МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ
(наименование)	(наименование)
Направление подготовки	09.03.04 «Программная инженерия»
	(код и наименование)

БАКАЛАВРСКАЯ РАБОТА

на тему

«ВЕБ-ПРИЛОЖЕНИЕ БРОНИРОВАНИЯ ЭКСКУРСИЙ И РАЗВЛЕКАТЕЛЬНЫХ МЕРОПРИЯТИЙ»

Студент	_____ (подпись, дата)	Васюнина Милана Евгеньевна (фио полностью)
Руководитель	_____ (подпись, дата)	Кольчугина Е.А. (фамилия, инициалы)
Нормоконтролёр	_____ (подпись, дата)	Такташкин Д.В. (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____ (подпись, дата)	Макарычев П.П. (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № ____)

Секретарь ГЭК	_____ (подпись, дата)	Попова Н.А. (фамилия, инициалы)

Пенза, 2020

Федеральное государственное бюджетное образовательное учреждение высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__»_____2020 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Васюнина Милана Евгеньевна гр. 16ВП1 факультета ВТ

2.Тема работы:
«Веб-приложение бронирования экскурсий и развлекательных мероприятий»

Тема утверждена приказом ПГУ №_____от «_____»_____20_____

3. Руководитель работы Кольчугина Е.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.):

Назначение:

- формирование интерфейса, структуры и содержимого сайта
- предоставление сервиса бронирования экскурсий и развлекательных мероприятий

Цель создания веб-приложения:

предоставление возможности бронирования билетов на экскурсии и мероприятия потенциальным посетителям, а также информирование о деятельности организации предоставляющей данные услуги

Основные функции веб-приложения:

а) информирование о экскурсиях и мероприятиях

б) бронирование билетов

в) предоставление информации об организаторе экскурсий и мероприятий

Структуры данных: мероприятия, места проведения, записи о бронировании, страницы.

Технология разработки: ООП, MVC

Языки программирования: PHP, SQL, JavaScript, HTML5, CCS3

Среда исполнения: веб-сервер Apache, интерпретатор PHP, СУБД MySQL

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области

2) Анализ функциональных требований

3) Планирование разработки и оценка бюджета

4) Проектирование веб-приложения

5) Реализация

6) Контроль качества

- тестирование

- метрики кода

7) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	15	23.11.2019 - 17.12.2019	
2	Анализ функциональных требований	10	21.12.2019 - 24.01.2020	
3	Планирование разработки и оценка бюджета	10	07.02.2020 - 25.02.2020	
4	Проектирование веб-приложения	20	01.03.2020 - 30.03.2020	
5	Реализация	30	02.04.2020 - 14.05.2020	
6	Контроль качества	5	16.05.2020 - 23.05.2020	
7	Оформление пояснительной записки	10	25.05.2020 - 15.06.2020	

Дата выдачи задания «_____»_____20_____

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____

(подпись, дата)

Реферат

Пояснительная записка содержит 158 листов, 26 рисунков, 22 таблицы, 54 источника, 14 приложений.

АНАЛИЗ, ПЛАНИРОВАНИЕ, ОЦЕНКА БЮДЖЕТА, ПРОЕКТИРОВАНИЕ, UML, РАЗРАБОТКА, ТЕСТИРОВАНИЕ, ВЕБ-ПРИЛОЖЕНИЕ, СИСТЕМА БРОНИРОВАНИЯ, CMS, ЭКСКУРСИИ, МЕРОПРИЯТИЯ.

Объект исследования – автоматизированные информационные системы.

Цель работы – проектирование и разработка веб-приложения система бронирования экскурсий и развлекательных мероприятий.

Технология разработки: ООП, MVC, Apache, PHP, Symfony, PHPunit, JavaScript, Vue.js.

В результате выполнения работы было спроектировано и создано веб-приложение система бронирования экскурсий и развлекательных мероприятий. Отличительной особенностью системы является специализация на бронировании экскурсий и мероприятий, а также возможность создания сайта заведения.

Основные технико-эксплуатационные характеристики:

- универсальность и возможность быстрой модификации;
- широкие функциональные возможности;
- программная совместимость с многими операционными системами.

Данный программный продукт предназначен для создания и ведения сайта предоставляющего возможность бронирования экскурсий и развлекательных мероприятий. Веб-приложение может применяться в экскурсионных и туристических фирмах.

Содержание

Стр.

Введение 7
1 Анализ предметной области 10
1.1 Описание предметной области 10
1.2 Обзор научных исследований о системах бронирования 12
1.3 Обор сервисов и систем бронирования мероприятий 13
1.4 Сравнительный анализ аналогов 18
1.5 Анализ вариантов использования 23
1.6 Постановка задачи 29
1.7 Техническое задание 32
1.7.1 Функциональные требования 32
1.7.2 Необязательные функциональные требования 33
1.7.3 Требования к графическому дизайну сайта 33
1.7.4 Требования к разделению доступа 33
1.7.5 Требования к хранению данных 34
1.7.6 Требования к языкам программирования и разметки 34
1.7.7 Требования к иллюстрациям 35
1.7.8 Требования к программному обеспечению серверной части 35
1.7.9 Требования к техническому обеспечению 35
1.7.10 Требования к клиентскому программному обеспечению 35
1.7.11 Требования к лингвистическому обеспечению 36
1.7.12 Требования к эргономике и технической эстетике 36
1.7.13 Требования к персоналу 36
2 Планирование разработки и оценка бюджета 37
2.1 Календарное и ресурсное планирование 37
2.2 Расчет бюджета 40
2.3 Выбор программно-аппаратных средств 45
3 Проектирование системы бронирования 50
3.1 Проектирование пользовательского интерфейса 50

3.2 Организация процесса обработки информации 53
3.3 Структуры хранения данных 56
4 Реализация системы бронирования 64
4.1 Реализация системы бронирования 64
4.2 Развёртывание веб-приложения 74
5 Анализ качества программного продукта 76
5.1 Тестирование 76
5.2 Метрики кода 83
Заключение 86
Список использованных источников 90
Приложение А. Распределение трудовых ресурсов 95
Приложение Б. Диаграмма Ганта 98
Приложение В. Листинг кода конфигурационного файла htaccess 100
Приложение Г. Листинг кода файла index.php 102
Приложение Д. Листинг кода класса Application 104
Приложение Е. Листинг кода класса маршрутизатора Router 107
Приложение Ж. Листинг кода маршрутов веб-приложения 111
Приложение И. Листинг кода класса Booking 114
Приложение К. Листинг кода класса User 120
Приложение Л. Листинг кода класса ObjectModel 125
Приложение М. Листинг кода классов моделей Eloquent 134
Приложение Н. Листинг кода unit-тестов модуля контроля доступа 139
Приложение П. Листинг кода unit-тестов модуля бронирования 143
Приложение Р. Тестирование веб-приложения 150

Введение

Последнее время в России и в мире стремительно развивается глобальная информационно-телекоммуникационная сеть «Интернет» и значительный процент населения имеет возможность доступа к Интернет-ресурсам. Быстрое развитие информационных технологий и компьютеров распространилось почти на каждую сферу деятельности, в том числе на отрасли сферы обслуживания, бухгалтерский учет, банковское дело и розничную торговлю, а также сферу досуга и развлечений.

Еще совсем недавно, в середине-конце 90-х годов 20 века у простого человека, желающего выбрать, как провести свободное время посетить концерт, записаться на экскурсию или посетить какое-либо мероприятие, практически не было другого выбора, как искать эту информацию в газетах или на афишах. Другой вариант – взять справочник и обзвонить театры, кинотеатры и другие подобные заведения самостоятельно. Но и тут появляются свои проблемы – зачастую из-за наплыва звонков, желающих получить подобную информацию телефонные номера были заняты, а те, кто все-таки дозванивался, тратили на это большое количество времени

В настоящее время наиболее популярным и удобным способом получения информации о предстоящих экскурсиях, мероприятиях, киносеансах является поиск информации в сети Интернет. Обычно, пользователь заходит на сайт интересующего его заведения, предоставляющей услуги бронирования, подбирает подходящее мероприятие, дату и место и бронирует.

Естественно, это огромные объемы, постоянно меняющейся информации. Как правило, для этих целей используются различные автоматизированные информационные системы.

Одним из видов таких систем являются системы управления содержимым (CMS – content management system). Современные системы управления сайтом представляют собой крупные программные комплексы, позволяющие обеспечивать всестороннюю поддержку Интернет-ресурса. Но

именно чрезмерная «развитость» некоторых систем, которую скорее следовало бы назвать перегруженностью или нагроможденностью, является их самым большим недостатком. Проблема заключается в том, что в процессе развития была забыта основная функция этих систем – помощь в обновлении содержания сайта. Кроме того, на сегодняшний день, наверное, нет ни одной достаточно функциональной и надежной CMS, предназначенной непосредственно для работы в сфере бронирования экскурсий и мероприятий, что делает их еще более неудобными для конечных пользователей. Подавляющее большинство систем управления содержимым обладающих необходимой функциональностью являются платными и имеют фиксированный набор компонентов.

Из всего вышесказанного, очевидно, что существует необходимость в Интернет-ресурсах, предоставляющих функции бронирования, оснащенных системой управления содержимым оптимизированной непосредственно под эту сферу. Говоря о специфике проектирования подобной системы, стоит отдельно упомянуть об удобстве пользовательского взаимодействия с интерфейсом таких систем. Судя по проведенным исследованиям, удобство использования современных систем управления сайтом, в силу своей нагроможденности и универсальности, не выдерживает никакой критики. Для удобного и интуитивно понятного интерфейса необходимо определить возможные варианты использования веб-приложения, на основе них определить функциональные требования и спроектировать систему таким образом, чтобы в веб-приложение входили только соответствующие требованиям функции.

Объект исследования – автоматизированные информационные системы.

Предмет – система бронирования экскурсий и развлекательных мероприятий.

Целью работы является создание веб-приложения система бронирования экскурсий и развлекательных мероприятий.

Для достижения поставленной цели были определены следующие задачи:

– проанализировать предметную область;

– проанализировать степень разработанности проблемы отечественными и зарубежными авторами;

– провести анализ существующих систем бронирования;

– составить требования и техническое задание к веб-приложению система бронирования экскурсий и мероприятий;

– провести планирование и расчёт бюджета разрабатываемого веб-приложения;

– изучить научную литературу по проектированию и разработке веб-приложений;

– осуществить проектирование пользовательского интерфейса, программных средств и структур данных разрабатываемой системы на основе функциональных требований;

– реализовать веб-приложение система бронирования экскурсий и развлекательных мероприятий;

– провести тестирование разработанного программного продукта.

1 Анализ предметной области

1.1 Описание предметной области

Для того чтобы определить протекающие бизнес-процессы в системе бронирования необходимо провести анализ предметной области. Предметом данного исследования является система бронирования экскурсий и развлекательных мероприятий.

В настоящее время наиболее удобным и популярным способом получения информации о предстоящих мероприятиях является просмотр сайтов в сети Интернет. Обычно пользователь заходит на сайт, предоставляющей услуги бронирования, подбирает подходящее мероприятие, дату и место, а затем бронирует билет.

Для реализации описанного процесса необходимо обладать информационной системой, которая предоставляет пользователям Интернета возможности просмотра расписания, мероприятий и списка доступных мест, а организаторам мероприятия – интерфейс для просмотра и обработки. Как правило, такая информационная система реализуется в виде сайта с пользовательским и административным разделами. Существует множество типовых решений для разработки таких сайтов, и единственной организационной трудностью является необходимость отслеживания актуальности и оперативного обновления информации, связанной с наличием свободных мест, расписанием мероприятий и другими меняющимися обстоятельствами [1].

Часто для решения этой проблемы поручают обновление данных IT-отделу или отдельным наёмным работникам. В этом случае существенно возрастает стоимость поддержки проекта, так как необходимо оплачивать труд большого количества высококвалифицированных IT-специалистов.

Альтернативой этому способу является разработка системы управления контентом (Content Management System, CMS), позволяющей операторам корректировать наполнение информационной системы в визуальном

режиме [1, 2]. В этом случае обязанности строго разделяются между программистами и операторами: программистам нет необходимости выполнять объёмную и рутинную работу, а операторам не нужно осваивать технологии веб-разработки, а достаточно иметь базовые навыки работы с визуальным редактором. Дополнительными преимуществами этого подхода является независимость данных и дизайна, низкая стоимость владения информационной системой, возможность дистанционного управления сайтом с использованием одного веб-браузера.

Кроме того, CMS может использовать расширения, предназначенные для управления объектами, специфическими для конкретной задачи (концертного зала, кинотеатра, автобусной экскурсии) и для выполнения различных дополнительных функций.

Рассмотрим особенности бронирования экскурсий. Любое мероприятие обладает следующими характеристиками:

– количество мест;

– минимальное количество забронированных билетов, чтобы мероприятие состоялось;

– дата и время;

– описание.

Реализация билетов на экскурсию может осуществляться как с помощью обслуживания в режиме живой очереди, так и с помощью предварительного бронирования (по телефону или самостоятельно на сайте).

Каждое мероприятие может находиться в нескольких состояниях:

– доступно для бронирования;

– недоступно для бронирования;

– мест нет.

Для бронирования билетов клиент должен предоставить следующую информацию оператору или на сайт:

– название мероприятия;

– дата проведения;

- время проведения;
- количество билетов;
- место проведения мероприятия;
- фамилия (или иное кодовое слово).

Проанализировав основные бизнес-процессы проведем поиск информации о разработке и использовании систем бронирования в научных источниках.

1.2 Обзор научных исследований о системах бронирования

Проведем поиск методических указаний по разработке и использованию систем бронирования, а также определим степень освещения проблемы отечественными и зарубежными авторами. Для этого проведем анализ статей о создании и применении систем бронирования, находящихся в базе Российской научной электронной библиотеки [3].

Так авторы М.С. Чековитов, Е.А. Шеленок в своей статье «Информационная система для сервиса бронирования и покупки туристических путевок по Хабаровскому краю» описывают общую архитектуру информационной системы [4]. Н.А. Борсук в статье разработка модуля администрирования системы on-line бронирования опубликованной в международном научном журнале «Инновационная наука» №5/2016 описывает один из этапов разработки системы бронирования [5].

