Praktikum Pemrograman Web Lanjut

(Laporan Tugas Akhir)



Oleh :

Muhammad Rohman Irfanuddin (16/401045/SV/11549)

M. Ariq Srihadi W (16/396288/SV/10501)

Kelas Praktikum AB

Ilmu Komputer dan Sistem Informasi

Universitas Gadjah Mada

**Analisis Kebutuhan Sistem**

Disini kelompok kami memilih Sistem Informasi Reservasi Tiket Kereta Api. Sistem ini akan berisikan prosedur pendaftaran, pesan, pembayaran, canceling, dan pergantian jadwal.

**Kebutuhan Sistem**

1. **Kebutuhan Fungsional**
2. Sistem harus mampu melakukan input pendataan jadwal kereta api
3. Pengguna memasukkan tentang data-data kereta api yang meliputi nama kereta api, jenis kereta api, tujuan kereta api, waktu.
4. Pengguna bisa menampilkan data kereta api secara keseluruhan.
5. Pengguna mampu menampilkan data kereta api berdasarkan kategori tertentu.
6. Pengguna dapat mengelompokkan kereta api berdasarkan kategori tertentu.
7. Pengguna mampu menampilkan data kereta api yang terakhir/sering di naikinya.
8. Admin dapat melakukan input berita.
9. Pendataan anggota
10. Pengguna dapat memasukkan data user baru yang meliputi nama, no. ktp, alamat, no. rekening.
11. User dapat mencetak kartu anggota.
12. Admin dapat menampilkan data pemesan tiket berdasarkan kategori tertentu.
13. Admin dapat menampilkan laporan data pemesan tiket setiap tahunnya.
14. Sistem harus dapat melakukan transaksi pemesanan tiket
15. Sistem dapat melakukan transaksi pemesanan tiket.
16. Sistem dapat melakukan sinkronisasi antara jadwal kereta api dan pemesan.
17. Sistem dapat menampilkan kereta api jenis apa saja yang di pesan oleh user tertentu.
18. Sistem dapat membuat file pdf otomatis ketika proses pemesanan tiket selesai.
19. Sistem dapat mengetahui jadwal kapan saja kereta api yang sedang dipesan oleh pengguna.
20. Sistem dapat menampilkan data peminjam yang paling sering melakukan reservasi tiket online.
21. **Kebutuhan Nonfungsional**
22. Operasional :

* Menggunakan windows NT or higher, linux.
* Spesifikasi komputer minimal pentium II atau kalau menggunakan gadget minimal berprosesor snapdragon.
* Kebutuhan RAM minimal 256MB untuk PC maupun gadget.
* Menggunakan browser Mozilla Firefox atau Google Chrome.

**Keamanan**

* Sistem aplikasi dan database dilengkapi dengan password.
* Dilengkapi CCTV di lobi komputer stasiun.

**Informasi**

* Digunakan untuk menampilkan tata cara pendaftaran anggota baru.
* Digunakan untuk menampilkan informasi bila user salah memasukkan password.

**Kinerja**

* Waktu login tidak boleh lebih dari 25 menit.
* Dibuat seringan mungkin agar tidak loading terlalu lama.

**Domain Analysis**

* 1. **Introduction**

Sistem Ticketing, pemesanan tiket kereta api berbasis web

* 1. **Glossary**

Reservasi – Pemesanan

Search Ticket – Mencari tiket yang kosong

Ganti Jadwal – Mengganti jadwal perjalanan

Canceling – Membatalkan pemesanan tiket

View User Info – Melihat informasi user

Registrasi – Pendaftaran user baru

Login – Masuk dalam sistem

Laravel – Framework yang akan digunakan untuk membangun sistem

* 1. **General Knowledge About The Domain**

Sistem dapat melayani proses pemesanan tiket kereta api via website.

* 1. **Customers and Users**
* Customers : Publik
* User : Orang yang akan memesan tiket dan tidak mau ribet
  1. **The Environment**
* Sistem akan berbasis web
* Akan ada admin yang dapat mensinkronkan data jadwal kereta yang tersedia
* Pengguna yang dapat mengakses dari mana saja
  1. **Tasks and Procedures Currently Performed**
* Dapat melakukan registrasi user
* Dapat melakukan login
* Dapat melakukan reservasi tiket
* Dapat melakukan canceling reservation ticket
* Dapat melakukan pergantian jadwal perjalanan
* Dapat melihat jadwal kereta api yang ada
* Dapat melakukan cetak tiket
  1. **Competing Software**
* Dapat melakukan pemesanan dan pembayaran dalam genggaman
* Sistem enteng dan cepat
* Mencakup pemesanan semua jenis kereta api
  1. **Similarities to Other Domains**
* Dapat melakukan pemesanan tiket berbasis online
* Dapat melakukan sinkronisasi dengan data jadwal kereta api

**Functional Requirements**

* Sistem harus mampu melakukan input pendataan jadwal kereta api
* Pendataan anggota
* Sistem harus dapat melakukan transaksi pemesanan tiket
* Dapat meampilkan data jadwal kereta api yang available pada waktu tertentu

**Quality Requirements**

* Response time

< 10sm

* Throughput

Bisa melakukan 100 transaksi dalam waktu bersamaan

* Resource usage
* Menggunakan data dari PT KAI untuk sinkronisasi data
* Membuat database sendiri untuk menyimpan data pengguna dan transaksi
* Reliability

