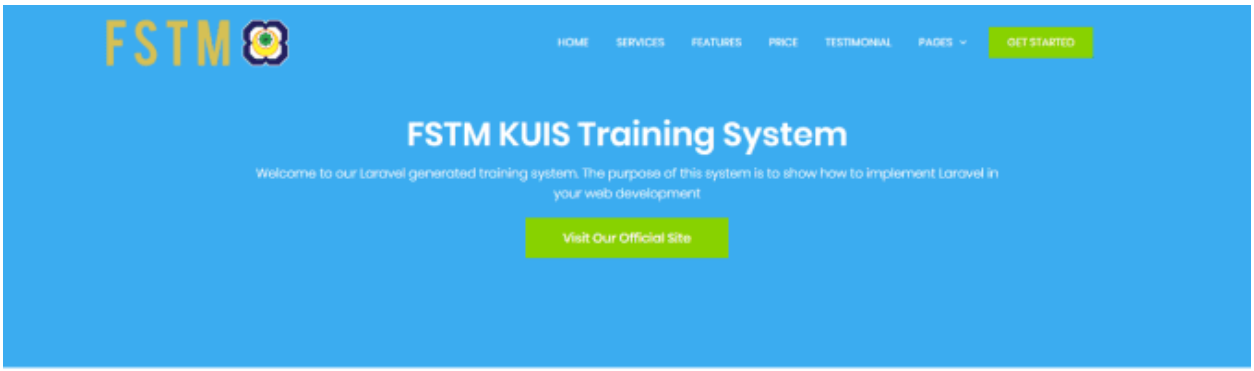
Cuba fungsi senarai semua rekod di sini;

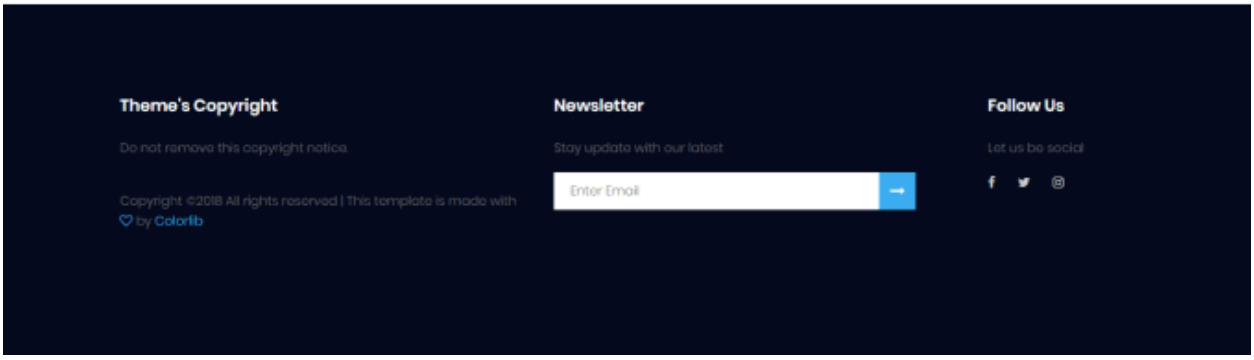


Kod Sumber dalam GITHUB – https://github.com/khirulnizam/start1

Tutorial 03 Ubah design Antaramuka

Berikut merupakan design Bootstrap yang boleh digunakan percuma. Bagaimana nak masukkan design ini dalam antaramuka sistem Laravel seperti rajah di bawah?



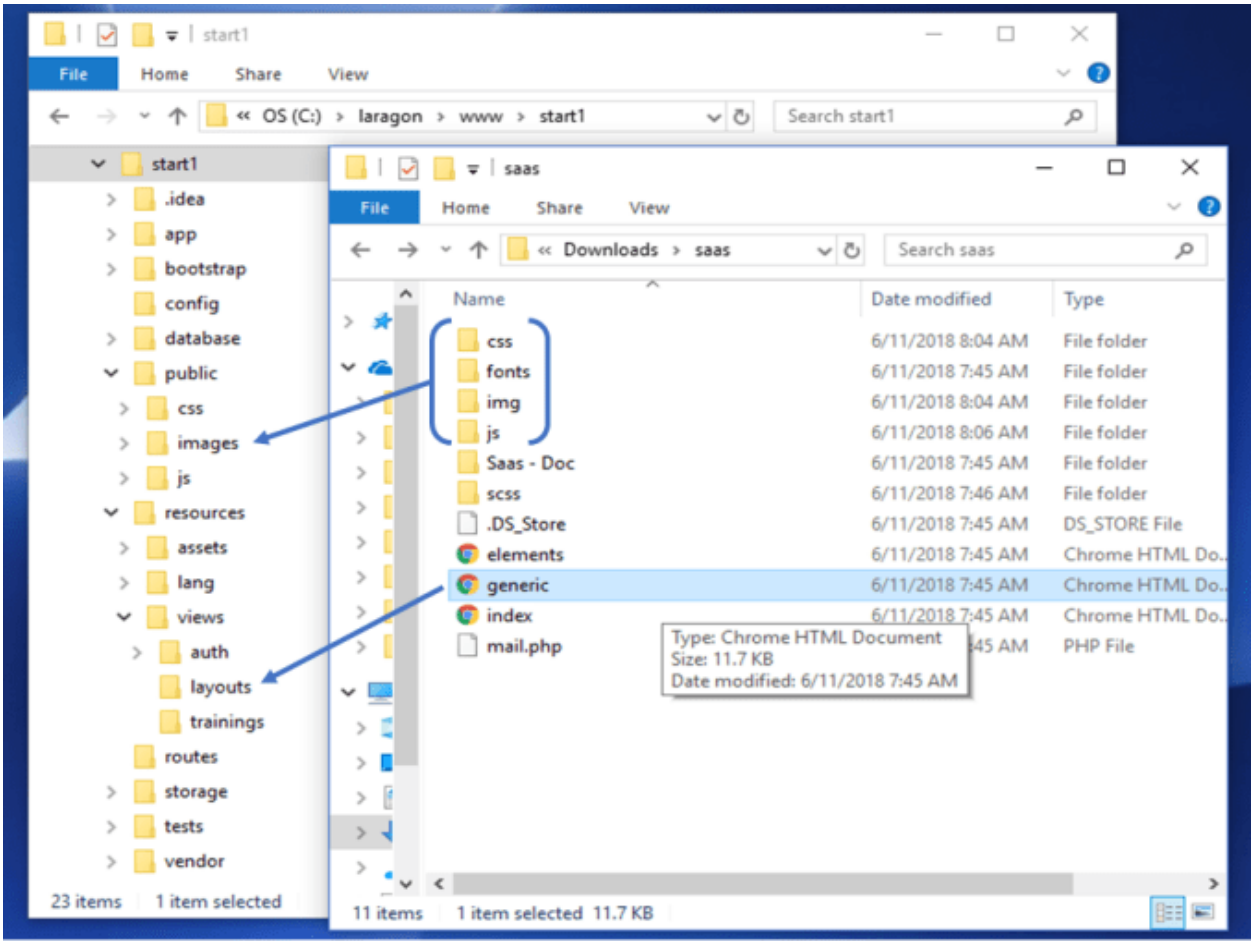


Muat-turun tema/template Bootstrap

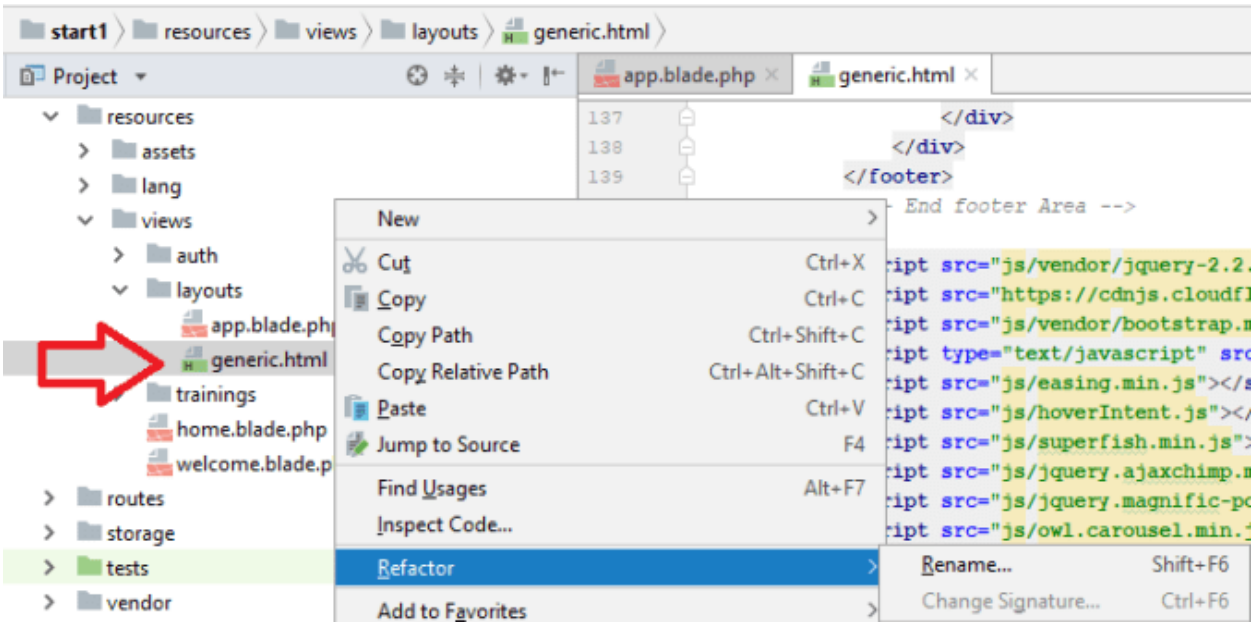
Buat carian untuk download template percuma/berbayar. Dalam tutorial ini, saya menggunakan template dari ColorLib, **https://colorlib.com/wp/template/saas/**

Selepas berjaya muat-turun fail template ColorLib Saas, ekstrak fail zip. Ini kandungan fail template.

Pindahkan senarai fail seperti dalam arahan rajah.



Tukar nama fail HMTL, contoh generic.html kepada *generic.blade.php* . Ini kerana untuk mengoptimasi kebolehan Blade dalam Laravel, fail HTML mesti dinamakan *namafail.blade.html* . (Menggunakan *PHPStorm*) Klik-kanan fail HTML, *refactor* dan *rename* .

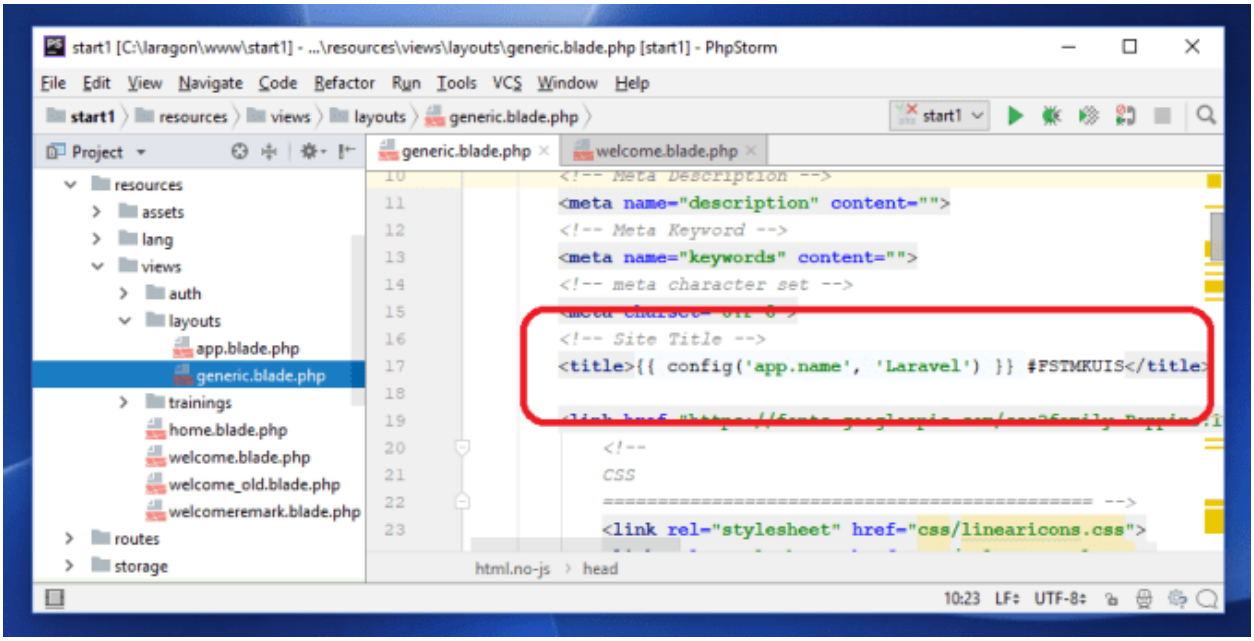


Penulis telah menukar template kepada layout yang ringkas. Boleh perolehi kod dalam **pastebin.com/tgaFCjCm** buat masa ini. Akan datang ada di GitHub.

Berikut beberapa penerangan tentang kod Blade yang biasa akan dijumpai. Skrip Blade biasanya disempadani oleh simbol berikut;

{{ … }}

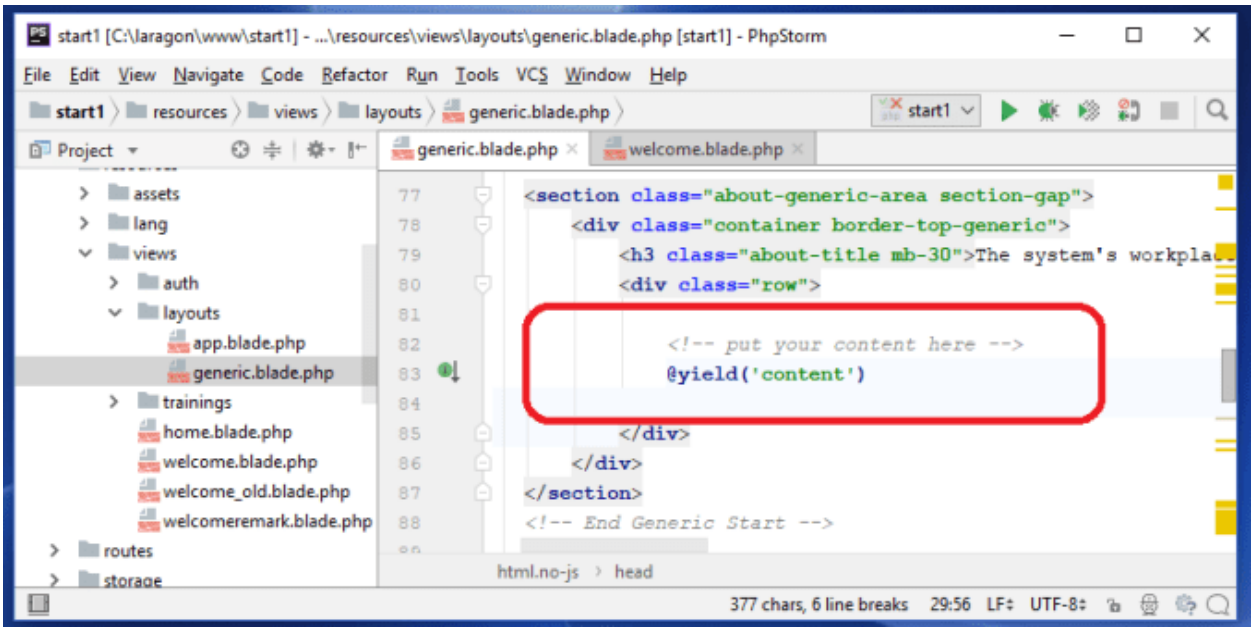
Seperti contoh di bawah, {{ config('app.name'),'Laravel' }} merujuk kepada arahan Blade untuk paparkan nama default applikasi web projek ini. Mana nak cari setting nama default applikasi, rujuk dalam fail ***.env*** .



Kod Sumber dalam GITHUB – https://github.com/khirulnizam/start1

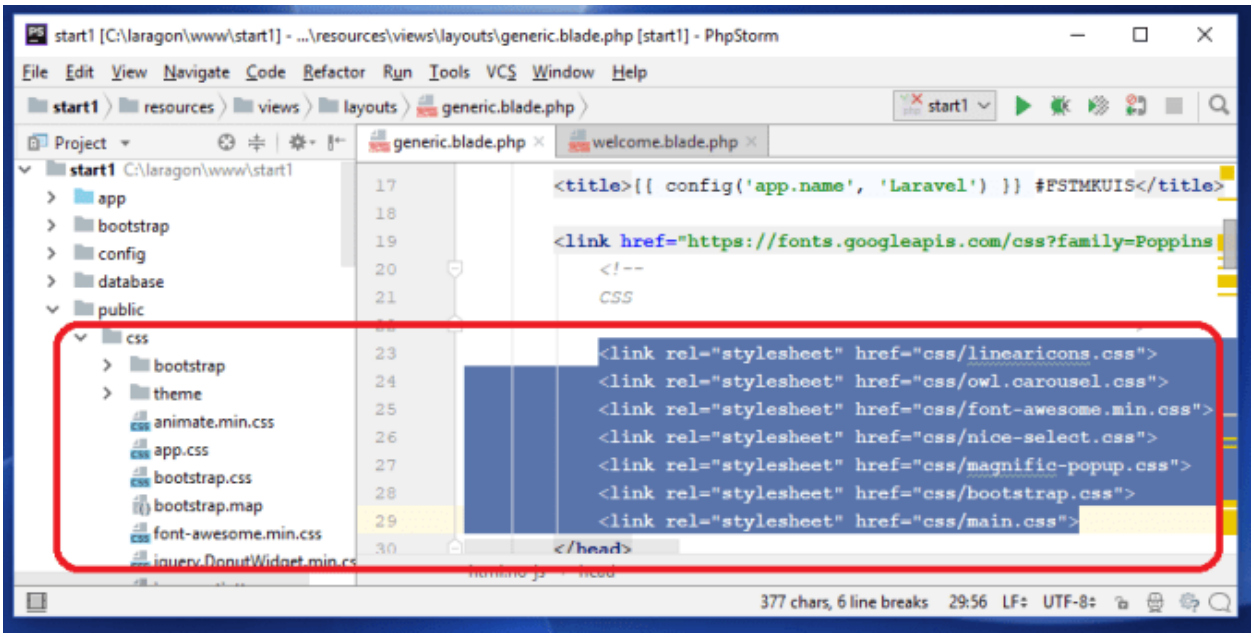
Memanggil fail kandungan page

generic.blade.php merupakan satu fail template HTML, maksudnya ia adalah fail yang akan digunakan sebagai template semua laman web dalam sistem anda. Yang akan menentukan kandungan berubah anda dalam arahan yield('content') seperti di bawah.

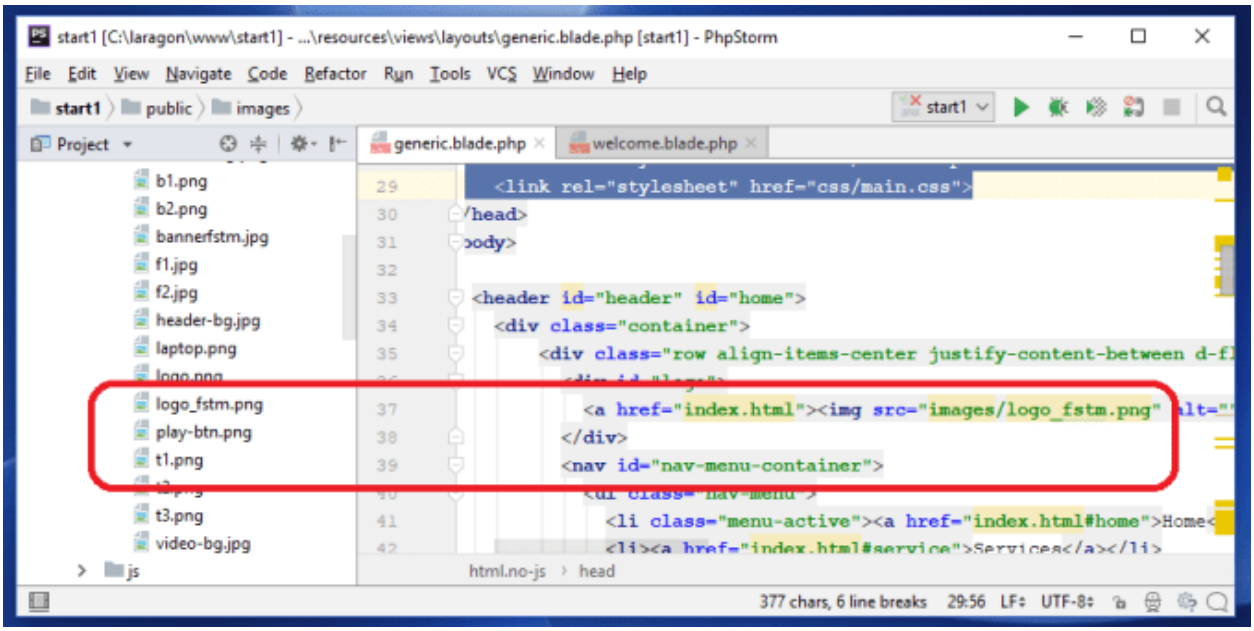


Folder CSS, imej & JavaScript

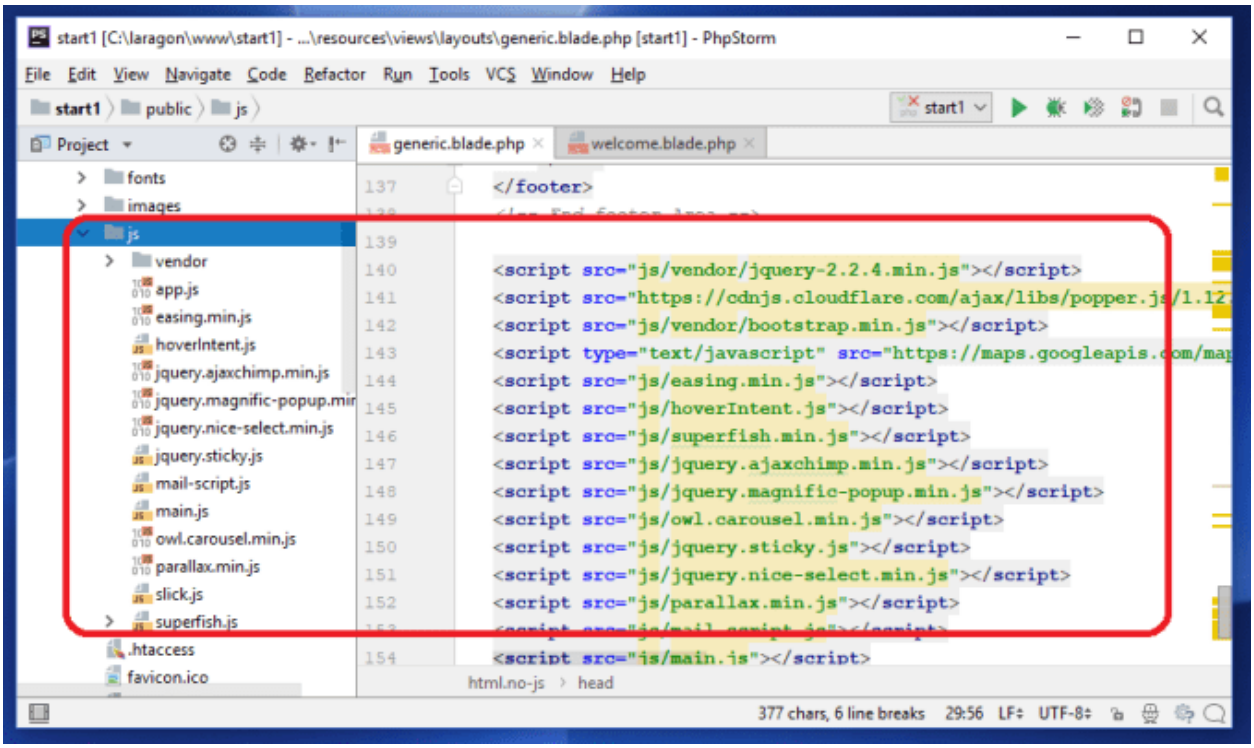
Fail kerangka Bootstrap CSS ada dalam folder ***public/css*** seperti yang telah kita pindahkan peringkat awal tadi.



Fail imej kita kumpulkan dalam folder ***public/images*** seperti yang telah kita pindahkan peringkat awal tadi.



Fail kerangka JavaScript ada dalam folder ***public/js*** seperti yang telah kita pindahkan peringkat awal tadi.

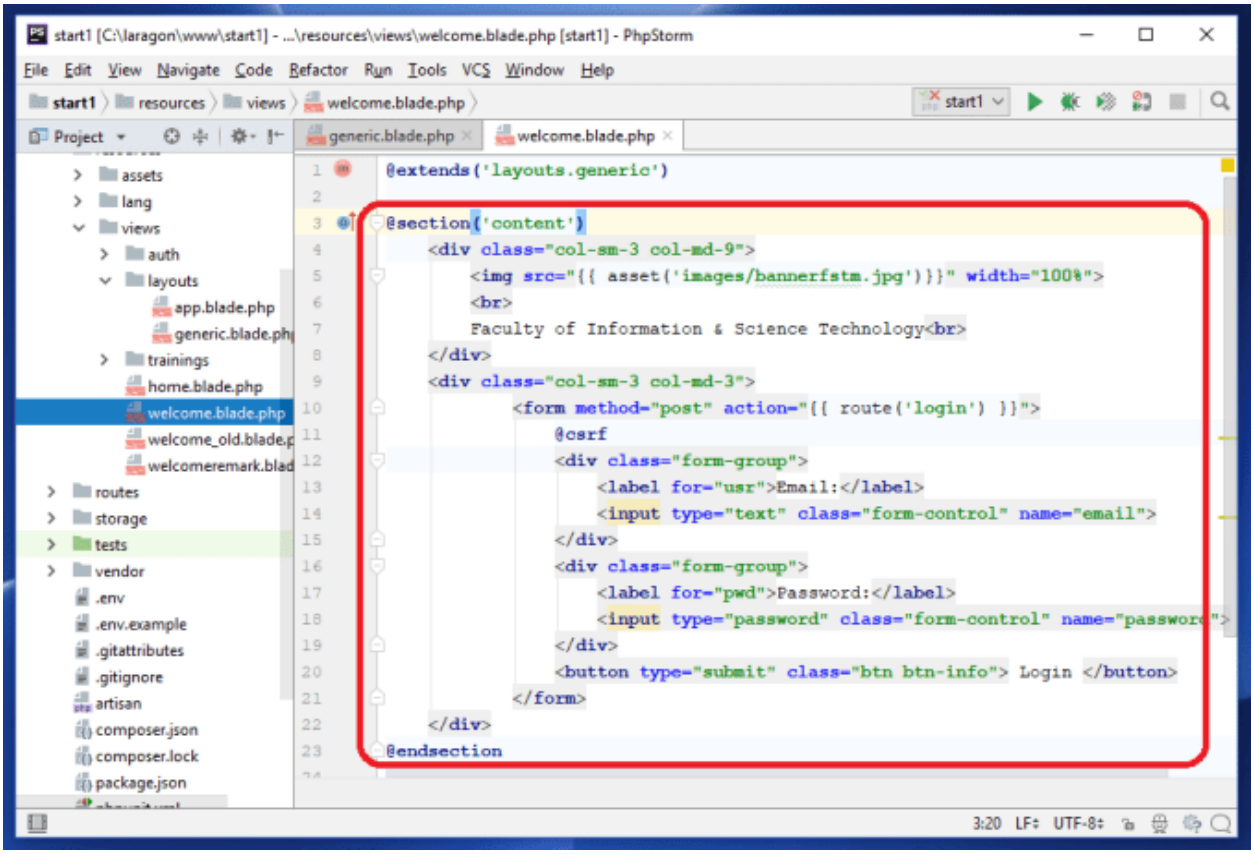


Laman pertama sistem

Laman pertama yang akan dipanggil dalam sistem anda berada dalam fail ***welcome.blade.php*** . Jadi dalam fail ini akan diletakkan kandungan untuk page *welcome* .