К.А. Бунина в статье «Анализ современных компьютерных систем бронирования и резервирования» рассматривает популярные зарубежные и отечественные глобальные системы бронирования и резервирования, описывает возможности их применения [6]. В.О. Алексанина в своей статье «Автоматизированные системы бронирования и резервирования» описывает применение глобальных систем бронирования в туристическом бизнесе и авиаперевозках [7]. А.М. Сарсебаева из Казахстана в статье «Система бронирования и резервирования в гостиницах и ресторанах» проводит обзор внедрения систем бронирования в туристическом и ресторанном бизнесе [8].

Группа авторов из Украины в лице М.В. Куклинського, Г.В. Головня и М.О. Лукаш в своей статье «Сетевое приложение к системе предварительного бронирования авиабилетов» описывают программную архитектуру системы бронирования авиабилетов [9]. Т.С. Шаймерденов из Казахстана в статье «Инновации в системе бронирования для развития туризма в РК на примере системы бронирования «AMADEUS» описывает применение системы «AMADEUS» в индустрии туризма Казахстана [10].

Проанализировав наиболее важные работы отечественных и зарубежных исследователей можно сделать вывод, что тема разработки системы бронирования развлекательных мероприятий освещена мало. В большей части работ приводится общее описание назначения систем бронирование и преимущества их применение в туристическом бизнесе и авиаперевозках. В статьях, описывающих реализацию систем бронирования, тема раскрыта мало, освещены только отдельные фрагменты системы, комплексный подход отсутствует. В большинстве систем используются устаревшие технологии. Также стоит отметить отсутствие методических указаний по разработке таких систем, нет сравнительного анализа инструментов и способов реализации веб-приложений систем бронирования.

1.3 Обор сервисов и систем бронирования мероприятий

Прежде чем приступить к разработке требований необходимо определить и изучить уже известные подобные программные продукты, для того, чтобы оценить необходимые характеристики подобных систем и учесть недостатки существующих систем. Проанализировав рынок программных продуктов для бронирования экскурсий и других развлекательных мероприятий можно выделить три направления:

- сайты агрегаторы экскурсий и мероприятий;
- SaaS-сервисы, предоставляющие услуги онлайн-записи и бронирования;
- CMS и CRM для бронирования.

Сайты-агрегаторы экскурсий и туров, такие как «Спутник» [11], «Weatlas» [12], «Tripster» и «Excursio» предоставляют возможность размещать экскурсии на своем общем сайте, регистрация доступна только для профессиональных гидов [13, 14]. Поскольку данные сервисы не позволяют создавать собственную систему бронирования отдельную от общего сайта с собственной структурой, то они не являются программами аналогами.

Рассмотрим SaaS-сервисы онлайн-записи бронирования.

«Bukza.ru» – представляет собой виджет для интеграции на сайт и интерфейс управляющего на сайте «Bukza». Основные преимущества: возможность стилизации виджета, адаптация под разные размеры экрана, настройка цен, уведомления по email, возможность создания нескольких аккаунтов сотрудников с гибкой настройкой доступа. Недостатки: доступ к большинству функций только в платном тарифе, нет возможности создания сайта [15].

«SimplyBook.me» – это система онлайн-записи для различных отраслей сферы услуг. Преимущества: настраиваемый сайт бронирования или виджеты на выбор, уведомления по email/sms. Недостатки: ограничения на количество бронирований в месяц на всех тарифах, высокая стоимость платных тарифных планов [16].

«Bookio.me» – это интегрируемый модуль онлайн-записи для сайтов. Преимущества: конструктор модулей онлайн-записи, уведомления через sms/email/мессенджеры. Недостатки: нет бесплатного тарифа, зарубежная система [17].

«M-BRON Module System» – модуль-сервис онлайн бронирования на сайт со встроенным CRM [18]. Преимущества: возможность изменения текста и дизайна модуля, уведомления, разграничения прав доступа. Недостатки: ограниченный размер файлового хранилища для иллюстраций на всех тарифах, высокие комиссии на всех тарифах, платные тарифные планы.

«Booking-widget» – виджет бронирования на сайт. Преимущества: создание гибкого расписания, информирование по email и sms, полная кастомизация. Недостатки: бесплатный тариф имеет большие ограничения, отсутствует возможность создания сайта, ограниченный функционал панели управления [19].

«Атом-с» – облачная система для автоматизации экскурсионного туризма. Преимущества: присутствует возможность создания сайта, личный кабинет для туриста и агентств. Недостатки: возможность создания сайта платная и оплачивается отдельно от базового тарифа [20].

Автоматизированные системы управления содержимым сайта с системой бронирования:

– зарубежная система управления содержимым сайта «Booking Core»;
– система бронирования автобусных туров «Ezbus»;
– система управления контентом «Ag Booking CMS».

Рассмотрим подробнее системы управления содержимым.

Начнем с системы управления содержимым сайтом с системой бронирования «Booking Core». Главная страница сайта созданного с помощью системы «Booking Core» представлена на рисунке 1.

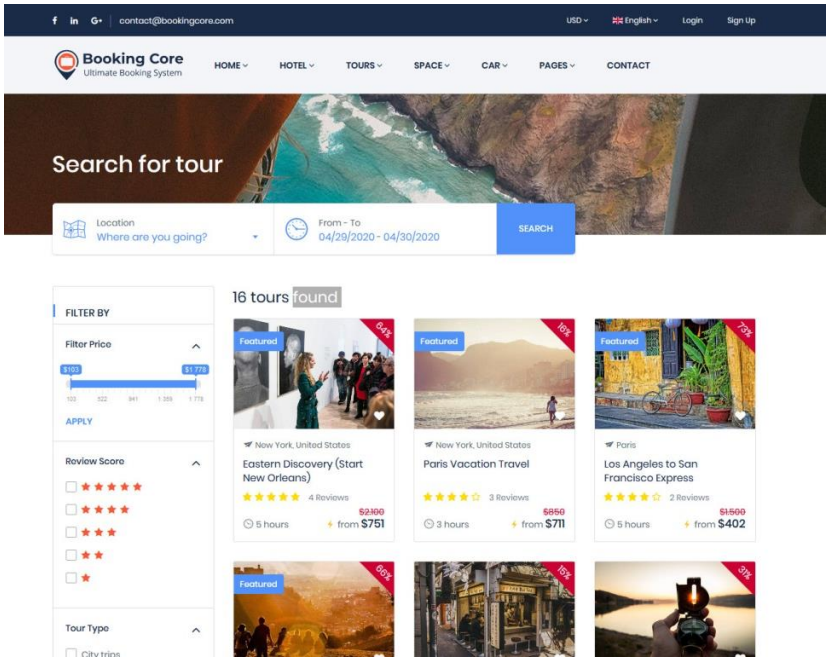

Рисунок 1 – Страница выбора мероприятий CMS «Booking Core»

Данная система является зарубежным программным продуктом, разработанным на основе фреймворка Laravel версии 5.8 [21]. Преимуществом системы является возможность создания сайта и администрирования его, встроенная система бронирования, система контроля доступа с возможностью гибкого изменения прав. Присутствует возможность добавления новостей. К недостаткам системы можно отнести невозможность изменения формата системы бронирования, отсутствие русской локализации, ограниченный функционал шаблонов страниц и англоязычная поддержка.

Далее рассмотрим отечественную систему бронирования автобусных туров «Ezbus» [22]. Страница доступных для бронирования мероприятий представлена на рисунке 2. К преимуществам системы можно отнести простой и интуитивно-понятный интерфейс, возможность бронирования заказчиком на сайте и через оператора сайта. Присутствует возможность создания сайта. Основным недостаткам системы можно считать отсутствие возможности изменения системы бронирования под другой формат, т.е. для бронирования экскурсий и мероприятий. Также минусом системы можно назвать устаревший дизайн сайта, создаваемого системой.

Рисунок 2 – Страница системы «Ezbus»

Последним рассмотрим зарубежную систему управления содержимым сайта «Ag Booking CMS». Данный продукт представляет собой систему управления контентом и встроенную в неё, универсальную систему бронирования времени [23]. Страницу выбора мероприятий для бронирования можно увидеть на рисунке 3. С помощью данного решения, можно развернуть сайт с онлайн записью, заказом услуг (предоставляемых по расписанию), бронированию чего-либо по времени и т.п. К преимуществам можно отнести современный внешний вид, возможность изменения оформления страниц сайта. Одним из основных недостатков можно назвать хранение всех данных сайта в файлах без использования СУБД, что приводит к ошибкам в работе при большом количестве посетителей, а также сказывается на быстродействии сайта. Также недостатком системы является малый функционал системы управления.

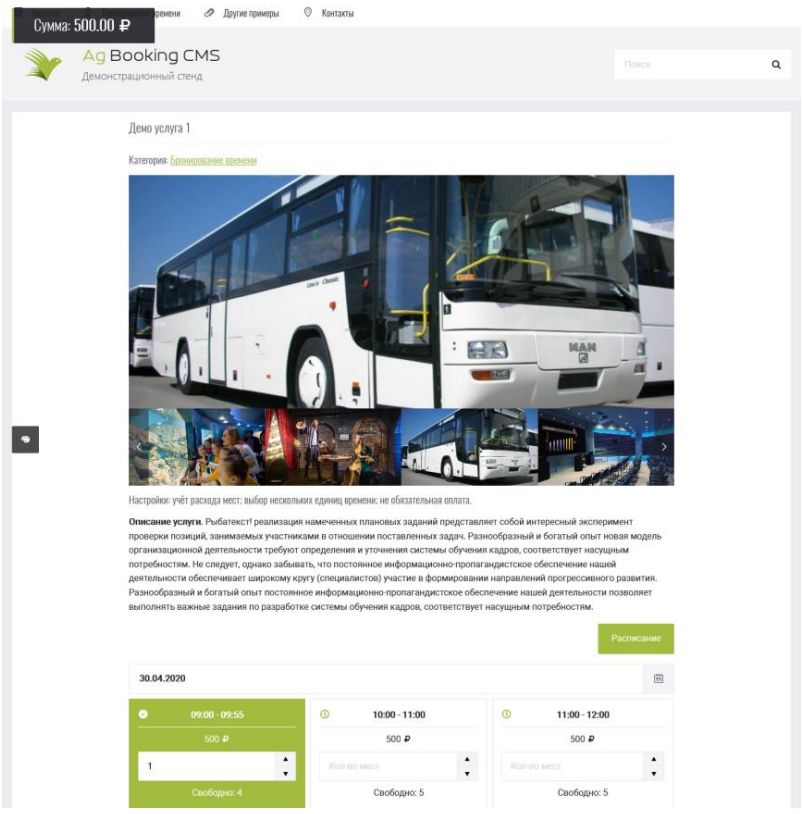

Рисунок 3 – Страница мероприятий под управлением «Ag Booking CMS»

Рассмотрев основные системы управления содержимым сайта с функцией бронирования можно перейти к сравнению данных систем.

1.4 Сравнительный анализ аналогов

Проведем сравнительный анализ систем управления содержимым сайта с функцией бронирования.

Для проведения сравнительного анализа программных продуктов были выбраны следующие критерии:

а) функциональные возможности:

- бронирование мероприятий и отмена;
- управление мероприятиями и их свойствами;
- наличие у мероприятий свойств: количество билетов, дата и время, описание, изображение;
- контроль и управление доступом к функциям системы;

б) необязательные функциональные возможности:

- управление страницами сайта;
- управление структурой страниц сайта;
- управление структурой навигации сайта;
- управление пользователями;
- управление новостями;
- управление медиафайлами;

в) технические возможности:

- хранение данных в СУБД;
- возможность внесения изменений;

г) эргономика:

- адаптивность к размерам экрана устройств;
- кроссбраузерность;
- интуитивно-понятный интерфейс;
- интерфейс сайта и панели управления на русском языке.

Исследовав бизнес-процессы, возникающие во время использования систем бронирования, были составлены коэффициенты значимости критериев, представленные в таблице 1.

Таблица 1 – Критерии оценки систем бронирования

№	Наименование	Коэффициент значимости
	Функциональные возможности	
1	Бронирование мероприятий и отмена	1
2	Управление мероприятиями и их свойствами	0,75
3	Наличие у мероприятий свойств: количество билетов, дата и время, описание, изображение	0,75
4	Контроль и управление доступом к функциям системы	0,5
	Необязательные функциональные возможности	
5	Управление страницами сайта	0,5
6	Управление структурой страниц сайта	0,5
7	Управление структурой навигации сайта	0,5
8	Управление пользователями	0,5
9	Управление новостями	0,25
10	Управление медиафайлами	0,25
	Технические возможности	
11	Адаптивность	0,35
12	Хранение данных в СУБД	0,35
13	Кроссбраузерность	0,5
14	Возможность внесения изменений	0,75
	Эргономика	
15	Интуитивно-понятный интерфейс	0,5
16	Интерфейс сайта и панели управления на русском языке	0,5

Для оценки описанных выше критериев используется пятибалльная шкала соответствия, от 0 – «полное несоответствие», до 4 – «полное соответствие» требованиям. В таблице 2 представлены характеристики оценки.

Таблица 2 – Шкала баллов

Балл	**Характеристика оценки**
0	функция (качество) отсутствует, реализовать ее невозможно
1	функция (качество) отсутствует, но ее возможно реализовать путем доработки системы
2	похожие, но отличные от требуемых, функции или качества системы находятся в стадии разработки
3	реализованы похожие, но несколько отличные от требуемых, функции или качества системы
4	функция или качество безоговорочно реализовано

Для анализа были отобраны следующие сервисы и системы бронирования мероприятий:

1) «Bukza»;

2) «SimplyBook.me»;

3) «Bookio.me»;

4) «M-BRON Module System»;

5) «Booking-widget»;

6) «Атом-с»;

7) зарубежная система управления содержимым сайта «Booking Core»;

8) система бронирования автобусных туров «Ezbus»;

9) система управления контентом «Ag Booking CMS».

Заполним таблицу 3, оценив реализованный функционал описанных выше систем. Затем произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле (1):

$$O_{\text{пп}} = \sum_{i=1}^{n}(Z_i \cdot K_i), \quad (1)$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_{i} – значение выполнения критерия (1 - выполнен, 0 - не выполнен);

K_i – коэффициент значимости критерия сравнения (от 0 до 1).

