**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 «Веб-технологии»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| на тему | **Разработка WEB-сайта для аренды недвижимости** | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Выполнил: студент | | | III | | | 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) | | | | | |  | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| **Черняев Никита Георгиевич** | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество студента) | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Руководитель | | | Дрозин А.Ю. | | | | | | | | | | | | | |
|  | | | (фамилия, инициалы, степень, звание, должность) | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Защита « | |  | | » |  | | | | 20 | | 20 | г. | | Оценка | |  |
|  | | | | | | | | | | | | | | | | |
| Руководитель | | | | | | |  | | | | | |  | |  | |
|  | | | | | | | (подпись) | | | | | |  | | (инициалы, фамилия) | |
|  | | | | | | | | | | | | | | | | |
| Ведущий преподаватель | | | | | | |  | | | | | |  | | Дрозин А.Ю. | |
|  | | | | | | | (подпись) | | | | | |  | | (инициалы, фамилия) | |
|  | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | 20 | | г. | | | | | | |

**АННОТАЦИЯ**

В документе представлено описание основных этапов выполнения курсового проекта по дисциплине «Веб-технологии». Курсовой проект посвящен разработке WEB-сайта по заданной вариантом теме. Приведено техническое задание, по которому реализован данный проект.

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ**

**1 Цели создания и целевая аудитория**

Цели создания сайта:

1. Представление информации о фирме в сети Интернет с целью расширения сферы её деятельности.

2. Круглосуточное привлечение новых клиентов с целью увеличения объёма аренды недвижимости.

3. Осуществить удаленную демонстрацию недвижимости.

4. Помощь клиенту в выборе недвижимости.

Целевой аудиторией сайта являются клиенты, заинтересованные в аренде недвижимости.

**2 Структура сайта**

В разрабатываемом WEB-сайте фирмы аренды недвижимости предполагается создание двух независимых интерфейсов пользователей: интерфейс пользователей-посетителей и интерфейс пользователя-администратора сайта.

Для интерфейса администратора сайта необходима организация авторизированного доступа. Интерфейс пользователя-администратора обеспечивает администратору сайта возможность добавлять, изменять и удалять информацию некоторых разделов сайта. Информация, добавляемая администратором, сохраняется в базе данных и отображается в соответствующих разделах пользовательского интерфейса WEB-сайта.

Интерфейс пользователей-посетителей имеет свободный доступ – и содержит структурированную информацию о компании и ее продукции. Пользовательский интерфейс содержит статические разделы (разделы, для изменения которых необходимо привлечение разработчика или стороннего WEB-программиста) и динамические разделы (содержимое динамических разделов формируется программно по информации добавленной администратором сайта и хранящейся в базе данных).

Ниже будет более подробно описана структура и содержимое пользовательского и администраторского интерфейсов разрабатываемого WEB-сайта фирмы аренды недвижимости.

**2.1 Пользовательский интерфейс**

Ниже представлено более подробное описание структуры и содержания WEB-страниц пользовательского интерфейса WEB-сайта фирмы аренды недвижимости.

2.1.1 Структура сайта

Каждая страница включает в себя меню сайта. Навигация по сайту – представляет собой строку ссылок, каждая из которых ведет на определенную страницу, а также кнопки входа и регистрации. Пункт меню “Информация о фирме” – всплывающее подменю со ссылками на страницы с историей компании, услуги, ей предоставляемые и контактами.

2.1.2 Главная страница сайта

Главная страница сайта имеет следующую структуру:

1. Рекламная карусель, на которой располагаются баннеры с акциями, скидками и т.д.

2. Общая информация о фирме.

2.1.3 Страница “Каталог”.

Каталог недвижимости – страница, разделенная вертикально на две части.

1. Слева список доступной недвижимости со ссылкой на страницу конкретной единицы недвижимости.

2. Справа – карта, на которой отображаются те же элементы.

2.1.4 Страница “О товаре”.

Данная страница отображает информацию о выбранном товаре из каталога (галерея, цена, описание) и контактный номер телефона.

2.1.5 Страница “Новости”.

Данная страница содержит новости фирмы, которые можно просматривать по клику.

2.1.6 Страница “Новости”.

Данная страница содержит новости фирмы (скидки, акции, новые поставки товара и т.п.), отсортированные по дате (по новизне). На странице должна быть реализована возможность переключения между страницами новостей. На каждой странице должно располагаться девять новостей.

2.1.7 Страница “Отзывы”.

На данной странице содержатся все отзывы клиентов. На странице должна быть реализована возможность переключения между страницами отзывов. На каждой странице должно располагаться девять отзывов.

2.1.8 Страница “Контакты”.

Страница “Контакты” содержит номер телефона фирмы, электронную почту, адрес, график работы.

2.1.9 Страница “Личный кабинет”.

На данной странице содержится личная информация клиента (ФИО, почта, телефон и кнопка редактирования профиля).

**2.2 Интерфейс администратора**

Ниже представлено описание страниц раздела администратора.

2.2.1 Страница администрирования каталога

Страница “Каталог” интерфейса администратора предназначена для добавления, удаления и редактирования единиц недвижимости в каталоге. Страница содержит список товаров, разделенных по категориям, в каждой из которых содержится таблица, отображающая все товары выбранной категории.

Таблица товаров в каждой категории содержит следующие колонки:

1. Описание.

2. Расположение.

3. Изображение.

2.2.2 Страница администрирования новостей

Страница предназначена для редактирования новостей. Она содержит форму для ввода новостей со следующими полями:

1. Заголовок.

2. Текст.

3. Краткий текст.

4. Дата.

5. Изображение.

Текст новости – визуальный текстовый редактор, содержит кнопки выделения текста жирным шрифтом, курсивом, цветом, выпадающие меню для выбора размера шрифта, создания маркированных и нумерованных списков, вставки изображений.

2.2.3 Страница администрирования заявок пользователей

Данная страница содержит список полученных заявок в виде таблицы, отсортированных по типу (оформление товара, заявка на замер, заявка на обратную связь, заявка на рассылку).

**3. Пожелания по сайту**

Все страницы разрабатываемого сайта должны быть выдержаны в одном стиле. Корпоративные цвета: белый, зеленый, серый.

**4. Технические требования к сайту**

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

Требования к установленному на веб-сервере ПО:

1. Frontend (Node.js).

2. Backend (Laravel + MySQL)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 12

1. ПОСТАНОВКА ЗАДАЧИ 13

2. АНАЛИЗ И ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 14

3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ WEB-САЙТА 15

4. ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 21

5. РАЗРАБОТКА РАЗДЕЛА АДМИНИСТРАТОРА 23

5.1. Проектирование интерфейса раздела администратора 23

5.2. Разработка программных модулей раздела администратора 29

6. РАЗРАБОТКА РАЗДЕЛА ПОЛЬЗОВАТЕЛЯ 35

6.1. Проектирование интерфейса раздела пользователя 35

6.2. Разработка программных модулей раздела пользователя 47

7. ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО САЙТА 53

7.1. Анализ кроссбраузерности сайта 53

7.2. Профилирование разработанного сайта 55

7.3. Тестовые примеры работы 57

ЗАКЛЮЧЕНИЕ 69

СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ 70

ПРИЛОЖЕНИЕ А. КОД ПРИЛОЖЕНИЯ 71

**ВВЕДЕНИЕ**

Информационные технологии сегодня настолько плотно вошли в нашу жизнь, что стали двигателем развития не только отдельных компаний, но и всего общества в целом. Ввиду возрастания потребностей человечества в обработке все большего объема данных, средства получения информации совершенствовались от самых ранних механических изобретений до современных компьютеров. C появлением глобальной сети каждый человек получил интерактивный инструмент, позволяющий сообщить миру об услугах и товарах компании, привлечь единомышленников и покупателей.

Целями данного курсового проекта являются закрепление основных положений дисциплины «Web-технологии», а также приобретение и закрепление практических навыков использования современных средств и технологий в области проектирования и разработки WEB-сайта на примере WEB-сайта организации. Для реализации данной работы, были проведены следующие этапы:

* изучение предметной области, разработка и утверждение технического задания на проектирование WEB-сайта для фирмы аренды недвижимости;
* разработка дизайна WEB-сайта фирмы аренды недвижимости. Создание HTML макета WEB-сайта;
* разработка базы данных WEB-сайта;
* разработка раздела администратора и пользовательской части;
* оценка и оптимизация характеристик разработанного WEB-сайта.

**1 ПОСТАНОВКА ЗАДАЧИ**

Определим основные функциональные требования для приложения:

* разрабатываемый WEB-сайт фирмы по аренде недвижимости должен содержать раздел администратора, обеспечивающий возможность управления информацией, отображаемой в пользовательской части сайта. Необходимо предусмотреть авторизованный доступ к разделу администратора;
* необходимо предусмотреть валидацию всех полей ввода на стороне клиента, как на заполненность (в тех случаях, где это необходимо), так и по формату (Email, дата, гиперссылка и т.п.);
* серверная часть сайта должна разрабатываться с использованием шаблона Модель-Представление-Контроллер (MVC);
* для управления версиями исходного кода проекта, а также для упрощения групповой разработки, необходимо использовать систему контроля версий Git и WEB-ориентированный репозиторий GitHub.com;
* необходимо выполнить анализ и оптимизацию разработанного сайта с точки зрения кроссбраузерности;
* структура пользовательской части сайта: Главная страница, Каталог, О компании, Новости, Отзывы, Контакты, Личный кабинет;
* структура раздела администратора:О фирме, Каталог, Новости, Отзывы, Заявки;
* требования к дизайну: Корпоративные цвета – зеленый, белый, серый. Меню сайта многоуровневое, сверху.

**2 АНАЛИЗ И ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметной областью данного курсового проекта является – аренда недвижимости.

Основной вид деятельности фирмы – предоставление посреднических услуг для аренды недвижимости. Главной целью такой организации, является извлечение прибыли посредством предоставления каталога для клиентов. Как правило, в структуру фирмы входит система обслуживания покупателей.

Создаваемый WEB-сайт позволит обеспечить качественную рекламу фирмы, а также упростить ряд вышеперечисленных процессов, обеспечив доступ к нужной для клиента информации в любое удобное для него время.

С помощью сайта у клиентов появится возможность ознакомится с полным каталогом мебели, проконсультироваться со специалистами, вызвать замерщика и составить список товаров для дальнейшего заказа.