Kod boleh diperolehi dari **pastebin.com/W96GnW7D**

Kod Sumber dalam GITHUB – https://github.com/khirulnizam/start1

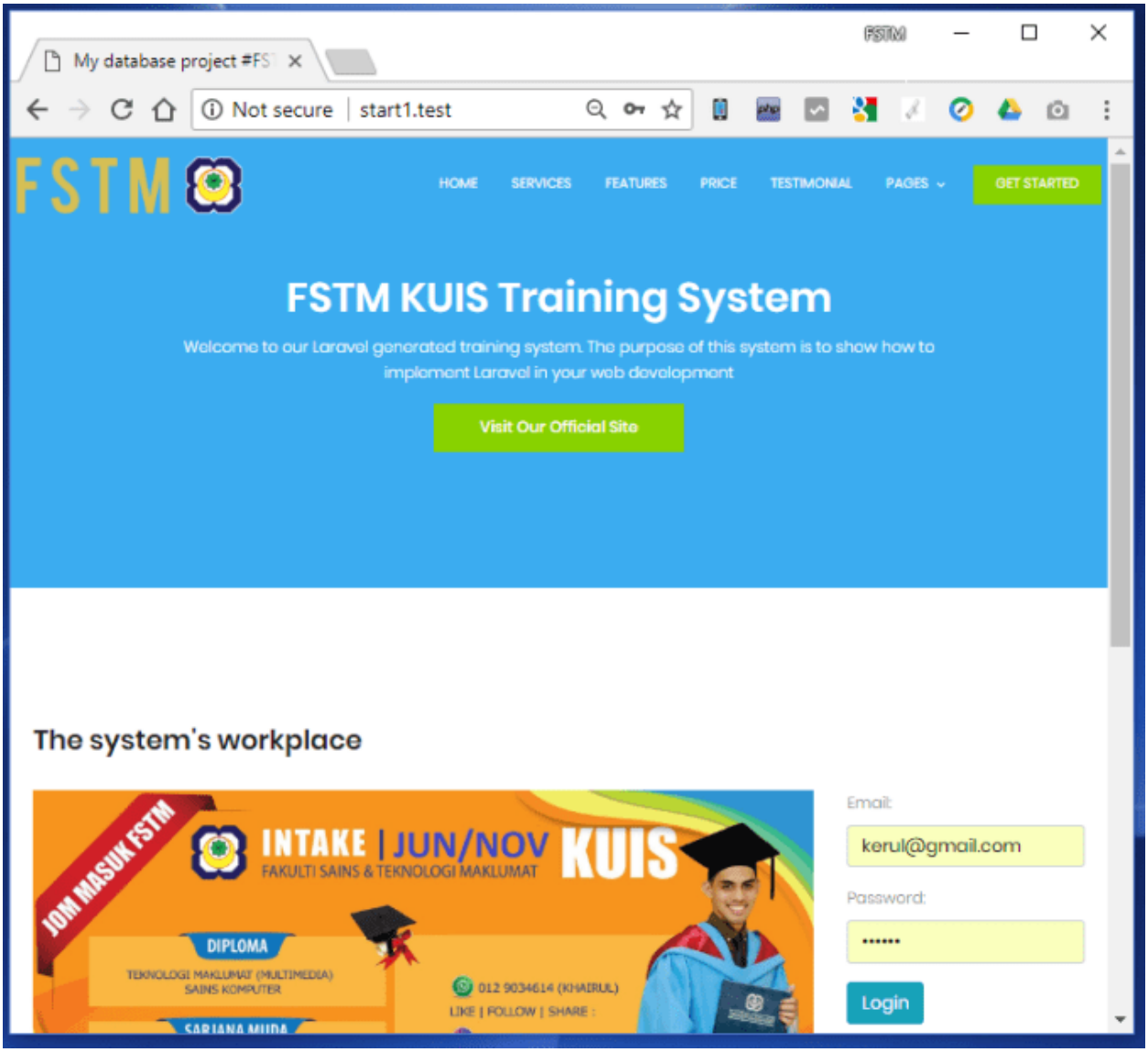


@extends(‘layouts.generic’) merujuk kepada fail antaramuka HTML ini akan menggunakan resources/views/layout/generic.blade.html sebagai fail layout template HTML.

Seterusnya @section('content') merujuk kepada persempadanan kandungan yang akan dipaparkan oleh @yield('content') yang terdapat dalam fail layout template *generic.blade.php* .

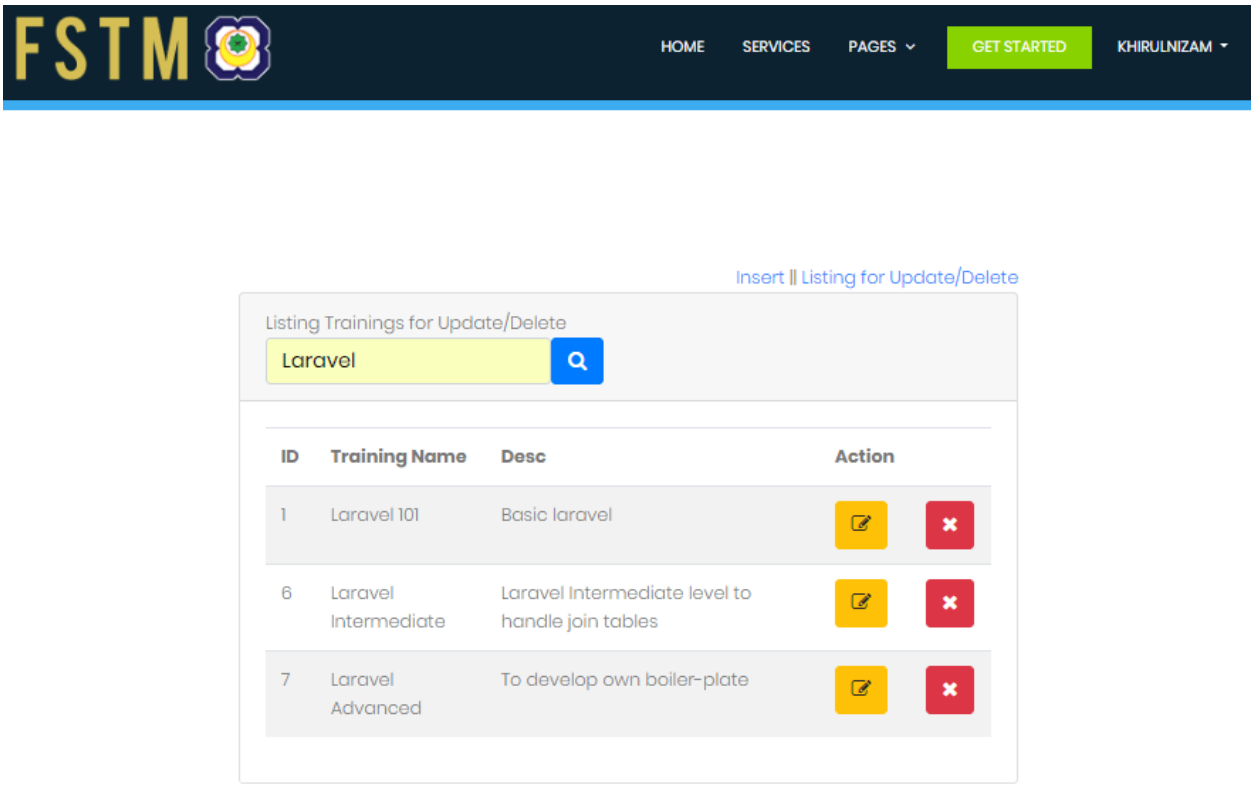
```
@section('content')
...
...
...
@endsection
```

Sila uji dalam browser antaramuka yang telah diubah.



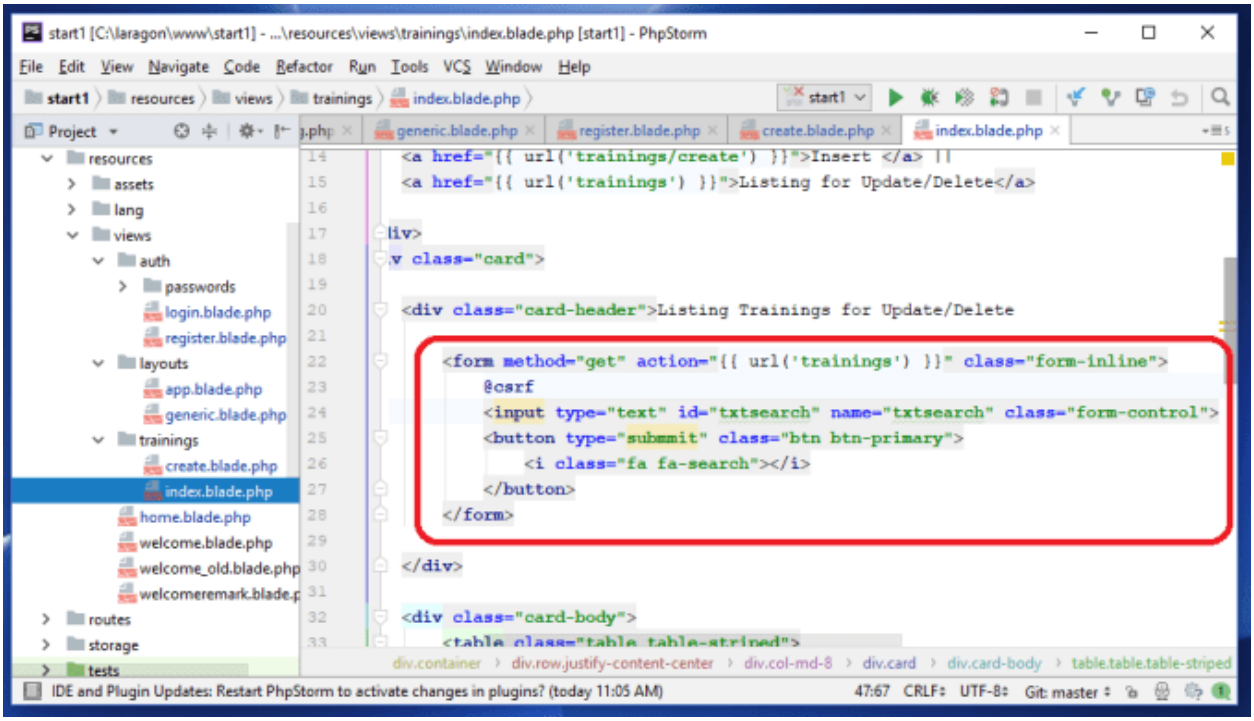
Kod Sumber dalam GITHUB – https://github.com/khirulnizam/start1

Tutorial 04 Carian Rekod



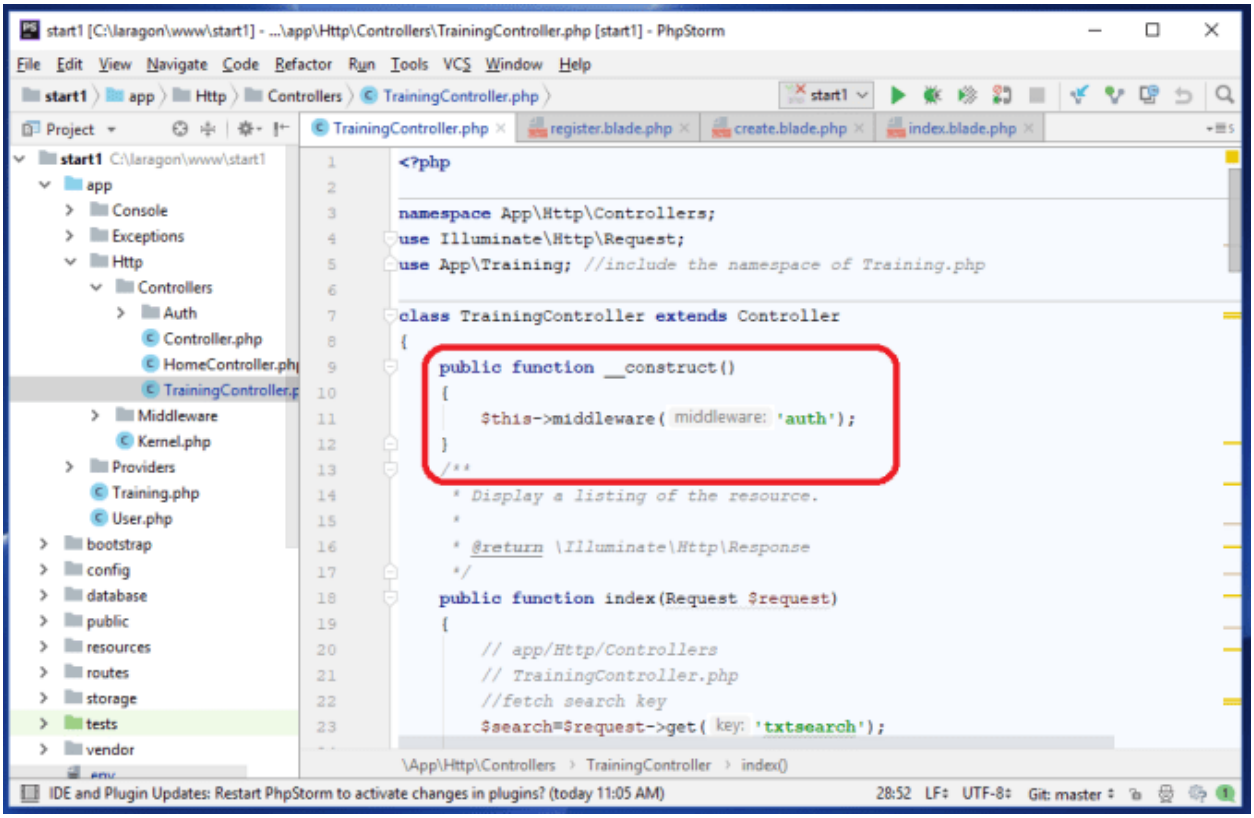
Gambar di atas menujukkan hasil tutorial yang kita akan buat menggunakan Laravel. Tutorial ini bersambung daripada modul Training dan TrainingController yang telah dibincangkan dalam **http://fstm.kuis.edu.my/blog/laravel2/** .

Kita akan sambung mengedit kod antaramuka dalam *resources/views/trainings/* ***index.blade.php*** untuk menambahkan kotak dan butang carian. Perhatikan pada kod FORM action , url('training') menunjuk kepada fungsi dalam *TrainingController* yang akan menerima arahan dan data carian tersebut.



app/Http/Controllers/TrainingsController.php

Kemudian edit kod logik proses dalam app/Http/Controllers/TrainingsController.php, tambahkan kod pengesahan pengguna di bahagian atas class TrainingController. Ini sepatutnya dilakukan pada peringkat awal lagi. Kod di bawah merupakan pengesahan pengguna, sekiranya pengguna belum login maka antaramuka ini akan terhalang. Sekiranya anda telah menulis kod berikut, maka boleh abaikan langkah ini.

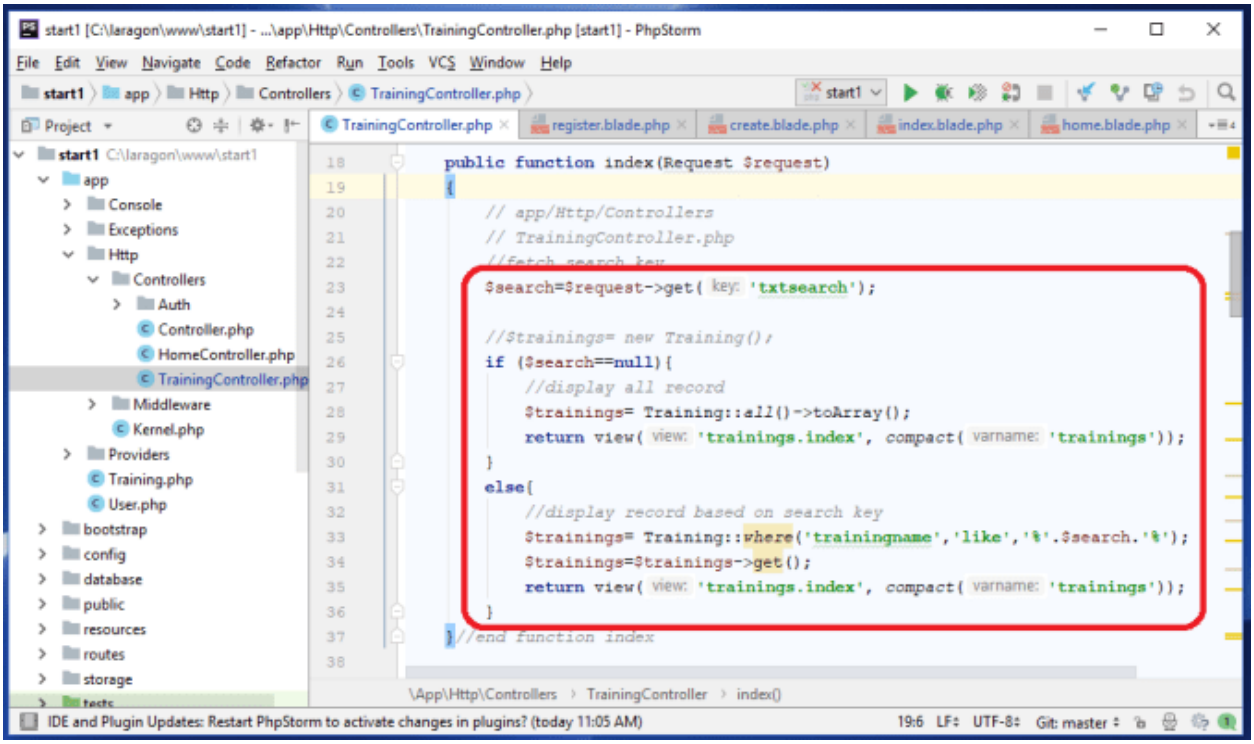


Seterusnya pergi ke function index() , dan tambahkan kod berikut untuk membuat carian.

txtsearch merupakan nama/id kotak teks yang kita telah buat di atas (dalam bahagian *form*).

Segmen kod berikut akan melakukan penyenaraian biasa sahaja.

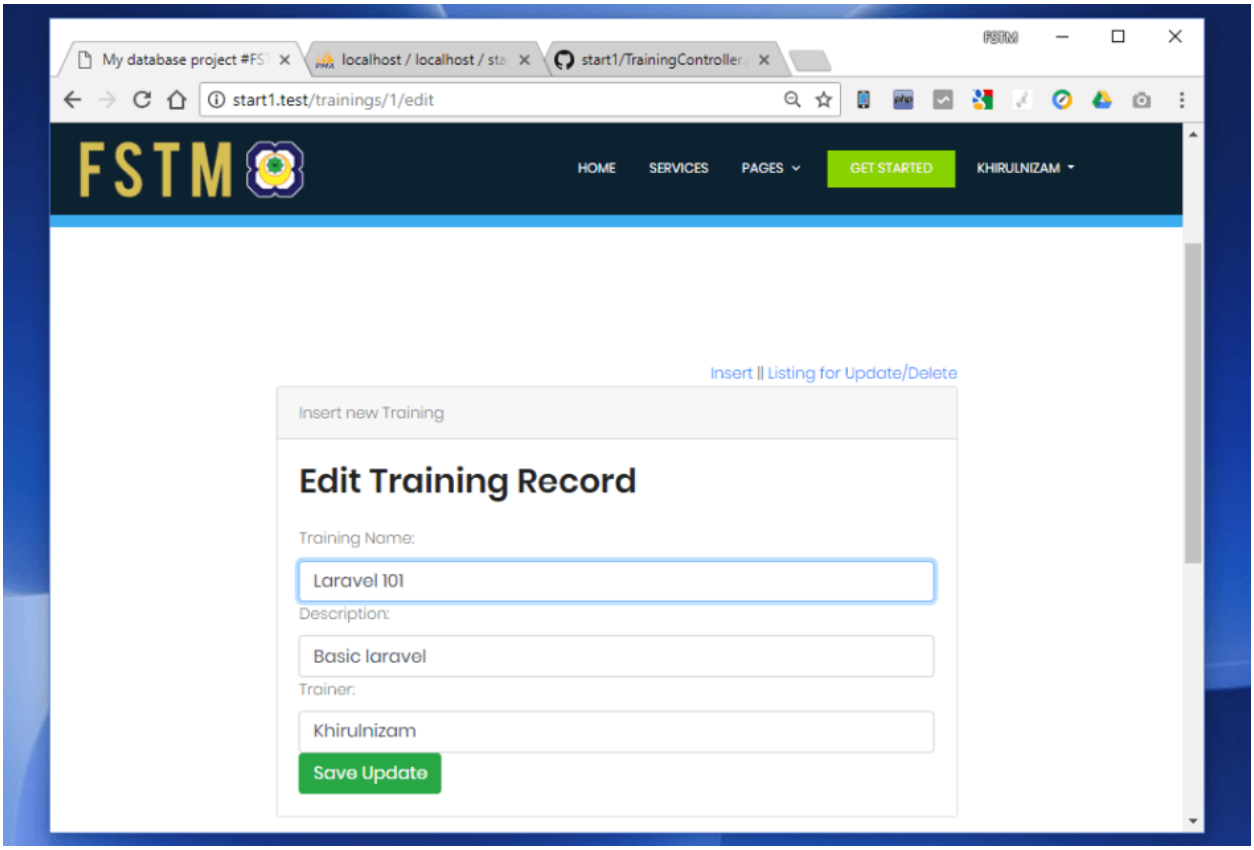
Manakala segment kod berikut akan menerima perkataan carian, dan memaparkan hasil carian yang berpadanan.



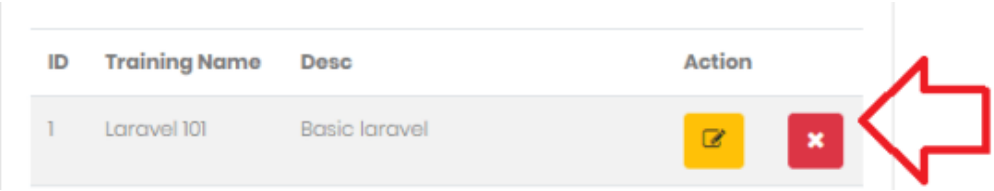
Butang edit/padam

Untuk menghasilkan butang edit dan pada, kita perlu mengubahsuai kod memaparkan rekod, mohon rujuk tutorial seterusnya dalam Tutotial 05.

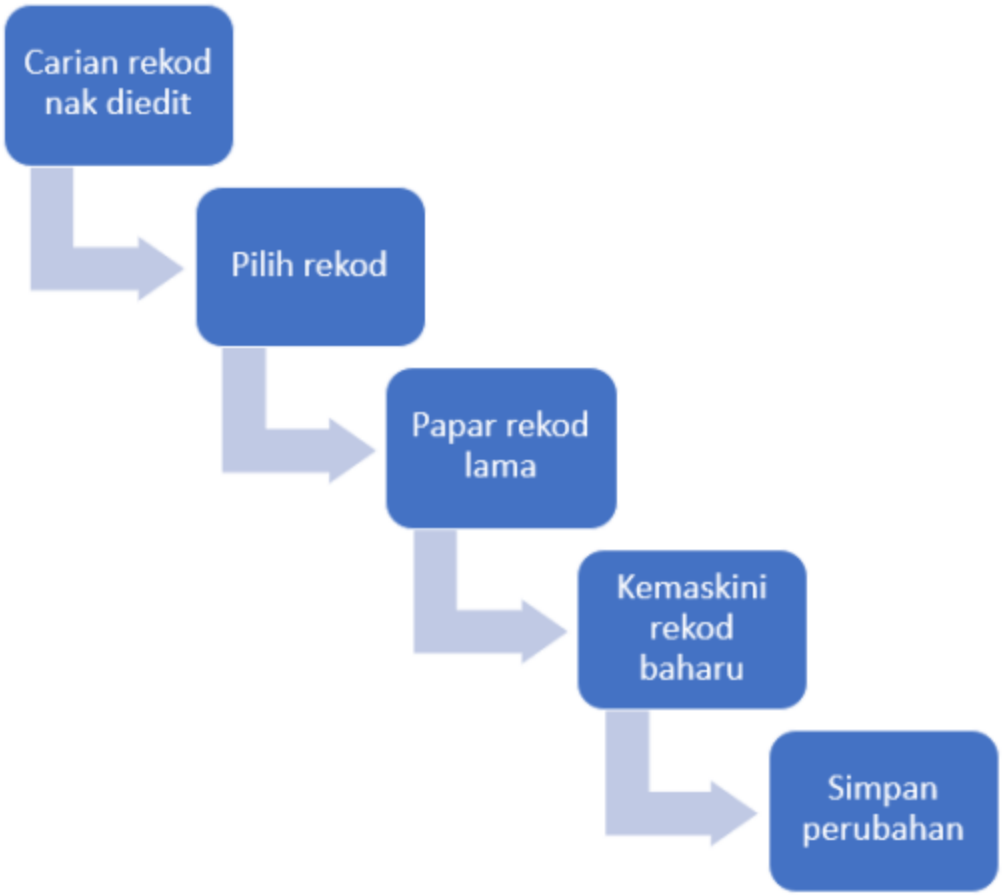
Tutorial 05 Kemaskini dan Padam Rekod



Bersambung dari asas Laravel tutorial ke 4; Carian rekod dalam Laravel . Dalam tutorial sebelum ini, kita terhenti pada penjanaan butang edit dan padam rekod seperti rajah di bawah.