Dapat melakukan transaksi dari manapun kapanpun

* Availability

Selalu on 24 jam, karena dibuat server cadangan

* Recovery from failure

Terdapat server cadangan yang siap membackup jika suatu saat terjadi down

* Allowances for maintainability and enhancement

Admin, CS, TS

* Allowances for reusability

Server Cadangan

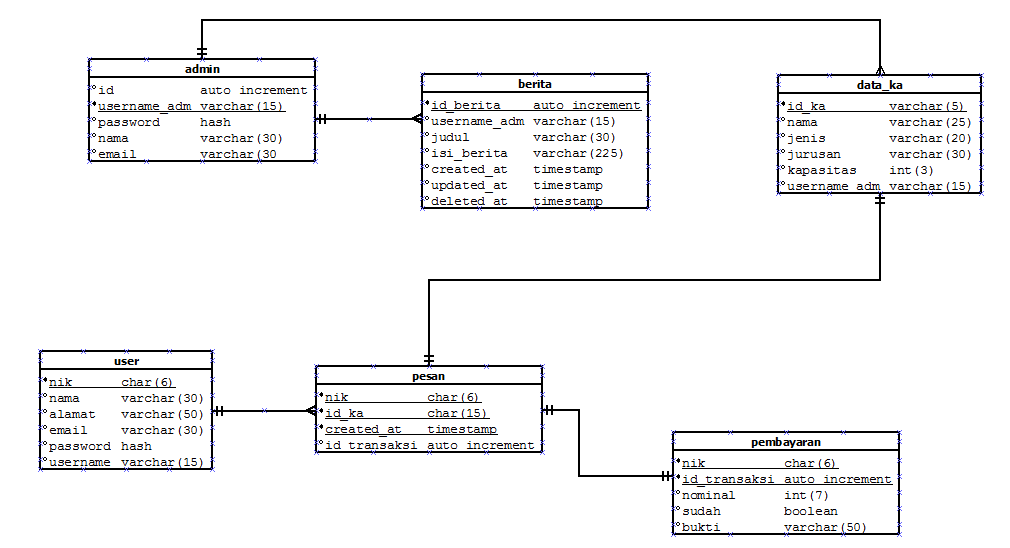
**Platform Requirements**

* Dapat berjalan dalam browser Google Chrome, Mozilla Firefox, dan Safari.
* Menggunakan Laravel
* Metode pengembangan software : Waterfall model
* Biaya: Pemesanan Server, biaya admin, CS, TS