**3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ WEB-САЙТА**

В результате анализа предметной области были выделены следующие таблицы для составления базы данных:

* Пользователи;
* Роли;
* Объекты недвижимости;
* Новости;
* Отзывы

Для создания данной базы данных в проекте была использована реляционная система управления базами данных MySQL, схема которой создается с помощью Laravel. Скрипты создания базы данных представлены далее:

Схема создания таблицы пользователей:

class CreateUsersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->engine = 'InnoDB';

$table->increments('id');

$table->string('name');

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

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

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

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

}

Схема создания таблицы ролей:

class CreateRolesTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->engine = 'InnoDB';

$table->increments('id');

//$table->string('user\_id');

$table->string('role');

//$table->foreign('user\_id')->references('id')->on('users')->onUpdate('cascade')->onDelete('cascade');

});

}

}

Схема таблицы для связи таблиц Пользователь и Роль:

class CreateRolesTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->engine = 'InnoDB';

$table->increments('id');

$table->string('user\_id');

$table->string('role');

$table->foreign('user\_id')->references('id')->on('users')->onUpdate('cascade')->onDelete('cascade');

});

}

Схема таблицы объектов недвижимости:

class CreateSubjects extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->increments('id');

$table->integer('user\_id')->unsigned()->default(1)->nullable();

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

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

$table->string('price')->default(0);

$table->boolean('checked')->default(false);

$table->json('images')->nullable();

$table->text('description')->nullable();

$table->timestamps();

});

Schema::table('subjects', function (Blueprint $table) {

$table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

});

}

}

Схема таблицы новостей:

class CreateNews extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->increments('id');

$table->string('header');

$table->string('image');

$table->string('news');

$table->string('author');

$table->string('news\_short');

$table->timestamps();

});

}

}

Схема таблицы комментариев:

lass CreateFeedBackTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('feed\_backs', function (Blueprint $table) {

$table->engine = 'InnoDB';

$table->increments('id');

$table->integer('user\_id')->unsigned()->default(1);

$table->json('data');

$table->timestamps();

});

Schema::table('feed\_backs',function (Blueprint $table){

$table->foreign('user\_id')->references('id')->on('users');

});

}

**4 ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ**

В ходе работы на стороне сервера было решено использовать PHP фреймворк Laravel, база данных MySQL. Клиентская часть была разработана на языке JS, с использованием Node.js.

**Node.js** – кроссплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык JavaScript. Среда исполнения предназначена для использования вне контекста браузера. Также в состав Node.js входит менеджер пакетов npm – это инструмент командной строки, который помогает в установке и удалении пакетов, управлении их версиями и зависимостями.

**PHP** – это скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

**MySQL** – самая популярная система управления реляционными базами данных с открытым исходным кодом (СУРБД) с моделью клиент-сервер.

**5 РАЗРАБОТКА РАЗДЕЛА АДМИНИСТРАТОРА**

**5.1 Проектирование интерфейса раздела администратора**

Интерфейс администратора позволяет администратору сайта управлять его содержимым, менять информацию о фирме, пополнять каталог новыми товарами, публиковать новости и обрабатывать заявки.

Для панели администратора организован авторизированный доступ, представленный на рисунке 1. Данная страница доступна по адресу [http://localhost:8000/admin](http://localhost:3000/admin) и представляет собой форму для входа в админ-панель, состоящую из поля логина администратора и его пароля. Если пользователь попытается обойти форму для входа, например, перейдя по ссылке [http://localhost:8000/admin/\*](http://localhost:3000/admin/*), то ему будет отказано в доступе и он будет отправлен на страницу для входа.

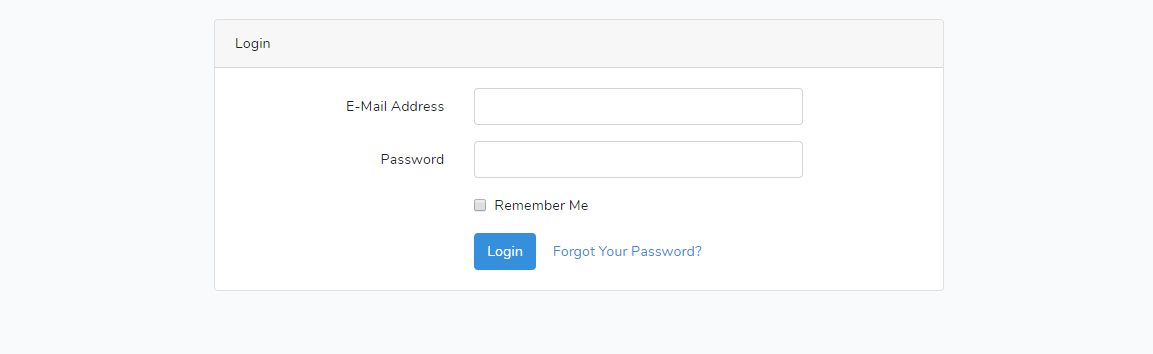


Рисунок 1 – Форма для входа в панель администратора

Сразу после удачного входа в панель администратора, пользователь видит меню со ссылками на различные страницы администрирования. Данная страница изображена на рисунке 2.



Рисунок 2 – Панель администратора сайта

На странице “Новости” администратору становится доступна таблица со списком новостей, которые можно выбрать для редактирования (Рисунок 3).

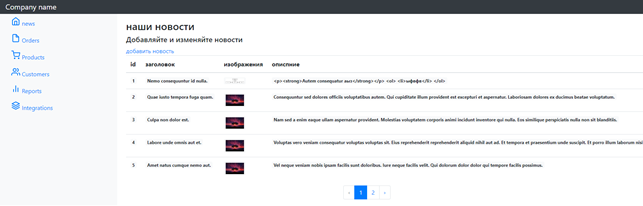


Рисунок 3 – Страница выбора новости для редактирования

Выбрав новость, пользователь попадает на страницу с формой редактирования конкретной новости. Она содержит текстовые поля с заголовком, автором, поле выбора изображения, дату новости. Кроме того, форма содержит богатые текстовые редакторы для самих текстов новостей.

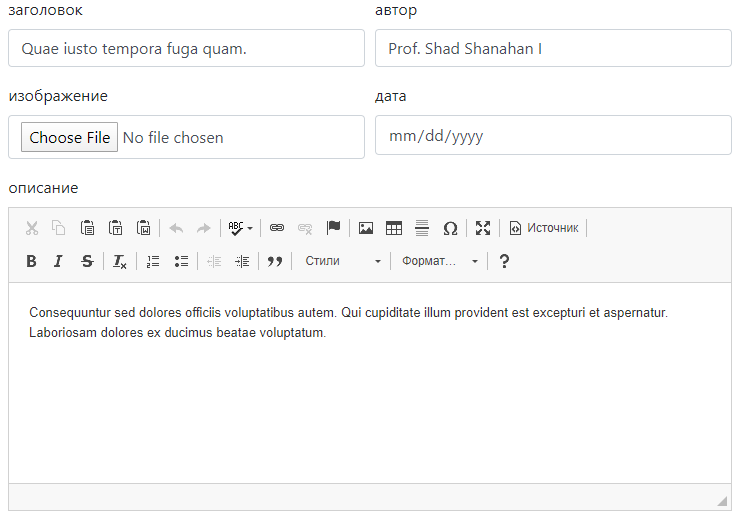


Рисунок 4 – Форма добавления/редактирования новостей

На странице заявок можно подтвердить/удалить заявку:

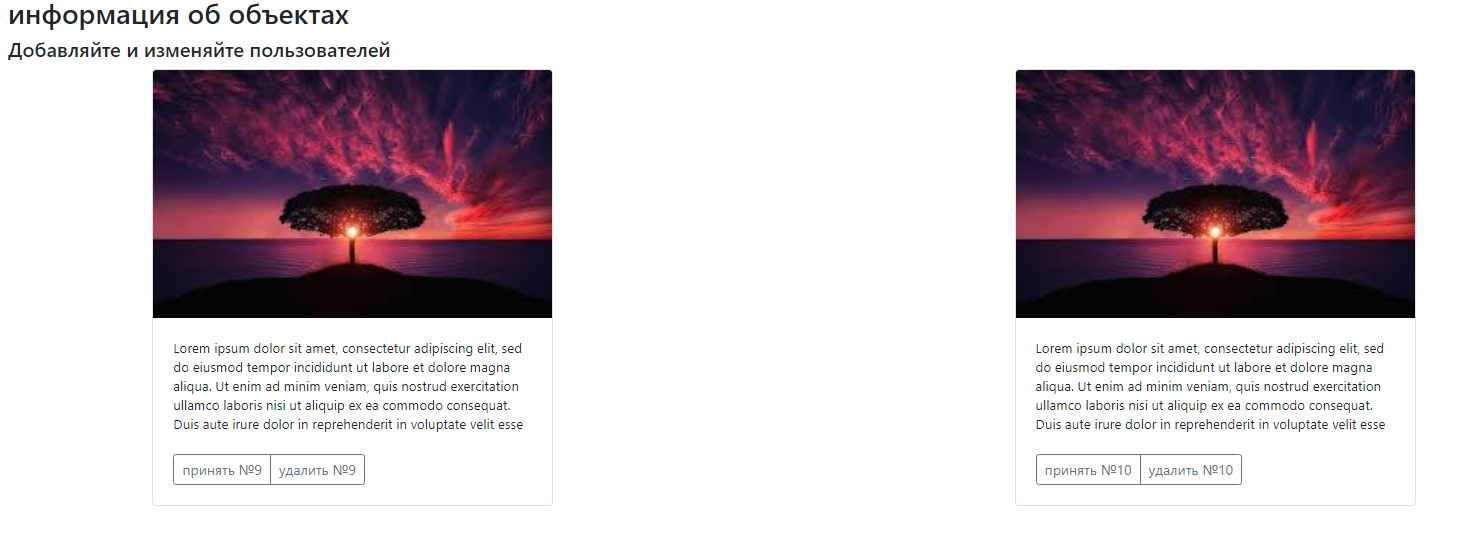


Рисунок 5 – Страница заявок

После перехода на страницу объектов становится доступен список объектов недвижимости:

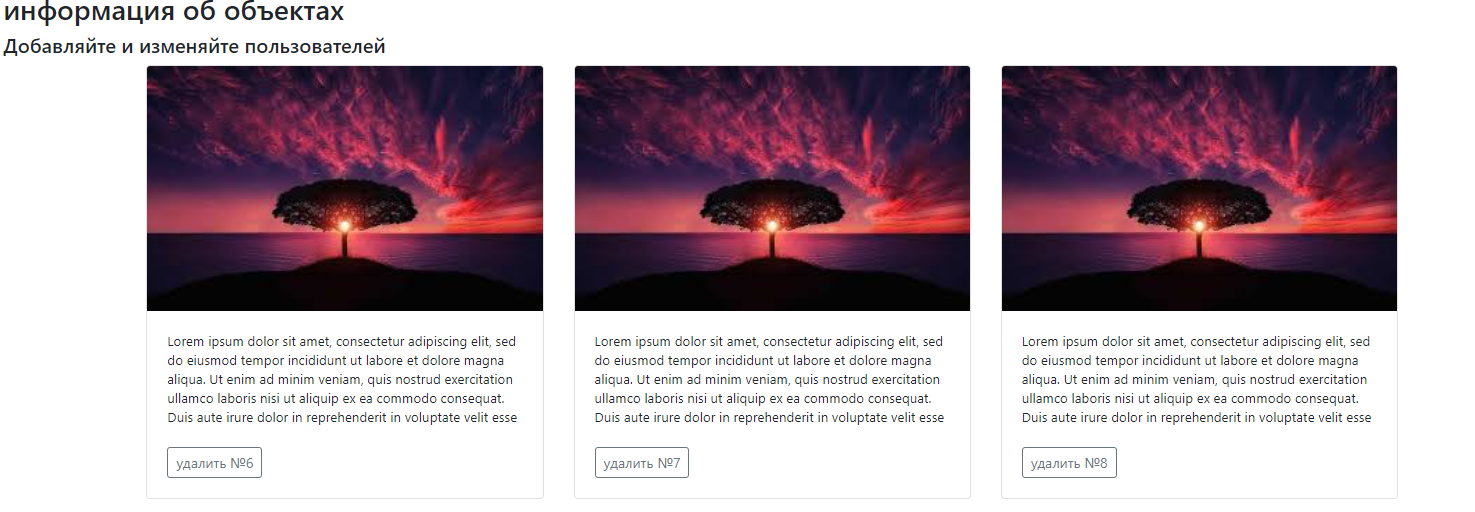


Рисунок 6 – Страница “Объекты”

При переходе на страницу редактирования пользователей, становится доступна таблица пользователей, которых можно редактировать (рисунок 7).

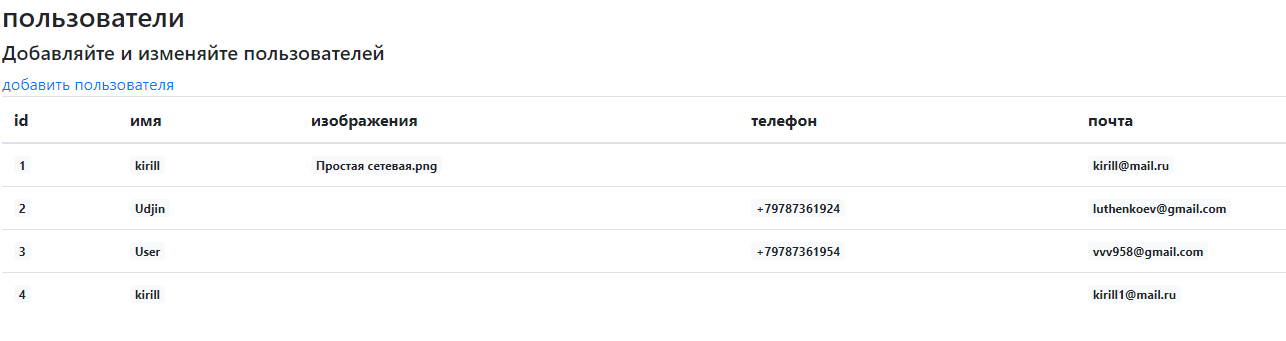


Рисунок 7 – Список доступных для редактирования пользователей

**5.2 Разработка программных модулей раздела администратора**

Разработка осуществлялась по шаблону проектирования MVC на фреймворке Laravel, о чем свидетельствует структура проекта, изображенная на рисунке 8.

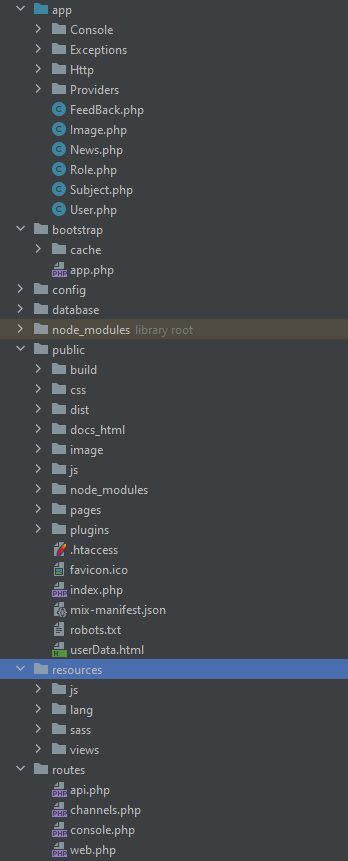


Рисунок 8 – Структура проекта

Папка Controllers содержит контроллеры, которые содержат методы для работы с моделью. В качестве примера метода из контроллера можно привести метод создания новости:

public function newsCreate(Request $request){

if(Gate::allows('admin-enter')){

$news=new News();

$news->author=$request->author;

$news->news=$request->news;

$news->news\_short=$request->news\_short;

$news->created\_at=$request->created\_at;

$news->header=$request->header;

if($request->image!=""){

$image = $request->file('image');

$avatarName = $image->getClientOriginalName();

$image->move(public\_path('image'),$avatarName);

$news->image = '/image/'.$avatarName;

}

$news->save();

return view('admin.admin')->with(['news'=>$news]);

}

return view('index');

}

Папка public содержит файл index.php, который является входной точкой к проекту. Через него проходят все запросы. Кроме того, public содержит общедоступные файлы, такие как JS и CSS.

В папке database содержатся миграции и сидеры для версионирования базы и заполнения ее данными. Например, сидер создания новостей:

*class NewsSeeder extends Seeder*

*{*

*/\*\**

*\* Run the database seeds.*

*\**

*\* @return void*

*\*/*

*public function run()*

*{*

*$faker = Faker::create('App\News');*

*for($i=0;$i<10;$i++){*

*DB::table('News')->insert([*

*'news'=>$faker->text(200),*

*'header'=>$faker->text(30),*

*'image'=>'/image/test.jpg',*

*'news\_short'=>$faker->text(50),*

*'author'=>$faker->name,*

*'created\_at'=>$faker->dateTimeBetween('-30 days','+30 days'),*

*]);*

*}*

*}*

*}*

В качестве View из шаблона проектирования MVC выступают представления, построенные на шаблонизаторе blade. Они находятся по пути resources/views. В качестве примера можно привести представление страницы контактов:

*<div class="row" style="margin-top: 70px">*

*<div class="col-md-6 col-sm-12 mx-auto" style="height: 60vh;" id="map">*

*</div>*

*<div class="col-6">*

*<h1>Контакты</h1>*

*<p>г.Севастополь ул.Большая Морская 22 2</p>*

*<p>почта example@mail.com</p>*

*<a class="mx-auto" href="tel:+79785555555">+79785555555</a>*

*</div>*

*</div>*

*<div class="row">*

*<div class="col-8 mx-auto" >*

*</div>*

*</div>*

*{{-- цыкл для вывода слайдера и рядом лежащего--}}*

*@include('layouts.footer')*

*<script type="text/javascript">*

*let lat =44.7419;*

*let long =33.7319;*

*ymaps.ready(init);*

*function init(){*

*var myMap = new ymaps.Map("map", {*

*center: [lat, long],*

*zoom: 10*

*}, {*

*searchControlProvider: 'yandex#search'*

*}),*

*yellowCollection = new ymaps.GeoObjectCollection(null, {*

*preset: 'islands#yellowIcon'*

*}),*

*yellowCoords = [[lat, long]];*

*for (var i = 0, l = yellowCoords.length; i < l; i++) {*

*yellowCollection.add(new ymaps.Placemark(yellowCoords[i]));*

*}*

*myMap.geoObjects.add(yellowCollection);*

*// Через коллекции можно подписываться на события дочерних элементов.*

*yellowCollection.events.add('click', function () { alert('Кликнули по желтой метке') });*

Для описания путей в файле web.php описаны основные точки доступа в приложение:

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

*return view('index');*

*});*

*Auth::routes();*

*Route::get('/admin','AdminController@index')->middleware('auth');*

*Route::get('/admin/news','AdminController@news')->middleware('auth');*

*Route::get('/admin/newsUpdate/{id}','AdminController@newsUpdatePage')->middleware('auth');*

*Route::post('/admin/newsUpdate/{id}','AdminController@newsUpdate')->middleware('auth');*

*Route::post('/admin/newsDelete/{id}','AdminController@newsDelete')->middleware('auth');*

*Route::post('/admin/newsCreate','AdminController@newsCreate')->middleware('auth');*

*Route::get('/admin/newsCreate','AdminController@newsCreatePage')->middleware('auth');*

*Route::get('/admin/users','AdminController@users')->middleware('auth');*

*Route::get('/admin/objects','AdminController@objects')->middleware('auth');*

*Route::get('/admin/nonCheckedObjects','AdminController@nonChecked')->middleware('auth');*

*Route::get('/admin/objectCheck/add/{id}','AdminController@Checked')->middleware('auth');*

*Route::get('/admin/objectCheck/delete/{id}','AdminController@objectDelete')->middleware('auth');*

*Route::get('/admin/object/{id}','AdminController@objectDelete')->middleware('auth');*

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

*Route::get('/home/logout', 'HomeController@logOut');*

*Route::get('/home/objects', 'HomeController@objects');*

*Route::get('/home/object/{id}', 'HomeController@object');*

*Route::get('/home/addObject', 'HomeController@addObjectPage');*

*Route::post('/object/add', 'HomeController@addObject');*

*Route::post('/object/update/{id}', 'HomeController@updateObject');*

*Route::post('/object/delete/{id}', 'HomeController@deleteObject');*

*Route::post('/userData/update','HomeController@updateUserData');*

*Route::get('/offer', 'OfferController@index')->name('offer');*

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

*return view('index');*

*})->name('index');*

*Route::get('/object/{id}','OfferController@object');*

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

*Route::get('/news/{id}','NewsController@getOne');*

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

*return view('welcome');*

*});*

*Route::get('/history','HistoryController@index');*

*Route::get('/services','aboutUsController@services');*

*Route::get('/reward','aboutUsController@reward');*

*Route::get('/partners','aboutUsController@partners');*

*Route::get('/contacts','aboutUsController@contacts');*

Модели приложения описаны в корне папки app. Модель пользователя:

*class User extends Authenticatable*

*{*

*use Notifiable;*

*public function isAdmin(){*

*if($this->roles[0]->role==='admin'){*

*return true;*

*}else{*

*return false;*

*}*

*}*

*/\*\**

*\* The attributes that are mass assignable.*

*\**

*\* @var array*

*\*/*

*protected $fillable = [*

*'name', 'email', 'password','phone','image',*

*];*

*/\*\**

*\* The attributes that should be hidden for arrays.*

*\**

*\* @var array*

*\*/*

*protected $hidden = [*

*'password', 'remember\_token',*

*];*

*/\*\**

*\* The attributes that should be cast to native types.*

*\**

*\* @var array*

*\*/*

*protected $casts = [*

*'email\_verified\_at' => 'datetime',*

*];*

*public function roles(){*

*return $this->belongsToMany('App\Role');*

*}*

*public function feedBacks(){*

*return $this->hasMany('App\FeedBack');*

*}*

*public function subject(){*

*return $this->hasMany('App\Subject');*

*}*

*public function subjects(){*

*return $this->belongsToMany('App\Subject');*

*}*

*}*

Модель объекта недвижимости:

*class Subject extends Model*

*{*

*public $timestamps =true;*

*protected $fillable=['user\_id','long','lat','images','description','price'];*

*public function user(){*

*return $this->belongsTo('App\User');*

*}*

*public function users(){*

*return $this->belongsToMany('App\User');*

*}*

*}*

**6 РАЗРАБОТКА РАЗДЕЛА ПОЛЬЗОВАТЕЛЯ**

**6.1 Проектирование интерфейса раздела пользователя**

Интерфейс пользователей-посетителей имеет свободный доступ и содержит структурированную информацию о компании, и её продукции. Пользовательский интерфейс содержит статические разделы (разделы, для изменения которых необходимо привлечение разработчика или стороннего WEB-программиста) и динамические разделы (содержимое которых формируется программно по информации добавленной администратором сайта и хранящейся в базе данных).

Верхнее меню состоит из логотипа компании, ссылок для перехода на другие страницы, ссылки для авторизации:



Рисунок 9 – Верхнее меню сайта

Кроме того, меню адаптируется под узкие экраны:

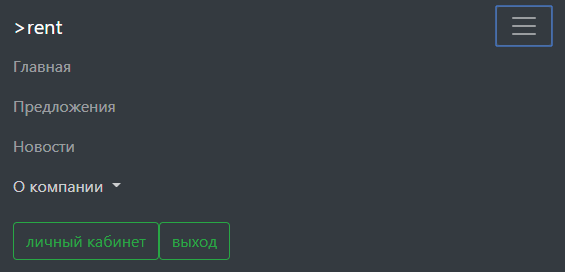


Рисунок 10 – Верхнее меню сайта

Главная страница сайта является лендинговой:

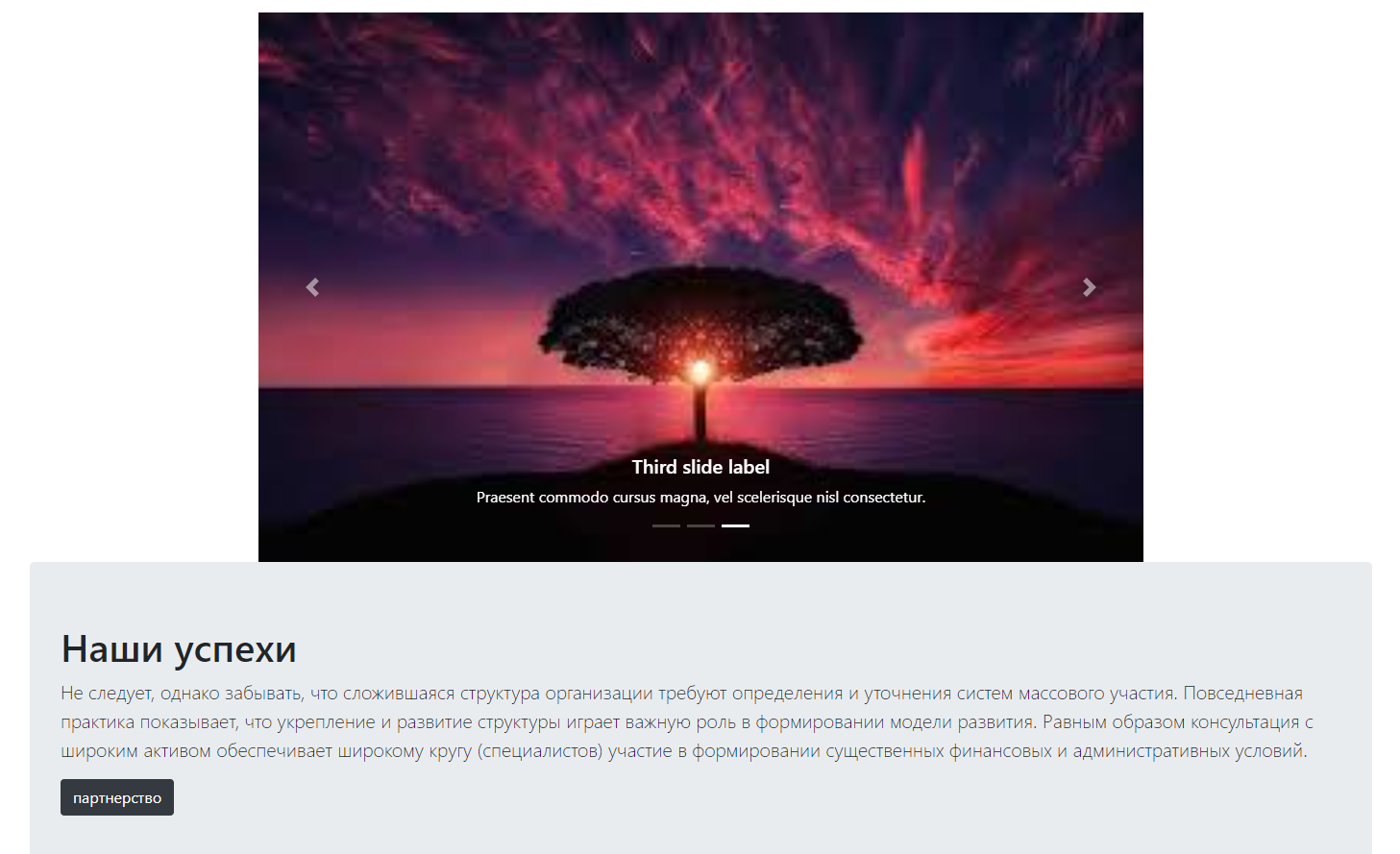


Рисунок 11 – Лендинговая страница

На странице предложений можно просмотреть список доступной недвижимости, а также ее расположение на карте. Страница состоит из двух частей: слева непосредственно список, справа - карта.

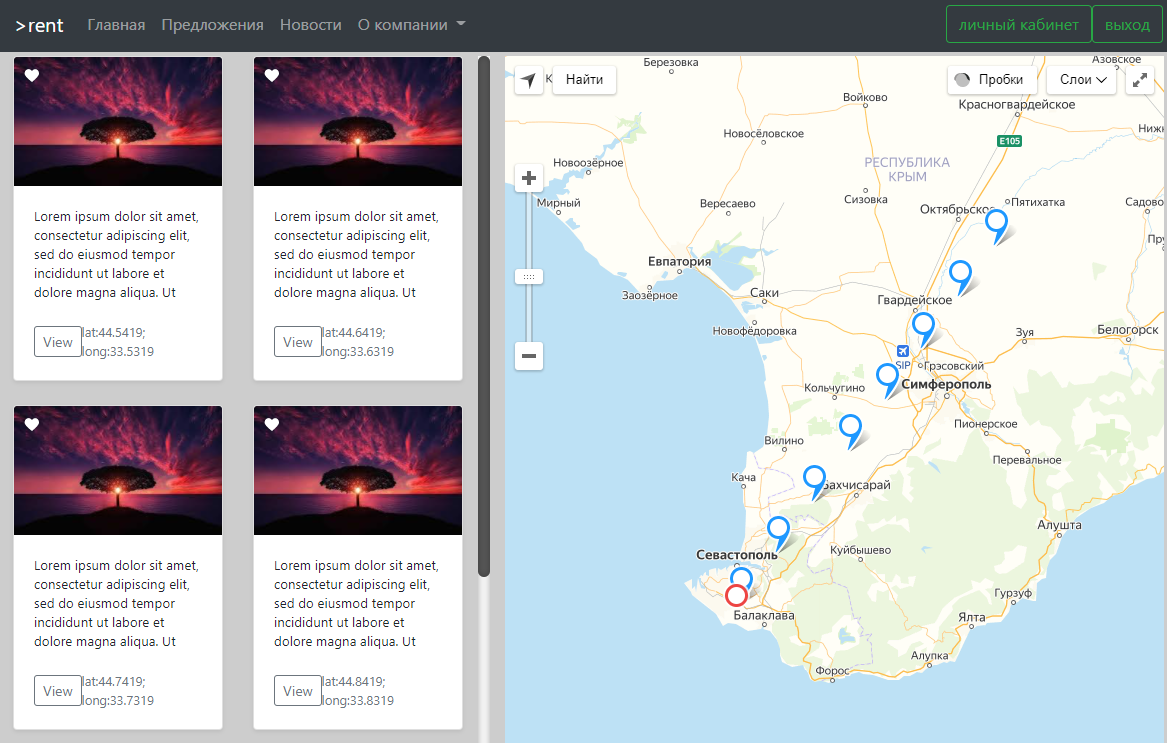


Рисунок 12 - Страница предложений

Кроме того, страница адаптирована для мобильных устройств:

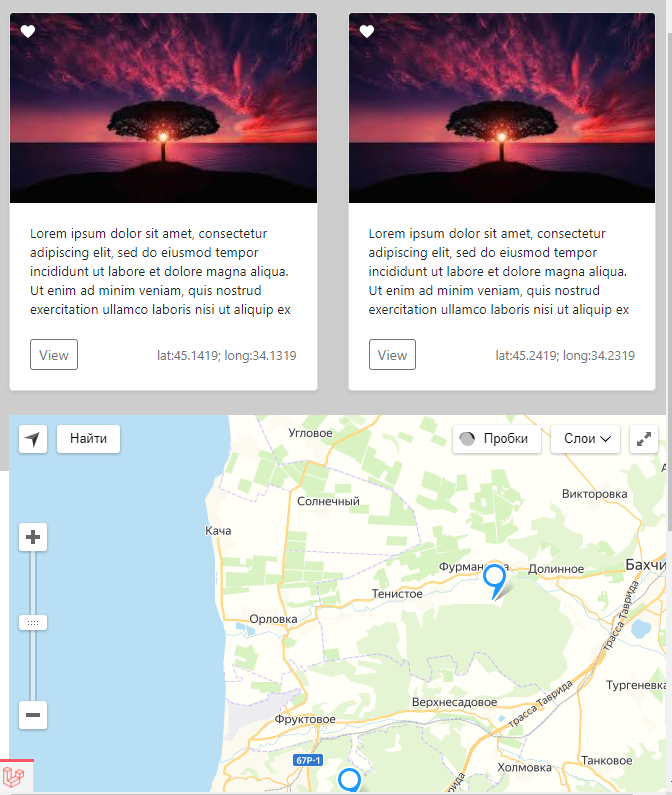


Рисунок 13 - Адаптация страницы предложений для мобильных устройств

При переходе на отдельное предложение, открывается его описание, цена, расположение на карте, контакты доступа.