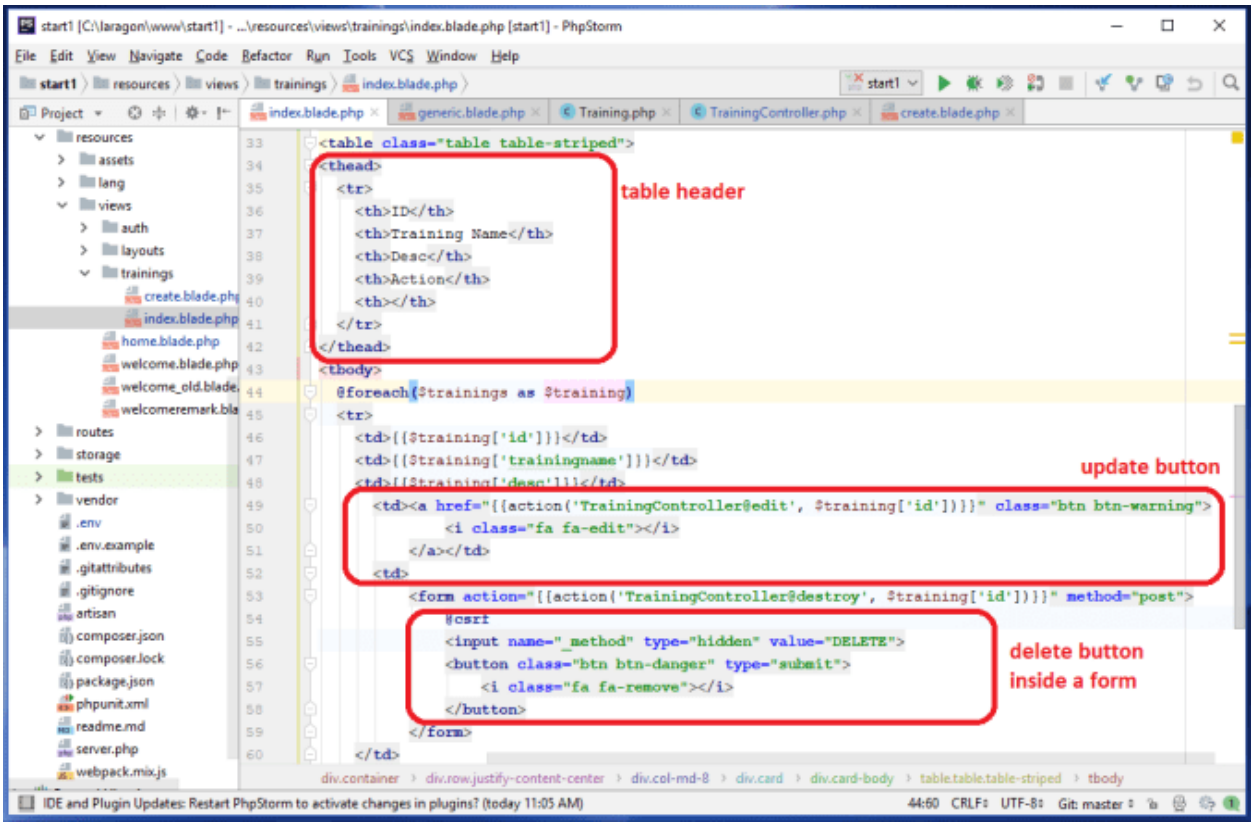
Untuk menghasilkan butang edit dan padam, kita perlu mengubahsuai kod memaparkan rekod dalam fail *resources/views/trainings/* ***index.blade.php*** . Secara logiknya, pengguna akan mencari rekod yang hendak dikemaskini, kemudian paparan data lama dalam borang, maklumat lama diedit, dan simpan maklumat baharu.



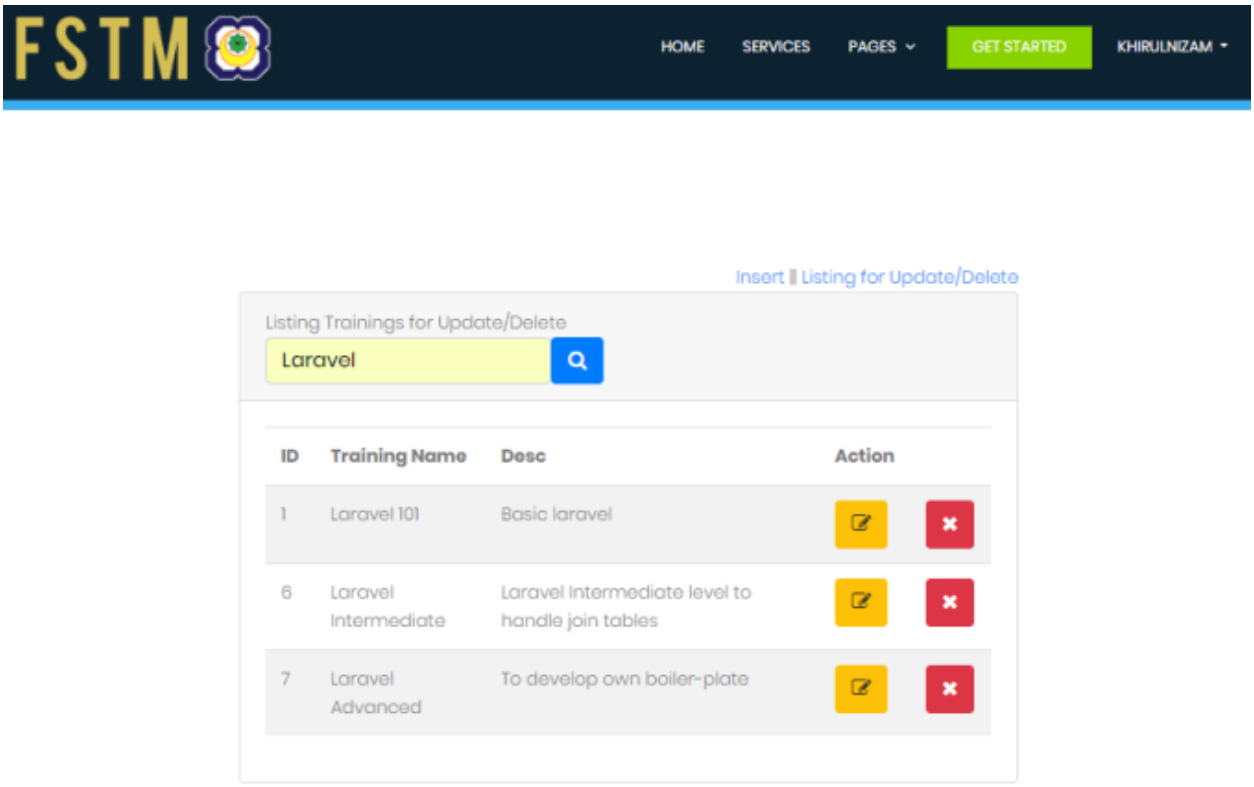
Antaramuka borang papar data lama

Jadi sekarang ini kita hendak sediakan butang untuk pengguna pilih mana rekod yang nak dikemaskini. Kod seperti di bawah boleh diperolehi terus dari GitHub – https://github.com/khirulnizam/start1/blob/master/resources/views/trainings/index.blade.php

Butang kemaskini dan padam rekod berada pada hujung paparan rekod, rekod berada dalam jadual/baris (table row).



Kod tadi akan menghasilkan paparan seperti di bawah.

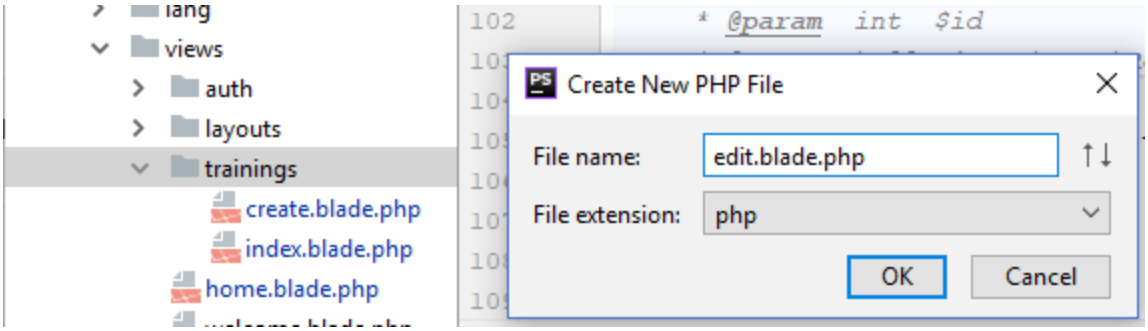


Pengguna akan klik pada butang edit (jingga) untuk memilih rekod mana yang mahu dikemaskini. Apa yang berlaku apabila pengguna klik butang edit diprogramkan dalam fail TrainingController.php. Buka fail tersebut dalam app/Http/Controller, fokus kepada `function edit()` .

(GitHub – https://github.com/khirulnizam/start1/blob/master/app/Http/Controllers/TrainingController.php).

Arahan `return view('trainings/edit')` akan membuka borang kemaskini dalam *resources/views/trainings/* ***edit.blade.php*** . Jadi anda perlu sediakan borang yang boleh kemaskini rekod *Training* .

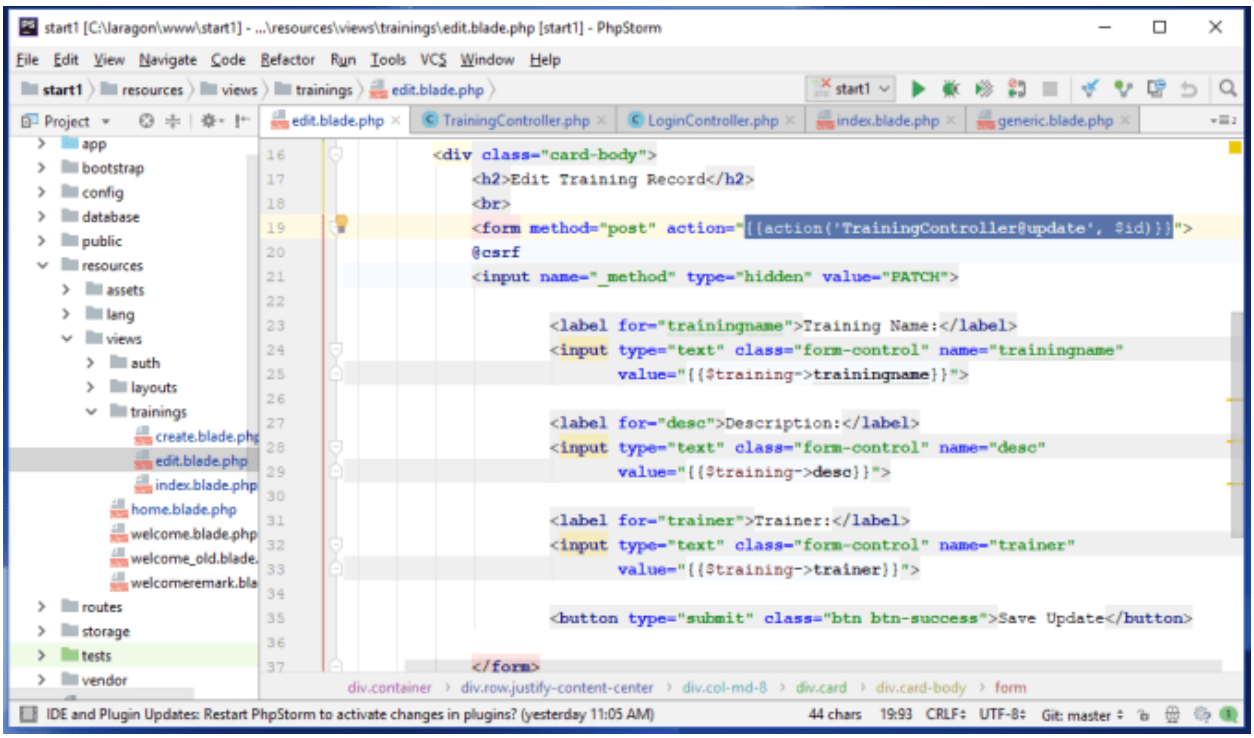
Untuk membuat borang HTML Form ini, klik-kanan pada forlder *resources/views/trainings* dan pilih *Create New PHP file* dari menu.



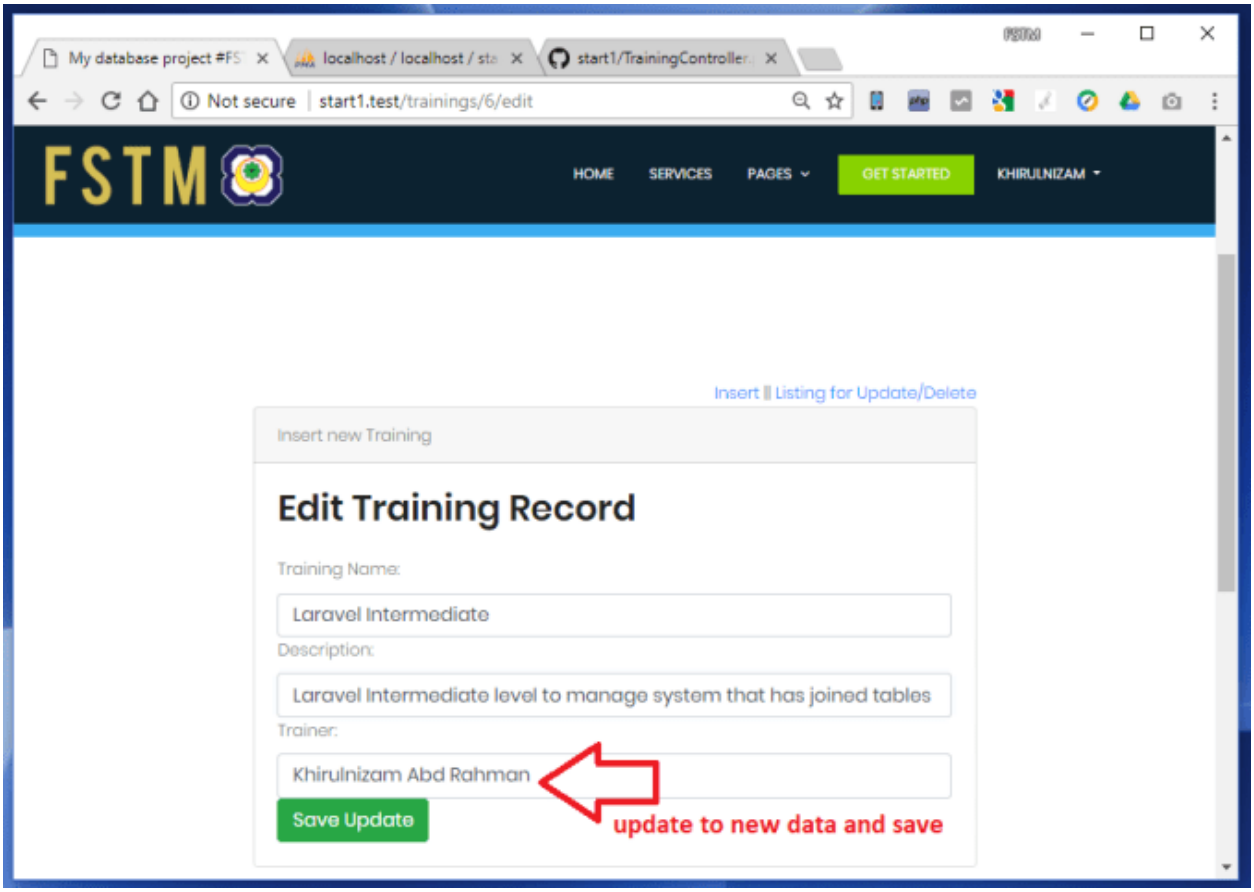
Apa yang penting adalah borang untuk paparkan data lama dan boleh diedit. Lihat pada kod berikut;

<input type=”text” class=”form-control” name=”trainingname” value=” {{$training->trainingname}} “>

Itu adalah arahan memaparkan data lama dalam value textbox.



Output kod di atas dari antaramuka *resources/views/trainings/* ***edit.blade.php***

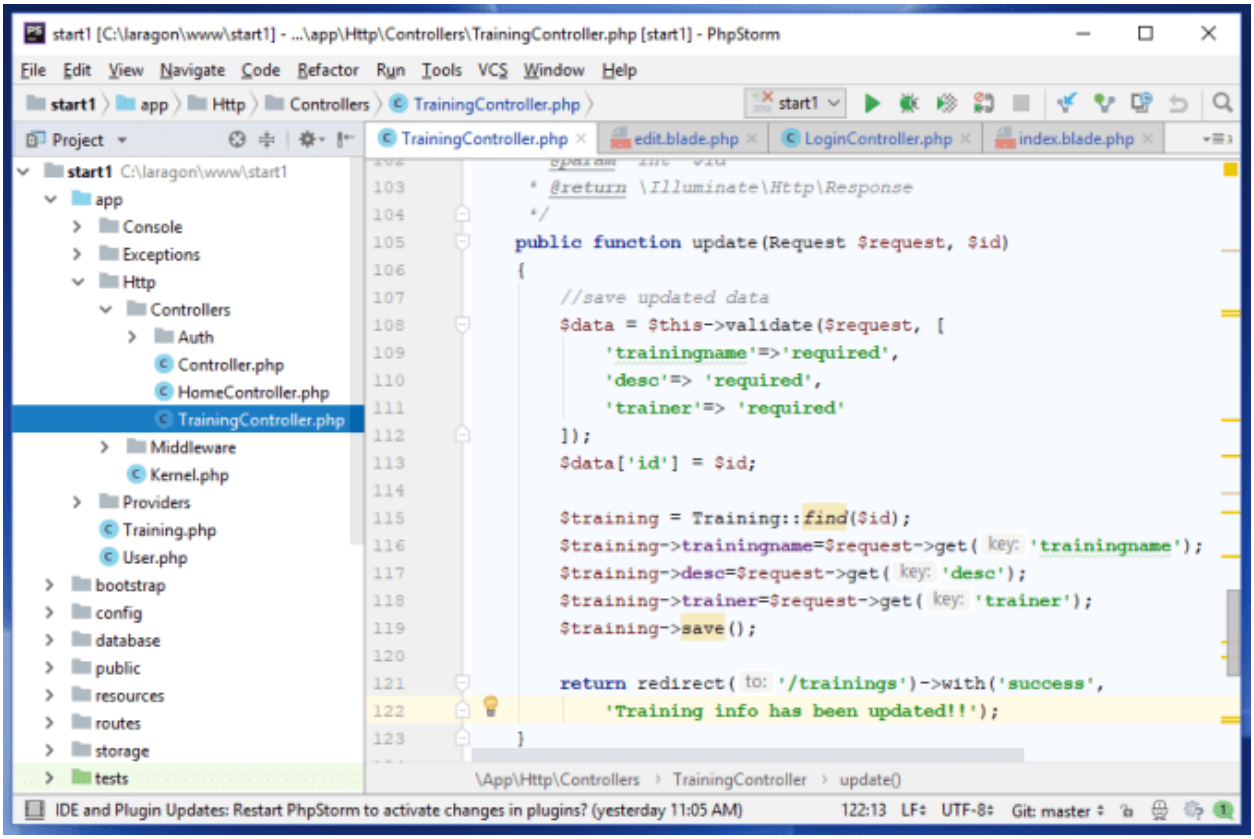


Ke mana data dari form ini akan dihantar? Perhatikan pada baris kod di bawah , data akan dihantar ke function update() dalam fail *TrainingController* dengan membawa id rekod.

```
<form method="post" action="{{action('TrainingController@update', $id)}}">
```

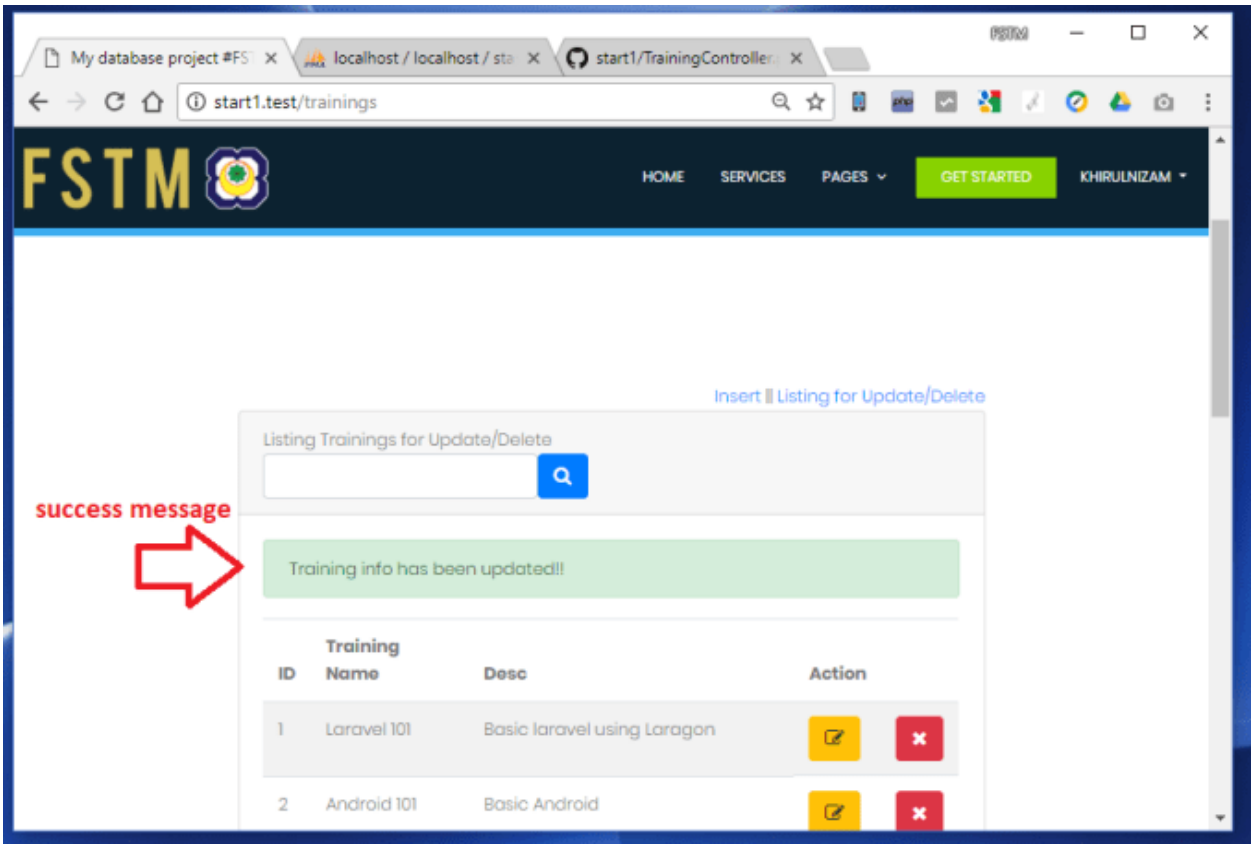
Proses menyimpan data dikemaskini

Seterusnya rujuk fail TrainingController.php (dalam *app/Http/Controllers/*), di bahagian *function update( )* . Tambah baris kod berikut;



Kod lengkap dalam GITHUB – https://github.com/khirulnizam/start1/blob/master/app/Http/Controllers/TrainingController.php

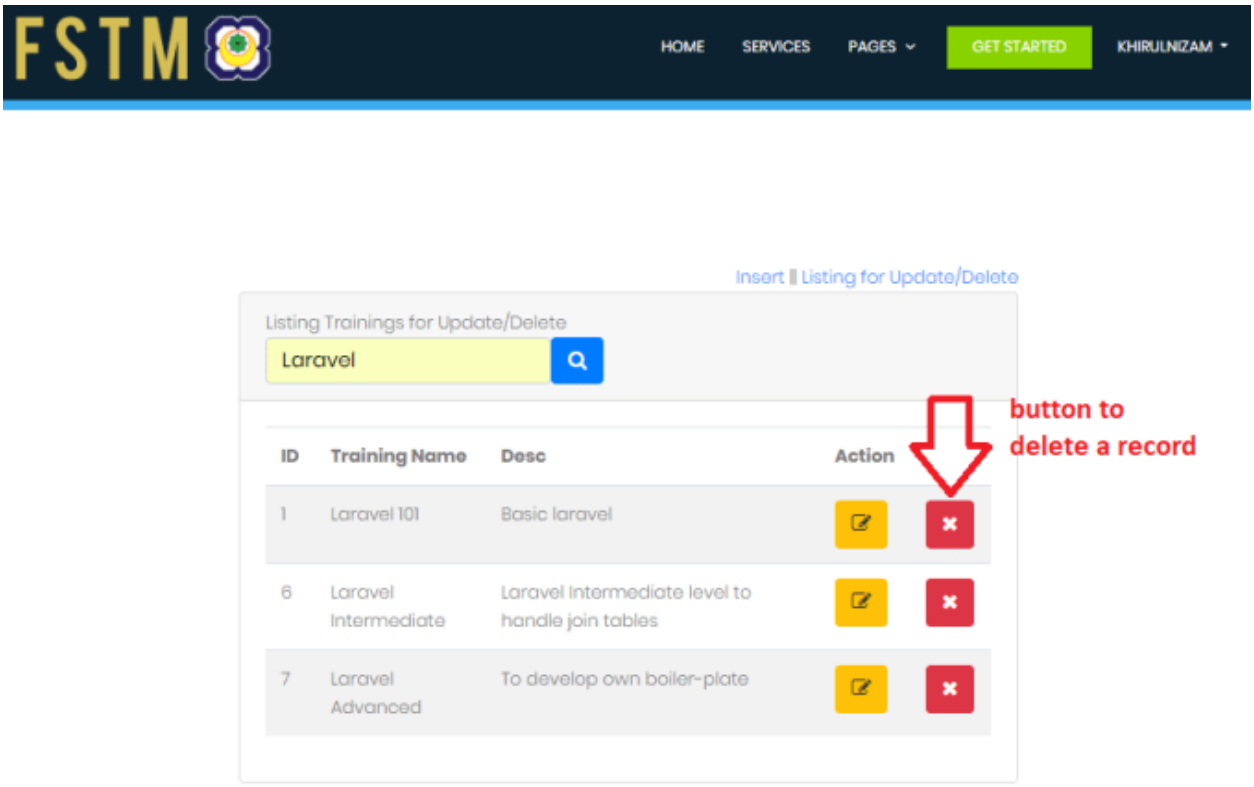
Selepas data baru disimpan, sistem akan hantar paparan ke trainings/index.blade.php dan mesej berjaya dipaparkan.



Bagaimana memaparkan mesej kejayaan tersebut, tambahkan ***div*** berikut pada trainings/index.blade.php (kod lengkap dalam GITHUB).

Padam satu rekod

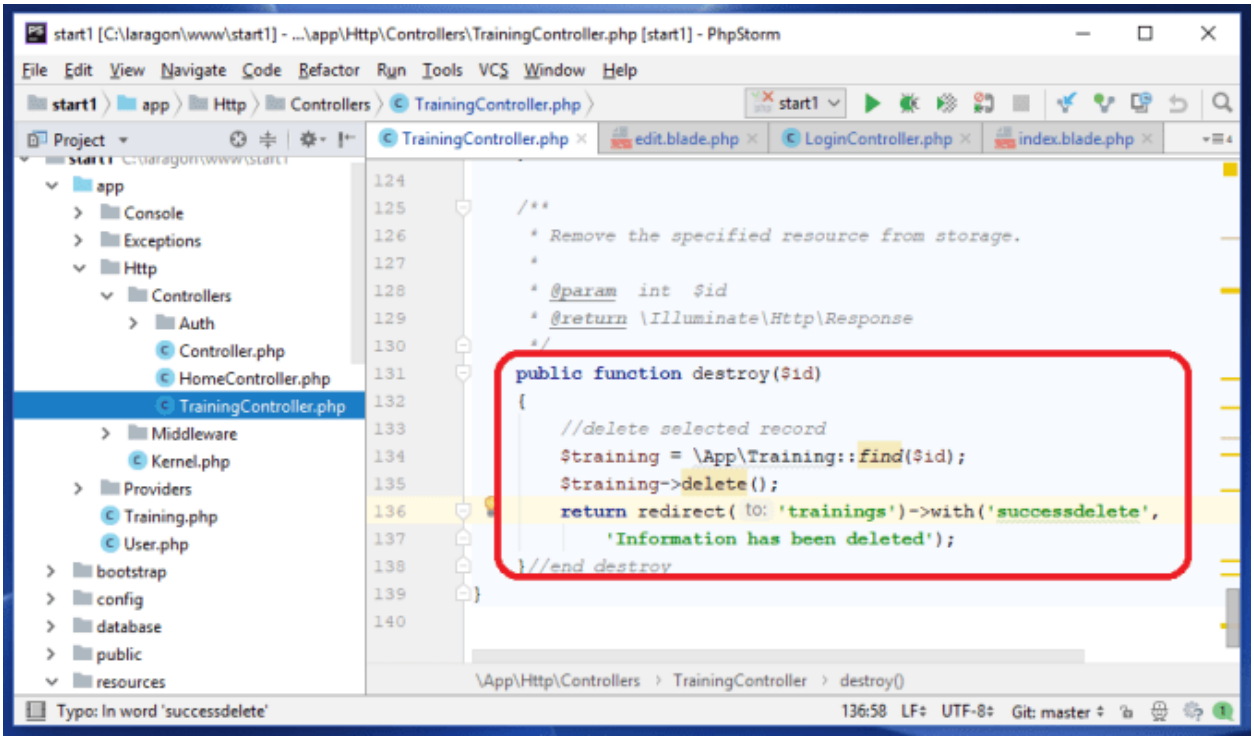
Terakhir untuk tutorial ini pilihan memadam rekod. Butang padam rekod berwarna merah berikut telah disediakan. Bersama borang tersebut ada borang (HTML form) untuk menghantar data dalam ***method=post*** .



