Visoka škola strukovnih studija za
informacione i komunikacione tehnologije

Car Finder

http://carfinderlazic.online/

Smer: Internet tehnologije

Modul: Web programiranje

Predmet: Web Programiranje PHP 2

Nenad Lazić 9/19

Sadržaj

Contents

1. Uvod

1.1 Korišćeni programski jezici/tehnologije

Za potrebe sajta korišćene su sledeće tehnologije:

- HTML
- CSS
- Javascript (ES6)
- Laravel 9 (PHP Framework)
- Artisan CLI
- Bootstrap 5 / 4
- Online templates
- Intervention Image za resize slike

Bootstrap premium template je korišćen.

1.2 Opis funkcionalnosti

Aplikacija je pravljena po uzoru na polovne automobile i poenta sajt je da razliciti korisnici imaju razlicite poglede na aplikaciju kao i da administratorski korisnici mogu da upraljaju entitetima na serveru odnosno u ovom slucaju automobilima, korisnicima, lokacijama i svim ostalim stvarima koje su neophodne za funkcionisanje aplikacije.
Celokupna baza sa svim podacima je napravljena kroz migracije i seedere.

Funkcionalnosti sajta su sledeće:

- Navigacioni meni kao i dugmice iz navigacije korisnici u zavisnosti od toga da li su ulogovani ili ne drugacije vide, kao i u zavisnosti od uloge (korisnik kada je ulogovan ima na desnoj strani navigaije dugme “+Sell Car” a kada je korisnik ulogovan i u ulozi admina je postoji dugme Admin koji vodi na rute zaduzene za admin korisnika).

- Korišćen je layout kako za front tako i za back aplikacije i rasporedjen je u posebne foldere unutar foldera resources/views

- Kada je korisnik ulogovan, moze videti svoju sliku u desnom delu navigaije (Levo od sell car dugmeta) i slika ce ili biti default.jpg ili slika koju korisnik ubaciti odlaskom na svoj profil.

- Prelaskom misa preko profilne slike, otvara se padajuci meni gde se moze otici na svoj profil ili direktno izlogovati.

Ruta “/”

- Na početnoj stranici u sekciji Top offers imamo izlistana 3 automobila iz baze podataka sa svim njihovim podacima. Svi podaci se dohvataju iz baze podataka kao i sve slike koje se skladište u storage folderu koji se ne nalazi direktno u public folderu vec u public folderu postoji soft link koji je povezan sa storage-om. Klikom na dugme view all offers -> bicete redirektovani na resource rutu cars.index (/cars), gde se nalazi ponuda svih automobile zakljucno sa filtriranjem i sortiranjem koje se aktivira klikom na dugme Filter Cars.

- Na dnu se nalazi sekcija koja unutar slidera izlistava automobile po odredenom kritererijumu i prikazuje automobile kao stavke slidera.

Ruta (/cars) - Resource ruta

- Otvaranjem ove rute otvara se index metod koji izlistava sve automobile iz baze podataka sa svim podacima. Za prikazivanje 1 automobile koriscena je Componenta, koja ja napravljena kroz Artisan CLI.

- Paginacija je prisutna i radi uskladjeno sa filtriranjem i sortiranjem.

- Svi elementi forme za filtriranje su dinamicki popunjeni podacima iz baze podataka.

- Ukoliko je korisnik admin, administratorske privilegije su te da pored svakog automobila (Komponente) postoji dugme Delete kojim se ajaxom salje delete zahtev koji brise entitet sa servera, izbacuje se poruka o uspesnom brisanju i izbrisani automobil se brise sa stranice bez reload-a.

- Logika je da admin korisnik moze sve automobile da brise ali ne moze da edituje tudje automobile. Korisnik koji je postavio oglas za automobil moze da brise i menja iskljucivo svoj automobil. Pokusajem da se edituje tudji automobil kroz url adresu (/cars/2/edit) korisnik ce biti redirektovan na rutu /cars sa porukom da ne moze da edituje automobil koji nije njegov i bice evidentiran u Log fajl kao warning poruka da je pokusao da edituje automobil koji nije njegov.

- Drop down meni za mark i model automobila je odradjen koriscenjem ajaxa gde se na metod change dohvataju modeli iskljucivo za tu konkretnu marku automobila.

Ruta (/cars/{id}) show metod

- Otvara se stranica za prikaz pojedinacog automobila.

- Otvaraju se sve slike automobila koje su uploadovane na server kao i svi podaci o tom automobilu i njegovom vlasniku.

- Ispod svi infomacija o pojedinacnom vozilu se nalazi sekcija gde se ucitavaju automobili po redosledu (cena opadajuce), 4 najskuplja automobila (Sekcija – You may be interested in)

Ruta (/cars/create)

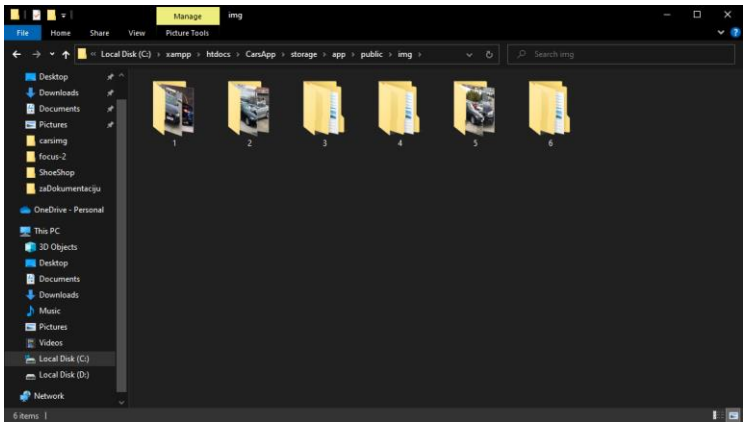
- Zasticena ruta koja je moguca iskljucivo ulogovanim korisnicima

- Otvara se forma za unos novog automobila.

- Polja su validirana kreiranjem requesta kroz Artisan CLI i postavljanje pravila u funkciju rules i ispisivanje custom poruka kroz funkciju messages

- Moguce je ubaciti vise fajlova odjednom. Svi fajlovi ce biti obradjeni u okviru metode

Store, gde ce biti napravljen folder koji ce imati naziv po id-u korisnika kako bi bio jedinstveno identifikovan unutar projekta i unutar svih slika. U okviru iste metode ce se napraviti podfolder “resized” gde ce biti skladistene resizovane slike koje ce biti kasnije koriscene u okviru admin panela. U bazi se cuva iskljucivo ime slike.



- Nakon uspesno ubacenog automobila u bazu podataka korisnik biva redirektovan sa porukom o uspesno postavljenom oglasu i oglas postaje javno dostupan na (/cars). Pre redirekcuje u Log fajl se upisuje poruka o uspesnosti unosenja novog entiteta.

- U slucaju greske ili nemogucnosti da se oglas ubaci u bazu podataka, greska se upisuje u Log fajl i korisnik dobija poruku da je u tom trenutku nemoguce ubaciti oglas i dobija jasnu poruku kao i kod uspesnog ubacivanja oglasa odnosno automobila u bazu podataka.

Ruta (/cars/{id}/edit)

- Samo autorizovan korisnik moze koristiti ovu rutu i to iskljucivo u slucaju da je on postavio taj automobil koji se pokusava editovati. U suprotnom u Log fajl se upisuje ip adresa korisnika koji je pokusao da edituje automobil koji nije njegov i dobija jasnu poruku da ne moze editovati automobil koji nije njegov.

- Kada korisnik uspesno pogodi svoj automobil, otvara se forma za edit koja je popunjena podacima o tom konkretnom automobilu koji se zeli promeniti. Ajaxom je popunjena marka i model automobila. I dalje postoji metod change da se ajaxom dohvataju iskljucivo modeli konkretno selektovane marke automobila.

- Validacija je odradjena Requestom
- Sve slike tog konkretnog automobila koji se edituje su ucitane i postoji dugme delete kojim se slika uklanja sa servera. Ova funkcionalnost se realizuje ajaxom, i korisnik dobija obavestenje da je uspesno izbrisao sliku sa servera. Kada se klikne na delete prvo se dobije popup da se potvrdi brisanje i samo u slucaju da se klikne ok, slika se trajno brise sa servera. Takodje korisnik moze da doda jos slika (takodje vise odjednom je moguce kao i prilikom inicjalnog kreiranja automobila) i sve slike ce biti skladistene na serveru.
- Ukoliko korisnik ciji je auto udje na metod show svog automobila imace ispisana 2 dugmeta (Edit i delete) koji takodje rade.

