Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский университет «Высшая школа экономики»

Образовательная программа «Прикладная математика»

бакалавр

Отчет
по проектной работе
Прогулки по Уханю

Выполнили студенты групп БПМ 194 и 195

Словеснов Максим Олегович
Сайфуллина Камилла Айдаровна

Руководитель проекта:

Доцент

Зотов Леонид Валентинович

_____ _____
(оценка) *(подпись)*

_____
(дата)

Москва 2020г.

Содержание

1. Введение

Веб-разработка особенно активно развивается с начала XXI века. За три десятилетия *web-development* из создания любительских сайтов-одностраничек перешёл в одну из самых денежнооборотных и влиятельных IT-индустрий. В рамках проекта "Прогулка по городу Ухань"необходимо было создать веб-сайт, посвященный достопримечательностям данного города, и выгрузить его в сеть Интернет. Но наша группа, уже будучи знакома с основами веб-разработки, решила пойти дальше и создать "генератор"городов, который предоставит удобный интерфейс административной панели для легкого добавления нового города и простого редактирования всего контента веб-сайта.

2. Вспомогательная литература

В процессе разработки проекта использовалось множество современных методов/Языков программирования и вспомогательных систем, которые используются не только большинством веб-разработчиков, но и передовыми IT-компаниями. При создании работы нельзя было обойтись без справочника по верстке CSS3/HTML5, документации препроцессора SCSS, а также документации Bootstrap4. Помимо этого, стоит отметить полезность ресурса, к которому мы периодически прибегали.

3. Технологии и методы

IT-индустрия, как было отмечено в разделе 1. , развивается неимоверными темпами. Достаточно трудно назвать стек технологий, который лучше всего справляется со всеми задачами веб-программирования. Однако можно довериться тем технологиям, которые хорошо себя зарекомендовали и заняли высокие позиции в рейтинге доверия программистов и разработчиков. Ниже приведён набор инструментов и языков программирования, применяемый в нашем проекте:

- HTML5 - язык разметки веб-сайтов
- CSS3 - язык таблиц стилей
- Gulp - таск-менеджер
- Bootstrap4 - набор инструментов с HTML и CSS шаблонами
- PHP(версия 7.1.0 и выше) - серверный язык программирования
- JQuery - фреймворк для языка программирования JavaScript
- SCSS - препроцессор стилей
- MySQL - система управления базами данных
- SQL - декларативный язык программирования, предназначенный для работы с базами данных

- PhpMyAdmin - веб-приложения для удобной работы с MySQL
- Apache2 - веб-сервер
- Composer - пакетный менеджер для PHP
- NPM - пакетный менеджер для JS и Node.js
- GIT - система контроля версий

3.1. Методы

Помимо инструментов и языков программирования, при создании проекта были применены специальные методы

3.1.1. MVC

MVC(Model-View-Controller) - схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер - таким образом, что модификация каждого компонента может осуществляться независимо. По данной схеме построен весь проект.

3.1.2. PSR-4

- стандарт написания кода.

4. Этапы создания проекта

В начале создания любого проекта первым дело необходимо построить грамотную архитектуру и настроить окружение. После этого можно переходить к самой разработке. Для успешного выполнения этих пунктов необходимо было выполнить несколько последовательных задач:

1. Настройка окружения
    - Построение иерархической структуры папок
    - Настройка локального сервера для проекта
    - Настройка **GIT**
    - Подключение **composer-** и **npm-** пакетных менеджеров
    - Подключение дополнительных компонентов и инструментов
2. Раработка проекта
    - Настройка MVC-модели
    - Подключение и настройка связей с БД
    - Настройка ЧПУ
    - Верстка и стилизация

4.1. Настройка окружения

4.1.1. Иерархия проекта

Первое, с чего необходимо начинать создание проекта - это построение папочной структуры. Именно он определяет, каким образом будет функционировать проект. При использовании различных **фреймворков** (программных платформ) обычно папочная структура задаются самостоятельно, поскольку она уже "вшита"в них. В нашем же случае, мы создаём папочную структуру самостоятельно. Она имеет следующее чёткое разделение:

- папка */application*, отвечающая за сам PHP-код
- папка */assets*, в которой хранятся "неподготовленные"стили и изображения
- папка */build*, в который лежат готовые стили
- файл *index.php*, на который ссылаются все остальные файлы
- папка */images*, в которой лежат фотографии для базы данных
- файл *.htaccess*, в котором прописаны настройки сервера для данного проекта
- файл *.gitignore*, в котором прописываются файлы и папки, которые не должны попасть в репозиторий

4.1.2. Локальный сервер

Для отслеживания php-кода и работы с базами данных мы использовали сервер Apache2 и PhpMyAdmin. Они отличаются своей простотой в использовании и удобством при создании небольших проектов.