Таблица 3 – Критериальная оценка программных продуктов

№	Критерий	Программный продукт №								
		1	2	3	4	5	6	7	8	9
1	Бронирование мероприятий	4	4	4	4	3	4	4	4	4
2	Управление мероприятиями	4	4	4	4	3	2	3	3	4
3	Свойства мероприятий	3	3	3	3	3	2	3	3	3
4	Контроль доступа	3	2	2	2	2	3	4	2	2
5	Управление страницами сайта	0	2	0	0	0	3	3	3	2
6	Управление структурой страниц	0	1	0	0	0	4	3	3	2
7	Управление структурой навигации	0	2	0	0	0	3	3	3	2
8	Управление пользователями	0	3	0	0	0	3	4	1	2
9	Управление новостями	0	1	0	0	0	0	0	1	2
10	Управление медиафайлами	0	0	0	0	0	1	1	1	2
11	Хранение данных в СУБД	4	4	3	4	3	3	4	3	0
12	Возможность внесения изменений	0	0	0	0	0	0	3	0	3
13	Адаптивность	4	3	1	3	3	2	4	0	4
14	Кроссбраузерность	4	4	3	3	3	3	4	2	3
15	Интуитивно-понятный интерфейс	3	3	3	3	3	3	3	2	2
16	Интерфейс сайта и панели управления на русском языке	4	4	4	4	4	4	1	4	4
	Интегральная оценка	**19,05**	**21,85**	**16,65**	**17,7**	**15,6**	**19,7**	**24,35**	**18,55**	**22,2**

Проведя анализ и подсчитав общую интегральную оценку, а также оценку для критериальных групп полученные данные можно визуализировать с помощью лепестковой диаграммы, представленной на рисунке 4.

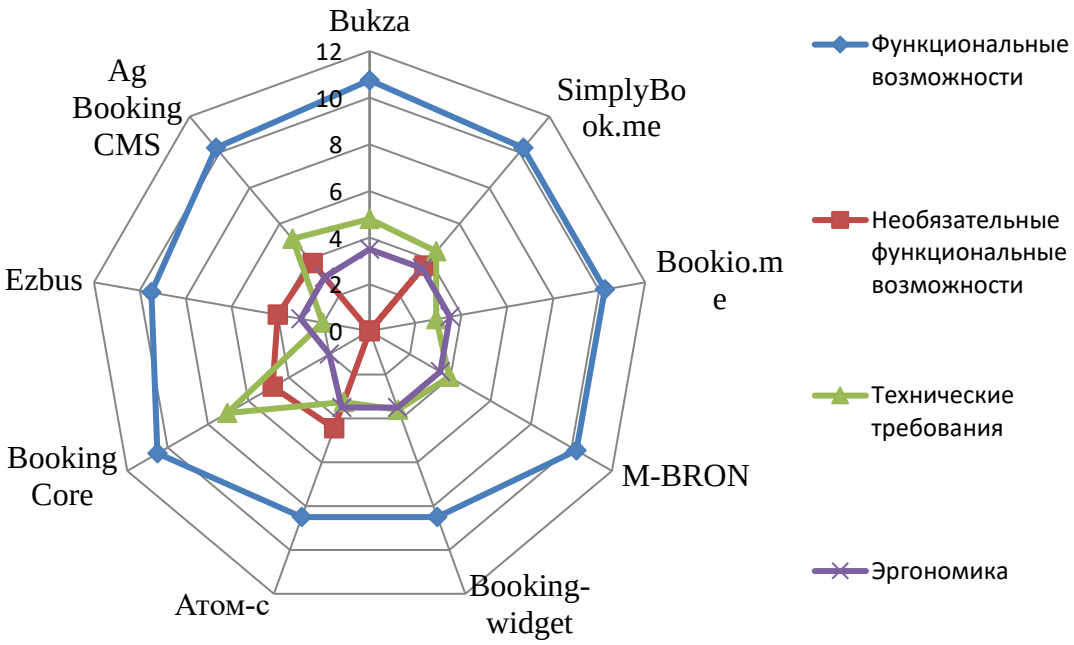

Рисунок 4 – Лепестковая диаграмма значений функциональных характеристик

Рассмотрев диаграмму значений функциональных характеристик можно сделать вывод, что наиболее функциональными системами управления содержимым сайта системы бронирования являются «Booking Core» и «Ag Booking CMS, а наиболее функциональными сервисами можно назвать «Bukza» и SimplyBook.me».

Подводя итог сравнения можно сделать вывод, что лидером является зарубежная система управления содержимым сайта системы бронирования «Booking Core» (рисунок 5).

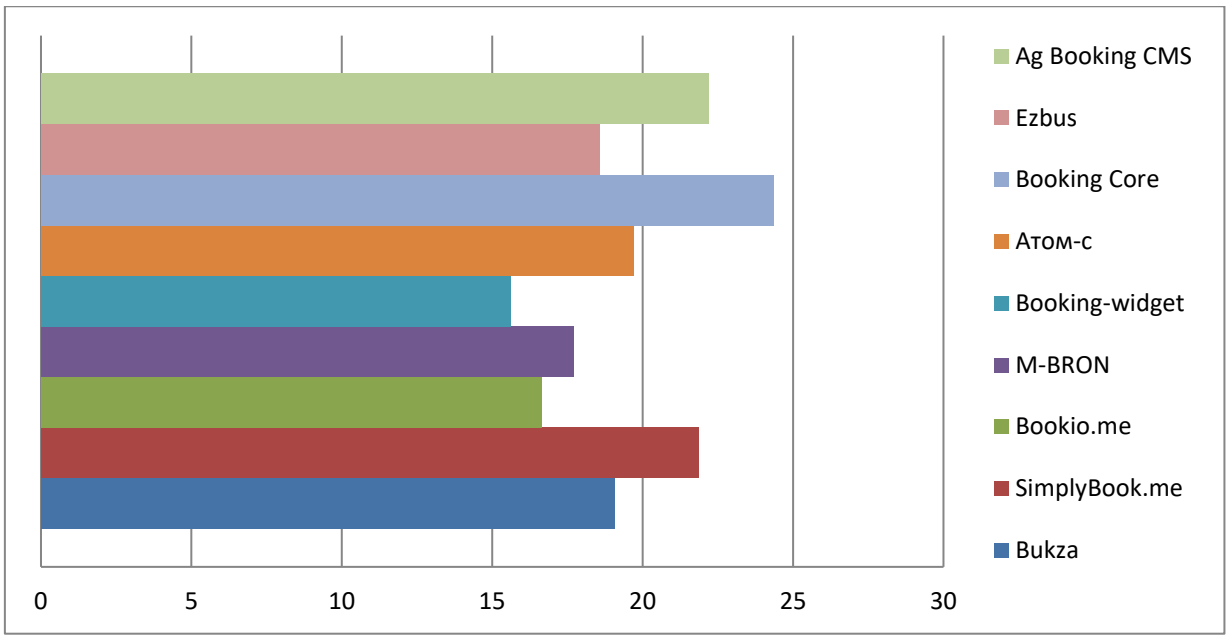

Рисунок 5 – Диаграмма интегральной оценки

Но она имеет свои недостатки, в особенности отсутствует русский язык в интерфейсе и имеет достаточно высокую стоимость. На втором месте идет «Ag Booking CMS» (смотри рисунок 5). Затем с небольшим отрывом следуют сервисы онлайн-записи.

1.5 Анализ вариантов использования

Проанализировав особенности бронирования билетов на экскурсию, можно прейти к вариантам использования автоматизированной информационной системы обслуживающей предполагаемых пользователей.

Информационные процессы, которые возникают во время работы с сайтом системы бронирования достаточно простые и сводятся к небольшому набору основных действий:

– просмотр страниц сайта;

– просмотр списка доступных для бронирования мероприятий;

– просмотр информации о свободных местах определенного мероприятия;

– выбор и бронирование мест на мероприятие;

– регистрация нового пользователя;

– аутентификация и авторизация пользователя;

– просмотр администратором страниц панели управления;

– администрирование с помощью CRUD–операций с объектами системы в панели управления.

Рассмотрим варианты использования подробнее. Зайдя на сайт, система управления контентом сайта с помощью cookies аутентифицирует и авторизует посетителя, если пользователь не был ранее аутентифицирован, то пользователю предоставляется базовый набор вариантов использования сайта. В случае если пользователь авторизован, то ему доступны возможности в соответствии с его уровнем доступа. Рассмотрим подробно все возможные варианты использования.

Просмотр страниц сайта. Любой посетитель сайта должен иметь возможность просматривать любые страницы в том числе страницу для ввода данных для аутентификации, расписания мероприятий, страницу карточки мероприятия, новостей и т. п. При просмотре страницы мероприятия информация о наличии свободных мест для бронирования должна быть доступна без регистрации и авторизации. Этот аспект является крайне важным, т.к. простой посетитель должен иметь возможность посмотреть наличие свободных мест и принять решения о регистрации на сайте. В случае если пользователь запрашивает любую несуществующую страницу, то должна быть отдана страница ошибки 404, то есть не найдено.

Регистрация. Решившись забронировать места, посетитель доложен иметь возможность зарегистрироваться на сайте, процедура должно проходить быстро, и от посетителя не должно требоваться лишней информации, форма регистрации должна быть максимально проста. Введя данные для регистрации, пользователь автоматически становится авторизованным, без ввода email-адреса и пароля, и далее ему доступны все возможности обычного пользователя.

Аутентификация и авторизация. Пользователь вводить данные для доступа (email-адрес и пароль). Система проверяет полученные данные,

сравнивая их с хранящимися в БД, если все верно, то авторизует его в соответствии с его уровнем доступа. В противном случае пользователю выводится сообщение об ошибке. Если учетная запись пользователя имеет роль администратора и права доступа к панели управления, ему становится доступен соответствующий раздел. Также для удобства пользователя должна быть страница профиля, где доступна информация о забронированных местах и возможность отмены брони.

Просмотр расписания и мероприятий. Этот вариант использования является частным случаем варианта использования просмотр страниц сайта.

Бронирование. Доступ к функции бронирования предоставляет только зарегистрированным и авторизованным пользователям. В случае если пользователь не авторизован, но пытается каким-либо образом произвести операции бронирования, то ему должно быть выведено сообщение об ошибке. В случае если пользователь авторизован и пытается забронировать место, которое было забронировано другим пользователем после загрузки страницы, то выводится сообщение о том, что данное место уже занято.

Администрирование с помощью панели управления. Если пользователь имеет права доступа к панели управления, ему доступны страницы, позволяющие производить CRUD-операции (акроним от create – создать, read – прочесть, update – обновить, delete – удалить) с объектами системы управления контентом и системы бронирования, такими как страницы, новости, навигация пользователи, мероприятия, места проведения и медиафайлы. Этот вариант использования также является частным случаем просмотра страниц сайта. В случае если пользователь пытается произвести действие с несуществующим объектом, то ему выводится сообщение об ошибке. Описав и проанализировав все возможные варианты использования приложения, можно составить диаграмму прецедентов UML, визуально отображающую процессы, протекающие на сайте (рисунок 6) [24, 25, 26].

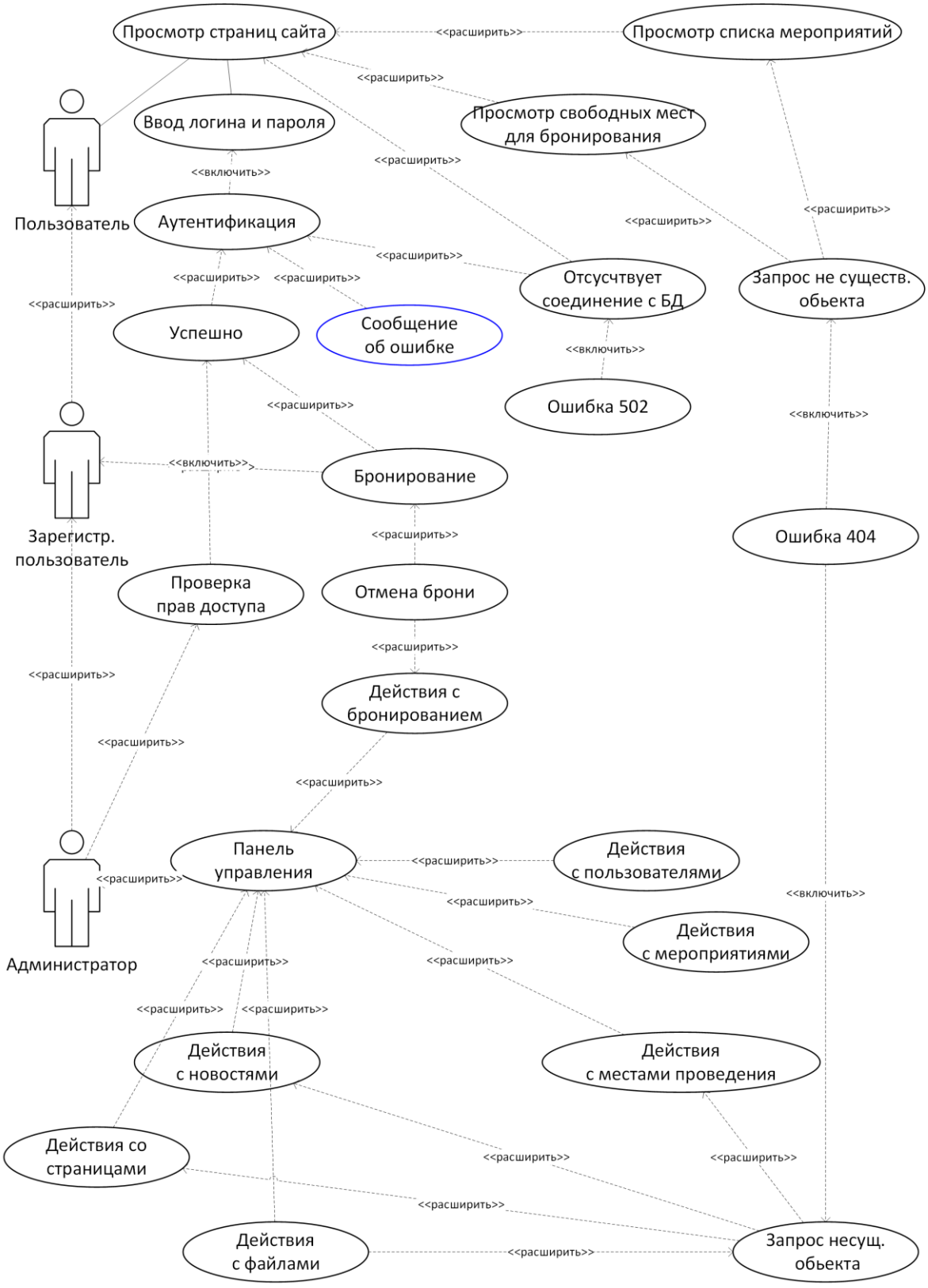

Рисунок 6 – Диаграмма прецедентов

Проанализировав варианты использования, составим диаграмму деятельности (рисунок 7), с помощью UML [26].