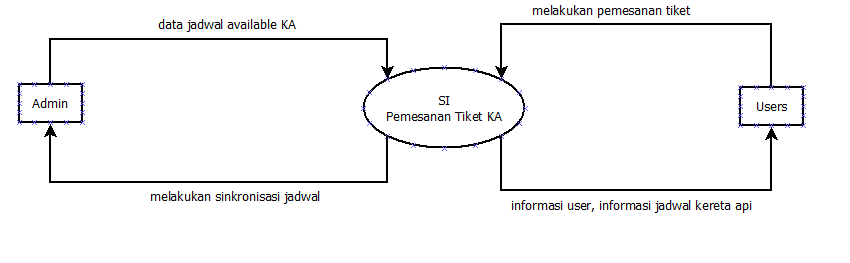
**Process Requirements**

* Registrasi
* Login
* Reservasi
* Cancelation
* Lihat Jadwal
* Cetak Tiket

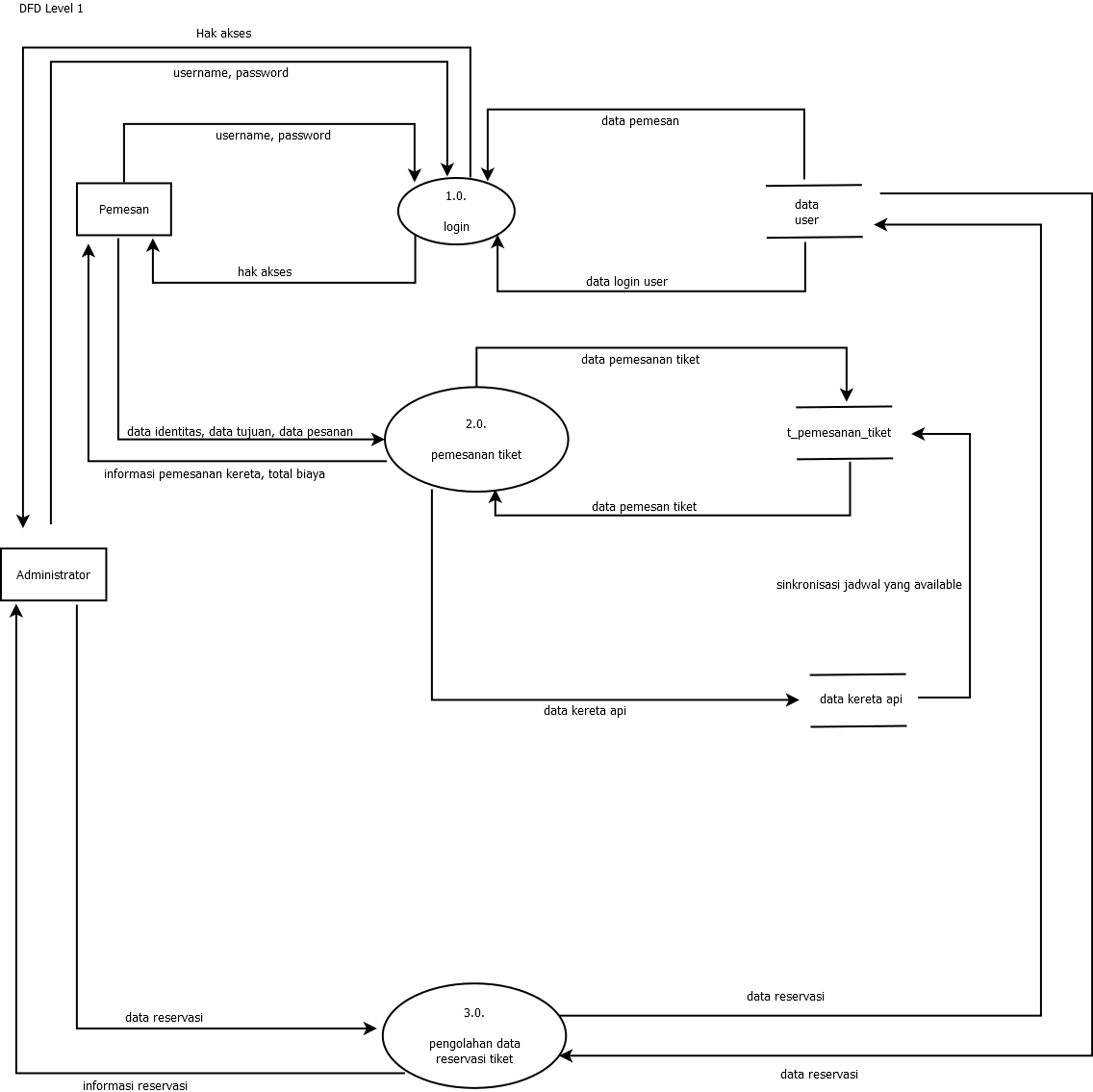
**ERD**



**Diagram Konteks**



**DFD**

****

Sistem yang akan dibuat: Sistem Reservasi Tiket Kereta Api

Kebutuhan :

* Server
* Aplikasi: Apache, MySQL

Entitas/pihak yang terlibat:

* Administrator
* User
* E-ticket

**Implementasi Sistem**

**­**Yang telah kami lakukan untuk membangun sistem ini adalah, diantaranya:

* Autentifikasi login user dan admin
* Membuat landing page sederhana
* Membuat tampilan admin
* Insert data kereta api
* Menampilkan data kereta api
* Membuat tampilan reserve

1. Autentifikasi login user dan admin

* Menghubungkan laravel dengan database, untuk menyimpan data-data login user dan admin.

Edit file .env yang ada di folder project. Edit bagian ini :

DB\_DATABASE=laravel

DB\_USERNAME=root

DB\_PASSWORD=

* Membuat auth dengan php artisan. Auth disini adalah bawaan laravel yaitu autentifikasi login untuk user saja. Jadi yang admin kita setting sendiri.

Jika sudah maka akan munculfile baru di controller, view, dan tambahan di file routes.

* Untuk membuat autentifikasi admin buat tabel baru admin dengan migration, dengan cara :

*php artisan make:migration create\_admins\_table —create=admins*.

* Buka file database/migrations/create\_admins\_table.php. Dan ganti data yang akan dibuat ditabel seperti berikut (sesuai kebutuhan).

public function up()

{

Schema::create('admins', function (Blueprint $table) {

$table->increments('id');

/\*tambahan kode untuk membuat tabel user\*/

$table->string('username\_adm')->unique();

$table->string('nama');

$table->string('email');

$table->string('password');

$table->rememberToken();

/\*\*/

$table->timestamps();

});

}

* Edit panjang default string di AppServiceProvider.php di folder app/Providers :

public function boot()

{

Schema::defaultStringLength(191);

}

* Buat model untuk tabel admins. Nama model biasanya adalah singular dari nama tabelnya.

Bisa duplicate dari User.php karena isinya hampir sama. Tinggal di ganti nama class sama isi dari tabel yang akan ditanngani model Admin.php ini.

protected $fillable = [

'username\_adm', 'nama', 'email', 'password',

];

* Atur konfirgurasi untuk auth. Di config/suth.php

providers' => [

'users' => [

'driver' => 'eloquent',

'model' => App\User::class,

],

'admins' => [

'driver' => 'eloquent',

'model' => App\Admin::class,

],

guards' => [

'web' => [

'driver' => 'session',

'provider' => 'users',

],

'api' => [

'driver' => 'token',

'provider' => 'users',

],

'admin' => [

'driver' => 'session',

'provider' => 'admins',

],

'admin-api' => [

'driver' => 'token',

'provider' => 'admins',

],

],

'passwords' => [

'users' => [

'provider' => 'users',

'table' => 'password\_resets',

'expire' => 60,

],

'admins' => [

'provider' => 'admins',

'table' => 'password\_resets',

'expire' => 15,

],

],

* Definisikan variabel guard di admin.php (admin model)

protected $guard = ‘admin’;

* Tambahkan routing untuk admin.