4.1.3. Настройка GIT

Правильная настройка окружения начинается с системы контроля версий. Для успешного использования системы GIT были выполнены следующие пункты:

1. Создание репозитория под названием **walking** и присвоение ему свойства **public**
2. Выбор названий **коммитов** и грамотное разделение **веток** для общего удобства работы с системой

4.1.4. Подключение пакетных менеджеров

Установка пакетных менеджеров также проходит в несколько этапов:

1. Инициализация пакетных менеджеров
2. Добавление необходимых для разработки пакетов
3. Установка зависимостей с этими пакетами
4. Добавление создавшихся папок */composer* и */node_modules* в файл *.gitignore*

4.1.5. Дополнительные инструменты

В качестве вспомогательного дополнения мы выбрали JQuery-фреймворк для использования технологии AJAX. Остальные инструменты мы подключили с использованием ранее упомянутые **composer** и **npm** . На выходе мы получили папочную структуру, изображённую на рис. 1.

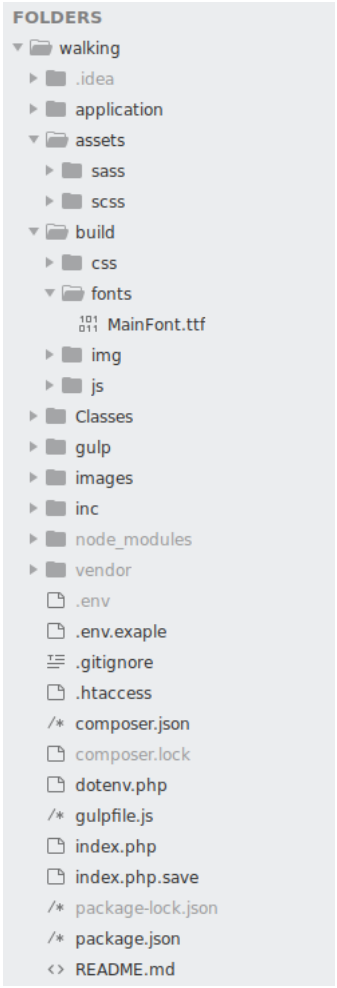

Рис. 1. Иерархия папок

4.2. Разработка проекта

4.3. MVC-модель

Данная модель была организована с использованием **классов** и **наследований классов**, которые являются составляющей ООП. Данная организация напоминает стандарт **PSR-4**, однако он далёк от идеального. Ниже представлена структура модели:

- в папке */application*, находятся все разбиение на роли

- папка */application/core* содержит базовые классы, от которых все остальные наследуются(в зависимости от их роли)
- папка */application/views* содержит файлы, отвечающие за отображения
- папка */application/models* содержит файлы, отвечающие за модели
- папка */application/controllers* содержит файлы, отвечающие за управление

Реализацию MVC-модели вы можете увидеть ни следующем рисунке:

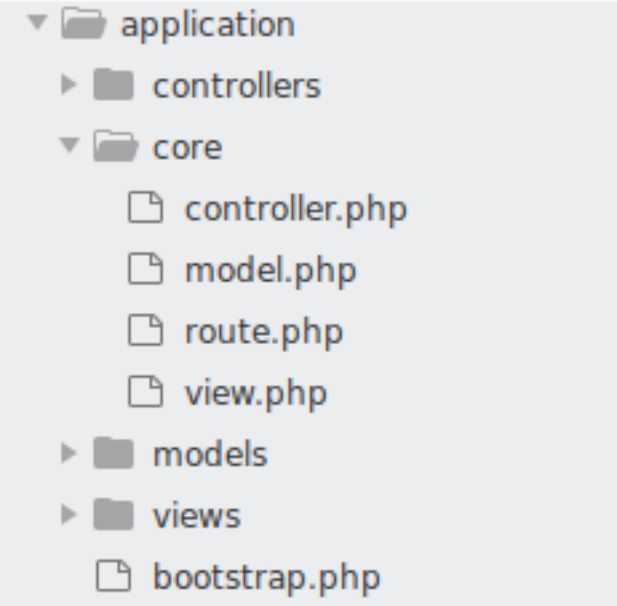

Рис. 2. Структура MVC-модели

4.4. База данных

Для удобной и качественной работы с БД используются глобальные переменные, настраиваемые с помощью пакета **dotenv**. Они помогают облегчить жизнь разработчикам при работе с БД. Помимо этого, соединение с базой данных происходит с использованием метода **PDO** и оптимизацией функций для простых запросов. Всё это помогает сделать код более читабельных, а систему - более защищенной от **SQL-инъекций** и атак подобного рода. Настройка базы данных хранится по адресу: */application/models/ModelBase.php* и является составляющим класса **ModelBase**. На рис. 3 можно увидеть код соединения с базой данных.

