ОГЛАВЛЕНИЕ

[ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ 4](#_Toc451527633)

[1. ВВЕДЕНИЕ 5](#_Toc451527634)

[1.1 Актуальность 5](#_Toc451527635)

[1.2 Цель 6](#_Toc451527636)

[1.3 Постановка задачи 6](#_Toc451527637)

[2. АНАЛИЗ АДМИНИСТРАТИВНОЙ ПАНЕЛИ САЙТА 8](#_Toc451527638)

[2.1 Анализ существующих аналогов 8](#_Toc451527639)

[2.1.1 WordPress 8](#_Toc451527640)

[2.1.2 Joomla 9](#_Toc451527641)

[2.1.3 Drupal 10](#_Toc451527642)

[2.2 Календарный план по разработке выпускной работы бакалавра 12](#_Toc451527643)

[2.3 Выводы по разделу 2 13](#_Toc451527644)

[3. ПРОЕКТИРОВАНИЕ АДМИНИСТРАТИВНОЙ ПАНЕЛИ САЙТА 14](#_Toc451527645)

[3.1 Выбор архитектуры ПО 14](#_Toc451527646)

[3.1.1 Архитектура клиентской части 14](#_Toc451527647)

[3.1.2 Архитектура серверной части 14](#_Toc451527648)

[3.2 Разработка wireframe’ов проекта 15](#_Toc451527649)

[3.2.1 Раздел «Категориии» (главная страница) 15](#_Toc451527650)

[3.2.2 Раздел «Избранные» 16](#_Toc451527651)

[3.2.3 Раздел «Контакты» 17](#_Toc451527652)

[3.2.4 Раздел «Фототуры» 17](#_Toc451527653)

[3.2.5 Раздел «Заявки» 19](#_Toc451527654)

[3.2.6 Раздел «Письма» 21](#_Toc451527655)

[3.2.7 Раздел «Смена пароля» 21](#_Toc451527656)

[3.2.8 Раздел «Добавить нового пользователя» 22](#_Toc451527657)

[3.2.9 Страница авторизации 23](#_Toc451527658)

[3.3 Разработка инфраструктуры проекта 23](#_Toc451527659)

[3.3.1 Код Gulp скрипта 24](#_Toc451527660)

[3.3 Проектирование базы данных 27](#_Toc451527661)

[3.3.1 Проектирование логической модели базы данных 28](#_Toc451527662)

[3.3.2 Связи между сущностями 29](#_Toc451527663)

[3.3.3 Представление сущностей в виде таблиц 30](#_Toc451527664)

[3.3.4 Разработка скриптов для реализации БД административной панели 35](#_Toc451527665)

[4. ЭКОНОМИЧЕСКАЯ ЧАСТЬ 42](#_Toc451527666)

[4.1 Цель экономической части 42](#_Toc451527667)

[4.2 Описание проекта 42](#_Toc451527668)

[4.3 Разработка перечня работ по проектированию административной панели 42](#_Toc451527669)

[4.4 Расчет затрат и цены разработки административной панели 45](#_Toc451527670)

[4.5 Выводы 48](#_Toc451527671)

[СПИСОК ЛИТЕРАТУРЫ 49](#_Toc451527672)

[ПРИЛОЖЕНИЕ А: Техническое задание 50](#_Toc451527673)

# ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

Node.js – платформа, основанная на JavaScript движке V8, и предлагающая асинхронное API для работы с сетью и диском. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера.

Backbone — JavaScript-библиотека, предназначена для разработки веб-приложений с поддержкой RESTful JSON интерфейса.

SQL – Structured Query Language – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

БД – база данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

API — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

WYSIWYG — является аббревиатурой от англ. What You See Is What You Get, «что видишь, то и получишь», свойство прикладных программ или веб-интерфейсов, в которых содержание отображается в процессе редактирования и выглядит максимально близко похожим на конечную продукцию, которая может быть печатным документом, веб-страницей или презентацией.

Wireframe — это низко детализированное представление дизайна, который показывает информационную структуру сайта и описывает взаимодействие пользователя с интерфейсом.

Gulp — это инструмент сборки веб-приложения, позволяющий автоматизировать повторяющиеся задачи, такие как сборка и минификация CSS- и JS-файлов, запуск тестов, перезагрузка браузера и т.д. Тем самым Gulp ускоряет и оптимизирует процесс веб-разработки.

# ВВЕДЕНИЕ

## Актуальность

До появления административной панели, web-сайт представлял собой небольшое количество статических страниц, и для того, чтобы изменить что-то на сайте, приходилось открывать код и менять всё прямо в нём. Это было достаточно сложно сделать, не обладая специальными знаниями. Владельцам сайтов приходилось часто привлекать к работе посторонних людей – программистов. Таким образом, нарушалась политика безопасности компании, а сам процесс изменения информации занимал слишком много времени. Со временем скорость обновления ресурса стала приобретать всё большее значение, страницы сайта становились всё более и более графически загруженными, на сайтах стали появляться анимации, аудио, flash-объекты, всё больше внимания уделялось смысловому наполнению сайта. На сегодняшний день практически каждый веб-сайт имеет как минимум несколько различных разделов и подразделов. Было очевидно, что без инструмента, который упрощал бы работу с сайтом, не обойтись. Решением этих проблем и стала административная панель.

Веб-сайт может работать и вообще без административной панели. Это допустимо, если он состоит из 1-й страницы. Однако если сайт будет расширяться и количество страниц перевалит, например, за 30, то могут возникнуть сложности. Если нужно будет изменить какую-либо информацию, содержащуюся на каждой или на многих страницах сайта  (например, номер телефона, или адрес какой-то ссылки), то вам придётся заходить отдельно на каждую страницу и вручную в коде заменять эту информацию.

В то же время, владельцу сайта, с административной панелью, достаточно зайти в административную панель сайта, и, используя красиво оформленные и вполне понятные любому человеку поля, легко управлять новостными лентами, каталогами товаров, модерировать форумы, добавлять видео, картинки, визуально изменять структуру сайта, добавляя новые поля и страницы, и производить множество других операций, которые без административной панели были бы не просто трудоёмкими, но часто даже невозможными. Пользователю не обязательно обладать навыками программирования или знаниями языка HTML, чтобы, например, опубликовать на своем сайте новость, статью или добавить изображение.

Принцип работы административной панели основан на разделении контента (содержания) и дизайна (оформления) сайта. Обычно дизайн сайта меняется редко, тогда как изменения контента могут происходить не только каждый день, но и даже каждый час.

Информация хранится в базе данных, например, в MySQL и вызывается из нее при загрузке страниц сайта.

Работа в административной панели не требует установки дополнительного ПО, поскольку сама панель находится на сервере, а доступ к ней осуществляется через обычный интернет-браузер. Предлагаемый электронный ресурс может редактироваться через веб-браузер независимо от места нахождения человека.

Современные административные панели, помимо редактирования материалов позволяют проводить операции над зарегистрированными аккаунтами, например ― удалять, заблокировать или создать новую учетную запись. Также есть возможность разграничения прав доступа учетных записей для для различных операций над разделами сайта.

Через административную панель можно проводить модерацию материалов, заявок от пользователей сайта.

При использовании административной панели, пользователи получает возможность организовать рассылку писем пользователям, например, знакомить их с акциями от компании.

Итак, почему на сегодняшний день актуально использовать административную панель на своем сайте?

* Экономия денег на поддержке.

Для эффективной работы сайта нужно постоянно менять его содержимое — добавлять информацию, менять цены, изменять контактны, актуализировать данные и много другое. Если на сайте есть административная панель, то можно самостоятельно заниматься его поддержкой, не привлекая людей со стороны.

* Предотвращение ошибок.

Работая с сайтом вручную, можно допустить массу ошибок, связанных с человеческим фактором, например, сделать опечатку в коде или использовать устаревший тег. Используя административную панель, количество таких ошибок сводится к минимуму.

* Управление доступом.

Если над сайтом работают несколько специалистов (по поддержке, продвижению, наполнению контента и так далее) может потребоваться разграничить их доступ. Без административной панели это сделать будет проблематично.

## Цель

Целью дипломной работы является разработка административной панели для сайта.

## Постановка задачи

Для достижения поставленной цели в дипломной работе необходимо решить следующие задачи:

* проанализировать существующие аналоги административных панелей;
* разработать инфраструктуру проекта;
* разработать wireframe’ы административной панели.
* составить требования к административной панели;
* разработать реляционную базу данных;
* разработать серверную часть приложения;
* разработать клиентскую часть приложения.

# АНАЛИЗ АДМИНИСТРАТИВНОЙ ПАНЕЛИ САЙТА

## Анализ существующих аналогов

На сегодняшний день наиболее популярны следующие аналоги амдинистративных панелей:

* WordPress
* Joomla
* Drupal

### WordPress

Система WordPress зарекомендовала себя как инновационная блог-платформа с высокой уровнем удобства использования.  Но развитие функционала системы обеспечило ей высокую популярность также и среди других форматов веб-сайтов. Сегодня индустрия веб-дизайна обеспечена высоким спросом на услуги в области разработки под WordPress. WordPress используется везде, начиная от личных блогов и заканчивая e-commerce сайтами.  WordPress – идеальный вариант для новостных сайтов или блогов.

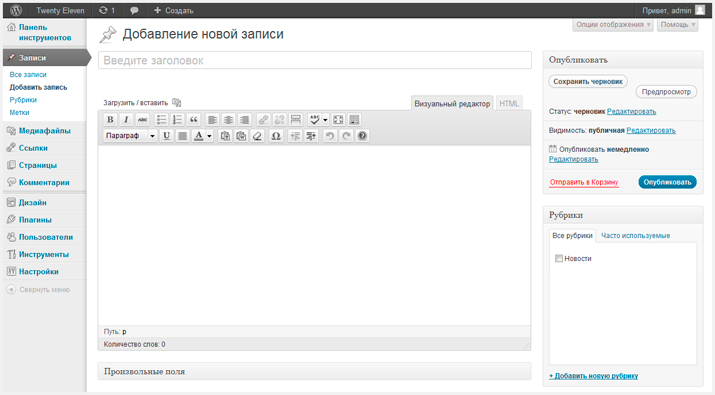


Рисунок 2.1.1 – Общий вид системы WordPress

Преимущества WordPress:

* Это наиболее популярная система: более половины пользователей предпочитают именно WordPress.
* Наиболее широкий набор плагинов, тем, виджетов для галерей, форумов, различные каталоги, магазины и так далее.
* WYSIWYG редактор, который позволяет создавать html страницы без знания языка разметки html.
* Система бесплатна

Недостатки WordPress:

* Система даёт огромное поле для выбора, поэтому сложно разобрать как правильно использовать все ее компоненты.
* Сложность установки.

### Joomla

Joomla следующая по популярности CMS, которой пользуются 9% пользователей. Главная особенность данной системы – это минимальный набор инструментов при первоначальной установке, который может дополняться по мере надобности. Это существенно уменьшает загромождение административной панели лишними элементами, а также уменьшает нагрузку на сервер и значительно экономит место на хостинге. Joomla дает возможность показывать интерфейс фронтальной и административной части на всех языках.