Ruta (/users)

- Odlaskom na ovu rutu odlazi se na stranicu o podacima o korisniku.
- Ispisani su podaci o korisniku, njegov broj, email, slika, ime, prezime sa leve strane gde postoje i tabovi.
- Tabovi su realizovani javascriptom
- Tab Personal info sadrzi podatke o korisniku i tu korisnik moze da ubaci svoju profilnu sliku. Da izmeni licne podatke. Nakon promena klikom na dugme save changes korisnik dobija poruku o uspesnoj izmeni svojih licnih podataka koji se u istom trenutku mogu videti.
- Tab password and security sluzi da korisnik promeni svoju lozinku. Unese se trenutna sifra i nova sifra na 2 mesta. U slucaju uspesnog menjanja se dobija poruka o uspesnoj promeni.
- Tab my cars sadrzi prikaz svih automobila koje je korisnik ubacio u aplikaciju. Klikom na 3 tacke je moguce editovati ili izmeti automobil.
- Tab Sign out sluzi da se korisnik izloguje.

Ruta za admine (/admin/index)

- Ruta koja je zasticena i iskljucivo administratori joj mogu pristupiti.
- Gadjanjem rute /admin se vrsi redirekcija na /admin/index
- /admin/index ima podatke o totalnom broju korisnika koji koriste aplikaciju, broj automobila koji su na prodaju u okviru aplikacije i sumi cena svih automobila.
- Koriscen je layout.
- Na desnoj strani header sekcije se nalazi ime i prezime korisnika kao i njegova slika ukoliko je ima, ukoliko nema, svi korisnici imaju defaultnu sliku.
- Sa leve strane se nalazi navigacioni meni
- U tabu main menu->user imaju linkovi home i sign out gde se admin moze vratiti na aplikaciju ili se izlogovati
- Tab tables sluzi za upravljanje tabelama u bazi podataka.

Admin->table->cars (Resource ruta u okviru admina)

- Postoji dugme za povratak na prethodnu stranicu
- Extenduje se layout i samo sredisnji deo se menja
- Postoji polje za pretragu koje radi uskladjeno sa pagijnacijom
- Svi automobili su prikazani tabelarno sa svojim najbitnijim podacima i slikom na pocetku. Klikom na sliku se u novom prozoru otvara pojedinacni prikaz proizvoda u aplikaciji
- Klikom na dugme Add new car se otvara forma za ubacivanje novog automobila, gde se takodje moze ubaciti vise slika odjednom.
- Forma je validirana Requestom
- Poslednje 2 kolone tabele su edit i delete koji takodje rade i vracaju poruku o uspesnoj odnosno neuspesnoj funkciji

- Klikom na edit se otvara forma za edit automobila koja radi slicno kao i edit u okviru aplikacije.
- Klikom na delete se dobija poruka da se potvrdi brisanjem i potvrdom se entitet brise iz baze podataka.

Admin->table->users

- Extenduje layout i otvara slican interfejs kao i admin->cars gde se tabelarno prikazuju svi korisnici, postoji paginacija i uskladjena je sa pretragom. Dugme za Add user otvara formu gde se dodaje novi user i koje je jedino mesto u aplikaciji gde se moze dodati novi user sa administratorskim privilegijama. Delete radi jedino u sluicaju da taj korisnik nema automobile koji su postavljeni. Edit funkcionalnost takodje radi i otvara formu koja je popunjena podacima o korisniku koji treba da se edituje.

Admin->table->Locations

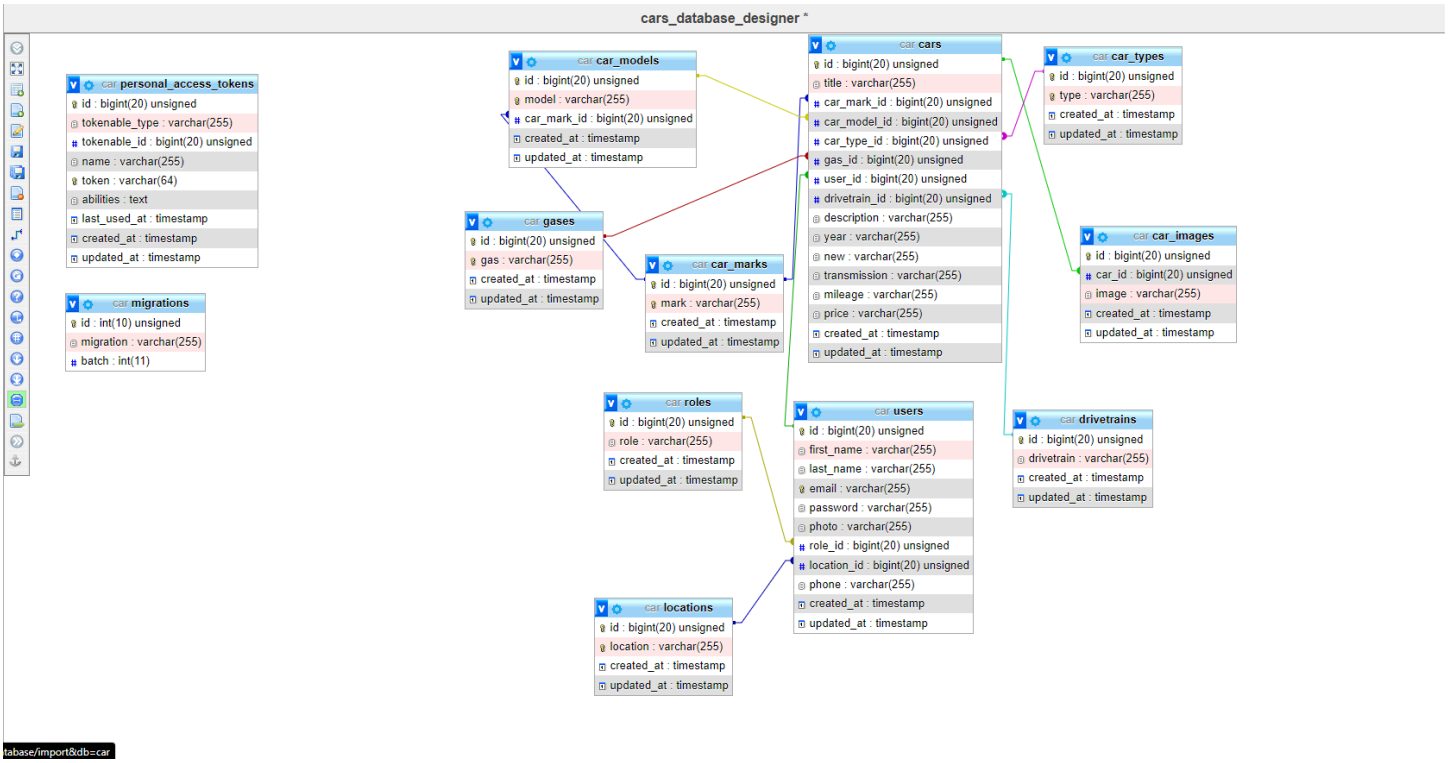
- Extenduje layout i ima iste funkcionalnosti kao i users i cars.

1.3 Template

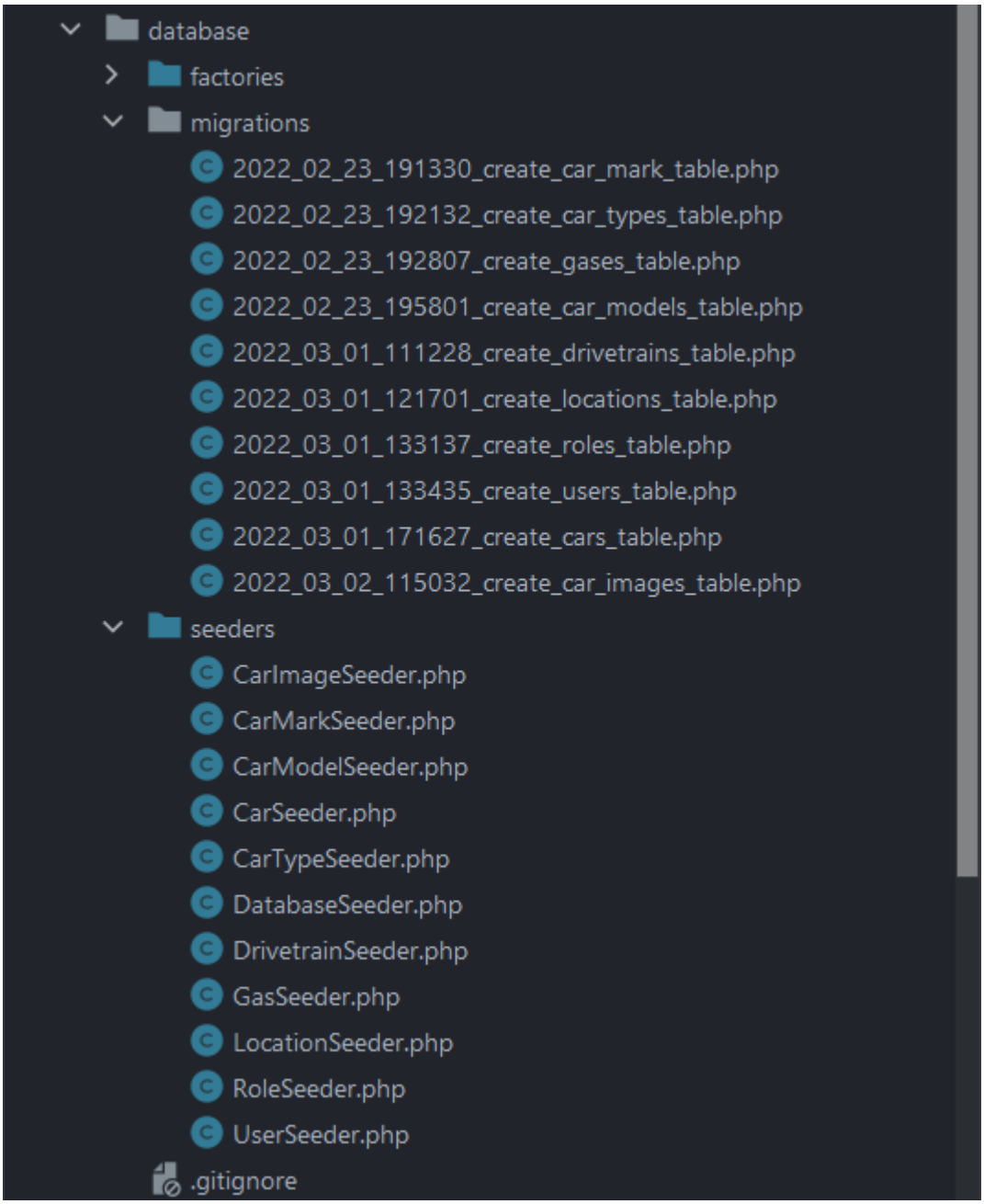
Za izradu sajt jeste korišćen template. Premium boostrap templates

