VISOKA ŠKOLA STRUKOVNIH STUDIJA ZA INFORMACIONE I KOMUNIKACIONE TEHNOLOGIJE

WEB PROGRAMIRANJE PHP2

DOKUMENTACIJA PROJEKTA

Student:

Aleksa Jovanovic 179/15

Beograd 2019.

SADRŽAJ

1.Opis funkcionalnosti

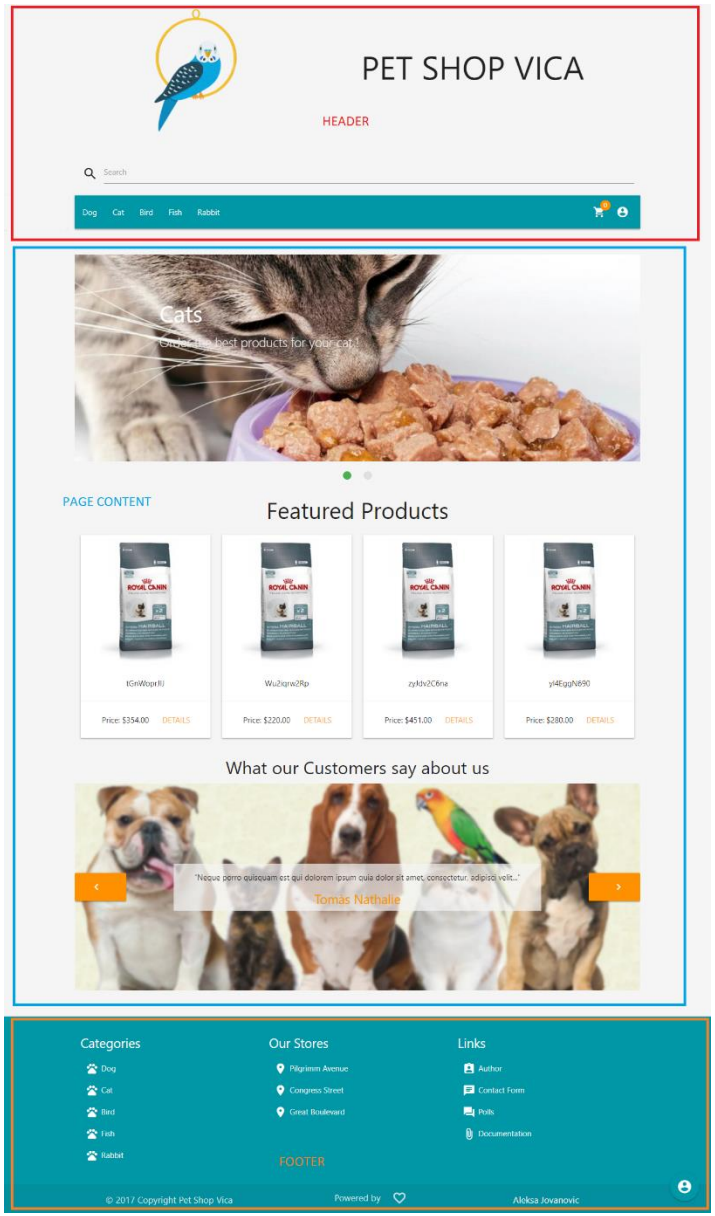
Aplikacija je realizovana kao internet prodavnica, korisnik ima pregled svih dostupnih proizvoda. Proizvode moze pretrazivati, sortirati i filtrirati, takodje ih moze dodavati u korpu ali tek kada se registruje i uloguje. Nakon dodavanja proizvoda u korpu, korisnik moze kreirati porudzbinu. Porudzbina se moze kreirati uspesno, samo ako su prethodno ispunjena dva uslova. Prvi uslov: korisnik prethodno mora popuniti svoj profil sa kontakt podacima. Drugi uslov: proizvodi koji se nalaze u korpi moraju biti na stanju. Nakon uspesno kreirane porudzbine, korisnik ima uvid u sve informacije vezane za samu porudzbinu, moze pratiti njen status (active, canceled ili delivered), ali je isto tako moze otkazati ili cak i obrisati. Pri nedostatku informacija uvek moze kontaktirati prodavca pomocu kontakt forme. Mogucnost glasanja u anketi i pregled trenutnog rezultata neposredno nakon glasanja.

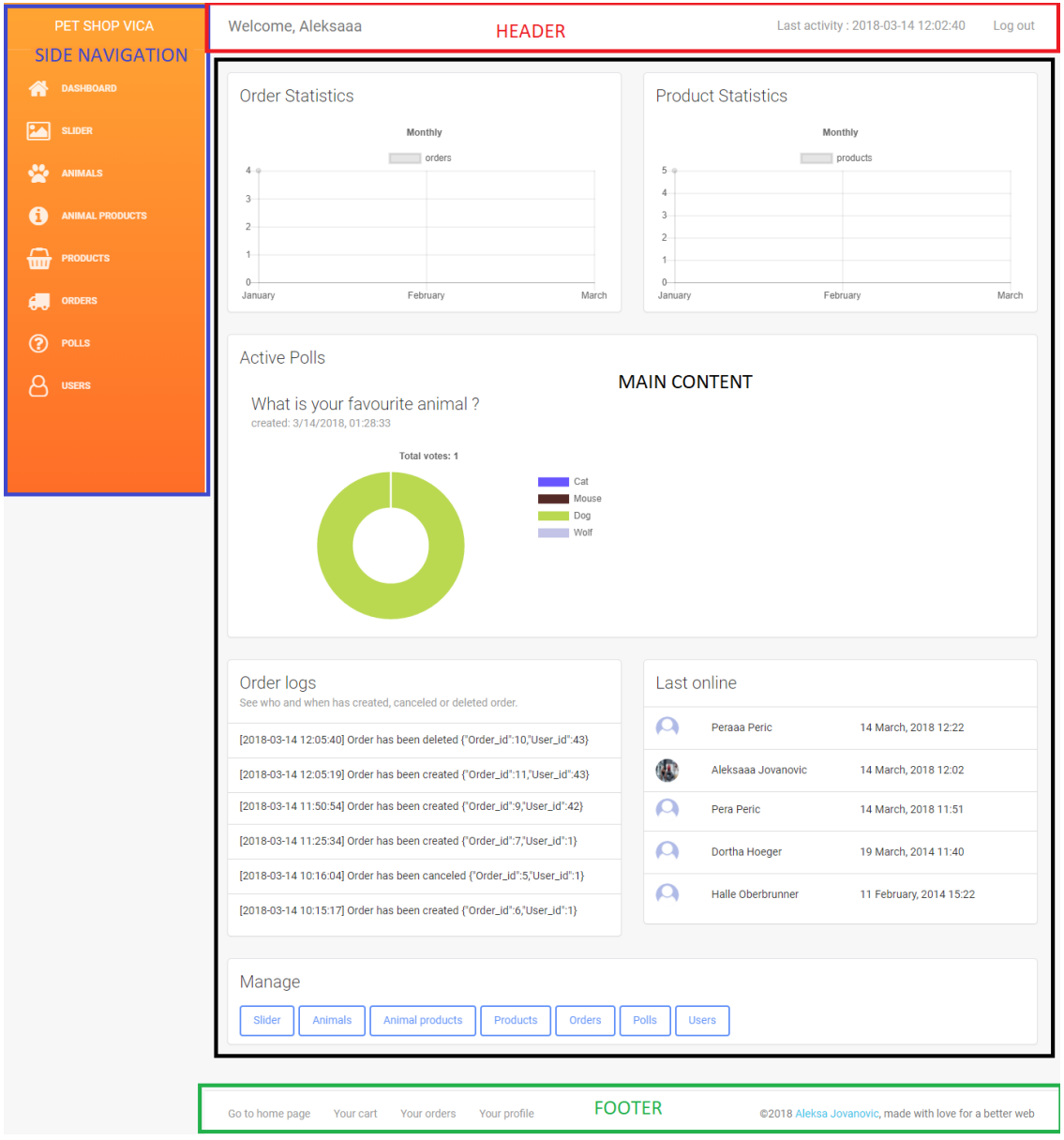
Administrator ima potpunu kontrolu nad internet prodavnicom. Upravljanje korisnicima, uvid u informacije kada su poslednji put bili ulogovani. Pregled statistickih podataka koji se ticu porudzbina i proizvoda na mesecnom nivou. Upravljanje porudzbinama (menjanje njihovog statusa). Uvid u logove porudzbina (kad i koji korisnik je kreirao, otkazao ili obrisao porudzbinu). Dodavanje, izmena i brisanje svih kategorija zivotinja, tipova proizvoda i na kraju samih proizvoda. Administrator moze kreirati, izmeniti ili brisati ankete, takodje moze videti njihove rezultate. Potpuno upravljanje slajderom koji se nalazi na pocetnoj strani.