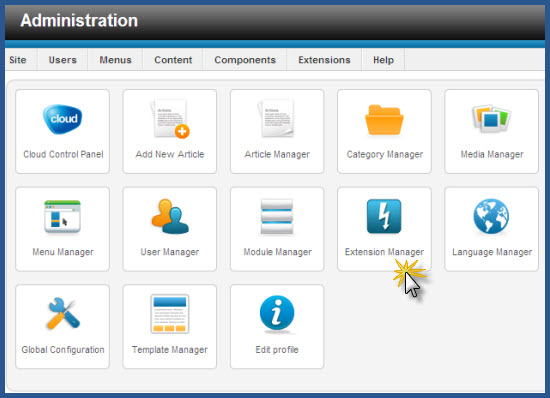


Рисунок 2.1.2 – Общий вид системы Jooomla

Преимущества Joomla:

* Несмотря на простоту, является полноценным инструментом для разработки.
* Поддержка протоколов контроля доступа (OpenID, LDAP, Gmail.com);
* Наличие удобной административной панели с широким набором функций: шаблоны, стили, управление меню и так далее.
* Простой процесс установки для неопытных пользователей.
* Наличие встроенного менеджера рассылки новостей, поддержка RSS.
* Красивый интерфейс.
* Система бесплатна

Недостатки Joomla:

* Много платных плагинов и тем.
* Необходимость регулярного обновления.
* Слишком проста для некоторых задач (например, для создания комерческих сайтов).

### Drupal

Около 7% пользователей предпочитают Drupal. Очень мощная система и позволяет создавать сайты любого уровня сложности. Обычно используется для сайтов, требующих сложной организации данных: для форумов, онлайн-магазинов, веб-блогов, корпоративных веб-сайтов.

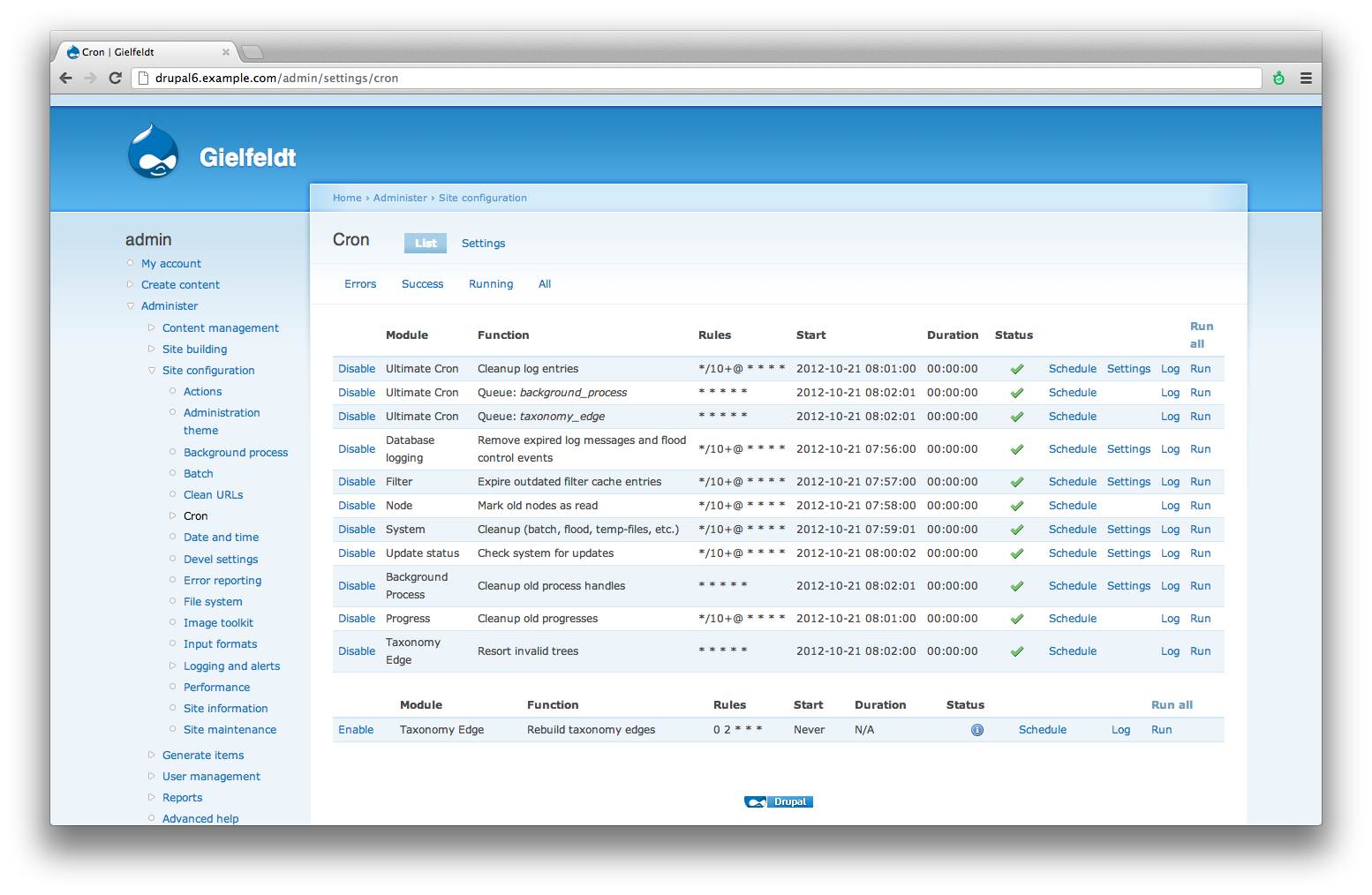


Рисунок 2.1.3 – Общий вид системы Drupal

Преимущества Drupal:

* Система очень гибкая. В отличие от joomla и wordpress можно менять структуру своего сайта как угодно.
* Огромное количество модулей (расширений), которые расширяют функциональные возможности drupal
* Приятный и дружественный интерфейс административной панели управления.Особенно мне нравится верхний тулбар администратора на фронтальной части сайта.
* Система бесплатна.

Недостатки Drupal:

* Высокий порог вхождения в изучении.
* Высокая нагрузка на сервер.
* В некоторых случаях для реализации того или иного функционала требуются навыки программирования.
* Сложность создании собственных темы.
* Не все модули переведены на русский язык

Данные системы разрабатываються для универсального применения, поэтому разработка собственной административной панели имеет ряд преимуществ:

* Простота в использовании.
* Наличие только необходимого функционала.
* Уменьшение возможных проблем с безопасностью.
* Более быстрая и оптимизированная работа системы.

## Календарный план по разработке выпускной работы бакалавра

На рисунке 2.2 изображен календарный план проекта разрабокти административной панели

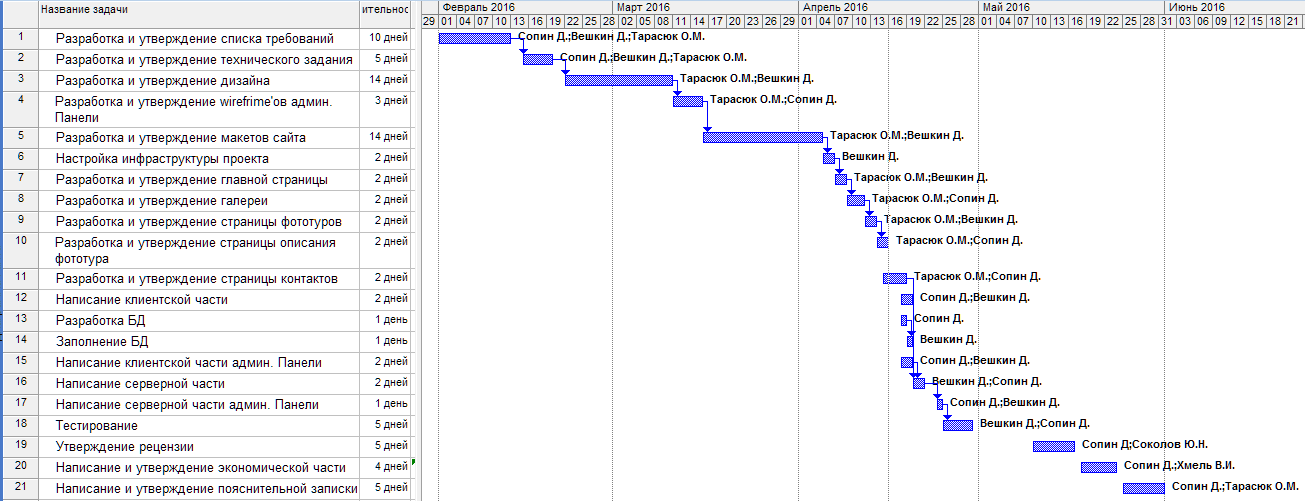
****

Рисунок 2.2 – Календарный план разработки административной панели

1. Этап разработки и утверждения списка требований

На данном этапе разрабатываются и утверждаются основные требования к административной панели: общий внешний вид, функционал предъявляемый к панели.

1. Этап разрабокти и утверждения технического задания.

На данном этапе разрабатываеться технического задание проекта согласно утвержденным ранее требованиям.

1. Разработка и утверждение wireframe’ов

Разработка и утверждение wireframe’ов административной паели согласно утвержденному дизайну.

1. Разработка и настройка инфраструктуры проекта

Разработка структуры папок проекта, создания gulp задач.

1. Разрботка базы данных проекта

Разработка базы данных проекта, нормализация таблиц, создание скриптов создание БД.

1. Разработка клиентской части проекта

Верстка административной панели соглано утвержденным wireframe’ов. Написание javasrcipt-кода с использование фреймворка backbone.

1. Разработка серверной части проекта

Разработка серверной части административной панели сайта с использованием фреймворка express, разработка необходимых API для взаимодействия с клиентской частью и базой данных. Реализация защиты доступа к API и статических файлов от несанкционированного доступа со стороны неавторизированных пользователей.

1. Тестирование

Тестроование разработанной административной панели на соответсвие утвержденному техническому заданию.

## Выводы по разделу 2

В данном разделе проведен анализ существующих административных панелей, обоснована разработка собственной административной панели. Построен календарный планы выполнения выпускной работы.

# ПРОЕКТИРОВАНИЕ АДМИНИСТРАТИВНОЙ ПАНЕЛИ САЙТА

## Выбор архитектуры ПО

### Архитектура клиентской части

Клиентская сторона приложения будет выполнена в виде SPA-приложения. Данная технология хорошо в первую очередь быстротой отклика на действия в пользовательском интерфейсе, так как не требуется полной или даже частичной перезагрузки WEB-страницы с сервера, а все визуальные элементы конструируются прямо в браузере с помощью JavaScript путем манипуляций с DOM-структурой документа.   
 Таким образом, WEB-приложения становятся очень похожи на обычные приложения для рабочих станций, загружающих информацию из сети Интернет, только средой исполнения для них является не операционная система, а браузер.

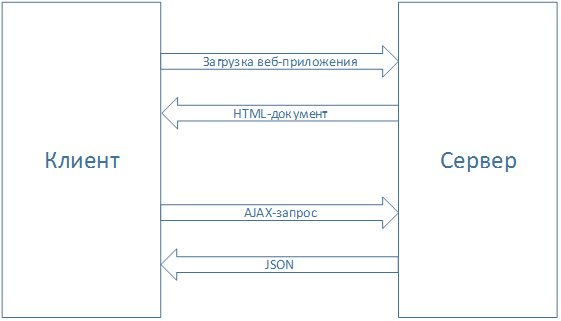


Рис. 3.1.1 Схема архитектуры SPA.