2.0 Baza Podataka i migracije i seederi

2.1 Designer

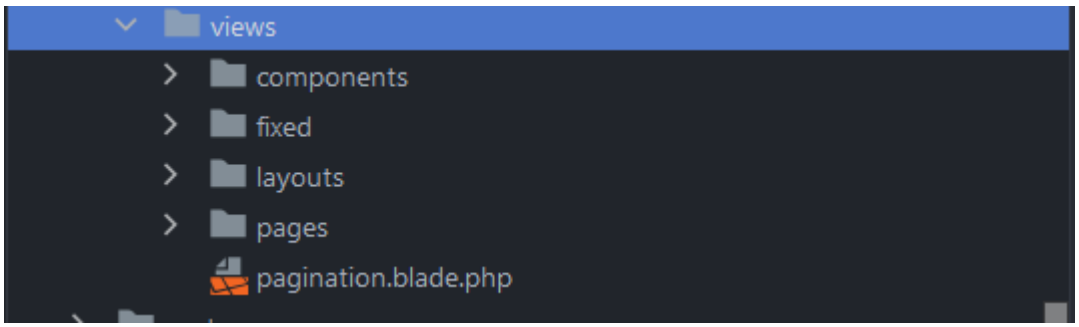


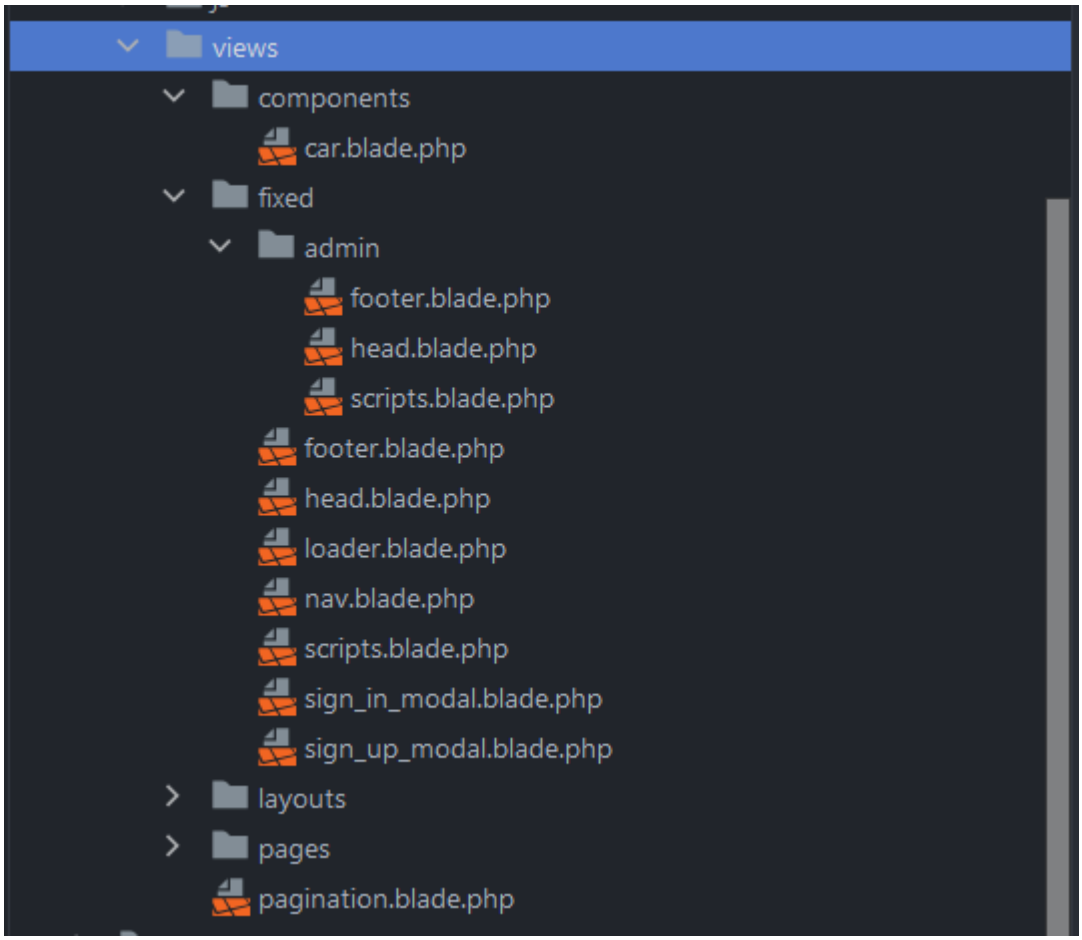
2.2 Migrations and seeders

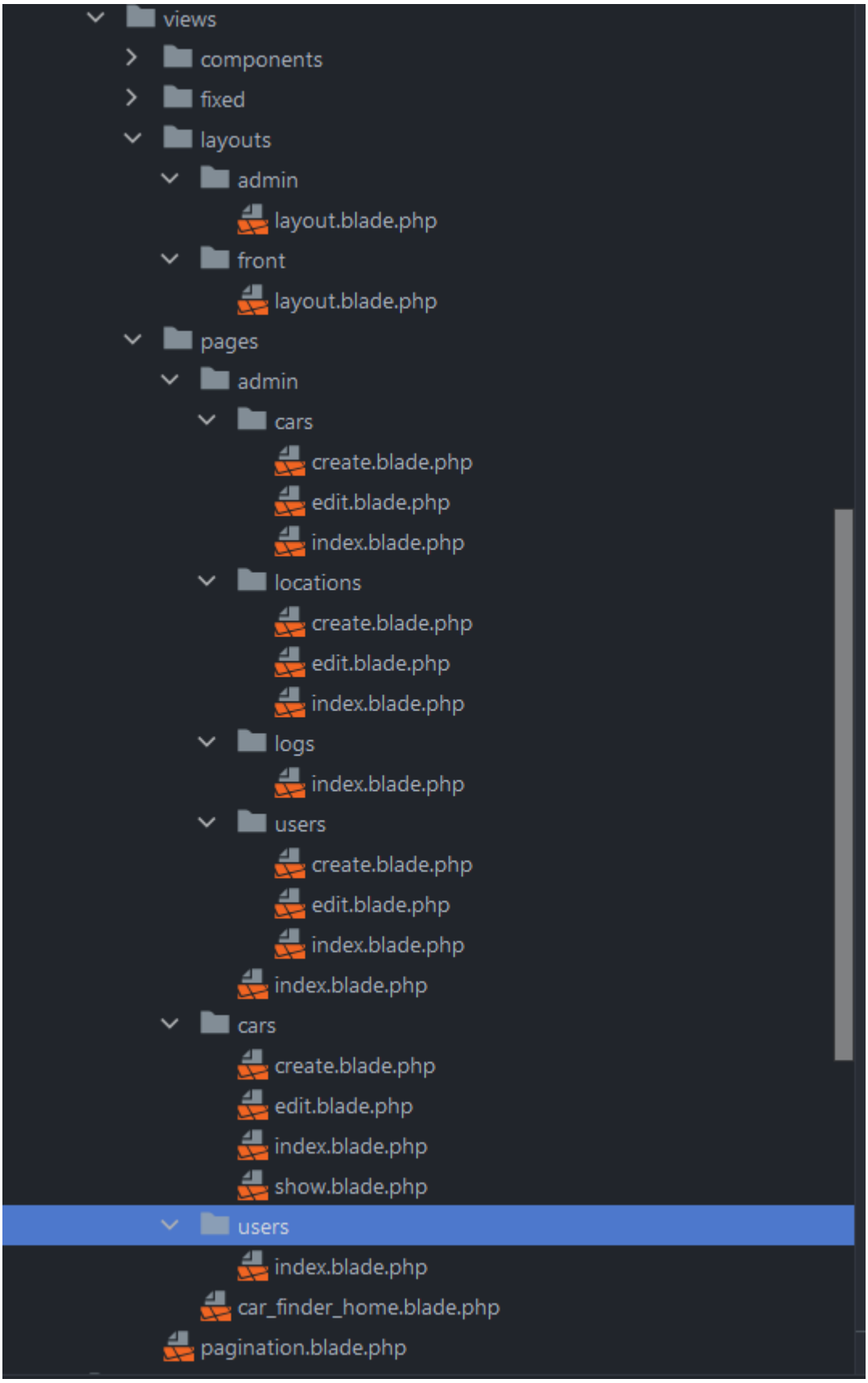


3.0 Organizacija views foldera

3.1 Organizacija




views
components
car.blade.php
fixed
admin
footer.blade.php
head.blade.php
scripts.blade.php
footer.blade.php
head.blade.php
loader.blade.php
nav.blade.php
scripts.blade.php
sign_in_modal.blade.php
sign_up_modal.blade.php
layouts
pages
pagination.blade.php


views
components
fixed
layouts
admin
layout.blade.php
front
layout.blade.php
pages
admin
cars
create.blade.php
edit.blade.php
index.blade.php
locations
create.blade.php
edit.blade.php
index.blade.php
logs
index.blade.php
users
create.blade.php
edit.blade.php
index.blade.php
index.blade.php
cars
create.blade.php
edit.blade.php
index.blade.php
show.blade.php
users
index.blade.php
car_finder_home.blade.php
pagination.blade.php

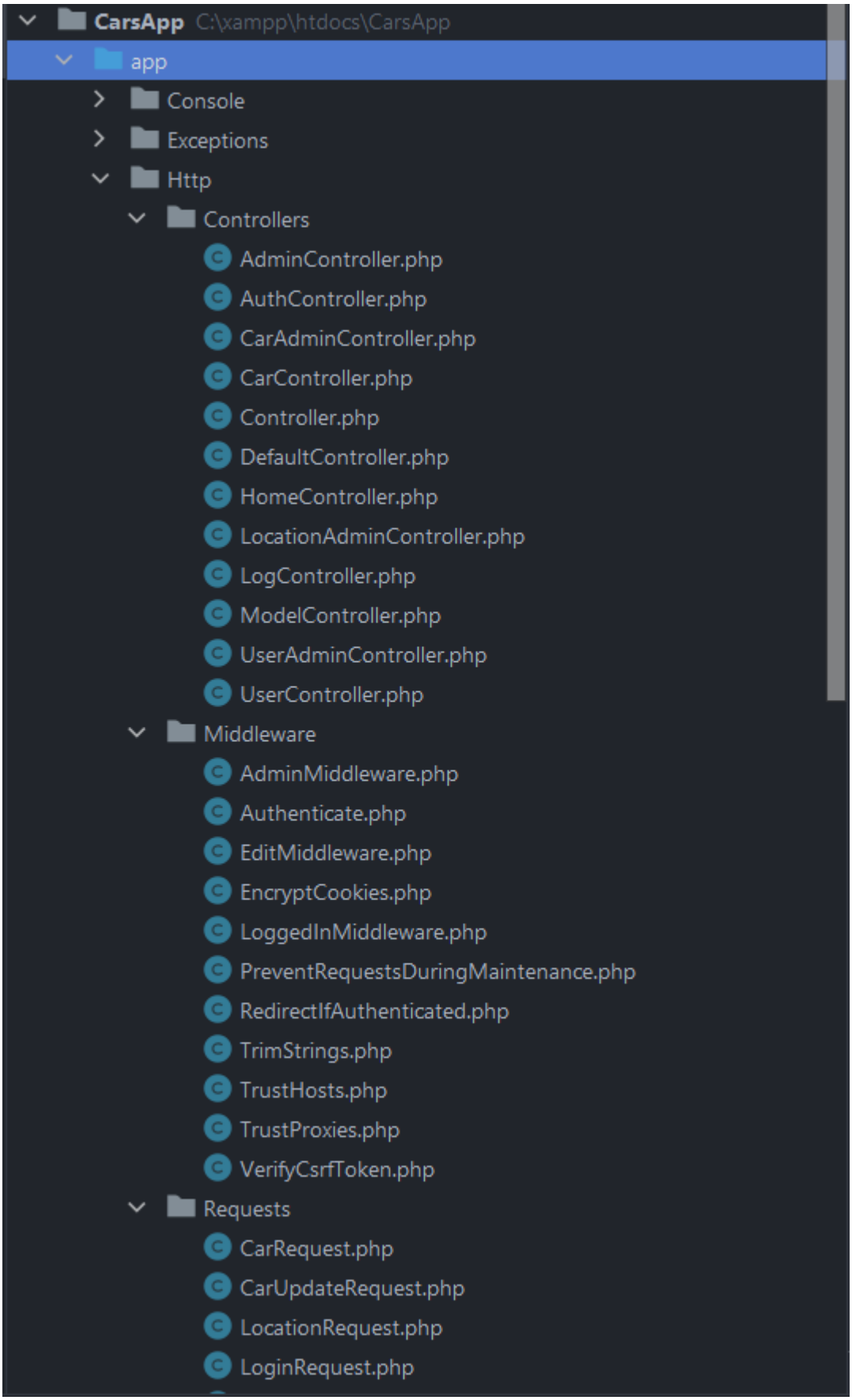
4.0 Routes – web.php

```
<?php

use Illuminate\Support\Facades\Route;
use \App\Http\Controllers\AuthController;
use \App\Http\Controllers\HomeController;
use \App\Http\Controllers\CarController;
use \App\Http\Controllers\UserController;
use \App\Http\Controllers\AdminController;
use \App\Http\Controllers\LogController;
use \App\Http\Controllers\CarAdminController;
use \App\Http\Controllers\UserAdminController;
use \App\Http\Controllers\LocationAdminController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/
Route::get('/', [HomeController::class, 'index'])->name('home');

Route::middleware(['loggedIn'])->group(function() {
    Route::middleware(['admin'])->group(function(){
        Route::redirect('/admin', '/admin/index');
        Route::prefix('admin')->group(function() {

            Route::resource('adminCar', CarAdminController::class);

            Route::resource('adminUser', UserAdminController::class);

            Route::resource('location', LocationAdminController::class);

            Route::get('/index', [AdminController::class, 'index'])->name('admin.index');

            Route::get('/log', [LogController::class, 'index'])->name('admin.log');
        });
    });

    Route::resource('cars', CarController::class)->only([