Korišćeni jezici	• HTML 4 i HTML 5 • CSS • JavaScript • PHP • MySQL
Korišćeni HTML šabloni (framework-ci)	https://www.creative-tim.com/product/light-bootstrap-dashboard
Korišćene biblioteke i njihovi url-ovi odakle ste ih preuzeli	• jQuery (https://code.jquery.com/jquery-3.3.1.min.js) • Materialize CSS/JS (http://next.materializecss.com/getting-started.html) • React.js (https://unpkg.com/react@16/umd/react.development.js) • Redux.js (https://cdnjs.cloudflare.com/ajax/libs/redux/3.7.2/redux.min.js) • Chart.js (https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.2/Chart.min.js)
Korišćene biblioteke iz framework-a	• Laravel
Korišćene php biblioteke van fremework-a	• Eloquent-sluggable (https://github.com/cviebrock/eloquent-sluggable)
Koje funkcionalnosti su realizove putem AJAX-a	• Dodavanje proizvoda u korpu • Anketa (glasanje i prikaz rezultata) • Pretraga proizvoda • Ceo admin panel
Navesti lokacije gde je napisan AJAX kod (fajlovi ili delovi html stranice)	• addToCart.js • pollShowResults.js • pollVote.js • productSearch.js • adminApp.js
Navesti lokacije gde je napisan sopstveni javaScript kod (fajlovi ili delovi html stranice)	• addToCart.js • pollShowResults.js • pollVote.js • productSearch.js • shopFunctions.js • makeOrder.js • quantityControlsCart.js • removeFromCart.js • updateCart.js • formValidations.js • adminApp.js

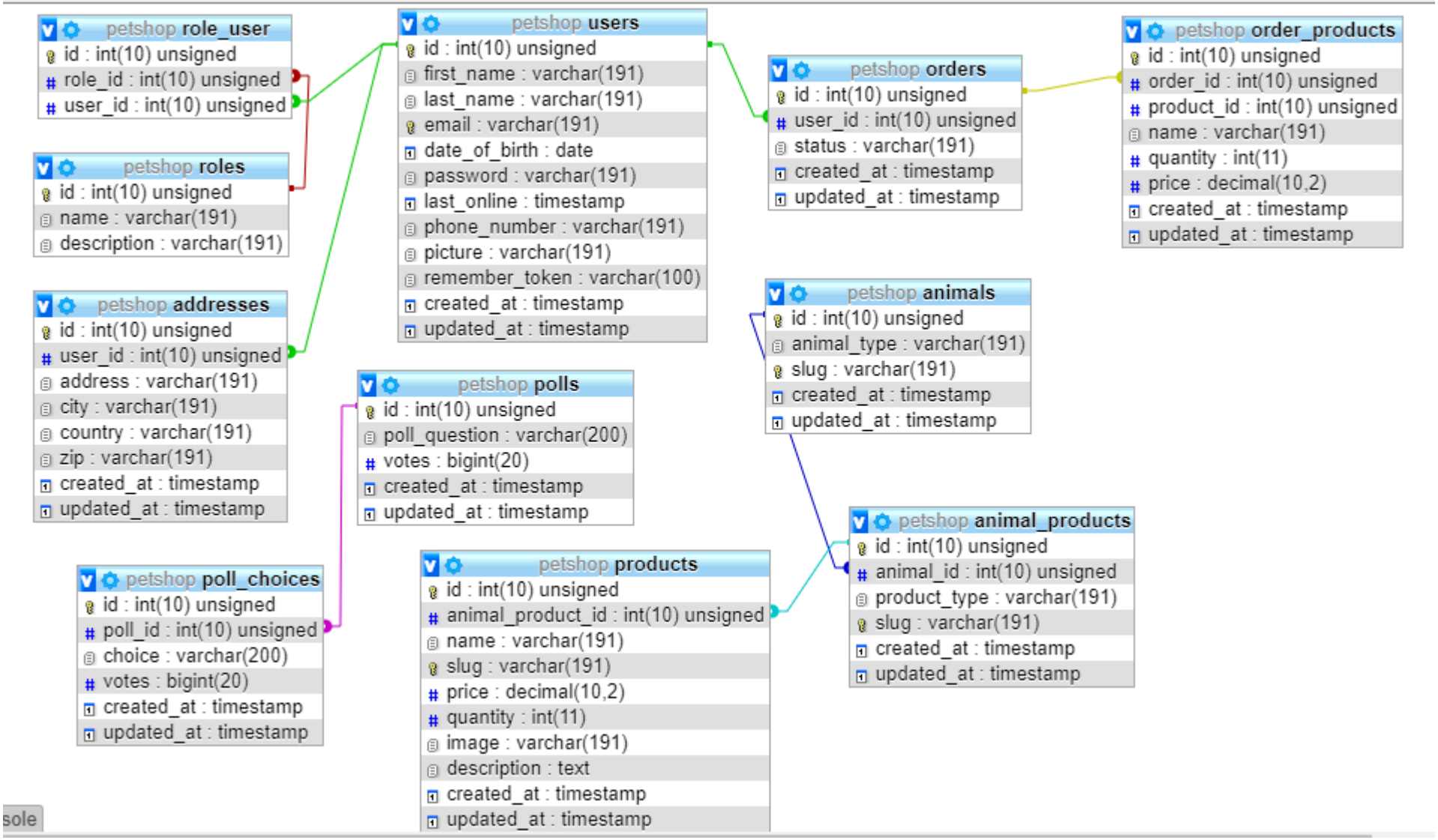
URL sajta/aplikacije	https://petshop.aleksajovanovic.com
Pristupni parametri	Prosledjeni na mail.

2. Skica struktura stranica





3. Dijagram baze podataka



3.MVC Organizacija

Kontroleri

HomeController
index
author
contact
contactSubmit
pollsIndex
pollsVote
documentationDownload

ShopController
indexAnimal
indexProductTypes
indexProduct
searchProduct

ProfileController
index
indexPersonal
indexContact
indexAddress
indexSettings
updatePersonal
updateContact
updatePicture
updateAddress
updatePassword
destroy

CartController
index
addToCart
removeFromCart
updateCart

OrdersController
index
createOrder

cancelOrder
deleteOrder

AnimalProductsController
index
store
show
update
destroy

AnimalsController
index
store
show
update
destroy

DashboardController
index
user

Dashboard/OrdersController
index
show
update
destroy

PollsController
index
store
show
update
destroy

ProductsController
index
store
show
update
destroy

SliderController
index

store
show
update
destroy

UsersController
index
store
show
updateBasic
updateContact
updatePassword
updatePicture
destroy

Modeli

Napomena : Korisceni su eloquent modeli.

Address
getUser

Animal
productTypes
products
scopeAnimalFilter
sluggable

AnimalProduct
animal
products
scopeAnimalProductFilter
sluggable

Order
getOrderProducts
contactInfo
customer
scopeOrderFilter
scopeInactiveOrders
scopeMonthStatistics

OrderProduct
getOrder

getTotalAttribute
scopeMonthStatistics

Poll
choices
scopePollFilter

PollChoice
poll

Product
productType
scopeSearch
scopePaginateProducts
scopeProductFilter
sluggable

Role
users

User
getAllOrders
getContactInfo
getDateOfBirthAttribute
setDateOfBirthAttribute
dontHaveContactInfo
roles
isAdmin
scopeSearchByName
scopeUserFilter
lastOnline

View

admin
Dashboard.blade.php

auth
login.blade.php
register.blade.php

auth/user
cart.blade.php
orders.blade.php
profileLayout.php

auth/user/profile
profileSettings.blade.php
profileView.blade.php

auth/user/profile/forms
profileAddress.blade.php
profileContact.blade.php
profileDelete.blade.php
profilePassword.blade.php
profilePersonal.blade.php
profilePicture.blade.php

errors
401.blade.php
404.blade.php

layouts
adminLayouts.blade.php
main.blade.php

mails
contactMessage.blade.php

pages
author.blade.php
contact.blade.php
home.blade.php

polls.blade.php

pages/shop
product.blade.php
products.blade.php

pages/shop/components
breadcrumbs.blade.php
productCard.blade.php
shopMenu.blade.php

layoutPartials
authCircles.blade.php
footer.blade.php
header.blade.php

4. javaScript kod

addToCart.js

```
const _addToCartShop = () => {
    let button = $("#_customer_AddToCartShop");
    let badge = $(".shopping-cart-badge");

    button.click(function(e) {
        e.preventDefault();
        let productId = $(this).attr("data-product-id");
        let productQuantity = $("#_customer_product_quantity").val();
        axios
            .post("/profile/cart", { id: productId, quantity: productQuantity
})
            .then(res => {
```

```
                badge.html(res.data.cartItemsCount);
                _M_.toast({ html: res.data.message });
            })
            .catch(err => _M_.toast({ html: err.response.data.message }));
    });
};

export default _addToCartShop;
```

removeFromCart.js

```
const _removeFromCart = () => {
    let removeCartButton = $(".cart-container .remove-from-cart-button");
    removeCartButton.click(function(e) {
        e.preventDefault();
        let productId = $(this).attr("data-product-id");

        axios
            .delete("/profile/cart", { params: { id: productId } })
            .then(res => {
                _M_.toast({ html: res.data.message });

                setTimeout(() => {
                    window.location = "/profile/cart";
                }, 700);
            })
            .catch(err => console.log(err.response));
    });
};
export default _removeFromCart;
```

updateCart.js

```
import { getCartProductsFromTable } from "./cartHelpers";
const _updateCart = () => {
    let updateCartButton = $(".cart-container #_update-cart-button");

    updateCartButton.click(function(e) {
        e.preventDefault();
        let products = getCartProductsFromTable();
        if (products.length <= 0) return 0;
        axios
            .put("/profile/cart", {
                orderData: products
            })
            .then(res => {
                _M_.toast({
                    html: res.data.message
                });
                setTimeout(() => {
                    window.location = "/profile/cart";
                }, 700);
            })
            .catch(err => {
                let errResp = err.response;
                _M_.toast({
                    html: errResp.data.message
                });
                console.log(errResp.data.message);
            });
    });
};
export default _updateCart;
```

