**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

|  |
| --- |
| Институт информационных технологий и управления в технических системах |
| (полное название института) |

|  |
| --- |
| кафедра «Информационные системы» |
| (полное название кафедры) |

**Пояснительная записка**

к курсовому проекту

по дисциплине «Проектный практикум»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| на тему | **Web-сайт команды озвучки** | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Выполнил: студент | | | IV | | | курса, группы: | | | | | **ИС/б-17-2-о** | | | | | |
|  | | | | | | | | | | | | | | | | |
| Направления подготовки (специальности) | | | | | | | | | | | 09.03.02 | | | | | |
|  | | | | | | | | | | | | | | | | |
| Информационные системы и технологии | | | | | | | | | | | | | | | | |
| (код и наименование направления подготовки (специальности)) | | | | | | | | | | | | | | | | |
| профиль (специализация) | | | | | |  | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| **Волобуев Юрий Сергеевич** | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество студента) | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Руководитель | | | Строганов В.А. | | | | | | | | | | | | | |
|  | | | (фамилия, инициалы, степень, звание, должность) | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Защита « | |  | | » |  | | | | 20 | | 21 | г. | | Оценка | |  |
|  | | | | | | | | | | | | | | | | |
| Руководитель | | | | | | |  | | | | | |  | |  | |
|  | | | | | | | (подпись) | | | | | |  | | (инициалы, фамилия) | |
|  | | | | | | | | | | | | | | | | |
| Ведущий преподаватель | | | | | | |  | | | | | |  | | Строганов В.А. | |
|  | | | | | | | (подпись) | | | | | |  | | (инициалы, фамилия) | |
|  | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | 21 | | г. | | | | | | |

**АННОТАЦИЯ**

В данной пояснительной записке представлено описание основных этапов выполнения курсового проекта по дисциплине «Проектный практикум». Курсовой проект посвящен разработке Web-сайта для команды озвучки. Приведено техническое задание, по которому реализован данный проект.

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ**

**1 Цели создания и целевая аудитория**

Цели создания сайта:

* Предоставление возможности свободно смотреть озвучки
* Обход блокировки по авторским правам на других площадках
* Создание инструмента для публикации актуальной информации о сроках выхода контента и предстоящих событиях.

Целевой аудиторией сайта является фандом мультсериала, заинтересованный в просмотре русской озвучки.

**2 Структура сайта**

В разрабатываемом Web-приложении предполагается два независимых интерфейса: интерфейсы пользователя и администратора.

Пользовательский интерфейс содержит:

* главную страницу с перечнями последних серий и новостей;
* страница с полным списком вышедших эпизодов;
* страница новостей;
* страницы просмотра видео с возможностью оставить комментарий.

Интерфейс администратора содержит:

− дополнительные панели для администрации видео и новостей;

− кнопки управления комментариями.

**3. Пожелания по сайту**

Все страницы разрабатываемого сайта должны быть выдержаны в одном стиле. Корпоративные цвета: белый, розовый, серый. Ширина: 1140px.

**4. Технические требования к сайту**

Сайт должен работать на основных современных браузерах (Google Chrome, Safari, Opera, Mozilla Firefox). Кроме того, сайт должен быть адаптивным: работать на устройствах с любым разрешением экрана (компьютерах, смартфонах, планшетах).

Инструменты для разработки:

* PHP
* Laravel
* Javascript

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 6

1. ПОСТАНОВКА ЗАДАЧИ 7

2. АНАЛИЗ ПРОБЛЕМНОЙ СИТУАЦИИ И ОБЗОР АНАЛОГОВ 8

2.1. Описание ситуации 8

2.2. Обзор аналогов 8

3. ПЛАН РЕАЛИЗАЦИИ ПРОЕКТА 9

4. ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 11

5. ОПИСАНИЕ РЕЗУЛЬТАТА 12

5.1. Реализация интерфейса 12

5.2. Реализация серверной части 16

6. ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО САЙТА 20

ЗАКЛЮЧЕНИЕ 24

СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ 25

ПРИЛОЖЕНИЕ А. ДИАГРАММА ГАНТА 26

ПРИЛОЖЕНИЕ Б. КОД ПРИЛОЖЕНИЯ 27

**ВВЕДЕНИЕ**

Данная работа состоит из пояснительной записки, включающей в себя аннотацию, содержание, введение, шесть разделов, заключение, список использованных источников и два приложения.

В первом разделе подробно описывается постановка задачи.

Во втором разделе был проведен анализ проблемной ситуации и обзор аналогов. Были выявлены проблемы, которые нас сервис может решить.

В третьем разделе описаны этапы работы на протяжении семестра.

В четвертом разделе были выбраны и описаны инструментальные средства, используемые при разработке сервиса.

В пятом разделе находится описание продуктового результата. Здесь описано состояние проекта на момент завершения семестра.

В шестом разделе было произведено тестирование разработанного сервиса на наличие валидации и правильной работы таких функций, как: просмотр видео, создание, удаление и редактирование видео и новостей, отправка комментариев и их удаление.

**1 ПОСТАНОВКА ЗАДАЧИ**

Определим основные функциональные требования для приложения:

* разрабатываемый Web-сайт команды озвучки должен состоять из двух частей: интерфейс пользователя и интерфейс администратора.
* необходимо реализовать авторизацию администратора;
* необходимо реализовать быстрый плеер для комфортного просмотра контента;
* необходимо реализовать админ панель для администрирования контента;
* разработанный интерфейс сервиса должен проходить тест на кроссбраузерность и адаптивность на разных устройствах;
* структура пользовательской части сайта: главная страница, список эпизодов, список новостей, страница контактов;

**2 АНАЛИЗ ПРОБЛЕМНОЙ СИТУАЦИИ И ОБЗОР АНАЛОГОВ**

## 2.1 Описание ситуации

На данный момент популярные площадки (YouTube, ВКонтакте) очень жестко отслеживают авторский контент. И так как размещение озвучки не считается “добросовестным использованием”, заказчику необходима своя площадка для размещения актуальных работ по озвучке.

## 2.2 Обзор аналогов

В процессе анализа проблемной ситуации был проведен обзор существующих аналогов, среди которых выделены сервисы, представленные ниже.

YouTube (https://youtube.com/):

1. Блокировка контента на территории отдельно взятой страны;
2. Полная блокировка контента.

ВКонтакте (https://vk.com/):

1. Массовые чистки контента по авторским правам;
2. Необходимость использования обходных путей для жителей Украины.

**3 ПЛАН РЕАЛИЗАЦИИ ПРОЕКТА**

Для определения времени осуществления мероприятий, направленных на достижение целей проекта, и для установления взаимосвязей между ними по временному параметру с учётом наиболее рисковых событий, составляется календарный план проекта. Календарное планирование заключается в создании и последующем уточнении расписания, которое учитывает состав работ, риски, ограничения. Поскольку календарный план в виде перечня исключительно плановых параметров работ без сравнения с фактическими сроками выполнения утрачивает свой смысл, нередко, вместо календарного плана, применяют название календарного графика.

В таблице 3.1 показан план реализации проекта, где описаны этапы проекта, сроки выполнения, результаты, участники проекта и задействованные материально-технические ресурсы.

Таблица 3.1 – План реализации проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Этапы проекта / конкретные мероприятия, детализирующие этапы | Cрок  выполнения | Результат | Задействованные материально-технические  ресурсы |
| Этап 1 | Проектирование БД | 10.02.21 – 25.02.21 | БД MySQL | OpenServer, MySQL |
| Этап 2 | Разработка дизайна | 25.02.21 – 03.03.21 | Макет в Figma | Figma |
| Этап 3 | Верстка публичной части сайта | 03.03.21 – 23.03.21 | Html, CSS, JS-файлы | Время, VSCode |
| Этап 4 | Разработка логики сайта | 23.03.21 – 25.03.21 | Функционал для сайта | VSCode |
| Этап 5 | Разработка административной панели | 26.03.21-04.04.21 | Функционал администрирования | VSCode |
| Этап 6 | Реализация хранения контента | 04.04.21-05.04.21 | Контент хранится в облачном хранилице | Yandex.Cloud |
| Этап 7 | Заливка сайта на хостинг | 06.04.21-07.04.21 | Функционирующий веб-сайт | SpaceWeb |

**4 ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ**

В ходе работы на стороне сервера было решено использовать язык PHP с фреймворком Laravel, в качестве основного языка был выбран язык Javascript и база данных MySQL. Клиентская часть была разработана на языке Javascript.

**PHP** – [скриптовый язык](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством [хостинг-провайдеров](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3) и является одним из лидеров среди языков, применяющихся для создания [динамических веб-сайтов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D0%B9%D1%82).

**Laravel** – веб-[фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA" \o "Фреймворк) с [открытым кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), предназначенный для разработки с использованием архитектурной модели [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Model View Controller* — модель-представление-контроллер). Laravel выпущен под [лицензией MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT).

**MySQL** – это популярная свободная объектно-реляционная система управления базами данных. MySQL базируется на языке SQL и поддерживает многочисленные возможности.

**OpenServer Panel** − это портативная программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.  OpenServer Panel широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

**5 ОПИСАНИЕ РЕЗУЛЬТАТА**

**5.1 Реализация интерфейса**

При запуске web-приложения открывается главная страница, на которой находятся три основных блока: header, блок с последними работами и блок с последними новостями (Рисунок 1). Header содержит лого и меню, позволяющее перейти в необходимый раздел сайта. При переходе на страницу “Список серий” пользователь попадает на страницу с полным перечнем серий с возможностью просмотра (Рисунок 2).