### Архитектура серверной части

Серверная часть будет выполнена в виде REST API.  REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

 Рис. 3.1.2 Схема архитектуры REST.

## Разработка wireframe’ов проекта

Разработка wireframe’ов административной части выполнялась на основе утвержденных требованиях и общего вида панели. Для разработки использовался сайт mybalsamiq.com, который позволяет быстро и качество выполнить wireframe’мы любой сложности. Сайт имеет много готовых инструментов, обладает понятным интерфейсом.

### Раздел «Категориии» (главная страница)

На данной странице будут находиться все категории и их имена. Категории можно удалить и создать новую.

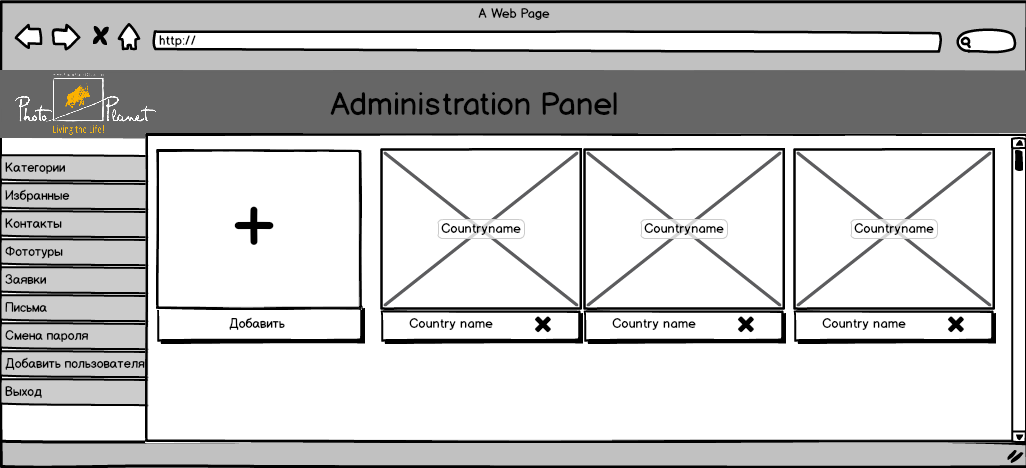
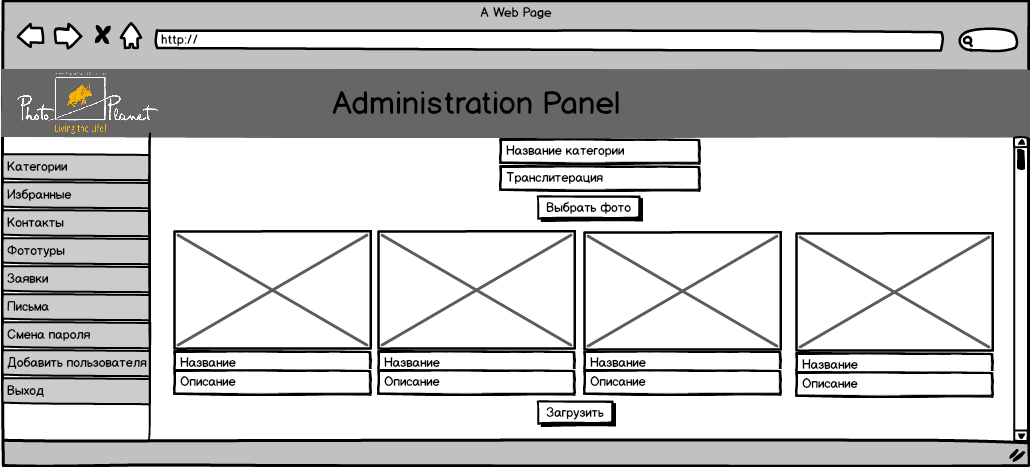


Рис. 3.2.1 – Страница «Категории»

#### Страница добавление категории

На данной странице можно создать новую категорию. Можно выбрать фотографии, ввести название, подкоректировать транслитерацию и ввести название и описание каждой фотографии.

 Рис. 3.2.2 – Страница добавления новой категории

#### Страница фотографий категории

На данной странице выводяться все фотографии выбраной категории. Также присутсвует возможность добавление новых фотографий, смены обложки категории, добавление/удаление категории из избранных, редактирования описание и названия, а также удаления фотографии.

### Раздел «Избранные»

На данной странице выводятся все фотографии из списка избранных. Есть возможность удаления фотографии из избранного, редактирования описания и удаления фотографии.

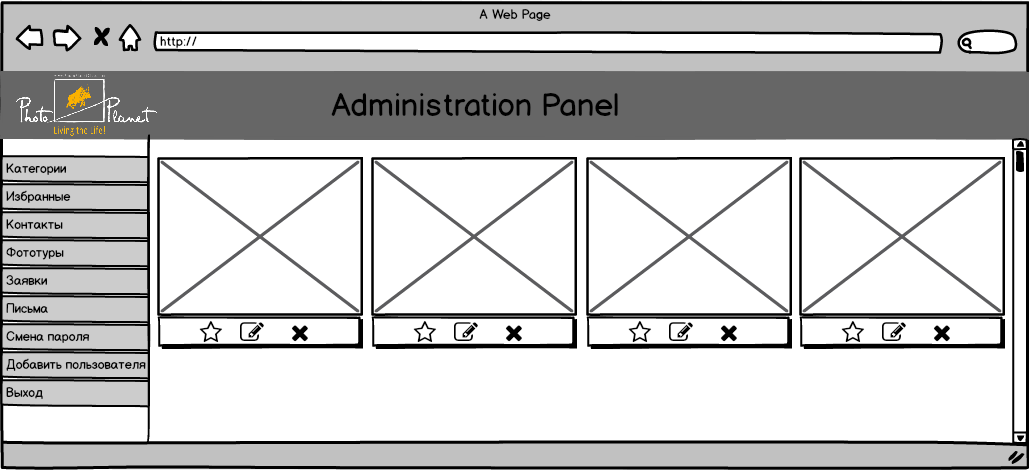


Рис. 3.2.3 – Страница фотографий категории

### Раздел «Контакты»

На данной странице можно редактировать контактные данные администратора - ссылки на социальные сети и адрес электронной почты.

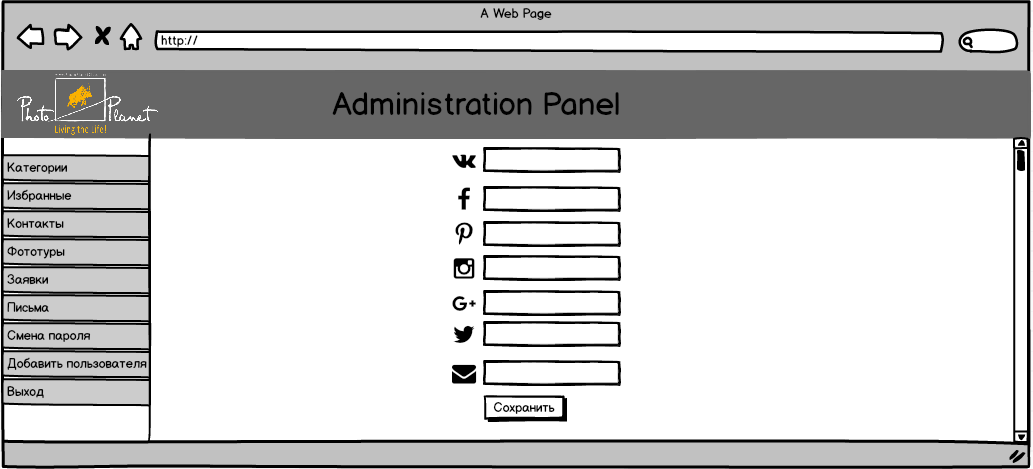


Рис. 3.2.4 – Страница «Контакты»

### Раздел «Фототуры»

На данной странице выводятся все созданные фототуры, разделенные на будущие и прошедшие. Также выводятся их описание, обложка. Есть возможность редактировать тур, удалить или создать новый.

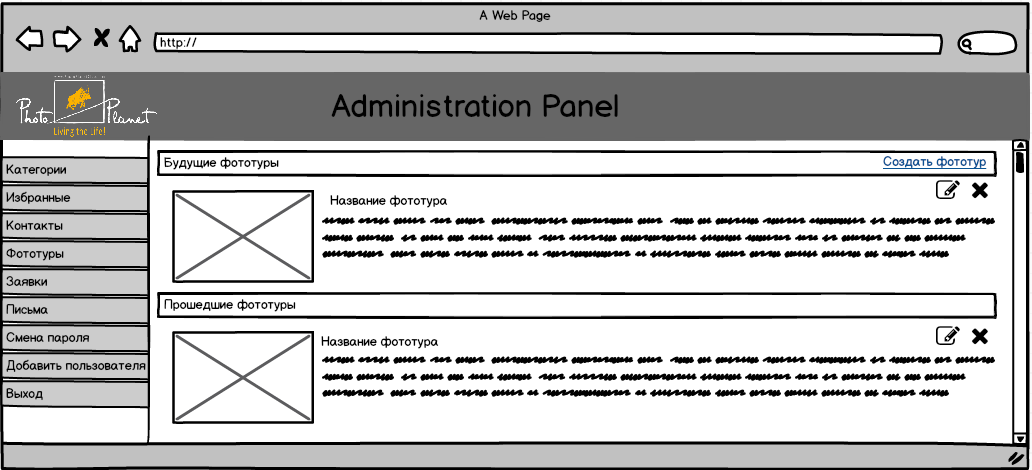


Рис. 3.2.5 – Страница «Фототуры»

#### Страница создания/редактирования фототура

Страницы создания и редактирования туров имееют одинаковую структуру. На странице присутсвуют все информационные поля фототура: название, дата начала и окончания, описание, карта с местопроведением, цена, количество мест, сложность, список что включено и не включено в стоимость, расписание и фотографии относящиеся к туру.