4.5. ЧПУ

ЧПУ - или Человекопонятный URL, настраивается всего одним файлом, который находится по адресу: */application/core/route.php*

```php
<?php
require_once ($_SERVER['DOCUMENT_ROOT'].'/inc/libs.php');
class ModelBase {
    public $db;
    public $name;
    public $password;
    public $host;
    public $DH;
    public $table;
    public $dataResult;


    function __construct($classname, $select=false) {
            $this->db=getenv('DB_NAME');
            $this->host=getenv('DB_HOST');
            $this->name=getenv('DB_USER');
            $this->password=getenv('DB_PASSWORD');

            try {
             $this->DH = new PDO("mysql:host=$this->host;dbname=$this->db", $this->name, $this->password);
             $this->DH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );


        // имя таблицы
        $table=strtolower($classname);
        $this->table = $table;
        // обработка запроса, если нужно
        $sql = $this->_getSelect($select);
        $test="SELECT * FROM $this->table" . $sql;
        if($sql) $this->_getResult("SELECT * FROM $this->table" . $sql);
             }  catch(PDOException $e) {
                echo "Хьюстон, у нас проблемы. \n";
                echo $e->getMessage();
            }
        }
```

Рис. 3. Файл базы данных

4.6. Стилизация

Стили проекта прописываются в папке */assets/scss* которые потом с помощью команды *gulp deploy* или *gulp sass* преобразуются в *css-формат* и минифицируются в единый файл *styles.min.css* в папке */build/css*.

5. Установка проекта и его использование

5.1. Установка проекта

Под словом *Установка* можно подразумевать две вещи: настройка проекта для его дальнейшей поддержки/разработки и загрузка его в сеть Интернет. Для удобного *"разворачивания"* были использованы ранее упомянутые инструменты: GIT, Composer и NPM. Установки проекта для двух этих целей ничем не различаются, поэтому обобщим правила для них:

1. Клонируем репозиторий с помощью команды в консоли, предварительно настроив **ssh-ключ**: *git@gitlab.com:slovenberg/walking.git* .

2. Устанавливаем **composer** и **npm** с помощью консольных команд *npm install && composer install*

3. Собираем стили с помощью команды *gulp deploy*

4. Настраиваем подключение к базе данных в файле *.env*

5.1.1. Дальнейшая разработка

Для того, чтобы продолжить работу с данным проектом, необходимо установить проект из git-репозитория

5.2. Использование проекта

Данный проект имеет возможность редактирование контента из собственноручно созданной **админ-панели**. Поэтому пользоваться сайтом можно как со стороных *обычного пользователя*, так и со стороны *администратора*.

5.2.1. Пользователь

Поскольку у нас настроен механизм ЧПУ, мы можем использовать красивые и понятные адреса.
Рассмотрим главную страничку, которая располагается по адресу *site.ru* или *site.ru/towns*.
На данной страничке мы видем список доступных для просмотра городов: Перейдя