        'create', 'edit'
    ]);

    Route::delete('/cars/images/{id}', [CarController::class, 'deleteImage']);

    Route::put('/users/password/{id}', [UserController::class, 'updatePassword'])-
>name('users.updatePassword');

    Route::resource('users', UserController::class);

});

Route::resource('cars', CarController::class)->except([
```

```
    'create', 'edit'
]);

Route::post('/register', [AuthController::class, 'register'])->name('register');

Route::post('/login', [AuthController::class, 'login'])->name('login');

Route::get('/signout', [AuthController::class, 'signout'])->name('signout');

Route::get('/models/{id}', [\App\Http\Controllers\ModelController::class, 'getModels'] );
```

5.0 Controllers

5.1 Slika organizacije kontrolera, middleware, requests


CarsApp C:\xampp\htdocs\CarsApp
app
Console
Exceptions
Http
Controllers
AdminController.php
AuthController.php
CarAdminController.php
CarController.php
Controller.php
DefaultController.php
HomeController.php
LocationAdminController.php
LogController.php
ModelController.php
UserAdminController.php
UserController.php
Middleware
AdminMiddleware.php
Authenticate.php
EditMiddleware.php
EncryptCookies.php
LoggedInMiddleware.php
PreventRequestsDuringMaintenance.php
RedirectIfAuthenticated.php
TrimStrings.php
TrustHosts.php
TrustProxies.php
VerifyCsrfToken.php
Requests
CarRequest.php
CarUpdateRequest.php
LocationRequest.php
LoginRequest.php

5.2 Admin Controller

```php
<?php

namespace App\Http\Controllers;

use App\Models\Car;
use Illuminate\Http\Request;

class AdminController extends DefaultController
{
    public function index(){
        return view('pages.admin.index');
    }


    public function cars(Request $request){
        $cars = new Car();

        if($request->has('keywords') && $request->get('keywords') != null){
            $keywords = $request->get('keywords');
            $cars = $cars->where(function ($query) use($keywords) {
                $query->orWhere('title', 'Like', '%' . $keywords . '%');
                $query->orWhere('description', 'Like', '%' . $keywords . '%');
            });
        }

        return view("pages.admin.cars.index", [
            "cars" => $cars->get()
        ]);
    }
}
```