Route::get('/admin', 'AdminController@index')->name('admin.dashboard');

/admin = untuk url yang akan kita route menggunakan mode get

AdminController@index = mengacu ke controller AdminController di function index()

name(admin.dashboard) = alias dari route ang ktia buat

* Buat AdminController di folder app/Http/Controllers. Bisa menduplicate dari HomeController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AdminController extends Controller

{

/\*\*

\* Create a new controller instance.

\*

\* @return void

\*/

public function \_\_construct()

{

$this->middleware('auth:admin');

}

/\*\*

\* Show the application dashboard.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

return view('admin');

}

}

Auth:admin = digunakan untuk memvalidasi bahwa kalo bukan admin ngga akan bisa mengakses jika belum melakukan login

Admin = dia akan memanggil admin.blade.php untuk viewnya

* Admin.blade.php adalah template dari yang kita buat, akan dijelaskan selanjutnya.

**Login Multiple user**

* Buat controller baru bernama AdminLoginController di folder Auth dengan mengetikkan perintah: php artisan make: controller Auth/AdminLoginController
* Tambahkan function \_\_construct, showLoginForm, dan login

public function \_\_construct()

{

$this->middleware('guest:admin');

}

function showLoginForm()

{

return view('auth.admin-login');

}

function login(Request $request)

{

// Validate the form data

$this->validate($request, [

'email' => 'required|email',

'password' => 'required|min:6',

]);

// Attempt to log the user in

if (Auth::guard('admin')->attempt(['email' => $request->email, 'password' => $request->password],

$request->remember)) {

// If successful, then redirect to their intended location

return redirect()->intended(route('admin.dashboard'));

}

// If unsuccessful, then redirect back to the login with the form data

redirect()->back()->withInput($request->only('email', 'remember'));

}

* Buat admin-login.blade.php di folder resources/views/auth

<div class="panel-heading">ADMIN Login</div>

<div class="panel-body">

<form class="form-horizontal" method="POST" action="{{ route('admin.login.submit') }}">

{{ csrf\_field() }}

* Tambahkan routing di web.php

Route::get('/admin/login', 'Auth\AdminLoginController@showLoginForm')->name('admin.login');

Route::post('/admin/login', 'Auth\AdminLoginController@login')->name('admin.login.submit');

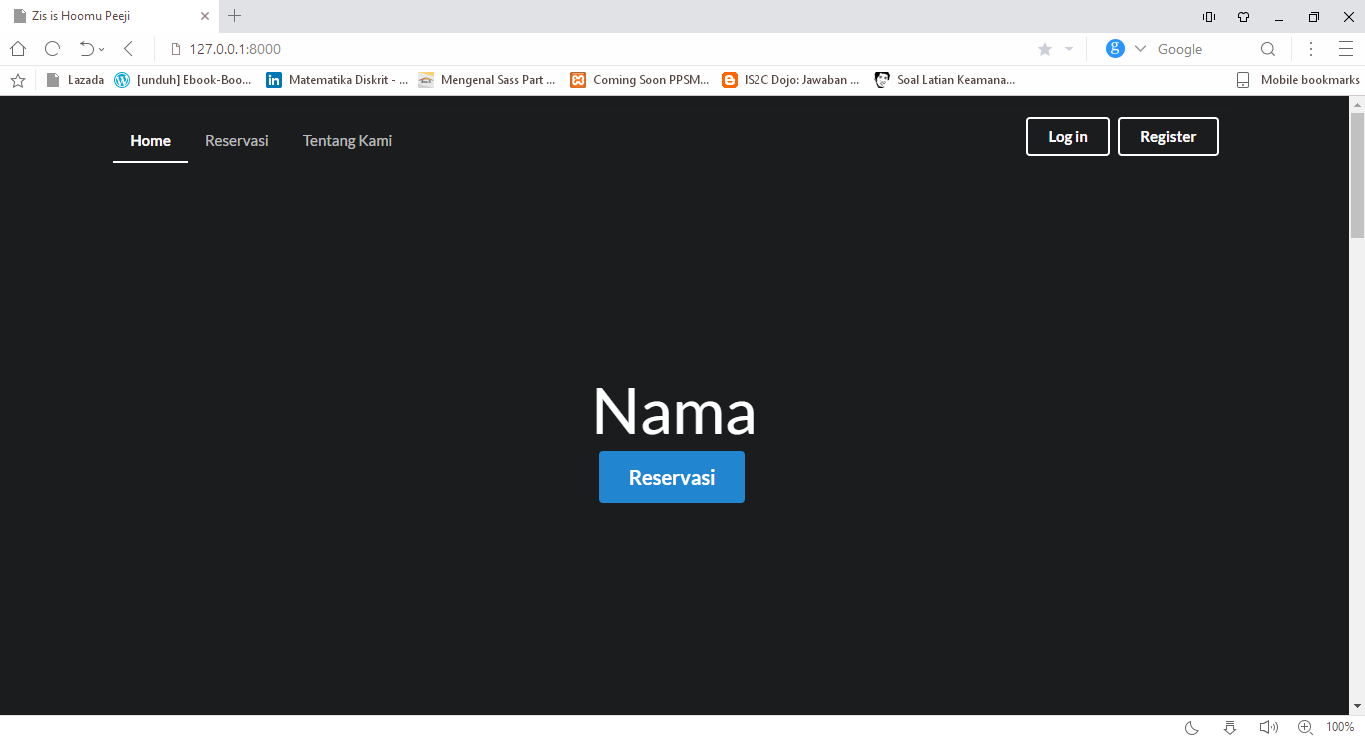
* Isikan data admin dari php artisan tinker