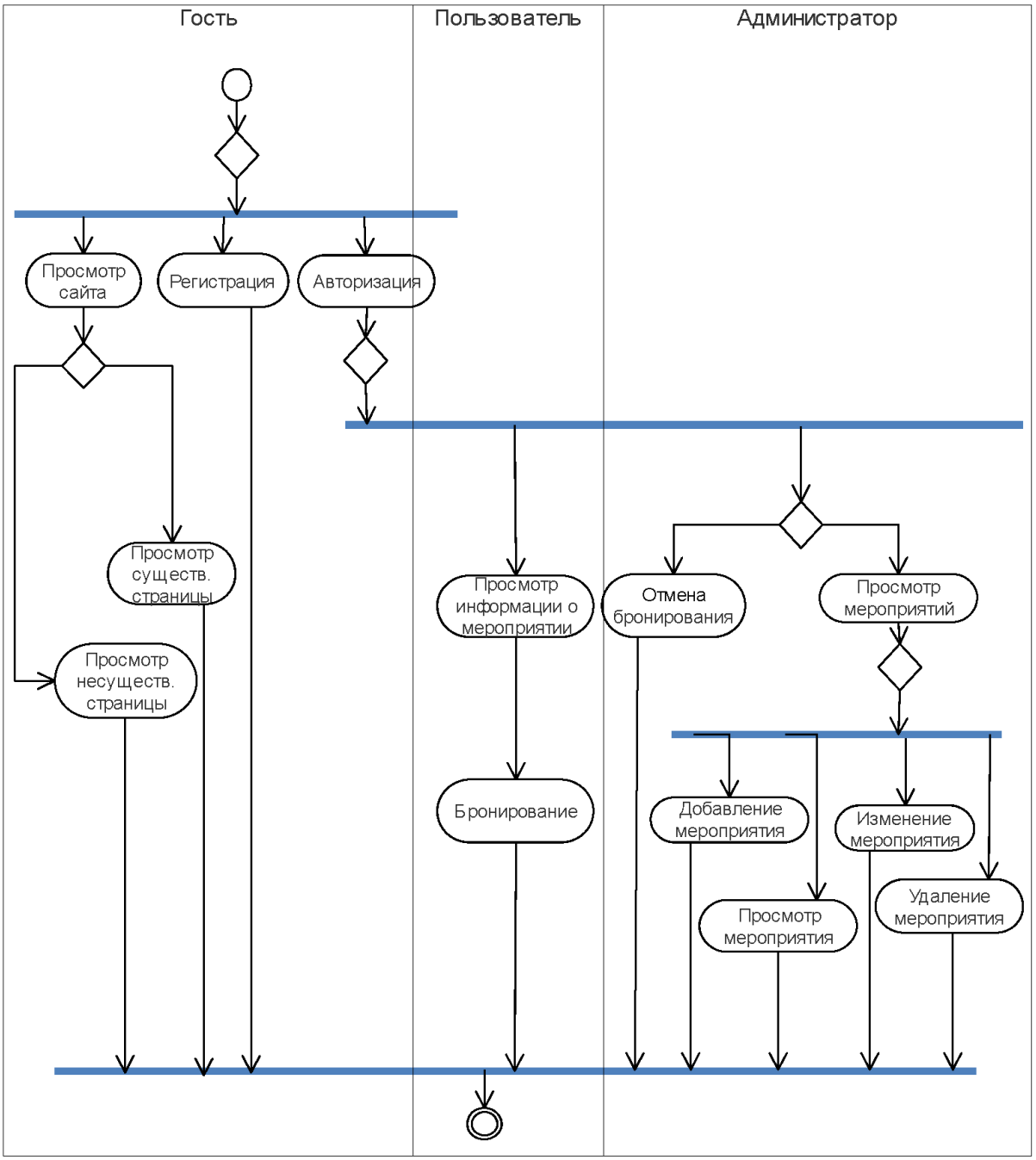

Рисунок 7 – Диаграмма деятельности

На основе вариантов использования, диаграммы прецедентов и диаграммы деятельности была сформирована последовательность действий для следующих прецедентов [25]:

1) бронирование билетов на мероприятие (рисунок 8);

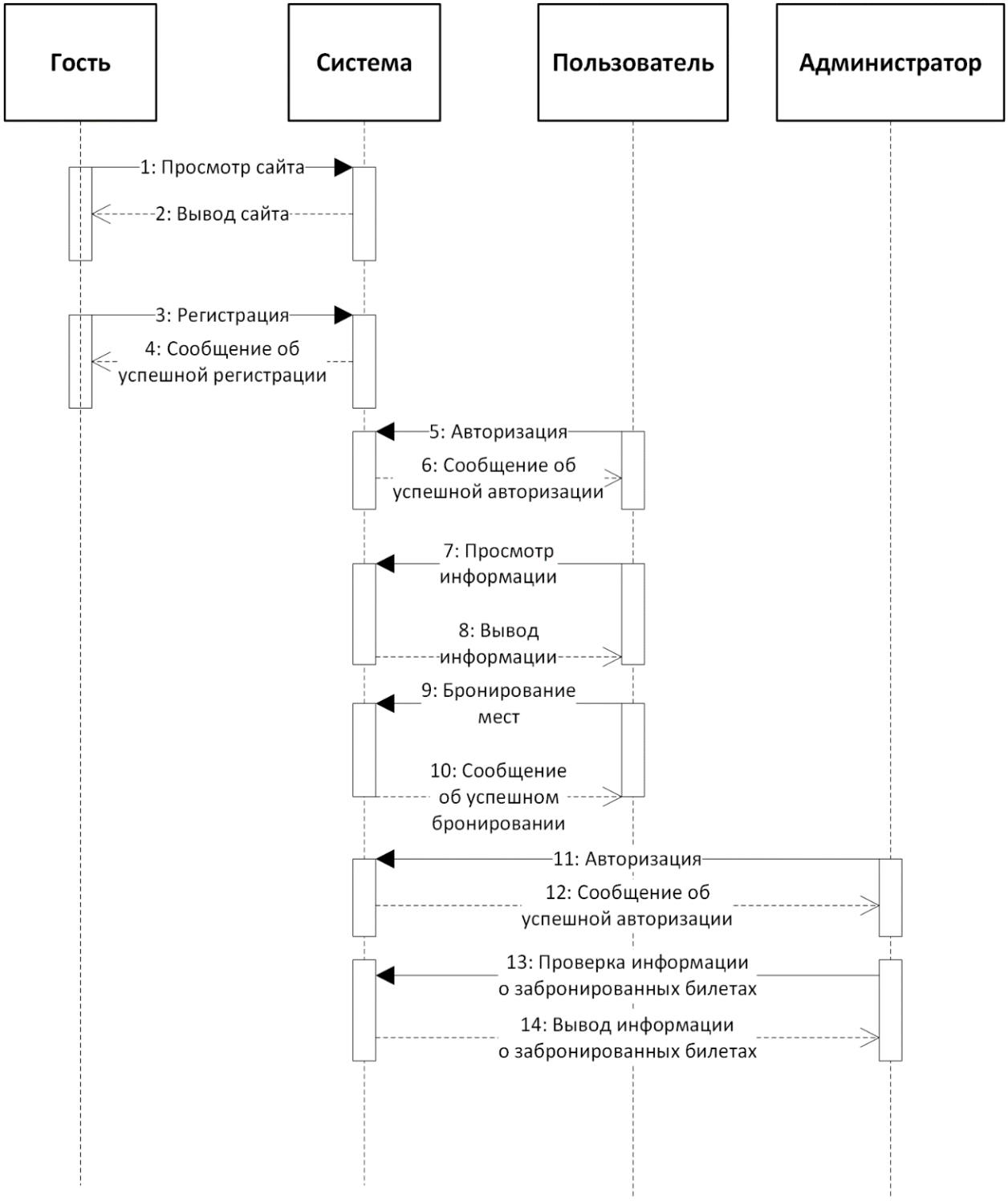

Рисунок 8 – Диаграмма последовательности действий для прецедента бронирование

2) администрирование с помощью CRUD-операций в панели управления (рисунок 9).

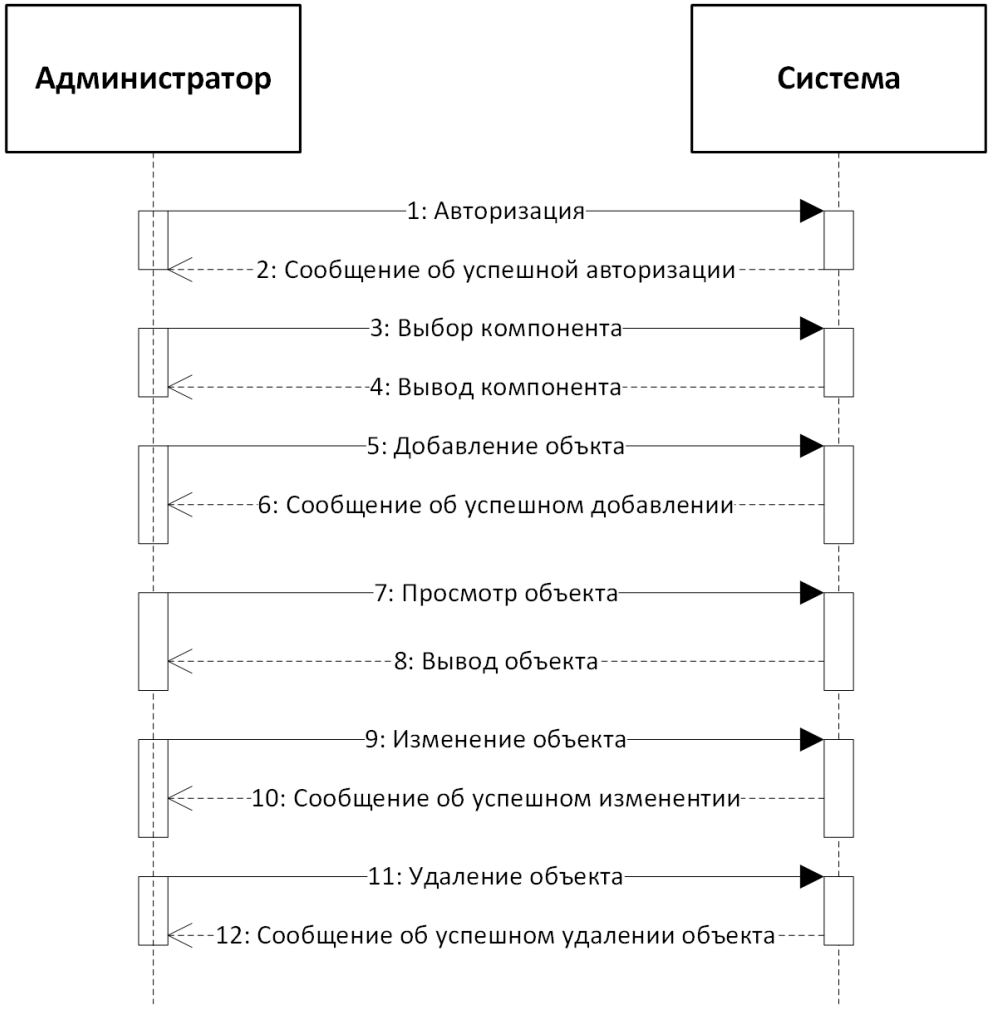

Рисунок 9 – Диаграмма последовательности действий для прецедента администрирование сайта

Проанализировав варианты использования системы бронирования экскурсий и развлекательных мероприятий, можно перейти к определению структурной модели сайта, составлению требований и постановке задачи.

1.6 Постановка задачи

Целью данной работы является создание системы бронирования экскурсий и развлекательных мероприятий, которая будет предоставлять возможность посмотреть доступные мероприятия, бронировать билеты на них, а также информировать об организации предоставляющей данные услуги.

Администратор сайта – менеджер заведения будет иметь информацию о забронированных билетах на экскурсии и мероприятия, с помощью данной

информации у него будет возможность анализировать спрос на те или иные мероприятия и составлять программу экскурсий и мероприятий, наиболее востребованных у аудитории. Также система управления содержимым сайта позволит вести сайт организации, что в свою очередь даст возможность предоставлять актуальную информацию о новостях, акциях и других важных событиях.

Поскольку основным способом поиска информации о экскурсиях и мероприятиях для потенциальных и предполагаемых пользователей являются поисковые системы и веб-сайты, то интерфейсом системы бронирования должен являться именно веб-сайт.

На основе, составленной при анализе вариантов использования UML диаграмме прецедентов, можно составить общую структуру сайта (рисунок 10).