5.3 Auth Controller

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\LoginRequest;
use App\Http\Requests\RegisterInModalRequest;
use App\Http\Requests\RegisterRequest;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Storage;

class AuthController extends Controller
{
    public function login(LoginRequest $request)
    {
        $user = User::where([
            ['email', '=', $request->get('email')],
            ['password', '=', md5($request->get('password'))]
        ])->first();

        if ($user !== null) {
            $request->session()->put('user', $user);
```

```
            return redirect()->route('cars.index');
        } else {
            Session::flash('#signup-modal');
            return redirect()->back();
        }
    }

    public function register(RegisterInModalRequest $request)
    {
        try {
            $user = new User();

            $user->first_name = $request->get('first_name');
            $user->last_name = $request->get('last_name');
            $user->phone = $request->get('phone');
            $user->email = $request->get('email');
            $user->password = md5($request->get('password'));
            $user->location_id = $request->get('location');
            $user->role_id = 2;

            $user->save();
            $lastId = $user->id;

            if ($user) {
                Storage::makeDirectory('public/img/' . $lastId . '/profile/');
                $user = User::find($user->id);
                $request->session()->put('user', $user);
                Log::info($request->ip() . ' registered to our application!');
                return redirect()->route('cars.index');
            }
        }catch (\Exception $exception){
            Log::error($request->ip() . ' failed registraction. Message: ' . $exception);
            return redirect()->route('home');
        }

    }

    public function signout(Request $request)
    {
        $request->session()->forget('user');

        return redirect()->route('home');
    }
}
```

5.4 CarAdminController

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\CarRequest;
use App\Http\Requests\CarUpdateRequest;
use App\Models\Car;
use App\Models\CarImage;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;
```

```php
use Image;

class CarAdminController extends DefaultController
{

    public function index(Request $request)
    {
        $cars = new Car();

        if($request->has('keywords') && $request->get('keywords') != null){
            $keywords = $request->get('keywords');
            $cars = $cars->where(function ($query) use($keywords) {
                $query->orWhere('title', 'Like', '%' . $keywords . '%');
                $query->orWhere('description', 'Like', '%' . $keywords . '%');
            });
            $cars = $cars->whereHas('car_mark', function ($query) use($keywords) {
               return $query->orWhere('mark', 'Like', '%' . $keywords . '%');
            });
        }

        return view("pages.admin.cars.index", [
            "cars" => $cars->paginate(2)
        ]);
    }


    public function create()
    {
        return view('pages.admin.cars.create');
    }


    public function store(CarRequest $request)
    {
        try {
            DB::beginTransaction();
            $car = new Car();

            $car->title = $request->get('title');
            $car->car_mark_id = $request->get('car_mark');
            $car->car_model_id = $request->get('model');
            $car->car_type_id = $request->get('car_type');
            $car->gas_id = $request->get('gas');
            $car->user_id = $request->session()->get('user')->id;
            $car->drivetrain_id = $request->get('drivetrain');
            $car->description = $request->get('description');
            $car->year = $request->get('year');
            $car->new = $request->get('condition');
            $car->transmission = $request->get('transmission');
            $car->mileage = $request->get('mileage');
            $car->price = $request->get('price');

            $car->save();
            $lastId = $car->id;

            if ($request->file('files') != null) {
                foreach ($request->file('files') as $file) {
                    $carImage = new CarImage();
                    $image = $file;
```

```
                    $originalName = $image->getClientOriginalName();

                    $fileToUpload = $originalName;

                    $image2 = Image::make($image->getRealPath());

                    $image2->resize(1000, 500)->save(public_path('assets/img/resized/' .
$request->session()->get('user')->id . '_resized_' . $fileToUpload));

                    $image->storeAs('public/img/' . $request->session()->get('user')->id .
'/', $fileToUpload);

                    $carImage->image = $fileToUpload;
                    $carImage->car_id = $lastId;

                    $carImage->save();
                }
            }
            DB::commit();
            Log::info($request->ip() . ' uploaded new entity(car): ' . $lastId . ' - id of new
entity');
            return redirect()->route('adminCar.index')->with('success', "Successfully inserted
car!");
        } catch (\Exception $ex) {
            DB::rollBack();
            Log::error('Error creating new car. Message: ' . $ex);
            return redirect()->route('adminCar.index')->with('error', "Error inserting car!
Message: " . $ex->getMessage());
        }
    }


    public function show($id)
    {
        //
    }


    public function edit($id)
    {
        $car = Car::find($id);

        return view('pages.admin.cars.edit', [
            'car' => $car
        ]);
    }


    public function update(CarUpdateRequest $request, $id)
    {
        try {
            DB::beginTransaction();
            $car = Car::find($id);

            $car->title = $request->get('title');
            $car->car_mark_id = $request->get('car_mark');
            $car->car_model_id = $request->get('model');
            $car->car_type_id = $request->get('car_type');
```

```
            $car->gas_id = $request->get('gas');
            $car->drivetrain_id = $request->get('drivetrain');
            $car->description = $request->get('description');
            $car->year = $request->get('year');
            $car->new = $request->get('condition');
            $car->transmission = $request->get('transmission');
            $car->mileage = $request->get('mileage');
            $car->price = $request->get('price');

            $car->save();

            if ($request->file('files') != null) {
                foreach ($request->file('files') as $file) {
                    $carImage = new CarImage();
                    $image = $file;

                    $originalName = $image->getClientOriginalName();

                    $fileToUpload = $originalName;

                    $image2 = Image::make($image->getRealPath());

                    $image2->resize(1000, 500)->save(public_path('assets/img/resized/' . $car-
>user_id . '_resized_' . $fileToUpload));

                    $image->storeAs('public/img/' . $car->user_id . '/', $fileToUpload);

                    $carImage->image = $fileToUpload;
                    $carImage->car_id = $id;

                    $carImage->save();
                }
            }
            DB::commit();
            Log::info($request->ip() . ' updated his car: ' . $id . ' - id of the updated
entity');
            return redirect()->route('adminCar.index')->with('success', "Successfully updated
car from admin dashboard!");
        } catch (\Exception $ex) {
            DB::rollBack();
            Log::error('Error updating entity. Message: ' . $ex);
            return redirect()->route('adminCar.index')->with('error', "Error updating car from
admin dashboard! Message: " . $ex->getMessage() . ' Please contact manager to fix this
problem.');
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

```
6  <?php

   namespace App\Http\Controllers;

   use App\Http\Requests\CarRequest;
   use App\Http\Requests\CarUpdateRequest;
   use App\Models\Car;
   use App\Models\CarImage;
   use App\Models\CarModel;
   use App\Models\Location;
   use App\Models\User;
   use http\Message;
   use Illuminate\Support\Facades\File;
   use Illuminate\Http\Request;
   use Illuminate\Support\Facades\DB;
   use Illuminate\Support\Facades\Log;
   use Illuminate\Support\Facades\Storage;
   use Image;

   class CarController extends Controller
   {
       public function __construct(){
           $this->middleware('edit')->only('edit');
       }

       public function index(Request $request)
       {
           $cars = new Car();

           if ($request->has('location') && $request->get('location') != null) {
               $location = $request->get('location');
               $cars = $cars->whereHas('user', function ($query) use($location) {
                   return $query->where('location_id', $location);
               });
           }

           if ($request->has('carType')) {
               $carTypes = $request->get('carType');

               $cars = $cars->whereIn('car_type_id', $carTypes);
           };

           if ($request->has('yearFrom')) {
               $cars = $cars->whereBetween('year', [$request->get('yearFrom') ? $request-
   >get('yearFrom') : Car::min('year'), $request->get('yearTo') ? $request->get('yearTo') :
   Car::max('year')]);
           }

           if ($request->has('condition')) {
               $cars = $cars->where('new', $request->get('condition'));
           }

           if ($request->has('mark')) {
               if ($request->get('mark') !== '0') {
                   $cars = $cars->where('car_mark_id', '=', $request->get('mark'));
               }
           }
```

```
        if ($request->has('model')) {
            if ($request->get('model') !== '0') {
                $cars = $cars->where('car_model_id', '=', $request->get('model'));
            }
        }

        if ($request->has('priceFrom')) {
            if ($request->get('priceFrom') !== null) {
                $cars = $cars->where('price', '>=', intval($request->get('priceFrom')));
                $cars = $cars->where('price', '<=', intval($request->get('priceTo') !==
null ? $request->get('priceTo') : '1500000'));
            };
        }

        if ($request->has('drivetrain')) {
            $drivetrains = $request->get('drivetrain');

            $cars = $cars->whereIn('drivetrain_id', $drivetrains);
        }

        if ($request->has('gas')) {
            $gases = $request->get('gas');