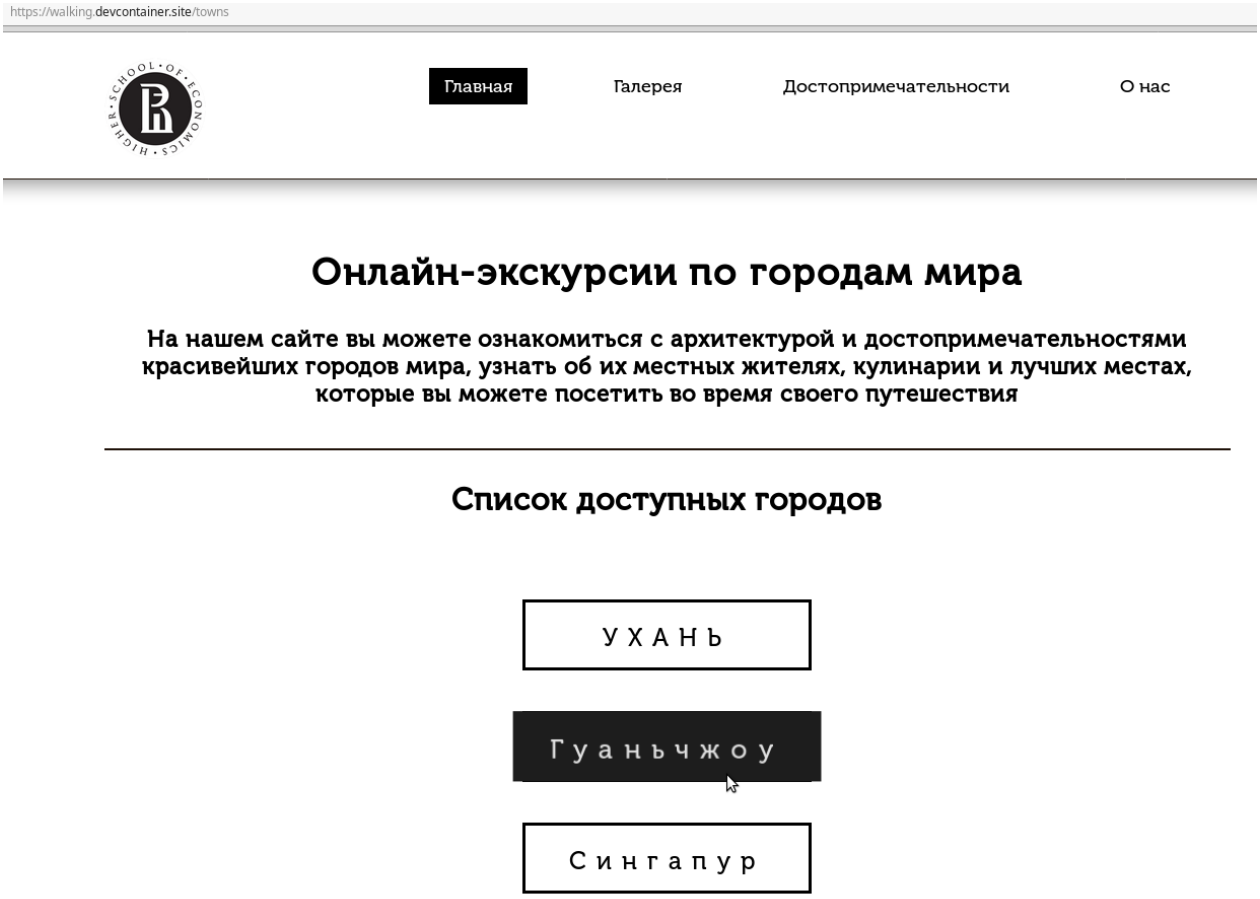

Рис. 4. Главная страница

по нажатиюю кнопки в один из городов, мы можем выбрать один из нескольких разделов для просмотра.