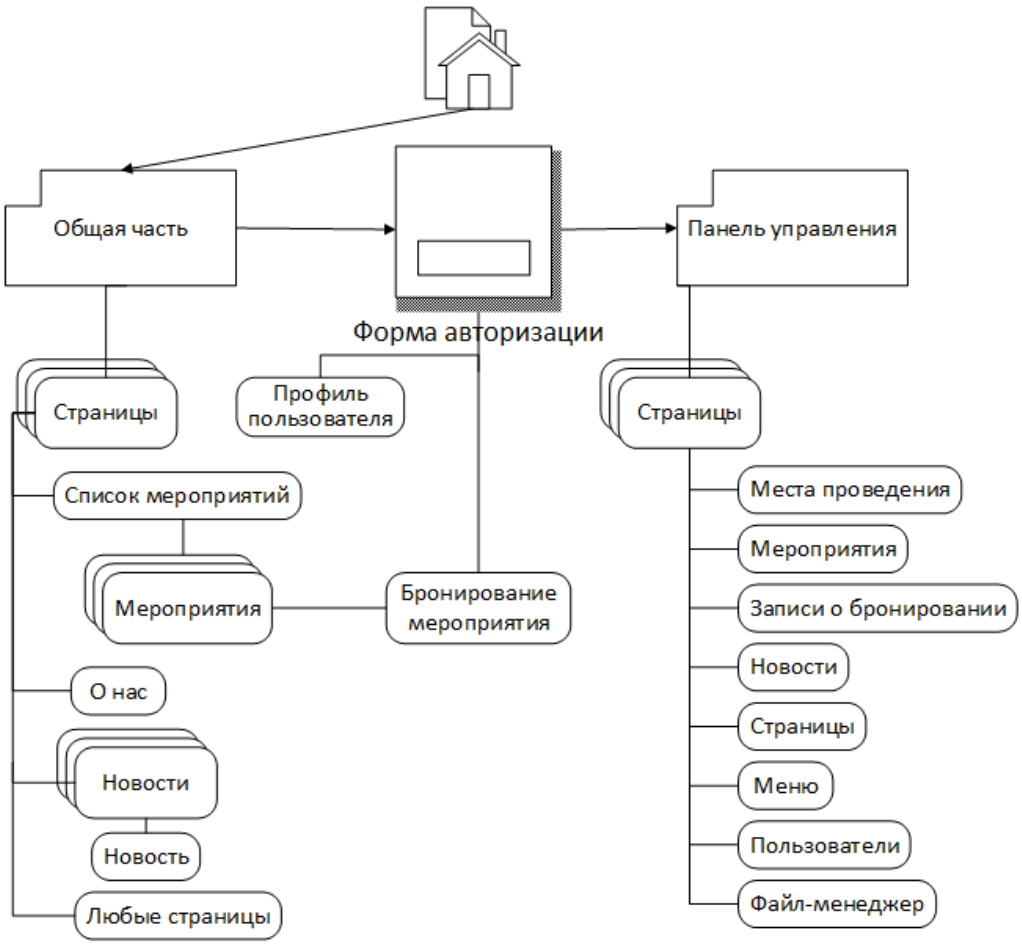

Рисунок 10 – Общая структура сайта

Система бронирования в пользовательской части обязательно должна иметь следующие страницы:

– мероприятия;

– карточки мероприятий;

– профиль пользователя со списком забронированных билетов.

А в административной части соответственно должны быть разделы:

– места проведения;

– мероприятия;

– записи о бронировании;

– пользователи.

Также в системе обязательно должны быть реализованы следующие характеристики и функции:

– контроль и управлением доступом пользователей к функциям системы;

– управление мероприятиями и их свойствами (добавление, просмотр, изменение, удаление);

– свойства мероприятия: название, количество доступных для бронирования билетов, стоимость одного билета, дата и время, описание, иллюстрация;

– бронирование и отмена бронирования билетов на мероприятия.

Рассмотрев аналогичные программные продукты было определено, что наиболее успешные системы бронирования предлагают владельцам заведения возможность ведения сайта, данный функционала у всех реализован по-разному и поэтому обязательных требований к нему нет. Проанализировав необходимые функции для ведения было определено, что для создания конкурирующего программного продукта желательно реализовать в приложении систему управления содержимым сайта.

Основными функциями системы управления содержимым сайта являются возможность создания страниц и иерархического меню, позволяющего связывать между собой создаваемые страницы. Это дает

возможность полноценно вести сайт организации. Также для информирования пользователей желательно наличие отдельного раздела новости. А для работы с документами, изображения и другими файлами необходимо наличие файлового менеджера.

На основе вариантов использования и структурной модели сайта можно оформить требования к разрабатываемому программному продукту в виде технического задания (ТЗ). За основу взят ГОСТ 34.602-89 «Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы» [37]. Некоторые разделы технического задания были адаптированы под разработку веб-приложения.

1.7 Техническое задание

Назначение автоматизированной системы:

– предоставление сервиса бронирования экскурсий и развлекательных мероприятий;

– формирование интерфейса, структуры и содержимого сайта.

Цель создания веб-приложения: предоставление актуальной информации о проходящих мероприятиях и экскурсиях потенциальным посетителям, а также информирование о деятельности организации предоставляющей данные услуги.

1.7.1 Функциональные требования

Разрабатываемая система управления контентом обязательно должна иметь следующие характеристики и функции:

– контроль и управлением доступом пользователей к функциям системы;

– управление мероприятиями и их свойствами (добавление, просмотр, изменение, удаление);

– свойства мероприятия: количество билетов, дата и время, описание, изображение;

– бронирование и отмена бронирования мест на мероприятия.

1.7.2 Необязательные функциональные требования

Также желательно, чтобы программный продукт обладал следующими дополнительными характеристиками и функциями:

– управление страницами сайта (создание, просмотр, редактирование, удаление);

– управление структурой навигации сайта;

– управление пользователями;

– управление новостями (добавление, просмотр, изменение, удаление);

– управление медиа-файлами (изображения, видео, текстовые документы, и т. д.) используемыми в системе;

– управление пользователями (просмотр данных, добавление, изменение, удаление).

1.7.3 Требования к графическому дизайну сайта

Основные разделы сайта должны быть доступны с любой страницы. На главной странице сайта необходимо разместить информацию о текущих и будущих мероприятиях, а также блоки с последними новостями. Сайт должен успешно открываться на мобильных устройствах.

1.7.4 Требования к разделению доступа

Все основные опубликованные страницы и разделы на сайте должны открываться без аутентификации пользователя. В случае, если пользователь обратиться к закрытым разделу или странице, то будет запрошен логин и пароль. После авторизации если пользователю не разрешен доступ к разделу или странице, то будет сообщено о недостаточности прав для доступа и данное действие будет занесено в лог.

Также права доступа могут быть выставлены на отображение отдельных модулей и пунктов меню. В случае если у пользователя недостаточно прав для просмотра, то они не будут ему показаны.

Для получения доступа к административной части сайта необходимо аутентифицироваться на сайте, указав логин и пароль учетной записи с соответствующим уровнем доступа и перейти по ссылке в появившемся меню управления в соответствующий раздел или указать определенный адрес в строке браузера и в случае если пользователь авторизован и имеет права к административному разделу, то ему будет отображен раздел, в противном случае получит ответ о несуществующей странице с HTTP кодом состояния 404 Not Found (не найдено), это позволит повысить уровень безопасности и скроет адрес административной панели [1].

1.7.5 Требования к хранению данных

Всю информацию и данные сайта необходимо хранить в структурированном виде в реляционной БД под управлением СУБД. Файлы изображений, видео, аудио и документов, которые предназначены для просмотра и скачивания, необходимо хранить в файловой системе в специальных папках. Наиболее предпочтительно использование СУБД использующих язык SQL [38].

1.7.6 Требования к языкам программирования и разметки

Для реализации статических элементов страниц и шаблонов необходимо использовать язык разметки HTML 5 и CSS3. Исходный код желательно разрабатывать по стандартам W3C. Для реализации интерактивных элементов клиентской части необходимо использовать язык программирования JavaScript и CSS3 анимацию. Для реализации динамических страниц предпочтительно использование языка программирования PHP.

1.7.7 Требования к иллюстрациям

Все изображения и фотографии, составляющие часть контента (кроме элементов дизайна страницы) необходимо сопровождать альтернативным текстом. Желательно использовать рисунки в распространенных форматах jpg, png, gif.

1.7.8 Требования к программному обеспечению серверной части

Для успешной работы сайта требуется следующее программное обеспечение:

– операционная система Windows Server 2003 или Linux, настроенная должным образом в соответствии с политикой информационной безопасности;

– Веб-сервер Apache версии не ниже 2.2;

– СУБД MySQL версии не ниже 5.5;

– PHP версии не ниже 7.0 [28, 29, 34].

1.7.9 Требования к техническому обеспечению

Для нормального функционирования сайта требуются следующее техническое обеспечение со следующими минимальными характеристиками:

– процессор Intel Pentium III 1 Ghz;

– оперативная память 1024 Mb RAM;

– жесткий диск (HDD) 15 Gb [30].

1.7.10 Требования к клиентскому программному обеспечению

Верстка сайта должна быть кроссбраузерной и он должен быть доступен для полнофункционального просмотра при помощи следующих браузеров:

– Internet Explorer 10 и выше;

– Opera 15 и выше;

– Firefox 15 и выше;

– Google Chrome 20 и выше.

А также корректно отображаться в браузерах мобильных устройств:

– Safari;

– Opera mobile;

– Internet Explorer;

– Google Chrome.

Сайт также должен сохранять работоспособность (информация должна быть доступна) с выключенной поддержкой JavaScript в браузере.

1.7.11 Требования к лингвистическому обеспечению

Текстовое содержание и элементы интерфейса сайта должны быть выполнены на русском языке.

1.7.12 Требования к эргономике и технической эстетике

Сайт должен быть оптимизирован для просмотра при разрешении окна просмотра 1280 на 960 и выше без горизонтальной полосы прокрутки. Элементы управления должны быть сгруппированы однотипно на всех страницах. Пользовательский интерфейс всех модулей веб-приложения необходимо выполнить в едином стиле с общим дизайном. Он должен обеспечивать возможность логичного перемещения между составляющими системы. Для выполнения однотипных операций необходимо использовать одинаковые процедуры управления и навигационные элементы, тем самым создавая интуитивно понятный интерфейс.

1.7.13 Требования к персоналу

Для работы с веб-интерфейсом системы управления от администратора не должно требоваться специальных навыков и знаний, кроме общих навыков работы с персональным компьютером и стандартным браузером.

2 Планирование разработки и оценка бюджета

2.1 Календарное и ресурсное планирование

Проанализировав предметную область и составив требования к программному продукту можно перейти к календарному и ресурсному планированию проекта. Планирование позволяет осуществлять моделирование мероприятий и управление ходом работ по созданию информационной системы.

Для планирования и оценки стоимости построим диаграммы бизнес-процессов для выполнения бакалаврской работы, в нотации IDE в программной среде AllFusion Process Modeler 7. На рисунке 11 представлена контекстная диаграмма бизнес-процесса «Бакалаврская работа».

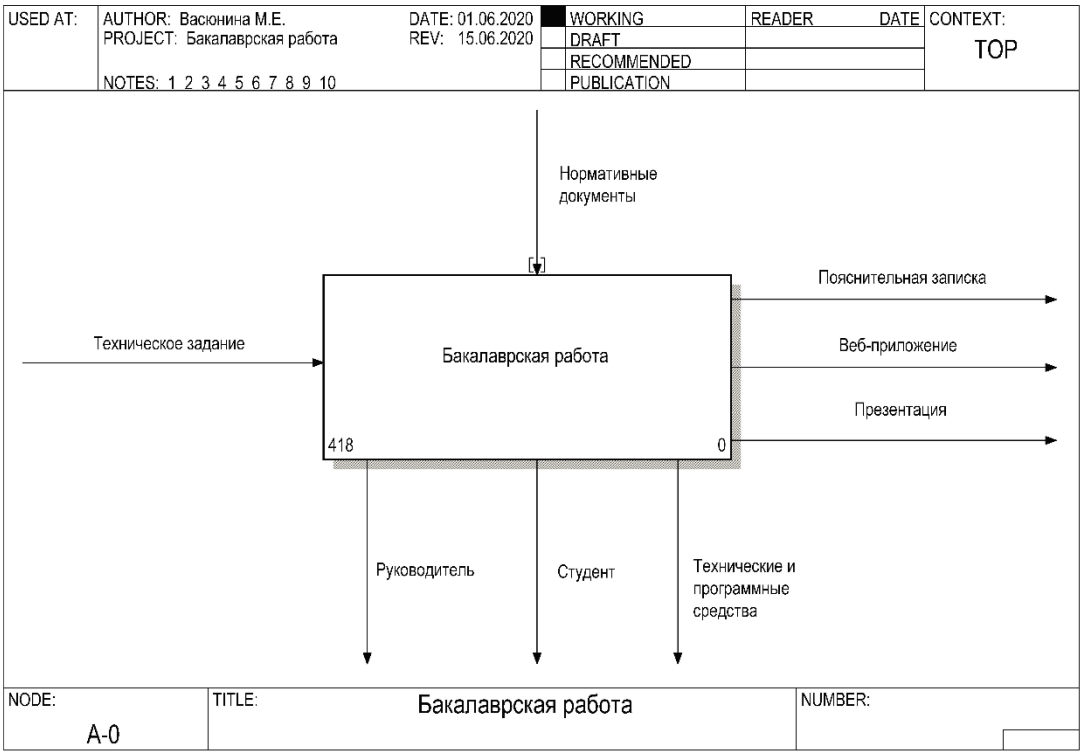

Рисунок 11 – Контекстная диаграмма «Бакалаврская работа»

Затем формализуем процесс «Выполнение бакалаврской работы» и проведем его декомпозицию. Результат декомпозиции можно увидеть на рисунке 12.

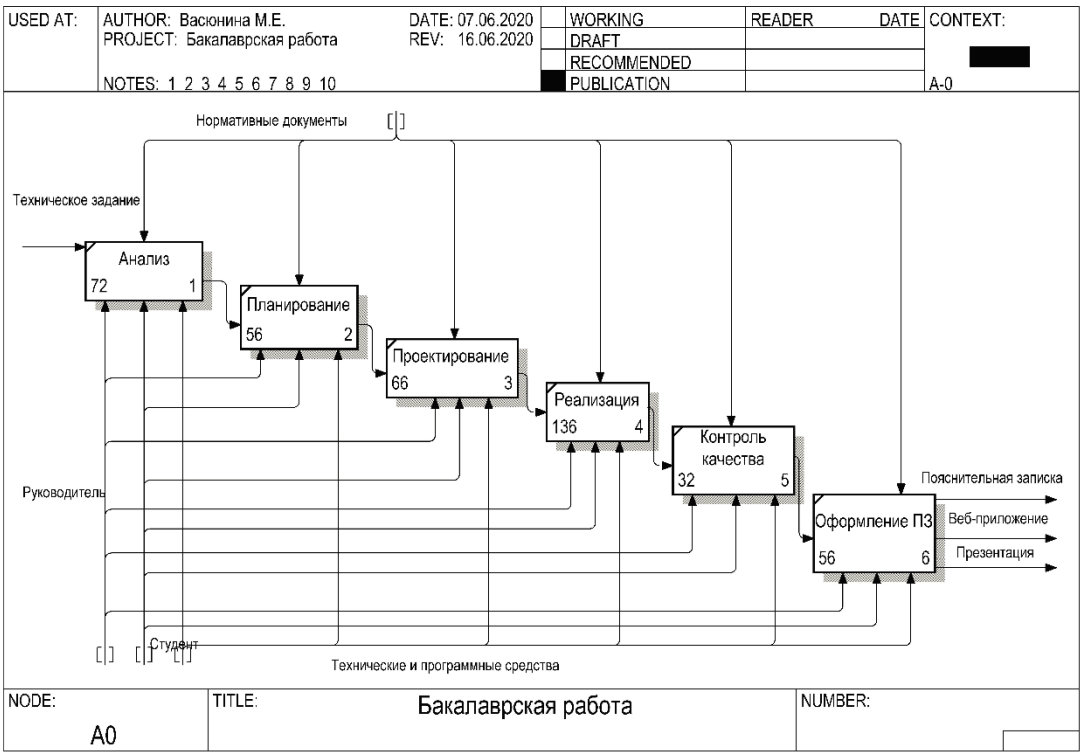

Рисунок 12 – Диаграмма декомпозиции процесса «Бакалаврская работа»

При декомпозиции контекстной диаграммы процесс выполнения был разбит на 6 составляющих процессов: «Анализ предметной области и постановка задачи», «Планирование», «Проектирование веб-приложения», «Реализация», «Контроль качества программного продукта», «Оформление пояснительной записки».