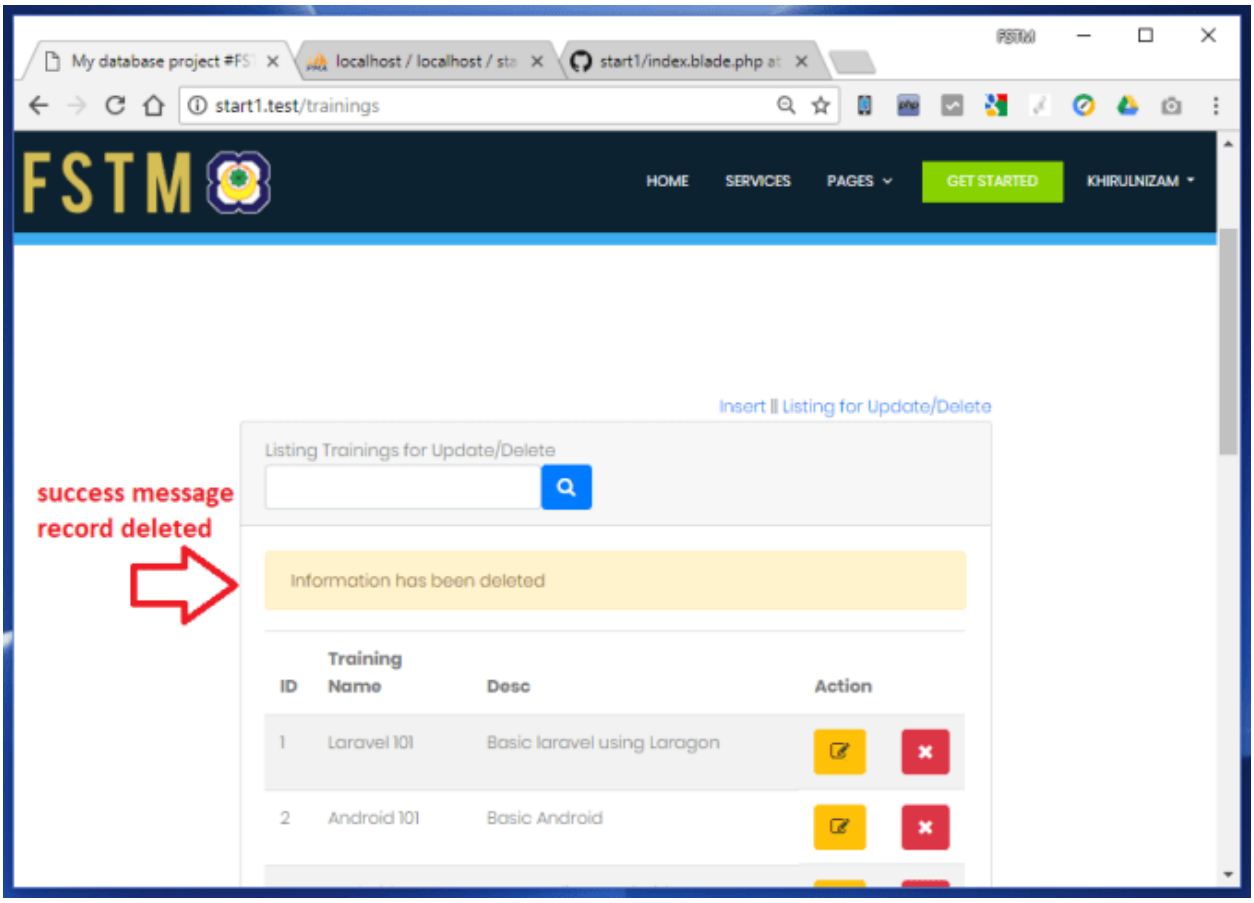
Kita perhati kod HTML untuk hasilkan butang padam berikut;

Pada *form action* sistem akan menghantar data id rekod yang pengguna pilih ke TrainingController/function destroy() method post adalah yang ditetapkan dalam route:list.

Rujuk fail app/Http/Controllers/TrainingController.php untuk menambah kod seterusnya dalam function destroy( ).



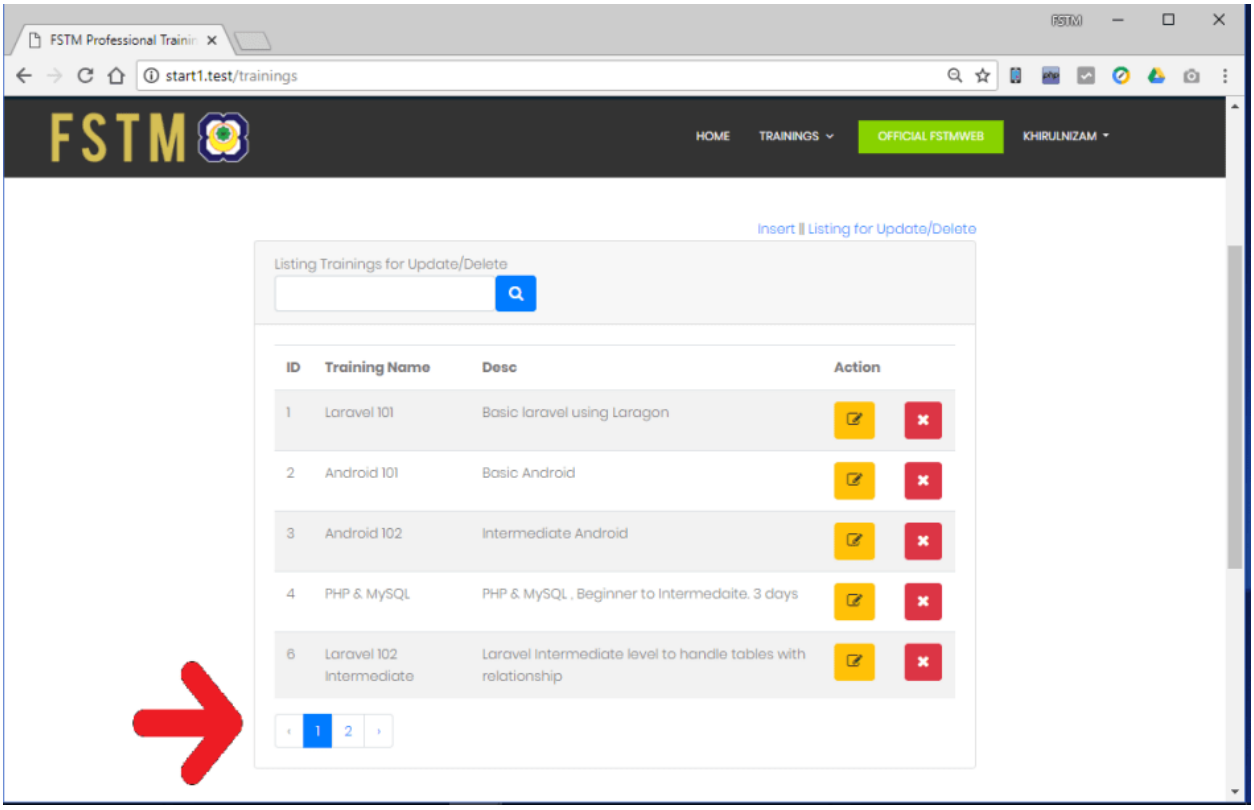
Paparan output selepas padam rekod.



Bagaimana memaparkan mesej kejayaan tersebut, tambahkan ***div*** berikut pada trainings/index.blade.php (kod lengkap dalam GITHUB).

[Selamat mencuba. Sila LIKE **fb.com/kuis.fstm** untuk menerima update tutorial seterusnya]

Tutorial 06 Paparan carian dengan *Pagination*

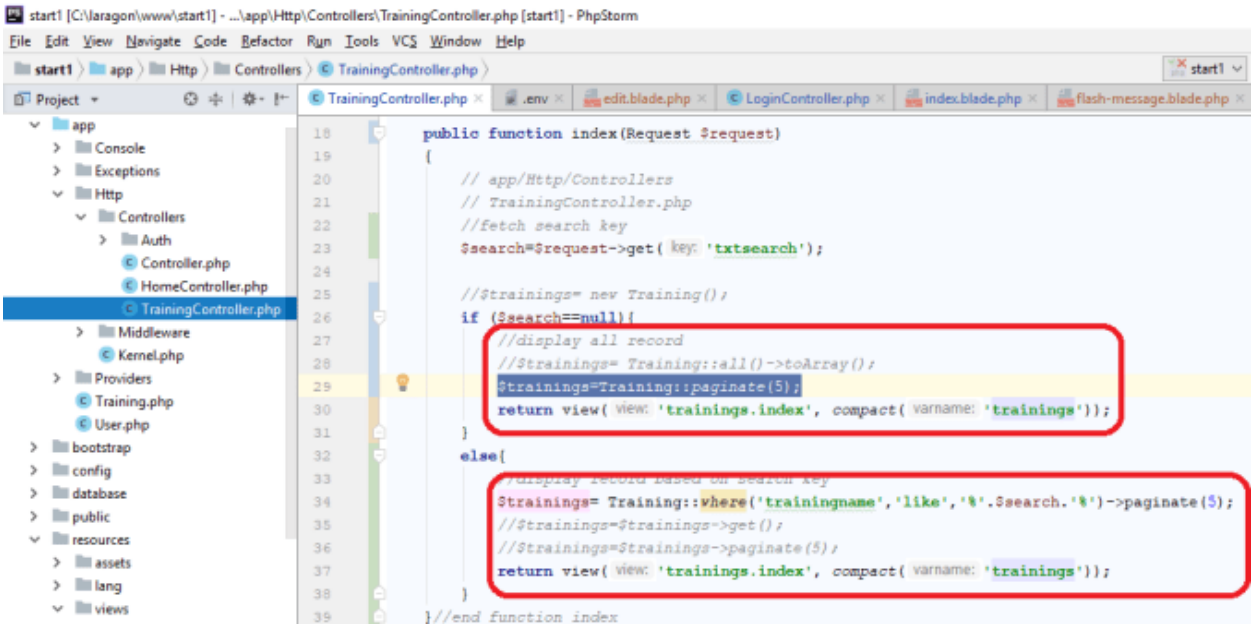


Rekod setakat sepuluh data tak perlu kot *pagination* . Tapi makin lama, data pastinya akan berkembang. Tak patut kita paparkan seribu rekod dalam satu page. *Defeat the purpose of searching!*

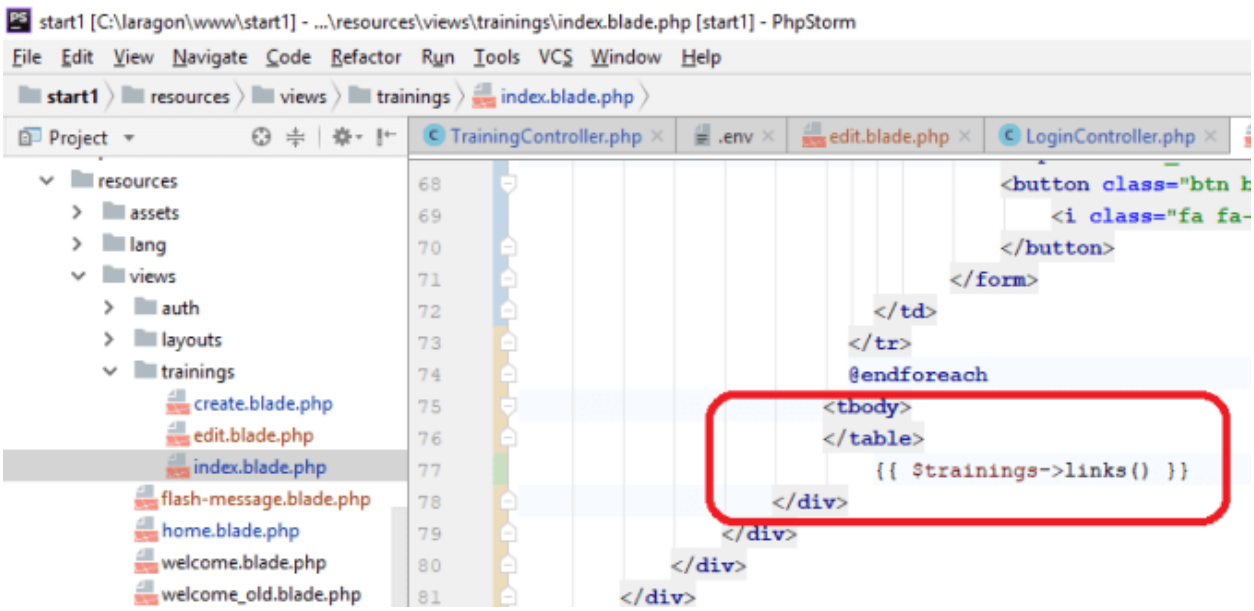
Pagination merupakan kaedah memaparkan sebahagian hasil carian dan pengguna boleh pilih untuk paparkan hasil seterusnya dengan klik pada butang page seterusnya. Lihat contoh seperti ditunjukkan oleh anak panah pada rajah di atas.

Paginate() merupakan tambahan kepada *resultset* kita untuk implementasi kemudahan pagination.

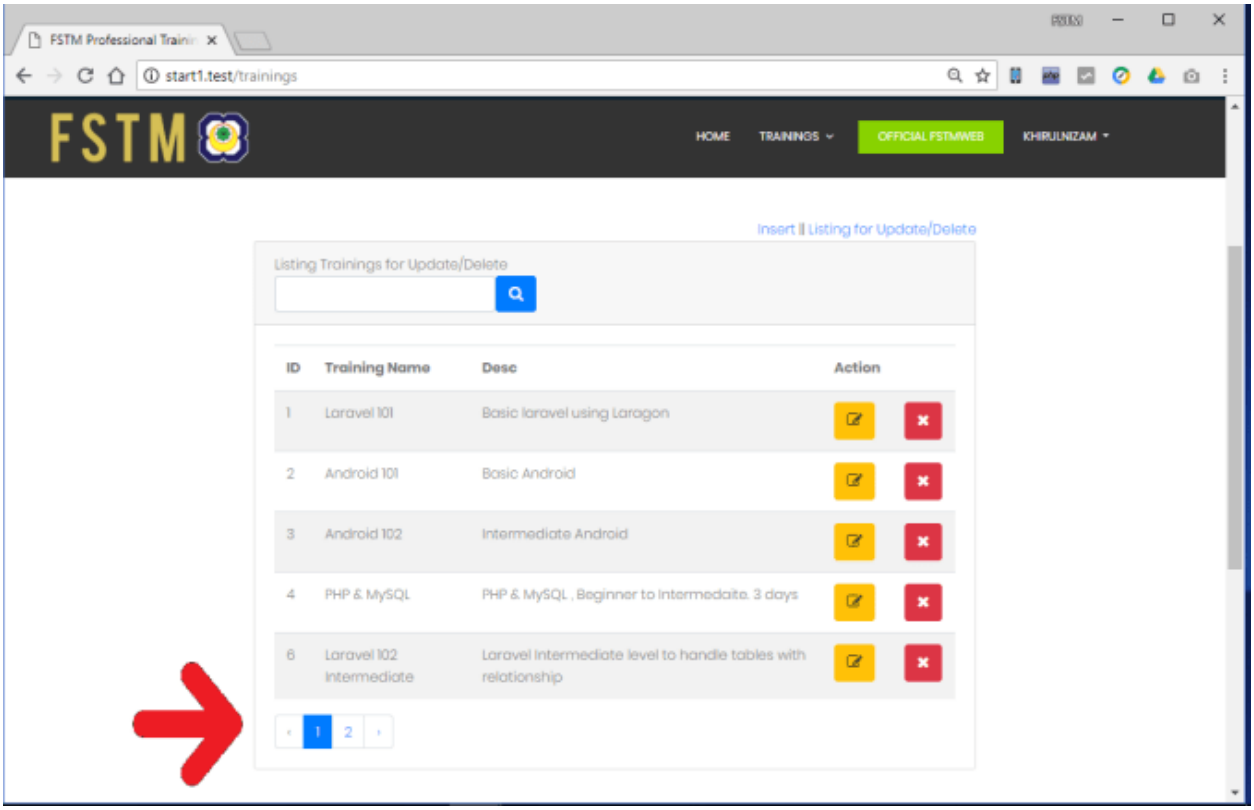
Dalam `function index()` , fail TrainingControllers.php (lokasi app/Http/Controllers/) sila tambah kod seperti rajah di bawah.



Seterusnya, untuk memaparkan resultset pilihan page ke-2, ke-3 dan seterusnya, tambah di bawah jadual paparan data/rekod.

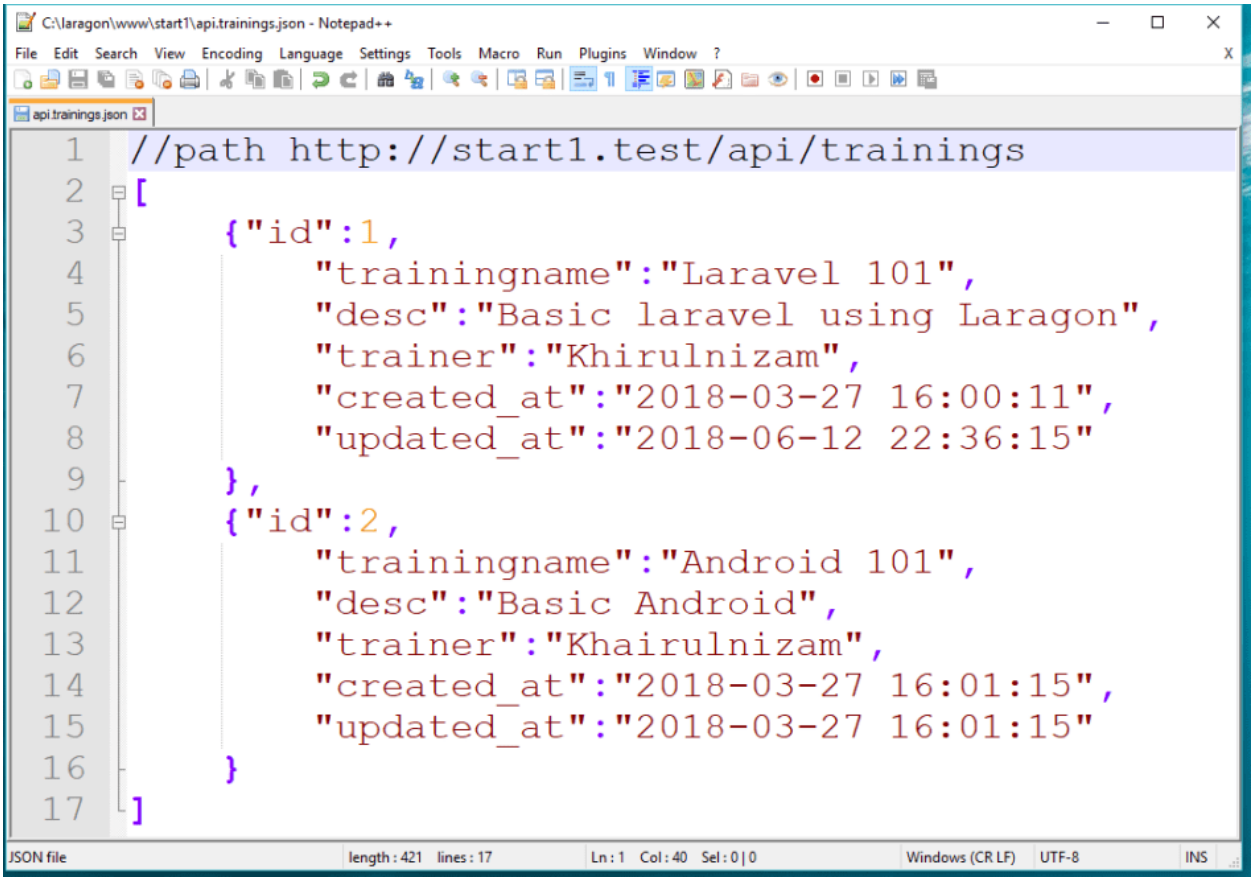


Dan akhirnya kita uji perubahan kod dalam browser, sepatutnya dapat paparan berikut.



[Selamat mencuba. Sila LIKE **fb.com/kuis.fstm** untuk menerima update tutorial seterusnya]

Tutorial 07 Laravel REST API



Tutorial seterusnya akan menyenarai rekod melalui format data JSON.

Penyediaan persekitaran Laravel menggunakan LARAGON mungkin anda perlu lalui dua tutorial ini sekiranya masih baru dalam Laravel –

T1: Kaedah install & guna Laragon (http://fstm.kuis.edu.my/blog/laravel1)

T2: Asas CRUD dalam Laravel (http://fstm.kuis.edu.my/blog/laravel2)

REST API

Dengan keperluan mendesak capaian pangkalan data melalui app mobile dan framework JavaScript, Sistem web pangkalan data sekarang ini perlu menyokong penghantaran data universal. RESTful API merupakan pilihan yang terbaik untuk membenarkan capaian kepada data tanpa perlu berkongsi katalaluan pangkalan data kepada semua orang.

Format penghantaran data yang biasa diguna adalah format JSON (atau XML). Dalam tutorial ini kita akan membincangkan bagaimana Laravel menjana data API dalam format JSON.

Contoh JSON format data

Di akhir tutorial ini kita akan menjana format data JSON seperti dalam gambar di atas.

HTTP Verbs
Dalam RESTful API, kita akan menggunakan HTTP verbs sebagai perlakuan operasi ke atas data.

GET: dapatkan data
POST: masukkan data baharu
PUT: kemaskini data
DELETE: padam data

routes/api.php

Ini merupakan fail yang penting untuk menjana API data. Tambah beberapa baris kod berikut berdasarkan *HTTP verbs* .

```
Use App\Training;//use Trainings model

// trainings API
// Let's create the basic endpoints for our application:
// create, retrieve the list, retrieve a single one,
update, and delete.
// On the routes/api.php file, we can simply do these:

Route::get('trainings', function() {
// If the Content-Type and Accept headers are set to
'application/json',
// this will return a JSON structure. This will be
cleaned up later.
return Training::all();
//this is to list all records
//available from http://start1.test/api/trainings
```

```
});

Route::get('trainings/{id}', function($id) {
return Training::find($id);
//this is to list one specific record by id
//available from http://start1.test/api/trainings/3
});

Route::post('trainings', function(Request $request) {
return Trainings::create($request-&gt;all);
//to insert a new record
});

Route::put('trainings/{id}', function(Request $request,
$id) {
$training = Trainings::findOrFail($id);
$training-&gt;update($request-&gt;all());

return $training;
//to search record
});

Route::delete('trainings/{id}', function($id) {
Training::find($id)-&gt;delete();

return 204;
//to delete one record with specific id
});
```

Dah selesai tambah kod berikut dalam routes/api.php, sila test melalui path yang ditambah /api/

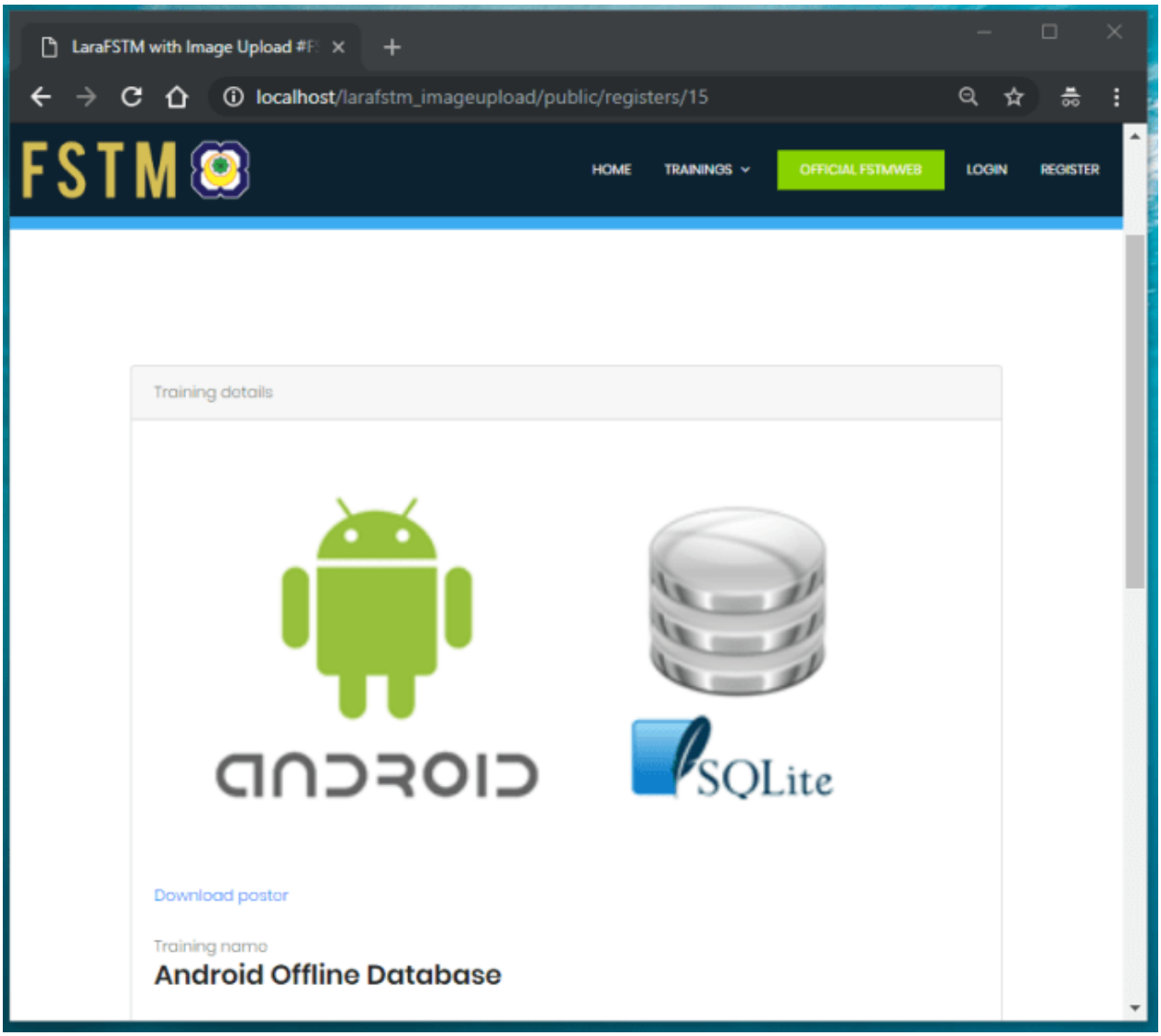
Contoh: *http://start1.test/* ***api*** */trainings*

Tutorial 08 Upload imej ke sistem Laravel

Gigih kawan-kawan kita dari UMP, dari KUIS yang buat projek akhir datang cuti Deepavali sedut ilmu Laravel. Semangat peserta berjangkit ke trainer untuk menggigihkan diri menambah koleksi modul Laravel seterusnya. Macam-macam kisah cabaran programmer2 nak sampai ke KUIS – kami kagum dengan anda semua. Kali ini nak sambung dengan cara-cara upload imej dalam sistem web berasaskan Laravel.