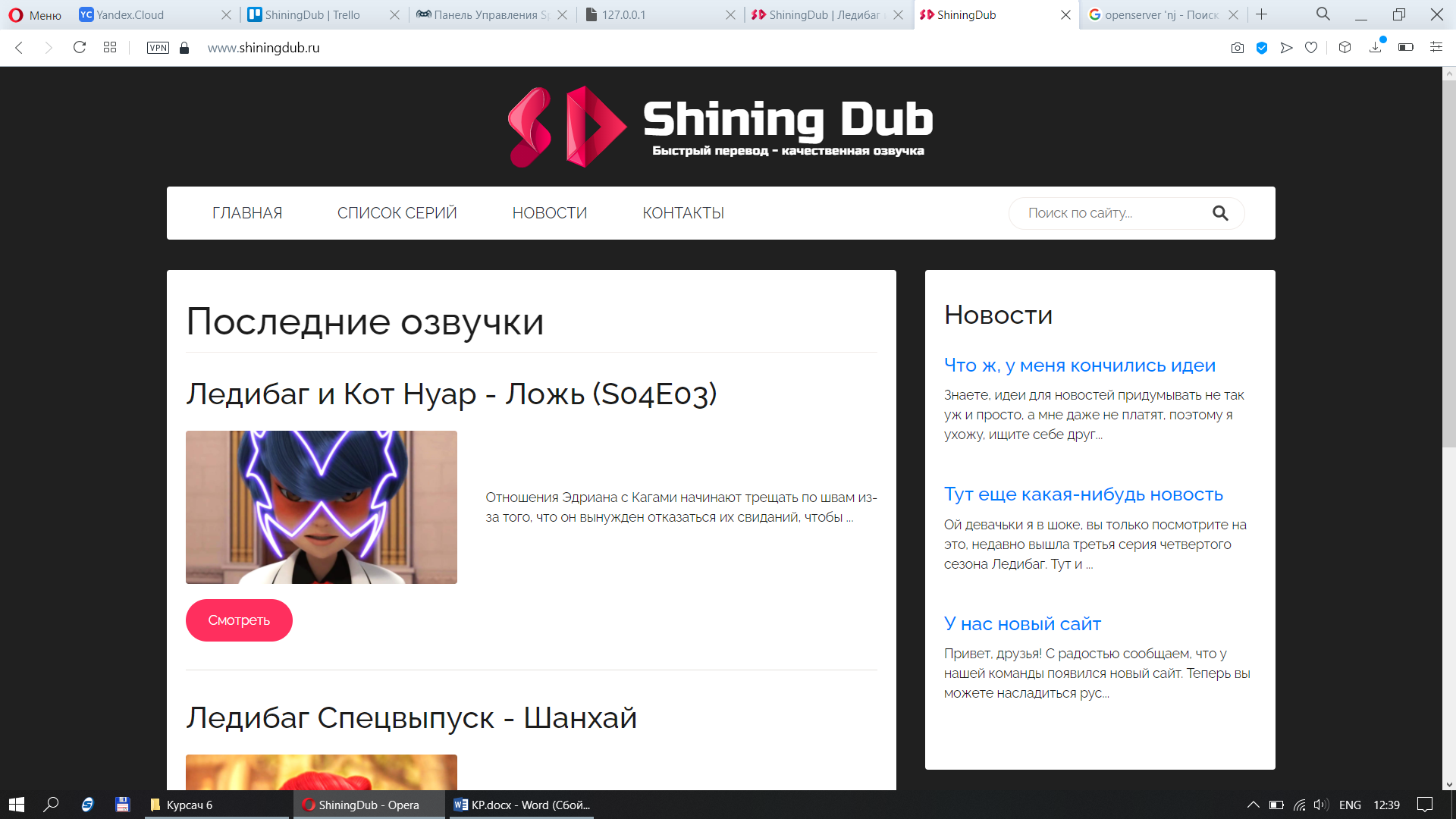


Рисунок 1 – Главный экран

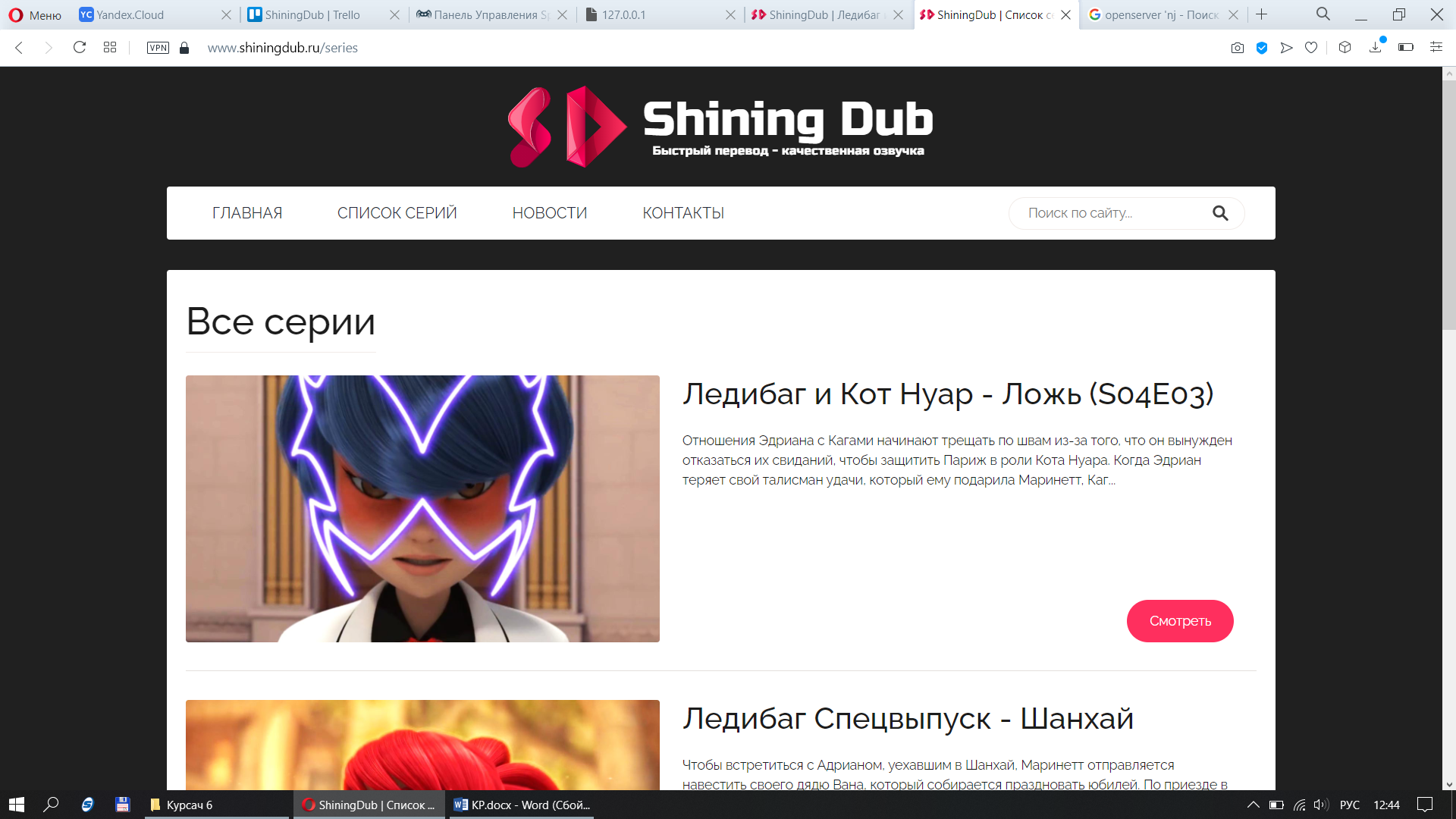


Рисунок 2 – Страница с полным списком серий

При нажатии на кнопку “Смотреть” пользователь попадает на страницу просмотра видео, с быстрым и удобным плеером и возможностью оставить комментарий (Рисунок 3).



Рисунок 3 – Страница просмотра видео

При нажатии пункта “Новости” главного меню, пользователь попадает на страницу с перечнем новостей, сортированных по дате добавления (Рисунок 4).

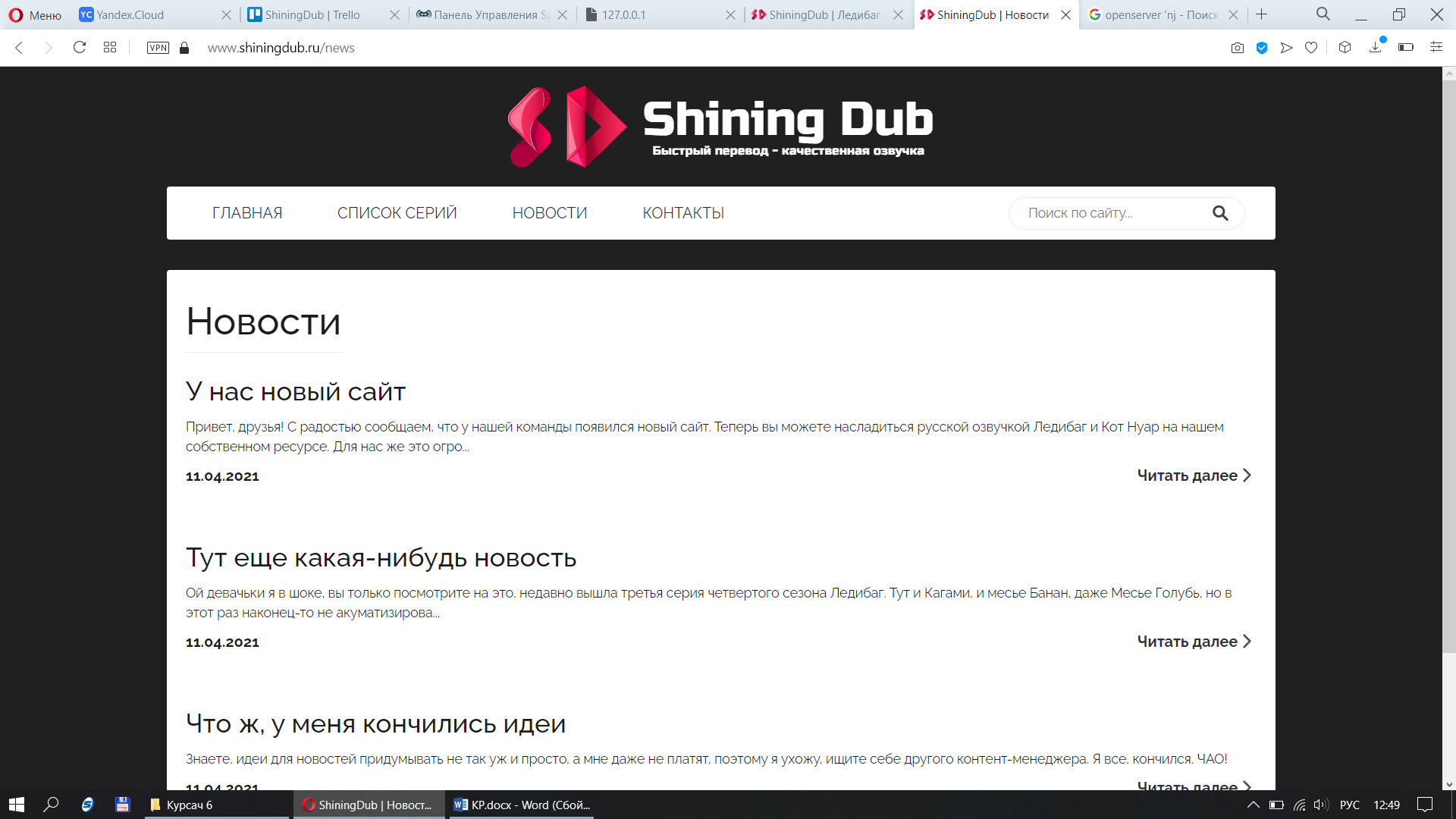


Рисунок 4 – Страница с новостями

При переходе по адресу <https://shiningdub.ru/login>, администратор попадет на страницу авторизации в админ панель (Рисунок 5).