quantityControlsCart.js

```
const _quantityControlsCart = () => {
    let addButton = $(".cart-container .cart-quantity-add");
    let removeButton = $(".cart-container .cart-quantity-remove");

    addButton.click(function(e) {
        e.preventDefault();
        let inputItem = `.cart-container
#_cart_product_quantity_${$(this).attr(
            "data-product-id"
        )}`;
        let productQuantity = $(inputItem).val();

        $(inputItem).val(++productQuantity);
    });
    removeButton.click(function(e) {
        e.preventDefault();
        let inputItem = `.cart-container
#_cart_product_quantity_${$(this).attr(
            "data-product-id"
        )}`;
        let productQuantity = $(inputItem).val();

        if (productQuantity > 1) $(inputItem).val(--productQuantity);
    });
};
export default _quantityControlsCart;
```

contactFormValidation.js

```
const contactFormValidate = () => {
```

```
    $("#contact-form").validate({
        errorElement: "span",
        errorClass: "helper-text",
        rules: {
            first_name: "required",
            last_name: "required",
            email: { required: true, email: true },
            message: "required"
        },
        messages: {
            first_name: "Enter your First name!",
            last_name: "Enter your Last name!",
            email: {
                required: "Email is required!",
                email: "Please enter a valid email address!"
            },
            message: "Message is required!"
        },
        errorPlacement: (error, element) => {
            error.appendTo(element.parent());
        }
    });
};

export default contactFormValidate;
```

loginFormValidation.js

```
const loginFormValidate = () =>
    $("#login-form").validate({
        errorElement: "span",
        errorClass: "helper-text",
```

```
        rules: {
            email: { required: true, email: true },
            password: "required"
        },
        messages: {
            email: {
                required: "Email is required!",
                email: "Please enter a valid email address!"
            },
            password: "Password is required!"
        },
        errorPlacement: (error, element) => {
            error.appendTo(element.parent());
        }
    });
export default loginFormValidate;
```

registerFormValidation.js

```
const registerFormValidate = () => {
    let dateOfBirthEl = document.querySelector("#date_of_birth");
    if (dateOfBirthEl) {
        let instance_DOB_register = _M_.Datepicker.init(dateOfBirthEl, {
            format: "dd mmmm, yyyy",
            minDate: new Date(1900, 1, 1),
            maxDate: new Date(2009, 11, 31),
            defaultDate: new Date(2009, 11, 31),
            yearRange: 15
        });
    }

    $("#register-form").validate({
```

```
        errorElement: "span",
        errorClass: "helper-text",
        rules: {
            first_name: "required",
            last_name: "required",
            email: { required: true, email: true },
            date_of_birth: "required",
            password: { required: true, minlength: 10 },
            password_confirmation: {
                required: true,
                equalTo: "#password"
            }
        },
        messages: {
            first_name: "Enter your firstname!",
            last_name: "Enter your lastname!",
            email: {
                required: "Email is required!",
                email: "Please enter a valid email address!"
            },
            date_of_birth: "Date of birth is required!",
            password: {
                required: "Provide a password",
                minlength: jQuery.validator.format("Enter at least {0}
characters!")
            },
            password_confirmation: {
                required: "Repeat your password!",
                equalTo: "Enter the same password as above!"
            }
        },
        errorPlacement: (error, element) => {
```

```
            error.appendTo(element.parent());
        }
    });
};

export default registerFormValidate;
```

cancelOrder.js

```
const _cancelOrder = () => {
    let button = $("._cancel-order-button");

    button.click(function(e) {
        e.preventDefault();
        let orderId = $(this).attr("data-order-id");

        axios
            .put("/profile/orders", { id: orderId })
            .then(res => {
                console.log(res);
                _M_.toast({ html: res.data.message });
                setTimeout(() => {
                    window.location = "/profile/orders";
                }, 700);
            })
            .catch(err => {
                console.log(err.response.data);
                _M_.toast({ html: err.response.data.message });
            });
    });
};
```

```
export default _cancelOrder;
```

deleteOrder.js

```
const _deleteOrder = () => {
    let button = $("._delete-order-button");

    button.click(function(e) {
        let orderId = $(this).attr("data-order-id");
        axios
            .delete("/profile/orders", { params: { id: orderId } })
            .then(res => {
                console.log(res);
                _M_.toast({ html: res.data.message });
                setTimeout(() => {
                    window.location = "/profile/orders";
                }, 700);
            })
            .catch(err => {
                console.log(err.response.data);
                _M_.toast({ html: err.response.data.message });
            });
    });
};

export default _deleteOrder;
```

pollVote.js

```
import { pollShowChart } from "./pollShowResults";
```

```
export const pollVote = () => {
    $(".poll-form").submit(function(e) {
        e.preventDefault();
        const data = {};
        data.poll_id = $(this).attr("data-poll-id");
        data.choice_id = $(this).serializeArray()[0].value;

        axios
            .post("/polls", data)
            .then(response => {
                console.log(response);
                const { data: { message, poll_results } } = response;

                _M_.toast({ html: message });
                pollShowChart(poll_results);
            })
            .catch(error => {
                console.log(error.response);
                const { response: { data: { message, poll_results }, status }
} = error;
                _M_.toast({ html: message });
                if (status == 401) return;

                pollShowChart(poll_results);
            });
    });
};
```

pollShowResults.js

```
import Chart from "chart.js";
export const pollShowChart = poll_results => {
```

```
    const dataSet = _.map(poll_results.choices, choice =>
parseInt(choice.votes, 10));
    const labels = _.map(poll_results.choices, "choice");
    const totalVotes = dataSet.reduce((acc, curr) => (acc += curr));

    const canvas = $(`#myChart_poll_${poll_results.id}`);
    const ctx = canvas[0].getContext("2d");
    $(canvas).fadeIn();
    const myChart = new Chart(ctx, {
        type: "bar",
        data: {
            labels: labels,
            datasets: [
                {
                    label: `Total: ${totalVotes} Votes`,
                    data: dataSet,
                    borderWidth: 2
                }
            ]
        },
        options: {
            hover: {
                mode: "label"
            },
            responsive: true,
            title: {
                display: true,
                text: "Poll results"
            },
            legend: { position: "bottom" },
            scales: {
                yAxes: [
```

```
                    {
                        ticks: {
                            beginAtZero: true,
                            max: totalVotes
                        }
                    }
                ]
            }
        }
    });
    myChart.update();
};
```

productSearch.js

```
import { _debounce } from "../HelperClasses/debounce_throttle";
const productSearchMaterialize = () => {
    let elem = document.querySelector(".product-search-autocomplete");
    if (!elem) return 0;
    let state;
    let instance = _M_.Autocomplete.init(elem, {
        onAutocomplete: function(value) {
            let product = state.find(x => x.name == value);
            window.location.href =
`/shop/${product.animal_slug}/${product.product_type_slug}/${product.slug}`;
        }
    });

    $(elem).keyup(
        _debounce(function(e) {
            if ($(this).val().length > 10 || $(this).val().length == 0) return
0;
```

```
            axios
                .get(`/shop/search?product=${$(this).val()}`)
                .then(res => {
                    let transformedData = Object.assign({},
...res.data.products.map(x => ({ [x.name]: `/${x.image}` })));
                    instance.updateData(transformedData);
                    $("input.autocomplete").blur();
                    $("input.autocomplete").focus();
                    state = res.data.products;
                })
                .catch(err => console.log(err.response));
        }, 600)
    );
};
export default productSearchMaterialize;
```