Cuma sebelum itu admin harap tuan-puan dah pernah buat tutorial yang terdahulu sebelum harung tutorial ini. Kalau belum sila ke Tutorial CRUD Pengenalan Laravel

Kod lengkap tutorial ini ada di
github.com/khirulnizam/larafstm_imageupload



Contoh page yang memaparkan imej yang telah dimuat-naik ke server

Kerangka tutorial ini;

Menambahkan medan dalam jadual (table) trainings untuk menyimpan nama fail imej yang dimuat-naik ke server.

Mengubahsuai setting storan dalam *config/filesystem.php*

Menyediakan link (virtual folder dalam public) yang akan mencapai fail imej dalam storan.

Menyediakan borang *(form)* memilih dan memuat-naik fail imej.

Menambah proses simpan fail imej dalam TrainingController.php function store().

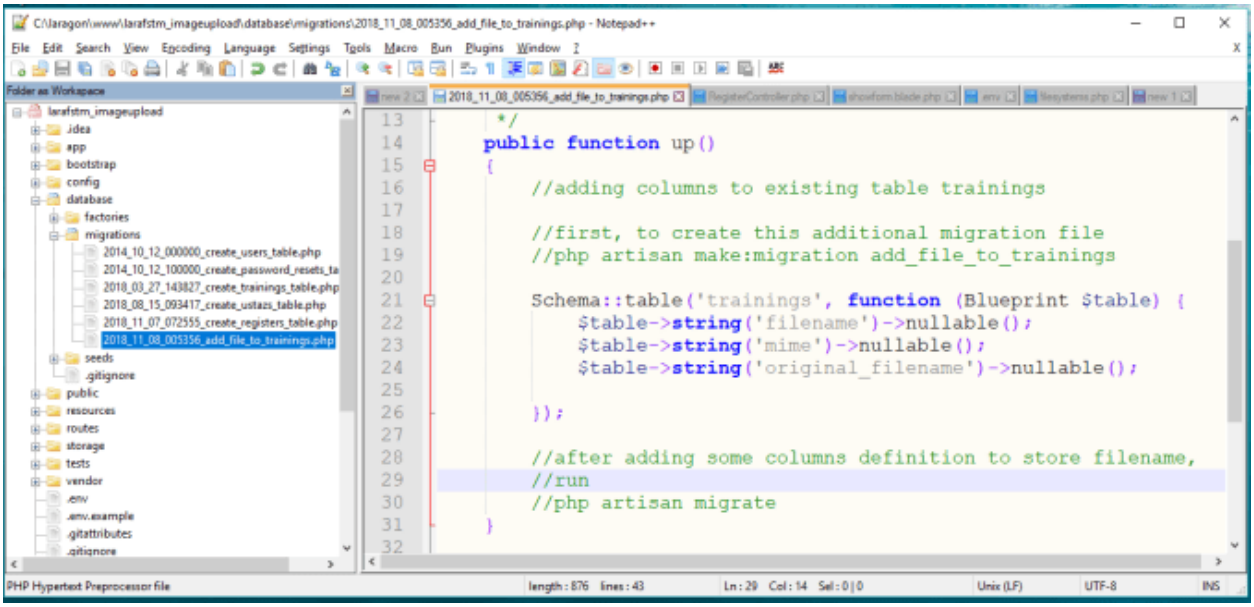
Membuat paparan imej yang telah dimuat-naik.

Menambahkan medan dalam jadual (table) *trainings*

Medan tambahan ini diperlukan untuk menyimpan nama fail imej yang dimuat-naik ke server. Kita telah menyediakan jadual ***trainings*** dalam tutorial lepas. Pengubah-suaian struktur jadual dalam projek Laravel mesti dilakukan dalam *migration file* . Buat satu migration-file tambahan dengan arahan berikut;

php artisan make:migration add_file_to_trainings

Ini akan menghasilkan migration-file dalam folder database/migrations. Masukkan kod berikut untuk menambah lajur pada table ***trainings*** yang sedia ada.



Klik untuk besarkan kod menambah lajur

Medan filename menyimpan nama fail imej, mime menyimpan jenis file , original_filename menyimpan nama fail imej yang asal.

```
Schema::table( 'trainings' , function (Blueprint $table )
{
$table ->string( 'filename' )->nullable();
$table ->string( 'mime' )->nullable();
```

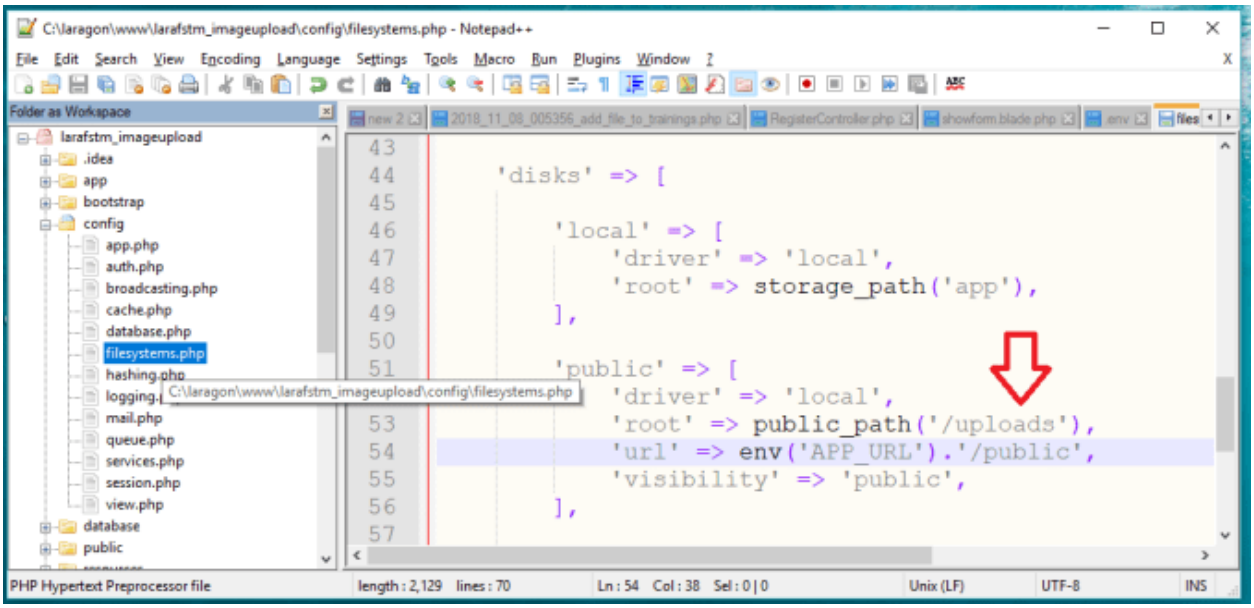
```
$table ->string( 'original_filename' )->nullable();

});
```

Seterusnya laksanakan migration untuk menambah medan-medan (field) tambahan ke pangkalan data;

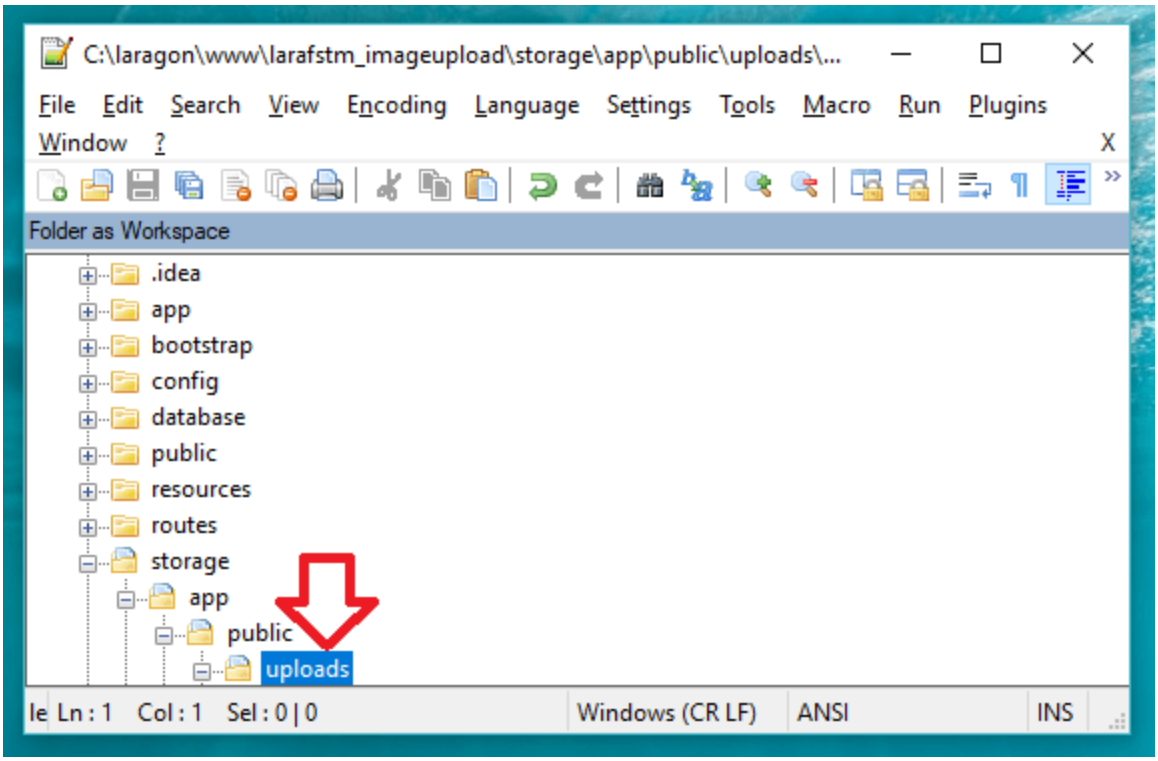
php artisan migrate

Mengubahsuai setting storan dalam *config/filesystem.php*

Sekiranya ada keperluan ubah setting dalam fail ini. Settings ini merujuk kepada di mana fail imej yang user muat-naik akan disimpan.



Dalam tutorial ini kita akan menambah folder *uploads* kepada folder ***storage/app/public*** . Jadi local folder untuk simpan fail imej berada dalam ***storage/app/public/uploads*** . Secara manual cipta folder *uploads* dalam lokasi yang ditetapkan.



Lokasi folder uploads tempat simpan fail imej yang user muat-naik

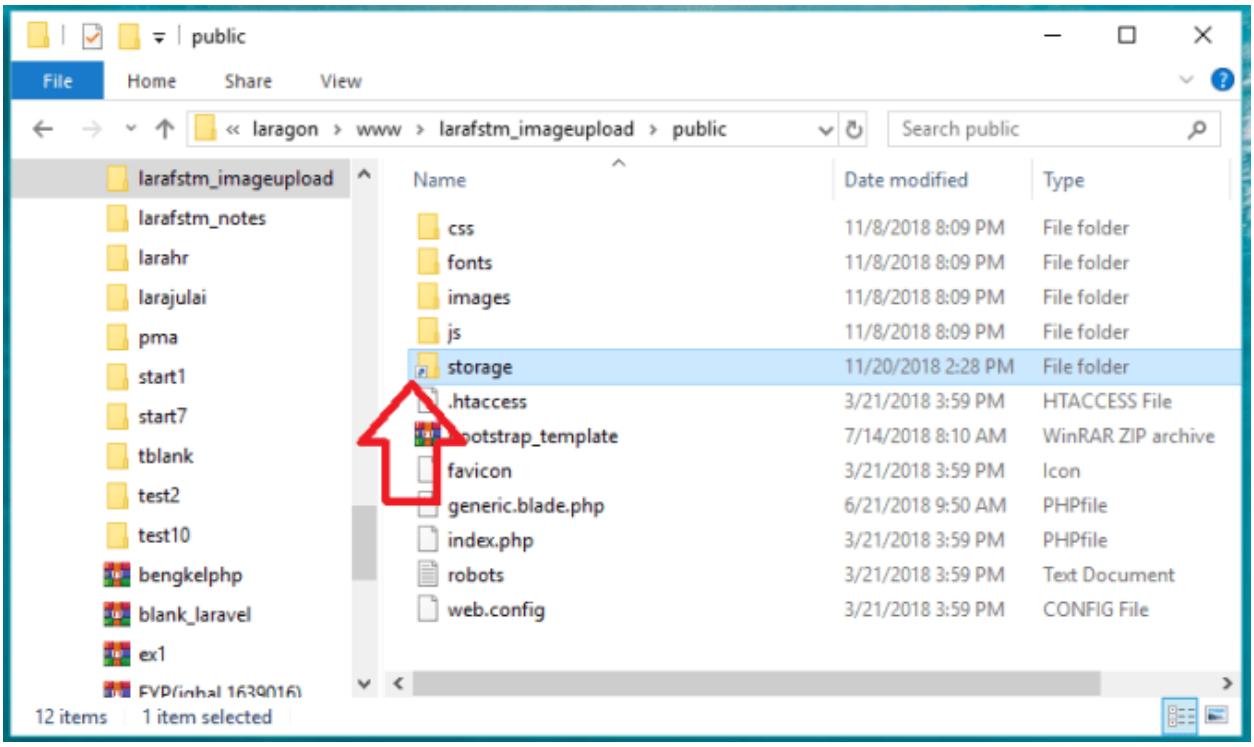
Menyediakan link (virtual folder dalam public)

Proses seterunya yang akan memberikan capaian public kepada fail imej dalam ***storage/app/public/uploads*** . Proses ini perlu dilakukan sebab storan fail imej yang dimuat-naik user berada dalam folder bukan-public, makanya *virtual folder* (link) perlu diwujudkan supaya imej boleh dipaparkan pada paparan public.

Arahan berikut akan menghasilkan virtual link tersebut;

php artisan storage:link

Hasil daripada arahan php artisan storage:link akan menghasilkan virtual-link yang akan membolehkan storan fail-fail image melalui folder *public* projek Laravel. Perhatikan pada icon folder terdapat anak panah kecil biru menandakan ia bukan folder fizikal, cuma sekadar virtual link.



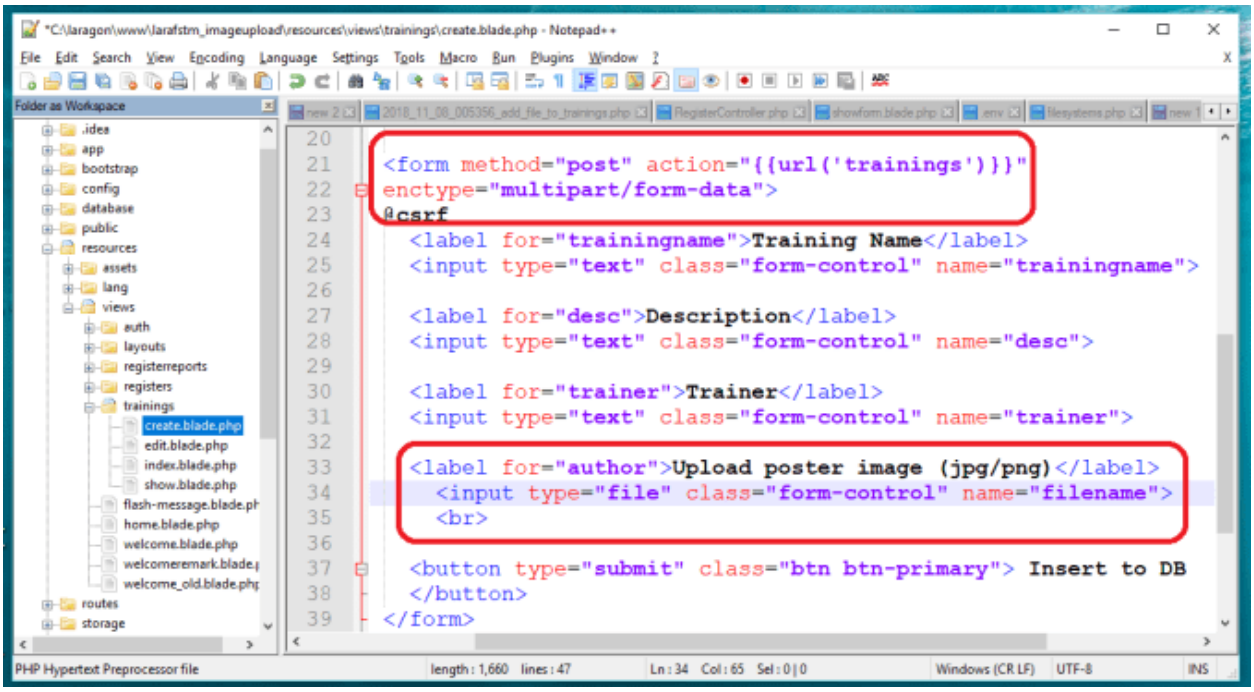
Virtual-link folder storage untuk simpan fail-fail imej dimuat-naik pengguna

Menyediakan borang *(form)* memilih dan memuat-naik fail imej.

Sedikit perubahan pada borang cipta rekod perlu ditambah untuk membenarkan pengguna memilih fail imej dan memuatnaik ke server. Jadi buka fail blade di resources/views/trainings/create.blade.php, dan tambah input-control file-upload.

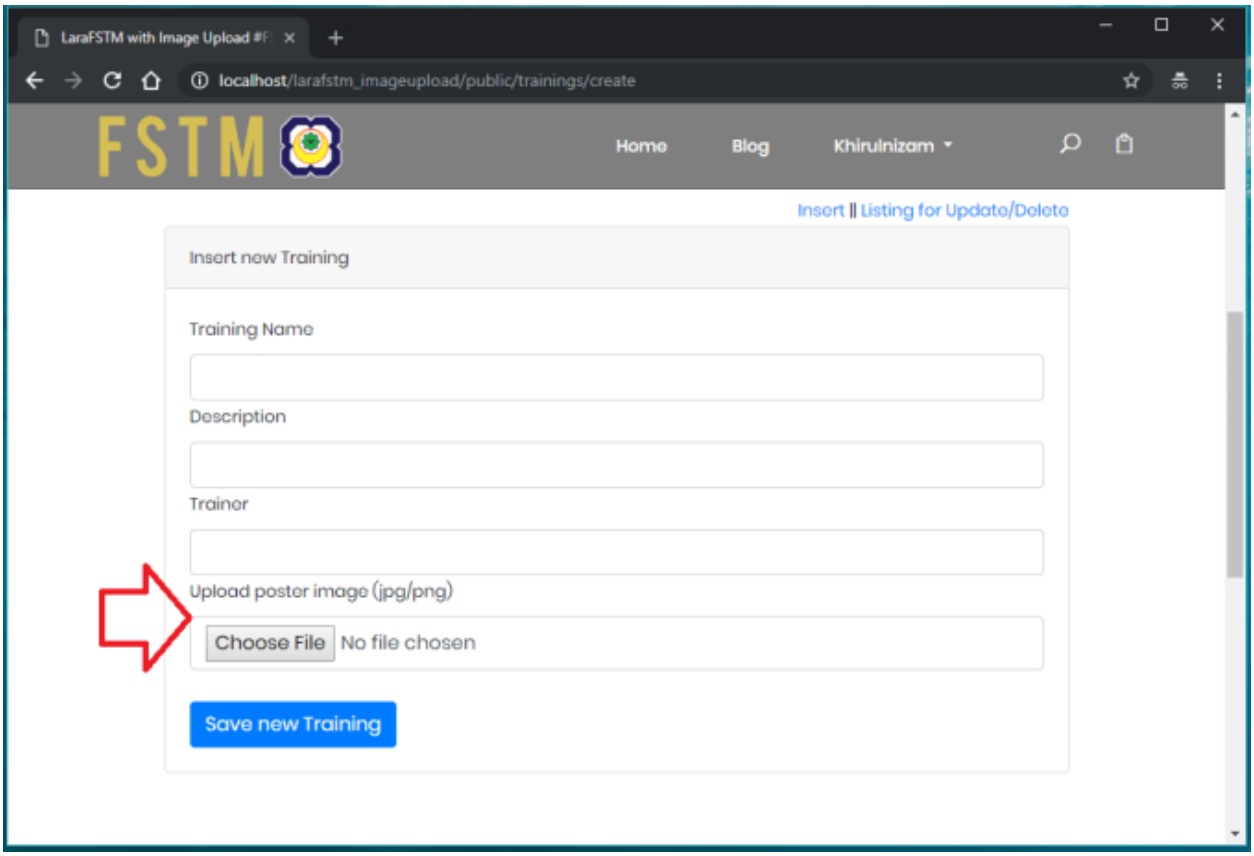
Tambah juga pada form atribut berikut;

```
enctype="multipart/form-data"
```



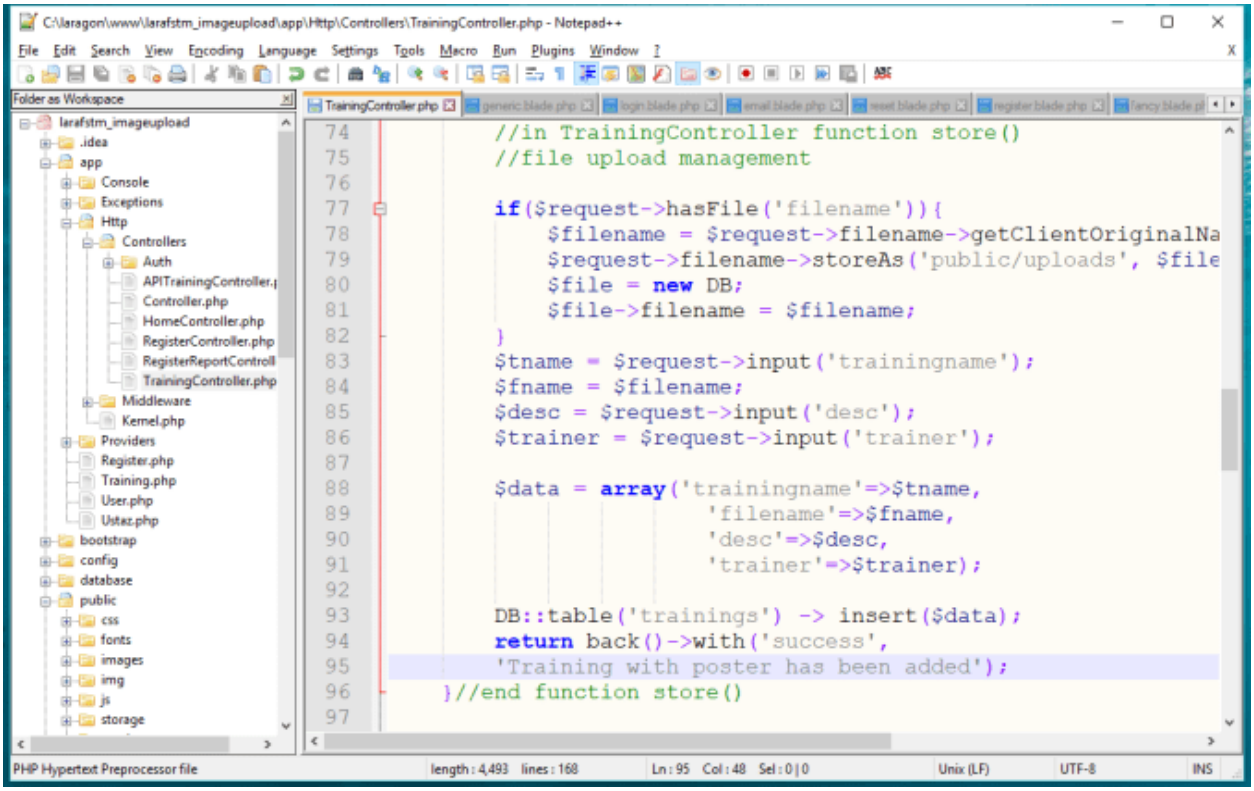
Tambahan enctype dan file-upload input control

Borang yang terhasil dengan muat-naik fail.



Menambah proses simpan fail imej

Seterusnya dalam TrainingController.php function store(), tambah kod untuk simpan fail imej yang user muat-naik ke server.