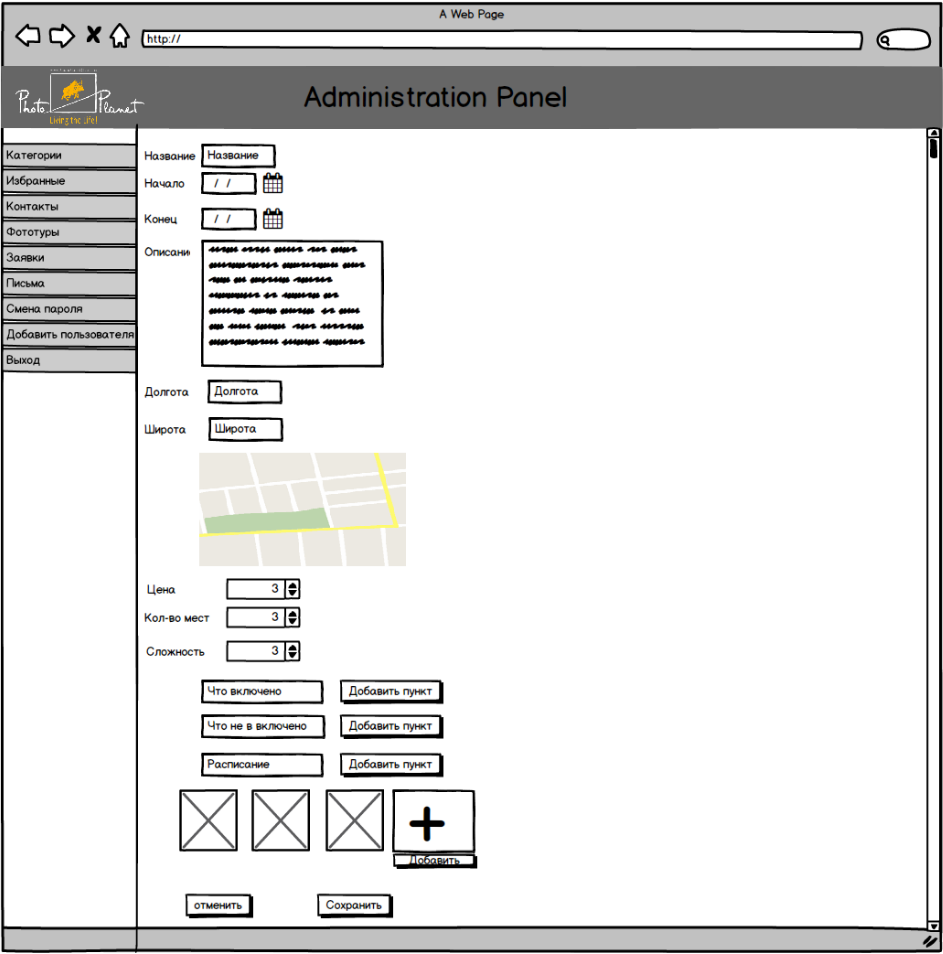
****

Рис. 3.2.6 – Страница Страница добавления/редактирования тура

### Раздел «Заявки»

На данной странице будет выводиться информация о фототурах на которые есть заявки. Данные будут выводиться в виде таблицы в которой будет присутсивовать общее количество новых, принятых, отклоненных заявок.

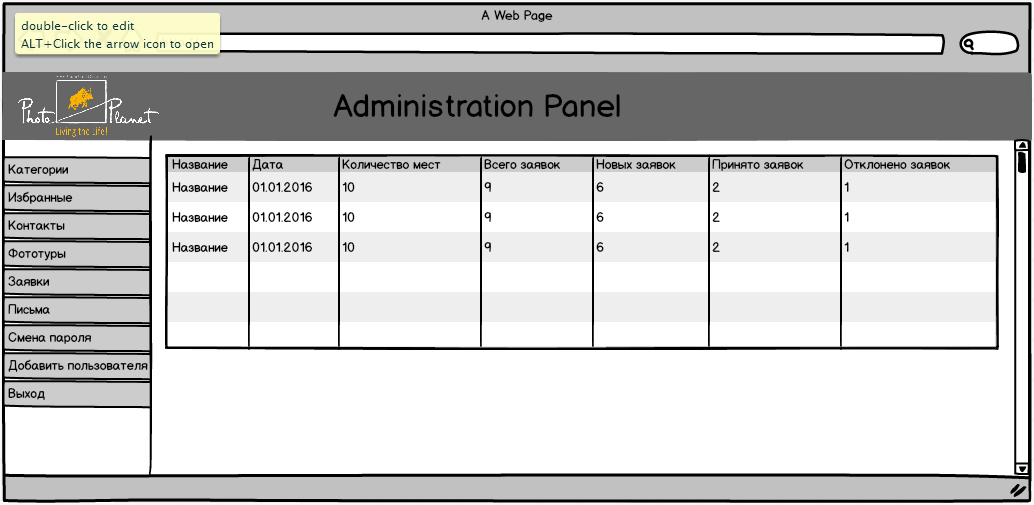


Рис. 3.2.7 – Страница «Заявок»

#### Страница модерации заявок

После выбора конкретного фототура будет выведен список заявок с их текущими статусами и кнопками управления(принять, отклонить и удалить).

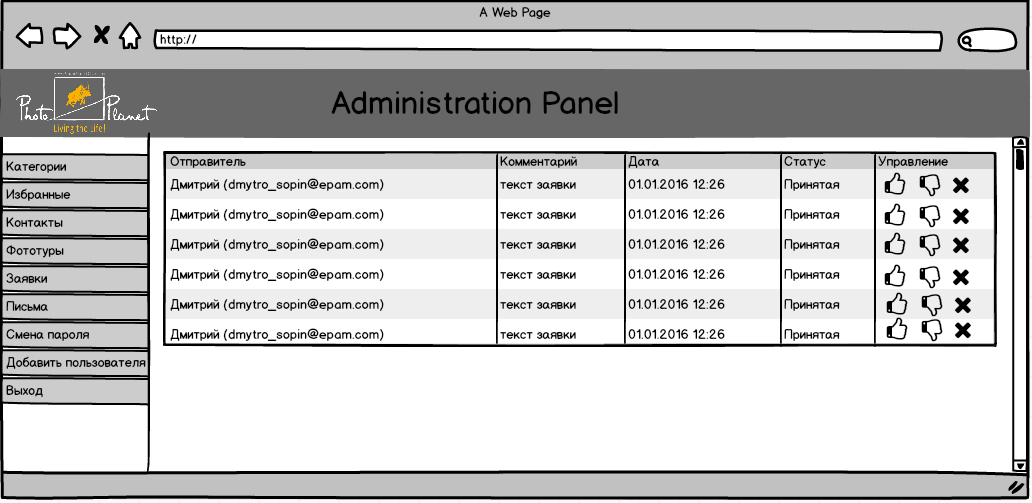


Рис. 3.2.8 – Страница модерации заявок

### Раздел «Письма»

На данной странице выводяться все письма, которые отправили пользователи сайта. Присутствует имя, адрес электронной почты, текст письма и дата отправки. Все письма можно удалять при помощи кнопки.

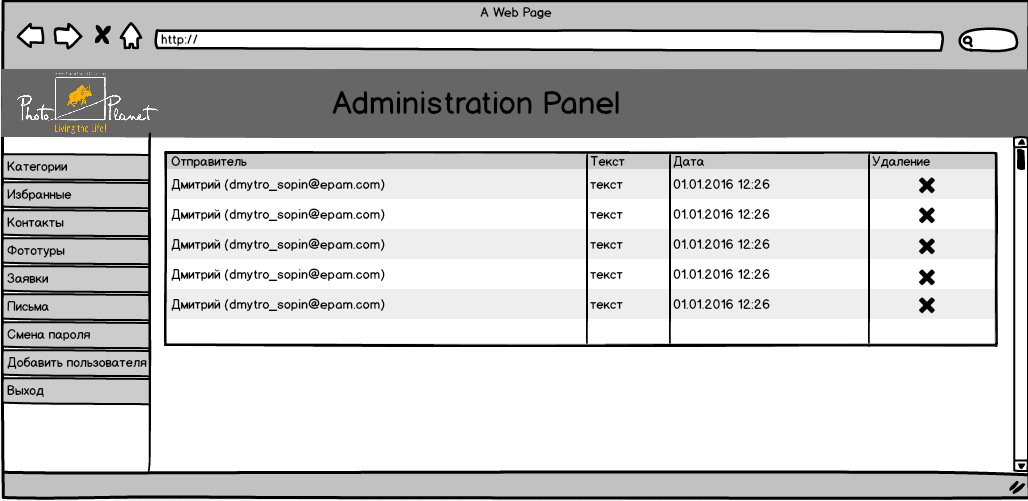


Рис. 3.2.9 – Страница «Письма»

### Раздел «Смена пароля»

На данной странице можно сменить пароль администратора, введя текущий и новый пароль.

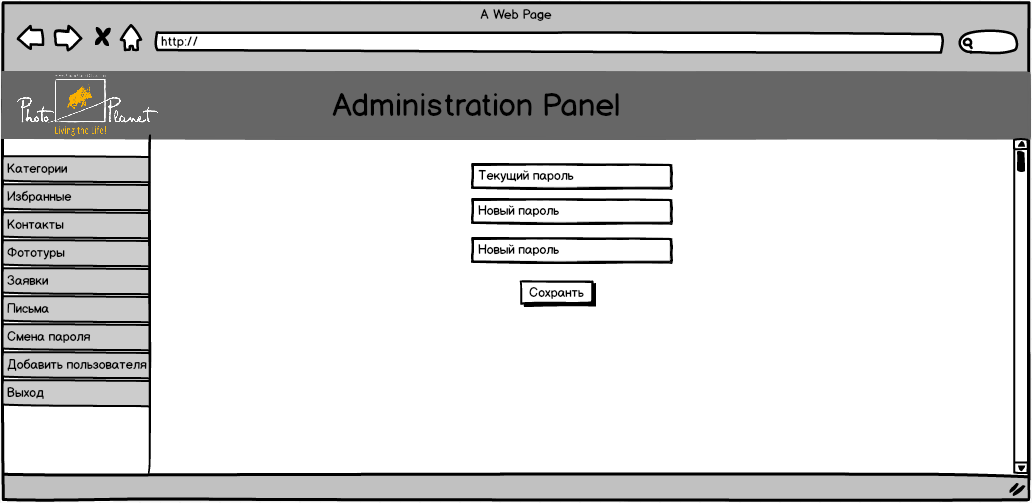


Рис. 3.2.10 – Страница «Смена пароля»

### Раздел «Добавить нового пользователя»

На данной странице можно добавить нового администратора сайта, указав его логин и пароль.

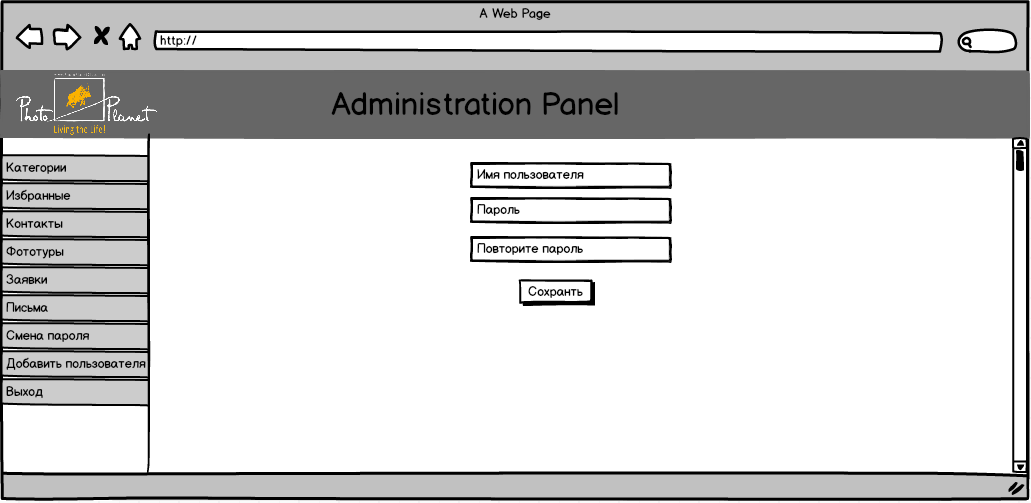


Рис. 3.2.11 – Страница «Добавления новго пользоавателя»

### Страница авторизации

На данной странице выполняется авторизация администратора для входа в панель. На данную страницу можно попасть нажав «Выход» в административной панели или через ссылку на основном сайте.