5. PHP kod

routes/web.php

```
<?php

use Illuminate\Support\Facades\Auth;



Route::get('/', 'HomeController@index')->name('home');

Route::get('/author', 'HomeController@author')->name('author');

Route::get('/contact', 'HomeController@contact')->name('contact');
```

```
Route::post('/contact', 'HomeController@contactSubmit')-
>name('contactSubmit');

Route::get('/polls', 'HomeController@pollsIndex')->name('polls');
Route::post('/polls', 'HomeController@pollsVote');

Route::get('/docdownload', 'HomeController@documentationDownload')-
>name('documentation');

Auth::routes();

Route::group(['prefix' => '/shop'], function () {
    Route::get('/search', 'ShopController@searchProduct');
    Route::get('/{animal}', 'ShopController@indexAnimal');
    Route::get('/{animal}/{productType}', 'ShopController@indexProductTypes');
    Route::get('/{animal}/{productType}/{product}',
'ShopController@indexProduct');
});

Route::group(['prefix' => '/profile', 'middleware' => 'auth'], function () {
    Route::get('/', 'ProfileController@index')->name('profile');
    Route::get('/personal', 'ProfileController@indexPersonal')-
>name('profilePersonal');
    Route::get('/contact', 'ProfileController@indexContact')-
>name('profileContact');
    Route::get('/address', 'ProfileController@indexAddress')-
>name('profileAddress');
    Route::get('/settings', 'ProfileController@indexSettings')-
>name('profileSettings');

    Route::post('/personal', 'ProfileController@updatePersonal')-
>name('updateProfilePersonal');
```

```
    Route::post('/contact', 'ProfileController@updateContact')-
>name('updateProfileContact');
    Route::post('/address', 'ProfileController@updateAddress')-
>name('updateProfileAddress');
    Route::post('/picture', 'ProfileController@updatePicture')-
>name('updateProfilePicture');
    Route::post('/password', 'ProfileController@updatePassword')-
>name('updateProfilePassword');

    Route::delete('/', 'ProfileController@destroy')->name('profileDestroy');

    //CART
    Route::get('/cart', 'CartController@index')->name('cart');
    Route::put('/cart', 'CartController@updateCart')->name('cartUpdate');
    Route::post('/cart', 'CartController@addToCart')->name('cartAdd');
    Route::delete('/cart', 'CartController@removeFromCart')-
>name('cartDelete');

    //ORDERS
    Route::get('/orders', 'OrdersController@index')->name('orders');
    Route::post('/orders', 'OrdersController@createOrder');
    Route::put('/orders', 'OrdersController@cancelOrder');
    Route::delete('/orders', 'OrdersController@deleteOrder');
});

Route::group(['middleware' => 'adminAuth'], function () {
    Route::group(['prefix' => '/dashboardApi'], function () {
        Route::group(['prefix' => '/home'], function () {
            Route::get('/', 'Dashboard\HomeController@index');
            Route::get('/user', 'Dashboard\HomeController@user');
        });
```

```
Route::group(['prefix' => '/users'], function () {
    Route::get('/', 'Dashboard\UsersController@index');
    Route::post('/', 'Dashboard\UsersController@store');
    Route::get('/{id}', 'Dashboard\UsersController@show');
    Route::put('/basic', 'Dashboard\UsersController@updateBasic');
    Route::put('/contact', 'Dashboard\UsersController@updateContact');
    Route::put('/auth', 'Dashboard\UsersController@updatePassword');
    Route::put('/picture', 'Dashboard\UsersController@updatePicture');
    Route::delete('/', 'Dashboard\UsersController@destroy');
});

Route::group(['prefix' => '/animals'], function () {
    Route::get('/', 'Dashboard\AnimalsController@index');
    Route::post('/', 'Dashboard\AnimalsController@store');
    Route::get('/{id}', 'Dashboard\AnimalsController@show');
    Route::put('/', 'Dashboard\AnimalsController@update');
    Route::delete('/', 'Dashboard\AnimalsController@destroy');
});

Route::group(['prefix' => '/animalproducts'], function () {
    Route::get('/', 'Dashboard\AnimalProductsController@index');
    Route::post('/', 'Dashboard\AnimalProductsController@store');
    Route::get('/{id}', 'Dashboard\AnimalProductsController@show');
    Route::put('/', 'Dashboard\AnimalProductsController@update');
    Route::delete('/', 'Dashboard\AnimalProductsController@destroy');
});

Route::group(['prefix' => '/products'], function () {
    Route::get('/', 'Dashboard\ProductsController@index');
    Route::post('/', 'Dashboard\ProductsController@store');
    Route::get('/{id}', 'Dashboard\ProductsController@show');
    Route::put('/', 'Dashboard\ProductsController@update');
```

```
            Route::delete('/', 'Dashboard\ProductsController@destroy');
        });

        Route::group(['prefix' => '/orders'], function () {
            Route::get('/', 'Dashboard\OrdersController@index');
            Route::get('/{id}', 'Dashboard\OrdersController@show');
            Route::put('/', 'Dashboard\OrdersController@update');
            Route::delete('/', 'Dashboard\OrdersController@destroy');
        });

        Route::group(['prefix' => '/slider'], function () {
            Route::get('/', 'Dashboard\SliderController@index');
            Route::post('/', 'Dashboard\SliderController@store');
            Route::get('/{id}', 'Dashboard\SliderController@show');
            Route::put('/', 'Dashboard\SliderController@update');
            Route::delete('/', 'Dashboard\SliderController@destroy');
        });

        Route::group(['prefix' => '/polls'], function () {
            Route::get('/', 'Dashboard\PollsController@index');
            Route::post('/', 'Dashboard\PollsController@store');
            Route::get('/{id}', 'Dashboard\PollsController@show');
            Route::put('/', 'Dashboard\PollsController@update');
            Route::delete('/', 'Dashboard\PollsController@destroy');
        });
    });
    Route::get('/dashboard/{path?}', function () {
        return view('admin.dashboard');
    })->where('path', '.*')->name('adminDashboard');
});
```

CartController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Product;
use App\OrderProduct;
use App\Order;
use App\MyClasses\DBBulkUpdate;
use Illuminate\Support\Facades\DB;

class CartController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $products = OrderProduct::where('order_id', session('cart'))->get();

        $subtotal = $products->sum(
            function ($product) {
                return (double)$product->price * (double)$product->quantity;
            }
        );

        return view('auth.user.cart', ['products' => $products, 'subtotal' =>
$subtotal]);
```

```
    }

    public function addToCart(Request $request)
    {
        $formData = $request->validate([
            'id' => 'required|integer',
            'quantity' => 'required|integer|min:1'
        ]);

        if ($request->session()->has('cart')) {
            try {
                $product = Product::findOrFail($formData['id']);
            } catch (\Exception $e) {
                return response()->json(['message' => 'Product not found,
Please try again!'], 404);
            }
            try {
                $productOrder = OrderProduct::firstOrNew(
                    ['order_id' => session('cart'), 'product_id' =>
$formData['id']],
                    ['name' => $product->name, 'price' => $product->price]
                );
                $productOrder->quantity = ($productOrder->quantity +
(int)$formData['quantity']);
                $productOrder->save();

                return response()->json(['message' => 'Successfully added to
cart', 'cartItemsCount' => OrderProduct::where('order_id', session('cart'))-
>count()], 200);
            } catch (\Exception $e) {
                return response()->json(['message' => 'Woops something went
wrong, please try again later!'], 503);
```

```
            }
        } else {
            return response()->json(['message' => 'Session has expired, please sign in !'], 401);
        }
    }

    public function removeFromCart(Request $request)
    {
        $request->validate([
            'id' => 'required|integer'
        ]);
        if (!$request->session()->has('cart')) {
            return response()->json(['message' => 'Session has expired, please sign in !'], 401);
        }

        try {
            $orderProduct = OrderProduct::where([['id', $request->id], ['order_id', session('cart')]])->first();
        } catch (\Exception $e) {
            return response()->json(['message' => 'Product is not in your cart !'], 404);
        }
        if ($orderProduct->getOrder->status === 'inactive') {
            $orderProduct->delete();

            return response()->json(['message' => 'Product successfully removed !'], 200);
        }
    }
```

```
    public function updateCart(Request $request)
    {
        if (!$request->session()->has('cart')) {
            return response()->json(['message' => 'Session has expired, please
sign in !'], 401);
        }

        $formData = $request->validate([
            'orderData' => 'required|array|',
            'orderData.*.id' => 'required|integer',
            'orderData.*.quantity' => 'required|integer|min:1',
        ]);

        $orderProducts = Order::findOrFail(session('cart'))->getOrderProducts-
>map(function ($item, $key) use ($formData) {
            $item->quantity = (int)$formData['orderData'][$key]['quantity'];

            return collect($item);
        });

        DB::transaction(function () use ($orderProducts) {
            $dbBulkInstance1 = new DBBulkUpdate('order_products',
$orderProducts);
            $dbBulkInstance1->setColumnsToUpdate(['quantity']);
            DB::statement($dbBulkInstance1->generateQuery(), $orderProducts-
>flatten()->toArray());
        });

        return response()->json(['message' => 'Successfully updated cart !'],
200);
    }
}
```

HomeController.php

```php
<?php

namespace App\Http\Controllers;

use App\Poll;
use App\Product;
use App\PollChoice;
use Illuminate\Http\Request;
use App\Mail\ContactMessage;
use App\MyClasses\PollFileHelper;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Facades\Storage;

class HomeController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $products = Product::inRandomOrder()->take(4)->with('productType.animal')->get();
        $slider = collect(json_decode(Storage::get('slider.json')))->sortBy('order');
```

```
        return view('pages.home', ['products' => $products, 'slider' =>
$slider]);
    }

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

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

    public function contactSubmit(Request $request)
    {
        $request->validate([
            'first_name' => 'required|string|max:100',
            'last_name' => 'required|string|max:100',
            'email' => 'required|string|email|max:100',
            'message' => 'required|string'
        ]);
        Mail::to('aleksa.jovanovic.179.15@gmail.com')->send(new
ContactMessage($request->all()));

        return redirect()->route('contact')->with('message', 'The contact mail
has been sent!');
    }

    public function pollsIndex()
    {
        $polls = Poll::with('choices')->get();
```

```
        return view('pages.polls.polls', ['polls' => $polls]);
    }

    public function pollsVote(Request $request)
    {
        if (!Auth::check()) {
            return response()->json(['message' => 'You must be logged in to be
able to vote for this poll !'], 401);
        }
        $request->validate([
            'poll_id' => 'required|integer',
            'choice_id' => 'required|integer'
        ]);
        $pollFileHelper = new PollFileHelper(['poll_id' => $request->poll_id,
'user_id' => Auth::user()->id]);
        $poll = Poll::where('id', $request->poll_id)->with('choices')-
>first();
        if ($pollFileHelper->alreadyVoted()) {
            return response()->json(['message' => 'You already voted for this
poll !', 'poll_results' => $poll], 403);
        }
        $pollFileHelper->write();

        $poll->increment('votes');
        PollChoice::where('id', $request->choice_id)->increment('votes');

        return response()->json(['message' => 'Thank you for voting !',
'poll_results' => Poll::where('id', $request->poll_id)->with('choices')-
>first()], 200);
    }
```

```
    public function documentationDownload()
    {
        return response()->download(public_path('documentation.pdf'),
'documentation.pdf', ['Content-Type: application/pdf']);
    }
}
```

OrdersController.php

```
<?php

namespace App\Http\Controllers;

use App\Order;
use App\Product;
use App\OrderProduct;
use Illuminate\Http\Request;
use App\MyClasses\DBBulkUpdate;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Gate;
use App\MyClasses\LogMessagesHelper;

class OrdersController extends Controller
{
    protected $logger;

    public function __construct(LogMessagesHelper $logger)
    {
        $this->logger = $logger;
    }
```