Fungsi store() dalam TrainingController untuk simpan fail imej dimuat-naik

```
//// TrainingController.php
public function store(Request $request )
{
//validate insert new record process is here
$training = $this ->validate(request(), [
'trainingname' => 'required' ,
'desc' => 'required' ,
'trainer' => 'required'

]);
```

```
//in TrainingController function store()
//file upload management

if ( $request ->hasFile( 'filename' )){
$filename = $request ->filename->getClientOriginalName();
$request ->filename->storeAs( 'public/uploads' ,
$filename );
$file = new DB;
$file ->filename = $filename ;
}
$tname = $request ->input( 'trainingname' );
$fname = $filename ;
$desc = $request ->input( 'desc' );
$trainer = $request ->input( 'trainer' );

$data = array ( 'trainingname' => $tname ,
'filename' => $fname ,
'desc' => $desc ,
'trainer' => $trainer );

DB::table( 'trainings' ) -> insert( $data );
return back()->with( 'success' ,
'Training with poster has been added' );
} //end function store()
```

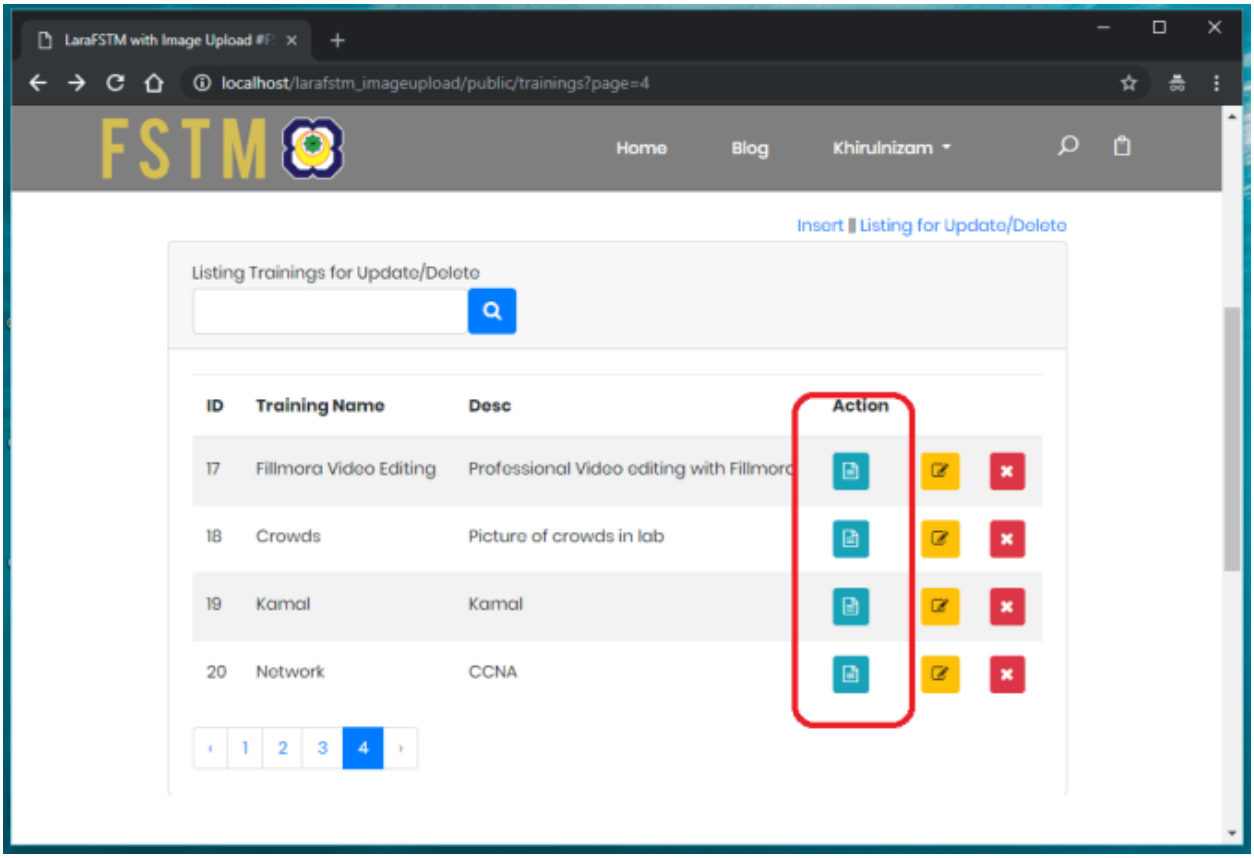
Kod penuh di GITHUB
https://github.com/khirulnizam/larafstm_imageupload/blob/master/app/Http/Controllers/TrainingController.php

Paparan imej yang telah dimuat-naik

Page seterusnya akan memilih rekod yang telah disimpan bersama fail imej – show().

Tambahan butang show-all boleh dibuat dengan menambah lajur pada senarai rekod seperti di bawah. Untuk menambahkan lajur dan butang-butang tersebut gunakan kod berikut;

```
<td>
<a href=
"{{action('TrainingController@show',$training['id'])}}"
class = "btn btn-info btn-sm" >
<i class = "fa fa-file-text-o" ></i>
</a>
</td>
```



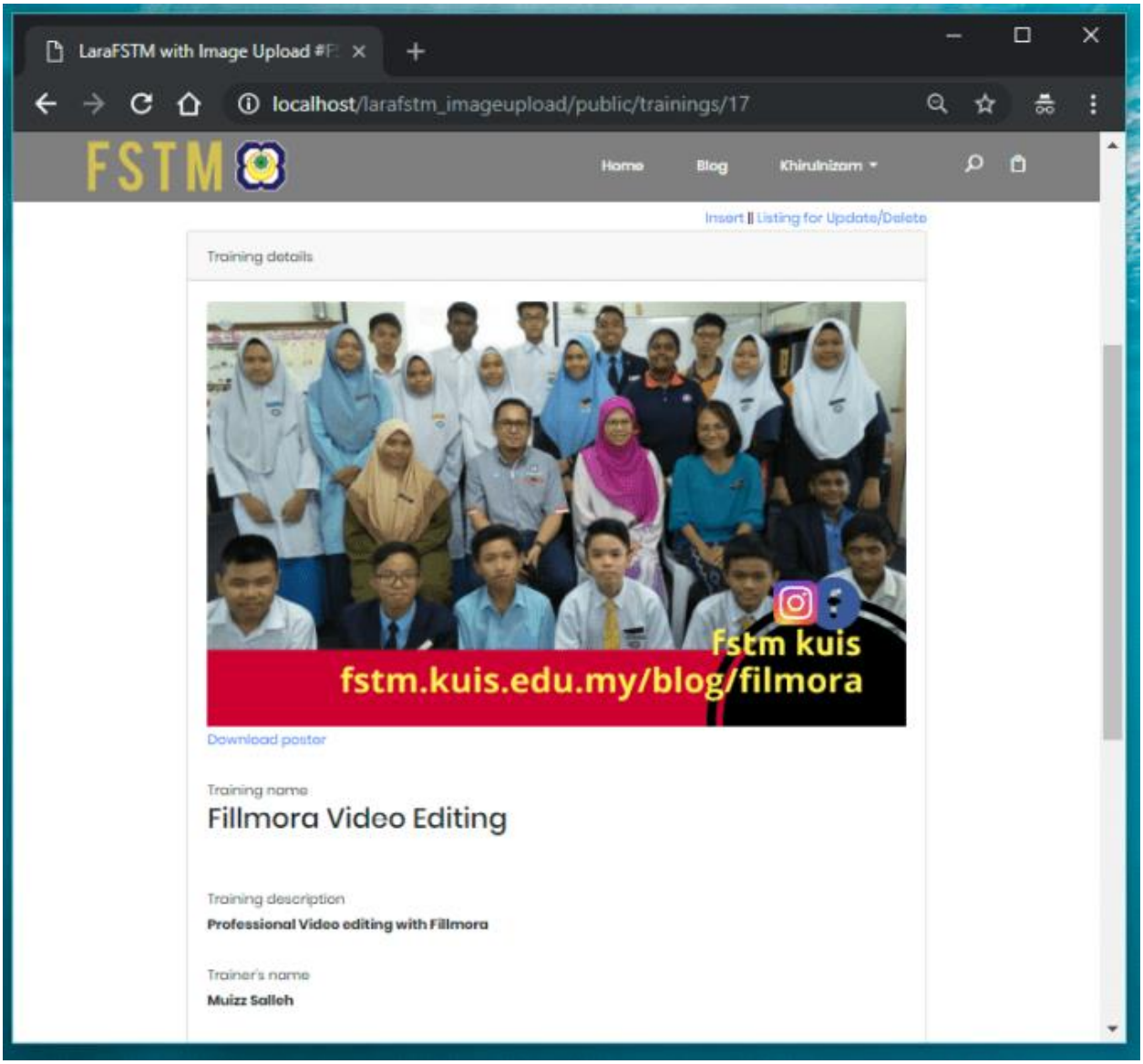
Butang papar semua maklumat dalam rekod

Seterusnya pada TrainingController@show function – tambah kod untuk mengaut semua data dari rekod yang dipilih.

```
public function show($id)
{
//TrainingController@show
//display complete record
$training = Training::find($id);
return view('trainings.show',compact('training','id'));
```

}

Function show( ) pada akhir kod akan redirect kepada page *trainings/show.blade.php* seperti di bawah.



Paparan imej yang telah dimuat-naik

Perhatikan paparan imej terhasil daripada kod segmen berikut;

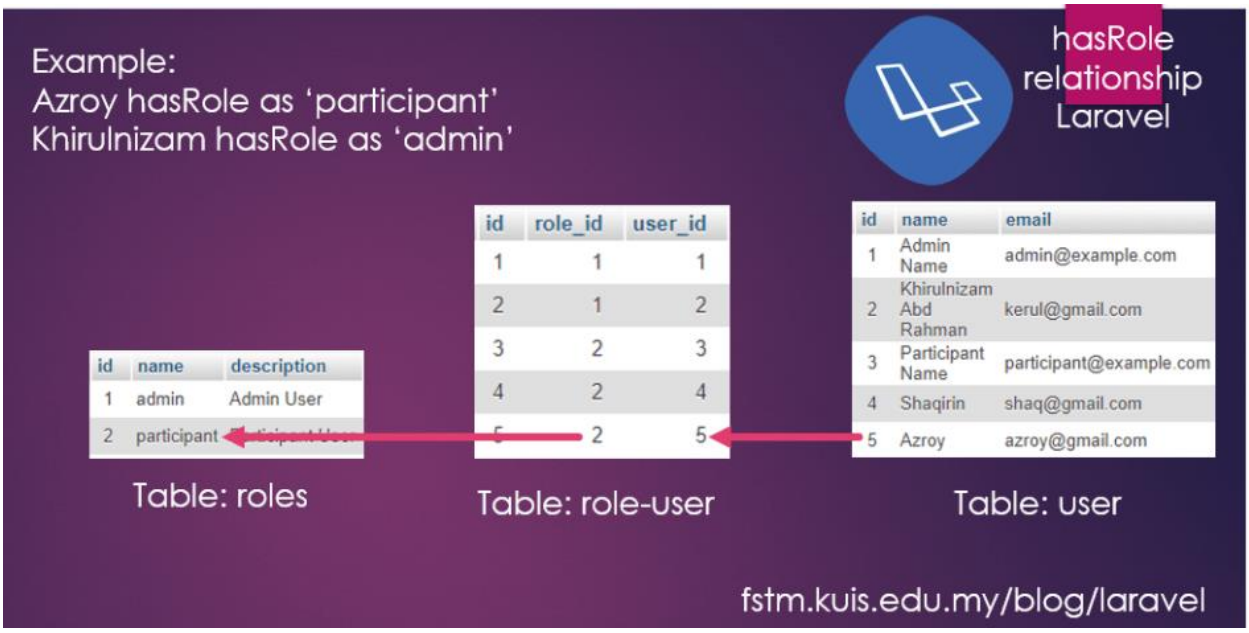
```
<!-- Sebahagian kod HTML dalam
trainings/show.blade.php segment ini memaparkan imej dari
virtual folder public/storage/uploads -->
```

```
< img class = "card-img-top"
src="{{url('storage/uploads/'.$training->filename)}}"
alt="{{$training->filename}}">
< a href="{{url('storage/uploads/'.$training-
>filename)}}"> Download poster </ a >
```

(Kod lengkap cara memaparkan imej tersebut boleh diperolehi dari https://github.com/khirulnizam/larafstm_imageupload/blob/master/resources/views/trainings/show.blade.php)

Selamat mencuba dan berjaya!

Tutorial 09 Peranan pengguna (user roles ACL)



User hasRole relationship in Laravel visualised

Kod lengkap projek Laravel kali ini InsyaAllah dah siap dan boleh muat-turun dari GITHUB – https://github.com/khirulnizam/lara_acl , cuma tutorial masih dalam proses pembangunan. Tuan/puan sila bookmark dulu, kami akan umumkan bila sudah siap. Templet Bootstrap untuk admin dasboard/panel diubahsuai dari SB-ADMIN . Telah disediakan templet SB-ADMIN dalam layouts dan juga bootstrap framework dalam *public/sbadmin*

Rangka tutorial

- Controller menambah admin baharu
- Antaramuka tambah admin baharu
- Database migration
- Controller store new admin
- Antaramuka kemaskini maklumat admin (update/delete)

Intro projek tutorial ACL Laravel

Tutorial kali ini cuba menerangkan dalam bentuk proof-of-concept bagaimana nak implement beberapa peranan pengguna (user-role) dalam Laravel. Dalam contoh sebelum ini, pengguna ada dua kategori; guest & admin. Guest tidak perlu login untuk melihat maklumat awam (public). Tetapi untuk menambah rekod, kemaskini atau padam rekod perlu pengguna yang sah yang mesti melalui proses login.

Bagaimana kalau kita ada peranan pengguna yang lain contohnya peserta boleh mendaftar kursus, mengedit user-profile atau membatalkan kursus yang telah ditdaftar?

Senarai peranan pengguna (user-roles)

Tetamu **(guest)** – boleh memaparkan maklumat umum contohnya carian semua kursus.

Peserta **(participant)** – perlu daftar, login dan boleh daftar kursus. Boleh juga melihat kembali senarai kursus yang telah didaftar, batal penyertaan, memperbaharui maklumat profile.

Penyelenggara **(admin)** – hanya didaftarkan oleh admin yang lain, perlu login untuk menyelenggara senarai kursus dan penyertaan. Boleh buat carian, kemaskini rekod training, padam rekod training, senaraikan penyertaan peserta.

Model Training dan migrations

Model Training beserta *migration file* boleh dicipta dengan arahan berikut;

php artisan make:model Training -m

Selepas laksanakan arahan di atas, buka dalam folder *database/migrations/create_trainings_table* dan tambah arahan berikut dalam *function up()* ;

```
public function up()
{
Schema::create( 'trainings' , function (Blueprint $table
) {
$table ->increments( 'id' );
$table ->string( 'trainingname' );
$table ->string( 'desc' );
$table ->string( 'trainer' );
$table ->timestamps();
});
}
```

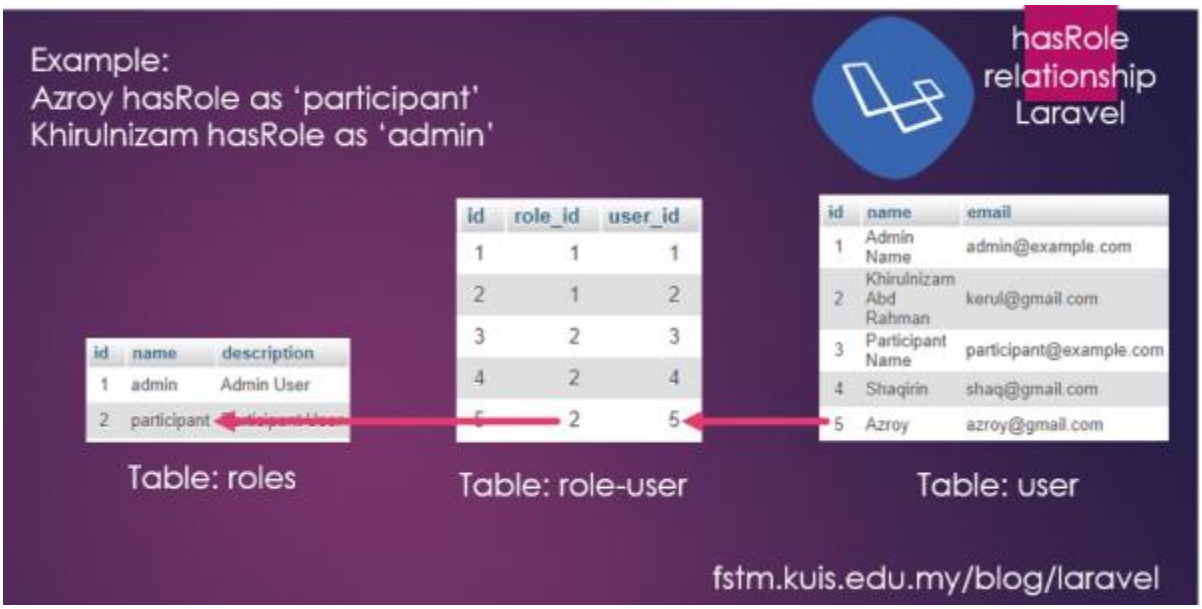
Model User dan migrations

Model beserta *migration file* telah tersedia untuk table *users* . Biasanya cuma perlu *make:auth* untuk implement ***Auth::user*** facilities.

php artisan make:auth

Model Role dan migrations

Konsep/carta rekabentuk database Role seperti di bawah. Kita cuma akan hasilkan dua roles, iaitu ‘admin’ dan ‘participant’. Sekiranya anda perlukan multiple user-roles sila google ‘permission’.



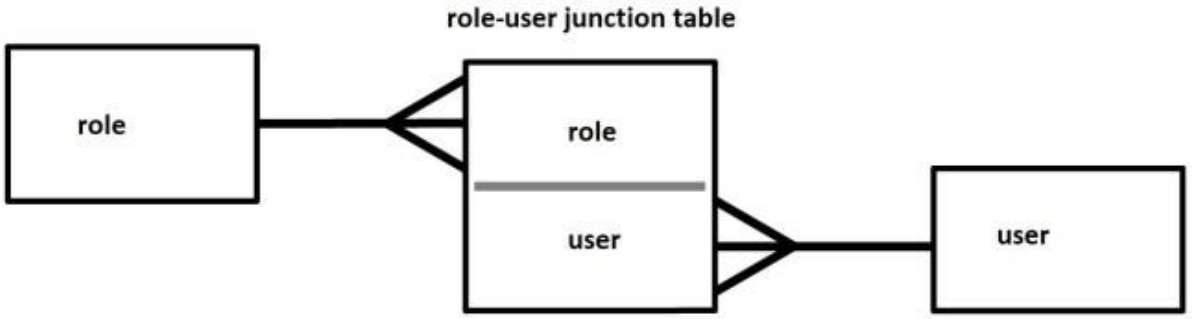
User hasRole relationship in Laravel visualised

php artisan make:model Role -m

Selepas laksanakan arahan di atas, buka dalam folder *database/migrations/create_roles_table* dan tambah arahan berikut dalam *function up()* ;

```
public function up()
{
Schema::create( 'roles' , function (Blueprint $table ) {
$table -&gt;increments( 'id' );
$table -&gt;string( 'name' );
$table -&gt;string( 'description' );
$table -&gt;timestamps();
});
}
```

Oleh kerana hubungan (*relationship*) antara pengguna dan peranan adalah *many-to-many* , makanya kita perlukan satu table tambahan penghubung roles dengan users.



Untuk mencipta migration file laksanakan arahan di bawah;

php artisan make:migration creata_role_user_table

Buka fail yang terhasil dalam database/migration/create_role_user_table dan tambah kod baris berikut;

```
public function up()
{
Schema::create('role_user', function (Blueprint $table ) {
$table -&gt;increments('id');
$table -&gt;integer('role_id')-&gt;unsigned();
$table -&gt;integer('user_id')-&gt;unsigned();
$table -&gt;timestamps();
});
}
```

Untuk menetapkan hubungan many-to-many antara User dan Role , dalam User model tambahkan *function roles( )* dan sediakan kod seperti di bawah;

```
public function roles()
{
return $this -&gt;belongsToMany(Role:: class );
//file location app/User.php
}
```

Manakala dalam Role model, tambahkan *function user( )* pula;

```
public function users()
```

```
{
return $this -&gt;belongsToMany(User:: class );
//file location app/Role.php
}
```

Seeder untuk sediakan data cubaan table users & roles

Untuk menghasilkan dummy data, tambah dalam fail *database/seeds/* ***UserTableSeeder*** *.php;*

```
public function run()
{
//
$role_admin = Role::where( 'name' , 'admin' )->first();
$role_participant = Role::where( 'name' , 'participant'
)->first();
//add admin user

$admin = new User();
$admin ->name = 'Admin Name' ;
$admin ->email = 'admin@example.com' ;
$admin ->password = bcrypt( 'secret' );
$admin ->save();
$admin ->roles()->attach( $role_admin );

$admin = new User();
$admin ->name = 'Khirulnizam Abd Rahman' ;
$admin ->email = 'kerul@gmail.com' ;
$admin ->password = bcrypt( 'abc123' );
$admin ->save();
$admin ->roles()->attach( $role_admin );
```

```
//add participant users
$participant = new User();
$participant ->name = 'Participant Name' ;
$participant ->email = 'participant@example.com' ;
$participant ->password = bcrypt( 'secret' );
$participant ->save();
$participant ->roles()->attach( $role_participant );

$participant = new User();
$participant ->name = 'Shaqirin' ;
$participant ->email = 'shaq@gmail.com' ;
$participant ->password = bcrypt( 'shaq123' );
$participant ->save();
$participant ->roles()->attach( $role_participant );

$participant = new User();
$participant ->name = 'Azroy' ;
$participant ->email = 'azroy@gmail.com' ;
$participant ->password = bcrypt( 'azroy123' );
$participant ->save();
$participant ->roles()->attach( $role_participant );

}
```

Dalam fail *database/seeds/* ***RoleTableSeeder*** *.php;*

```
public function run()
{
//
$role_employee = new Role();
$role_employee ->name = 'admin' ;
$role_employee ->description = 'Admin User' ;
$role_employee ->save();

$role_manager = new Role();
$role_manager ->name = 'participant' ;
$role_manager ->description = 'Participant User' ;
$role_manager ->save();
}
```

Dalam fail *database/seeds/* ***TrainingTableSeeder*** *.php;*

```
public function run()
{
// $this->call(UsersTableSeeder::class);
DB::table( 'trainings' )->insert([
'trainingname' => 'Laravel 101' ,
'desc' => 'Web development using PHP framework Laravel' ,
'trainer' => 'Khirulnizam Abd Rahman' ,
'filename' => 'sig-fstm.png' ,
]);

DB::table( 'trainings' )->insert([
'trainingname' => 'Android SQLite' ,
'desc' => 'Android Studio apps development with local
database using SQLite' ,
'trainer' => 'Khirulnizam Abd Rahman' ,
'filename' => 'android-sqlite.png' ,
]);

DB::table( 'trainings' )->insert([
'trainingname' => 'Video Editing Filmora' ,
'desc' => 'An easy video editing using Filmora editor ' ,
'trainer' => 'Muizz Salleh' ,
'filename' => 'muiz-filmora.png' ,
]);

}
```

Dalam fail *database/seeds/* ***DatabaseSeeder*** *.php*

```
public function run()
{
// Role comes before User seeder here.
$this ->call(RoleTableSeeder:: class );
// User seeder will use the roles above created.
$this ->call(UserTableSeeder:: class );
// Training seeder dummy data
$this ->call(TrainingTableSeeder:: class );

}
```

Seterusnya dalam model app/User.php , tambahkan tiga fungsi-fungsi berikut;

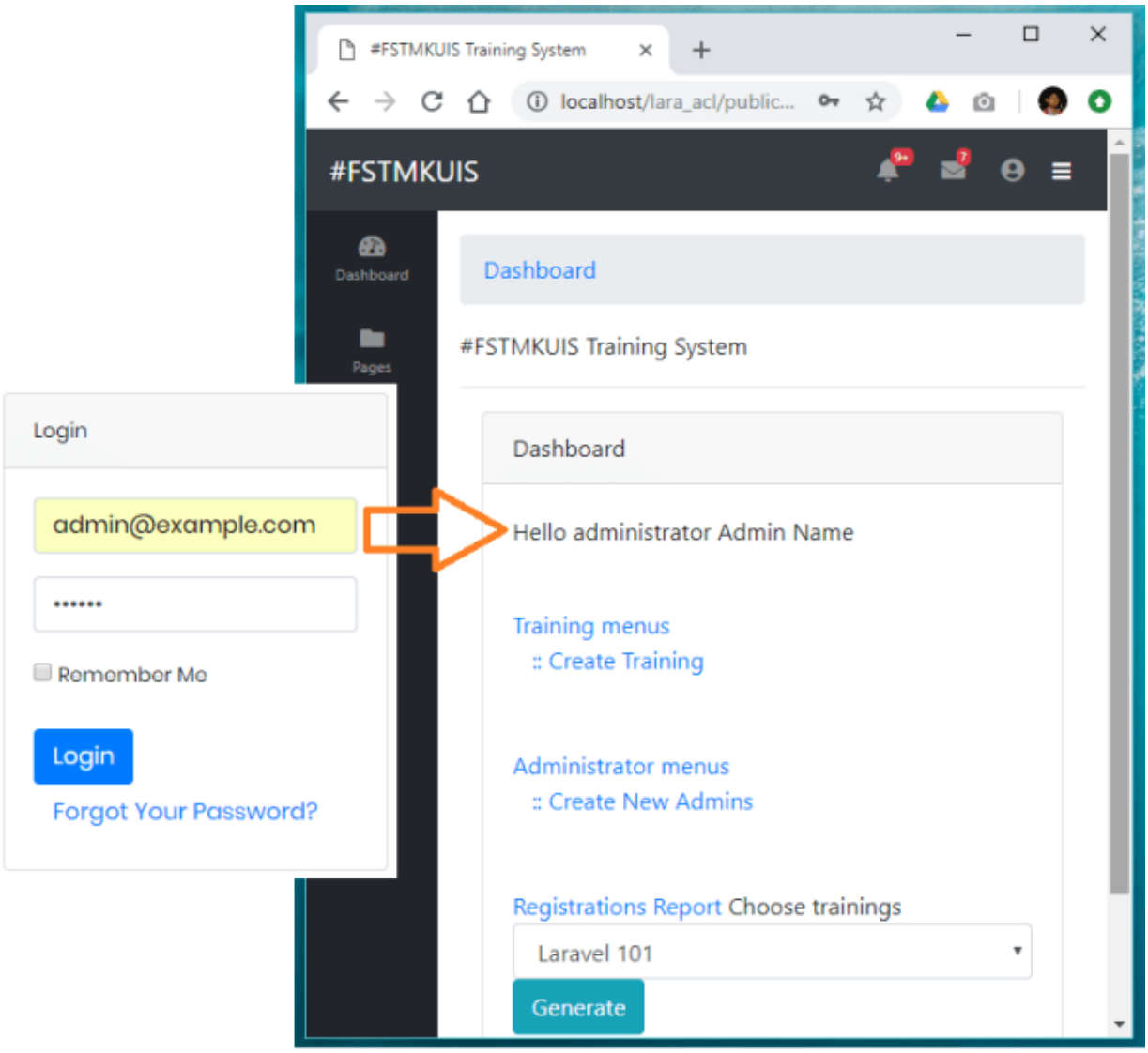
```
public function authorizeRoles( $roles )
{
if ( is_array ( $roles )) {
return $this ->hasAnyRole( $roles ) ||
abort(401, 'This action is unauthorized.' );
}
return $this ->hasRole( $roles ) ||
abort(401, 'This action is unauthorized.' );
}
/**
* Check multiple roles
* @param array $roles
*/

public function hasAnyRole( $roles )
{
return null !== $this ->roles()->whereIn( 'name' , $roles
)->first();
}
/**
* Check one role
* @param string $role
*/

public function hasRole( $role )
{
return null !== $this ->roles()->where( 'name' , $role )-
>first();
}
```

Percubaan login sebagai admin / participant

Berdasarkan UserTableSeeder di atas, terdapat pengguna yang diberikan role ‘admin’ dan ‘participant’. Cuba paparkan skrin login dan cuba kedua-dua

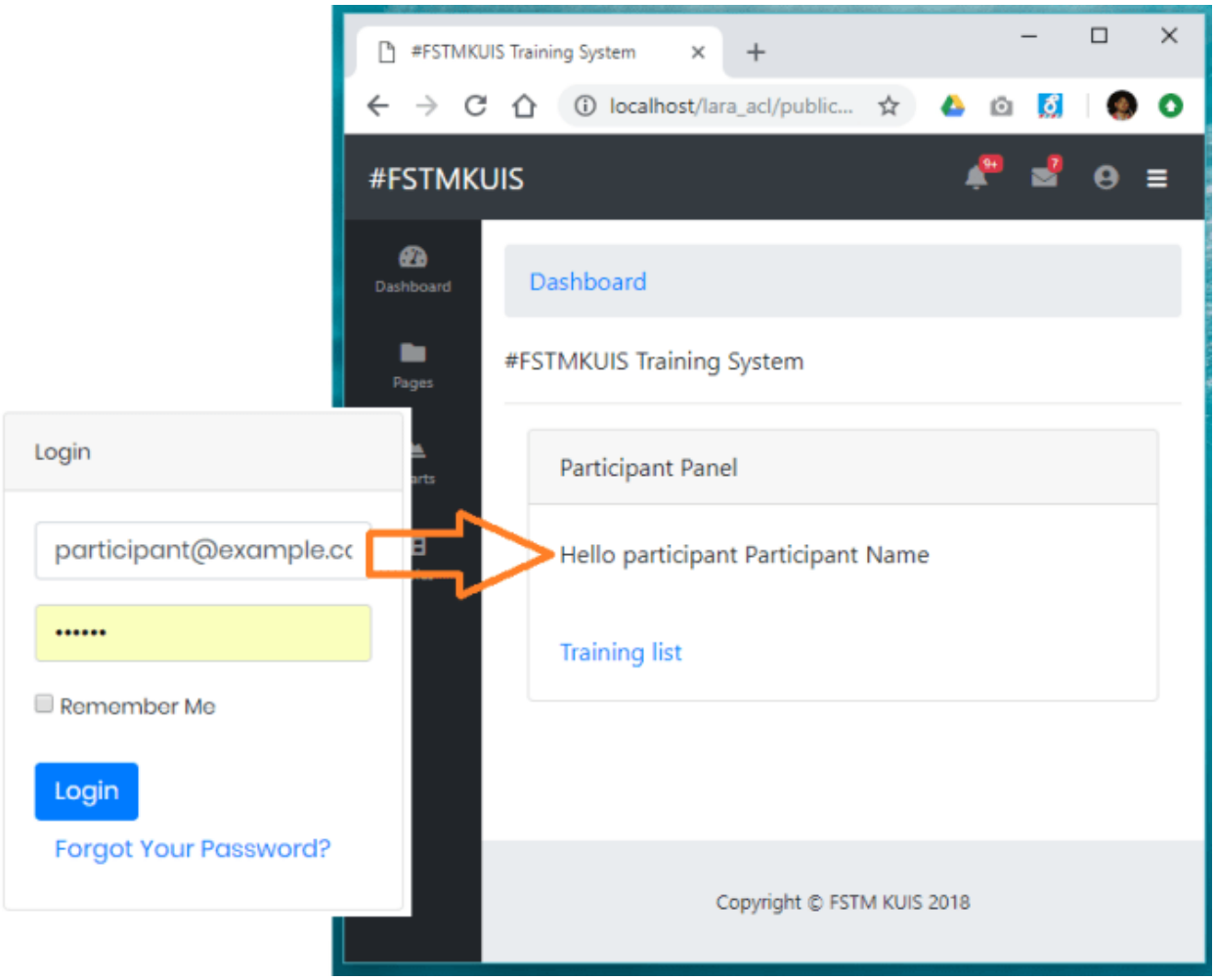
peranan pennggguna berikut. Perhatikan bahawa kedua-dua peranan user mendapat skrin panel pengguna yang berbeza.