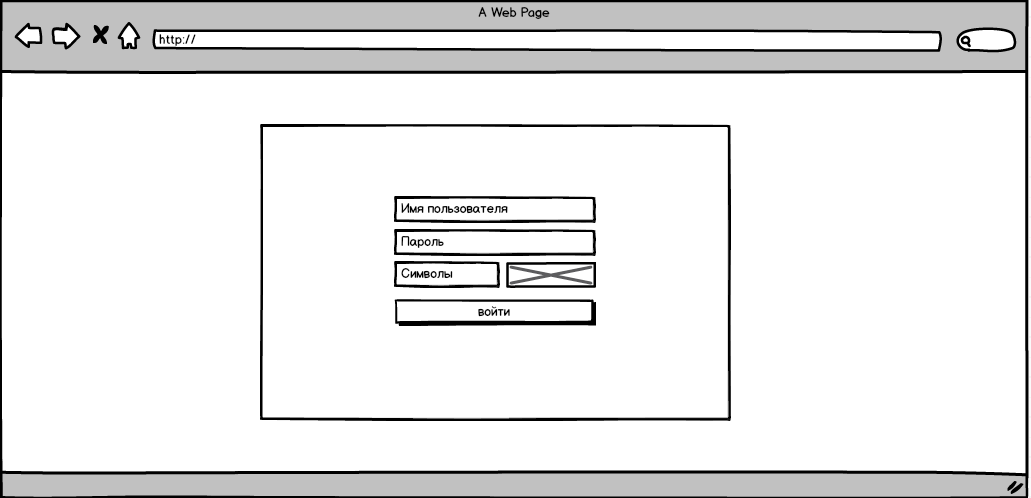


Рис. 3.2.12 – Страница авторизации

## Разработка инфраструктуры проекта

Для сборки проекта будет использоваться Gulp, который позволяет создавать задачи для потоковой работы. При помощи данного сборщика будут разработаны скрипты, которые будут решать следующие задачи:

* Минифицаия всех javascript-файлов проекта.
* Обьединение всех минифицированных javascript-файлов проекта в один.
* Компиляция шаблонов сайта.
* Компиляция sass стилей в синтаксис css.
* Объеденение css файлов проекта в один.
* Подстановка браузерных префиксов для улучшение кроссбраузерности.
* Автоматическая сборка проекта при изменения файлов.

Данные действия позволяют получить на выходе проект, в котором:

* Максимально уменьшен размер файлов за счет их минифиции.
* Будет сокращено количество http-запросов за счет объеденения файлов в один.
* Достигнута максимальная кроссбраузерность за счет префиксов.

В результате загрузка страниц у пользователя будет максимально быстрой, а сервер будет менее нагружен за счет сокращения http-запросов к нему.

### Код Gulp скрипта

**var *gulp*** = require(**'gulp'**);  
**var *watch*** = require(**"gulp-watch"**);  
**var *sourcemaps*** = require(**"gulp-sourcemaps"**);  
**var *autoprefixer*** = require(**"gulp-autoprefixer"**);  
**var *cssnano*** = require(**"gulp-cssnano"**);  
**var *sass*** = require(**"gulp-sass"**);  
**var *del*** = require(**"del"**);  
**var *browserify*** = require(**'gulp-browserify'**);  
**var *concat*** = require(**'gulp-concat'**);  
**var *hbsfy*** = require(**'hbsfy'**);  
**var *uglify*** = require(**"gulp-uglify"**);  
**var *imagemin*** = require(**"gulp-imagemin"**);  
  
**var *path*** = {  
 **src**: {  
 **scss**: [**"public/src/scss/main.scss"**],  
 **mainjs**: [**"public/src/js/main.js"**],  
 **js**: [**"public/src/js/\*\*/\*.js"**],  
 **images**: **"images/src/\*\*/\*"**,  
 **copy**: {  
 **html**: [  
 **"public/src/\*.html"**,  
 **"public/src/\*.txt"**,  
 **"public/src/\*.jsp"** ],  
 **fonts**: [**"public/src/fonts/\*\*/\*"**],  
 **libs**: [**"public/src/libs/\*\*/\*"**]  
 },  
 **templates**: [**"public/src/\*\*/\*.hbs"**]  
 },  
 **build**: {  
 **css**: **"public/build/css"**,  
 **js**: **"public/build/js"**,  
 **images**: **"images/build/"**,  
 **copy**: {  
 **html**: **"public/build/"**,  
 **fonts**: **"public/build/fonts/"**,  
 **libs**: **"public/build/libs/"** },  
 **templates**: **"public/build/templates/"** },  
 **watch**: {  
 **scss**: **"public/src/scss/\*\*/\*.scss"** }  
};  
  
**var *adminPath*** = {  
 **src**: {  
 **scss**: [**"admin\_public/src/scss/main.scss"**],  
 **mainjs**: [**"admin\_public/src/js/main.js"**],  
 **js**: [**"admin\_public/src/js/\*\*/\*.js"**],  
 **images**: **"images/src/\*\*/\*"**,  
 **copy**: {  
 **html**: [  
 **"admin\_public/src/\*.html"**,  
 **"admin\_public/src/\*.txt"**,  
 **"admin\_public/src/\*.jsp"** ],  
 **fonts**: [**"admin\_public/src/fonts/\*\*/\*"**],  
 **libs**: [**"admin\_public/src/libs/\*\*/\*"**]  
 },  
 **templates**: [**"admin\_public/src/\*\*/\*.hbs"**]  
 },  
 **build**: {  
 **css**: **"admin\_public/build/css"**,  
 **js**: **"admin\_public/build/js"**,  
 **images**: **"images/build/"**,  
 **copy**: {  
 **html**: **"admin\_public/build/"**,  
 **fonts**: **"admin\_public/build/fonts/"**,  
 **libs**: **"admin\_public/build/libs/"** },  
 **templates**: **"admin\_public/build/templates/"** },  
 **watch**: {  
 **scss**: **"admin\_public/src/scss/\*\*/\*.scss"** }  
};  
  
**var *buildTasks*** = [**"scss:build"**, **"js:build"**];  
**var *adminBuildTasks*** = [**"admin:scss:build"**, **"admin:js:build"**];  
  
***gulp***.**task**(**"scss:build"**, **function** () {  
 ***gulp***.**src**(***path***.**src**.**scss**)  
 .**pipe**(***sourcemaps***.init())  
 .**pipe**(***sass***())  
 .**pipe**(***autoprefixer***())  
 .**pipe**(***cssnano***())  
 .**pipe**(***sourcemaps***.write())  
 .**pipe**(***gulp***.**dest**(***path***.**build**.**css**));  
});  
  
***gulp***.**task**(**"admin:scss:build"**, **function** () {  
 ***gulp***.**src**(***adminPath***.**src**.**scss**)  
 .**pipe**(***sourcemaps***.init())  
 .**pipe**(***sass***())  
 .**pipe**(***autoprefixer***())  
 .**pipe**(***cssnano***())  
 .**pipe**(***sourcemaps***.write())  
 .**pipe**(***gulp***.**dest**(***adminPath***.**build**.**css**));  
});  
  
***gulp***.**task**(**"js:build"**, **function** () {  
 ***gulp***.**src**(***path***.**src**.**mainjs**)  
 .**pipe**(***browserify***({  
 **transform**: **'hbsfy'**,  
 **debug**: **true** }))  
 .**pipe**(***gulp***.**dest**(***path***.**build**.**js**));  
});  
  
***gulp***.**task**(**"admin:js:build"**, **function** () {  
 ***gulp***.**src**(***adminPath***.**src**.**mainjs**)  
 .**pipe**(***browserify***({  
 **transform**: **'hbsfy'**,  
 **debug**: **true** }))  
 .**pipe**(***gulp***.**dest**(***adminPath***.**build**.**js**));  
});  
  
***gulp***.**task**(**"img:build"**, **function** () {  
 ***gulp***.**src**(***path***.**src**.**images**)  
 .**pipe**(***imagemin***({  
 **progressive**: **true**,  
 **svgoPlugins**: [{**removeViewBox**: **false**}],  
 **optimizationLevel**: 5  
 }))  
 .**pipe**(***gulp***.**dest**(***path***.**build**.**images**));  
});  
  
***gulp***.**task**(**"clean"**, **function** () {  
 **return *del***([**"public/build/\*"**], {**force**: **true**});  
});  
  
***gulp***.**task**(**"admin:clean"**, **function** () {  
 **return *del***([**"admin\_public/build/\*"**], {**force**: **true**});  
});  
  
**for** (**var *i* in *path***.**src**.**copy**) {  
 **if** (***path***.**src**.**copy**.hasOwnProperty(***i***)) {  
 ***gulp***.**task**(**'copy\_'** + ***i***, (**function** (i) {  
 **return function** () {  
 ***gulp***.**src**(***path***.**src**.**copy**[i])  
 .**pipe**(***gulp***.**dest**(***path***.**build**.**copy**[i]));  
 }  
 })(***i***));  
 ***buildTasks***.push(**"copy\_"** + ***i***);  
 }  
}  
  
**for** (**var *j* in *adminPath***.**src**.**copy**) {  
 **if** (***adminPath***.**src**.**copy**.hasOwnProperty(***j***)) {  
 ***gulp***.**task**(**'admin:copy\_'** + ***j***, (**function** (i) {  
 **return function** () {  
 ***gulp***.**src**(***adminPath***.**src**.**copy**[i])  
 .**pipe**(***gulp***.**dest**(***adminPath***.**build**.**copy**[i]));  
 }  
 })(***j***));  
 ***buildTasks***.push(**"admin:copy\_"** + ***j***);  
 }  
}  
  
***gulp***.**task**(**'watch'**, **function** () {  
 **for** (**var** i **in *path***.**src**.**copy**) {  
 **if** (***path***.**src**.**copy**.hasOwnProperty(i)) {  
 ***watch***(***path***.**src**.**copy**[i], (**function** (i) {  
 **return function** (event, cb) {  
 ***gulp***.start(**'copy\_'** + i);  
 }  
 })(i));  
 }  
 }  
  
 **for** (**var** j **in *adminPath***.**src**.**copy**) {  
 **if** (***adminPath***.**src**.**copy**.hasOwnProperty(j)) {  
 ***watch***(***adminPath***.**src**.**copy**[j], (**function** (i) {  
 **return function** (event, cb) {  
 ***gulp***.start(**'admin:copy\_'** + i);  
 }  
 })(j));  
 }  
 }  
  
 ***watch***(***path***.**watch**.**scss**, **function** (event, cb) {  
 ***gulp***.start(**'scss:build'**);  
 });  
  
 ***watch***(***path***.**src**.**js**, **function** (event, cb) {  
 ***gulp***.start(**'js:build'**);  
 });  
  
 ***watch***(***path***.**src**.**templates**, **function** (event, cb) {  
 ***gulp***.start(**'js:build'**);  
 });  
  
 ***watch***(***adminPath***.**watch**.**scss**, **function** (event, cb) {  
 ***gulp***.start(**'admin:scss:build'**);  
 });  
  
 ***watch***(***adminPath***.**src**.**js**, **function** (event, cb) {  
 ***gulp***.start(**'admin:js:build'**);  
 });  
  
 ***watch***(***adminPath***.**src**.**templates**, **function** (event, cb) {  
 ***gulp***.start(**'admin:js:build'**);  
 });  
  
 ***watch***(***path***.**src**.**images**, **function** (event, cb) {  
 ***gulp***.start(**'img:build'**);  
 });  
});  
  
***gulp***.**task**(**'build'**, ***buildTasks***);  
***gulp***.**task**(**'admin:build'**, ***adminBuildTasks***);  
***gulp***.**task**(**'default'**, [**"build"**, **"admin:build"**, **'watch'**]);

## Проектирование базы данных

Разрабатываемое ПО будет обрабатывать и хранить большие объемы данных, поэтому целесообразно хранить эту информацию в базах данных. Базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД).