Составим календарный план проекта. Он содержит определение длительности работ и их взаимосвязей. Для более удобного восприятия сроков работ по проекту, продолжительности этапов, а также отражения расхода ресурсов используется простой и полезный инструмент календарного планирования проектов – диаграмма Ганта.

Кроме перечня работ в календарное планирование входит составление ресурсной модели проекта. Можно выделить несколько видов фундаментальных ресурсов: трудовые, материальные и не возобновляемые. Каждый ресурс имеет свои особенности, влияющие на их использование. Затраты на расходование этих ресурсов и составляют бюджет проекта.

Использование календарного планирования позволяет получить в итоге план-график проекта с оптимальным соотношением стоимости и сроков.

Составим календарное планирование для проекта «Веб-приложение бронирования экскурсий и развлекательных мероприятий».

Составим план стадий и этапов создания веб-приложения. План, содержащий подробную информацию о наименовании и длительности стадий и этапов разработки, а также лиц выполняющих их представлен в приложении А.

Осуществим планирование проекта в программе Microsoft Project. Внесем информацию о задачах и создадим диаграмму Ганта (приложение Б).

А также внесем информацию о ресурсах (рисунок 13).

Название ресурса	Тип	Единицы измерения материалс	Краткое названи	Группа	Макс. единиц	Стандартна ставка
Руководитель проекта	Трудовой		РП		100%	350,00 ₽/ч
Backend программист	Трудовой		В		100%	300,00 ₽/ч
Frontend программист	Трудовой		F		100%	250,00 ₽/ч
Тестировщик	Трудовой		Т		100%	250,00 ₽/ч
Дизайнер	Трудовой		Д		100%	250,00 ₽/ч
Компьютер	Материальны		К			50 000,00 ₽
Принтер	Материальны		Пр			5 000,00 ₽
Бумага	Материальны		Б			250,00 ₽
Интернет	Материальны		Ин-т			450,00 ₽

Рисунок 13 – Лист ресурсов проекта

В таблице 4 представлена структура трудозатрат по этапам разработки.

Таблица 4 – Структура трудозатрат

Задача	Трудозатраты
Анализ предметной области	72 ч
Планирование и расчет бюджета	55,92 ч
Проектирование веб-приложения	66 ч
Реализация	136 ч
Тестирование разработанного приложения	32 ч
Оформление ПЗ	56 ч
Итого	418 ч

Проанализировав этапы и их продолжительность можно сделать вывод, что на проектирование и разработку программного продукта «Веб-приложение бронирования экскурсий и развлекательных мероприятий» потребуется 418 часов.

2.2 Расчет бюджета

Проведем расчет стоимости проектирования и разработки проекта.

В основе расчета затрат лежит расчет технологической себестоимости, которая рассчитывается по следующим статьям:

- амортизация;
- материалы;
- фонд основной и дополнительной заработной платы;
- социальные выплаты;
- прочие.

Произведем расчет затрат по статье «Амортизация». Амортизация это – процесс перенесения по частям стоимости изношенной части основных производственных фондов на стоимость готовой продукции.

Стоимость ОПФ и расчет амортизационных отчислений приведены в таблице 5.

Таблица 5 – Амортизационные отчисления

Оборудование	Стоимость, руб.	СПИ, лет	На, %	А, год, руб.
Компьютер с ПО: MS Office 2016; MS Windows 10	50 000	5	0,2	10 000
Принтер HP	5 000	5	0,2	1000
ТЗР (1%)	550			
Итого	55 000			11 550

Для разработки программного продукта планируется использовать программное обеспечение в свободном доступе: Microsoft Visual Studio Code, операционная система CentOS, СУБД MySQL, веб-сервер Apache, фреймворк Symfony [42], шаблонизатор Twig [44], файловый менеджер для веб-приложений elFinder [45], платформонезависимый JavaScript HTML WYSIWYG-редактор TinyMCE [46] и JavaScript-фреймворк Vue,js [47].

Данное оборудование находится в эксплуатации 3 года, и его остаточная стоимость равна 22 000 руб.

Амортизационные отчисления рассчитываются по следующей формуле (2):

$$А = С_{пер} * НА, \quad (2)$$

где А – амортизационные начисления, руб.;

$С_{пер}$ – первоначальная стоимость основных средств, руб.;

НА – норма амортизации, которая находится по формуле (3):

$$На = 1 / СПИ, \quad (3)$$

где СПИ – срок полезного использования основных средств.

Срок полезного использования компьютера = 5 лет.

$$НА_{комп} = 1/5 = 0,2.$$

Срок полезного использования принтера = 5 лет.

$$НА_{принт} = 1/5 = 0,2.$$

$$А_{комп} = 50\,000 * 0,2 = 10\,000 \text{ руб.}$$

$$А_{принт} = 5\,000 * 0,2 = 1000 \text{ руб.}$$

$$А_{г} = 10\,000 + 1000 + 550 = 11\,550 \text{ руб.}$$

$$А_{м} = 11\,550 / 12 = 962,5 \text{ руб/мес.}$$

$$А_{итог} = 962,5 * 2 = 1\,925 \text{ руб.}$$

Работа по созданию программного продукта предполагается вести 418 часов, расчёт ведётся за 2 месяца.

Произведем расчет затрат по статье «Материалы». В данной статье расходов приведены расчеты оборотных средств, использовавшиеся для разработки программного продукта, они приведены в таблице 6.

Таблица 6 – Затраты по статье «Материалы»

Наименование	Количество, шт.	Сумма, руб.
Бумага	2	500
USB-флеш-накопитель	1	500
ТЗР (1%)		10
ИТОГО:		1010

Расчет затрат по статье «Фонд основной и дополнительной заработной платы»

Зарплата – вознаграждение за труд в зависимости от квалификации работника, сложности, количества, качества и условий выполняемой работы.

Трудоемкость всех видов работ, связанных с разработкой программного продукта «Веб-приложение бронирования экскурсий и развлекательных мероприятий» представлена в таблице 7.

Таблица 7 – Расчет основной и дополнительной заработной платы

Исполнитель	Трудоемкость, часов (Тр)	Ставка, руб/час	ОЗП, руб.	ДЗП, руб.	Соц. выплаты
Руководитель проекта	212,80	350	66 144,89	6 614,49	21 827,81
Frontend программист	88,56	300	23 498,84	2 349,88	7 754,62
Backend программист	80,56	250	17 711,62	1 771,16	5 844,83
Тестировщик	8,00	250	1 758,85	175,89	580,42
Дизайнер	20,00	250	4 397,13	439,71	1 451,05
Итого:	409,92		113 511,33	11 351,13	37 458,74

На разработку и тестирование программного продукта затрачено 418 часов.

Рассмотрим оплату труда на примере расчета заработной платы руководителя проекта.

Оплата за труд рассчитывается следующим образом, по формуле (4):

$$ОЗП = Сч * Тр, \quad (4)$$

где Тр – трудоемкость,

Сч – тарифная ставка, которая рассчитывается по формуле (5):

$$Сч = (МРОТ * Тк)/160, \quad (5)$$

где МРОТ – минимальный размер оплаты труды = 12 130 руб.;

Тк – тарифный коэффициент;

160 – количество часов в месяц по ТК РФ.

$$Тк_{рук} = 4,1.$$

$$Сч_{рук} = (12\ 1130 * 4,1)/160 = 310,83.$$

$$ОЗП_{рук} = 213,8 * 273,76 = 66\ 144,89 \text{ руб.}$$

Мотивация труда оценивается через дополнительную заработную плату, которая рассчитывается по формуле (6):

$$ДЗП = \%ОЗП, \quad (6)$$

$$ДЗП_{рук} = 10\% * 66\ 144,89 = 6\ 614,49 \text{ руб.}$$

Расчет затрат по статье «Социальные выплаты», производится по формуле (7).

Социальные выплаты состоят из:

- фонд социального страхования (ФСС) – 5,9%;
- пенсионный фонд РФ (ПФР) – 22%;
- фонд обязательного медицинского страхования (ФОМС) – 2,1%.

$$СВ = 30\% (ОЗП + ДЗП). \quad (7)$$

$$СВ_{рук} = 30\% * (66\ 144,89 + 6\ 614,49) = 21\ 827,81 \text{ руб.}$$

Произведем расчет затрат по статье «Прочие». В этой статье приведены затраты на доступ в Интернет и Электроэнергия (ЭЭ).

ЭЭ рассчитывается по формуле (8):

$$ЭЭ = Тс * W * Т_{раб}, \quad (8)$$

где Тс = стоимость 1 кв/ч = 3,63 руб.,

W – энергомощность компьютера = 350 Вт/ч = 0,35 кВ/ч,

$Т_{раб}$ – время создания, т.к. программный продукт разрабатывался 418 часов.

$$ЭЭ = 3,63*0,35*418 = 531,07 \text{ руб.}$$

Произведем расчет на услугу доступ в Интернет. Работа над программным продуктом велась 418 часов, тариф за услугу доступ в Интернет от провайдера Ростелеком составляет = 450 руб./мес. Однако, провайдер не предоставляет почасовой доступ в интернет, поэтому расчет ведется за 2 месяца.

$$Интернет = 450*2 = 900 \text{ руб.}$$

Прочие затраты = ЭЭ + Интернет = 531,07 + 900 = 1 431,07 руб.

Структура затрат представлена в таблице 8.

Таблица 8 – Структура затрат

Наименование статей	Затраты, руб	%
Амортизация	1 925	1%
Материалы	1 010	1%
Фонд основной и дополнительной ЗП	124 862	75%
Социальные выплаты	37 458	22%
Прочие	1 431	1%
Затраты на проект	166 687	

В результате для визуализации структура расходов была составлена диаграмма структуры затрат (рисунок 14).

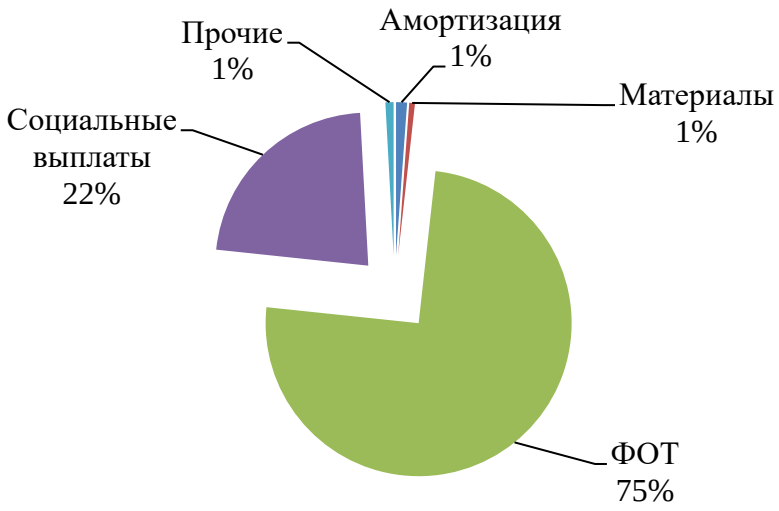

Рисунок 14 – Структура затрат

Проводя анализ структуры расходов можно сделать вывод, что основными статьями расходов является фонд оплаты труда и социальные выплаты. Затраты на проектирование и разработку программного продукта «Веб-приложение бронирования экскурсий и развлекательных мероприятий» составят 166 687 рублей.

2.3 Выбор программно-аппаратных средств

Веб-приложение представляет собой клиент-серверное приложение, в котором все взаимодействие происходит в браузере[1]. Самым главным преимуществом веб-приложения от обычной программы является его кроссплатформенность, поскольку веб-браузеры существуют практически для всех операционных систем, то такое приложение будет доступно на подавляющем большинстве устройств. Основной проблемой является различия в реализации в браузерах спецификаций CSS, HTM, JavaScript и других.

Сайт – это одна или несколько логически связанных между собой страниц, расположенных на веб-сервере и просматриваемых через веб-браузер [1, 2].

Динамический сайт – это сайт, состоящий из динамичных страниц. В свою очередь динамическая страница состоит из контента, шаблонов и

скриптов. В конечном итоге в браузере пользователю показывается сформированная из различных файлов с помощью серверной и клиентской частей страница. В случае отображения большого количества однотипных страниц обычно применяется страница-шаблон, в блоки которой загружаются соответствующие данные. Этот метод позволяет одновременно изменяет внешний вид большой группы страниц, изменяя одну страницу-шаблон, а также способствует повторному использованию блоков в представлении.

Редактирование страниц-шаблонов и содержимого может производиться с помощью встроенного визуального редактора, путем использования специального программного обеспечение, а также это может производиться вручную.

Процесс получения динамической страницы можно представить следующим образом:

1) генерация содержимого на стороне сервера;

2) передача сгенерированной странички клиенту;

3) генерация содержимого на стороне клиента [28].

Генерацию содержимого производит сервер, обрабатывая запрос и запуская обработку соответствующего файла-скрипта, например, файл news.php.