NB :

Username admin : adminirfan

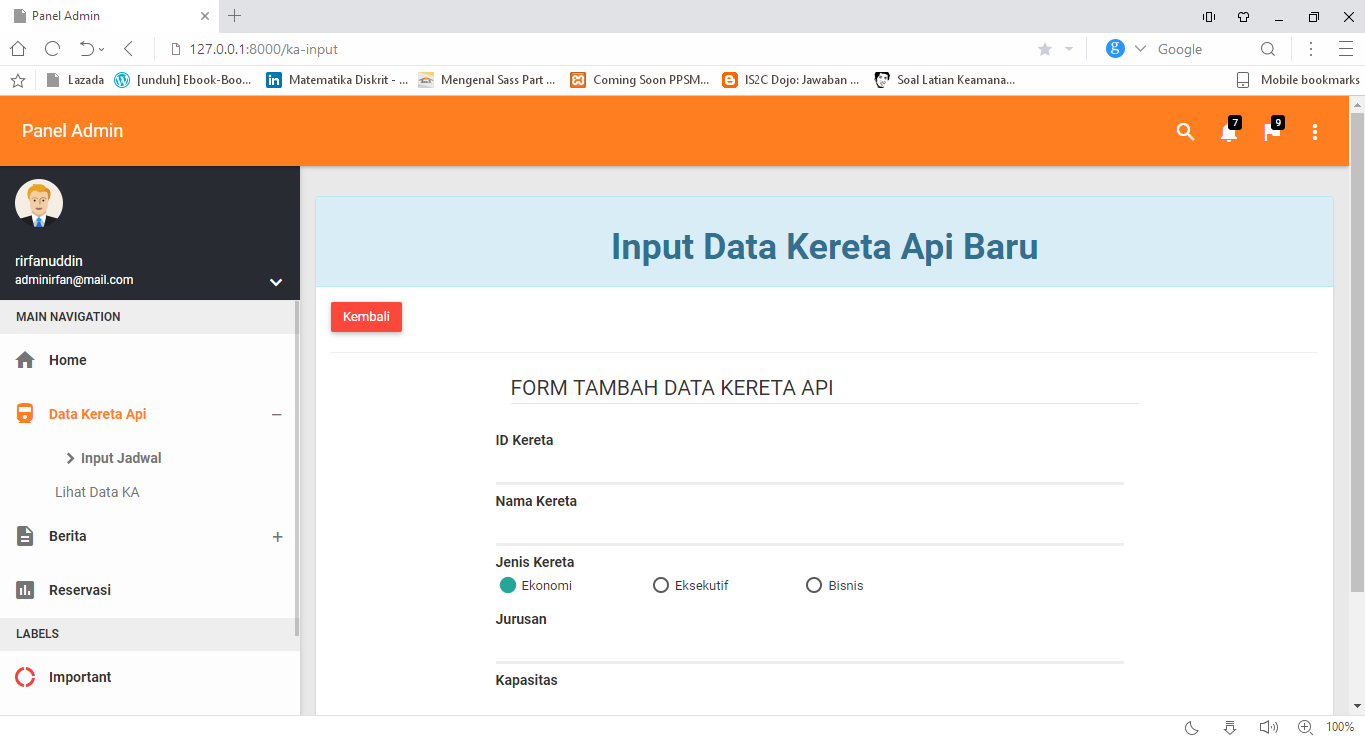
Password admin : adminirfan

1. Membuat landing page sederhana



1. Templating halaman admin.

* Kita cari dulu template yang sesuai untuk admin. Kita ekstract dan taruh di /public/adm
* Kemudian di admin.blade.php itu kita extend ke master-admin.blade.php yang berada di folder layouts. Sehingga sedemikian hingga bisa dilihat di koding di master.blade.php, admin.blade.phpnya, tetapi file resource css, js, dll tetap berada di folder /public/adm.



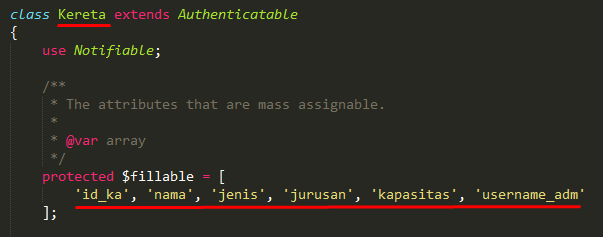
* Keterangan tambahan, di templating ini, terdapat beberapa kata kunci:

@yield(‘namanya’) = halaman tersebut bisa disisipi halaman dari halaman anak

@section(‘namanya’) = halaman anak yang biasanya akan merubah yield dari sang induk

1. Insert data kereta api

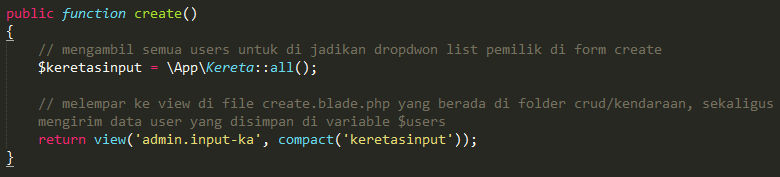
* Membuat model untuk tabel keretas yaitu bernama Kereta.php di folder app