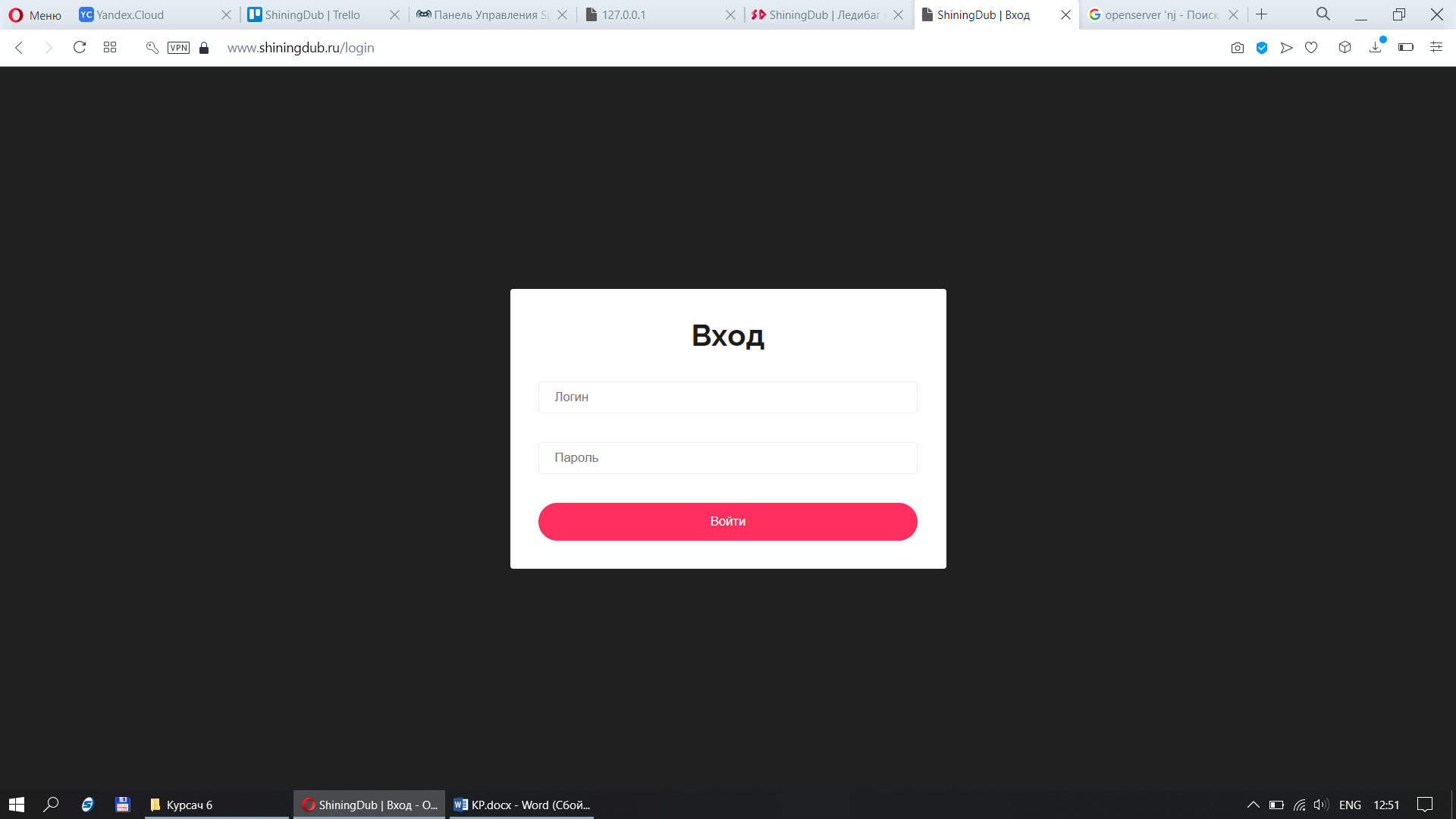


Рисунок 5 – Страница авторизации

После авторизации администратору доступен функционал администрирования контента. Например: добавление, удаление и редактирование видео (Рисунки 6-7), те же функции для новостей, удаление комментариев (Рисунок 8).