Admin username: *admin@example.com* ; dengan password *secret.*



User admin login, paparan skrin admin panel

Participant username: *participant@example.com* ; dengan password *secret* .



User participant login, paparan skrin participant panel

Perbezaan paparan ini boleh dilakukan dengan menujukan kepada skrin yang berbeza apabila jenis peranan berbeza login masuk sistem. Ini boleh dilakukan dalam ***app/HomeController.php***

```
public function index(Request $request )
{
if (Auth::user()->hasRole( 'admin' )){
$trainings = Training::all()->toArray();
return view( 'home' , compact( 'trainings' ));
}
else if (Auth::user()->hasRole( 'participant' )){
return view( 'homeparticipant' );
}
}
```

Apabila peranan pengguna adalah ‘admin’ (hasRole), HomeController akan redirect ke ***resources/views/home.blade.php*** , manakala sebaliknya apabile peranan pengguna (hasRole) adalah ‘participant’- maka akan dipaparkan pula ***resources/views/homeparticipant.blade.php***

resources/views/homeparticipant.blade.php

```
@extends('layouts.sbadmin')
@section('content')

< div class = "container" >

< div class = "row justify-content-center" >

< div class = "col-md-10" >

< div class = "card" >

< div class = "card-header" >Participant Panel</ div >


< div class = "card-body" >
@if (session('status'))

< div class = "alert alert-success" >
{{ session('status') }}
</ div >

@endif


Hello participant {{ Auth::user()->name }}


< a href = "{{ url('trainings') }}" > Training list</ a >


</ div >
```

```
</ div >

</ div >

</ div >

</ div >

@endsection
```

resources/views/home.blade.php

```
@extends('layouts.sbadmin')
@section('content')

< div class = "container" >

< div class = "row justify-content-center" >

< div class = "col-md-10" >

< div class = "card" >

< div class = "card-header" >Dashboard</ div >


< div class = "card-body" >
@if (session('status'))

< div class = "alert alert-success" >
{{ session('status') }}
</ div >

@endif


Hello administrator {{ Auth::user()->name }}
```

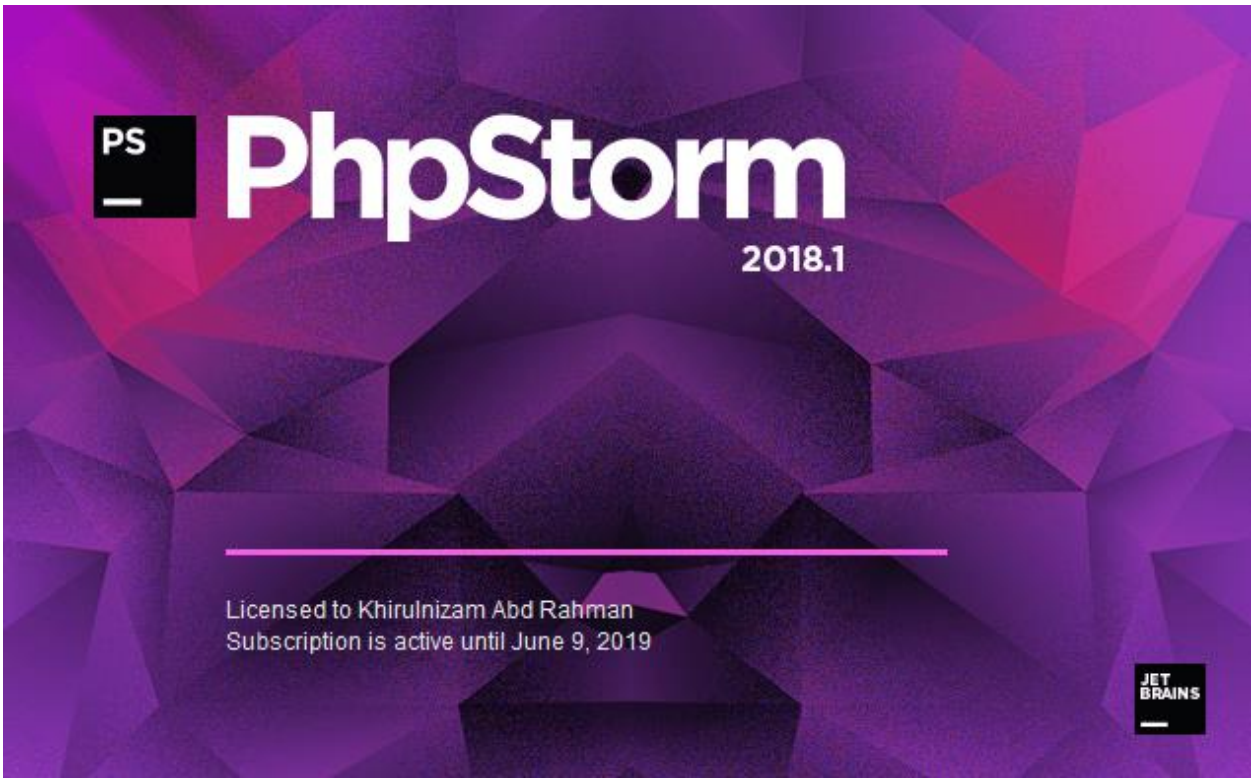
```
< a href = "{{ url('trainings') }}" > Training menus</ a
>

< a href = "{{ url('trainings/create') }}" >  
:: Create Training</ a >


< a href = "{{ url('admins') }}" > Administrator menus</
a >

< a href = "{{ url('admins/create') }}" >   ::
Create New Admins</ a >


< a href = "#" > Registrations Report</ a >
Choose trainings

< form method = "GET" action =
"{{url('registerreports')}}" class = "form-inline" >
@csrf
< select name = "id" class = "form-control" >
@foreach($trainings as $training)
< option value = "{{$training['id']}}" >
{{$training['trainingname']}} </ option >
@endforeach
</ select >

< button type = "submit" class = "btn btn-info" >
Generate </ button >
</ form >


</ div >

</ div >

</ div >
```

```
</ div >

</ div >

@endsection
```

Kod lengkap projek Laravel kali ini InsyaAllah dah siap dan boleh muat-turun dari **GITHUB – https://github.com/khirulnizam/lara_acl**

Selamat mencuba dan berjaya!
Rujukan: https://medium.com/@ezp127/laravel-5-4-native-user-authentication-role-authorization-3dbae4049c8a

Tutorial 10 PHPStorm IDE Laravel



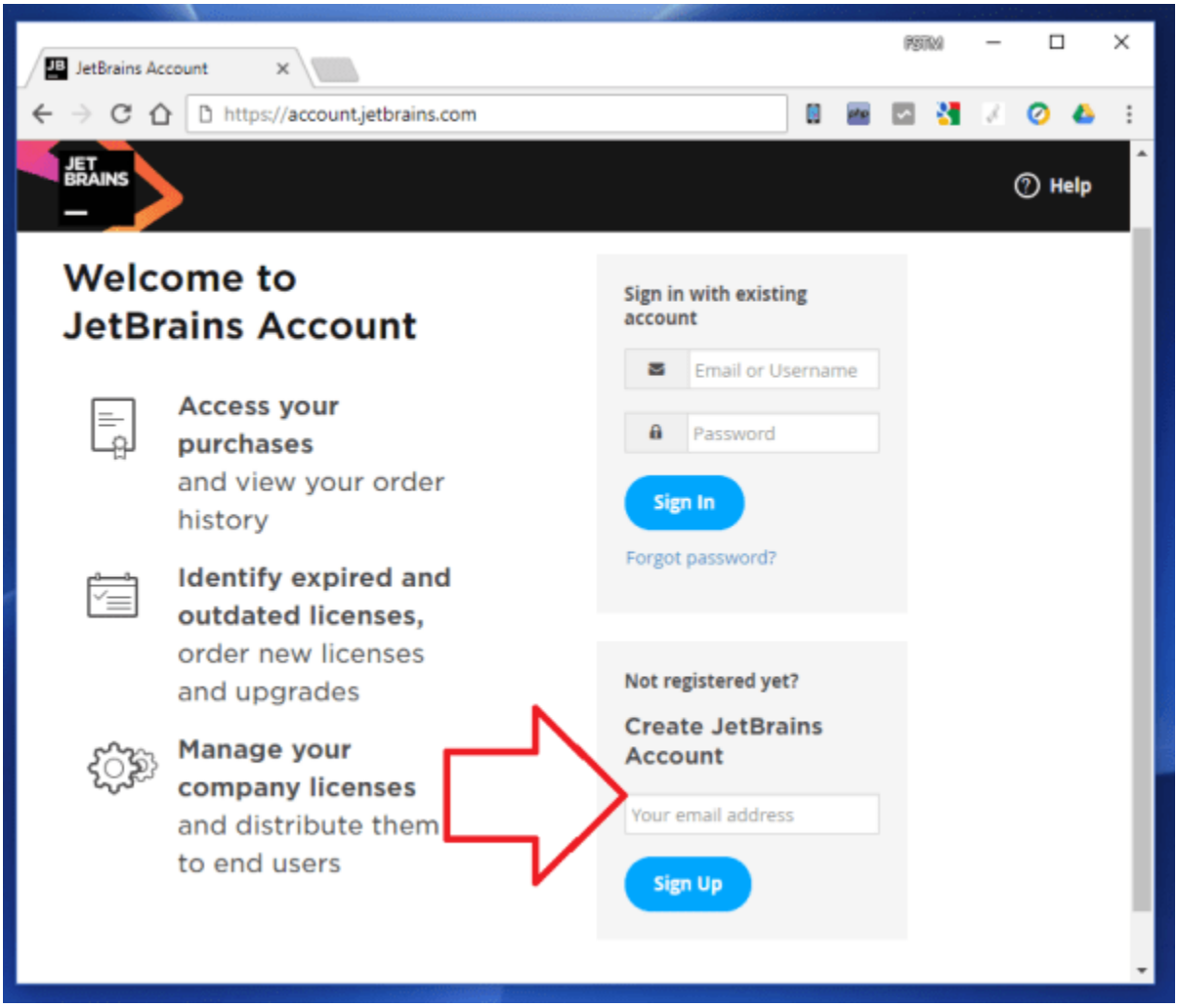
IDE (Integrated Development Environment) yang dimaksudkan bukan sahaja berupaya menulis kod HTML, CSS, JavaScript, PHP & Laravel, ia juga mampu membantu mengesan larat sintax, cadangan autocomplete. IDE seperti Eclipse for PHP, Visual Studio dan PHPStorm antara yang berkebolehan sebagai IDE yang terbaik, manakala Atom, VSCode & Sublime bertindak hanya sebagai editor kod sahaja. Penulis memilih PHPStorm sebagai IDE kegemaran (artikel ini cuma pendapat peribadi penulis berdasarkan pengalaman).

Pemasangan Perisian PHPStorm

PHPStorm boleh dimuat-turun dari laman syarikat pembangunnya iaitu JetBrain (yang terkenal dengan IDE Java IntelliJ – IDE terbaik Java). Namun ianya bukan perisian percuma (sekiranya memerlukan sumber terbuka/pilihan percuma rujuk *Eclipse for PHP*). Masa percubaan hanya 30hari, selepas itu sekiranya syarikat anda ada bajet boleh beli lesen.

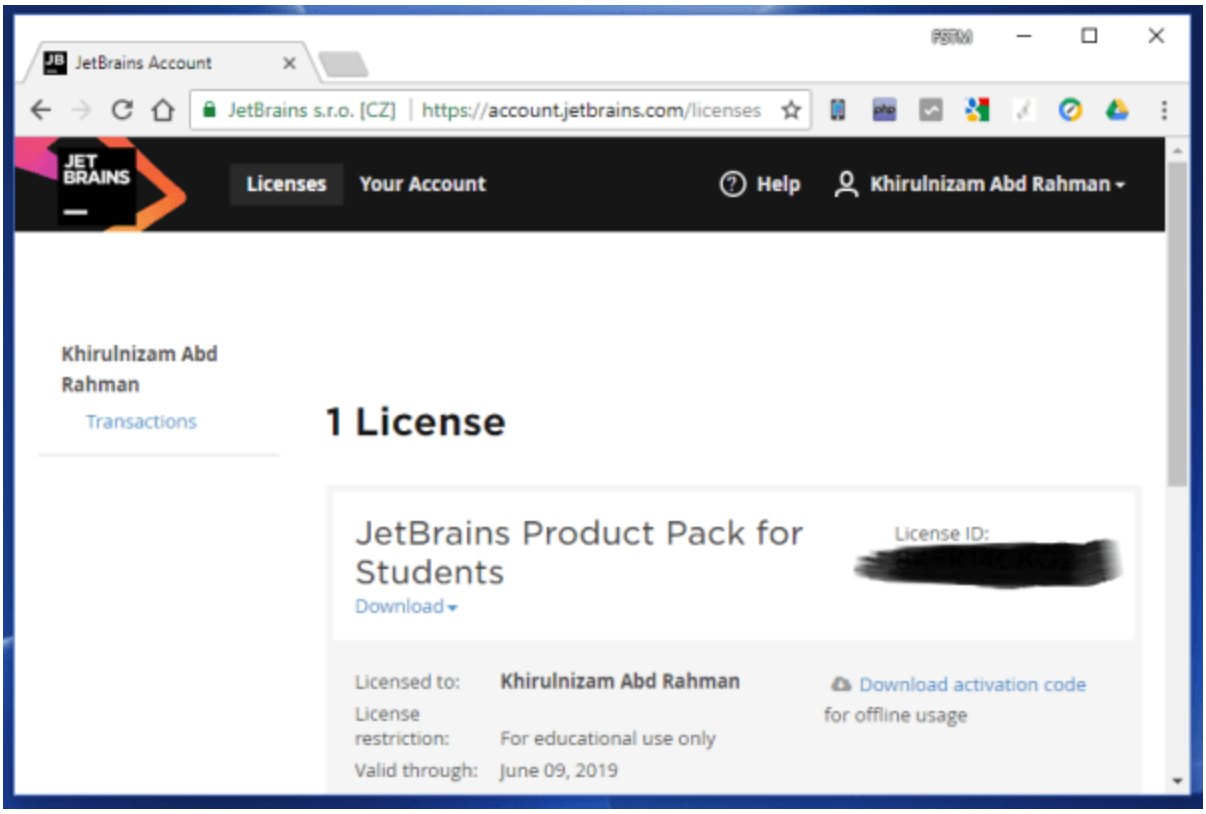
Lesen percuma pendidikan

Pastikan anda berdaftar dulu dalam website JetBrains – **https://account.jetbrains.com** .



Sekiranya anda pelajar atau pensyarah IPTA/IPTS boleh dapatkan lesen penggunaan percuma dengan daftar menggunakan email *edu* yang disediakan institusi. Penulis akan menunjukkan cara permohonanan lesen percuma menggunakan email ***@kuis.edu.my .

Boleh mohon lesen untuk pendidikan/kelas/makmal komputer di sini , atau lesen individu di sini . Untuk permohonan lesen individu pelajar/pensyarah, sila sediakan email yang ada ***edu*** pada alamat email anda atau boleh sertakan kad matrix/pelajar bersama permohonan.



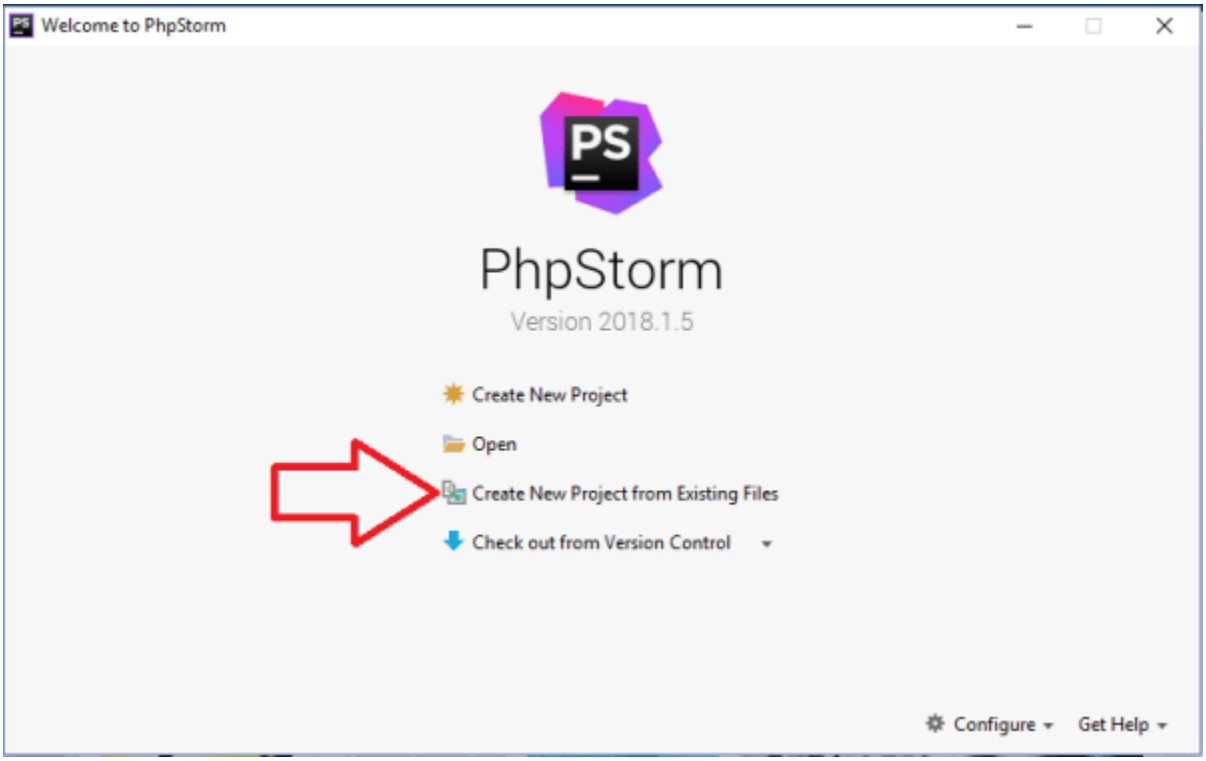
Selepas anda berjaya mendapatkan lesen pendidikan, login akaun anda melalui perisian PHPStorm yang telah anda download dan install. Perisian PHPstorm desktop boleh diperolehi di sini;

https://www.jetbrains.com/phpstorm/

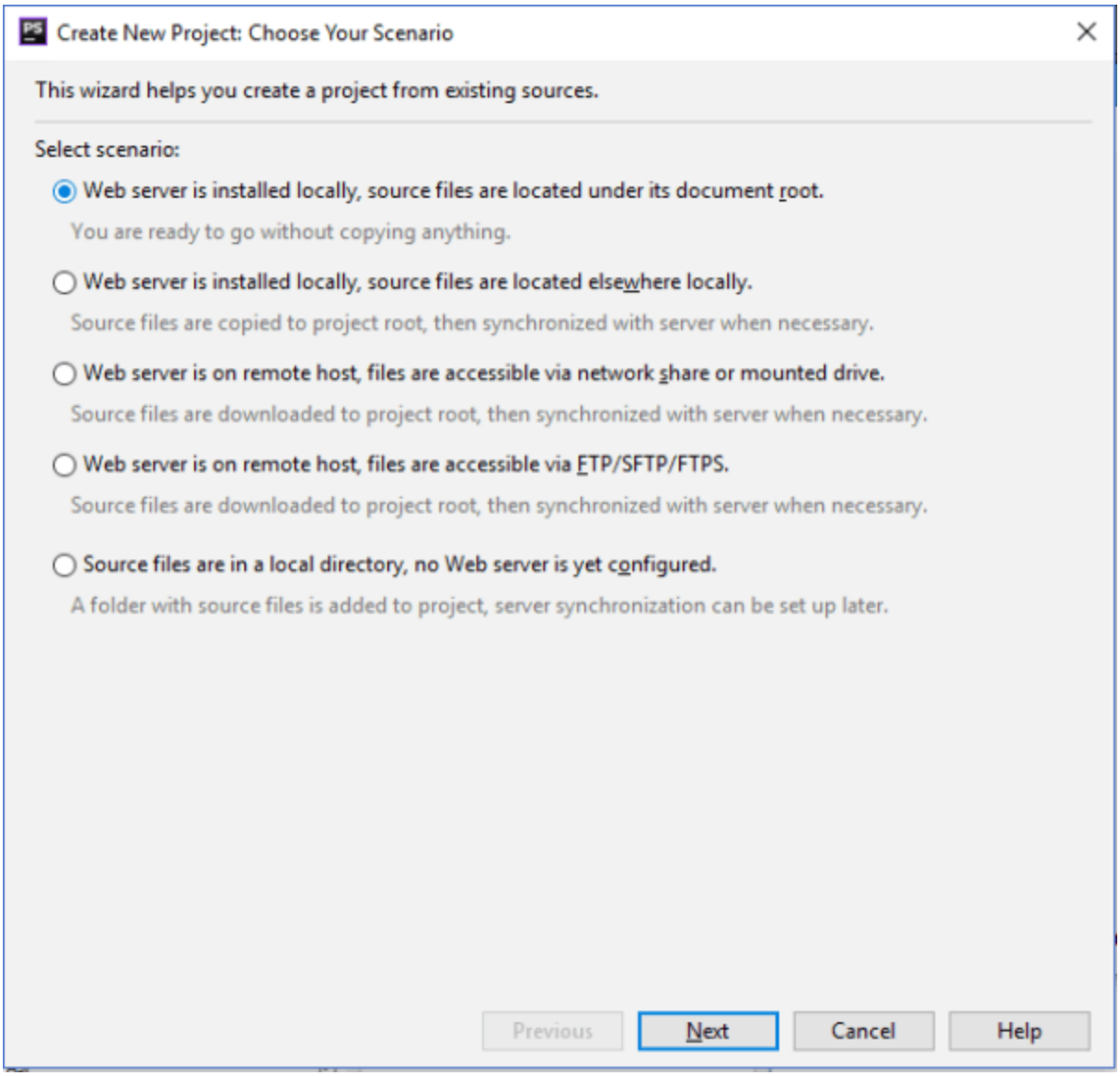
Projek Laravel

Pengujian ini dibuat melalui PC dengan Windows10 dan RAM 8GB (4GB kurang disyorkan memandangakan perisian agak berat). Projek yang akan dibangunkan menggunakan framework Laravel. Memandangkan BLADE juga disokong dalam PHPStorm.