Наиаиболее важные цели, необходимые перед проектированием базы данных (БД):

* исследование предметной области и выделение основных задач ,для решения которых предназначена БД;
* создание списка таблиц БД ;
* создание структуры каждой таблицы;
* определение первичных и вторичных ключей ;
* создание связей между таблицами и ссылочная целостность ;
* нормализация БД.

Сущность – любой различимый объект (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных.

Атрибут – поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей.

Связь – ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой.

### Проектирование логической модели базы данных

Сущности и атрибуты БД

1. Сущность – Фотограии
   1. Уникальный идентификатор фотографии.
   2. Относительный путь фотографии.
   3. Идентификатор пренадлжености к стране .
   4. Флаг пренадлежности к избранным фотографиям.
   5. Название фотографии.
   6. Описание фотографии.
2. Сущность – Страны
   1. Уникальный идентификатор страны.
   2. Имя страны.
   3. Транслитерация имени страны.
   4. Путь к обложке страны.
3. Сущность – Фототуры
   1. Уникальный идентификатор тура.
   2. Название тура.
   3. Описание тура.
   4. Дата начала тура.
   5. Дата окончания тура.
   6. Стоимость тура.
   7. Количество мест.
   8. Сложность тура.
   9. Координата широты проведения тура.
   10. Координата долготы проведения тура.
   11. Список, что включено в стоимость тура.
   12. Список, что не включено в стоимость тура.
   13. Фотографии относящиеся к туру.
   14. Расписание тура по дням.
   15. Обложка тура.
4. Сущность – Контакты
   1. Адрес электронной почты администратора.
   2. Ссылка на социальную сеть Google+.
   3. Ссылка на социальную сеть Instagram.
   4. Ссылка на социальную сеть Pinterest.
   5. Ссылка на социальную сеть vkontakte.
   6. Ссылка на социальную сеть facebook.
   7. Ссылка на социальную сеть twitter.
5. Сущность – отзывы обратной связи
   1. Уникальный идентификатор отзыва.
   2. Имя отправителя.
   3. Адрес электронной почты отправителя.
   4. Текст отзыва.
   5. Дата отправления.
6. Сущность – Заявки на фототуры
   1. Уникальный идентификатор заявки.
   2. Идентификатор тура на который сделана заявка.
   3. Имя отправителя.
   4. Адрес электронной почты отправителя.
   5. Текст заявки.
   6. Дата отправления.
   7. Идентификатор текущего статуса заявки.
7. Сущность – Статусы заявки
   1. Уникальный идентификатор статуса.
   2. Название статуса.
8. Сущность – пользователи
   1. Уникальный идентификатор пользователя.
   2. Уникальный логин пользователя.
   3. Хеш пароля пользователя.

### Связи между сущностями

Для сущностей базы данных административной панели были определены связи.

Связь один ко многим реализована между сущностями Страна – Фотографии (Рисунок 3.1.3).



Рисунок 3.3.1 – Связь один ко многим Страна – Фотографии

Связь один ко многим реализована между сущностями Заявки – Фототур (Рисунок 3.2.3).



Рисунок 3.3.2 – Связь один ко многим Фототур – Заявки

Связь один ко многим реализована между сущностями Заявка – Статус (Рисунок 3.3.3).



Рисунок 3.3.3 – Связь один ко многим Заявка – Статус

### Представление сущностей в виде таблиц

Таблицы *Photos* и *Countries* предназначены для хранения информации о фотографиях и категориях.

В таблице *Photos* будет храниться имя файла с фотографией, country\_id - внешний ключ на таблицу Countries, описание, заголовок фотографии, а также относится ли данная фотография к категории "Избранное".

В таблице *Countries* будет храниться id странны, на который будет ссылаться таблица Photos и название страны.

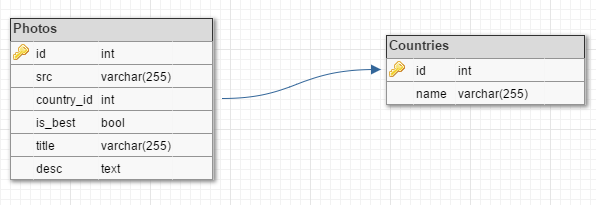


Рис. 3.3.4 Физические модели таблиц Photos и Countries

Таблица 3.3.1 - *Photos*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Photos* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | int | \* |  | \* | \* |
| *src* | Имя файла | varchar(255) |  |  | \* | \* |
| *coutry\_id* | id страны | int |  | \* | \* |  |
| *is\_best* | избранное фото | bool |  |  | \* |  |
| *title* | заголовок фото | varchar(255) |  |  |  |  |
| *desc* | описание фото | text |  |  |  |  |

Таблица 3.3.2 - *Countries*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Countries* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | int | \* |  | \* | \* |
| *name* | Название страны | varchar(255) |  |  | \* |  |

В таблице Users будут хранится login и хеш пароля пользователя. Эти данные необходимы для авторизации в административную панель.

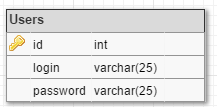


Рис. 3.3.5 Физическая модель таблицы Users

Таблица 3.3.3 - *Users*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Users* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | int | \* |  | \* | \* |
| *login* | логин пользователя | varchar(25) |  |  | \* | \* |
| *password* | хеш пароля | varchar(25) |  |  | \* |  |

В таблице Feedbacks будут храниться вопросы и пожелания пользователей со страницы "Контакты". Таблица будет хранить имя, электронный адрес, текст пользователя и время отправки.

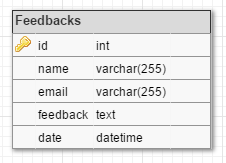


Рис. 3.3.6 Физическая модель таблицы Feedbacks

Таблица 3.3.4 - *Feedbacks*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Feedbacks* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | int | \* |  | \* | \* |
| *name* | имя | varchar(255) |  |  | \* |  |
| *email* | email пользователя | varchar(255) |  |  | \* |  |
| *feedback* | вопрос пользователя | text |  |  | \* |  |
| *date* | дата отправки | datetime |  |  | \* |  |

В таблице *Contacts* будут храниться ссылки на социальные сети (google+, instagram, pinterest,vk, facebook, twitter) и email администратора. На сайте будут выводиться только ссылки на сохраненные социальные сети.



Рис. 3.3.7 Физическая модель таблицы Contacts

Таблица 3.3.5 - *Contacts*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Contacts* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *email* | email администратора | varchar(255) |  |  | \* |  |
| *googlePlus* | ссылка на google+ | varchar(255) |  |  | \* |  |
| *instagram* | ссылка на instagram | varchar(255) |  |  | \* |  |
| *pinterest* | ссылка на pinterest | varchar(255) |  |  | \* |  |
| *vkontakte* | ссылка на vk | varchar(255) |  |  | \* |  |
| *facebook* | ссылка на facebook | varchar(255) |  |  | \* |  |
| *twitter* | ссылка на twitter | varchar(255) |  |  | \* |  |

В таблицах *Tours* и *Requests* будет храниться информация о фототурах и заявок на фототуры от пользователей.

Таблица *Tours* будет содержать информацию о фототуре: название, описание, дата начала и окончания, сложность, количество свободных мест и стоимость.

Таблица *Requests* будет содержат информацию заявки от пользователя: имя, email, текст заявки, дату подачи, идентификатор статуса и ссылку на выбранный тур.

Таблица *Requests\_statuses будет содержать название статусов заявок.*

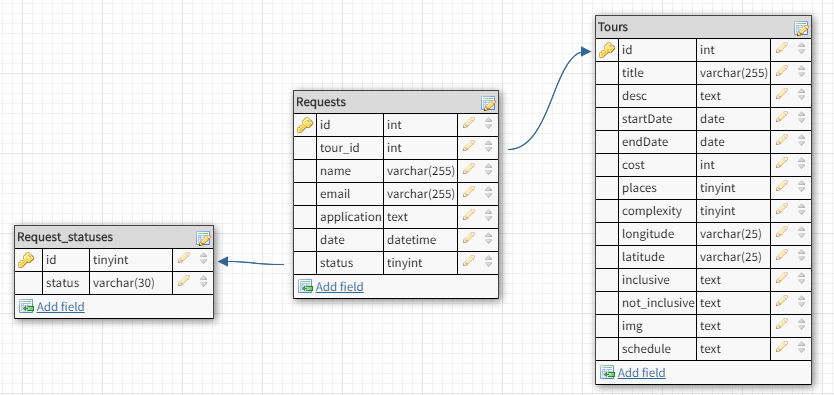
 Рис. 3.3.8 Физическая модель таблиц Tours, Requests и Requests\_statuses

Таблица 3.3.5 - *Tours*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Tours* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | int | \* |  | \* | \* |
| *title* | название | varchar(255) |  |  | \* |  |
| *desc* | описание | text |  |  | \* |  |
| *startDate* | дата начала | date |  |  | \* |  |
| *endDate* | дата окончания | date |  |  | \* |  |
| *complexity* | сложность | tinyint |  |  | \* |  |
| *places* | количество мест | tinyint |  |  | \* |  |
| *cost* | стоимость | int |  |  | \* |  |

Таблица 3.3.6 - *Requests*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Requests* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | int | \* |  | \* | \* |
| *name* | имя | varchar(255) |  |  | \* |  |
| *email* | email | varchar(255) |  |  | \* |  |
| *request* | текст заявки | text |  |  | \* |  |
| *date* | дата подачи | datetime |  |  | \* |  |
| *stratus* | текущий статус | tinyint |  | \* | \* |  |
| *tour\_id* | ссылка на тур | int |  |  | \* |  |

Таблица 3.3.7 - *Requests\_statuses*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица *Requests* | | | | | | |
| *Поле* | *Название* | *Тип данных* | *PRIMARY KEY* | *FOREIGN*  *KEY* | *NOT NULL* | *UNIQUE* |
| *id* | id | tinyint | \* |  | \* | \* |
| *status* | название статуса | varchar(30) |  |  | \* |  |

### Разработка скриптов для реализации БД административной панели

CREATE TABLE `Photos` (

`id` int NOT NULL,

`src` varchar(255) NOT NULL,

`country\_id` int NOT NULL,

`is\_best` bool NOT NULL DEFAULT 'false',

`title` varchar(255) NOT NULL,

`desc` TEXT NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `Countries` (

`id` int NOT NULL,

`name` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `Users` (

`id` int NOT NULL,

`login` varchar(25) NOT NULL UNIQUE,

`password` varchar(25) NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `Feedbacks` (

`id` int NOT NULL,

`name` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`feedback` TEXT NOT NULL,

`date` DATETIME NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `Contacts` (

`email` varchar(255) NOT NULL,

`googlePlus` varchar(255),

`instagram` varchar(255),

`pinterest` varchar(255),

`vkontakte` varchar(255),

`facebook` varchar(255),

`twitter` varchar(255)

);

CREATE TABLE `Tours` (

`id` int NOT NULL,

`title` varchar(255) NOT NULL,

`desc` TEXT NOT NULL,

`startDate` DATE NOT NULL,

`endDate` DATE NOT NULL,

`cost` int NOT NULL,

`places` tinyint NOT NULL,

`complexity` tinyint NOT NULL,

`longitude ` varchar(25) NOT NULL,

`latitude` varchar(25) NOT NULL,

`inclusive` TEXT NOT NULL,

`not\_inclusive` TEXT NOT NULL,

`img` TEXT NOT NULL,

`schedule` TEXT NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `Requests` (

`id` int NOT NULL,

`tour\_id` int NOT NULL,

`name` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`application` TEXT NOT NULL,

`date` DATETIME NOT NULL,

`status` tinyint NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `Request\_statuses` (

`id` tinyint NOT NULL,

`status` varchar(30) NOT NULL,

PRIMARY KEY (`id`)

);

ALTER TABLE `Photos` ADD CONSTRAINT `Photos\_fk0` FOREIGN KEY (`country\_id`) REFERENCES `Countries`(`id`);

ALTER TABLE `Requests` ADD CONSTRAINT `Requests\_fk0` FOREIGN KEY (`tour\_id`) REFERENCES `Tours`(`id`);

ALTER TABLE `Requests` ADD CONSTRAINT `Requests\_fk1` FOREIGN KEY (`status`) REFERENCES `Request\_statuses`(`id`);

## Разработка протокола взаимодействия клиентской части с серверной частью

В соответствии с выбранной архитектурой был разработан протокол взаимодействия между клиентской и серверной частей. Содержимое ответа и запроса строится с использованием нотации JSON. Описание протокола представлено в таблице 4.4.