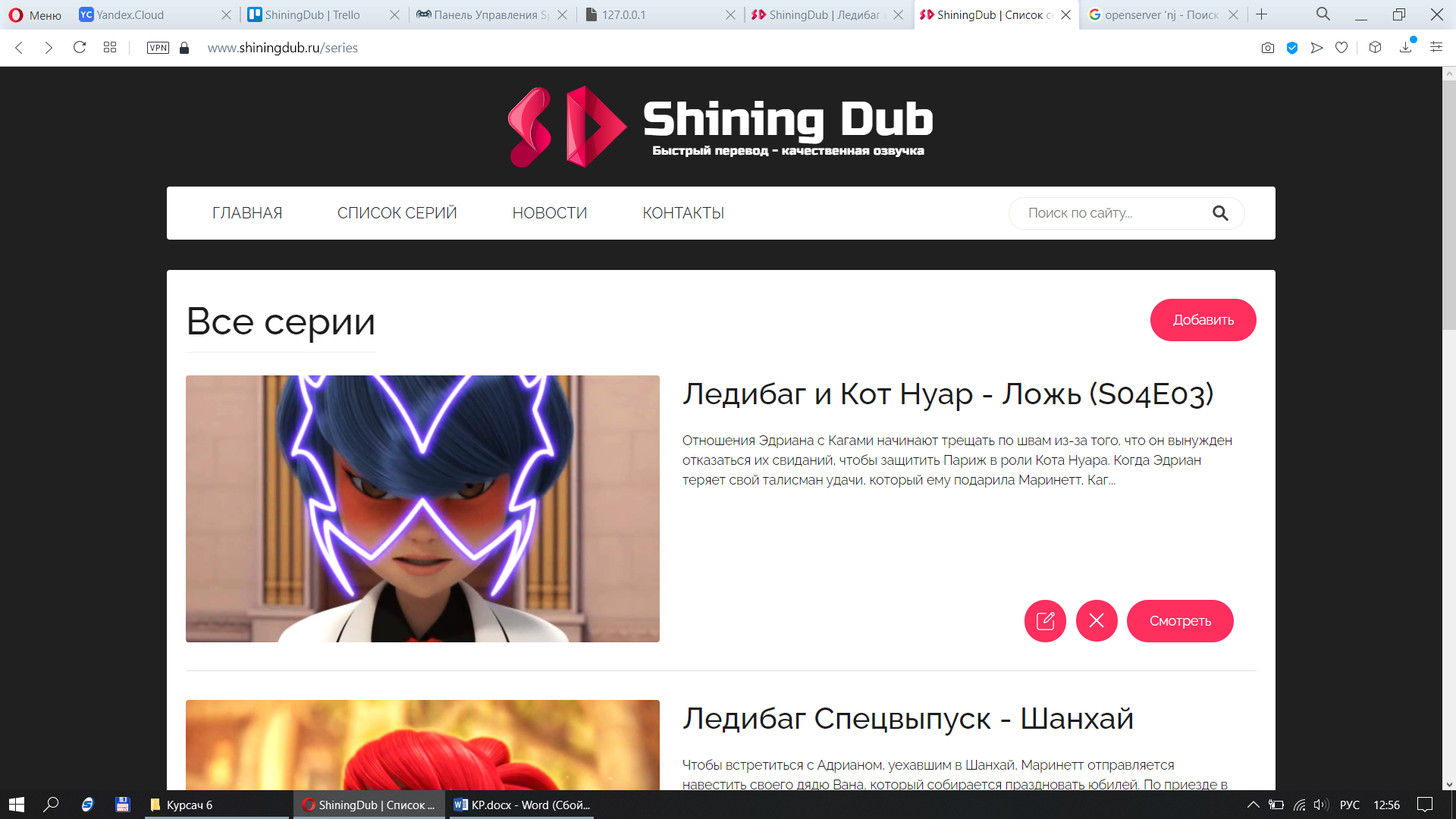


Рисунок 6 – Оверлей администратора

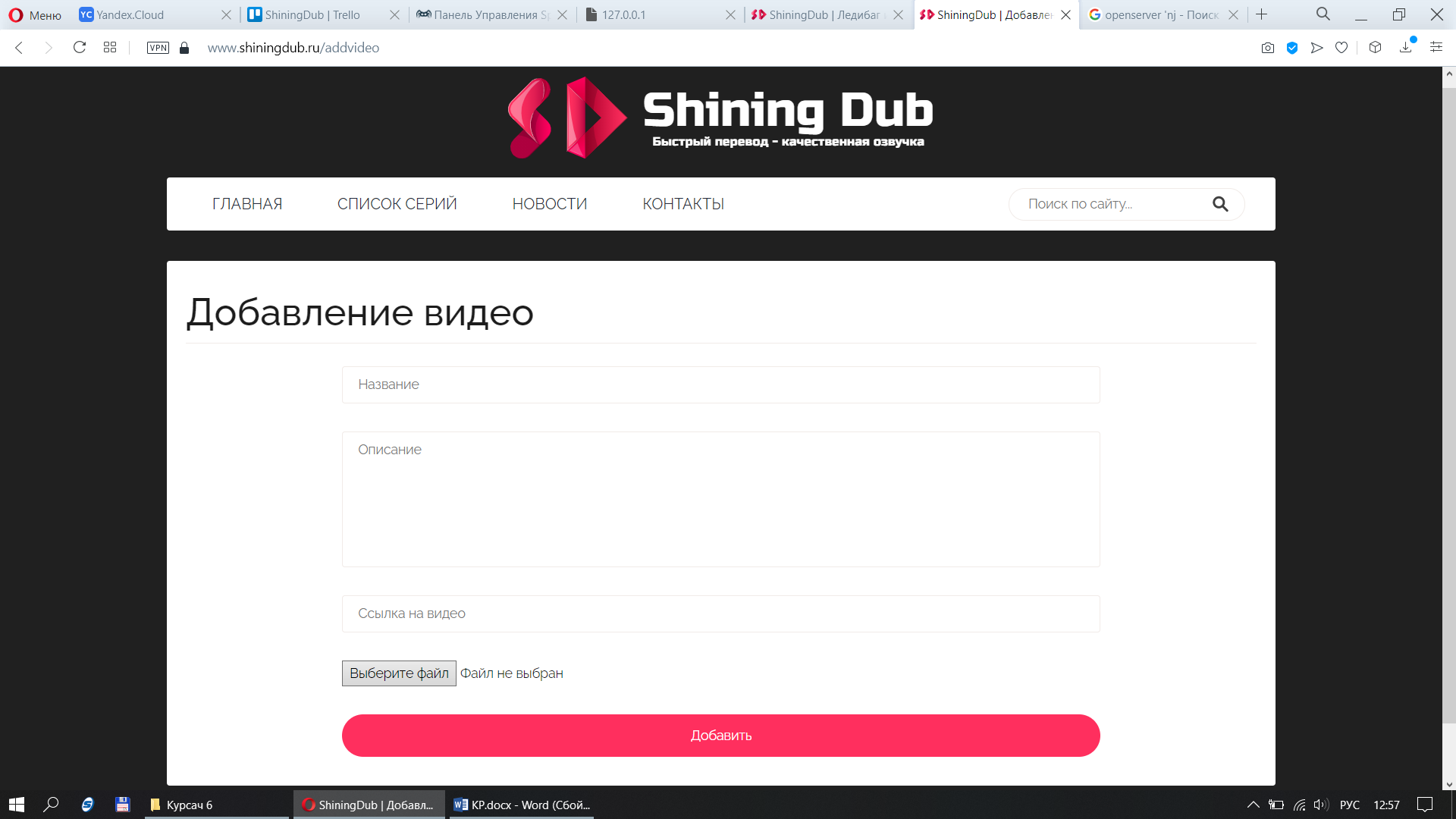


Рисунок 7 – Страница добавления видео

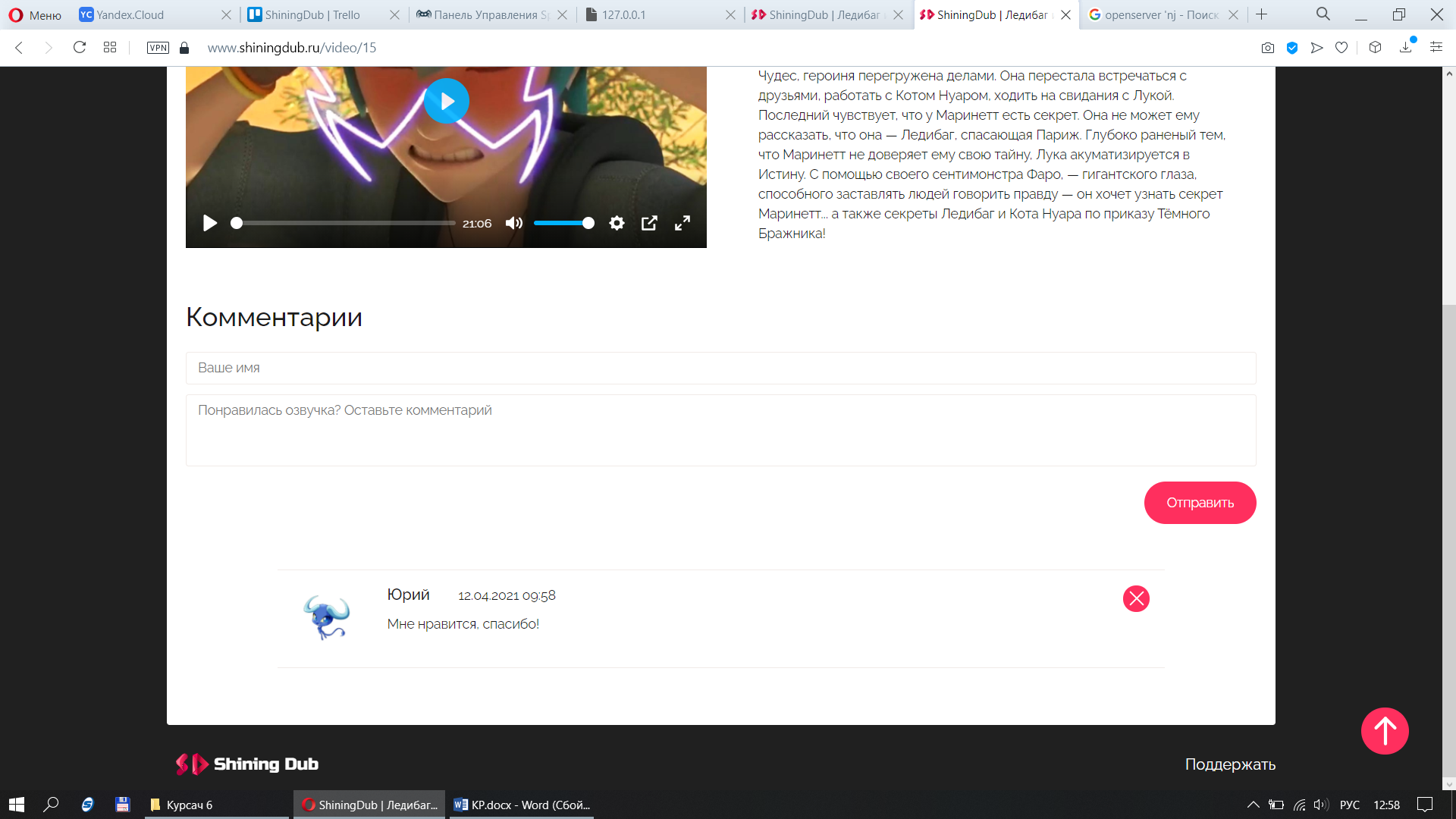


Рисунок 8 – Функция удаления комментария

**5.2 Реализация серверной части**

Для начала необходимо создать базу данных. Для данного сайта БД будет иметь такие таблицы как: Новости, Видео, Комментарии и Пользователи (Рисунок 9).

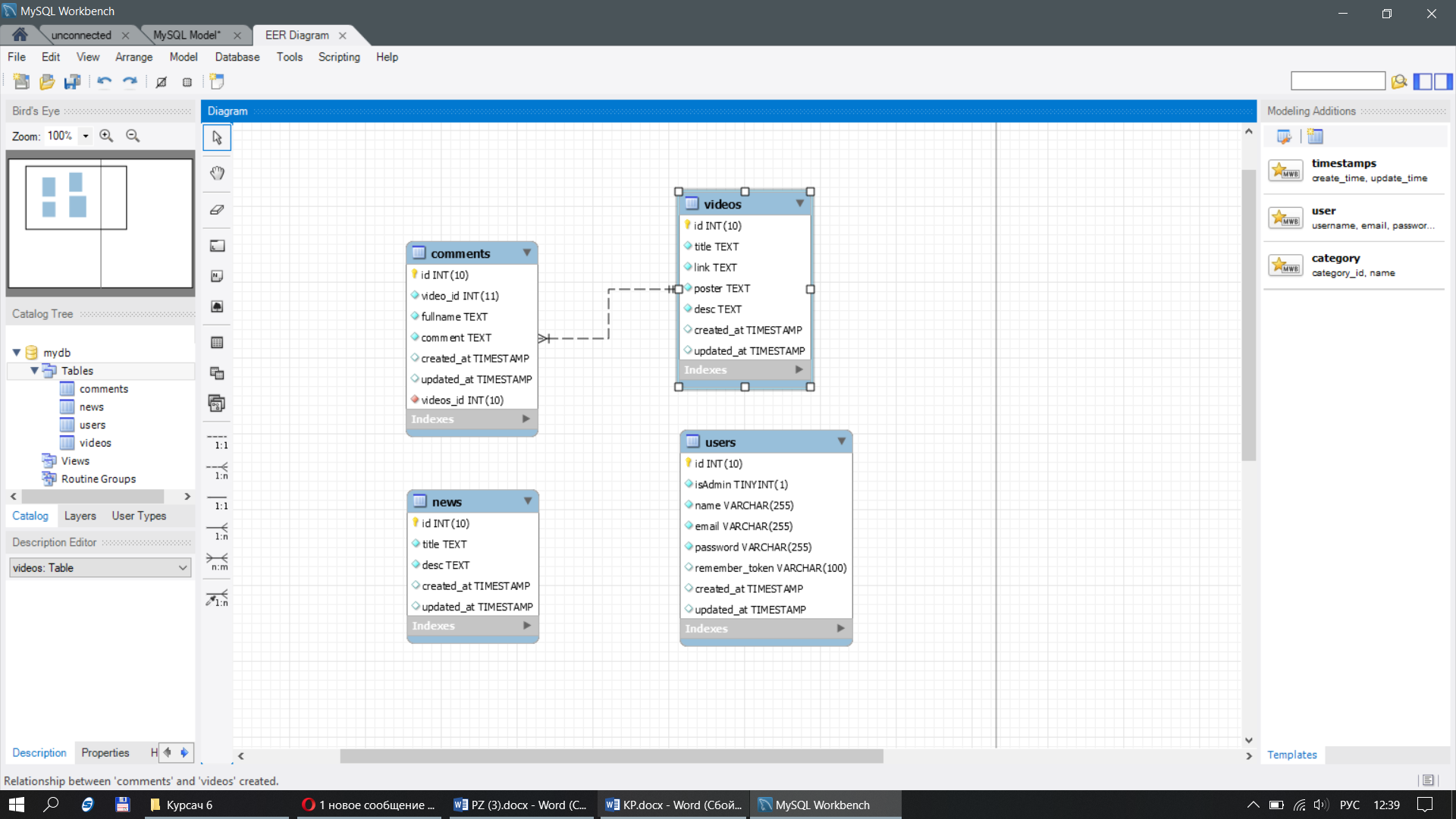


Рисунок 9 – Модель базы данных

Для того, чтобы организовать связь с базой данных в Laravel предусмотрены некоторые команды, выполняемые в консоли.

Для начала создадим миграции посредством команды:

**php** artisan make:migration Video -m

Далее нам необходимо заполнить поля в модели:

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Video extends Model

{

protected $fillable = [

'title',

'poster',

'link',

'desc'

];

public function comments()

{

return $this->hasMany(Comment::class);

}

}

И посредством команды **php** artisan migrate мигрируем таблицу в базу данных.