```
    public function index()
    {
        $user = Auth::user();
        $orders = $user->getAllOrders()->inactiveorders()->get();
        $userAddress = $user->getContactInfo;

        return view('auth.user.orders', ['user' => $user, 'userAddress' =>
$userAddress, 'orders' => $orders]);
    }

    public function createOrder(Request $request)
    {
        if (Auth::user()->dontHaveContactInfo()) {
            return response()->json(['message' => 'To make an order you must
update your profile with address and phone information !'], 403);
        }
        if (!$request->session()->has('cart')) {
            return response()->json(['message' => 'Session has expired, please
sign in !'], 401);
        }

        $request->validate([
            'orderData' => 'required|array|',
            'orderData.*.id' => 'required|integer',
            'orderData.*.quantity' => 'required|integer|min:1',
        ]);

        $cartId = session('cart');
        $orderDataCollection = collect($request->orderData)->sortBy('id')-
>values();
        $productsId = $orderDataCollection->pluck('id');
        $errorArray = [];
```

```
        $orderProducts = OrderProduct::where('order_id', $cartId)->get()->map(function ($item, $key) {
            $item->quantity = (int) request('orderData')[$key]['quantity'];

            return  collect($item);
        });

        $products = Product::findOrFail($productsId)->map(function ($item, $key) use (&$errorArray,$orderDataCollection) {
            $orderDataQuantity = $orderDataCollection[$key]['quantity'];
            if ($orderDataQuantity > $item->quantity) {
                $errorArray[] = ['id' => $item->id, 'product' => $item->name, 'maxQuant' => $item->quantity];
            }
            $item->quantity = ($item->quantity - $orderDataQuantity);

            return collect($item);
        });

        //POKAZI PRODUCTE KOJIH NEMA NA LAGERU
        if (count($errorArray) > 0) {
            return response()->json(['message' => $errorArray], 400);
        }
        //BULK UPDATE TABELE
        DB::transaction(function () use ($orderProducts,$products) {
            $dbBulkInstance1 = new DBBulkUpdate('order_products', $orderProducts);
            $dbBulkInstance1->setColumnsToUpdate(['quantity']);

            $dbBulkInstance2 = new DBBulkUpdate('products', $products);
            $dbBulkInstance2->setColumnsToUpdate(['quantity']);
```

```
            DB::statement($dbBulkInstance1->generateQuery(), $orderProducts->flatten()->toArray());
            DB::statement($dbBulkInstance2->generateQuery(), $products->flatten()->toArray());
            Order::find(session('cart'))->update(['status' => 'active']);
        });

        $userId = Auth::user()->id;

        $newOrderCart = Order::create(['user_id' => $userId]);

        $this->logger->write('Order has been created', ['Order_id' => $newOrderCart->id, 'User_id' => $userId]);
        session(['cart' => $newOrderCart->id]);

        return response()->json(['message' => 'Successfully ordered !', 'cartId' => $cartId], 200);
    }

    public function cancelOrder(Request $request)
    {
        $request->validate(['id' => 'required|integer']);
        try {
            $order = Order::findOrFail($request->id);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Order not found'], 404);
        }

        if (Gate::allows('cancelOrDeleteOrder', $order)) {
            $order->status = 'canceled';
            $order->save();
```

```
            $this->logger->write('Order has been canceled', ['Order_id' =>
$order->id, 'User_id' => Auth::user()->id]);

            return response()->json(['message' => 'You have canceled your
order !'], 200);
        }

        return response()->json(['message' => 'You are not allowed to cancel
this order'], 401);
    }

    public function deleteOrder(Request $request)
    {
        $request->validate(['id' => 'required|integer']);
        try {
            $order = Order::findOrFail($request->id);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Order not found'], 404);
        }
        if (Gate::allows('cancelOrDeleteOrder', $order)) {
            $order->delete();

            $this->logger->write('Order has been deleted', ['Order_id' =>
$order->id, 'User_id' => Auth::user()->id]);

            return response()->json(['message' => 'Successfully deleted order
!'], 200);
        }

        return response()->json(['message' => 'You are not allowed to delete
this order'], 401);
```

```
    }
}
```

ProfileController.php

```
<?php

namespace App\Http\Controllers;

use App\User;
use Illuminate\Http\Request;
use App\Rules\IsOldPassword;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Storage;
use App\Rules\IsDifferentFromOldPassword;
use Illuminate\Support\Facades\Validator;

class ProfileController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('auth.user.profile.profileView', ['userData' =>
['personalInfo' => Auth::user(), 'contactInfo' => Auth::user()-
>getContactInfo]]);
    }
```

```
    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function indexPersonal()
    {
        return view('auth.user.profile.forms.profilePersonal', ['userData' =>
Auth::user()]);
    }

    public function indexContact()
    {
        return view('auth.user.profile.forms.profileContact', ['userData' =>
Auth::user()]);
    }

    public function indexAddress()
    {
        return view('auth.user.profile.forms.profileAddress', [
                'userData' => Auth::user()->getContactInfo,
                'countries' => json_decode(Storage::get('countries.json'))
        ]);
    }

    public function indexSettings()
    {
        return view('auth.user.profile.profileSettings', []);
    }

    /**
     * Update the specified resource in storage.
```

```
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function updatePersonal(Request $request)
    {
        $formData = $request->validate([
            'first_name' => 'required|string|max:100',
            'last_name' => 'required|string|max:100',
            'date_of_birth' => 'required|string|date_format:"j M,
Y"|after_or_equal:"1 January 1900"|before_or_equal:"31 December 2009"|max:70',
        ]);

        Auth::user()->fill([
            'first_name' => $formData['first_name'],
            'last_name' => $formData['last_name'],
            'date_of_birth' => $formData['date_of_birth']
        ])->save();

        return redirect()->back()->with('authUserMessage', 'Successfully
updated personal data!');
    }

    public function updateContact(Request $request)
    {
        $formData = $request->validate([
            'email' => 'required|string|email|max:100|unique:users,email,' .
auth()->user()->id,
            'phone_number' => 'required|digits:10',
        ]);

        Auth::user()->fill([
```

```
            'email' => $formData['email'],
            'phone_number' => $formData['phone_number']
        ])->save();

        return redirect()->back()->with('authUserMessage', 'Successfully
updated contact info !');
    }

    public function updatePicture(Request $request)
    {
        $request->validate([
            'profilePicture' => 'required|image',
        ]);

        if (!$request->file('profilePicture')->isValid()) {
            return redirect()->back()->withErrors(['profilePicture' => 'Error
with uploading picture']);
        }

        $previousPicture = Auth::user()->picture;

        if (File::exists($previousPicture) && $previousPicture !=
'images/user_images/default-profile-picture.png') {
            File::delete($previousPicture);
        }

        $picture = $request->file('profilePicture');
        $newPictureName = time() . '.' . $picture-
>getClientOriginalExtension();
        $uploadedPicture = $picture->move('images/user_images',
$newPictureName);
```

```
        Auth::user()->picture = $uploadedPicture->getPath() . '/' .
$newPictureName;
        Auth::user()->save();

        return redirect()->route('profile')->with('authUserMessage',
'Successfully updated profile picture !');
    }

    public function updateAddress(Request $request)
    {
        $formData = $request->validate([
            'country' => 'required|string|min:2|max:100',
            'city' => 'required|string|min:2|max:100',
            'zip' => 'required|digits:5',
            'address' => 'required|string|min:2|max:100',
        ]);

        Auth::user()->getContactInfo->fill([
            'country' => $formData['country'],
            'city' => $formData['city'],
            'zip' => $formData['zip'],
            'address' => $formData['address'],
        ])->save();

        return redirect()->back()->with('authUserMessage', 'Successfully
updated address info !');
    }

    public function updatePassword(Request $request)
    {
        $validator = Validator::make($request->all(), [
```

```
            'old_password' => ['required', 'string', 'max:50', 'min:10', new
IsOldPassword],
            'new_password' => ['required', 'string', 'max:50', 'min:10', new
IsDifferentFromOldPassword]
        ]);

        if ($validator->fails()) {
            return redirect(route('profileSettings') .
'#password_change_tab_id')
                        ->withErrors($validator)
                        ->withInput();
        }

        Auth::user()->password = bcrypt($formData['new_password']);
        Auth::user()->save();

        return redirect()->back()->with('authUserMessage', 'Successfully
changed password !');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'current_password' => ['required', 'string', 'min:10', new
IsOldPassword],
        ]);
```

```
        if ($validator->fails()) {
            return redirect(route('profileSettings') .
'#account_delete_tab_id')
                        ->withErrors($validator)
                        ->withInput();
        }

        $previousPicture = Auth::user()->picture;
        if (File::exists($previousPicture) && $previousPicture !=
'images/user_images/default-profile-picture.png') {
            File::delete($previousPicture);
        }

        User::destroy(Auth::user()->id);
        Auth::logout();

