МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Гусев Владислав Яковлевич

Институт информационных технологий и управления в технических системах

курс 3 группа ГИС/б-32-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №5

по дисциплине: «Веб-технологии»

по теме: «Исследование возможностей фреймворка Laravel

для разработки веб-приложени»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. пр. Забаштанский А.К.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2019

1. ЦЕЛЬ РАБОТЫ

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

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

В ходе выполнения лабораторной работы необходимо с использова-нием фреймворка Laravel реализовать «Редактор Блога» и «Загрузка со-общений блога» в разделе администратора персонального сайта и «Мой Блог» в пользовательском разделе (см. ЛР №6).

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

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

1. ХОД РАБОТЫ

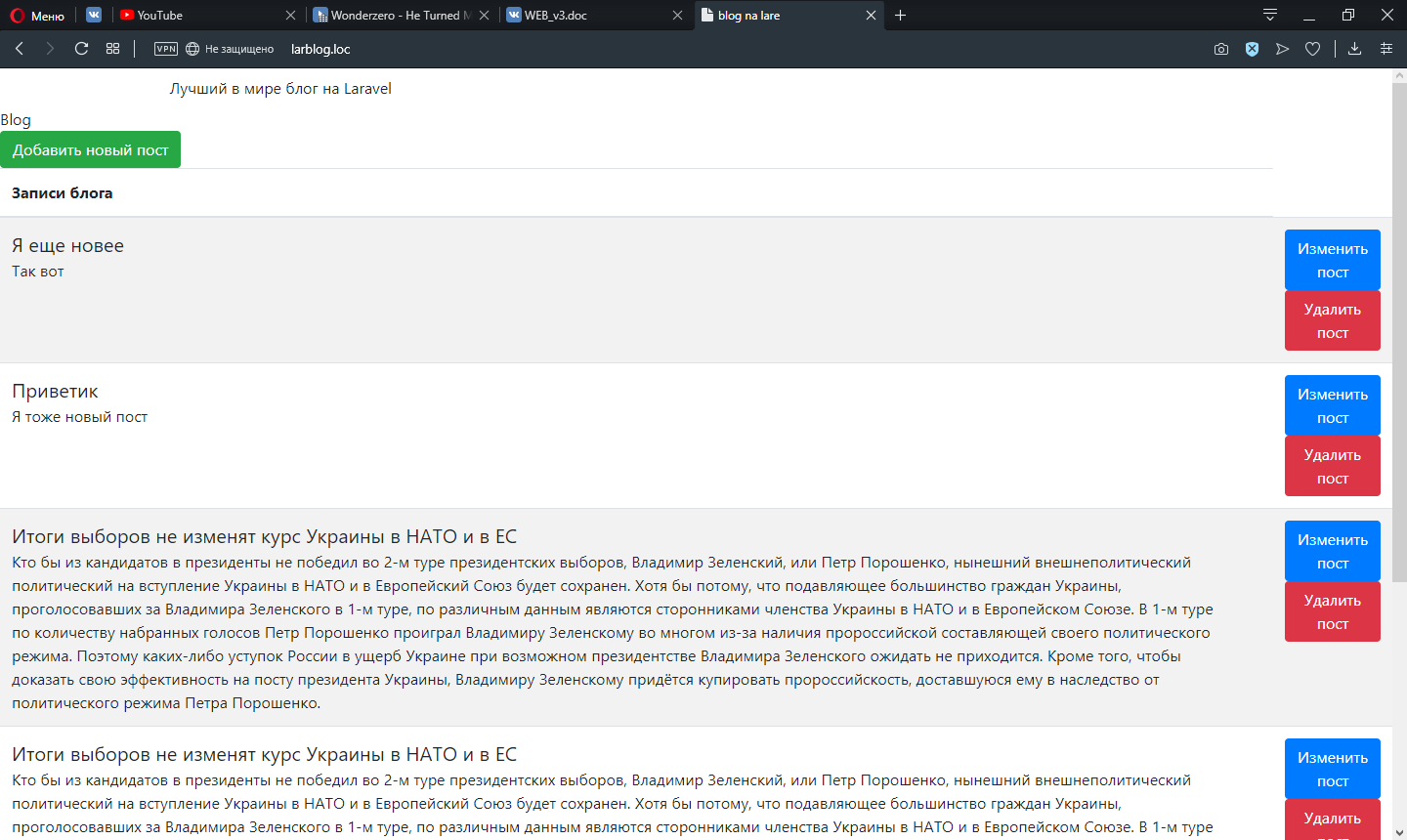


Рисунок 1 – Общий вид блога

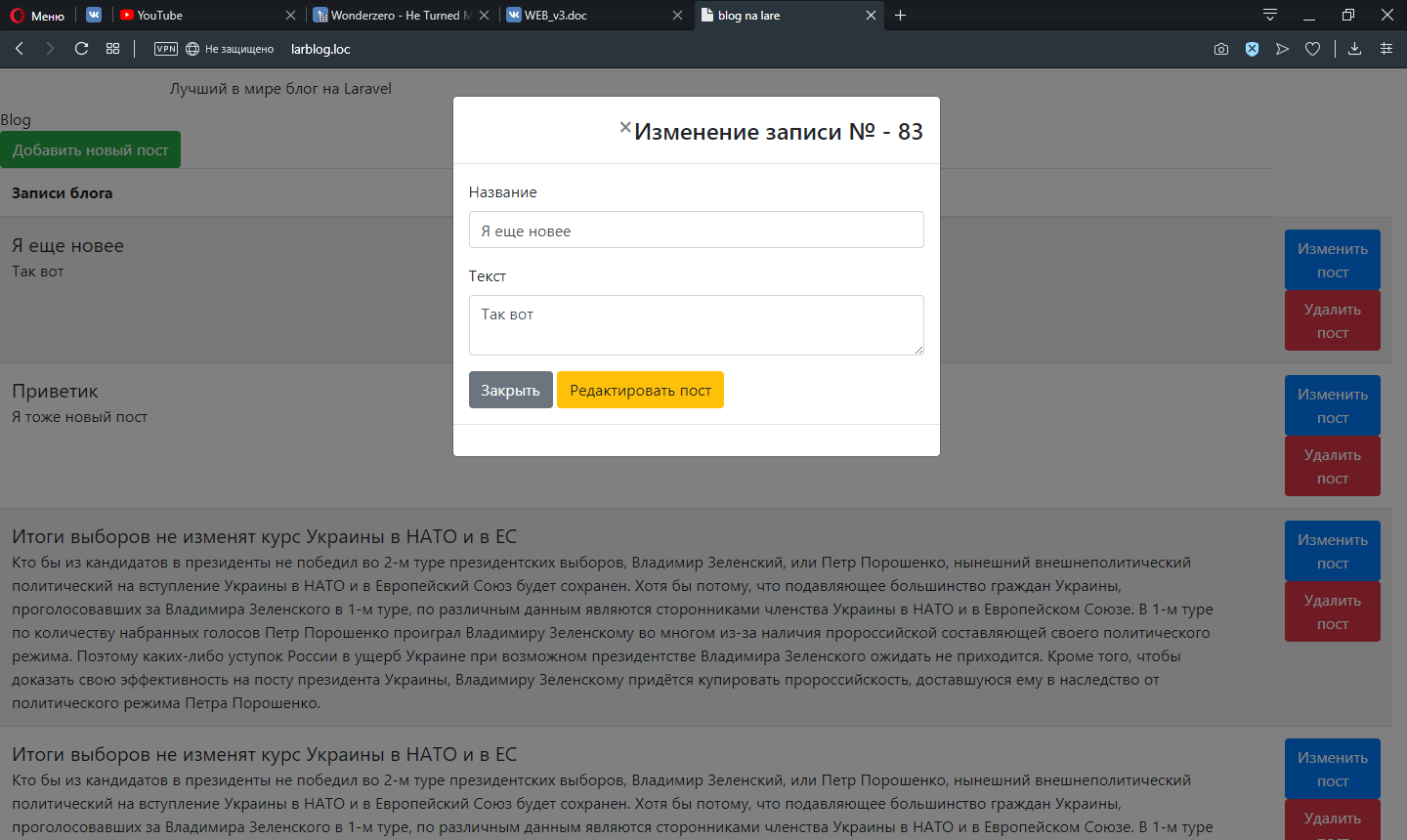


Рисунок 2 – Изменение записи

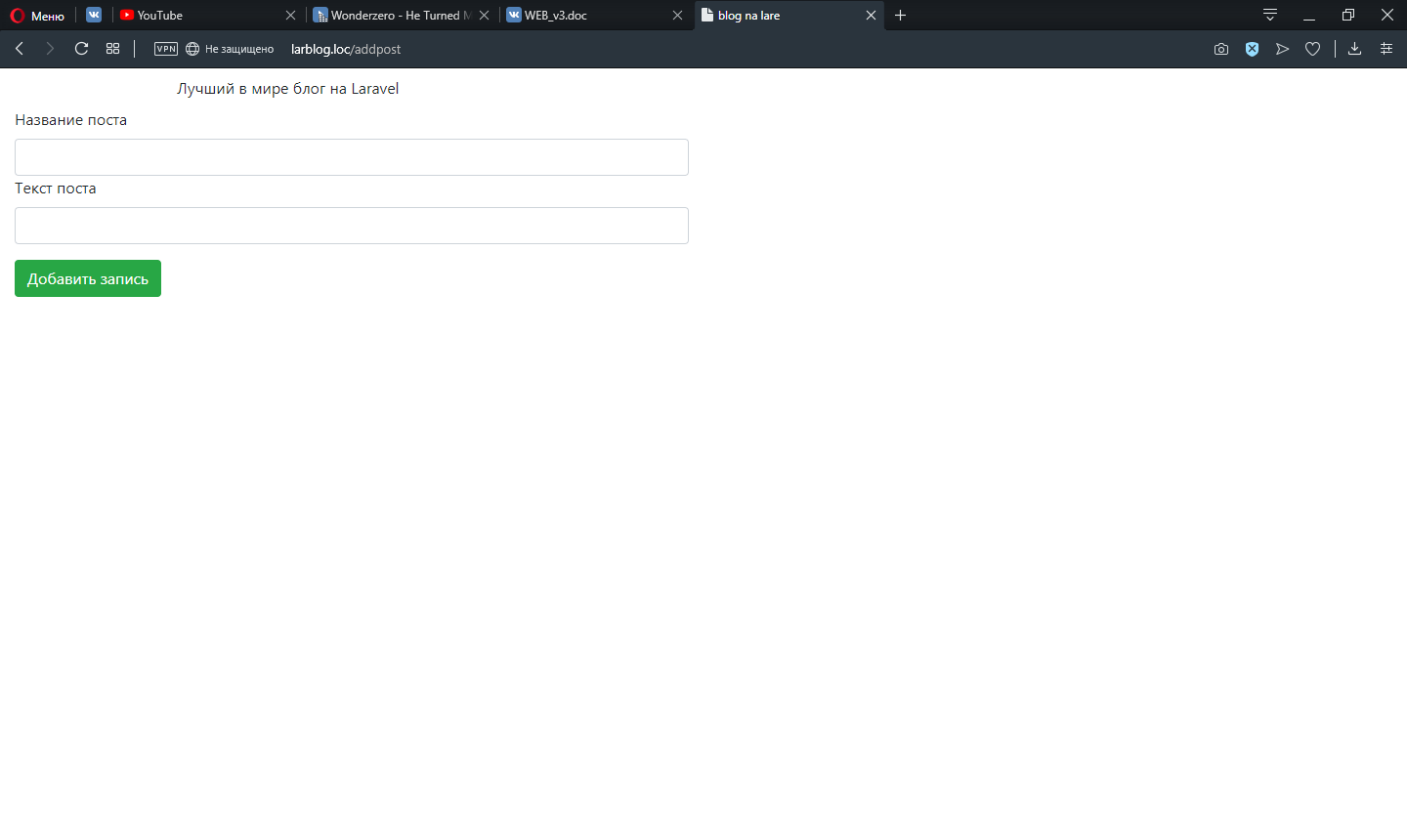


Рисунок 3 –Вид формы добавления записи блога

Класс routes

<?php

use App\post;

use Illuminate\Http\Request;

/\*\*

\* Вывести список всех задач

\*/

Route::get('/', function () {

$posts = post::orderBy('created\_at', 'desc')->paginate(5);

return view('posts', [

'posts' => $posts

]);

});

/\*\*

\* Добавить новую задачу

\*/

Route::get('/addpost', function (Request $request) {

return view('addpost');

});

Route::post('addpost/post', function (Request $request) {

$validator = Validator::make($request->all(), [

'title' => 'required|max:255',

]);

if ($validator->fails()) {

return redirect('/')

->withInput()

->withErrors($validator);

}

$post = new post;

$post->title = $request->title;

$post->text = $request->text;

$post->save();

return redirect('/');

});