            $cars = $cars->whereIn('gas_id', $gases);
        }

        if ($request->has('mileageFrom')) {
            if ($request->get('mileageFrom') !== null) {
                $cars = $cars->where('mileage', '>=', intval($request-
>get('mileageFrom')));
                $cars = $cars->where('price', '<=', intval($request->get('mileageTo') !==
null ? $request->get('mileageTo') : '15000000'));
            };
        }

        if ($request->has('transmission')) {
            $transmissions = $request->get('transmission');
            $cars = $cars->whereIn('transmission', $transmissions);
        }

        if ($request->has('sort')) {
            $order = $request->get('sort');

            if ($order == 'newest') {
                $cars = $cars->orderBy('year', 'desc');
            }
            if ($order == 'lth') {
                $cars = $cars->orderBy('price', 'asc');
            }

            if ($order == 'htl') {
                $cars = $cars->orderBy('price', 'desc');
            }
        }


        $cars = $cars->paginate(6);
```

```
        $totalOffers = Car::count();


        return view('pages.cars.index', [
            "totalOffers" => $totalOffers,
            "cars" => $cars
        ]);
    }


    public function create()
    {
        return view('pages.cars.create');
    }


    public function store(CarRequest $request)
    {
        try {
            DB::beginTransaction();
            $car = new Car();

            $car->title = $request->get('title');
            $car->car_mark_id = $request->get('car_mark');
            $car->car_model_id = $request->get('model');
            $car->car_type_id = $request->get('car_type');
            $car->gas_id = $request->get('gas');
            $car->user_id = $request->session()->get('user')->id;
            $car->drivetrain_id = $request->get('drivetrain');
            $car->description = $request->get('description');
            $car->year = $request->get('year');
            $car->new = $request->get('condition');
            $car->transmission = $request->get('transmission');
            $car->mileage = $request->get('mileage');
            $car->price = $request->get('price');

            $car->save();
            $lastId = $car->id;

            if ($request->file('files') != null) {
                foreach ($request->file('files') as $file) {
                    $carImage = new CarImage();
                    $image = $file;

                    $originalName = $image->getClientOriginalName();

                    $fileToUpload = $originalName;

                    $image2 = Image::make($image->getRealPath());

                    $image2->resize(1000, 500)->save(public_path('assets/img/resized/' .
$request->session()->get('user')->id . '_resized_' . $fileToUpload));

                    $image->storeAs('public/img/' . $request->session()->get('user')->id .
'/', $fileToUpload);

                    $carImage->image = $fileToUpload;
                    $carImage->car_id = $lastId;
```

```
                    $carImage->save();
                }
            }
            DB::commit();
            Log::info($request->ip() . ' uploaded new entity(car): ' . $lastId . ' - id of
new entity');
            return redirect()->route('cars.index')->with('success', "Successfully inserted
car!");
        } catch (\Exception $ex) {
            DB::rollBack();
            Log::error('Error creating new car. Message: ' . $ex);
            return redirect()->back()->with('error', 'Failed to create entity. Please make
sure you entered all fields.');
        }
    }


    public function show($id)
    {
        $car = Car::find($id);

        return view('pages.cars.show', [
            'car' => $car
        ]);
    }


    public function edit(Request $request, $id)
    {
        $carToEdit = Car::find($id);

        return view('pages.cars.edit', [
            'car' => $carToEdit
        ]);
    }


    public function update(CarUpdateRequest $request, $id)
    {
        try {
            DB::beginTransaction();
            $car = Car::find($id);

            $car->title = $request->get('title');
            $car->car_mark_id = $request->get('car_mark');
            $car->car_model_id = $request->get('model');
            $car->car_type_id = $request->get('car_type');
            $car->gas_id = $request->get('gas');
            $car->user_id = $request->session()->get('user')->id;
            $car->drivetrain_id = $request->get('drivetrain');
            $car->description = $request->get('description');
            $car->year = $request->get('year');
            $car->new = $request->get('condition');
            $car->transmission = $request->get('transmission');
            $car->mileage = $request->get('mileage');
            $car->price = $request->get('price');

            $car->save();
```

```
            if ($request->file('files') != null) {
                foreach ($request->file('files') as $file) {
                    $carImage = new CarImage();
                    $image = $file;

                    $originalName = $image->getClientOriginalName();

                    $fileToUpload = $originalName;

                    $image2 = Image::make($image->getRealPath());

                    $image2->resize(300, 200)->save(public_path('assets/img/resized/' .
$request->session()->get('user')->id . '_resized_' . $fileToUpload));

                    $image->storeAs('public/img/' . $request->session()->get('user')->id .
'/', $fileToUpload);

                    $carImage->image = $fileToUpload;
                    $carImage->car_id = $id;

                    $carImage->save();
                }
            }
            DB::commit();
            Log::info($request->ip() . ' updated his car: ' . $id . ' - id of the updated
entity');
            return redirect()->route('cars.index')->with('success', "Successfully updated
your car!");
        } catch (\Exception $ex) {
            DB::rollBack();
            Log::error('Error updating entity. Message: ' . $ex);
        }
    }


    public function destroy(Request $request, $id)
    {
        if ($request->hasHeader('Accept') && $request->header('Accept') ==
'application/json') {
            $itemToDelete = Car::find($id);
            $carImagesToDelete = CarImage::where('car_id', $itemToDelete->id)->get();
            try {
                foreach ($carImagesToDelete as $deleteOne) {
                    CarImage::destroy($deleteOne->id);
//                    if(File::exists(public_path('storage/img/' . $itemToDelete->user->id
. '/' . $deleteOne->image))){
//                        File::delete(public_path('storage/img/' . $itemToDelete->user-
>id . '/' . $deleteOne->image));
//                    }
                }
                Car::destroy($itemToDelete->id);
                Log::info($request->ip() . ' deleted his entity(car)');
                return ["Message" => "Succesfully deleted!"];
            } catch (\Exception $ex) {
                Log::error('Error deleting car with his all photos from database.
Exception message: ' . $ex);
                return ["ErrorMessage", "Error deleting your item" . $ex];
            }
        } else {
```

```
                $itemToDelete = Car::find($id);
                $carImagesToDelete = CarImage::where('car_id', $itemToDelete->id)->get();
                try {
                    foreach ($carImagesToDelete as $deleteOne) {
                        CarImage::destroy($deleteOne->id);
                    }
                    Car::destroy($itemToDelete->id);
    //                if(File::exists(public_path('storage/img/' . $itemToDelete->user->id .
    '/' . $deleteOne->image))){
    //                    File::delete(public_path('storage/img/' . $itemToDelete->user->id .
    '/' . $deleteOne->image));
    //                }
                    Log::info($request->ip() . ' deleted his entity(car)');
                    return redirect()->route('users.index')->with('success', 'Successfully
    deleted entity');
                } catch (\Exception $ex) {
                    Log::error('Error deleting car with his all photos from database.
    Exception message: ' . $ex);
                    return redirect()->route('users.index')->with('error', 'Error deleting
    your entity" . $ex');
                }
            }

        }

        public function deleteImage(Request $request, $id){
            try {
                CarImage::find($id)->delete();
                return response()->json(["success" => "Deleted image from server"]);
            }catch (\Exception $exception){
                return response()->json(["error" => "Error with server " . $exception]);
            }
        }
    }
```

5.6 HomeController

```
<?php

namespace App\Http\Controllers;

use App\Models\CarModel;
use App\Models\User;
use Illuminate\Http\Request;

class HomeController extends DefaultController
{
    public function index()
    {
        return view('pages.car_finder_home');
    }
}
```

5.7 LocationAdminController

```
<?php

namespace App\Http\Controllers;
```

```
use App\Http\Requests\LocationRequest;
use App\Models\Location;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class LocationAdminController extends Controller
{

    public function index(Request $request)
    {
        $locations = new Location();

        if($request->has('keywords') && $request->get('keywords') !== null) {
            $keywords =  $request->get('keywords');
            $locations = $locations->where('location', 'like', '%' . $keywords . '%');
        }

        return  view('pages.admin.locations.index', [
            'locations' => $locations->get()
        ]);
    }


    public function create()
    {
        return  view('pages.admin.locations.create');
    }

    public function store(LocationRequest $request)
    {
        try {
            $location = new Location();

            $location->location = $request->get('location');

            $location->save();

            Log::info($request->ip() . ' added new location');
            return redirect()->route('location.index')->with('success', 'Successfully added
new location');
        } catch (\Exception $exception){
            Log::info($request->ip() . ' Error adding new location!');
            return redirect()->route('location.index')->with('error', 'Error inserting new
location! ' . $exception->getMessage());
        }
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }
```

```
    public function edit($id)
    {
        $location = Location::find($id);

        return view('pages.admin.locations.edit', [
            'location' => $location
        ]);
    }


    public function update(LocationRequest $request, $id)
    {
        try {
            $location = Location::find($id);

            $location->location = $request->get('location');

            $location->save();