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

ShopController.php

```
<?php

namespace App\Http\Controllers;

use App\Animal;
use App\Product;
use Illuminate\Http\Request;
use App\MyClasses\Breadcrumbs;
```

```
class ShopController extends Controller
{
    private $data;

    public function __construct()
    {
        if ($animal = request()->route('animal')) {
            $animalModel = Animal::where('slug', $animal)->firstOrFail();
            $this->data = [
                'animal_type' => $animalModel->animal_type,
                'animal_slug' => $animalModel->slug,
                'product_types' => $animalModel->productTypes,
                'base_link' => url("/shop/$animalModel->slug/"),
                'breadcrumbs' => Breadcrumbs::generate(),
                ];
        }
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function indexAnimal($animal)
    {
        $animalModel = Animal::where('slug', $animal)->firstOrFail();

        $sortQuery = request()->input('sort') ?? false;

        $this->data['products'] = $animalModel->products($sortQuery)-
>paginateProducts();
```

```
        return view('pages.shop.products', ['pageData' => $this->data]);
    }

    public function indexProductTypes($animal, $productType)
    {
        $animalModel = Animal::where('slug', $animal)->firstOrFail();

        $sortQuery = request()->input('sort') ?? false;

        $this->data['products'] = $animalModel->products($sortQuery)
                                    ->where('animal_products.slug',
$productType)
                                        ->paginateProducts();

        return view('pages.shop.products', ['pageData' => $this->data]);
    }

    public function indexProduct($animal, $productType, $product)
    {
        $this->data['products'] = Product::where('slug', $product)-
>firstOrFail();

        return view('pages.shop.product', ['pageData' => $this->data]);
    }

    public function searchProduct(Request $request)
    {
        if ($request->query('product')) {
            $products = Product::search($request->query('product'))->get();

            return response()->json(['products' => $products], 200);
        }
```

```
        return response()->json(['errorMessage' => 'You must provide query
string (?product=product_name)!'], 422);
    }
}
```

AnimalProductsController

```
<?php

namespace App\Http\Controllers\Dashboard;

use App\AnimalProduct;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Http\Resources\AnimalProductCollection;

class AnimalProductsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $animalProducts = AnimalProduct::animalProductFilter()-
>with('animal:id,animal_type')->simplePaginate(20)->appends(request()-
>except('page'));

        return new AnimalProductCollection($animalProducts);
    }
```

```
    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
            'product_type' =>
'required|string|max:100|min:2|unique:animal_products,product_type',
            'animal_id' => 'required|integer',
        ]);
        try {
            AnimalProduct::create([
                'animal_id' => $request->animal_id,
                'product_type' => $request->product_type
            ]);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Animal product type is not
created!'], 500);
        }

        return response()->json(['message' => 'Animal product type
successfully created!'], 200);
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
```

```
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $animalProduct = AnimalProduct::where('id', $id)-
>with('animal:id,animal_type')->get();
        if (count($animalProduct) == 0) {
            return response()->json(['message' => 'Animal product type not
found!'], 404);
        }

        return  new AnimalProductCollection($animalProduct);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
            'product_type' =>
'required|string|max:100|min:2|unique:animal_products,product_type',
            'animal_id' => 'required|integer',
        ]);

        try {
            $animalProduct = AnimalProduct::findOrFail($request->id);
```

```
            $animalProduct->product_type = $request->product_type;
            $animalProduct->animal_id = $request->animal_id;
            $animalProduct->save();
        } catch (\Exception $e) {
            return response()->json(['message' => 'Animal product type is not
updated!'], 500);
        }

        return response()->json(['message' => 'Animal product type
successfully updated!'], 200);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
        ]);

        try {
            AnimalProduct::destroy($request->id);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Animal product type is not
deleted'], 500);
        }
```

```
        return response()->json(['message' => 'Successfully deleted animal
product type!'], 200);
    }
}
```

AnimalsController.php

```
<?php

namespace App\Http\Controllers\Dashboard;

use App\Animal;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Http\Resources\Animal as AnimalResource;

class AnimalsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $animals = Animal::animalFilter()->simplePaginate(20)-
>appends(request()->except('page'));

        return AnimalResource::collection($animals);
    }

    /**
```

```
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
            'animal_type' =>
'required|string|max:50|min:2|unique:animals,animal_type',
        ]);

        try {
            Animal::create([
                'animal_type' => $request->animal_type
            ]);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Animal is not created!'],
500);
        }

        return response()->json(['message' => 'Animal successfully created!'],
200);
    }

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

```
    {
        return new AnimalResource(Animal::findOrFail($id));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request)
    {
        $request->validate([
            'animal_id' => 'required|integer',
            'animal_type' =>
'required|string|max:50|min:2|unique:animals,animal_type',
        ]);

        try {
            $animal = Animal::findOrFail($request->animal_id);
            $animal->animal_type = $request->animal_type;
            $animal->save();
        } catch (\Exception $e) {
            return response()->json(['message' => 'Animal is not updated'],
500);
        }

        return response()->json(['message' => 'Successfully updated animal!'],
200);
    }
```

```
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
        ]);

        try {
            Animal::destroy($request->id);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Animal is not deleted'],
500);
        }

        return response()->json(['message' => 'Successfully deleted animal!'],
200);
    }
}
```

Dashboard/HomeController.php

```
<?php

namespace App\Http\Controllers\Dashboard;

use App\Poll;
use App\User;
```

```php
use App\Order;
use App\OrderProduct;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Auth;

class HomeController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $orderStatistics = Order::monthStatistics()->get();
        $productStatistics = OrderProduct::monthStatistics()->get();
        $pollsStatistics = Poll::with('choices:poll_id,choice,votes')->get();
        $orderLogs =
array_slice(array_reverse(file(storage_path('/logs/custom_info_log.log'))), 0,
6);
        $latestOnlineUsers = User::lastOnline()->get();

        return ['orderStatistics' => $orderStatistics,
                'productStatistics' => $productStatistics,
                'pollsStatistics' => $pollsStatistics,
                'orderLogs' => $orderLogs,
                'latestOnlineUsers' => $latestOnlineUsers,
            ];
    }

    public function user()
```

```
    {
        return Auth::user();
    }
}
```

OrdersController.php

```
<?php

namespace App\Http\Controllers\Dashboard;

use App\Order;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Http\Resources\Order as OrderResource;

class OrdersController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $orders = Order::orderFilter()->with('customer')->simplePaginate(20)->appends(request()->except('page'));

        return OrderResource::collection($orders);
    }

    /**
```

```
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $order = Order::where('id', $id)->with(['customer', 'contactInfo',
'getOrderProducts'])->first();

        if (!$order) {
            return response()->json(['message' => 'Order not found!'], 404);
        }

        return new OrderResource($order);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
            'status' => 'required|in:active,delivered,canceled'
        ]);

        $order = Order::findOrFail($request->id);
```

```
        $order->status = $request->status;
        $order->save();

        return response()->json(['message' => 'Successfully changed order
status !'], 200);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
        ]);

        try {
            Order::destroy($request->id);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Order is not deleted'],
500);
        }

        return response()->json(['message' => 'Successfully deleted order !'],
200);
    }
}
```

PollsController.php

```php
<?php

namespace App\Http\Controllers\Dashboard;

use App\Poll;
use App\PollChoice;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\File;
use App\Http\Resources\Poll as PollResource;
use App\MyClasses\PollFileHelper;

class PollsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $polls = Poll::pollFilter()->withCount('choices')->simplePaginate(20)->appends(request()->except('page'));

        return PollResource::collection($polls);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
```

```
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
           'poll_question' => 'required|string|min:2|max:100',
           'poll_choices.*' => 'required|string|min:2|max:50'
        ]);

        $poll = Poll::create([
            'poll_question' => $request->poll_question,
            'votes' => 0
        ]);

        PollFileHelper::createFile($poll->id);

        $choiceModelArray = [];
        foreach ($request->poll_choices as $choise) {
            $choiceModelArray[] = new PollChoice(['choice' => $choise, 'votes'
=> 0]);
        }

        $poll->choices()->saveMany($choiceModelArray);

        return response()->json(['message' => 'Successfully added poll !'],
200);
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
```

```
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $poll = Poll::where('id', $id)->with('choices')->firstOrFail();

        return new PollResource($poll);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
            'poll_question' => 'required|string|min:2|max:100',
            'poll_choices.*.id' => 'integer',
           ' poll_choices.*.choice' => 'required|string|min:2|max:50'
         ]);

        $poll = Poll::find($request->id);
        $poll->poll_question = $request->poll_question;

        foreach ($request->poll_choices as $poll_choice) {
            $choice = PollChoice::firstOrNew(['id' => $poll_choice['id']]);
            $choice->choice = $poll_choice['choice'];
            $poll->choices()->save($choice);
```

```
        }

        return response()->json(['message' => 'Successfully updated poll ! '],
200);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
        ]);

        try {
            $filePath = storage_path('/app/polls_' . $request->id . '.txt');
            if (File::exists($filePath)) {
                File::delete($filePath);
            }
            Poll::destroy($request->id);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Poll is not deleted'],
500);
        }

        return response()->json(['message' => 'Successfully deleted poll !'],
200);
    }
```

```
}
```

ProductsController.php

```
<?php

namespace App\Http\Controllers\Dashboard;

use App\Product;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\File;
use App\Http\Resources\ProductCollection;

class ProductsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $products = Product::productFilter()->with('productType.animal')-
>simplePaginate(20)->appends(request()->except('page'));

        return new ProductCollection($products);
    }

    /**
     * Store a newly created resource in storage.
     *
```

```
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
            'product_type_id' => 'required|integer',
            'name' => 'required|string|min:2|max:50',
            'price' => 'required|numeric',
            'quantity' => 'required|digits_between:1,10',
            'image' => 'required|image',
            'description' => 'required|string'
        ]);

        $image = $request->image;

        if (!$image->isValid()) {
            return response()->json(['errors' => ['image' => ['Error with
uploading picture']]], 422);
        }

        try {
            $newImageName = time() . '.' . $image-
>getClientOriginalExtension();
            $uploadedImage = $image->move('images/product_images',
$newImageName);
            $newImagePath = $uploadedImage->getPath() . '/' . $newImageName;