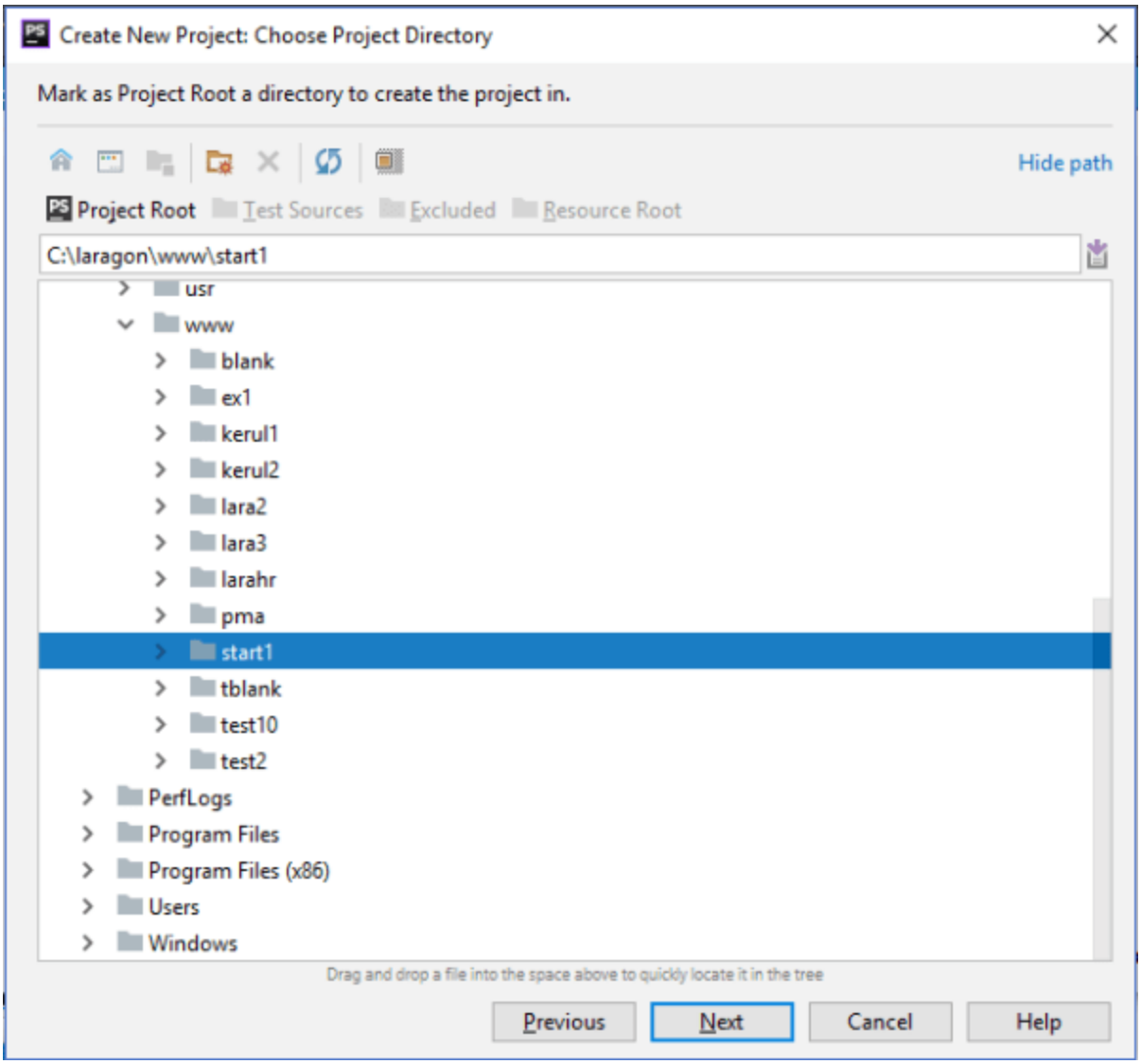
Bina projek daripada folder yang dah diwujudkan dan *point* kepada folder projek PHP/Laravel. (sekiranya anda ingin tahu cara membuat projek baharu Laravel sila ke link ini http://fstm.kuis.edu.my/blog/laravel1)



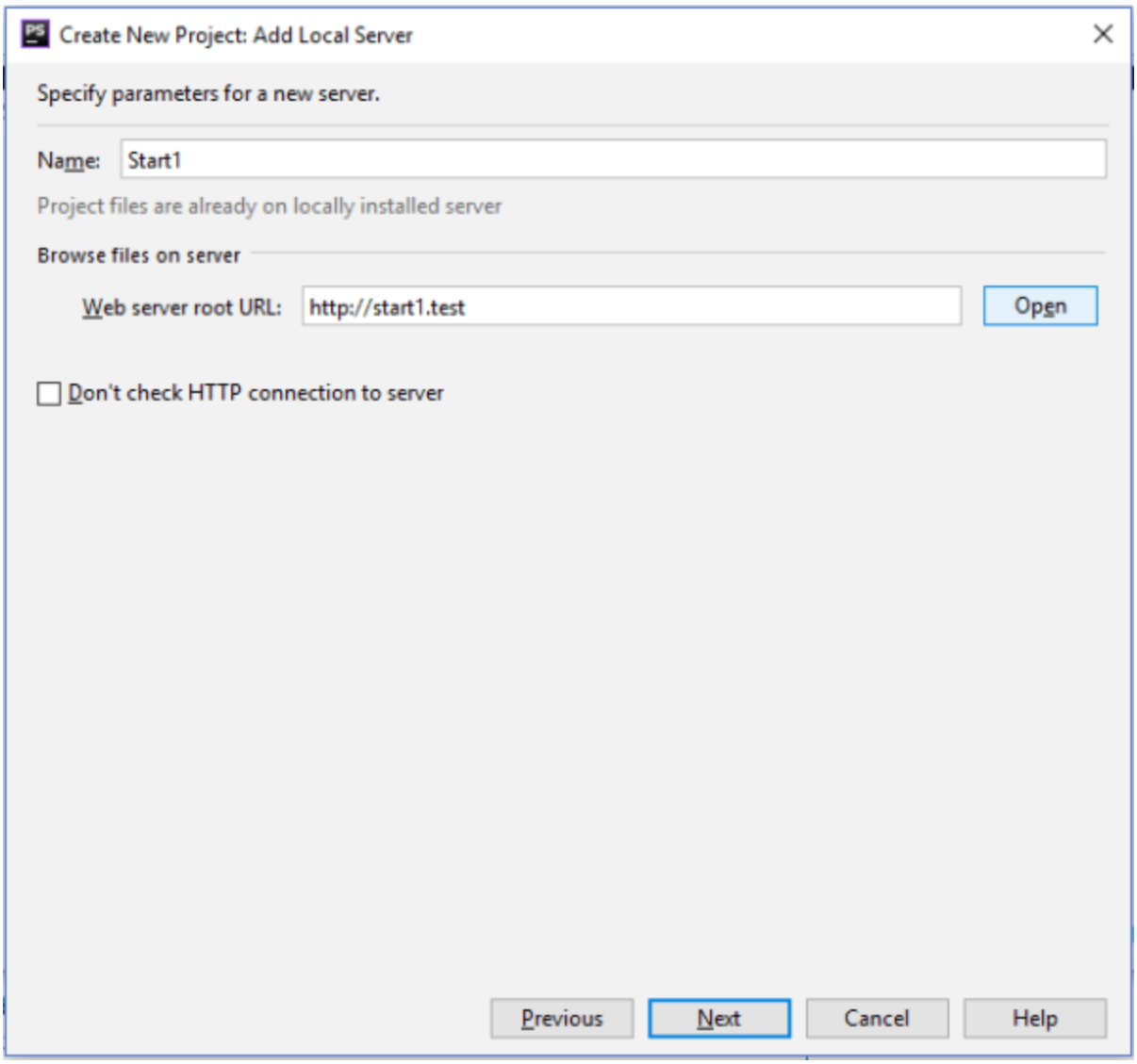
Set lokasi fail dan server (persekitaran *localhost*).



Tentukan lokasi fail projek Laravel anda.

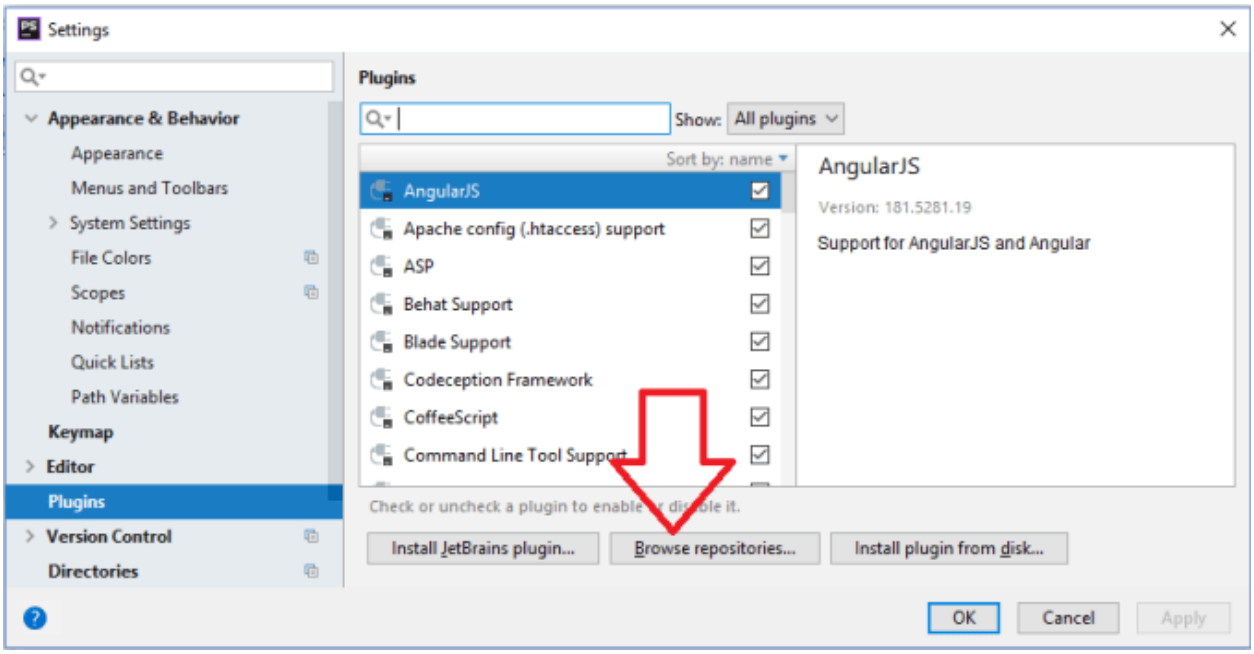


Tentukan maklumat web server (seperti yang disediakan oleh Laragon). Klik NEXT dan kemudian FINISH.

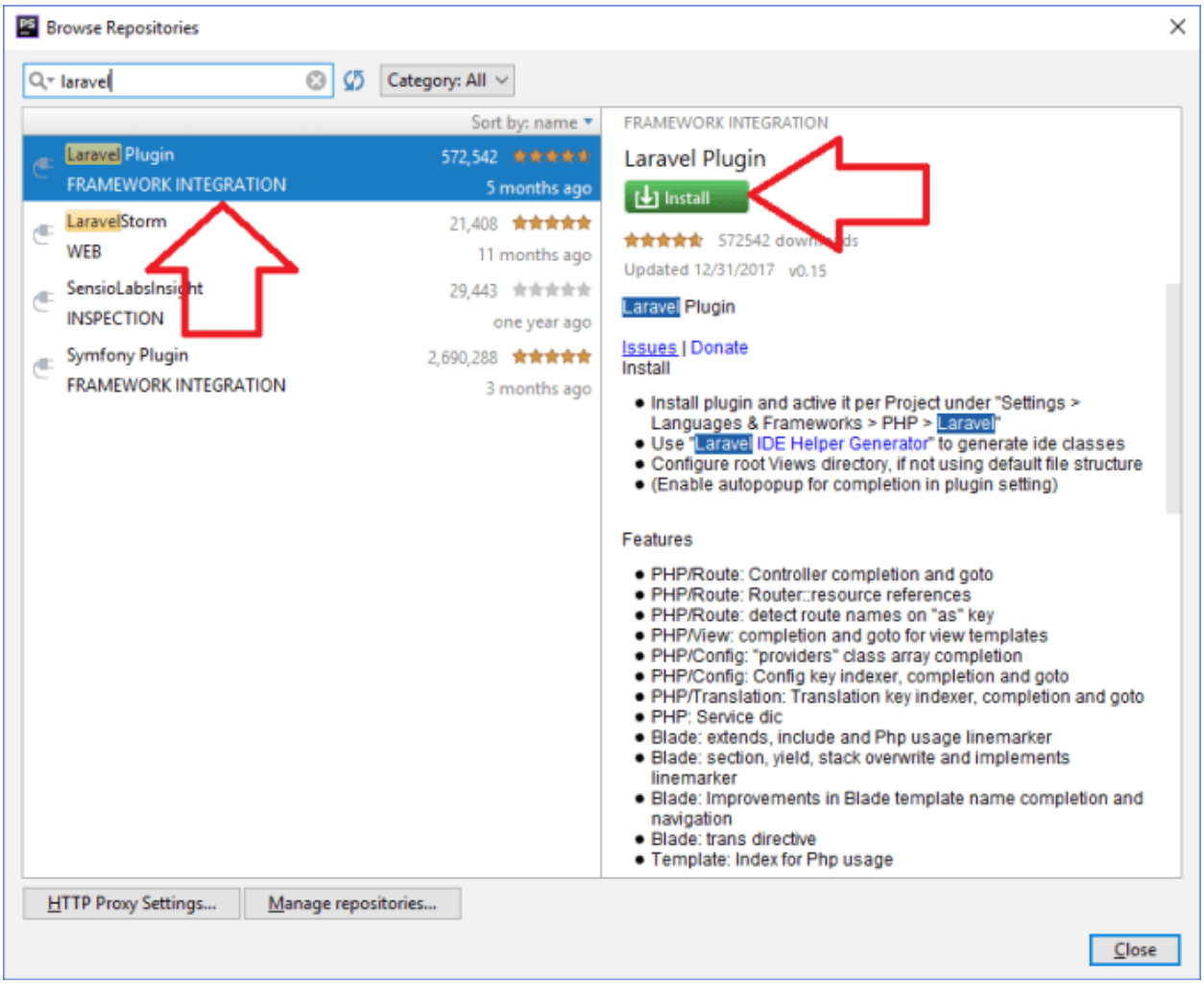


Pengurusan kod Laravel dalam PHPStorm

Untuk install plug-ins Laravel, pergi ke menu *File->Settings->Plugins* (rujuk gambar di bawah).

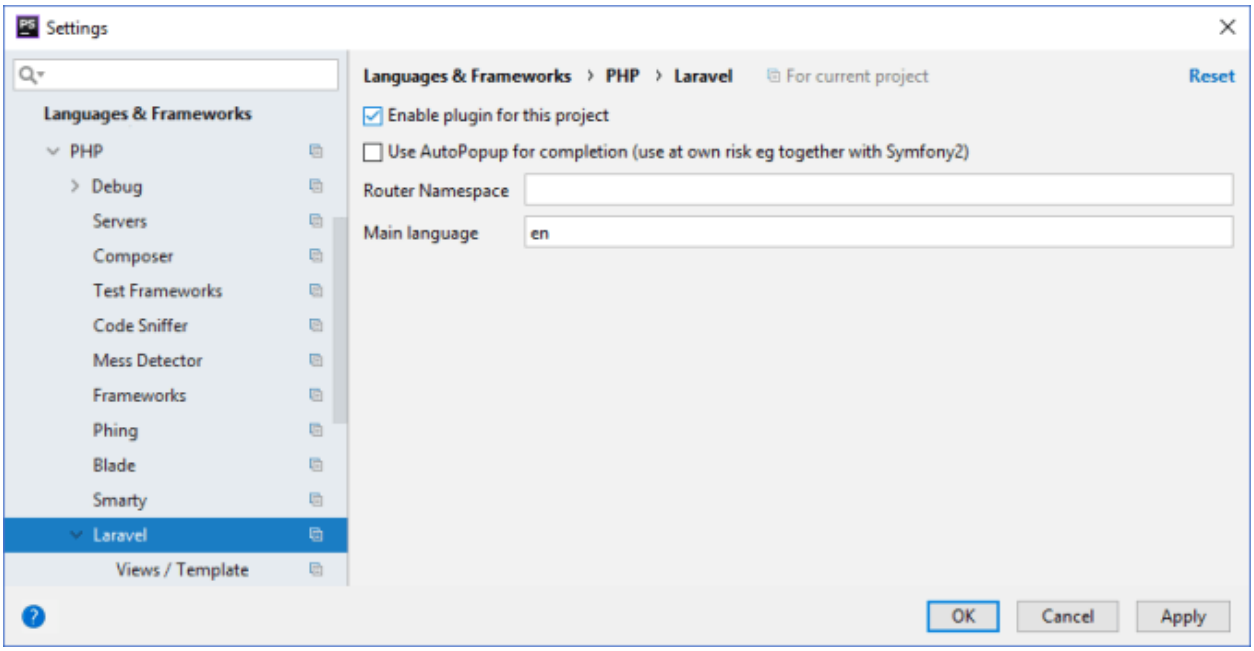


Seterusnya carian Laravel, dan klik butang install seperti di bawah.



Pastikan restart PHPStorm untuk mengaktifkan plug-ins Laravel yang baru kita install.

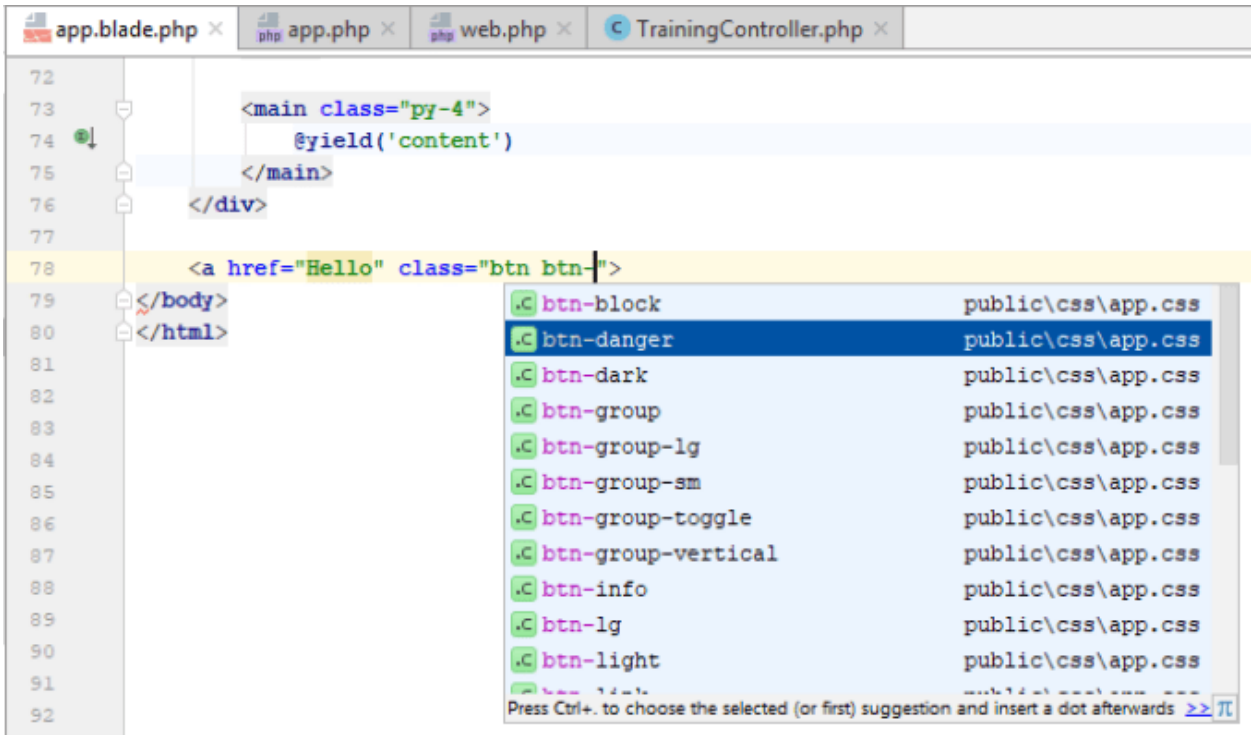
Kemudian aktifkan plug-ins Laravel dalam setiap projek yang guna Laravel di *File->Settings->Languages&Frameworks->Laravel*



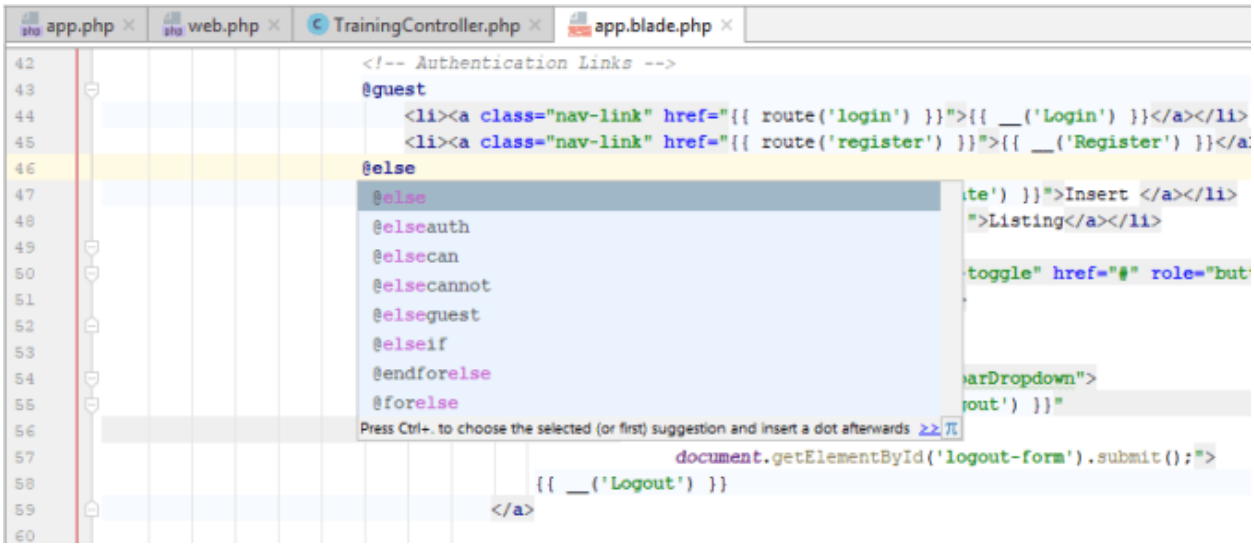
(Install plug-ins untuk Laravel boleh juga muat-turun dari sini untuk install secara manual https://plugins.jetbrains.com/plugin/7532-laravel-plugin)

Penulisan kod HTML/PHP/Laravel/Blade

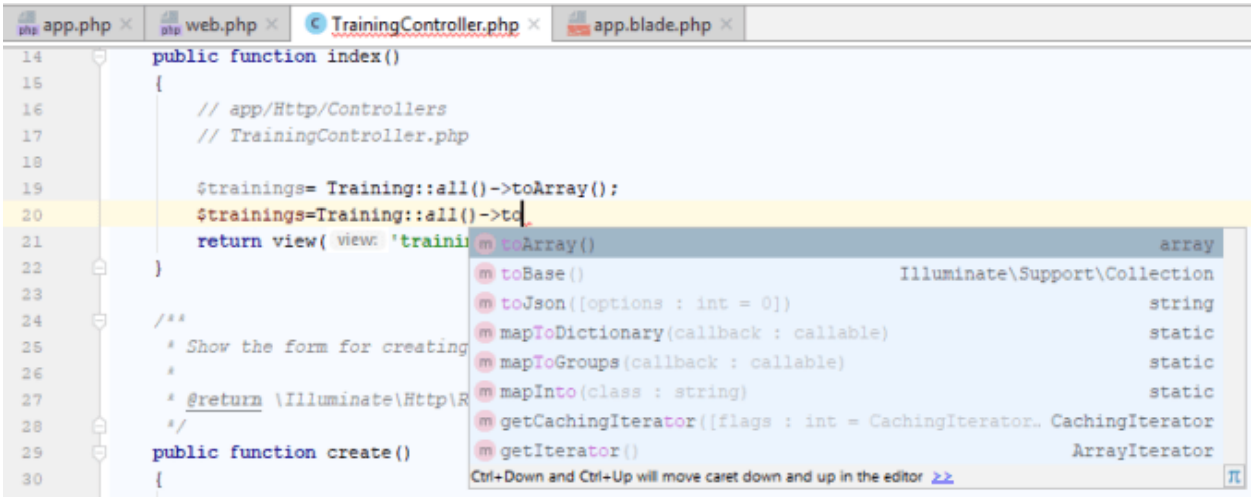
Contoh *suggestion* dalam kod HTML/Bootstrap/PHP



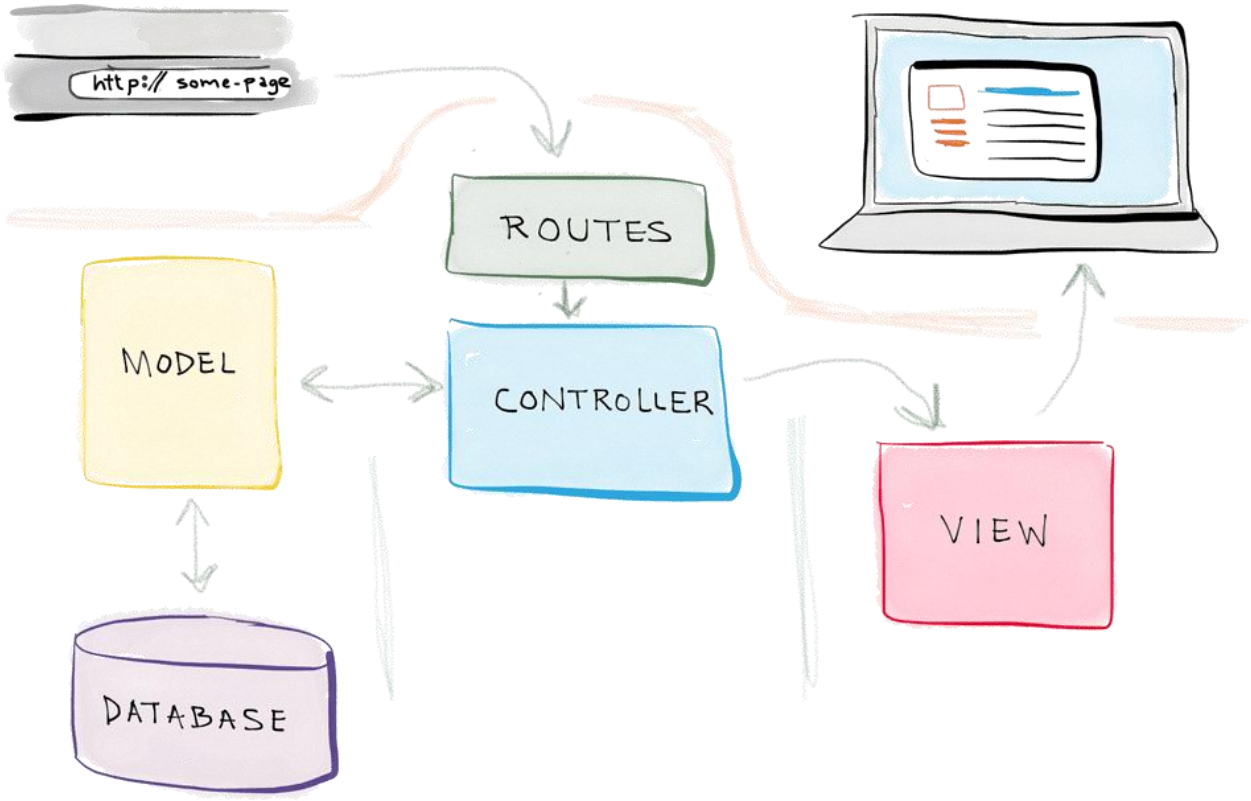
Contoh *suggestion* dalam kod Blade



Contoh *suggestion* dalam kod Laravel



Tambahan: MVC dan Penulis



Laravel mengaplikasi konsep rekabentuk MVC.

Model – merujuk kepada capaian/query kepada pangkalan data dan mengembalikan data yang diminta.

Views – laman yang memaparkan data atau antaramuka.

Controllers – mengendalikan permintaan pengguna, capaian data kepada Model dan menghantar data kepada View.

(Sumber: Self Taught Coders)

Penulis tutorial:
KHIRULNIZAM ABD RAHMAN, Pensyarah Jabatan Sains Komputer, FSTM KUIS.

Beliau merupakan seorang trainer dalam bidang pengaturcaraan server dan antaramuka web (web front-end & backend) semenjak tahun 2000. Disamping itu juga amat berminat dalam pembangunan applikasi mobile Android, JSON, LARAVEL dan PHP-MySQL.

Antara kursus yang beliau kendalikan;

- Android Studio fstm.kuis.edu.my/blog/android
- Pembangunan Web dengan PHP&MySQL fstm.kuis.edu.my/blog/php
- Pembangunan Sistem Web menggunakan kerangka Laravel Khirulnizam.com

Blog peribadi beliau di khirulnizam.com . Beliau boleh dihubungi melalui email khirulnizam@gmail.com , atau Whatsapp: http://wasap.my/60129034614