Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Web-технологии”

Лабораторная работа №5

“Исследование возможностей фреймворка Laravel

для разработки веб-приложений”

Выполнил:

ст.гр. ИС/б-17-2

Долженко И.А.

Проверил:

Овчинников А.Л.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Ознакомиться с возможностями фреймворка Laravel и приобрести практические навыки реализации веб-приложений с его помощью.

2 ПОСТАНОВКА ЗАДАЧИ

В ходе выполнения лабораторной работы необходимо с использованием фреймворка Laravel реализовать «Редактор Блога» и «Загрузка сообщений блога» в разделе администратора персонального сайта и «Мой Блог» в пользовательском разделе.

Предусмотреть возможность редактирования и удаления записей блога на странице «Редактор Блога», реализовав валидацию средствами Laravel.

Для этого разработать: миграцию, Eloquent модель, маршруты роутинга, контроллеры, валидации, Blade-макеты пользовательской и административной части сайта, а также необходимые представления на основе имеющихся HAML/SCSS файлов.

3 ХОД РАБОТЫ

Первым делом создадим файлы миграции, в которых опишем поля соответствующих таблиц в базе данных.

Файл миграции для постов:

class CreatePostsTable extends Migration

{

*/\*\**

*\* Run the migrations.*

*\**

*\* @return void*

*\*/*

    public function up()

    {

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

            $table->id();

            $table->string('title');

            $table->string('image')->nullable();

            $table->string('text');

            $table->timestamps();

        });

    }

*/\*\**

*\* Reverse the migrations.*

*\**

*\* @return void*

*\*/*

    public function down()

    {

        Schema::dropIfExists('posts');

    }

}

Файл миграции для комментариев:

class CreateCommentsTable extends Migration

{

*/\*\**

*\* Run the migrations.*

*\**

*\* @return void*

*\*/*

    public function up()

    {

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

            $table->id();

            $table->bigInteger('post\_id')->unsigned();

            $table->string('fullname');

            $table->string('comment');

            $table->foreign('post\_id')->references('id')->on('posts')->onDelete('cascade');

            $table->timestamps();

        });

    }

*/\*\**

*\* Reverse the migrations.*

*\**

*\* @return void*

*\*/*

    public function down()

    {

        Schema::dropIfExists('comments');

    }

}

Затем опишем роуты в файле routes/web.php:

Route::get('/', function () {

*return* view('welcome');

})->name("main");

Route::get('/about', function () {

*return* view('about');

})->name("about");

Route::resource('blog', 'PostController')->names('posts');

Route::post('/comment/create', 'CommentController@create')->name('api.comment.create');

Route::patch('/post/update', 'PostController@update')->name('api.post.update');

Auth::routes();

Route::get('/home', function(){

*return* redirect()->route('main');

})->name('home');

Route::get('/logout', '\App\Http\Controllers\Auth\LoginController@logout');

Содержимое основного layout’а, в котором будем выводить другие представления, представлено ниже:

<!doctype *html*>

<html *lang*="ru">

<head>

    <meta *charset*="utf-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1">

    <meta *name*="csrf-token" *content*="{{ csrf\_token() }}">

    <title>@yield('title')</title>

    <link *href*="{{ asset('css/bootstrap.min.css') }}" *rel*="stylesheet">

    <link *href*="{{ asset('css/main.css') }}" *rel*="stylesheet">

</head>

<body>

    <nav *class*="navbar navbar-expand-md navbar-dark sticky-top" *style*="background-color: #563d7c">

        <div *class*="container">

            <button *class*="navbar-toggler" *type*="button" *data-toggle*="collapse" *data-target*="#navbar" *aria-controls*="navbar" *aria-expanded*="false" *aria-label*="Toggle navigation">

                <span *class*="navbar-toggler-icon"></span>

            </button>

            <div *class*="collapse navbar-collapse" *id*="navbar">

                <ul *class*="navbar-nav">

                    <li *class*="nav-item">

                        <a *class*="nav-link {{ Request::is('/') ? 'active' : '' }}" *href*="{{ route('main') }}">Главная</a>