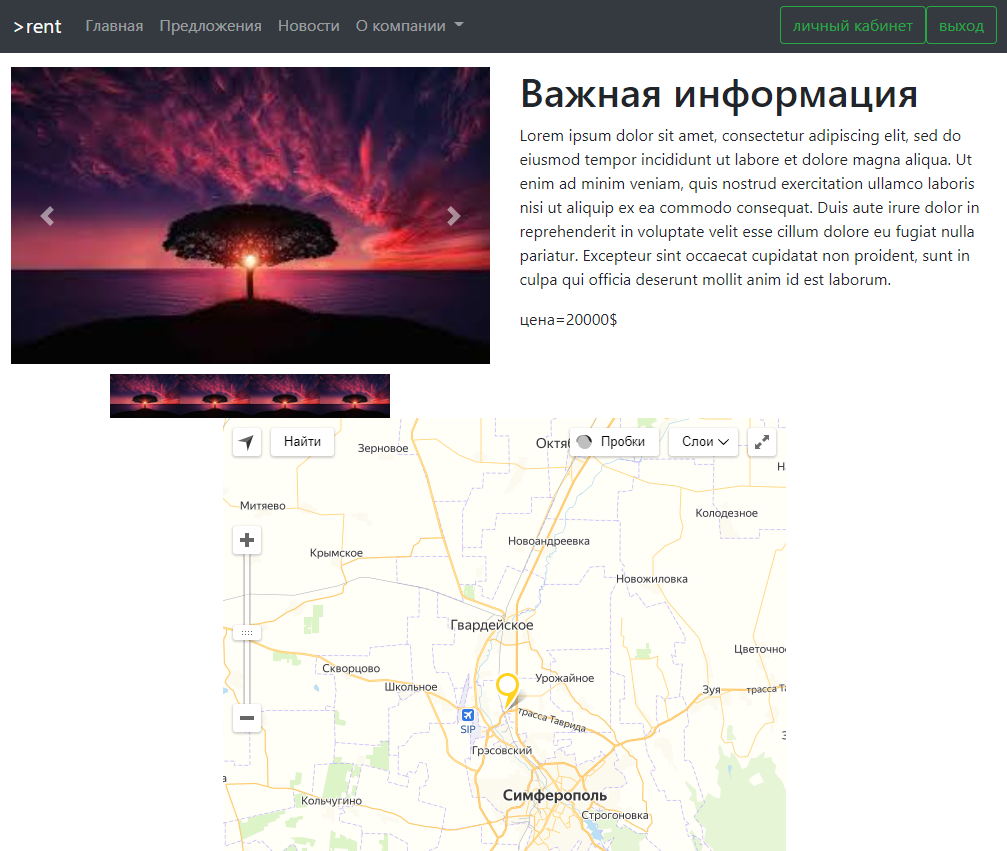


Рисунок 14 – Страница предложения

На странице новостей содержится список существующих новостей:

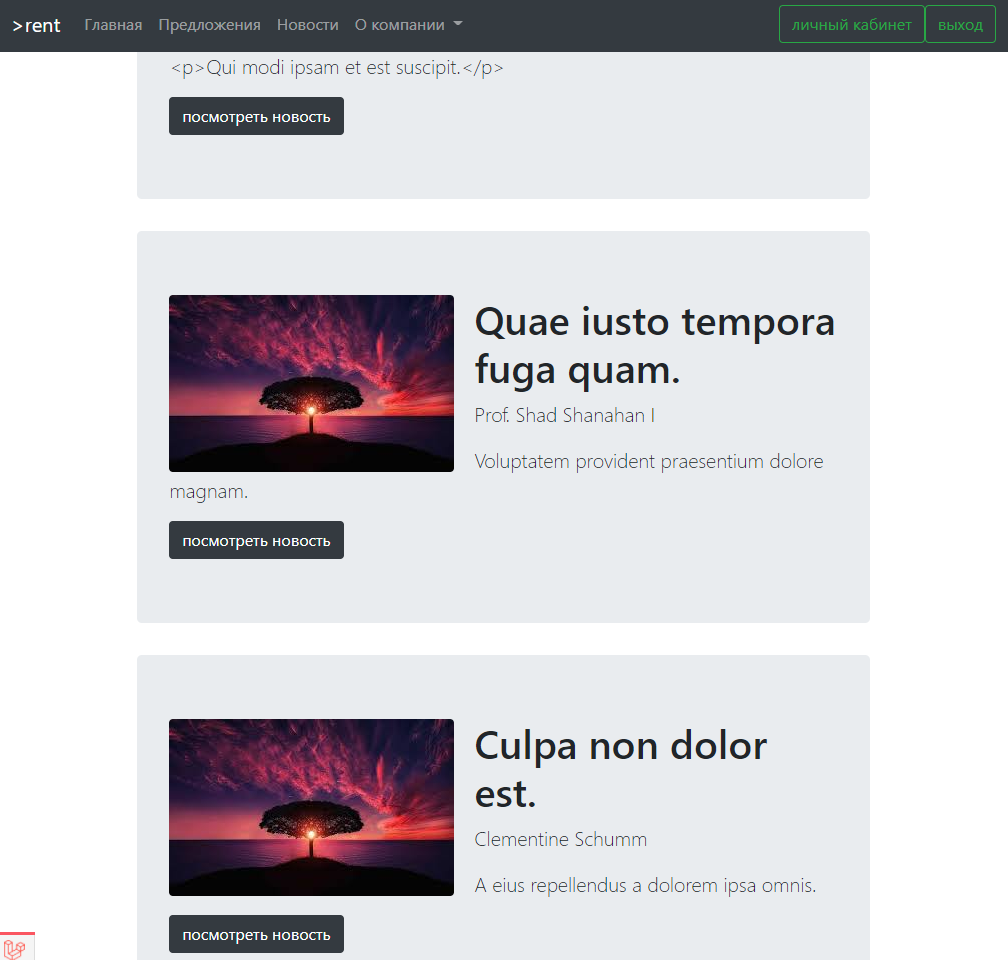


Рисунок 15 - Страница новостей

Также можно рассмотреть отдельную новость:

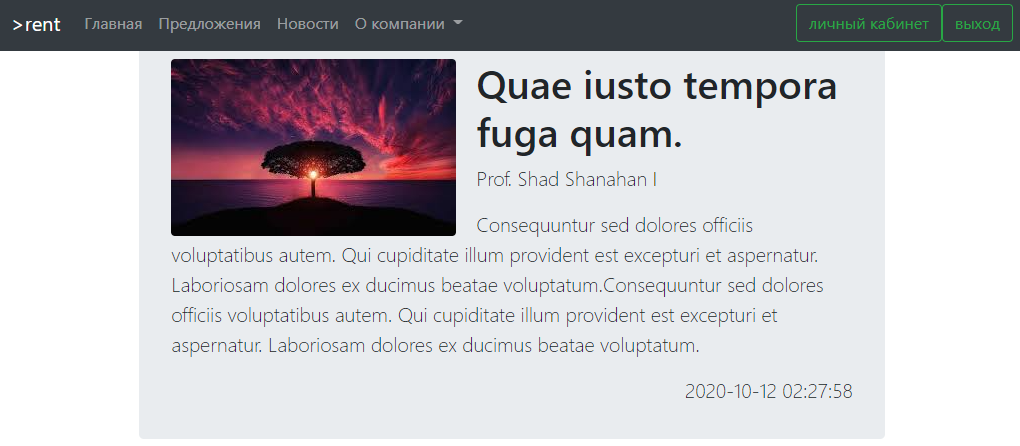


Рисунок 16 - Просмотр отдельной новости

На страницах “О Компании - история”, “О Компании - услуги”, “О Компании - как с нами связаться” представлен статический контент и отзывы пользователей:

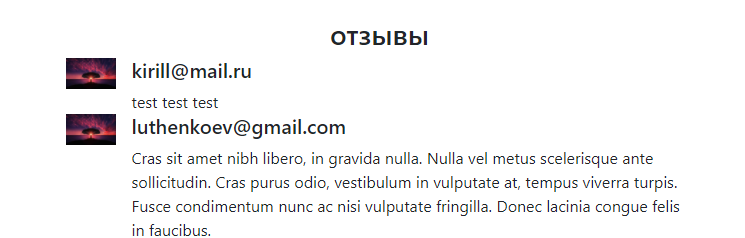


Рисунок 17 - Отзывы пользователей

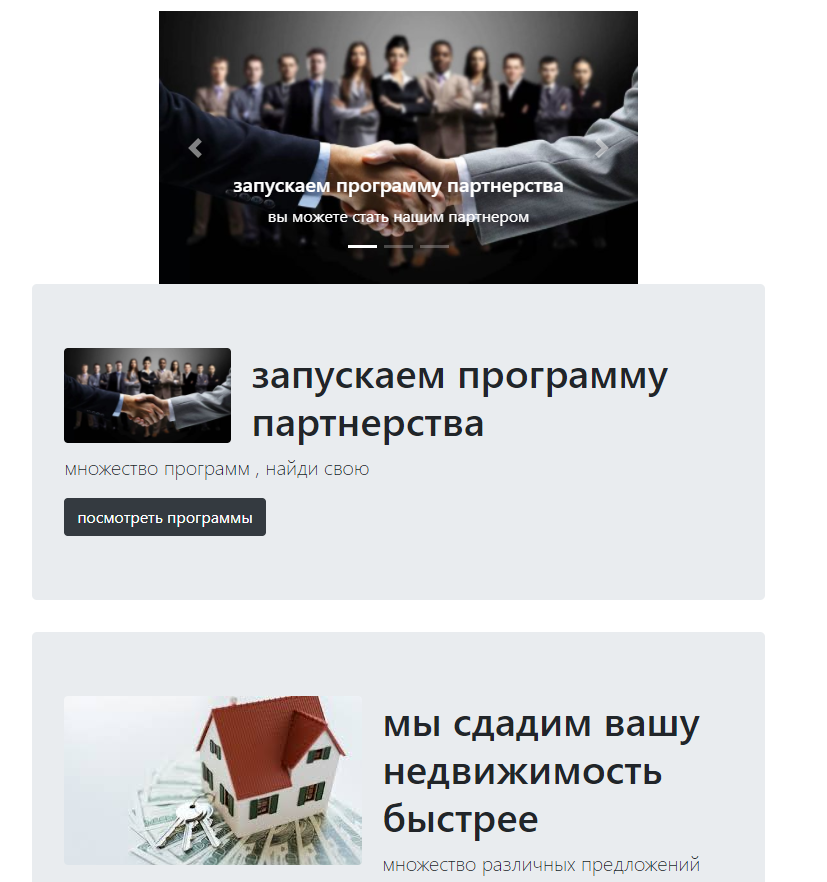


Рисунок 18 - Страница списка услуг компании

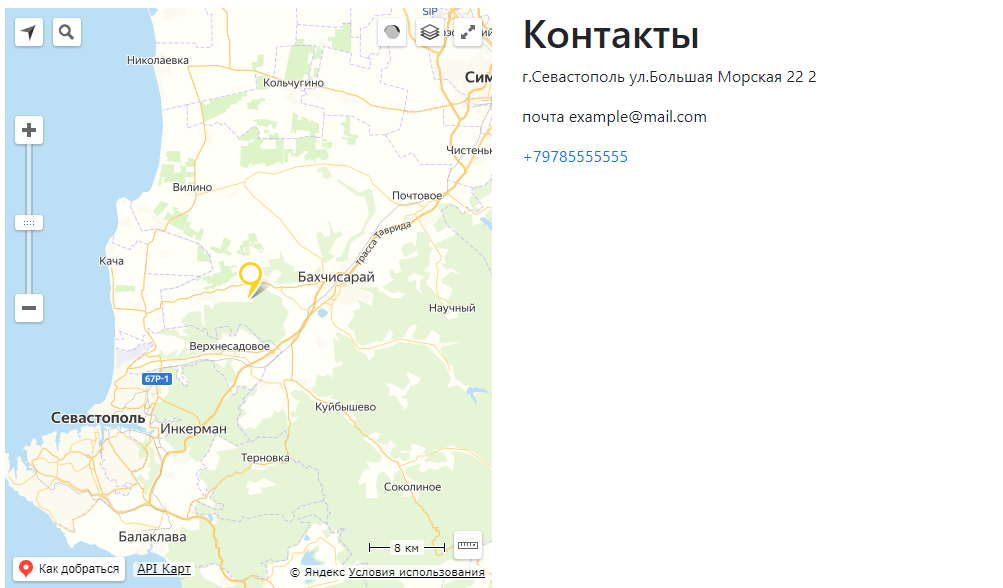


Рисунок 19 - Страница с контактами и местонахождением компании

**6.2 Разработка программных модулей раздела пользователя**

При разработке раздела пользователя использовались те же программные средства, что и при создании части, доступной для администратора. Разница лишь в том, что при заходе на страницы, доступные для обычных пользователей не проверяются роли пользователя.

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте был разработан WEB-сайт фирмы по аренде недвижимости. Было разработано два независимых интерфейса пользователей: интерфейс пользователей-посетителей и интерфейс пользователя-администратора сайта.

Для интерфейса администратора был организован авторизированный доступ. Интерфейс пользователя-администратора обеспечивает администратору сайта возможность добавлять, изменять и удалять информацию некоторых разделов сайта. Информация, добавляемая администратором, сохраняется в базе данных и отображается в соответствующих разделах пользовательского интерфейса WEB-сайта.

Для интерфейса пользователей-посетителей организован свободный доступ. Пользовательский интерфейс содержит статические разделы и динамические разделы.

Были закреплены навыки проектирования базы данных, проведения их нормализации, построения физической модели и разработки программного обеспечения.

В результате выполнения курсового проекта был получен WEB-сайт фирмы по аренде недвижимости, обеспечивающий предоставление информации о фирме в сети Интернет с целью расширения сферы её деятельности, круглосуточного привлечения новых клиентов и осуществления удаленное демонстрации товара и предоставления помощи в выборе предлагаемой продукции.

**СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ**

1. Методические указания к выполнению курсовой работы по дисциплине «Web-технологии» для студентов направления 09.03.02 «Информационные системы и технологии» / А. Л. Овчинников. — Севастополь: Изд-во СевГУ, 2015. — 31 с.

2. Node.js [Электронный ресурс]. URL: https://nodejs.org/en/ (дата обращения: 20.10.2020).

3. PHP [Электронный ресурс]. URL: https://http://doc.php.net/ (дата обращения: 20.10.2020).

4. MySQL [Электронный ресурс]. URL: https://dev.mysql.com/doc/ (дата обращения: 20.10.2020).

**ПРИЛОЖЕНИЕ А**

КОД ПРИЛОЖЕНИЯ

Файл web.php

<?php

use Illuminate\Support\Facades\Route;

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

return view('index');

});

Auth::*routes*();

Route::*get*('/admin','AdminController@index')->middleware('auth');

Route::*get*('/admin/news','AdminController@news')->middleware('auth');

Route::*get*('/admin/newsUpdate/{id}','AdminController@newsUpdatePage')->middleware('auth');

Route::*post*('/admin/newsUpdate/{id}','AdminController@newsUpdate')->middleware('auth');

Route::*post*('/admin/newsDelete/{id}','AdminController@newsDelete')->middleware('auth');

Route::*post*('/admin/newsCreate','AdminController@newsCreate')->middleware('auth');

Route::*get*('/admin/newsCreate','AdminController@newsCreatePage')->middleware('auth');

Route::*get*('/admin/users','AdminController@users')->middleware('auth');

Route::*get*('/admin/objects','AdminController@objects')->middleware('auth');

Route::*get*('/admin/nonCheckedObjects','AdminController@nonChecked')->middleware('auth');

Route::*get*('/admin/objectCheck/add/{id}','AdminController@Checked')->middleware('auth');

Route::*get*('/admin/objectCheck/delete/{id}','AdminController@objectDelete')->middleware('auth');

Route::*get*('/admin/object/{id}','AdminController@objectDelete')->middleware('auth');

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

Route::*get*('/home/logout', 'HomeController@logOut');

Route::*get*('/home/objects', 'HomeController@objects');

Route::*get*('/home/object/{id}', 'HomeController@object');

Route::*get*('/home/addObject', 'HomeController@addObjectPage');

Route::*post*('/object/add', 'HomeController@addObject');

Route::*post*('/object/update/{id}', 'HomeController@updateObject');

Route::*post*('/object/delete/{id}', 'HomeController@deleteObject');

Route::*post*('/userData/update','HomeController@updateUserData');

Route::*get*('/offer', 'OfferController@index')->name('offer');

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

return view('index');

})->name('index');

Route::*get*('/object/{id}','OfferController@object');

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

Route::*get*('/news/{id}','NewsController@getOne');

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

return view('welcome');

});

Route::*get*('/history','HistoryController@index');

Route::*get*('/services','aboutUsController@services');

Route::*get*('/reward','aboutUsController@reward');

Route::*get*('/partners','aboutUsController@partners');

Route::*get*('/contacts','aboutUsController@contacts');

Файл ConfirmPasswordController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Providers\RouteServiceProvider;

use Illuminate\Foundation\Auth\ConfirmsPasswords;

class ConfirmPasswordController extends Controller

{

use ConfirmsPasswords;

*/\*\**

*\* Where to redirect users when the intended url fails.*

*\**

*\** ***@var*** *string*

*\*/*

protected $redirectTo = RouteServiceProvider::*HOME*;

*/\*\**

*\* Create a new controller instance.*

*\**

*\** ***@return*** *void*

*\*/*

public function \_\_construct()

{

$this->middleware('auth');

}

}

Файл ForgotPasswordController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use Illuminate\Foundation\Auth\SendsPasswordResetEmails;

class ForgotPasswordController extends Controller

{

use SendsPasswordResetEmails;

}

Файл LoginController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Providers\RouteServiceProvider;

use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller

{

use AuthenticatesUsers;

*/\*\**

*\* Where to redirect users after login.*

*\**

*\** ***@var*** *string*

*\*/*

protected $redirectTo = '/admin';

*/\*\**

*\* Create a new controller instance.*

*\**

*\** ***@return*** *void*

*\*/*

public function \_\_construct()

{

$this->middleware('guest')->except('logout');

}

}

Файл RegisterController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Providers\RouteServiceProvider;

use App\User;

use Illuminate\Foundation\Auth\RegistersUsers;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller

{

use RegistersUsers;

*/\*\**

*\* Where to redirect users after registration.*

*\**

*\** ***@var*** *string*

*\*/*

protected $redirectTo = '/admin';

*/\*\**

*\* 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, [

'name' => ['required', 'string', 'max:255'],

'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],

'password' => ['required', 'string', 'min:8', 'confirmed'],

]);

}

*/\*\**

*\* Create a new user instance after a valid registration.*

*\**

*\** ***@param*** *array $data*

*\** ***@return*** *\App\User*

*\*/*

protected function create(array $data)

{

return User::*create*([

'name' => $data['name'],

'email' => $data['email'],

'password' => Hash::*make*($data['password']),

]);

}

}

Файл ResetPasswordController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Providers\RouteServiceProvider;

use Illuminate\Foundation\Auth\ResetsPasswords;

class ResetPasswordController extends Controller

{

use ResetsPasswords;

*/\*\**

*\* Where to redirect users after resetting their password.*

*\**

*\** ***@var*** *string*

*\*/*

protected $redirectTo = RouteServiceProvider::*HOME*;

}

Файл VerificationController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Providers\RouteServiceProvider;

use Illuminate\Foundation\Auth\VerifiesEmails;

class VerificationController extends Controller

{

use VerifiesEmails;

*/\*\**

*\* Where to redirect users after verification.*

*\**

*\** ***@var*** *string*

*\*/*

protected $redirectTo = RouteServiceProvider::*HOME*;

*/\*\**

*\* Create a new controller instance.*

*\**

*\** ***@return*** *void*

*\*/*

public function \_\_construct()

{

$this->middleware('auth');

$this->middleware('signed')->only('verify');

$this->middleware('throttle:6,1')->only('verify', 'resend');

}

}

Файл aboutUsController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class aboutUsController extends Controller

{

public function services(){

return view('services');

}

public function contacts(){

return view('contacts');