            Product::create([
                'animal_product_id' => $request->product_type_id,
                'name' => $request->name,
                'price' => $request->price,
```

```
                'quantity' => $request->quantity,
                'image' => $newImagePath,
                'description' => $request->description
            ]);
        } catch (\Exception $e) {
            return response()->json(['message' => 'Product is not created!'],
500);
        }

        return response()->json(['message' => 'Product successfully
created!'], 200);
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $product = Product::where('id', $id)->with('productType.animal')-
>get();

        if (count($product) == 0) {
            return response()->json(['message' => 'Product not found!'], 404);
        }

        return new ProductCollection($product);
    }

    /**
```

```
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
            'product_type_id' => 'required|integer',
            'name' => 'required|string|min:2|max:50',
            'price' => 'required|numeric',
            'quantity' => 'required|digits_between:1,10',
            'image' => 'image',
            'description' => 'required|string'
        ]);

        $product = Product::findOrFail($request->id);
        if ($request->hasFile('image')) {
            $image = $request->image;
            if (!$image->isValid()) {
                return response()->json(['errors' => ['image' => ['Error with
uploading picture']]], 422);
            }
            if (File::exists($product->image)) {
                File::delete($product->image);
            }

            $newImageName = time() . '.' . $image-
>getClientOriginalExtension();
```

```
            $uploadedImage = $image->move('images/product_images',
$newImageName);
            $newImagePath = $uploadedImage->getPath() . '/' . $newImageName;

            $product->image = $newImagePath;
        }

        try {
            $product->fill([
                'animal_product_id' => $request->product_type_id,
                'name' => $request->name,
                'price' => $request->price,
                'quantity' => $request->quantity,
                'description' => $request->description
            ])->save();
        } catch (\Exception $e) {
            return response()->json(['message' => 'Product is not updated!'],
500);
        }

        return response()->json(['message' => 'Product successfully
updated!'], 200);
    }

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

```
        $request->validate([
            'id' => 'required|integer',
        ]);

        try {
            $product = Product::findOrFail($request->id);

            if (File::exists($product->image)) {
                File::delete($product->image);
            }
            $product->delete();
        } catch (\Exception $e) {
            return response()->json(['message' => 'Product is not deleted'],
500);
        }

        return response()->json(['message' => 'Successfully deleted product
!'], 200);
    }
}
```

SliderController.php

```
<?php

namespace App\Http\Controllers\Dashboard;

use Illuminate\Http\Request;
use App\MyClasses\JsonSliderHelper;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Storage;
```

```php
class SliderController extends Controller
{
    private $sliderJSON;

    public function __construct()
    {
        $this->sliderJSON = Storage::get('slider.json');
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return response()->json(['slides' => $this->sliderJSON], 200);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
            'head' => 'required|string|max:50|min:2',
            'slogan' => 'required|string|max:50|min:2',
            'order' => 'required|integer',
```

```
            'captionPos' => 'required|in:center-align,left-align,right-align',
            'img' => 'required|image',
        ]);

        $img = $request->img;
        if (!$img->isValid()) {
            return response()->json(['errors' => ['picture' => ['Error with uploading picture']]], 422);
        }

        $newImgName = time() . '.' . $img->getClientOriginalExtension();
        $uploadedImg = $img->move('images/slider_images', $newImgName);
        $newImgPath = $uploadedImg->getPath() . '/' . $newImgName;

        $jsonSliderHelper = new JsonSliderHelper($this->sliderJSON);

        $jsonSliderHelper->addSlide([
            'id' => 0,
            'head' => $request->head,
            'slogan' => $request->slogan,
            'order' => $request->order,
            'captionPos' => $request->captionPos,
            'img' => $newImgPath
        ])->write();

        return response()->json(['message' => 'Successfully added new slide!'], 200);
    }

    /**
     * Display the specified resource.
     *
```

```
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $slide = collect(json_decode($this->sliderJSON))->firstWhere('id',
$id);
        if (!$slide) {
            return response()->json(['message' => 'Slide not found !'], 404);
        }

        return response()->json(['slide' => $slide], 200);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
            'head' => 'required|string|max:50|min:2',
            'slogan' => 'required|string|max:50|min:2',
            'order' => 'required|integer',
            'captionPos' => 'required|in:center-align,left-align,right-align',
            'img' => 'image',
        ]);
```

```
        $slide = collect(json_decode($this->sliderJSON))->firstWhere('id',
$request->id);
        $imgPath = $slide->img;

        if ($request->hasFile('img')) {
            $img = $request->img;
            if (!$img->isValid()) {
                return response()->json(['errors' => ['img' => ['Error with
uploading picture']]], 422);
            }
            $slide = collect(json_decode($this->sliderJSON))->firstWhere('id',
$request->id);
            if (File::exists($slide->img)) {
                File::delete($slide->img);
            }

            $newImgName = time() . '.' . $img->getClientOriginalExtension();
            $uploadedImg = $img->move('images/slider_images', $newImgName);
            $newImgPath = $uploadedImg->getPath() . '/' . $newImgName;

            $imgPath = $newImgPath;
        }

        $jsonSliderHelper = new JsonSliderHelper($this->sliderJSON);

        $jsonSliderHelper->updateSlide([
            'id' => $request->id,
            'head' => $request->head,
            'slogan' => $request->slogan,
            'order' => $request->order,
            'captionPos' => $request->captionPos,
            'img' => $imgPath
```

```
        ])->write();

        return response()->json(['message' => 'Successfully updated slide!'],
200);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
        ]);
        $slide = collect(json_decode($this->sliderJSON))->firstWhere('id',
$request->id);

        if (File::exists($slide->img)) {
            File::delete($slide->img);
        }

        $jsonSliderHelper = new JsonSliderHelper($this->sliderJSON);
        $newSlides = $jsonSliderHelper->removeSlide($request->id)->write();

        return response()->json(['message' => 'Successfully deleted slide!',
'slides' => $newSlides], 200);
    }
}
```

UsersController.php

```php
<?php

namespace App\Http\Controllers\Dashboard;

use App\User;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\File;
use App\Http\Resources\User as UserResource;
use App\Http\Resources\UserDetails as UserDetailsResource;

class UsersController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $users = User::userFilter()->simplePaginate(20)->appends(request()->except('page'));

        return UserResource::collection($users);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
```

```
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
            'first_name' => 'required|string|max:100',
            'last_name' => 'required|string|max:100',
            'email' => 'required|string|email|max:100|unique:users',
            'password' => 'required|string|min:10',
            'role' => 'required|integer|min:1',
            'date_of_birth' => 'required|string|date_format:"j M,
Y"|after_or_equal:"1 January 1900"|before_or_equal:"31 December 2009"|max:70',
        ]);

        try {
            $user = User::create([
                    'first_name' => $request->first_name,
                    'last_name' => $request->last_name,
                    'email' => $request->email,
                    'password' => bcrypt($request->password),
                    'date_of_birth' => $request->date_of_birth,
                ]);

            $user->roles()->attach($request->role);
        } catch (\Exception $e) {
            return response()->json(['message' => 'User is not created'],
500);
        }

        return response()->json(['message' => 'Successfully added new user!'],
200);
    }
```

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

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function updateBasic(Request $request)
    {
        $request->validate([
            'user_id' => 'required|integer',
            'first_name' => 'required|string|max:100',
            'last_name' => 'required|string|max:100',
            'email' => 'required|string|email|max:100|unique:users,email,' .
$request->user_id,
            'date_of_birth' => 'required|string|date_format:"j M,
Y"|after_or_equal:"1 January 1900"|before_or_equal:"31 December 2009"|max:70',
            'role_id' => 'required|integer|min:1'
        ]);

        try {
```

```
            $user = User::findOrFail($request->user_id);
            $user->update([
                'first_name' => $request->first_name,
                'last_name' => $request->last_name,
                'email' => $request->email,
                'date_of_birth' => $request->date_of_birth
            ]);

            $user->roles()->update(['role_id' => $request->role_id]);
        } catch (\Exception $e) {
            return response()->json(['message' => 'User is not updated'],
500);
        }

        return response()->json(['message' => 'Successfully updated user!'],
200);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function updateContact(Request $request)
    {
        $request->validate([
            'user_id' => 'required|integer',
            'phone_number' => 'required|digits:10',
            'country' => 'required|string|min:2|max:100',
            'city' => 'required|string|min:2|max:100',
            'zip' => 'required|digits:5',
```

```
            'address' => 'required|string|min:2|max:100',
        ]);

        try {
            $user = User::findOrFail($request->user_id);

            $user->phone_number = $request->phone_number;
            $user->save();

            $user->getContactInfo()->update([
                'country' => $request->country,
                'city' => $request->city,
                'zip' => $request->zip,
                'address' => $request->address
            ]);
        } catch (\Exception $e) {
            return response()->json(['message' => 'User is not updated'],
500);
        }

        return response()->json(['message' => 'Successfully updated user
contact info!'], 200);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function updatePassword(Request $request)
    {
```

```
        $request->validate([
            'user_id' => 'required|integer',
            'password' => 'required|string|max:50|min:10',
        ]);

        try {
            User::where('id', $request->user_id)
            ->update(['password' => bcrypt($request->password)]);
        } catch (\Exception $e) {
            return response()->json(['message' => 'User is not updated'],
500);
        }

        return response()->json(['message' => 'Successfully updated user
password!'], 200);
    }

    public function updatePicture(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
            'picture' => 'required|image',
        ]);
        $user = User::findOrFail($request->id);
        $previousPicture = $user->picture;