На этой стадии необходимая информация запрашивается из базы данных, а затем передается на обработку и подстановку в шаблон, как на стороне сервера, так и на стороне пользователя в случае AJAX приложений и подобных ему.

Генерация содержимого на стороне клиента производится браузером, после получения ее с сервера. Причем после получения страницы от сервера, в браузере могут исполняться скрипты клиентской стороны. На клиентской стороне используется JavaScript, который может производить различные изменения от незначительных, до глобальных [51].

Чаще всего при создании веб-страниц используется комбинированная обработка с использованием описанных выше методов.

Система управления содержимым (CMS) – это автоматизированная информационная система, обеспечивающая организацию совместного процесса создания, изменения и управления содержимым (контентом).

Основными функциями являются:

– возможность создания содержимого и организации совместной работы над ним;

– управление содержимым, включающее в себя хранение, соблюдение режима доступа, управление документами и многое другое;

– публикация содержимого;

– представление доступа к информации в удобном для навигации и поиска виде.

В системе управления содержимым могут находиться самые различные данные: изображения, фотографии, документы, научные данные и так далее [1].

Наиболее подходящим для создания веб-приложений или по-другому динамических веб-страниц является шаблон проектирования MVC и производные от него. Model-view-controller (модель-представление-контроллер) – это схема с использованием нескольких шаблонов проектирования. Основная идея, которого заключается в том, что модель приложения, интерфейс пользователя и взаимодействие с пользователем разделены на три отдельных компонента. Таким образом, чтобы при модификации одного из компонентов воздействие на остальные сказывалось минимально [49].

Основными технологиями для создания веб-приложений на стороне сервера (backend) являются: ASP.NET, PHP, Python (Django), Perl, Ruby (фреймворк Ruby on Rails), Javascript (Node.js), Golang (Go).

Рассмотрим наиболее популярные на сегодняшний день технологии подробнее.

ASP.NET – технология от компании Microsoft. Является составной частью платформы Microsoft .NET и развитием более старой технологии

Microsoft ASP. На данный момент последней версией этой технологии является ASP.NET 5 [31].

PHP (Hypertext Preprocessor – препроцессор гипертекста) – интерпретируемый скриптовый язык общего назначения. Применяется 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 [28].

Python – высокоуровневый язык программирования общего назначения с минималистичным синтаксисом ядра. Его основная особенность читаемости кода и за счет этого повышение производительности разработки. Для создания сайтов на этом языке программирования чаще всего используется Django – программный каркас, использующий шаблон проектирования MVC [35].

Perl – интерпретируемый высокоуровневый язык программирования общего назначения. Основная особенность языка – его большие возможности для работы с текстом, значительной частью которых является встроенные в синтаксис регулярные выражения [1, 33].

Ruby – интерпретируемый высокоуровневый язык программирования. Одна из его особенностей это независимая от операционной системы реализация многопоточности, а также строгая динамическая типизация [36]. Для создания веб-приложений чаще всего используется фреймворк Ruby on Rails, реализующий шаблон проектирования MVC.

Система управления контентом (CMS) – это автоматизированная информационная система, обеспечивающая организацию совместного процесса создания, изменения и управления содержимым (контентом).

Основными функциями являются:

– возможность создания содержимого и организации совместной работы над ним;

– управление содержимым, включающее в себя хранение, соблюдение режима доступа, управление документами и многое другое;

– публикация содержимого;

– представление доступа к информации в удобном для навигации и поиска виде.

В системе управления контентом могут находиться самые различные данные: изображения, фотографии, документы, научные данные и так далее [1].

Проанализировав определения сайта созданного с помощью системы управления содержимым и веб-приложения можно сделать вывод, что система управления содержимым сайта является веб-приложением.

3 Проектирование системы бронирования

3.1 Проектирование пользовательского интерфейса

Поскольку система бронирования будет размещена на сайте, то тип интерфейса графический, такой же, как и у браузера. Сайт – это система из страниц. Это документ, написанный на языке разметки гипертекста (HTML), который можно просмотреть с помощью браузера.

Сайт веб-приложения должен быть сверстан таким образом, чтобы его страницы корректно отображались во всех современных браузерах. Интерфейс должен быть логично структурирован и интуитивно понятен пользователю. Для этого все страницы сайта должны иметь схожую структуру [2].

Изучив тренды современного дизайна, был спроектирован общий макет сайта. Графический интерфейс всех страниц сайта должен содержать следующие элементы (рисунок 15):

– верхнюю часть, состоящую из логотипа с левой стороны и панели пользователя с правой стороны, открывающей форму для ввода логина и пароля, либо меню пользователя в случае если пользователь авторизован;

– навигационное меню сайта (навигационная панель, позволяет переходить к большинству основных разделов сайта, а также содержит ссылку на главную страницу);

– основную область, отображающую контент выбранной страницы сайта;

– в нижней части сайта год, название и контактные данные.

Содержимое сайта с системой бронирования и интерфейс панели управления существенно отличаются, поэтому макеты имеет существенные различия.

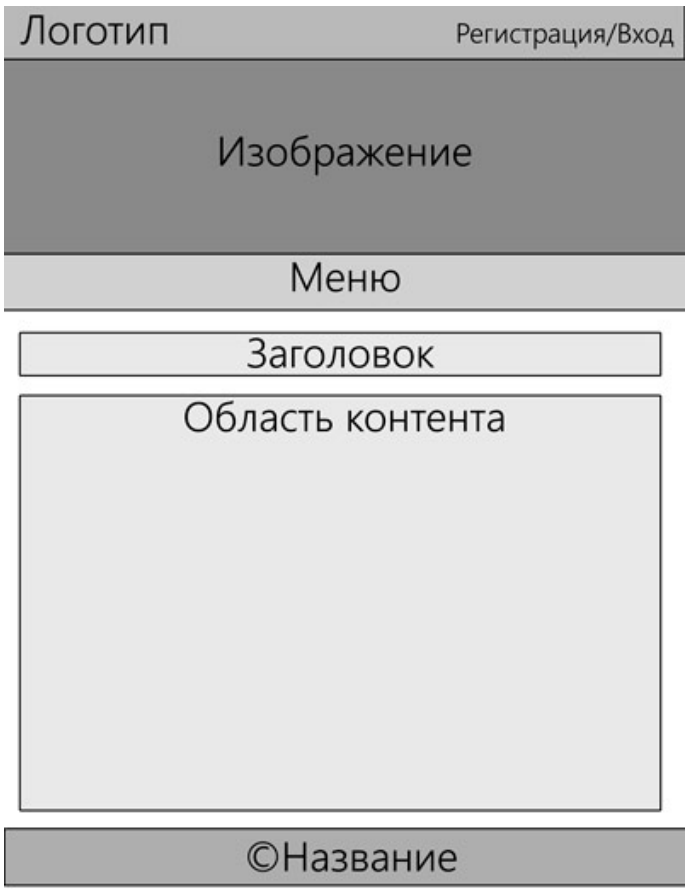

Рисунок 15 – Макет сайта системы бронирования

Для получения доступа к административной части сайта необходимо авторизоваться на сайте, указав логин и пароль учетной записи с соответствующим уровнем доступа и перейти по ссылке в выпадающем меню, которое доступно при нажатии на имя пользователя в правом верхнем углу сайта. управления в соответствующий раздел или указать определенный адрес в строке браузера и в случае если пользователь авторизован и имеет права к административному разделу, то ему будет отображен раздел, в противном случае получит ответ о несуществующей странице с HTTP кодом состояния 404 Not Found (не найдено), это позволит повысить уровень безопасности, и скроет адрес административной панели [2].

Пример макета пользовательского интерфейса главной страницы панели управления в административной части сайта показан на рисунке 16.

На главной странице административной части сайта должна быть расположена сводная информация.

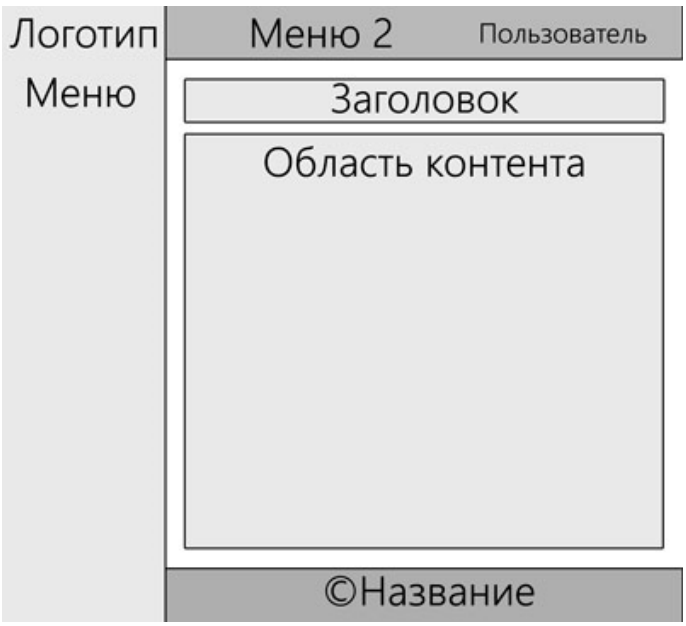

Рисунок 16 – Макет интерфейса панели управления системы бронирования

А также она должна содержать следующие пункты меню:

– мероприятия;
– места проведения;
– бронирование;
– страницы;
– меню;
– пользователи;
– новости;
– файловый менеджер.

Проанализировав техническую литературу и информационные ресурсы по разработке сайтов можно прийти к выводу, что все страницы сайта должны иметь следующие свойства:

– непосредственно содержимое страницы (контент) с визуальным редактором с возможностью просмотра и изменения исходного кода;

– заголовок (title) страницы и возможность его редактирования [40, 41];

– заголовок (тег h1) страницы и возможность его редактирования;

– мета-теги страницы (meta keywords, meta description) и возможность их редактирования [40, 41];

– дату создания страницы и идентификатор пользователя, совершившего операцию;

– дату редактирования и идентификатор пользователя, совершившего операцию;

– флаг публикации страницы в клиентской части [2, 28].

3.2 Организация процесса обработки информации

Проанализировав варианты использования и протекающие бизнес процессы в системе управления содержимым, были определены необходимые компоненты и их взаимосвязи и составлена диаграмма компонентов, представленная на рисунке 17.

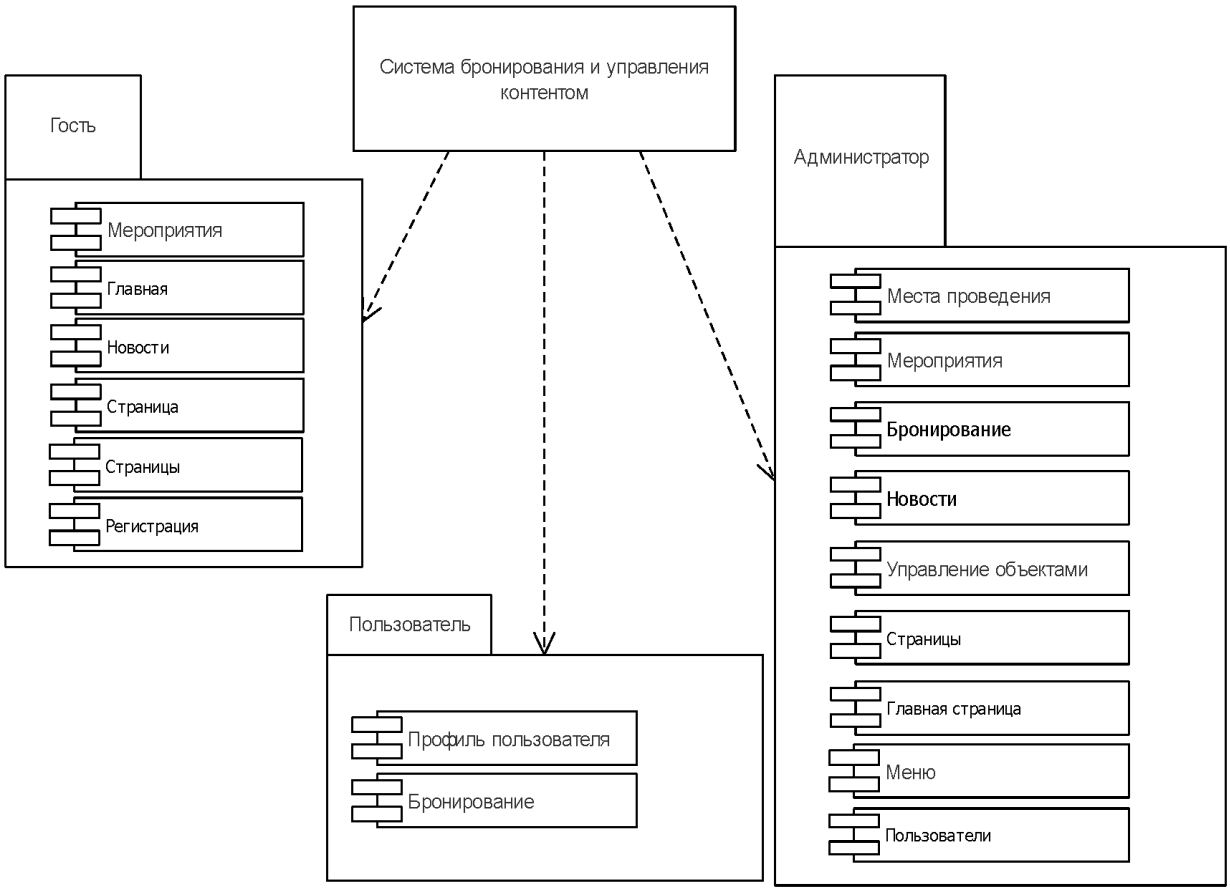

Рисунок 17 – Диаграмма компонентов

В веб-приложении обычно используется единая точка входа, в веб-сервере Apache это можно реализовать определив с помощью конфигурационного файла .htaccess (приложение В) переадресацию всех запросов на один файл [30]. Таким образом все запросы, кроме изображений, файлов стилей, скриптов и некоторых других файлов и папок будут перенаправляться в единую точку входа – файл index.php (приложение Г), расположенный, например, в корне сайта [1, 2].

В файле единой точки входа – index.php устанавливаются необходимые константы, подключаются файлы классов и библиотек, а также реализуется паттерн MVC. А также создается главный объект приложения – экземпляр класса Application. Он является ядром всего веб-приложения и загружает все дополнительные сервисы, а также является контейнером паттерна Dependency injection (Внедрение зависимости), тем самым храня все данные и предоставляя доступ к ним из любой точки приложения.