}

}

Файл AdminController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\User;

use Gate;

use App\News;

use App\Subject;

use Illuminate\Support\Facades\DB;

class AdminController extends Controller

{

public function index(){

if(Gate::*allows*('admin-enter')){

return view('admin.admin');

}

return view('index');

}

public function nonChecked()

{

$subjects=Subject::*where*('checked',0)->paginate(5);

return view('admin.objectsNonChecked')->with(['subjects'=>$subjects]);

}

public function news(){

if(Gate::*allows*('admin-enter')){

$news=News::*paginate*(5);

return view('admin.news')->with(['news'=>$news]);

}

return view('index');

}

public function newsUpdatePage($id){

if(Gate::*allows*('admin-enter')){

$news=News::*find*($id);

return view('admin.newsUpdate')->with(['news'=>$news]);

}

return view('index');

}

public function newsUpdate($id,Request $request){

if(Gate::*allows*('admin-enter')){

$news=News::*find*($id);

$news->author=$request->author;

$news->news=$request->news;

$news->news\_short=$request->news\_short;

$news->created\_at=$request->created\_at;

$news->header=$request->header;

if($request->image!=""){

$image = $request->file('image');

$avatarName = $image->getClientOriginalName();

$image->move(public\_path('image'),$avatarName);

$news->image = '/image/'.$avatarName;

}

$news->save();

return view('admin.admin')->with(['news'=>$news]);

}

return view('index');

}

public function newsDelete($id){

if(Gate::*allows*('admin-enter')){

DB::*table*('news')->where('id', $id)->delete();

return view('admin.admin');

}

return view('index');

}

public function newsCreate(Request $request){

if(Gate::*allows*('admin-enter')){

$news=new News();

$news->author=$request->author;

$news->news=$request->news;

$news->news\_short=$request->news\_short;

$news->created\_at=$request->created\_at;

$news->header=$request->header;

if($request->image!=""){

$image = $request->file('image');

$avatarName = $image->getClientOriginalName();

$image->move(public\_path('image'),$avatarName);

$news->image = '/image/'.$avatarName;

}

$news->save();

return view('admin.admin')->with(['news'=>$news]);

}

return view('index');

}

public function newsCreatePage(){

if(Gate::*allows*('admin-enter')){

return view('admin.newsCreate');

}

return view('index');

}

public function Checked($id){

if(Gate::*allows*('admin-enter')){

$subject=Subject::*find*($id);

$subject->checked=1;

$subject->save();

return view('admin.admin');

}

return view('index');

}

public function users(){

if(Gate::*allows*('admin-enter')){

$users=User::*paginate*(5);

return view('admin.users')->with(['users'=>$users]);

}

return view('index');

}

public function objects(){

if(Gate::*allows*('admin-enter')){

$subjects=Subject::*paginate*(5);

return view('admin.objects')->with(['subjects'=>$subjects]);

}

return view('index');

}

public function objectDelete($id){

if(Gate::*allows*('admin-enter')){

$subjects=Subject::*find*($id);

DB::*table*('subjects')->where('id',$id)->delete();

return view('admin.admin');

}

return view('index');

}

}

Файл Controller.php

<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

use Illuminate\Foundation\Bus\DispatchesJobs;

use Illuminate\Foundation\Validation\ValidatesRequests;

use Illuminate\Routing\Controller as BaseController;

class Controller extends *BaseController*

{

use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

}

Файл HistoryController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\FeedBack;

class HistoryController extends Controller

{

public function index(){

$feedBacks=FeedBack::*all*();

return view('history')->with(['feedBacks'=>$feedBacks]);

}

}

Файл HomeController.php

<?php

namespace App\Http\Controllers;

use App\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Image;

use App\Subject;

class HomeController extends Controller

{

*/\*\**

*\* Create a new controller instance.*

*\**

*\** ***@return*** *void*

*\*/*

public function \_\_construct()

{

$this->middleware('auth');

}

*/\*\**

*\* Show the application dashboard.*

*\**

*\** ***@return*** *\Illuminate\Contracts\Support\Renderable*

*\*/*

public function index()

{

if(Auth::*check*()) {

return view('homeProfile')->with(['user'=>Auth::*user*()]);

}else{

return redirect()->route('index');

}

}

public function objects()

{

if(Auth::*check*()) {

$subjects=Subject::*where*('user\_id',Auth::*user*()->id)->get();

return view('homeSubjects')->with(['subjects'=>$subjects]);

}else{

return redirect()->route('index');

}

}

public function object($id)

{

if(Auth::*check*()) {

$subject=Subject::*where*('id',$id)->get();

$subject=Subject::*find*($id);

return view('homeSubject')->with(['subject'=>$subject]);

}else{

return redirect()->route('index');

}

}

public function updateObject(Request $request,$id)

{

if(Auth::*check*()) {

$subject=Subject::*find*($id);

$subject->price=$request->price;

$subject->lat=$request->lat;

$subject->long=$request->long;

$subject->description=$request->description;

if($request->image!=""){

$image = $request->file('image');

$avatarName = $image->getClientOriginalName();

$image->move(public\_path('image'),$avatarName);

$image=json\_decode($subject->images,true);

array\_push($image,$avatarName);

$subject->images = json\_encode($image);

}

$subject->save();

return view('homeSubject')->with(['subject'=>$subject]);

}else{

return redirect()->route('index');

}

}

public function addObjectPage(Request $request){

if(Auth::*check*()) {

return view('homeAddSubject');

}else{

return redirect()->route('index');

}

}

public function addObject(Request $request)

{

if(Auth::*check*()) {

$subject=new Subject();

$subject->price=$request->price;

$subject->lat=$request->lat;

$subject->long=$request->long;

$subject->description=$request->description;

if($request->image!=""){

$image = $request->file('image');

$avatarName = $image->getClientOriginalName();

$image->move(public\_path('image'),$avatarName);

$subject->images = json\_encode(['k1'=>'/image/'.$avatarName]);

}

$subject->save();

return view('homeAddSubject');

}else{

return redirect()->route('index');

}

}

public function deleteObject(Request $request,$id)

{

if(Auth::*check*()) {

$subject=Subject::*where*('id',$id)->delete();

return view('homeSubject')->with(['subject'=>$subject]);

}else{

return redirect()->route('index');

}

}

public function logOut()

{

if(Auth::*check*()) {

Auth::*logout*();

return redirect()->route('index');

}

return redirect()->route('index');

}

public function updateUserData(Request $request)

{

if(Auth::*check*()) {

$user=Auth::*user*();

$user->email=$request->email;

$user->name=$request->name;

$user->phone=$request->phone;

if($request->password!=""){

$user->password=bcrypt($request->password);

}

if($request->pic!=""){

$image = $request->file('pic');

$avatarName = $image->getClientOriginalName();

$image->move(public\_path('image'),$avatarName);

$user->image = $avatarName;

}

$user->save();

return view('homeProfile')->with(['user'=>Auth::*user*()]);

}else{

redirect()->route('index');

}

}

}

Файл NewsController.php

<?php

namespace App\Http\Controllers;

use App\News;

use Illuminate\Http\Request;

class NewsController extends Controller

{

public function index(){

$news=News::*paginate*(8);

return view('news')->with(['news'=>$news]);

}

public function getOne($id){

$news=News::*find*($id);

return view('news\_one')->with(['news'=>$news]);

}

}

Файл OfferController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Subject;

class OfferController extends Controller

{

public function index(){

$subjects=Subject::*where*('checked',1)->get();

return view('offer')->with(['subjects'=>$subjects]);

}

public function object(Request $Request,$id=null){

if($id) {

$subject = Subject::*find*($id);

return view('object')->with(['subject' => $subject]);

}

return redirect('/offer');

}

}

Файл Authenticate.php

<?php

namespace App\Http\Middleware;

use Illuminate\Auth\Middleware\Authenticate as Middleware;

class Authenticate extends *Middleware*

{

*/\*\**

*\* Get the path the user should be redirected to when they are not authenticated.*

*\**

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

*\** ***@return*** *string|null*

*\*/*

protected function redirectTo($request)

{

if (! $request->expectsJson()) {

return route('login');

}

}

}

Файл CheckForMaintenanceMode.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode as Middleware;

class CheckForMaintenanceMode extends *Middleware*

{

*/\*\**

*\* The URIs that should be reachable while maintenance mode is enabled.*

*\**

*\** ***@var*** *array*

*\*/*

protected $except = [

//

];

}

Файл RedirectIfAuthenticated.php

<?php

namespace App\Http\Middleware;

use App\Providers\RouteServiceProvider;

use Closure;

use Illuminate\Support\Facades\Auth;

class RedirectIfAuthenticated

{

*/\*\**

*\* Handle an incoming request.*

*\**

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

*\** ***@param*** *\Closure $next*

*\** ***@param*** *string|null $guard*

*\** ***@return*** *mixed*

*\*/*

public function handle($request, Closure $next, $guard = null)

{

if (Auth::*guard*($guard)->check()) {

return redirect(RouteServiceProvider::*HOME*);

}

return $next($request);

}

}

Файл TrimStrings.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\TrimStrings as Middleware;

class TrimStrings extends *Middleware*

{

*/\*\**

*\* The names of the attributes that should not be trimmed.*

*\**

*\** ***@var*** *array*

*\*/*

protected $except = [

'password',

'password\_confirmation',

];

}

Файл TrustProxies.php

<?php

namespace App\Http\Middleware;

use Fideloper\Proxy\TrustProxies as Middleware;

use Illuminate\Http\Request;

class TrustProxies extends *Middleware*

{

*/\*\**

*\* The trusted proxies for this application.*

*\**

*\** ***@var*** *array|string*

*\*/*

protected $proxies;

*/\*\**

*\* The headers that should be used to detect proxies.*

*\**

*\** ***@var*** *int*

*\*/*

protected $headers = Request::*HEADER\_X\_FORWARDED\_ALL*;

}

Файл Kernel.php

<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends *HttpKernel*

{

*/\*\**

*\* The application's global HTTP middleware stack.*

*\**

*\* These middleware are run during every request to your application.*

*\**

*\** ***@var*** *array*

*\*/*

protected $middleware = [

\App\Http\Middleware\TrustProxies::class,

\Fruitcake\Cors\HandleCors::class,

\App\Http\Middleware\CheckForMaintenanceMode::class,

\Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,

\App\Http\Middleware\TrimStrings::class,

\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,

];

*/\*\**

*\* The application's route middleware groups.*

*\**

*\** ***@var*** *array*

*\*/*

protected $middlewareGroups = [

'web' => [

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

// \Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

'api' => [

'throttle:60,1',

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

];