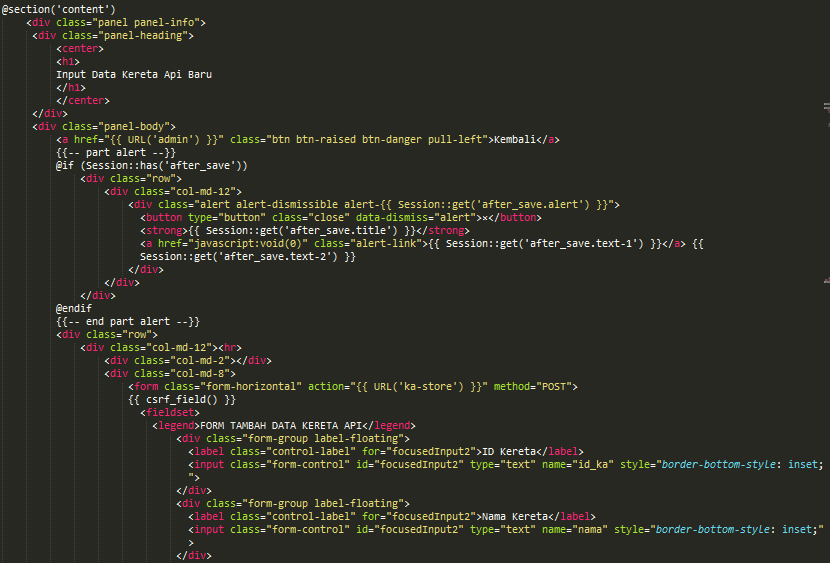
* Membuat route di web.php menggunaka metode get

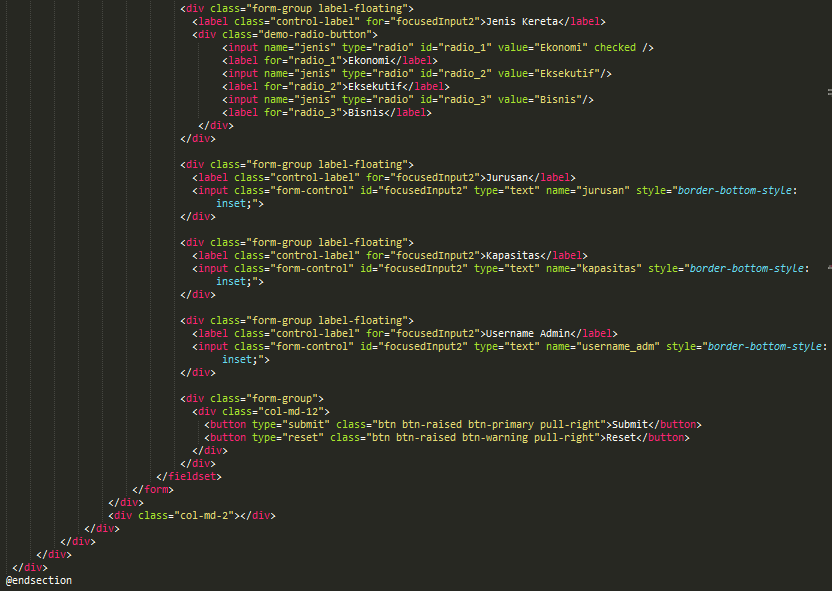


* Kemudian di bagian admin controller buat method create()