        $picture = $request->picture;
        if (!$picture->isValid()) {
            return response()->json(['errors' => ['picture' => ['Error with
uploading picture']]], 422);
        }
```

```
        if (File::exists($previousPicture) && $previousPicture !=
'images/user_images/default-profile-picture.png') {
            File::delete($previousPicture);
        }

        $newPictureName = time() . '.' . $picture-
>getClientOriginalExtension();
        $uploadedPicture = $picture->move('images/user_images',
$newPictureName);
        $newPicturePath = $uploadedPicture->getPath() . '/' . $newPictureName;

        $user->picture = $newPicturePath;
        $user->save();

        return response()->json(['message' => 'Successfully updated user
picture!'], 200);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request)
    {
        $request->validate([
            'id' => 'required|integer',
        ]);

        try {
            User::destroy($request->id);
```

```
        } catch (\Exception $e) {
            return response()->json(['message' => 'User is not deleted'],
500);
        }

        return response()->json(['message' => 'Successfully deleted user!'],
200);
    }
}
```

RegisterController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\User;
use App\Role;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
-
    | Register Controller
    |--------------------------------------------------------------------------
-
    |
    | This controller handles the registration of new users as well as their
```

```
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;

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

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

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

```
        return Validator::make($data, [
            'first_name' => 'required|string|max:100',
            'last_name' => 'required|string|max:100',
            'email' => 'required|string|email|max:100|unique:users',
            'date_of_birth' => 'required|string|date_format:"j M,
Y"|after_or_equal:"1 January 1900"|before_or_equal:"31 December 2009"|max:70',
            'password' => 'required|string|min:10|confirmed',
        ]);
    }

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

        $user->roles()->attach(Role::where('name', 'customer')->first());

        return $user;
    }
}
```

LoginController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

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

```
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

AdminAuth.php (Middleware)

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Auth;

class AdminAuth
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next, $guard = null)
    {
        if (Auth::check() && Auth::user()->isAdmin()) {
```

```
            return $next($request);
        } elseif ($request->ajax()) {
            return response()->json(['message' => 'Session has expired, please sign in !'], 401);
        }

        return redirect('/');
    }
}
```

Rules/IsDifferentFromOldPassword.php

```
<?php

namespace App\Rules;

use Illuminate\Contracts\Validation\Rule;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Auth;

class IsDifferentFromOldPassword implements Rule
{
    /**
     * Create a new rule instance.
     *
     * @return void
     */
    public function __construct()
    {
        //
```

```
    }

    /**
     * Determine if the validation rule passes.
     *
     * @param  string  $attribute
     * @param  mixed  $value
     * @return bool
     */
    public function passes($attribute, $value)
    {
        return !Hash::check($value, Auth::user()->password);
    }

    /**
     * Get the validation error message.
     *
     * @return string
     */
    public function message()
    {
        return 'Your new password is same as your old one!';
    }
}
```

Rules/IsOldPassword.php

```
<?php

namespace App\Rules;

use Illuminate\Contracts\Validation\Rule;
```

```php
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Auth;

class IsDifferentFromOldPassword implements Rule
{
    /**
     * Create a new rule instance.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }

    /**
     * Determine if the validation rule passes.
     *
     * @param  string  $attribute
     * @param  mixed  $value
     * @return bool
     */
    public function passes($attribute, $value)
    {
        return !Hash::check($value, Auth::user()->password);
    }

    /**
     * Get the validation error message.
     *
     * @return string
     */
```

```
    public function message()
    {
        return 'Your new password is same as your old one!';
    }
}
```

Mail/ContactMessage.php

```
<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class ContactMessage extends Mailable
{
    use Queueable, SerializesModels;
    public $messageData;
    public $date;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct($message)
    {
        $this->messageData = $message;
        $this->date = date('j M, Y H:i');
    }
```

```
    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->from('admin@aleksajovanovic.com')-
>view('mails.contactMessage');
    }
}
```

Listeners/InitializeCart.php

```
<?php

namespace App\Listeners\Users;

use App\Order;

class InitializeCart
{
    /**
     * Create the event listener.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }
```

```
    /**
     * Handle the event.
     *
     * @param  object  $event
     * @return void
     */
    public function handle($event)
    {
        $orderCart = Order::firstOrCreate(['user_id' => $event->user->id,
'status' => 'inactive']);
        session(['cart' => $orderCart->id]);
    }
}
```

Listeners/UpdateLastActivity.php

```
<?php

namespace App\Listeners\Users;

use Carbon\Carbon;

class UpdateLastActivity
{
    /**
     * Create the event listener.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }
```

```
    /**
     * Handle the event.
     *
     * @param  object  $event
     * @return void
     */
    public function handle($event)
    {
        $event->user->last_online = Carbon::now();
        $event->user->save();
    }
}
```

MyClasses (Helper klase koje sam pisao)

Breadcrumbs.php

```
<?php

namespace App\MyClasses;

class Breadcrumbs
{
    public static function generate()
    {
        $breadCrumb = [];
        $breadCrumbs = [];

        foreach (request()->segments() as $segment) {
            $breadCrumb[] = $segment;
            $breadCrumbs[title_case($segment)] = '/' . implode('/',
$breadCrumb);
        }
```

```
        return  $breadCrumbs ;
    }
}
```

DBBulkUpdate.php

```
<?php

namespace App\MyClasses;

class DBBulkUpdate
{
    protected $table;
    protected $columnsToUpdate;
    protected $data;

    public function __construct($table, $data)
    {
        $this->table = $table;
        $this->data = $data;
    }

    public function setColumnsToUpdate(array $columns)
    {
        $columnCollection = collect($columns);
        $this->columnsToUpdate = implode(',', $columnCollection->map(function
($value) {
            return "$value=VALUES($value)";
        })->toArray());
    }

    public function questionMarks()
```

```
    {
        $placeholder = implode(',', $this->data[0]->map(function () {
            return '?';
        })->toArray());

        return implode(',', array_fill(0, $this->data->count(),
"($placeholder)"));
    }

    public function columnNames()
    {
        return implode(',', $this->data[0]->keys()->toArray());
    }

    public function generateQuery()
    {
        return "INSERT INTO {$this->table}({$this->columnNames()}) VALUES
{$this->questionMarks()} ON DUPLICATE KEY UPDATE $this->columnsToUpdate";
    }
}
```

JsonSliderHelper.php

```
<?php

namespace App\MyClasses;

class JsonSliderHelper
{
    private $collection;
```

```php
public function __construct($json)
{
    $this->collection = collect(json_decode($json));
}

public function addSlide($slideData)
{
    $slideToObject = json_decode(json_encode($slideData));

    $this->collection->push($slideToObject);
    $this->updateId();

    return $this;
}

public function updateSlide($slideData)
{
    $this->collection->each(function ($item, $key) use ($slideData) {
        if ($item->id == $slideData['id']) {
            $item->head = $slideData['head'];
            $item->slogan = $slideData['slogan'];
            $item->order = $slideData['order'];
            $item->captionPos = $slideData['captionPos'];
            $item->img = $slideData['img'];
        }
    });
    $this->updateId();

    return $this;
}
```

```
    public function removeSlide($id)
    {
        $this->collection = $this->collection->filter(function ($item, $key)
use ($id) {
            return $item->id != $id;
        })->values();
        $this->updateId();

        return $this;
    }

    protected function updateId()
    {
        $this->collection->each(function ($item, $key) {
            $item->id = $key + 1;
        });
    }

    public function write()
    {
        $file = fopen(storage_path('/app/slider.json'), 'w+');
        fwrite($file, $this->collection->toJson(JSON_PRETTY_PRINT));
        fclose($file);

        return  $this->collection;
    }
}
```

LogMessagesHelper.php

```
<?php
```

```php
namespace App\MyClasses;

use Monolog\Logger;
use Monolog\Handler\StreamHandler;
use Monolog\Formatter\LineFormatter;

class LogMessagesHelper
{
    private $log;

    public function __construct()
    {
        $this->log = new Logger('custom_info_logger');
        $stream = new StreamHandler(storage_path() .
'/logs/custom_info_log.log', Logger::INFO);

        $formatter = new LineFormatter("[%datetime%] %message% %context% \n",
'Y-m-d H:i:s');

        $stream->setFormatter($formatter);
        $this->log->pushHandler($stream);
    }

    public function write($message, $context = [])
    {
        $this->log->info($message, $context);
    }
}
```

PollFileHelper.php

```php
<?php
```

```php
namespace App\MyClasses;

class PollFileHelper
{
    private $filePath;
    private $poll_id;
    private $user_id;

    public function __construct($data)
    {
        $this->poll_id = $data['poll_id'];
        $this->user_id = $data['user_id'];

        $this->filePath = storage_path("/app/polls_{$this->poll_id}.txt");
    }

    private function generateRow()
    {
        return  $this->poll_id . "\t" . $this->user_id . "\n";
    }

    public function alreadyVoted()
    {
        try {
            $fileData = file($this->filePath);
        } catch (\Exception $e) {
            return false;
        }

        foreach ($fileData as $row) {
            $ids = explode("\t", $row);
```

```
            if ($ids[0] == $this->poll_id && $ids[1] == $this->user_id) {
                return true;
            }
        }

        return false;
    }

    public function write()
    {
        $file = fopen($this->filePath, 'ab');
        fwrite($file, $this->generateRow());
        fclose($file);
    }

    public static function createFile($id)
    {
        $file = fopen(storage_path("/app/polls_$id.txt"), 'ab');
        fwrite($file, '');
        fclose($file);
    }
}
```