Таблица 4.4 – api взаимодействия сервера и клиента

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Адрес запроса | Тип запроса | Атрибуты запроса | Описание запроса |
| 1 | api/feedback/ | GET | - | Возвращает все сообщения обратной связи |
| 2 | api/feedback/:id | GET | - | Возвращает сообщение обратной связи с данным id |
| 3 | api/feedback/:id | DELETE | - | Удаляет сообщение обратной связи с данным id |
| 4 | api/feedback/add | POST | name: string  email: string  feedback: string | Добавляет сообщение обратной связи |
| 5 | api/socials/ | GET | - | Возвращает всю контактную информацию |
| 6 | api/socials/ | PUT | email: string  vkontakte: string  instagram: string  pinterest: string  facebook: string  twitter: string  googlePlus: string | Обновляет контактную информацию |
| 7 | api/gallery/upload | POST | id: int  is\_best: bool  titlte: string  desc: string  Files: binary | Загружает фотографии в категории по id, записывает о них данные в БД |
| 8 | api/gallery/photo/update/:id | PUT | titlte: string  desc: string | Обновляет название и описание фото по id |
| 9 | api/gallery/countries | GET | - | Возвращает список стран и их обложки |
| 10 | api/gallery/cover/:country\_id/:photo\_id | GET | - | Меняет обложку альбома по id категории и фотографии |
| 11 | api/gallery/countries/add | POST | name: string  international: string | Добавляет новую категорию в БД |
| 12 | api/gallery/best | GET | - | Возвращает все избранные фотографии |
| 13 | api/gallery/add/best/:id | GET | - | Добавляет фото в избранное по id |
| 14 | api/gallery/remove/best/:id | GET | - | Удаляет фото из избранного по id |
| 15 | api/gallery/all | GET | - | Возврщает все фотографии |
| 16 | api/gallery/remove/:id | GET | - | Удаляет фото по id |
| 17 | api/gallery/country/:location | GET | - | Получает фотографии по имени категории |
| 18 | api/gallery/country/remove  /:location | GET | - | Удаляет категорию по ее имени |
| 19 | api/gallery/getGallery | GET | - | Возращает коллекцию фотография для галереи |
| 20 | api/tours | GET | - | Возвращает список фототуров |
| 21 | api/tours/create/new | POST | schedule: json  not\_inclusive: json  inclusive: json  startDate: date  endDate: date  files: binary  longitude: string  latitude: string | Добавляет новый фототур |
| 22 | api/tours/requests | GET | - | Возрщает список туров на которые есть заявки |
| 23 | api/tours/requests/:id | GET | - | Возвращает список заявок по id фототура |
| 24 | api/tours/requests  /accept/:id | POST | - | Помечает заявку на тур как приятую по id заявки |
| 25 | api/tours/requests/  reject/:id | POST | - | Помечает заявку на тур как отклоненную по id заявки |
| 26 | api/tours/requests  /delete/:id |  |  | Удаляет заявку на тур по id заявки |
| 27 | api/tours/prev | GET | - | Возвращает прошедшие фототуры |
| 28 | api/tours next | GET | - | Возвращает будущие фототуры |
| 29 | api/tours/:id | GET | - | Возращает полную информацию о туре по id |
| 30 | api/ ours/:id | POST | schedule: json  not\_inclusive: json  inclusive: json  startDate: date  endDate: date  files: binary  longitude: string  latitude: string | Обновляет информацию о туре по его id |
| 31 | api/tours/remove/:id | GET | - | Удаляет фототура по id |
| 32 | /api/password/change | POST | login: string  password: string | Изменяет пароль администратора |
| 33 | /api/password/add | POST | login: string  password: string | Добавляет ногово администратора в БД |
| 34 | /api/password/login | POST | login: string  password: string  captcha: string | Провеяет наличие логина и пароля в БД |
| 35 | /api/password/logout | GET | - | Выход из административной панели сайта |

## Выводы по разделу 4

В даннном разделе была выбрана архитектура клиентской и серверной части административной панели. Разработана база данных под управление СУБД MySQL. Разработы wireframe’ы для всех страниц сайта. Разработана инфраструктура проекта. Описаны протоколы взаимодействия клиента с сервервом.

# ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## Цель экономической части

Целью данной экономической части является расчет себестоимости и цены (отпускная цена) разработки административной панели веб-сайта.

## Описание проекта

Административная панель – это программа, которая помогает владельцам управлять содержимым сайта, не требуя от них при этом каких-либо особых знаний и навыков. Административная панель выполняет две основные функции: формируют страницы для пользователей и помогают владельцам управлять содержимым сайта.

Административная панель должна выполнять следующие функции:

* Раздел «галерея»
* Добавление новой категории.
* Редактирование существующей (добавление и удаление фотографий, назначение фотографии как обложкой категории).
* Раздел «фототуры»
* Добавление нового фототура.
* Редактирование существующего фототура (описание, маршрут и т.д.).
* Удаление фототура.
* Получение списка участников фототура.
* Получение списка заявок на фототур.
* Модерация заявок на фототура (подтверждение, отклонение).
* Раздел «об авторе»
* Редактирование информации об авторе.
* Раздел «контакты»
* Редактирование контактной информации (email и т.д.).
* Авторизация в административную панель.
* Смена пароля администратора.

Для работы над данным проектом необходимо арендовать два персональных компьютера, среда разработки WebStorm и графический редактор Adobe Photoshop.

## Разработка перечня работ по проектированию админ-панели

В данном разделе необходимо составить перечень работ по проектированию административной панели и рассчитать продолжительность и трудоемкость работ.

Для разработки административной панели требуется программист Junior, оклад которого составляет 11000 грн / мес (22 рабочих дня в месяц). Оклад дизайнера состоит 8 000 грн / мес (таблица 4.3.1).

Таблица 4.3.1 – Средние месячные, дневные и часовые заработные платы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Должность | Месячная ЗП, грн | Дневная ЗП, грн | Часовая ЗП, грн |
| 1 | Программист Junior | 11000 | 500 | 62,5 |
| 2 | Дизайнер | 8000 | 363,63 | 45,45 |

Таким образом, дневная заработная плата программиста Junior составляет 500 грн, а дизайнера – 363,63 грн.

Продолжительность отдельного этапа определяется таким соотношением:

где T – продолжительность этапа, определяется в рабочих днях, округляется до целого;

t – трудоемкость этапа (человеко-дни);

n – принятое количество исполнителей, занятых одновременно выполнением этапа.

Результаты расчета трудоемкости представлены в таблице 4.3.2

Таблица 4.3.2 – Перечень работ и трудоемкость по разработке административной панели.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Этапы | Трудоемкость, чел./дн | | Стоимость, грн |
| Программист | Дизайнер |
| 1 | Разработка и утверждение ТЗ | 3 | 3 | 2590,89 |
| 2 | Разработка и утверждение дизайна | 1 | 2 | 1227,26 |
| 3 | Разработка и утверждение wirefrime’ов панели | 1 | 4 | 1954,52 |
| 4 | Разработка и утверждение макетов панели | 1 | 10 | 4136,3 |
| 5 | Разработка и заполнение БД | 2 | - | 1000 |
| 6 | Написание клиентской части | 10 | - | 5000 |
| 7 | Написание серверной части | 5 | - | 2500 |
| 8 | Тестирование | 3 | - | 1500 |
| 9 | Написание документации | 5 |  | 2500 |
| Итого: | | 31 | 19 | 22408,97 |

Таким образом, согласно подсчетам на разработку системы требуется 31 день программисту и 19 дней дизайнеру.

Заработная плата (ЗП) программиста Junior и дизайнера состоит из основной и дополнительной заработных плат.

Фонд основной заработной платы определяется по формуле:

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

где , – трудоемкость работы программиста и дизайнера соответственно.

, – среднедневные заработные платы программиста и дизайнера соответственно.

Отсюда

|  |
| --- |
| грн |

Дополнительная заработная плата () – оплата отпусков, доплаты за дополнительный рабочий день, премии за достижения в работе, составляет 20% от основной заработной платы. Таким образом:

|  |
| --- |
| грн |

Общий фонд заработной платы составляет

|  |
| --- |
| грн |

Начисления на заработную плату в процентах от основной и дополнительной заработной плат (единый социальный налог) составляет 22%.

|  |  |
| --- | --- |
| грн |  |

## Расчет затрат и цены разработки административной панели

В таблице 4.4.1 представлены расходы на материалы.

Таблица 4.4.1 – Расходы на материалы.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Материал | Назначение | Количество, шт. | Цена за единицу, грн | Сумма, грн |
| 1 | ПО JetBrains WebStorm | Среда разработки | 1 | 330 | 330 |
| 2 | ПО Adobe Photoshop | Графический редактор | 1 | 215 | 215 |
| 3 | Флэшка Transcend JetFlash 16GB | Хранение ПО | 1 | 250 | 250 |
| 4 | Бумага офисная IQ economy А4 80г / м2., 500 л. | Печать документации, отчетов, пометок по разработке | 2 | 80 | 160 |
| 5 | Папка-регистратор одностор. PР 7,5 см, собранная | Хранение документации | 5 | 40 | 200 |
| 6 | Канцтовары в ассортименте | Для оформления документации | 2 | 200 | 400 |
| 7 | Прозрачные файлы EconoMix А4 100шт матовые 31107-50 | Для хранения документации | 2 | 82 | 162 |
| 8 | Степлер | Для оформления документации | 1 | 32 | 32 |
| 13 | Итого: | | | | 1749 |

Таким образом, расходы на материалы на проектирование административной панели составляют 1749 грн.

Для выполнения работ, связанных с разработкой административной панели, необходимо 2 персональных компьютера. Работы будут выполняться на арендуемых ПК, где стоимость машинного времени (СМВ) составляет 7 грн / час. Данная величина уже включает в себя амортизационные отчисления, также услуги печати.