                    </li>

                    <li *class*="nav-item">

                        <a *class*="nav-link {{ Request::is('about') ? 'active' : '' }}" *href*="{{ route('about') }}">Обо мне</a>

                    </li>

                    <li *class*="nav-item">

                        <a *class*="nav-link {{ Request::is('blog') ? 'active' : '' }}" *href*="{{ route('posts.index') }}">Мой блог</a>

                    </li>

                </ul>

                <ul *class*="navbar-nav ml-auto">

                    @if (!auth()->check())

                        <li *class*="nav-item">

                            <a *class*="nav-link {{ Request::is('login') ? 'active' : '' }}" *href*="{{ route('login') }}">Вход</a>

                        </li>

                    @else

                        <li *class*="nav-item">

                            <div *class*="nav-link active">Пользователь: {{ Auth::user()->name }}</div>

                        </li>

                    @endif

                    @if (!auth()->check())

                        <li *class*="nav-item">

                            <a *class*="nav-link {{ Request::is('register') ? 'active' : '' }}" *href*="{{ route('register') }}">Регистрация</a>

                        </li>

                    @else

                        <li *class*="nav-item">

                            <a *class*="nav-link {{ Request::is('logout') ? 'active' : '' }}" *href*="{{ url('/logout') }}">Выход</a>

                        </li>

                    @endif

                </ul>

            </div>

        </div>

    </nav>

    @yield('content')

</body>

</html>

Содержимое файла представления страницы блога:

@extends('layout.header')

@section('title', 'Блог')

@section('content')

<script *src*="{{ 'js/sendComment.js' }}"></script>

<section *class*="content container">

    @if (Auth::user() && Auth::user()->isAdmin)

        <form *action*="{{ route('posts.store') }}" *method*="POST" *enctype*="multipart/form-data" *class*="mb-4">

            @csrf

            <div *class*="form-group">

                <input *type*="text" *class*="form-control tooltip-input" *value*="{{ old('title') }}" *name*="title" *placeholder*="Тема сообщения" *autocomplete*="off">

            </div>

            <div *class*="form-group">

                <div *class*="custom-file">

                    <input *type*="file" *name*="image" *accept*=".png, .jpeg, .jpg" *class*="form-control" *id*="uploadInput" *aria-describedby*="uploadInputBtn">

                    <label *class*="custom-file-label" *for*="uploadInput">Загрузите изображение</label>

                </div>

            </div>

            <div *class*="form-group">

                <textarea *class*="form-control" *name*="text" *rows*="4" *placeholder*="Введите сообщение">{{ old('text') }}</textarea>

            </div>

            <button *class*="btn btn-primary" *type*="submit">Добавить</button>

        </form>

        @foreach ($errors->all() as $error)

            <div>{{ $error }}</div>

        @endforeach

    @endif

    <h2 *class*="mb-4 mt-4">Блог</h2>

    <div *class*='posts-block'>

        @if ($data->isEmpty())

            <h6>Нет постов</h6>

        @endif

        @foreach ($data as $item)

            <div *class*="card card-blog mb-4">

                <div *class*="row no-gutters">

                    <div *class*="col-md-4">

                        <img *src*="{{ $item->image ? $item->image : 'img/not-found.png' }}" *class*="card-img">

                    </div>

                    <div *class*="col-md-8 d-flex flex-column">

                        <div *class*="card-body">

                            <h5 *class*="card-title">{{ $item->title }}</h5>

                            <p *class*="card-text">{{ $item->text }}</p>

                            @if (Auth::user() && Auth::user()->isAdmin)

                                <button

*class*="btn edit-btn"

*type*="button"

*data-id*="{{ $item->id }}"

*data-title*="{{ $item->title }}"

*data-text*="{{ $item->text }}"

                                >

                                    <img *src*="{{ 'img/edit.svg' }}"/>

                                </button>

                                <form *action*="{{ route('posts.destroy', $item->id) }}" *method*="POST">

                                    @csrf

                                    @METHOD("DELETE")