Чтобы настроить действия с базой данных были созданы контроллеры, типовой пример такого файла показан ниже:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Video;

class VideoController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

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

return view('serieslist', compact('data'));

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

$data = $request->input();

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

if ($request->hasFile('poster')) {

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

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

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

}

$item->save();

return redirect()->route("series");

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$data = Video::with(['comments' => function ($q) {

$q->orderBy('created\_at', 'desc');

}])->get()->find($id);

return view('video', compact('data'));

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

$data = Video::find($id);

return view('editvideo', compact('data'));

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

$data = $request->input();

$item = Video::findOrFail($id);

if ($request->hasFile('poster')) {

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

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

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

}

$item->update($data);

// $item = Video::findOrFail($id);

// $item->update($data);

return redirect('series');

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

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

return redirect('/series');

}

public function search(Request $request){

$search = $request->input('search');

$data = Video::query()

->where('title', 'LIKE', "%{$search}%")

->orWhere('desc', 'LIKE', "%{$search}%")

->get();

return view('search', compact(['data', 'search']));

}

}

**6 ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО САЙТА**

Для начала протестируем корректность работы видеоплеера (Рисунок 9).

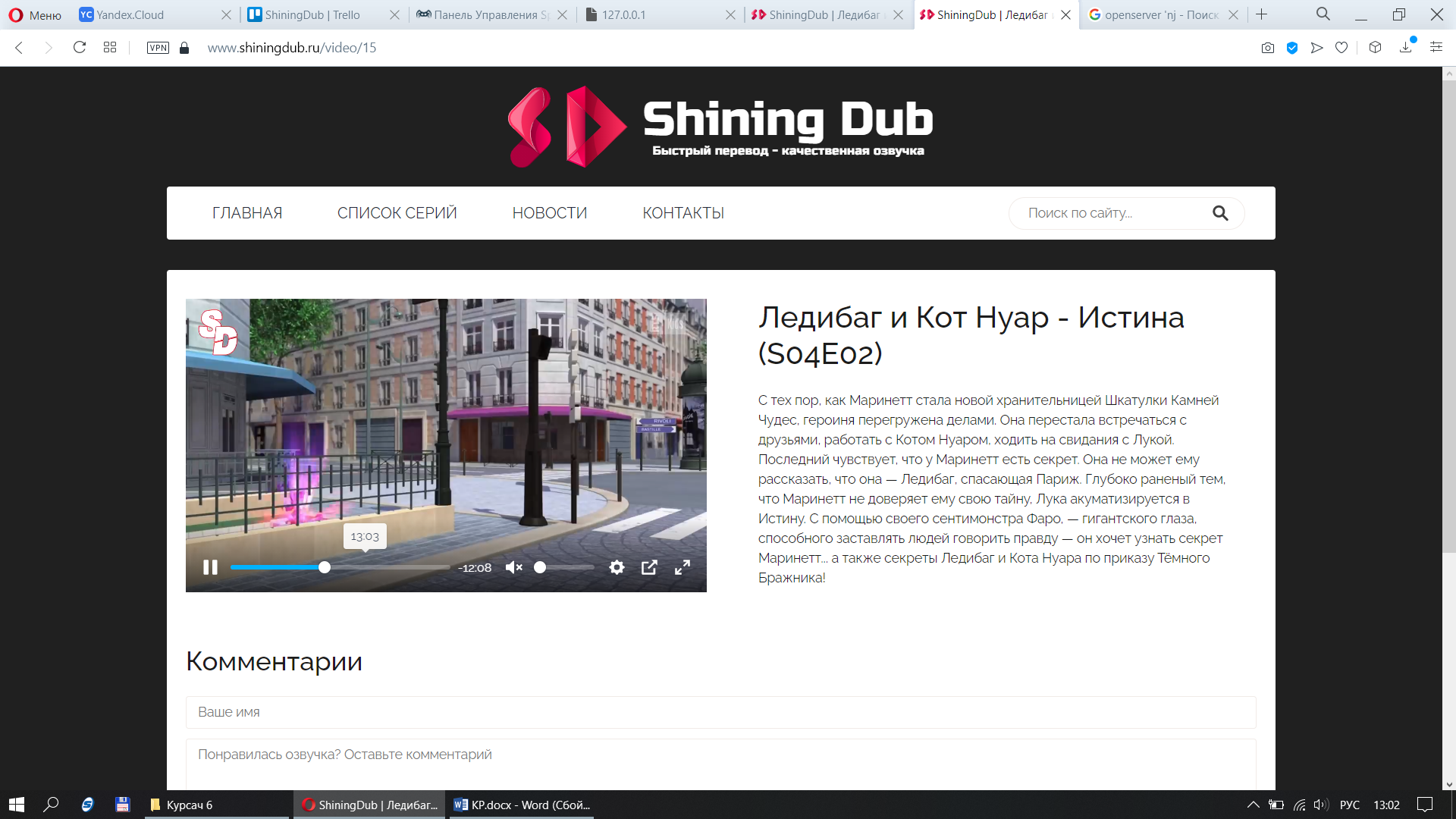


Рисунок 9 – Работа видеоплеера

Также проверим функцию добавления комментария. При его отправке аватар комментатора должен рандомно выбираться из массива картинок (Рисунки 10-11). Картинки выбираются случайно при каждом переходе на страницу.