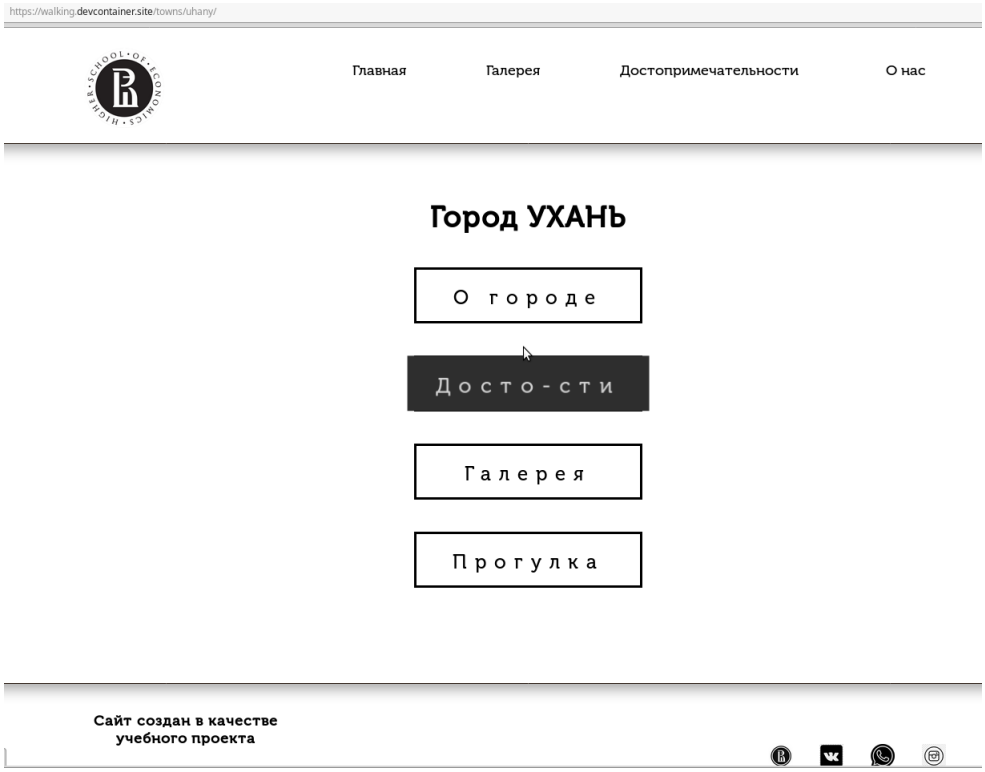

Рис. 5. Разделы города

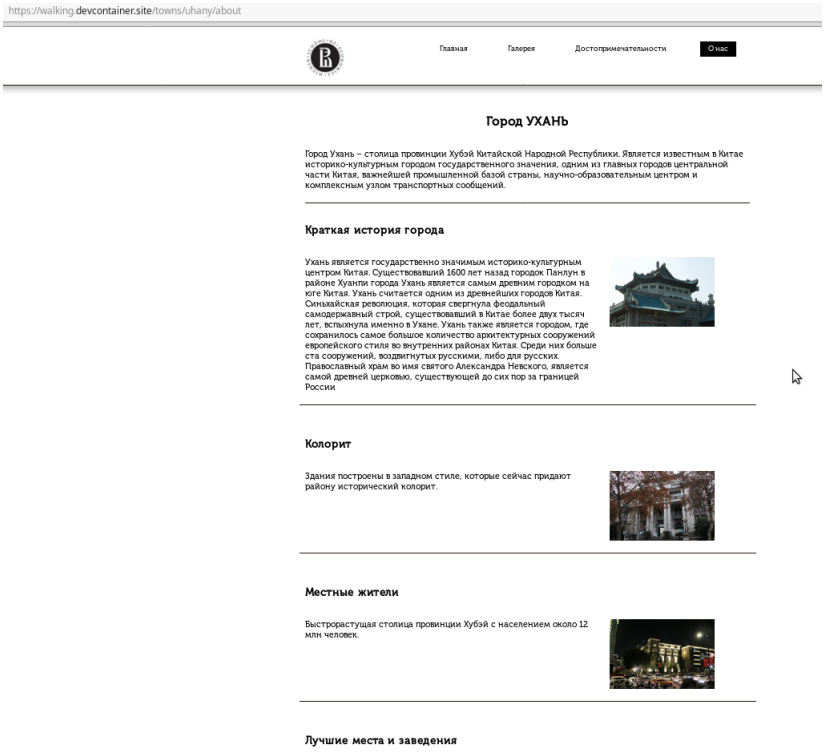

Рис. 6. О городе

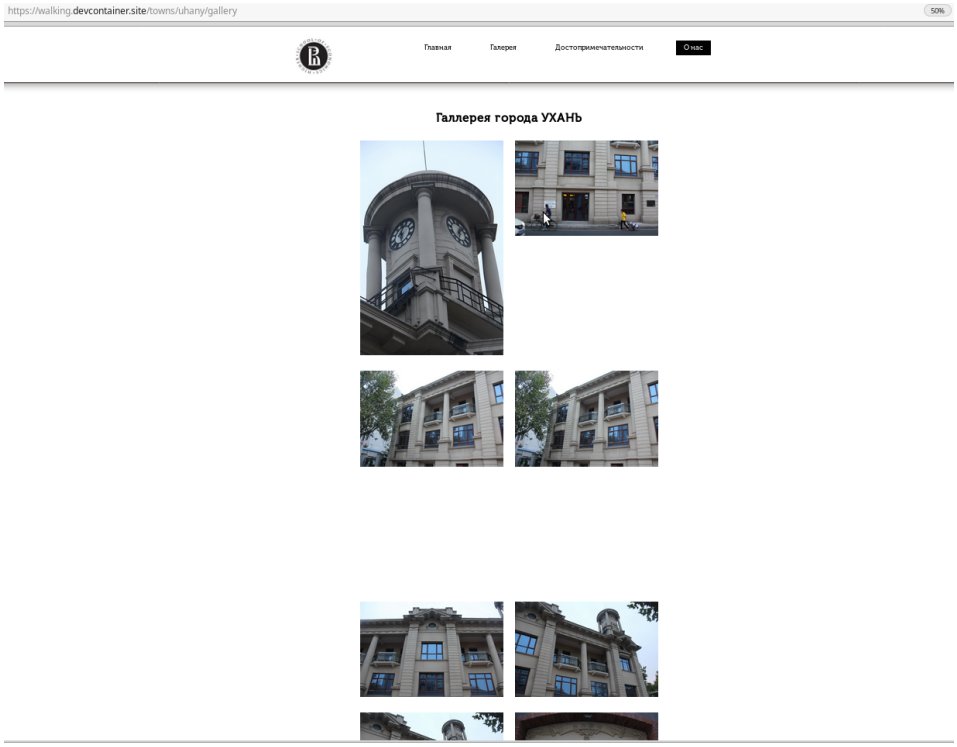

Рис. 7. Галлерея

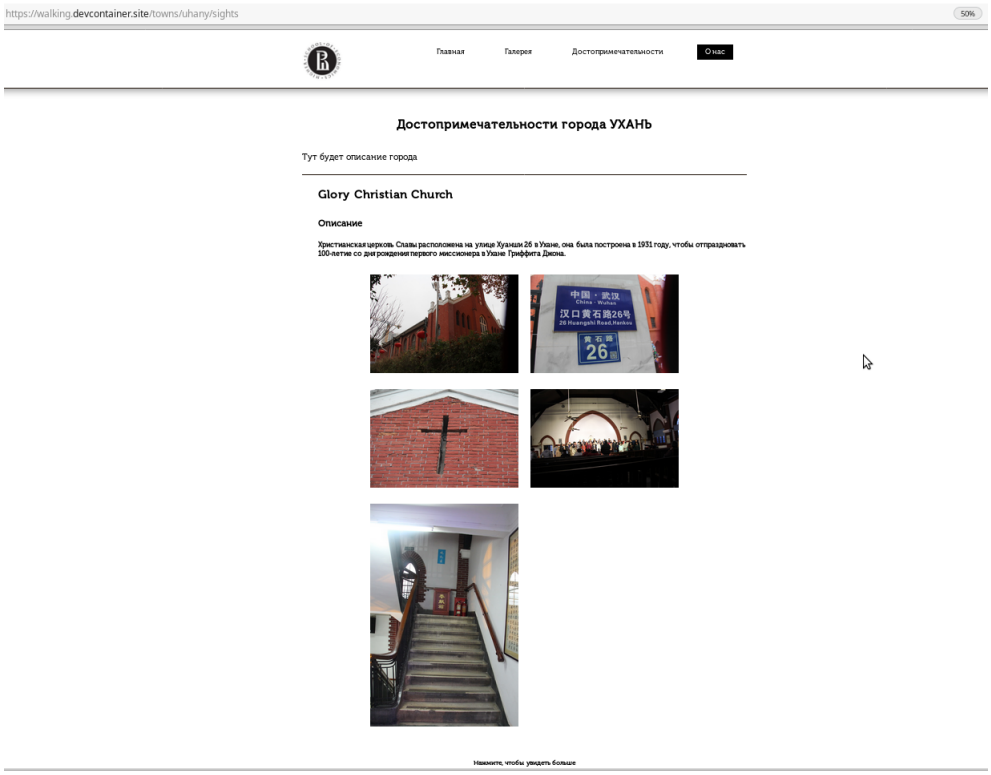

Рис. 8. Достопримечательности

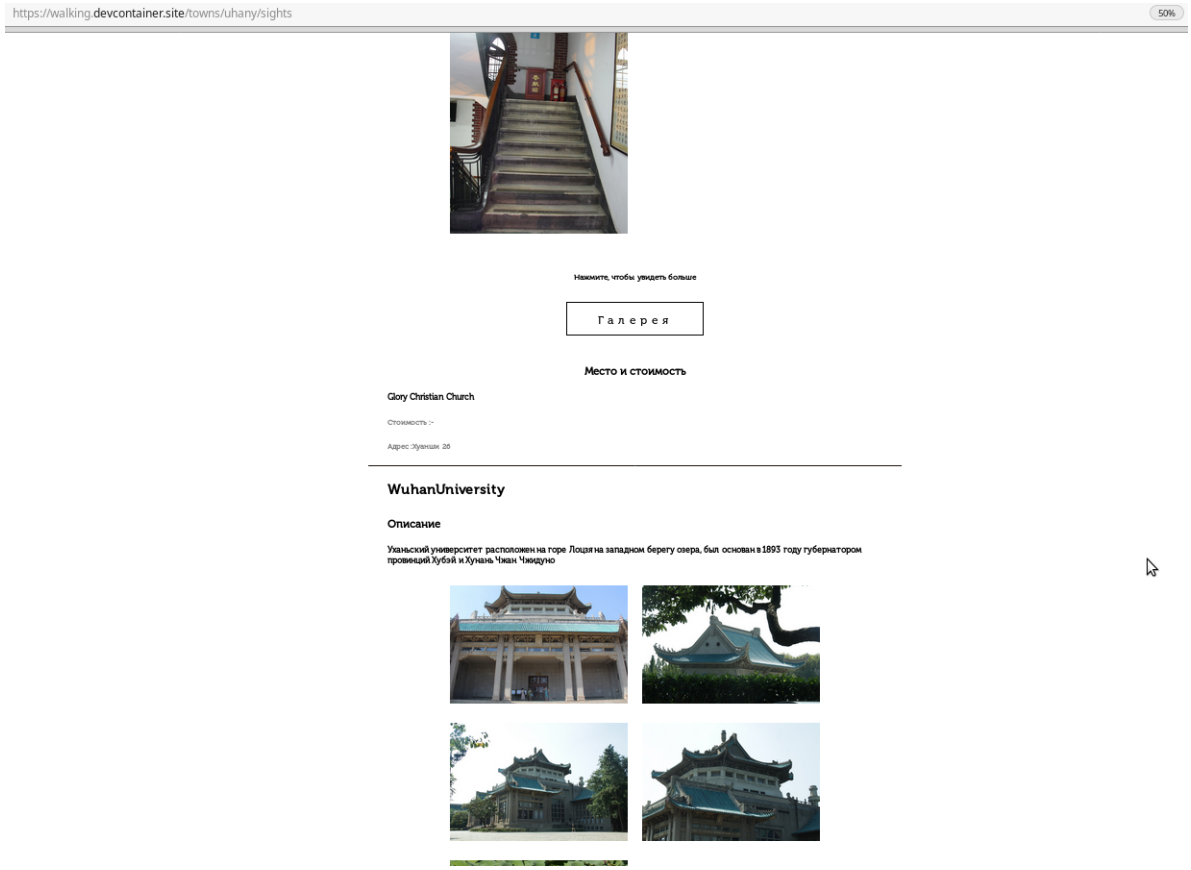

Рис. 9. Достопримечательности

Стоит обрать внимание на адрес сайта на всех этих фотографиях: его структура очень чёткая и понятная. например, имея адрес сайта *site.ru/towns/moskva/about* , мы сразу можем понять, о чем примерно будет страница.

5.2.2. Администратор

Сайт выглядит весьма посредственно и обыденно, однако стоит посмотреть на администраторскую часть.
Для того, чтобы перейти в панель Адмнистратора, стоит написать следующий адрес: *site.ru/admin*. Перед нами появилась таблица со всеми городами.

После этого, дописав в адресной строке название города из таблицы через слеш, мы перейдём в раздел редактирования города.

Переходя аналогичным образом по разделам, как и в пользовательском режиме (только с начальным словом */admin* вместо */towns*), мы можем редактировать, удалять и добавлять контент.