*/\*\**

*\* The application's route middleware.*

*\**

*\* These middleware may be assigned to groups or used individually.*

*\**

*\** ***@var*** *array*

*\*/*

protected $routeMiddleware = [

'auth' => \App\Http\Middleware\Authenticate::class,

'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,

'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,

'can' => \Illuminate\Auth\Middleware\Authorize::class,

'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,

'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,

'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,

'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,

];

}

Файл RouteServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends *ServiceProvider*

{

*/\*\**

*\* This namespace is applied to your controller routes.*

*\**

*\* In addition, it is set as the URL generator's root namespace.*

*\**

*\** ***@var*** *string*

*\*/*

protected $namespace = 'App\Http\Controllers';

*/\*\**

*\* The path to the "home" route for your application.*

*\**

*\** ***@var*** *string*

*\*/*

public const *HOME* = '/home';

*/\*\**

*\* Define your route model bindings, pattern filters, etc.*

*\**

*\** ***@return*** *void*

*\*/*

public function boot()

{

//

parent::*boot*();

}

*/\*\**

*\* Define the routes for the application.*

*\**

*\** ***@return*** *void*

*\*/*

public function map()

{

$this->mapApiRoutes();

$this->mapWebRoutes();

//

}

*/\*\**

*\* Define the "web" routes for the application.*

*\**

*\* These routes all receive session state, CSRF protection, etc.*

*\**

*\** ***@return*** *void*

*\*/*

protected function mapWebRoutes()

{

Route::middleware('web')

->namespace($this->namespace)

->group(base\_path('routes/web.php'));

}

*/\*\**

*\* Define the "api" routes for the application.*

*\**

*\* These routes are typically stateless.*

*\**

*\** ***@return*** *void*

*\*/*

protected function mapApiRoutes()

{

Route::prefix('api')

->middleware('api')

->namespace($this->namespace)

->group(base\_path('routes/api.php'));

}

protected function authenticated(Request $request, $user)

{

if($user->isAdmin()) {

return view('admin');

}

return view('index');

}

}

Файл AuthServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;

use Illuminate\Support\Facades\Gate;

class AuthServiceProvider extends *ServiceProvider*

{

*/\*\**

*\* The policy mappings for the application.*

*\**

*\** ***@var*** *array*

*\*/*

protected $policies = [

// 'App\Model' => 'App\Policies\ModelPolicy',

];

*/\*\**

*\* Register any authentication / authorization services.*

*\**

*\** ***@return*** *void*

*\*/*

public function boot()

{

$this->registerPolicies();

Gate::*define*('admin-enter', function ($user) {

return $user->isAdmin();

});

}

}

Файл FeedBack.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class FeedBack extends Model

{

public $timestamps = false;

protected $fillable=['user\_id','data'];

public function user(){

return $this->belongsTo('App\User');

}

}

Файл News.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class News extends Model

{

protected $table='News';

protected $fillable=['news','news\_short','author','header','image','created\_at'];

}

Файл Role.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Role extends Model

{

public $timestamps = false;

protected $fillable=['role'];

public function user(){

return $this->belongsToMany('App\User');

}

}

Файл Subject.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

use App\User;

class Subject extends Model

{

public $timestamps =true;

protected $fillable=['user\_id','long','lat','images','description','price'];

public function user(){

return $this->belongsTo('App\User');

}

public function users(){

return $this->belongsToMany('App\User');

}

}

Файл User.php

<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use App\FeedBack;

use App\Subject;

class User extends *Authenticatable*

{

use Notifiable;

public function isAdmin(){

if($this->roles[0]->role==='admin'){

return true;

}else{

return false;

}

}

*/\*\**

*\* The attributes that are mass assignable.*

*\**

*\** ***@var*** *array*

*\*/*

protected $fillable = [

'name', 'email', 'password','phone','image',

];

*/\*\**

*\* The attributes that should be hidden for arrays.*

*\**

*\** ***@var*** *array*

*\*/*

protected $hidden = [

'password', 'remember\_token',

];

*/\*\**

*\* The attributes that should be cast to native types.*

*\**

*\** ***@var*** *array*

*\*/*

protected $casts = [

'email\_verified\_at' => 'datetime',

];

public function roles(){

return $this->belongsToMany('App\Role');

}

public function feedBacks(){

return $this->hasMany('App\FeedBack');

}

public function subject(){

return $this->hasMany('App\Subject');

}

public function subjects(){

return $this->belongsToMany('App\Subject');

}

}

Файл Handler.php

<?php

namespace App\Exceptions;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

use Throwable;

class Handler extends *ExceptionHandler*

{

*/\*\**

*\* A list of the exception types that are not reported.*

*\**

*\** ***@var*** *array*

*\*/*

protected $dontReport = [

//

];

*/\*\**

*\* A list of the inputs that are never flashed for validation exceptions.*

*\**

*\** ***@var*** *array*

*\*/*

protected $dontFlash = [

'password',

'password\_confirmation',

];

*/\*\**

*\* Report or log an exception.*

*\**

*\** ***@param*** *\Throwable $exception*

*\** ***@return*** *void*

*\**

*\** ***@throws*** *\Exception*

*\*/*

public function report(Throwable $exception)

{

parent::*report*($exception);

}

*/\*\**

*\* Render an exception into an HTTP response.*

*\**

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

*\** ***@param*** *\Throwable $exception*

*\** ***@return*** *\Symfony\Component\HttpFoundation\Response*

*\**

*\** ***@throws*** *\Throwable*

*\*/*

public function render($request, Throwable $exception)

{

return parent::*render*($request, $exception);

}

}

Файл DatabaseSeeder.php

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

*/\*\**

*\* Seed the application's database.*

*\**

*\** ***@return*** *void*

*\*/*

public function run()

{

$this->call(roleSeeder::class);

$this->call(UserSeeder::class);

$this->call(FeedBackSeeder::class);

$this->call(SubjectSeeder::class);

$this->call(NewsSeeder::class);

}

}

Файл FeedBackSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\FeedBack;

use App\User;

use Eastwest\Json\Facades\Json;

class FeedBackSeeder extends Seeder

{

*/\*\**

*\* Run the database seeds.*

*\**

*\** ***@return*** *void*

*\*/*

public function run()

{

{

$feedBack=new FeedBack;

$feedBack1=new FeedBack;

$user=User::*find*(1);

$user2=User::*find*(2);

$data=json\_encode(['img'=>'image/test.jpg','header'=>'test','text'=>'test test test']);

$data1=json\_encode(['img'=>'image/test.jpg','header'=>'first','text'=>'Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

']);

$feedBack->data=$data;

$user->feedBacks()->save($feedBack);

$feedBack1->data=$data1;

$user2->feedBacks()->save($feedBack1);

}

}

}

Файл NewsSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\News;

use Faker\Factory as Faker;

class NewsSeeder extends Seeder

{

*/\*\**

*\* Run the database seeds.*

*\**

*\** ***@return*** *void*

*\*/*

public function run()

{

$faker = *Faker*::*create*('App\News');

for($i=0;$i<10;$i++){

DB::*table*('News')->insert([

'news'=>$faker->text(200),

'header'=>$faker->text(30),

'image'=>'/image/test.jpg',

'news\_short'=>$faker->text(50),

'author'=>$faker->name,

'created\_at'=>$faker->dateTimeBetween('-30 days','+30 days'),

]);

}

}

}

Файл roleSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\User;

use App\Role;

class roleSeeder extends Seeder

{

*/\*\**

*\* Run the database seeds.*

*\**

*\** ***@return*** *void*

*\*/*

public function run()

{

Role::*create*(['role'=>'user']);

Role::*create*(['role'=>'admin']);

}

}

Файл SubjectSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\Subject;

use App\User;

use Faker\Factory as Faker;

class SubjectSeeder extends Seeder

{

*/\*\**

*\* Run the database seeds.*

*\**

*\** ***@return*** *void*

*\*/*

public function run()

{

$images=['k1'=>'/image/test.jpg','k2'=>'/image/test.jpg','k3'=>'/image/test.jpg','k4'=>'/image/test.jpg'];

for($i=0;$i<10;$i++){

$subject=new Subject;

$user=User::*find*(($i%2)+1);

$subject->lat=44.5419+0.1\*$i;

$subject->long=33.5319+0.1\*$i;

$subject->price=20000;

if($i<7){

$subject->checked=true;

}

$subject->images=json\_encode($images);

$subject->description="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.";

$subject->user()->associate( $user);

$subject->save();

$subject->users()->attach( $user);

}

}

}

Файл UserSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\User;

class UserSeeder extends Seeder

{

*/\*\**

*\* Run the database seeds.*

*\**

*\** ***@return*** *void*

*\*/*

public function run()

{

$user=User::*create*([

'name'=>'Udjin',

'phone'=>'+79787361924',

'email'=>'luthenkoev@gmail.com',

'password'=>bcrypt('joncon21')

]);

$user1=User::*create*([

'name'=>'User',

'phone'=>'+79787361954',

'email'=>'vvv958@gmail.com',

'password'=>bcrypt('joncon21')

]);

$user->roles()->attach(2);

$user->save();

$user1->roles()->attach(1);

$user1->save();

}

}

Файл index.php

<?php

*/\*\**

*\* Laravel - A PHP Framework For Web Artisans*

*\**

*\** ***@package*** *Laravel*

*\** ***@author*** *Taylor Otwell <taylor@laravel.com>*

*\*/*

define('LARAVEL\_START', microtime(true));

/\*

|--------------------------------------------------------------------------

| Register The Auto Loader

|--------------------------------------------------------------------------

|

| Composer provides a convenient, automatically generated class loader for

| our application. We just need to utilize it! We'll simply require it

| into the script here so that we don't have to worry about manual

| loading any of our classes later on. It feels great to relax.

|

\*/

require *\_\_DIR\_\_*.'/../vendor/autoload.php';

/\*

|--------------------------------------------------------------------------

| Turn On The Lights

|--------------------------------------------------------------------------

|

| We need to illuminate PHP development, so let us turn on the lights.

| This bootstraps the framework and gets it ready for use, then it

| will load up this application so that we can run it and send

| the responses back to the browser and delight our users.

|

\*/

$app = require\_once *\_\_DIR\_\_*.'/../bootstrap/app.php';

/\*

|--------------------------------------------------------------------------

| Run The Application

|--------------------------------------------------------------------------

|

| Once we have the application, we can handle the incoming request

| through the kernel, and send the associated response back to

| the client's browser allowing them to enjoy the creative

| and wonderful application we have prepared for them.

|

\*/

$kernel = $app->make(Illuminate\Contracts\Http\Kernel::class);

$response = $kernel->handle(

$request = Illuminate\Http\Request::*capture*()

);

$response->send();

$kernel->terminate($request, $response);