                                    <button *class*="btn delete-btn" *type*="submit">

                                        <img *src*="{{ 'img/trash.svg' }}"/>

                                    </button>

                                </form>

                            @endif

                        </div>

                        <div *class*="card-footer">

                            <small *class*="text-muted">{{ $item->created\_at }}</small>

                        </div>

                    </div>

                </div>

                <div *class*="card-comment">

                    <h6 *class*="card-comment\_\_title">

                        Комментариев: {{ count($item->comments) }}

                    </h6>

                    <div *class*="card-comment\_\_container">

                        @foreach ($item->comments as $comment)

                            <div *class*="comment-item">

                                <div *class*="d-flex">

                                    <div *class*="comment-item\_\_name">{{ $comment->fullname }},</div>

                                    <div *class*="comment-item\_\_date">{{ $comment->created\_at }}</div>

                                </div>

                                <div *class*="comment-item\_\_text">{{ $comment->comment }}</div>

                            </div>

                        @endforeach

                    </div>

                    @if (auth()->check())

                        <div *class*="card-comment\_\_add">

                            <div *class*="input-group">

                                <input *data-id*="{{ $item->id }}" *type*="text" *class*="form-control" *name*="comment" *placeholder*="Введите ваш комментарий" *aria-describedby*="send-btn" *autocomplete*="off">

                                <div *class*="input-group-append">

                                    <button

*class*="btn btn-outline-secondary"

*type*="button"

*id*="send-btn"

*onClick*="send({{ $item->id }})"

                                    >

                                        Отправить

                                    </button>

                                </div>

                            </div>

                        </div>

                    @endif

                </div>

            </div>

        @endforeach

    </div>

    <div *class*="d-flex justify-content-center">

        @if ($data->total() > $data->count())

            {{ $data->links() }}

        @endif

    </div>

</section>

<div *class*="modal hidden">

    <div *class*="container">

        <div *class*="body">

            <h4 *class*="text-center mb-4">Редактирование</h4>

            <form>

                <div *class*="form-group">

                    <input *id*="editTitle" *type*="text" *class*="form-control tooltip-input" *placeholder*="Тема сообщения" *autocomplete*="off">

                </div>

                <div *class*="form-group">

                    <textarea *id*="editMessage" *class*="form-control" *rows*="4" *placeholder*="Введите сообщение"></textarea>

                </div>

                <button *id*="saveBtn" *class*="btn btn-primary w-100" *type*="button">Сохранить</button>

            </form>

        </div>

    </div>

</div>

<script>

    const inputFile = document.getElementById("uploadInput");

    const inputLabel = document.getElementsByClassName("custom-file-label")[0];

    inputFile.addEventListener("change", e => {

        inputLabel.innerHTML = e.target.files[0].name;

    });

</script>

<script *src*="{{ 'js/editBlog.js' }}"></script>

@endsection('content')

Страница блога для администратора:

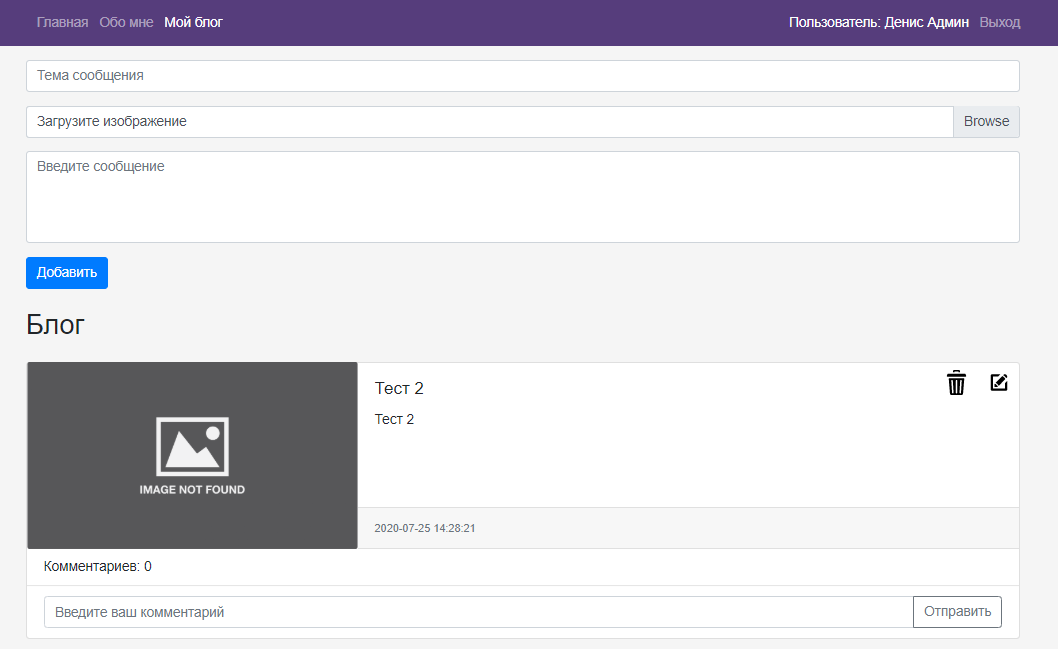


Рисунок 1 – Страница блога для администратора

Страница блога для неавторизованного пользователя:

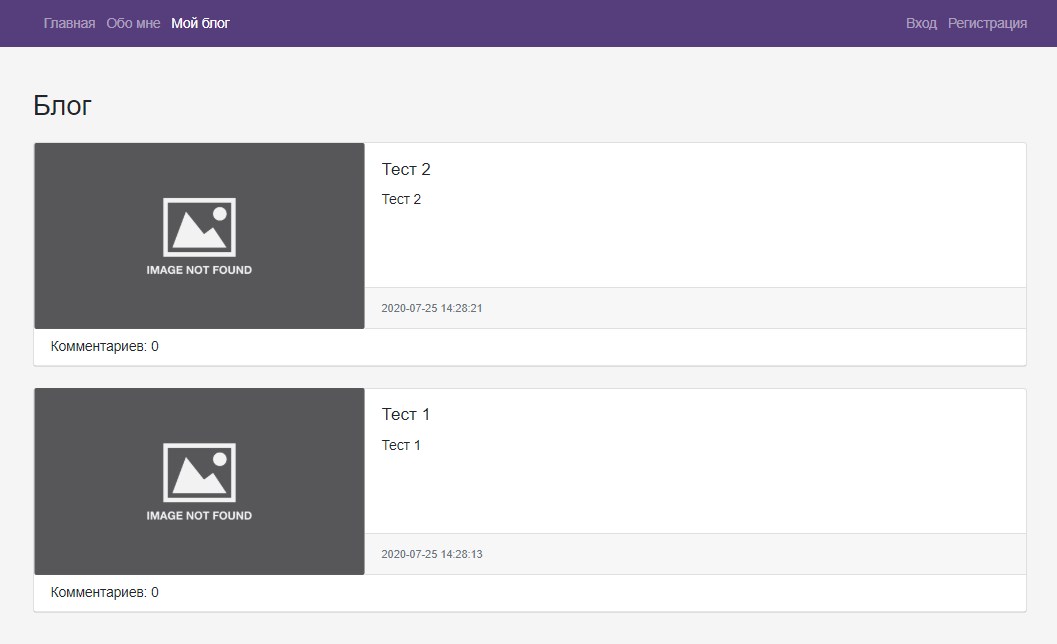


Рисунок 2 – Страница блога для неавторизованного пользователя

Результат добавления комментария авторизованным пользователем:

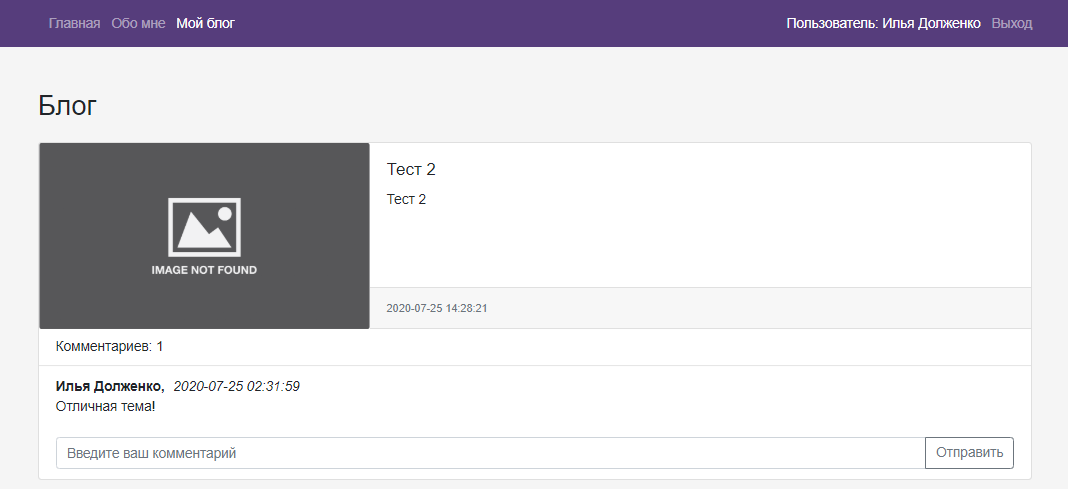


Рисунок 3 – Содержимое страницы после добавления комментария

Для создания комментария в его контроллере описан метод create:

class CommentController extends Controller

{

    public function create(Request $request) {

        $date = date('m/d/Y h:i:s', time());

        $item = new Comment();

        $item->fullname = Auth::user()->name;

        $item->created\_at = $date;

        $item->updated\_at = $date;

        $item->fill($request->input());

        $item->save();

*return* response()->json($item);

    }

}

Администратор имеет возможность редактировать и удалять посты, для этого в контролере PostController содержатся методы update и destroy, а для отображения страницы с постами и их добавления методы index и store соответственно:

class PostController extends Controller

{

    public function index()

    {

        $data = Post::orderBy('created\_at', 'desc')->with(['comments'])->paginate(3);

*return* view('blog', compact('data'));

    }

    public function store(postRequest $request)

    {

        $data = $request->input();

        $item = (new Post()) -> fill($data);

*if* ($request->hasFile('image')) {

            $imageName = time() . '.' . request()->image->getClientOriginalExtension();

            request()->image->move(public\_path('img/uploads'), $imageName);

            $item->image = 'img/uploads/' . $imageName;

        }

        $item->save();

*return* redirect()->route("posts.index");

    }

    public function update(postRequest $request)

    {

        $data = $request->input();

        $item = Post::findOrFail($data['id']);

        $item->update($data);

    }

    public function destroy($id)

    {

        Post::findOrFail($id)->delete();

*return* redirect('/blog');

    }

}

Процесс редактирования постов представлен ниже:

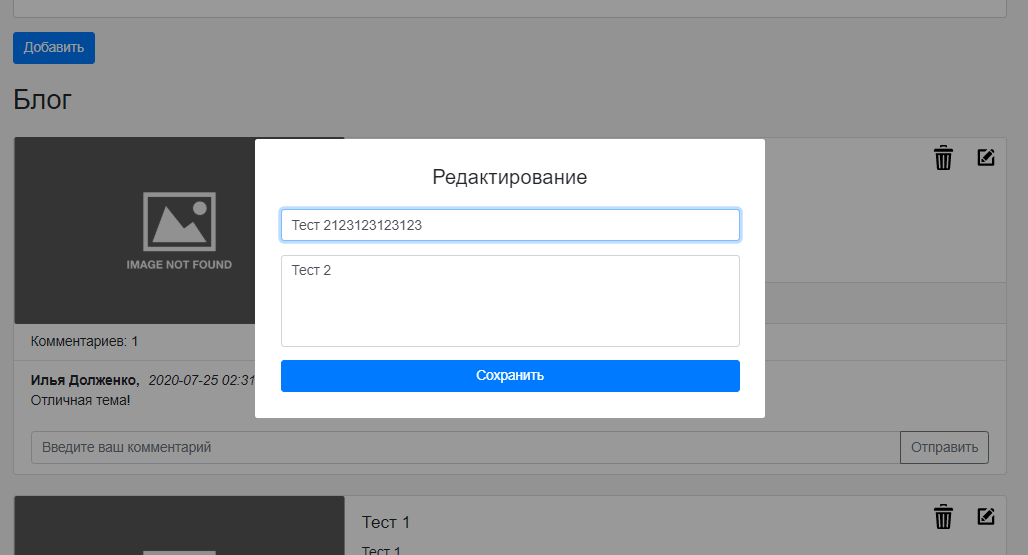


Рисунок 4 – Процесс редактирования поста

В форме добавления нового поста присутствует валидация, которая не позволяет отправить пустую форму, либо форму с поляем, содержащими меньше 3 символов. Данная форма представлена на рисунке 5.

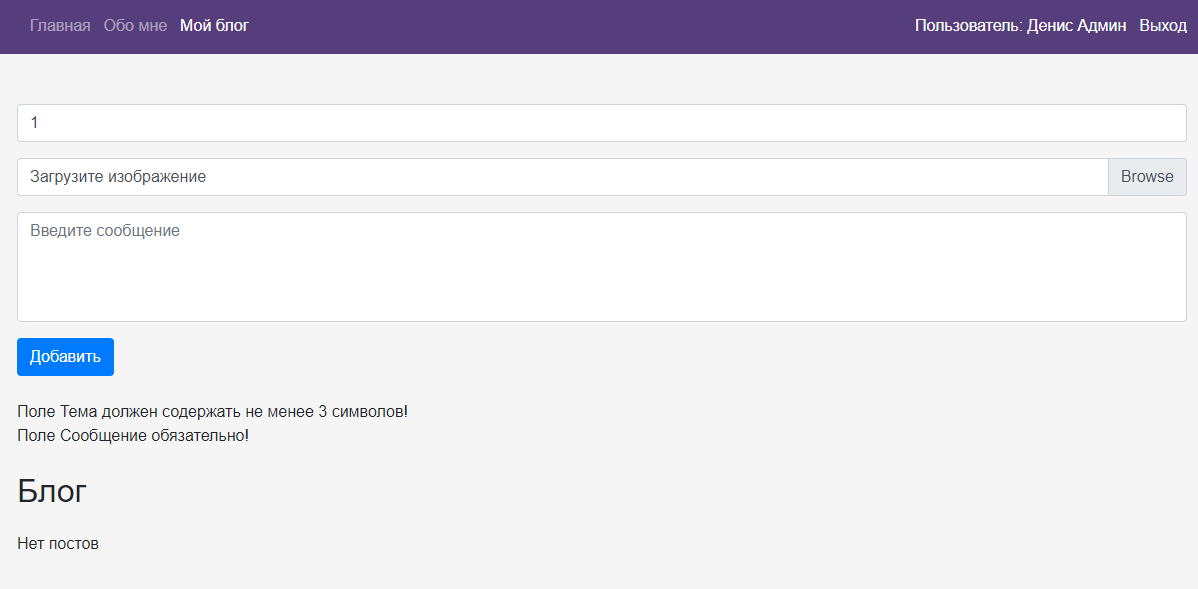


Рисунок 5 – Результат отправки формы, присутствует валидация

ВЫВОД

В ходе выполнения лабораторной работы были изучены возможности фреймворка Laravel и приобретены практические навыки реализации веб-приложений с его помощью. Был переписан сайт, созданный в прошлых лабораторных работах, с помощью фреймворка Laravel. В работе были применены средства самого фреймворка для создания авторизации, были проведены миграции, с помощью команд были созданы контроллеры и их модели.