/\*\*

\* Редактировать задачу

\*/

Route::get('/editpost/post/{id}', function (Request $request) {

$validator = Validator::make($request->all(), [

'title' => 'required|max:255',

]);

if ($validator->fails()) {

return redirect('/')

->withInput()

->withErrors($validator);

}

$post = post::where('id', '=', $request->id)->first();;

$post->title = $request->title;

$post->text = $request->text;

$post->save();

return redirect('/');

});

/\*\*

\* Удалить существующую задачу

\*/

Route::delete('/post/{id}', function ($id) {

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

return redirect('/');

});Класс coment\_model

<?php

include\_once 'app/core/BaseActiveRecord.php';

class CommentsModel extends BaseActiveRecord

{

// список полей таблицы

public $id;

public $name\_user;

public $id\_blog;

public $text;

public static $table = 'users\_comments';

public static function GetParams(){

$st="(id,

id\_user,

id\_blog,

text)";

return $st;

}

}

// делаем PDO доступным для всех AR классов

BaseActiveRecord::$pdo = $pdo;

// получаем запись из БД с id = 1

//$test = TestModel::find(1);

//var\_dump($test->id);

//var\_dump($test->group);

//var\_dump($test->first);

//var\_dump($test->three);

//var\_dump($test);

Класс миграции для блога

<?php

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateBlogTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->increments('id');

$table->string('title',100);

$table->string('text',1000);

$table->string('img');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::drop('posts');

}

}

Blade-шаблон страницы

@extends('layouts.app')

@section('content')

<!-- Форма создания задачи... -->

<!-- Текущие задачи -->

@if (count($posts) > 0)

<div class="panel panel-default">

<div class="panel-heading">

Blog

</div>

<div class="panel-body">

<a href="/addpost"><button type="button" class="btn btn-success">Добавить новый пост</button></a>

<table class="table table-striped task-table">

<!-- Заголовок таблицы -->

<thead>

<th>Записи блога</th>

<th>&nbsp;</th>

</thead>

<!-- Тело таблицы -->

<tbody>

@foreach ($posts as $post)

<tr>

<!-- Имя задачи -->

<td class="table-text">

<div style="font-size: 20px">{{ $post->title }}</div>

<div>{{ $post->text }}</div>

</td>

<td>

<!-- Кнопка Удалить -->

<td>

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal"

data-id="{{ $post->id }}"

data-title="{{ $post->title }}"

data-text="{{ $post->text }}"

>Изменить пост</button>

<form action="/post/{{ $post->id }}" method="POST">

{{ csrf\_field() }}

{{ method\_field('DELETE') }}

<button type="submit" class="btn btn-danger">Удалить пост</button>

</form>

</td>

</tr>

@endforeach

</tbody>

</table>

<?php echo $posts->appends(['sort' => 'votes'])->render(); ?>

</div>

<!-- Модальное окно -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

<h4 class="modal-title" id="exampleModalLabel">Редактировать</h4>

</div>

<div class="modal-body">

<form action="/editpost/post/{{$post->id}}" method="get">

{{ csrf\_field() }}

{{ method\_field('DELETE') }}

<div class="form-group">

<input type="text" name="id" id="id" value="" hidden>

<label for="recipient-name" class="form-control-label">Название</label>

<input type="text" id="name" name="title" class="form-control" id="recipient-name">

</div>

<div class="form-group">

<label for="message-text" class="form-control-label">Текст</label>

<textarea class="form-control" name="text" id="message-text"></textarea>

</div>

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="submit" class="btn btn-warning">Редактировать пост</button>

</form>

</div>

<div class="modal-footer">

</div>

</div>

</div>

</div>

</div>

<script type="text/javascript">

$("#exampleModal").on("show.bs.modal", function () {

var button = $(document.activeElement) // Кнопка, что спровоцировало модальное окно

var id = button.data('id');

var title = button.data('title') // Извлечение информации из данных-\* атрибутов

var text = button.data('text')

var modal = $(this)

modal.find('.modal-title').text('Изменение записи № - ' + id)

modal.find('.modal-body #id').val(id)

modal.find('.modal-body #name').val(title)

modal.find('.modal-body textarea').val(text)

})

</script>

@endif

@endsection

ВЫВОДЫ

В ходе выполнения лабораторной работы были с использованием фреймворка Laravel реализованы «Редактор Блога» и «Загрузка со-общений блога» в разделе администратора персонального сайта и «Мой Блог» в пользовательском разделе.