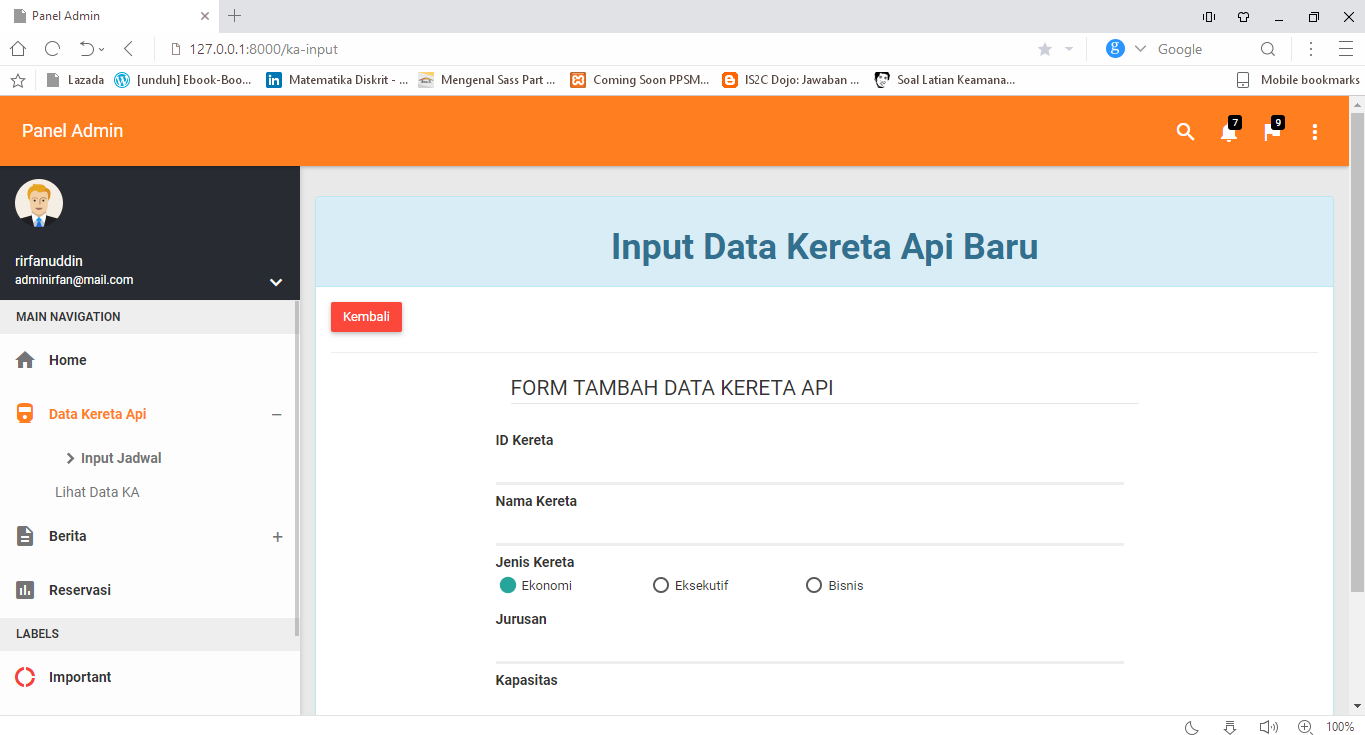


* Return view, kembalikan ke input-ka.blade.php di folder admin. Buat filnya, dan isinya seperti berikut