            Log::info($request->ip() . ' updated location with id: ' . $id);
            return redirect()->route('location.index')->with('success', 'Successfully updated
location');
        } catch (\Exception $exception){
            Log::info($request->ip() . ' Error updating location!');
            return redirect()->route('location.index')->with('error', 'Error updating existing
location! ' . $exception->getMessage());
        }
    }


    public function destroy(Request $request,$id)
    {
        try {
            Location::destroy($id);

            Log::info($request->ip() . ' deleted location ');
            return redirect()->route('location.index')->with('success', 'Successfully deleted
user!');

        }catch (\Exception $exception){
            Log::info($request->ip() . ' tried unsuccessfully deleting location with id: ');
            return redirect()->route('location.index')->with('error', 'Error deleting
location! ' . $exception->getMessage());
        }
    }
}
```

5.8 UserAdminController

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\RegisterRequest;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Storage;
```

```
class UserAdminController extends Controller
{

    public function index(Request $request)
    {
        $users = new User();

        if ($request->has('keywords') && $request->get('keywords') != null) {
            $keywords = $request->get('keywords');
            $users = $users->where('first_name', 'Like', '%' . $keywords . '%');
        }

        return view('pages.admin.users.index', [
            'users' => $users->paginate(5)
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('pages.admin.users.create');
    }


    public function store(RegisterRequest $request)
    {

        try {
            $user = new User();

            $user->first_name = $request->get('first_name');
            $user->last_name = $request->get('last_name');
            $user->phone = $request->get('phone');
            $user->email = $request->get('email');
            $user->password = md5($request->get('password'));
            $user->location_id = $request->get('location');
            $user->role_id = $request->get('role');
            $user->photo = $request->file('photo')->getClientOriginalName();

            $user->save();
            $lastId = $user->id;

            if ($user) {
                Storage::makeDirectory('public/img/' . $lastId . '/profile/');
                $request->file('photo')->storeAs('public/img/' . $lastId . '/profile/',
$request->file('photo')->getClientOriginalName());
                Log::info($request->ip() . ' registered new user via Admin dashboard');
                return redirect()->route('adminUser.index')->with('success', 'Successfully
created new used!');
            }
        } catch (\Exception $exception) {
            Log::error($request->ip() . ' failed registraction via Admin dashboard! Message: '
. $exception);
            return redirect()->route('adminUser.index')->with('error', 'Error processing your
```

```
data. ' . $exception->getMessage());
        }
    }

    /**
     * Display the specified resource.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }


    public function edit($id)
    {
        $user = User::find($id);

        return view('pages.admin.users.edit', [
            'user' => $user
        ]);
    }


    public function update(Request $request, $id)
    {
        try {
            $user = User::find($id);

            $user->first_name = $request->get('first_name');
            $user->last_name = $request->get('last_name');
            $user->email = $request->get('email');
            $user->phone = $request->get('phone');
            $user->location_id = $request->get('location');
            $user->role_id = $request->get('role');

            $user->save();

            return redirect()->route('adminUser.index')->with('success', "Successfully updated
user!");

        } catch (\Exception $exception) {
            Log::warning('Exception updating profile from admin dashboard! ' . $exception-
>getMessage());
            return redirect()->route('adminUser.index')->with('error', 'Error updating
profile. ' . $exception->getMessage());
        }
    }


    public function destroy(Request $request,$id)
    {
        $user = User::find($id);
        try {
            User::destroy($user->id);

            Log::info($request->ip() . ' deleted user with id: ' . $user->id);
```

```
            return redirect()->route('adminUser.index')->with('success', 'Successfully deleted
user!');

        } catch (\Exception $exception) {
            Log::info($request->ip() . ' tried unsuccessfully deleting user with id: ' .
$user->id);
            return redirect()->route('adminUser.index')->with('error', 'Error deleting user!
Cant delete user that has posted cars! ');
        }
    }
}
```

5.9 UserController

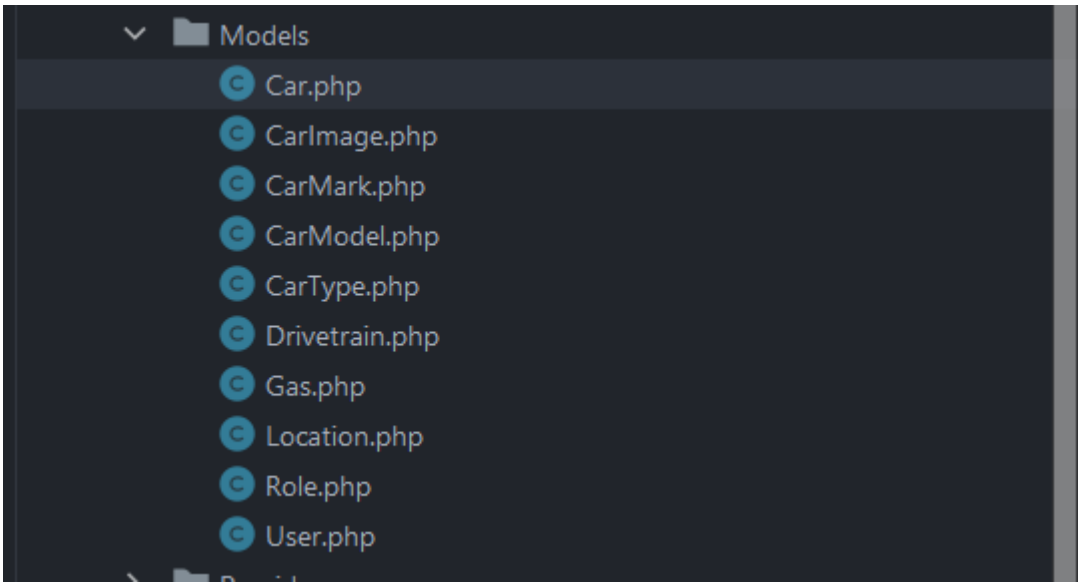
```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;
use Image;

class UserController extends Controller
{

    public function index()
    {
        return view('pages.users.index');
    }


    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
```

```
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }


    public function update(Request $request, $id)
    {
        $user = User::find($id);
        $image = null;
        try {
            if ($request->file('profile_picture') != null) {
                $image = $request->file('profile_picture');
                $image->storeAs('public/img/' . $user->id . '/profile/', $image->getClientOriginalName());
            }


            $user->first_name = $request->get('first_name');
            $user->last_name = $request->get('last_name');
            $user->email = $request->get('email');
            $user->phone = $request->get('phone');
            if($image){
            $user->photo = $image->getClientOriginalName();
            }
            $user->save();

            $request->session()->put('user', $user);
            return redirect()->route('users.index')->with('success', "Successfully updated profile!");

        } catch (\Exception $exception) {
            Log::warning('User updating exception! ' . $exception->getMessage());
            return redirect()->route('users.index')->with('error', 'Error updating your profile');
        }

    }

    public function updatePassword(Request $request, $id)
    {
        $user = User::find($id);

        $currentPasswordDatabase = $user->password;

        $current = md5($request->get('current_password'));
        $new = md5($request->get('new_password'));
        $newRepeat = md5($request->get('new_password_repeat'));

//        dd($current . ' ' . $new);
```

```
        if($currentPasswordDatabase != $current){
            return redirect()->route('users.index')->with('error', 'Current password doesnt
match with actual one');
        }
        if($new !== $newRepeat){
            return redirect()->route('users.index')->with('error', 'Passwords doesnt match');
        }

        try {
            $user->password = $new;
            $user->save();

            return redirect()->route('users.index')->with('success', 'Password successfully
updated!');
        }catch (\Exception $exception){
            dd($exception);
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

6.0 Models

6.1 Slika modela



6.2 Car

```php
<?php

namespace App\Models;

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

class Car extends Model
{
    use HasFactory;

    public function car_mark(){
        return $this->belongsTo(CarMark::class);
    }

    public function car_model() {
        return $this->belongsTo(CarModel::class);
    }

    public function car_type() {
        return $this->belongsTo(CarType::class);
    }

    public function gas() {
        return $this->belongsTo(Gas::class);
    }

    public function user() {
        return $this->belongsTo(User::class);
    }

    public function drivetrain() {
        return $this->belongsTo(Drivetrain::class);
    }

    public function car_images(){
        return $this->hasMany(CarImage::class);
    }
}
```

6.3 CarImage

```php
<?php

namespace App\Models;

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

class CarImage extends Model
{
    use HasFactory;

    function car() {
        return $this->belongsTo(Car::class);
    }
}
```

6.4 CarMark

```php
<?php

namespace App\Models;

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

class CarMark extends Model
{
    use HasFactory;

    public function car_models(){
        return $this->hasMany(CarModel::class);
    }

    public function cars(){
        return $this->hasMany(Car::class);
    }
}
```

6.5 CarModel

```php
<?php

namespace App\Models;

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

class CarModel extends Model
{
    use HasFactory;

    public function car_mark(){
        return $this->belongsTo(CarMark::class);
    }

    public function cars(){
        return $this->hasMany(Car::class);
    }
}
```

6.6 CarType