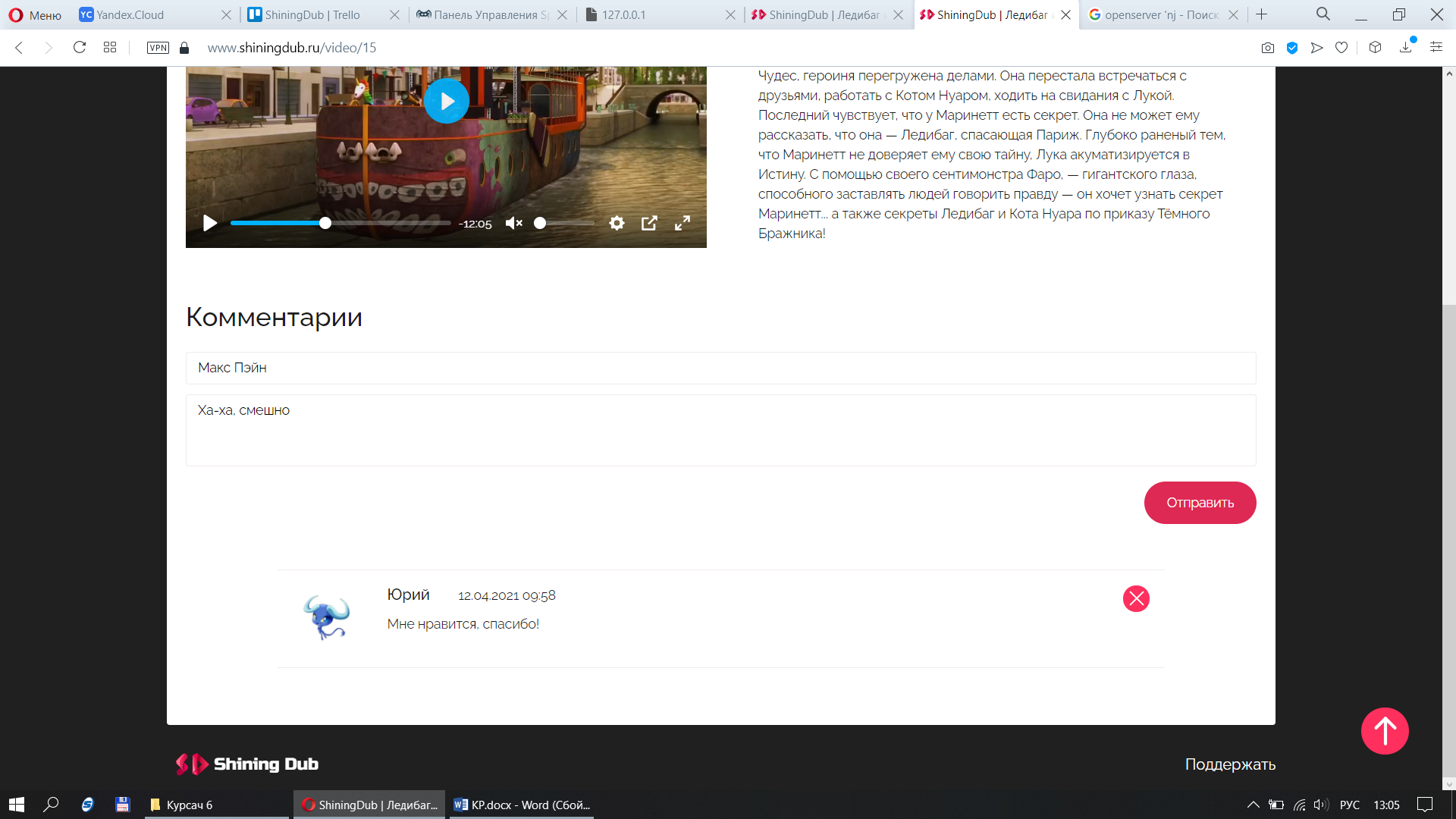


Рисунок 10 – Поле ввода комментария

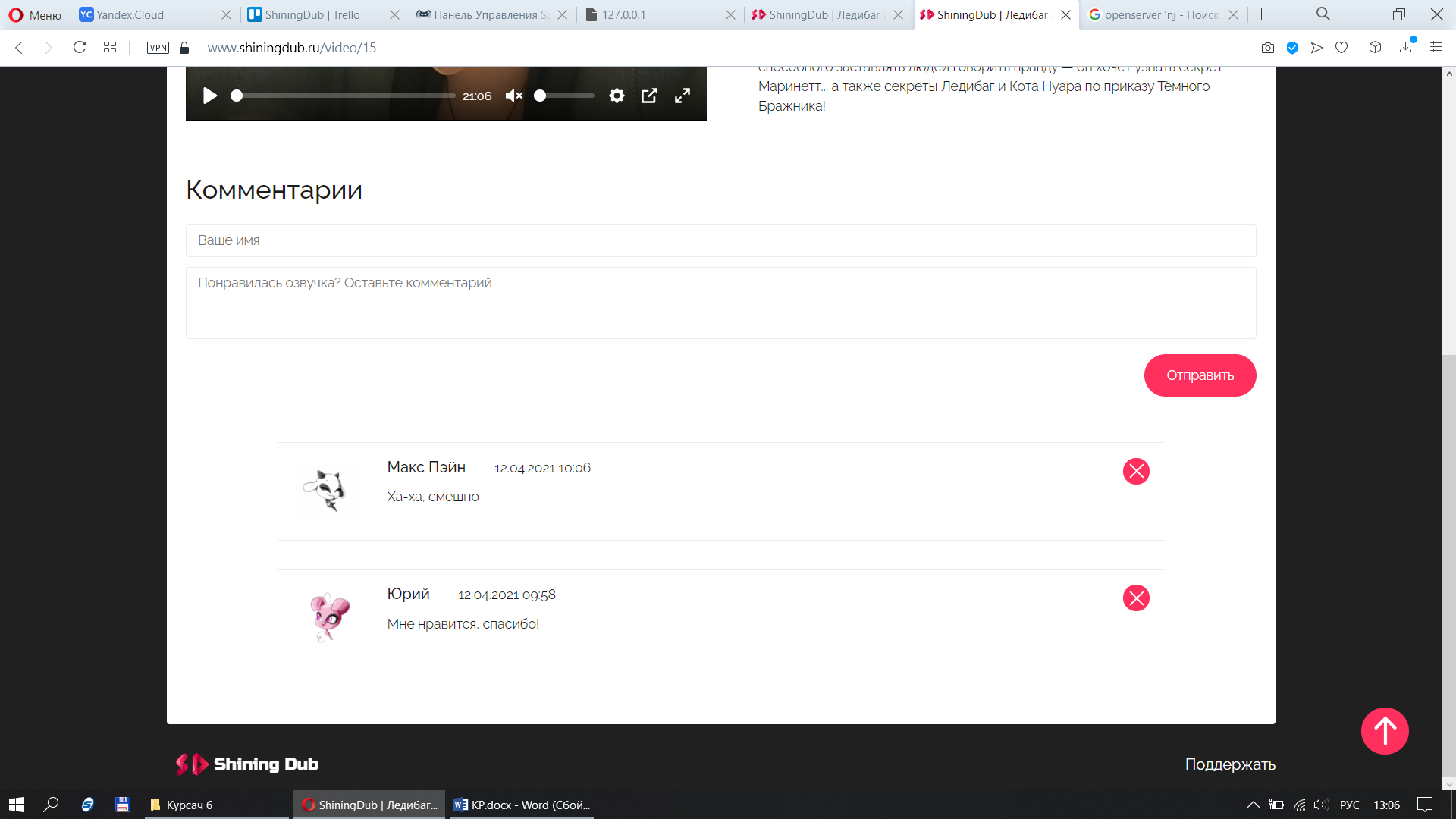


Рисунок 11 – Успешно добавленный комментарий

При переходе на несуществующий route, пользователь увидит ошибку 404 и предложение вернуться на главную сайта (Рисунок 12).

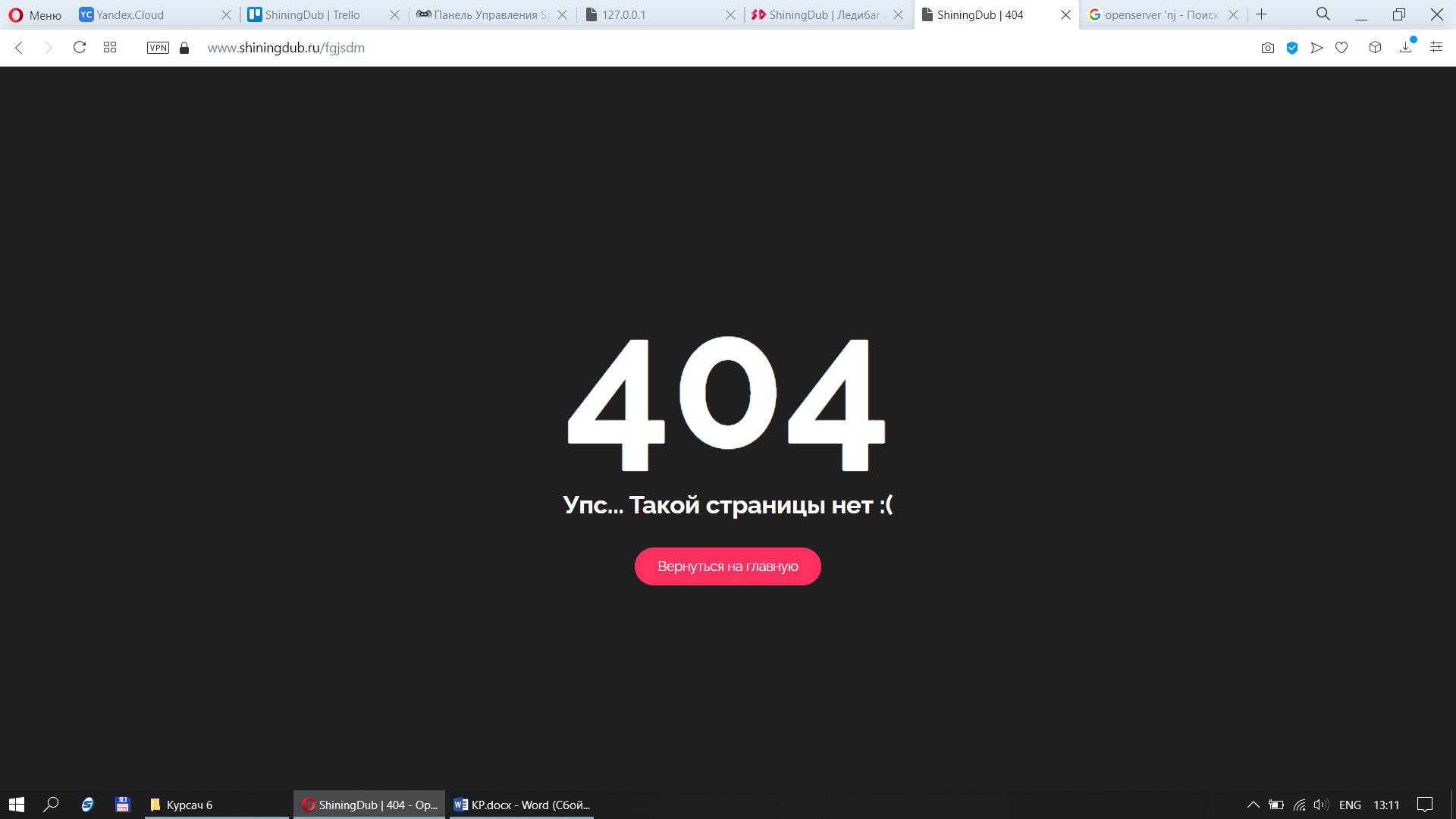


Рисунок 12 – Страница 404

Для удобства пользователя была реализована адаптивная верстка. Мобильная версия сайта полностью подходит под все разрешения экранов (Рисунки 13-16).