где , – время работы программиста и дизайнера в часах соответственно.

– стоимость машинного времени за 1 час.

Поскольку рабочий день длится 8 часов, то:

|  |
| --- |
| грн |

Себестоимость разработки административной панели равна сумме всех вышеуказанных расходов:

Цена разработки рассчитывается по формуле

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

где П – плановая прибыль (25% от себе стоимости разработки);

грн

|  |
| --- |
| НДС – налог на добавленную стоимость (20% от себестоимости и плановой прибыли). |
|  |
| грн |
| Цена разработки без учета НДС:  грн  Цена разработки с учетом НДС: |
| грн |

Калькуляционные статьи на разработку административной панели представлены в таблице 4.4.2.

Таблица 4.4.2 – Статьи калькуляции на разработку.

|  |  |
| --- | --- |
| Статья калькуляции | Затраты, грн |
| Материалы | 1749 |
| Основная заработная плата |  |
| Дополнительная заработная плата |  |
| Фонд заработной платы |  |
| Начисления на заработную плату (единый и социальный налог) |  |
| Стоимость машинного времени |  |
| Себестоимость разработки административной панели |  |
| Плановая прибыль |  |
| Цена без НДС |  |
| НДС |  |
| Цена разработки |  |

## Выводы по разделу 5

Таким образом, себестоимость разработки 37355,72 грн, отпускная цена с учетом НДС разработки составляет 56033,58 грн (прибыль составляет 25% от себестоимости).

Для расчета данных показателей были учтены все необходимые статьи калькуляции, применимые к данному продукту, а именно: были учтены расходы на материалы, основные и дополнительные заработные платы программиста и дизайнера, начисления на заработную плату за счет единого социального фонда, стоимость машинного времени. Перечисленные статьи позволяют рассчитать себестоимость продукта, на основе которой возможны оценка и формирование прибыли для дальнейшей реализации продукта на рынке.

# СПИСОК ЛИТЕРАТУРЫ

1. Достоинства и недостатки популярных cms [Электронный ресурс]. Режим доступа – <http://zaurmag.ru/raznoe/dostoinstva-i-nedostatki-populyarny-h-cms.html>
2. Топ-5 наиболее популярных CMS: какую выбрать? [Электронный ресурс]. Режим доступа – <https://habrahabr.ru/post/151879/>
3. WYSIWYG [Электронный ресурс]. Режим доступа – <https://ru.wikipedia.org/wiki/WYSIWYG>
4. Что такое Wireframe? [Электронный ресурс]. Режим доступа – <https://medium.com/ux-crash-course/ux-wireframe-5a108e057324#.3znjwuasj>
5. REST [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/REST>
6. REST архитектура [Электронный ресурс]. – Режим доступа: <http://html-templates.info/blog/znakomtes-arhitektura-REST>
7. Архитектура REST [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/38730/>
8. Одностраничное приложение [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Одностраничное_приложение>
9. Что такое Gulp [Электронный ресурс]. – Режим доступа: <http://getinstance.info/articles/tools/introduction-to-gulp/>

# ПРИЛОЖЕНИЕ А: Техническое задание

1 Введение

1.1 Наименование, шифр работы и основания для выполнения работы.

Проводимой по настоящему техническому заданию работе присваивается наименование: «Панель администрирования веб сайта» (далее по тексту – административная панель).

1.2. Краткая характеристика области применения

Данная административная панель предназначена для управления контентом сайта.

2 Основания для разработки

2.1 Основание для проведения разработки

Основание для проведения разработки является задание на дипломный проект бакалавра кафедры «Компьютерные системы и сети» Национального аэрокосмического университета им. Н.Е. Жуковского «Харьковский авиационный институт» на тему «Разработка панели администрирования веб сайта».

2.2 Наименование и условное обозначение темы разработки

Наименование темы разработки – «Панель администрирования веб сайта».

Условное обозначение темы разработки (шифр темы) – «AdminPanel».

3. Назначение разработки.

3.1 Функциональное назначение разработки.

Основным назначением административной панели является управление контентом сайта.

Разрабатываемая административая панель предназначена для управления контентом сайта.

3.2 Эксплуатационное назначение.

Административная панель предназначена эксплуатации на ПК, установленном в офисе или в домашних условиях.

Для работы с административрной панелью необходимо наличие на компьютере подключение к сети Интернет и браузера последней версии.

4. Требования к программе или программному изделию.

4.1 Требования к функциональным характеристикам.

4.1.1 Требования к составу выполняемых функций.

В административной панели необходимо выполнять следующие функции:

* Раздел портфолио
* Добавление новой категории.
* Редактирование существующей (добавление и удаление фотографий, назначение фотографии как обложкой категории).
* Раздел публикации
* Добавление новой публикации.
* Удаление публикации.
* Редактирование публикации.
* Раздел фототуры
* Добавление нового фототура.
* Редактирование существующего фототура (описание, маршрут и т.д.).
* Удаление фототура.
* Получение списка участников фототура.
* Получение списка заявок на фототур.
* Модерация заявок на фототура (подтверждение, отклонение).
* Раздел контакты/обратная связь
* Редактирование контактной информации (номер телефона, email и т.д.).
* Авторизация в административную панель.
* Смена пароля администратора.

4.1.2 Требования к организации входных данных.

Данные в административной панели должны быть представлены в виде изображений и текстовых сообщений.

4.1.3 Требования к организации выходных данных.

Все данные выводятся в виде изображений и текстовых сообщений.

4.1.4 Требования к временным характеристикам.

Требования к временным характеристикам не предъявляются.

4.2 Требования к надежности.

Система должна сохранять работоспособность и обеспечивать восстановление своих функций при возникновении следующих внештатных ситуаций:

* при сбоях в системе электроснабжения аппаратной части, приводящих к перезагрузке ОС, восстановление программы должно происходить после перезапуска ОС и запуска исполняемого файла системы;
* при ошибках в работе аппаратных средств (кроме носителей данных и программ) восстановление функции системы возлагается на ОС;
* при ошибках, связанных с программным обеспечением (ОС и драйверы устройств), восстановление работоспособности возлагается на ОС.

Для защиты аппаратуры от бросков напряжения и коммутационных помех должны применяться сетевые фильтры.

4.2.1 Требования к обеспечению устойчивого функционирования программы.

Устойчивое функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. организацией бесперебойного питания технических средств;
2. использованием лицензионного программного обеспечения;
3. регулярным выполнением требований нормативных документов, касающихся защиты информации путем испытания программных средств на наличие компьютерных вирусов.

4.2.2 Отказы из-за некорректных действий оператора.

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4.3 Условия эксплуатации.

4.3.1 Климатические условия эксплуатации.

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики программы, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Административная панель предназначена для эксплуатации на ПК установленных в офисных помещениях и домашних условиях.

4.3.2 Требования к видам обслуживания.

Для работы с административной панелью необходимо наличие на компьютере браузера и подключение к сети Интернет.

4.3.3 Требования к численности и квалификации персонала.

Для эксплуатации административной панель определены следующие роли:

* Системный администратор;
* Администратор веб сайта.

Основными обязанностями системного администратора являются:

* Модернизация, настройка и мониторинг работоспособности комплекса технических средств (серверов, рабочих станций);
* Установка, модернизация, настройка и мониторинг работоспособности системного и базового программного обеспечения;
* Установка, настройка и мониторинг прикладного программного обеспечения;
* Ведение учетных записей пользователей системы.

Системный администратор должен обладать высоким уровнем квалификации и практическим опытом выполнения работ по установке, настройке и администрированию программных и технических средств, применяемых в системе.

Основными обязанностями администратора веб сайта являются:

* Добавление, редактирование и удаление материалов сайта;
* Модерация заявок пользователь;

Администратор веб сайта должен практическим опытом выполнения работ в CMS системах.

Рекомендуемая численность для эксплуатации административной панели кадры:

* Системный администратор – 1 штатная единица;
* Администратор веб сайта – 1 штатная единица.

4.4 Требования к составу и параметрам технических средств.

Техническое обеспечение системы должно максимально использовать технические средства.

Требования к техническим характеристикам ПК системного администратора и администратора веб сайта:

* процессор с тактовой частотой не менее 1.5 ГГц;
* объем оперативной памяти – не менее 4Гб;
* свободного дискового пространства не менее 100 Гб;
* устройство чтения компакт-дисков (DVD-ROM);
* сетевой адаптер – 100 Мбит.
* клавиатуру;
* манипулятор «мышь».

4.5 Требования к информационной и программной совместимости

4.5.1 Требования к информационным структурам и методам решения

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

4.5.2 Требования к исходным кодам и языкам программирования

При разработке административной панели должен использоваться языки Node.js и javascript.

4.5.3 Требования к программным средствам, используемым программой

Административная панель должна быть реализована в виде веб-сайта, способного выполняться под управлением ОС Windows 7 в браузере.

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы. Допускается использование соответствующего пакета обновления.

Параметры ОС настроены таким образом, чтобы исполнение программы было возможным. Для этого, как минимум, должны быть правильно настроены переменные окружения среды выполнения программы и корректно установлены права пользователя.

4.5.4 Требования к защите информации и программ

Защита от влияния внешних воздействий должна обеспечиваться средствами программно-технического комплекса Заказчика.

4.6 Специальные требования

Работа в административной панели должна осуществляться при помощи браузера через интерфейс веб-сайта.

5 Требования к маркировке и упаковке.

5.1 Требование к маркировке

Диск, с записанной на него административной панелью, должен иметь маркировку с обозначением типа (наименования), номера версии, порядкового номера, даты разработки.

Маркировка должна быть нанесена на диск в виде наклейки выполненной полиграфическим способом с учетом требований ГОСТ 9181-74.

5.2 Требования к упаковке

Упаковка диска с административной панелью должна осуществляться в упаковочную тару предприятия-изготовителя, представляющую собой жёсткий футляр для DVD-дисков.

Упаковка программного изделия должна проводиться в закрытых вентилируемых помещениях при температуре от +15 до +40 °С и относительной влажности не более 80% при отсутствии агрессивных примесей в окружающей среде.

5.3 Требование к транспортированию и хранению

Административная панель поставляется на DVD-диске, который должен быть помещен в жёсткий футляр, обеспечивающий длительное хранение в складских отапливаемых помещениях в условиях пригодных для хранения оптических дисков в течение 5 лет.

Допускается транспортирование программного изделия в транспортной таре всеми видами транспорта (в том числе в отапливаемых герметизированных отсеках самолетов без ограничения расстояний). При перевозке в железнодорожных вагонах вид отправки – мелкий малотоннажный.

При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания пыли и атмосферных осадков. Климатические условия транспортирования:

* температура окружающего воздуха, °С - от +5 до +50;
* атмосферное давление, кПа - такое-то;
* относительная влажность воздуха при +25 °С - такая-то.

6 Требования к программной документации

6.1 Предварительный состав программной документации

Состав программной документации должен включать:

* техническое задание;
* пояснительная записка;

7 Стадии и этапы разрабокти.

7.1 Стадии разработки

Разработка должна проводиться по следующим стадиям:

* разработка технического задания;
* рабочее проектирование;

7.2 Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочее проектирование должны быть выполнены перечисленные ниже виды работ:

1. Разработка базы данных для хранения материалов сайта;
2. Разработка серверной части приложения;
3. Разработка клиентской части приложения;
4. Интеграция клиентской и серверной части;