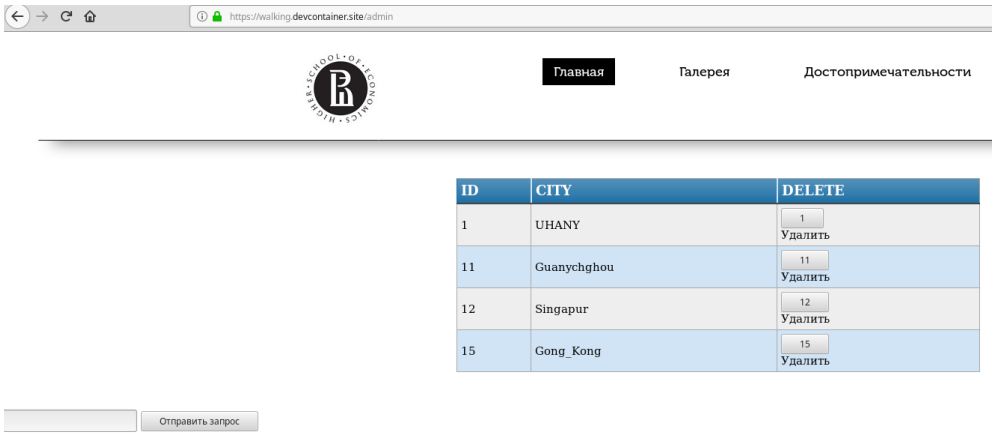

Рис. 10. Админ-панель

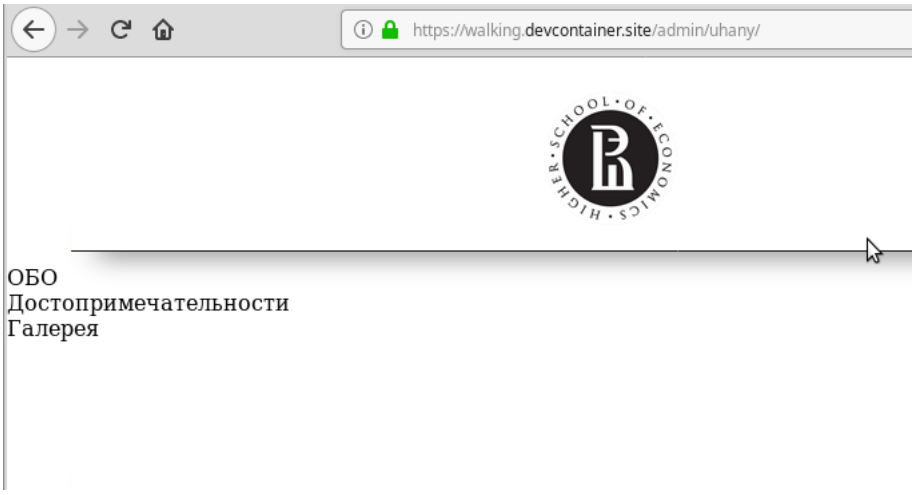

Рис. 11. Выбор пункта

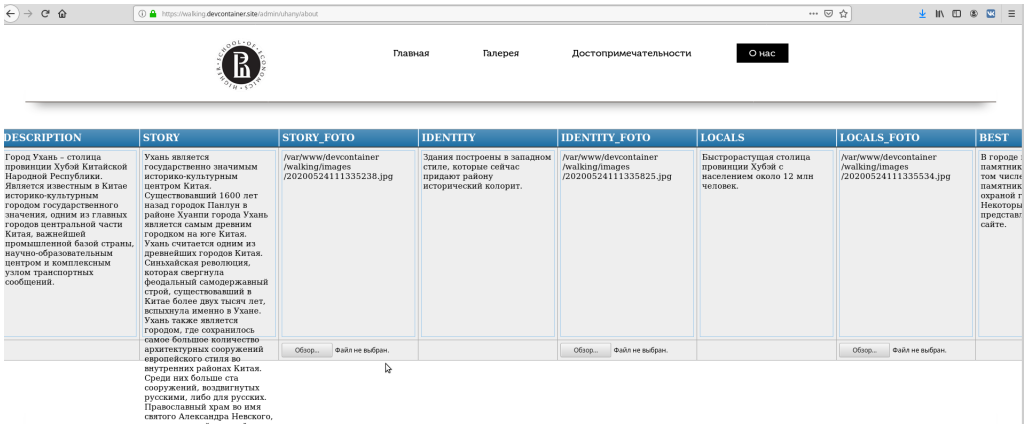

Рис. 12. Редактор графы о городе

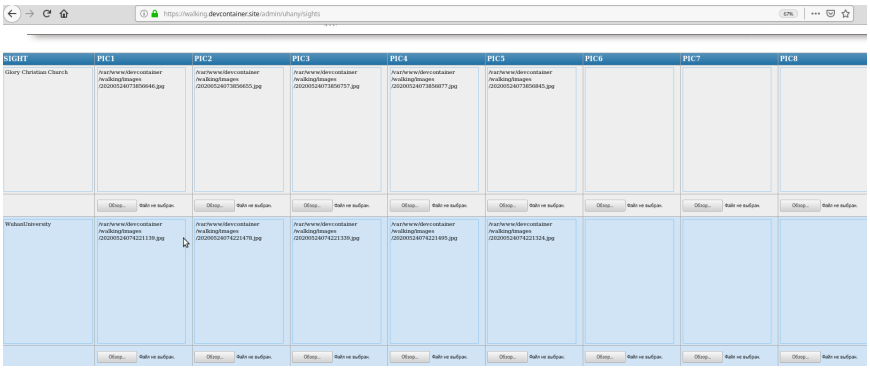

Рис. 13. Редактор достопримечательностей

6. Анализ результата

В результате проведенной работы разработан сайт, отвечающий основным требованиям как администратора, так и пользователя.
Благодаря разработанной иерархической системе построения сайта, администратор сайта:

1. имеет возможность оперативно получать доступ к той или иной подсистеме и при необходимости вносить в нее изменения,
2. система устроена таким образом, что при передаче прав новому администратору, позволяет в сжатые сроки разобраться в структуре построения сайта и принять администрование в полном объеме,
3. вне зависимости от уровня знаний администратора, в том числе при отсутствии специализированных знаний в области IT-технологий, вносить/удалять/корректировать все фронотовые разделы сайта, в том числе, текстовую, графическую части, а также часть, содержащую фотоматериалы,
4. система устроена таким образом, что на сайт можно добавлять бесконечное число городов, с описательным и фото- разделами.

Со строны пользователя:

1. сайт отвечает условиям простоты и понятности, а именно имеют место:
   (a) простота в использовании разделов меню,
   (b) возможность перехода в раздел данных о любом из городов без необходимости возврата на предыдущую страницу в разделе данного конкретного города,
   (c) нестандартный дизайн в выдержанных тонах с воспринимаемыми, не раздражающими шрифтами,
   (d) - современные решения, в частности касающиеся анимации кнопок методами псевдоклассов и CSS3,

2. возможность получения всей необходимой информации о достопримечательностях интересующего города,

3. сохранение фронтального вида, основных функций сайта без каких-либо сбоев при открытии его через любой сервер.

Таким образом, на выходе получен удобный и функциональный сайт-конструктор для администратора, позволяющий благодаря профессиональной разработке и настройке передать возможность использования в том числе лицу, не обладающему специализированными знаниями, и а также удобный отвечающим всем современным требованиям со стороны пользователя.

7. Заключение

Разработанный сайт "Прогулки по Уханю"в полном объеме отвечает требованиям технического задания. Сверх данного задания разработанный проект позволяет в максимально сжатые сроки редактировать сайт (в том числе лицами, неимеющими специального образования), добавляя в него данные о любом новом городе.
На основании изложенного, разработчиками сайта не только исполнены основные требования научного руководителя, но и добавлены и реализованы доп. функции на сайте сверх поставленных задач.

Список литературы / References

[1] Ahmedshin F. R., “Infrastructure DevOps Development”, *National Research University*, 2020.