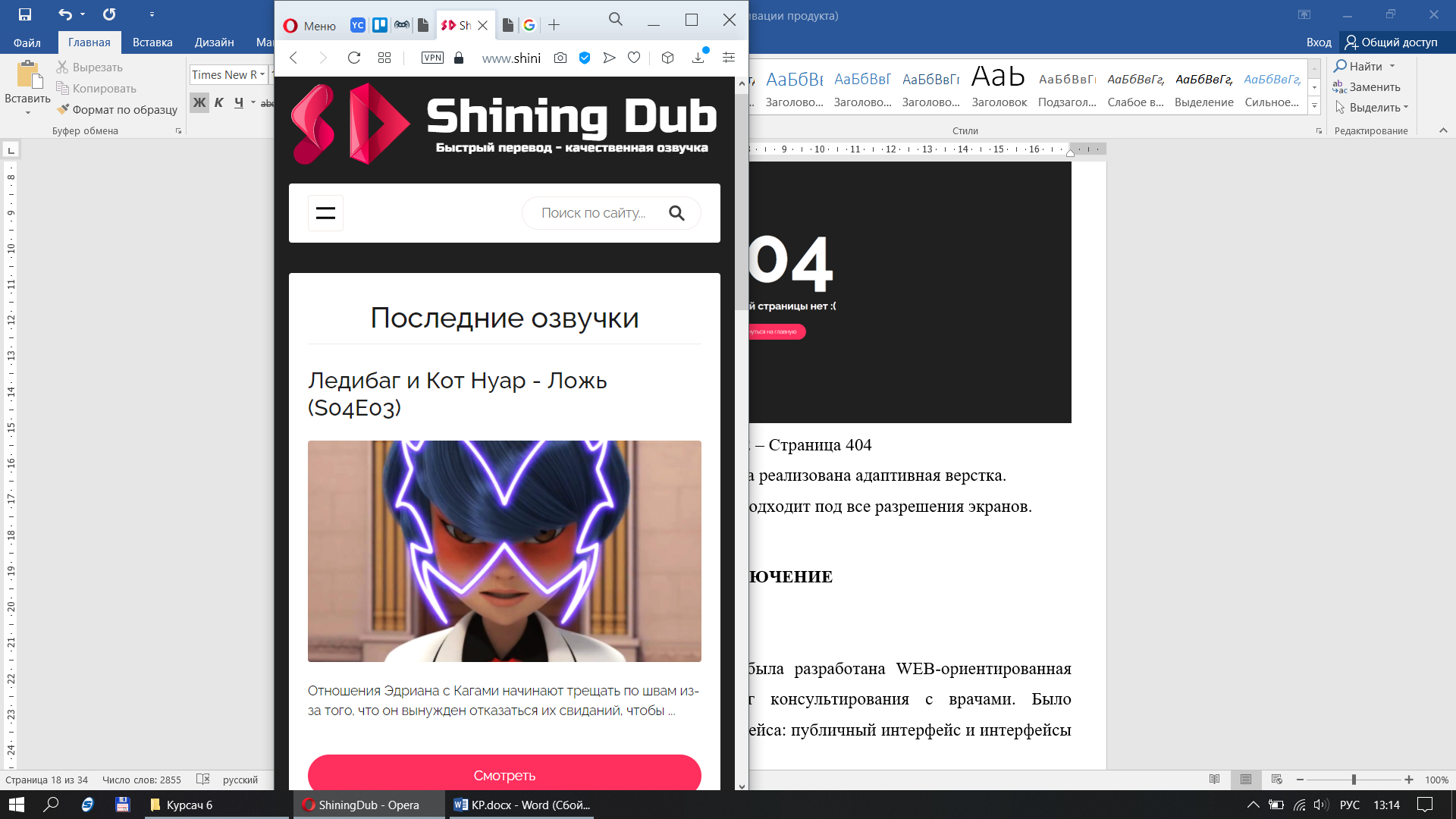


Рисунок 13 – Главная страница мобильной версии

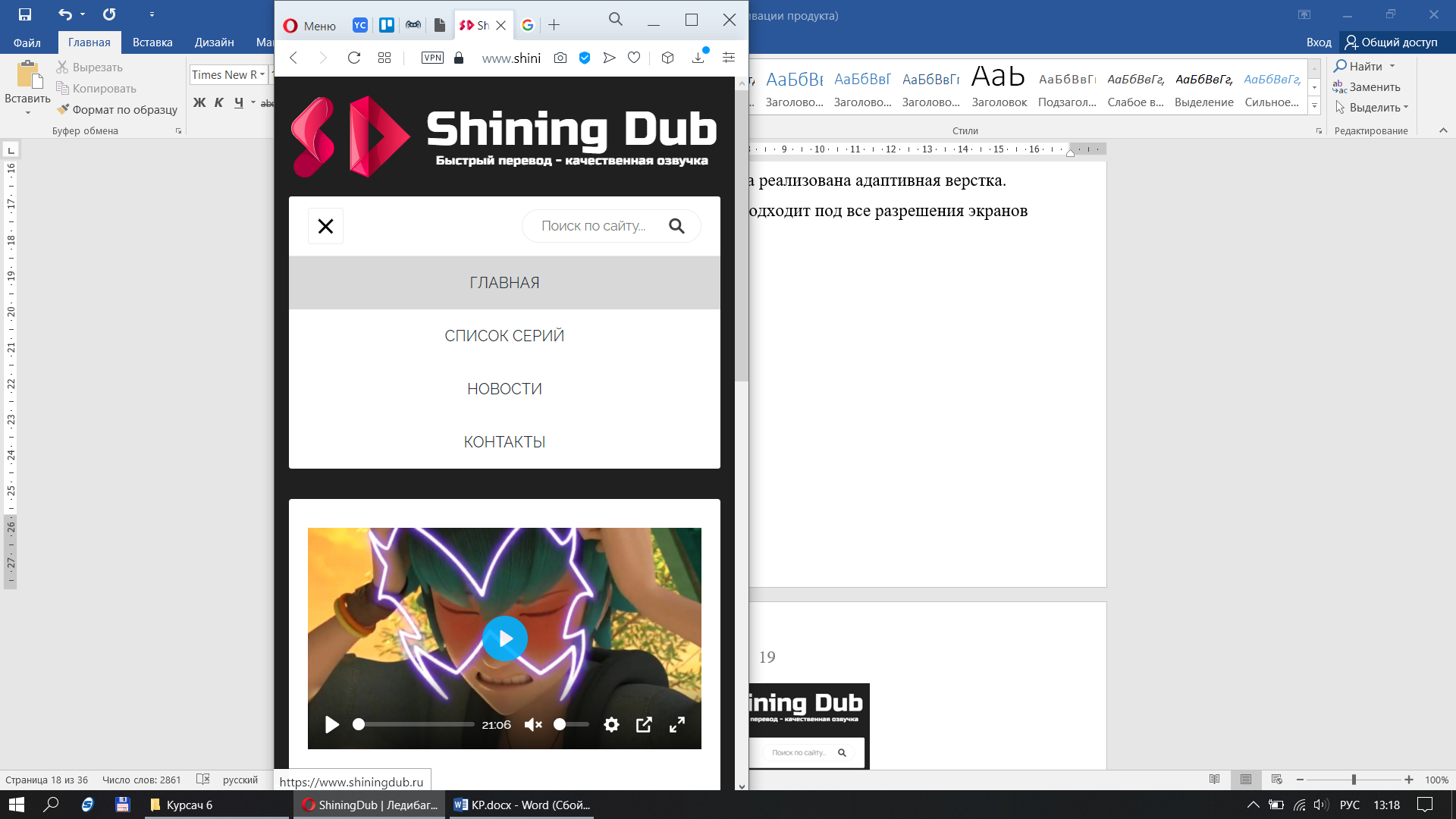


Рисунок 14 – Меню сайта в мобильной версии

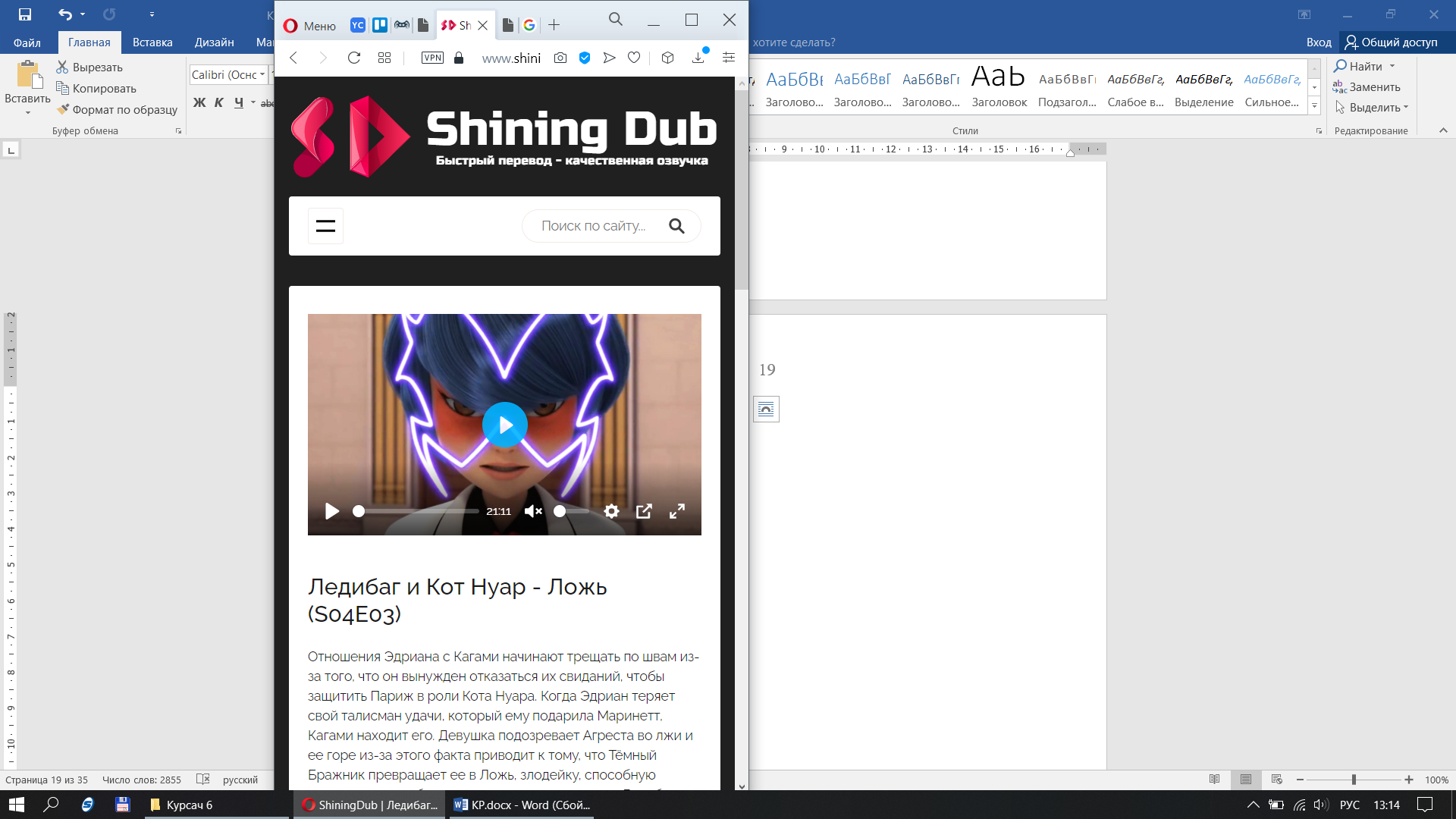


Рисунок 15 – Страница просмотра видео в мобильной версии

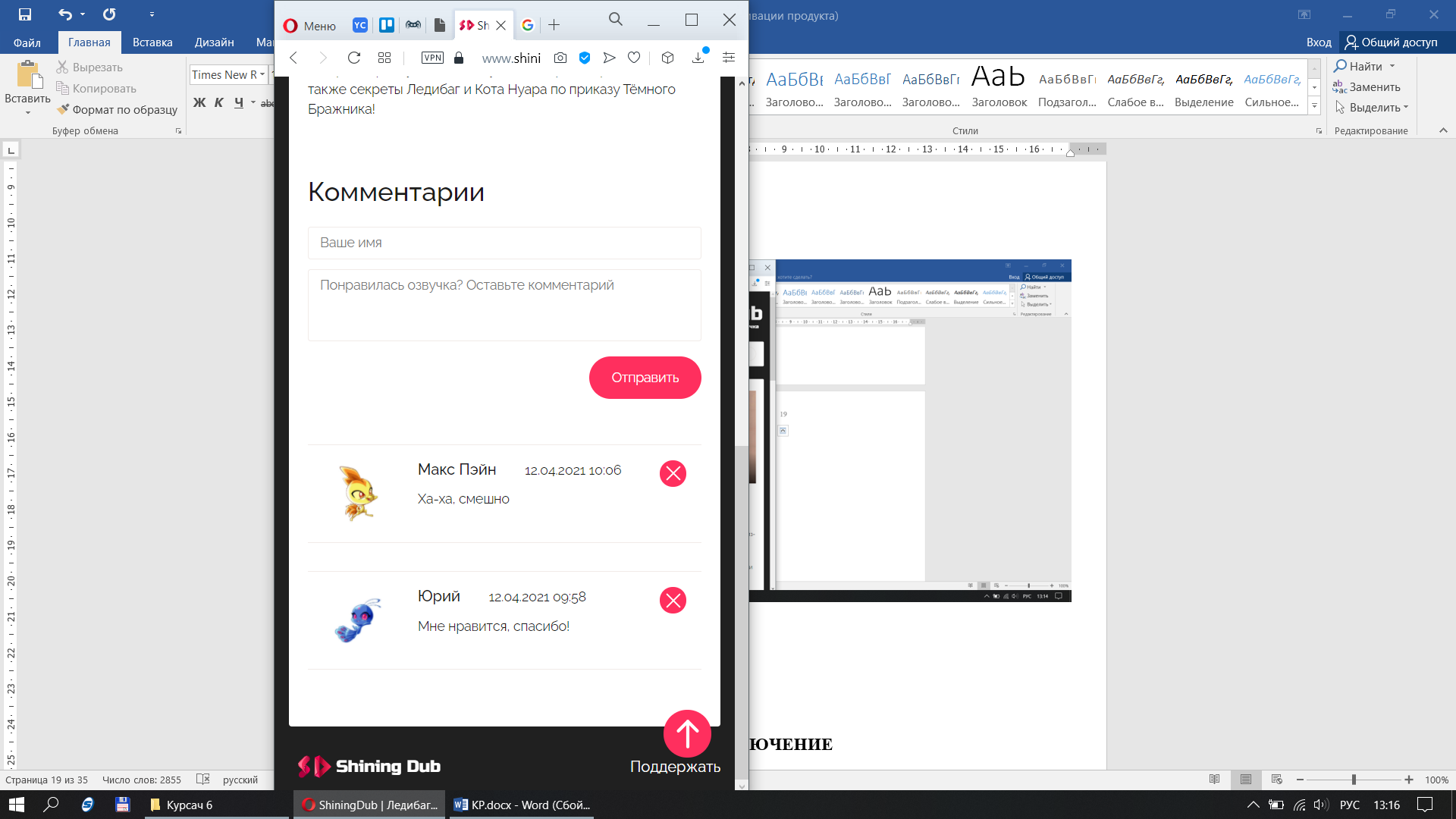


Рисунок 16 – Комментарии в мобильной версии

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте был разработан Web-сайт команды озвучки. Было разработано два независимых интерфейса: интерфейс пользователя и интерфейс администратора.