* Tampilan akhir

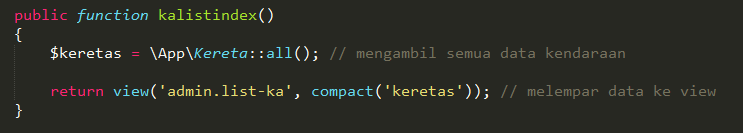


1. Menampilkan data kereta api

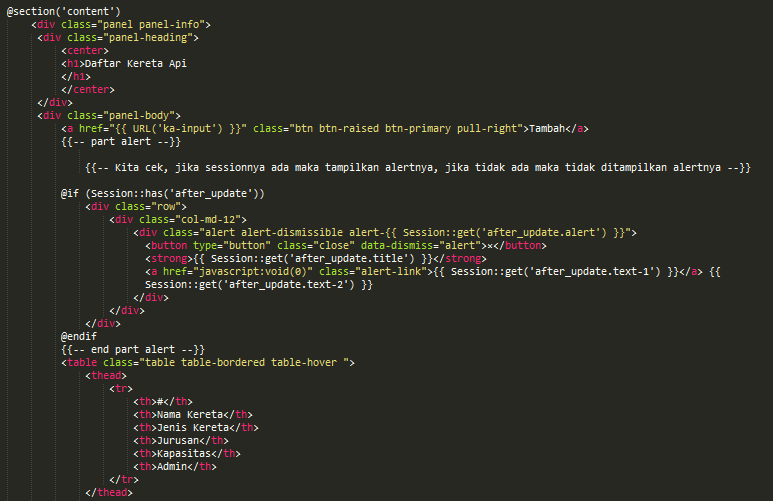
* Buat route di web.php



* Buat function kalistindex() di AdminController.php

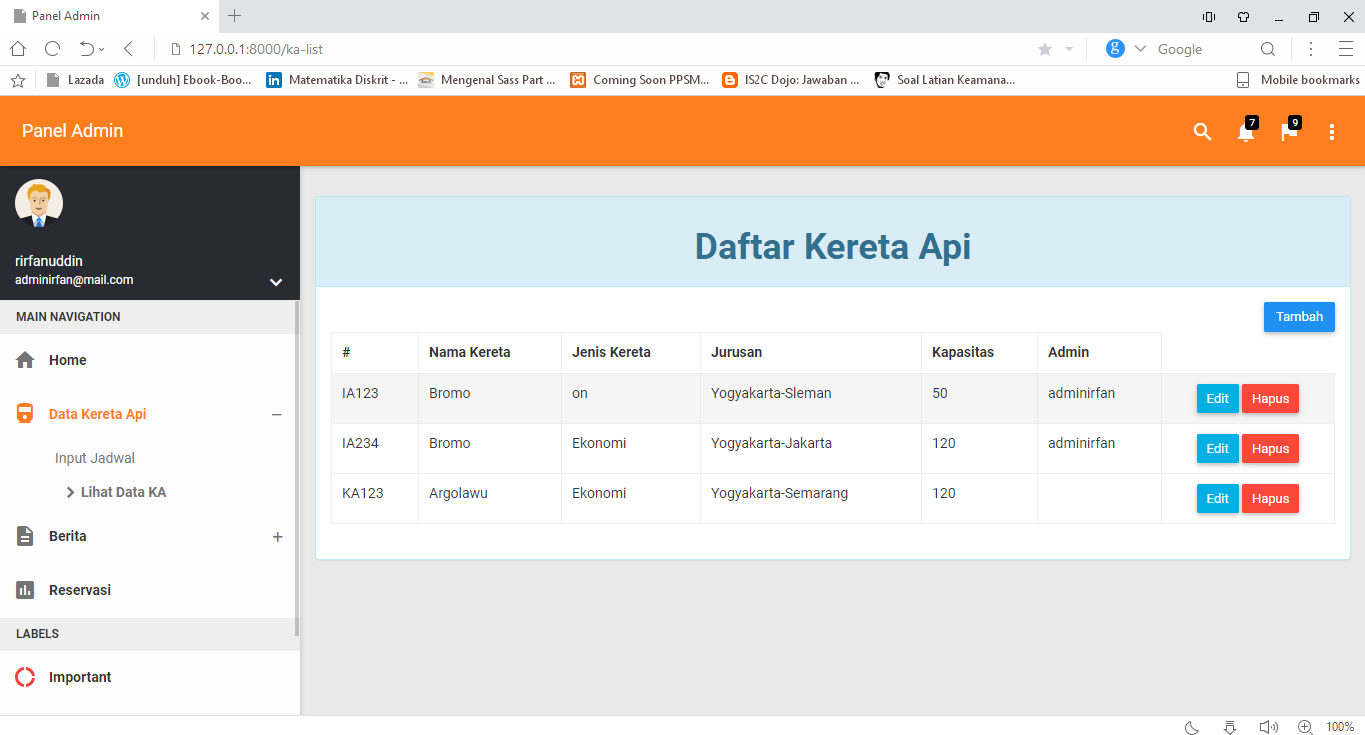


* Buat view list-ka.blade.php

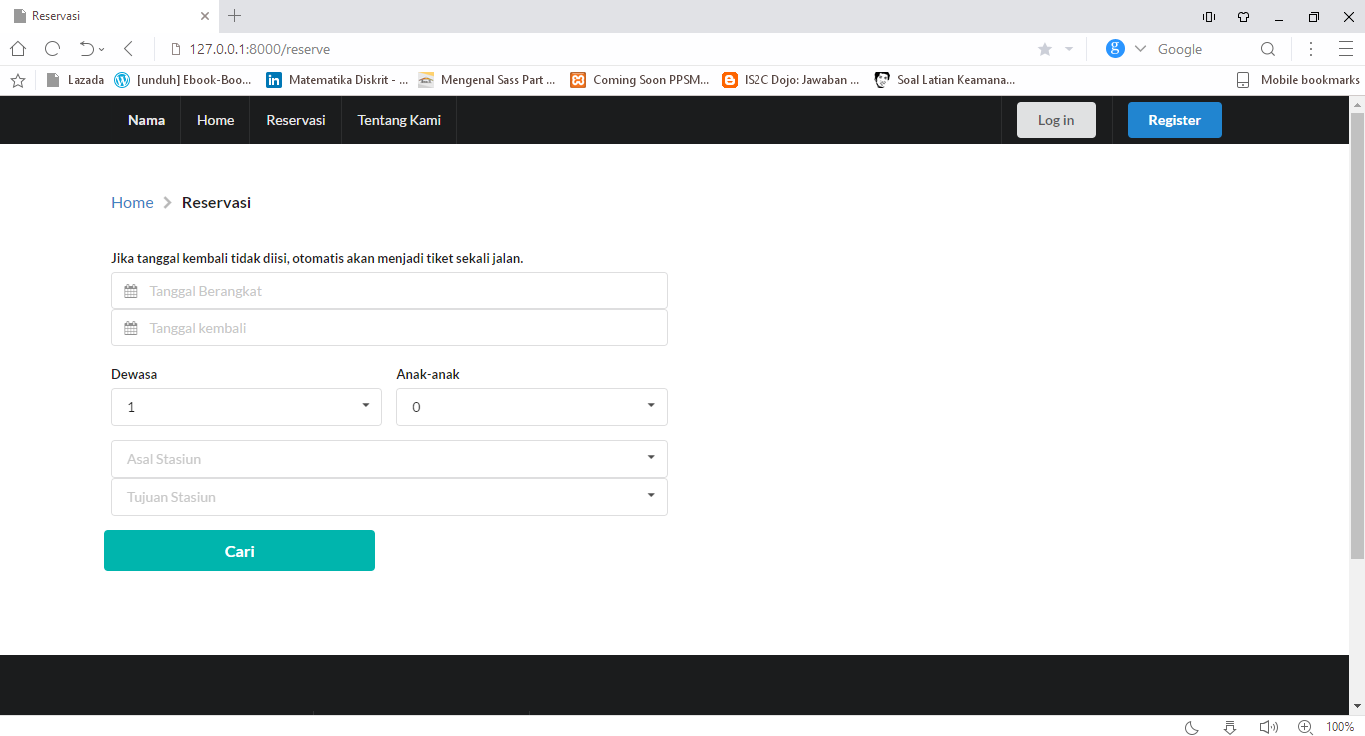




* Tampilan akhir



1. Membuat tampilan reserve

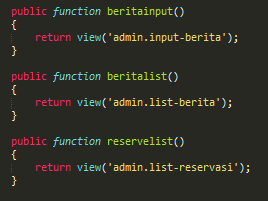


1. Agar halaman tertentu hanya bisa diakses jika sudah login sebagai admin

* Di route di web.php buat route:



* Di controller AdminController di function beritainput(), beritalist(), dll buat seperti ini



Karena diatasnya terdapat method \_\_construct() jadi dia bisa melakukan autentifikasi apakah user biasa, guest atau admin