```php
<?php

namespace App\Models;

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

class CarType extends Model
{
    use HasFactory;

    public function cars(){
```

```
        return $this->hasMany(Car::class);
    }
}
```

6.7 Drivetrain

```
<?php

namespace App\Models;

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

class Drivetrain extends Model
{
    use HasFactory;

    public function cars(){
        return $this->hasMany(Car::class);
    }
}
```

6.8 Gas

```
<?php

namespace App\Models;

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

class Gas extends Model
{
    use HasFactory;

    public function cars(){
        return $this->hasMany(Car::class);
    }
}
```

6.9 Location

```
<?php

namespace App\Models;

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

class Location extends Model
{
    use HasFactory;

    public function users(){
        return $this->hasMany(User::class);
    }
}
```

6.10 Role

```php
<?php

namespace App\Models;

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

class Role extends Model
{
    use HasFactory;

    public function users() {
        return $this->hasMany(User::class);
    }
}
```

6.11 User

```php
<?php

namespace App\Models;

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

class User extends Model
{
    use HasFactory;

    public function location(){
        return $this->belongsTo(Location::class);
    }

    public function role(){
        return $this->belongsTo(Role::class);
    }

    public function cars(){
        return $this->hasMany(Car::class);
    }
}
```

7.0 Middlewares

7.1 LoggedInMiddleware

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class LoggedInMiddleware
{
```

```
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        if(!$request->session()->has('user')){

            Log::warning($request->ip() . ' tried to access protected routes without
permission!');
            return redirect()->route('cars.index')->with('error', 'Please login/register to
our application to perfome this action');
        }

        return $next($request);
    }
}
```

7.2 AdminMiddleware

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class AdminMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        if ($request->session()->has('user') && $request->session()->get('user')->role->role
=== 'Admin') {
            Log::info($request->ip() . ' approached Admin page');
            return $next($request);
        }

        Log::warning($request->ip() . 'tried approaching Admin page without permissions!');
        return redirect()->route('cars.index')->with('error', 'Your account doesnt have admin
privileges.');

    }
}
```

8.0 Javascript

8.1 carsAdmin.js

```
const deleteBtns = document.querySelectorAll('.btnDelete');


deleteBtns.forEach((btn) => {
    btn.addEventListener('click', (e) => {
        e.preventDefault();
        if(confirm("Are you sure you want to delete?")) {
            deleteCar(e.target.getAttribute('data-id')).then(response =>
response.json()).then(data => {
                console.log(data);
                document.getElementById('message').innerHTML = `
                <div class="alert alert-success">
                        <ul>
                            <li>${data.Message}</li>
                        </ul>
                </div>
                `
                document.getElementById('car-' + e.target.getAttribute('data-
id')).parentElement.parentElement.remove();
                setTimeout(() => {
                    document.getElementById('message').innerHTML = '';
                }, 2500)
            }).catch(error => {
                console.log(error);
                document.getElementById('message').innerHTML = `
                <div class="alert alert-danger">
                        <ul>
                            <li>${error.ErrorMessage}</li>
                        </ul>
                </div>
                `
            });

        }

    })
})

function deleteCar(id){
    return fetch('/cars/' + id, {
        headers: {
            "Accept": "application/json",
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        },
        method: "delete"
    });
}
```

8.2 deleteImage.js

```
const imagesToDelete = document.querySelectorAll('.delete-img');

imagesToDelete.forEach((element) => {
    element.addEventListener('click', (e) => {
        e.preventDefault();
```

```
        if (confirm("Are you sure you want to delete this picture? You wont be able to recover
it after deleting!") == true) {
            deleteImage(e.target.getAttribute('data-id'))
                .then(response => response.json())
                .then(data => {
                    document.getElementById('msg').innerHTML = `
                <div class="alert-success">
                    <p>${data.success}</p>
                </div>
                `
                    element.parentElement.remove();
                })
                .catch(err => {
                    document.getElementById('msg').innerHTML = `
                <div class="alert-success">
                    <p>${err.error}</p>
                </div>
                `
                });
        }

    })
})

async function deleteImage(id) {
    return await fetch('/cars/images/' + id, {
        headers: {
            "Accept": "application/json",
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        },
        method: "delete"
    })
}
```

8.3 main.js

```
const model = document.getElementById('model');
const mark = document.querySelector('#mark');
const deleteBtns = document.querySelectorAll('.btnDelete');
const modelToPreselect = document.getElementById('modelToPreselect').value;

// console.log(document.getElementById('modelToPreselect'));

if(mark.value == 0){
    mark.addEventListener('change', function (e) {
        fetchModelOnValue(e.target.value);

    })
}
else {
    fetchModelOnValue(mark.value);
    mark.addEventListener('change', function (e) {
        fetchModelOnValue(e.target.value);

    })
}

deleteBtns.forEach((btn) => {
```

```
    btn.addEventListener('click', (e) => {
        e.preventDefault();
        if(confirm("Are you sure you want to delete?")) {
            deleteCar(e.target.getAttribute('data-id')).then(response =>
response.json()).then(data => {
                document.getElementById('message').innerHTML = `
                <div class="alert alert-success">
                        <ul>
                            <li>${data.Message}</li>
                        </ul>
                </div>
                `
                window.scrollTo(0, 0);
                document.getElementById('car-' + e.target.getAttribute('data-id')).remove();
                setTimeout(() => {
                    document.getElementById('message').innerHTML = '';
                }, 2500)
                // window.scrollTo(0, 0);
                // setTimeout(() => {
                //     location.reload();
                // }, 1500);
            }).catch(error => {
                document.getElementById('message').innerHTML = `
                <div class="alert alert-danger">
                        <ul>
                            <li>${error.ErrorMessage}</li>
                        </ul>
                </div>
                `
            });

        }

    })
})

async function fetchModel(markId){
    const data = await fetch('/models/' + markId);

    return await data.json();
}

function deleteCar(id){
    return fetch('/cars/' + id, {
        headers: {
            "Accept": "application/json",
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        },
        method: "delete"
    });
}

function fetchModelOnValue(value){
    fetchModel(value).then(data => {
        model.removeAttribute('disabled');
        let html = '<option value="0">Any model</option>'
        data.forEach((option) => {
            html += `
            <option value="${option.id}" ${option.id == modelToPreselect ? "selected" : ""}
```

```
>${option.model}</option>
            `
        })
        model.innerHTML = html;
    });
}
```

8.4 user.js

```
const contentDivs = document.querySelectorAll('.col-lg-8.col-md-7.mb-5');
const allUserBtns = document.querySelectorAll('.card-nav-link.d');

// const personalInfoBtn = document.getElementById('personal-info-btn');
// const passwordSecurityBtn = document.getElementById('password-security-btn');
// const myCarsBtn = document.getElementById('my-cars-btn');
// const wishListBtn = document.getElementById('wishlist-btn');

// personalInfoBtn.classList.remove('active');
contentDivs.forEach((div) => {
    div.style.display = 'none'
})

contentDivs[0].style.display = 'block';

allUserBtns.forEach((btn, index) => {
    btn.addEventListener('click', (e) => {
        e.preventDefault();
        contentDivs.forEach((div) => {
            div.style.display = 'none';
            contentDivs[index].style.display = 'block';
            allUserBtns.forEach((btn) => {
                btn.classList.remove('active');
            })
            e.target.classList.add('active');
            document.getElementById('breadcrumb').innerText = e.target.text.trim();
        })
    })
})

// personalInfoBtn.addEventListener('click', (e) => {
//     e.preventDefault();
//     contentDivs.forEach((div) => {
//         div.style.display = 'none';
//         contentDivs[0].style.display = 'block';
//         allUserBtns.forEach((btn) => {
//             btn.classList.remove('active');
//         })
//         personalInfoBtn.classList.add('active');
//     })
// })
//
// passwordSecurityBtn.addEventListener('click', (e) => {
//     e.preventDefault();
//     contentDivs.forEach((div) => {
//         div.style.display = 'none';
//         contentDivs[1].style.display = 'block';
//         allUserBtns.forEach((btn) => {
```

```
//             btn.classList.remove('active');
//         })
//         passwordSecurityBtn.classList.add('active');
//     })
// });
```

8.5 userAdmin.js

```
const deleteBtns = document.querySelectorAll('.delete');

deleteBtns.forEach((btn) => {
    btn.addEventListener('click', (e) => {
        e.preventDefault();
        let idToDelete = e.target.getAttribute('data-delete');

        deleteUser(idToDelete).then(res => res.json()).then(data =>
location.reload()).catch(error => {
            document.getElementById('message').innerHTML = `
            <div class="alert alert-danger">
                    ${error.message}
            </div>
            `
        });
    })
})


function deleteUser(id) {
    return fetch('/admin/adminUser/' + id, {
        method: "delete",
        headers: {
            "Accept": "application/json",
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    })
}
```