Для интерфейса администратора был организован авторизированный доступ. Интерфейс пользователя позволяет просматривать видео, новости, оставлять комментарии. Интерфейс администратора позволяет полностью администрировать контент на сайте.

Для публичного интерфейса организован свободный доступ. Публичный интерфейс содержит как статические страницы, так и страницы просмотра видео и новостей.

Разработанный Web-сайт был протестирован на наличие корректной валидации формы, правильной работы функций администратора, функционирования комментариев.

**СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ**

1. Laravel [Электронный ресурс]. URL: https://laravel.com/docs/7.x (дата обращения: 10.04.2021).

2. PHP [Электронный ресурс]. URL: https://www.php.net/docs.php (дата обращения: 10.04.2021).

**ПРИЛОЖЕНИЕ А**

ДИАГРАММА ГАНТА

| Неделя | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Дата | | Фев  10 | фев | фев | фев | фев | фев | фев | март  10 | март | март | март | март | март | март | март | Апр  1 |
| Этап 1 | Проектирование БД |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Этап 2 | Разработка дизайна |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Этап 3 | Верстка публичной части сайта |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Этап 4 | Разработка логики сайта |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Этап 5 | Разработка административной панели |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Этап 6 | Реализация хранения контента |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Этап 7 | Залив проекта на хостинг |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Диаграмма Ганта – линейная диаграмма, демонстрирующая очередность и длительность отдельных действий по проекту.

Обозначения:

|  |  |
| --- | --- |
|  | - продолжительность выполнения работ |

**ПРИЛОЖЕНИЕ Б**

КОД ПРИЛОЖЕНИЯ

Файл web.php:

<?php

Route::get('/series', 'VideoController@index')->name('series');

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

Route::get('/news', 'NewsController@index')->name('news');

Route::get('/video/{id}', 'VideoController@show')->name('video');

Route::get('/news/{id}', 'NewsController@show')->name('newsdetails');

Route::post('/comment', 'CommentController@store')->name('comment');

Route::post('/add-video', 'VideoController@store')->name('add-video')->middleware('auth');

Route::post('/update-video/{id}', 'VideoController@update')->name('update-video')->middleware('auth');

Route::get('/edit-video/{id}', 'VideoController@edit')->name('edit-video')->middleware('auth');

Route::delete('/delete-video/{id}', 'VideoController@destroy')->name('delete-video')->middleware('auth');

Route::delete('/delete-comment/{id}', 'CommentController@destroy')->name('delete-comment')->middleware('auth');

Route::post('/add-news', 'NewsController@store')->name('add-news')->middleware('auth');

Route::post('/update-news/{id}', 'NewsController@update')->name('update-news')->middleware('auth');

Route::get('/edit-news/{id}', 'NewsController@edit')->name('edit-news')->middleware('auth');

Route::delete('/delete-news/{id}', 'NewsController@destroy')->name('delete-news')->middleware('auth');

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

return view("addvideo");

}) ->name('addvideo')->middleware('auth');

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

return view("addnews");

}) ->name('addnews')->middleware('auth');

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

return view("contacts");

}) ->name('contacts');

Auth::Routes();

HomeController.php:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Video;

use App\Models\News;

use Illuminate\Support\Facades\Crypt;

class HomeController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

$data = Video::all()->sortBy('created\_at')->reverse()->take(3);

$news = News::all()->sortBy('created\_at')->reverse()->take(3);

return view('welcome', compact(['data', 'news']));

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

//

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

//

}

}

VideoCintroller.php:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Video;

class VideoController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

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

return view('serieslist', compact('data'));

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

$data = $request->input();

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

if ($request->hasFile('poster')) {

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

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

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

}

$item->save();

return redirect()->route("series");

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$data = Video::with(['comments' => function ($q) {

$q->orderBy('created\_at', 'desc');

}])->get()->find($id);

return view('video', compact('data'));

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

$data = Video::find($id);

return view('editvideo', compact('data'));

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

$data = $request->input();

$item = Video::findOrFail($id);

if ($request->hasFile('poster')) {

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

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

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

}

$item->update($data);

// $item = Video::findOrFail($id);

// $item->update($data);

return redirect('series');

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

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

return redirect('/series');

}

}

Миграции:

UserMigration:

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->increments('id');

$table->boolean('isAdmin')->default(0);

$table->string('name');

$table->string('email')->unique();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('users');

}

}

VideoMigration:

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateVideosTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->increments('id');

$table->text('title');

$table->text('link');

$table->text('poster');

$table->text('desc');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('videos');

}

}

Вывод информации:

Список видео:

@include('header')

<main class="row">

<section class="col-md-12">

<div class="content-block">

<div class="content-header">

<h1 class="content-title">Все серии</h1>

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

<a class="button" href="{{URL::route('addvideo')}}">Добавить</a>

@endif

</div>

@if ($data->isEmpty())

<h3>Упс... Тут пусто :(</h3>

@endif

@foreach ($data as $item)

<article class="video-card">

<div class="video-img">

<img src="{{ $item->poster ? $item->poster : 'img/not-found.png' }}">

</div>

<div class="video-desc">

<div>

<h2>{{ $item->title }}</h2>

<p>{{ strlen($item->desc) >= 400 ? mb\_strcut($item->desc, 0, 400)."..." : $item->desc }}</p>

</div>

<div class="video-link">

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

<a href="{{URL::route('edit-video', $item->id)}}" class="button admin-button"><img src="{{ asset('img/edit.svg') }}" alt=""></a>

<form action="{{URL::route('delete-video', $item->id)}}" method="POST">

{!! method\_field('delete') !!}

{!! csrf\_field() !!}

<button class="button admin-button"><img src="{{ asset('img/delete.svg') }}" alt=""></button>

</form>

@endif

<a class="button" href="/video/{{ $item->id }}">Смотреть</a>

</div>

</div>

</article>

@endforeach

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

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

{{ $data->links() }}

@endif

</div>

</div>

</section>

</main>

@include('footer')

Список новостей:

@include('header')

<main class="row">

<section class="col-md-12">

<div class="content-block">

<div class="content-header">

<h1 class="content-title">Новости</h1>

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

<a class="button" href="{{URL::route('addnews')}}">Добавить</a>

@endif

</div>

@if ($data->isEmpty())

<h3>Упс... Тут пусто :(</h3>

@endif

@foreach ($data as $item)

<article class="news-item">

<h3>{{ $item->title }}</h3>

<p>{{ strlen($item->desc) >= 350 ? mb\_strcut($item->desc, 0, 350)."..." : $item->desc }}</p>

<div class="news-item-bottom">

<h6>{{ $item->created\_at->format('d.m.Y') }}</h6>

<div class="admin-news-link">

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

<a href="{{URL::route('edit-news', $item->id)}}" class="button admin-button"><img src="{{ asset('img/edit.svg') }}" alt=""></a>

<form action="{{URL::route('delete-news', $item->id)}}" method="POST">

{!! method\_field('delete') !!}

{!! csrf\_field() !!}

<button class="button admin-button"><img src="{{ asset('img/delete.svg') }}" alt=""></button>

</form>

@endif

<a href="/news/{{ $item->id }}" class="news-bottom-link">

<h6>Читать далее</h6>

<img src="{{ asset('img/continue.svg') }}" alt="">

</a>

</div>

</div>

</article>

@endforeach

</div>

</section>

</main>

@include('footer')

Layouts

Header:

<header id="header">

<a class="logo" href="{{URL::route('main')}}">

<img src="{{ asset('img/logo.svg') }}" alt="ShiningDub">

</a>

<nav class="nav-bar">

<ul class="nav-list">

<li class="nav-item"><a href="{{URL::route('main')}}" class="link">Главная</a></li>

<li class="nav-item"><a href="{{URL::route('series')}}" class="link">Список серий</a></li>

<li class="nav-item"><a href="{{URL::route('news')}}" class="link">Новости</a></li>

<li class="nav-item"><a href="{{URL::route('contacts')}}" class="link">Контакты</a></li>

</ul>

<button id="menu" class="nav-menu btn-menu-toggle"></button>

<div class="search-box">

<input type="text" class="search" placeholder="Поиск по сайту...">

<a href="" class="search-btn">

<img src="{{ asset('img/search.svg') }}" alt="">

</a>

</div>

</nav>

<ul class="nav-list nav-list-mob">

<li class="nav-item nav-item-mob"><a href="{{URL::route('main')}}" class="link">Главная</a></li>

<li class="nav-item nav-item-mob"><a href="{{URL::route('series')}}" class="link">Список серий</a></li>

<li class="nav-item nav-item-mob"><a href="{{URL::route('news')}}" class="link">Новости</a></li>

<li class="nav-item nav-item-mob"><a href="{{URL::route('contacts')}}" class="link">Контакты</a></li>

</ul>

</header>

Footer:

<footer class="footer justify-content-between pb-3">

<a class="footer-logo" href="{{URL::route('main')}}">

<img src="{{ asset('img/logo-footer.svg') }}" alt="ShiningDub">

</a>

<a class="footer-link" href="https://vk.com/app6471849\_-189310045">

<h6>Поддержать</h6>

</a>

</footer>