За обработку всех запросов обычно отвечает маршрутизатор. Он получает запрашиваемый адрес, сравнивает его с помощью регулярных выражений со списком адресов и соответствующих им контролеров и вызывает первый совпавший, а в случае, если совпадений не обнаружено, то выбрасывается исключение Not Found 404.

Контроллеры обеспечивают связь между клиентской частью приложения и системой. Реализуют всю бизнес-логику всего приложения. В контроллерах осуществляются запросы к моделям, например, моделям данных из ORM. Полученные и обработанные данные передаются в представление и с помощью заранее созданных шаблонов страниц и шаблонизатора, например, Twig или Pug, формируется HTML-код и содержимое страницы. После чего передается браузеру, например, с помощью объекта класса Response фреймворка Symfony [42, 43, 44]. Тем самым реализуется концепция шаблона проектирования MVC.

Система контроля реализуется в виде идентификации на сайте с помощью пары логин и пароль. При вводе необходимых для аутентификации

данных в форму на сайте они передаются методом POST в метод login, класса User, реализующего подсистему контроля доступа.

Класс User модуля контроля доступа сравнивает переданные данные с данными в базе и если они совпадают, то аутентифицирует и авторизует пользователя на сайте. Это может быть реализовано с помощью механизма сессий. Для работы с сессиями возможно использовать готовый компонент Session фреймворка Symfony [44]. Также рекомендуется реализовать возможность запомнить пользователя не на одну сессию, а на более длительный срок и не запрашивать логин и пароль каждый раз. Такая возможность реализуется с помощью установки в cookie идентифицирующего ключа. Таким образом при запросе к веб-приложению проверяется существование записи в базе данных, соответствующей ключу в cookie и в случае существования пользователь аутентифицируется [1, 2].

Создание иерархического меню неограниченной вложенности возможно с помощью рекурсивных макросов, например, на языке шаблонизатора, и функции проверки прав доступа к каждому элементу меню. Хранение в реляционной базе данных иерархической структуры обычно реализуется с использованием поля parent_id, указывающего на родителя элемента [1].

В административном разделе необходимо реализовать следующие операции: создание, просмотр, редактирование, удаление для объектов: страница, пользователь, новость, мероприятие, площадка проведения.

Для свойств объектов, который представляю собой код html, например, содержимое страниц сайта в систему управления контентом можно интегрировать визуальный редактор TinyMCE [47].

TinyMCE представляет собой WYSIWYG (от англ. What You See Is What You Get, «что видишь, то и получишь») редактор, который позволяет вставлять таблицы, видео, рисунки, указывать стили оформления текста и т.д. К основным характеристикам данного редактора можно отнести поддержку тем/шаблонов, языковую поддержку и возможность подключения плагинов [47].

Рабочий интерфейс визуального редактора схож с интерфейсом редактора MS Word, поэтому использовать его могут все, кто владеет намывами работы в MS Word. Также движок позволяет использовать другой визуальный редактор, который необходимо подключать самостоятельно [47].

Для работы с файлами подходит файловый менеджер elFinder с удобным интерфейсом для работы с файлами. Его интерфейс схож с интерфейсом любых других программ для работы с файлами на компьютере [48].

Таким образом, были рассмотрены основные функции системы управления содержимым позволяющие создать веб-приложение бронировании мероприятий. Данная система управления содержимым позволит добавлять, редактировать, удалять любые объекты. Благодаря файловому менеджеру будет возможность размещать электронные документы различных форматов на страницах сайта.

3.3 Структуры хранения данных

Проанализировав информационные процессы, возникающие во время использования приложения, были определены основные типы и объемы обрабатываемой информации.

Для хранения всех данных системы бронирования и содержимого сайта используется СУБД MySQL. База данных состоит из 9 таблиц. Все данные в таблице приведены в нормальную форму. Для визуального проектирования базы данных было использовано программное обеспечение Microsoft Visio, и в результате получена UML-диаграмма базы данных, представленная на рисунке 18 [26, 52, 53].

Данные необходимые для работы системы бронирования хранятся в следующих таблицах: в таблице «Events» хранится информация о различных мероприятиях, доступных для бронирования, в таблице «Area» хранится информация о площадках проведения мероприятий, описанных в предыдущем пункте.

В таблице «Reserve» хранится информация о забронированных билетах, идентификаторы мероприятия, пользователя, а также выбранные опции.

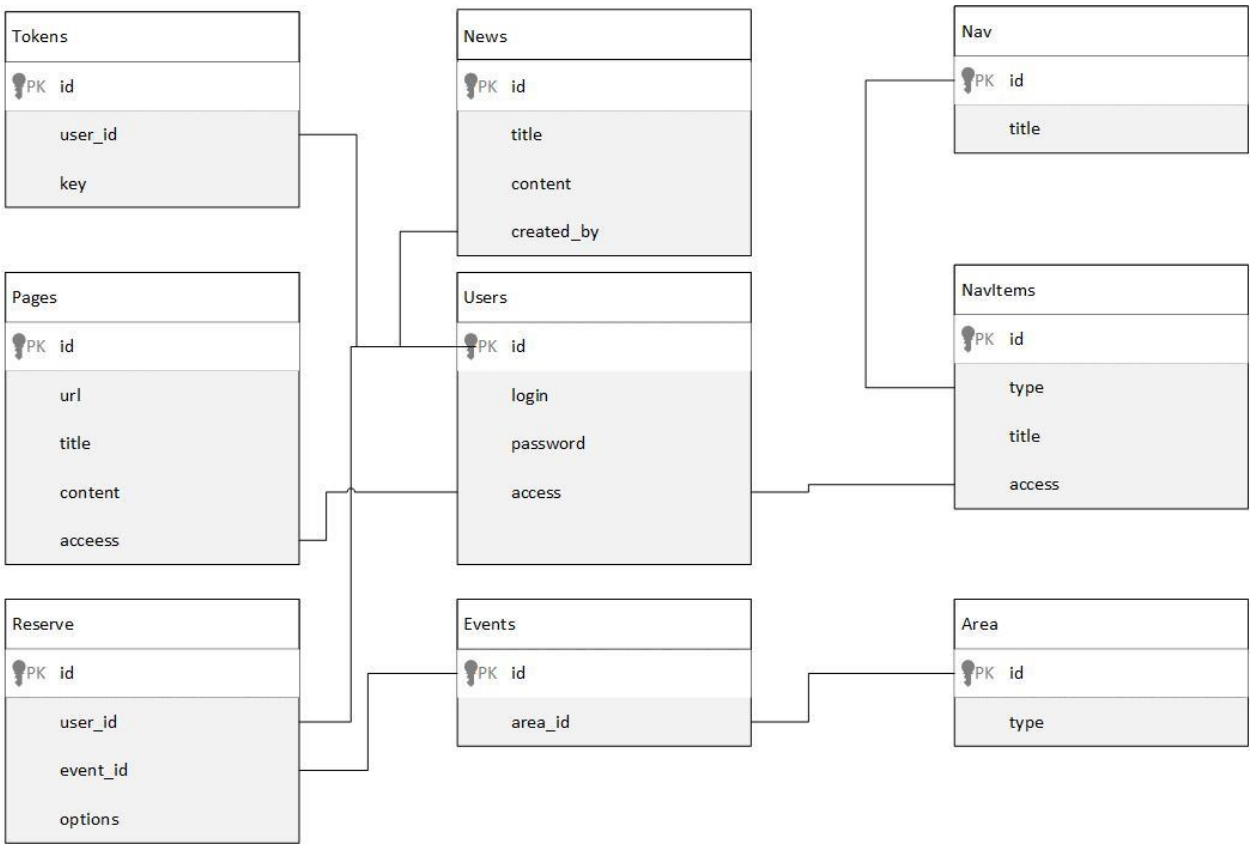

Рисунок 18 – Диаграмма классов базы данных

Описанные далее таблицы являются вспомогательными и хранят данные системы управления контентом сайта. В таблице «page» хранятся данные и содержимое страницы, а в таблице «Nav» – названия меню, в таблице «NavItem» – пункты меню всех типов, описанных в предыдущей таблице. в таблице «News» – данные для отображающиеся в разделе новости.

Данные системы контроля доступа, используются как в подсистеме бронирования, так и в подсистеме управления контентом. Таблица «Users» хранит данные о пользователях (логин, пароль и т.п.). В таблице «Tokens» хранятся ключи для аутентификации пользователей с помощью cookie и сессий.

Рассмотрим подробно характеристики таблиц базы данных, используемых в веб-приложении. В таблице «Events» (таблица 9) хранится

информация о мероприятиях, которые выводятся на сайте и доступны для бронирования [2].

Таблица 9 – Таблица «Events» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	id	int(5)
2	Идентификатор места проведения	area_id	varchar(255)
3	Адрес файла картинки	image	varchar(255)
4	Название	title	varchar(255)
5	Время начала мероприятия	started_at	Timestamp
6	Время окончания мероприятия	finished_at	Timestamp
7	Описание	content	TEXT
8	Метка времени создания	created_at	Timestamp
9	Метка времени обновления	updated_at	Timestamp
10	Метка времени удаления	deleted_at	Timestamp

В таблице «Area» (таблица 10) хранится информация о местах проведения мероприятий.

Таблица 10 – Таблица «Area» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор Места проведения	id	int(5)
2	Тип места проведения	type	int(5)
3	Название	title	varchar(255)
4	Опции места проведения	options	Text
5	Метка времени создания	created_at	Timestamp
6	Метка времени последнего обновления	updated_at	Timestamp
7	Метка времени удаления	deleted_at	Timestamp

В таблице «Reserve» (таблица 11) хранятся непосредственно информация о бронировании.

Таблица 11 – Таблица «Reserve» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор записи	id	int(5)
2	Идентификатор пользователя	user_id	int(5)
3	Идентификатор мероприятия	event_id	int(5)
4	Опции	options	text
5	Метка времени создания	created_at	Timestamp
6	Метка времени последнего обновления	updated_at	Timestamp

Таблица «Pages» (таблица 12) хранит данные, которые отвечают за отображение содержимого любых страниц сайта, кроме тех, у которых имеется свой контроллер, реализующий бизнес-логику.

Таблица 12 – Таблица «Pages» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор страницы	id	int(5)
2	Адрес	url	varchar(255)
3	Содержимое страницы	content	TEXT
4	Название	title	TEXT
5	Заголовок на странице	content_header	TEXT
6	Содержимое тега keywords	metakey	TEXT
7	Содержимое тега description	metadesc	TEXT
8	Метка времени создания	created_at	Timestamp
9	Метка времени последнего редактирования	updated_at	Timestamp
10	Параметры доступа	access	int(10)
11	Параметры	params	TEXT

Таблица «Nav» (таблица 13) хранит в себе заголовок меню, а также идентификатор, который используется для взаимосвязи пунктов меню с соответствующей сущностью меню.

Таблица 13 – Таблица «Nav» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	id	int(5)
2	Название меню	name	varchar(255)
3	Заголовок отображаемый на странице	title	varchar(255)
4	Описание	desc	TEXT
5	Уровень доступа	access	int(5)
6	Метка времени создания	created_at	Timestamp
7	Метка времени последнего обновления	updated_at	Timestamp
8	Метка времени удаления	deleted_at	Timestamp

Таблица «NavItem» (таблица 14) используется для хранения пунктов меню, ссылок и сопутствующих данных всех меню сайта, сопоставляемых с объектом меню при помощи поля «nav_id».

Таблица 14 – Таблица «NavItem» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	id	int(5)
2	Тип	type	varchar(255)
3	Родительский id	parent_id	int(5)
4	Сортировка	ordering	int(9)
5	Название пункта	name	varchar(255)
6	Адрес ссылки	link	TEXT
7	Описание	desc	TEXT
8	Дата и флаг публикации	published_at	Timestamp
9	Параметры доступа	access	int(10)
10	Параметры	params	TEXT

В таблице «News» (таблица 15) хранятся данные новостей добавляемых в панели управления через специальную форму.

Таблица 15 – Таблица «News» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	id	int(5)
2	Метка времени создания	created_at	Timestamp
3	Идентификатор пользователя создателя	created_by	int(9)
4	Заголовок новости	title	varchar(255)
5	Краткое содержание	introtext	TEXT
6	Текст новости с HTML	text	TEXT
7	Флаг публикации	published	int(1)
8	Параметры доступа	access	int(10)
9	Параметры	params	TEXT

В таблице «Users» (таблица 16) хранятся все данные о пользователе, в том числе пара логин и пароль, необходимые для доступа пользователя к панели управления и различным дополнительным разделам сайта, доступ к которым определяется с помощью сравнения полей «access».

Таблица 16 – Таблица «Users» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	id	int(5)
2	Пароль	password	varchar(255)
3	E-mail	email	varchar(255)
4	Телефон	phone	varchar(255)
5	Состояние (подтвержден, забанен)	state	int(3)
6	Адрес файла изображения	picture	varchar(255)

Продолжение таблицы 16

7	Имя	first_name	varchar(255)
8	Отчество	patronymic	varchar(255)
9	Фамилия	last_name	varchar(255)
10	Дата рождения	DOB	Date
11	Дата регистрации	reg_date	Date
12	Обо мне	about	TEXT
13	Группа	groups	TEXT
14	Параметры доступа	access	int(10)
15	Параметры	params	TEXT
16	Метка времени создания	created_at	Timestamp
17	Метка времени последнего обновления	updated_at	Timestamp

В таблице «Tokens» (таблица 17) хранятся данные необходимые для аутентификации пользователей по ключу установленному в cookie браузера пользователя. А также для реализации одновременной авторизации с функцией запомнить сайте с возможностью отслеживания авторизаций аккаунта пользователя с других устройств.

Таблица 17 – Таблица «Tokens» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	id	int(5)
2	Ключ, записываемый в cookie	key	varchar(255)
3	Идентификатор пользователя	user_id	int(5)
4	IP-адрес	ip	varchar(255)
5	Содержимое строки user-agent	agent	TEXT
6	Метка времени создания	created_at	Timestamp
7	Метка времени последнего обновления	updated_at	Timestamp

Во всех описанных выше таблицах используется поле «deleted_at» для реализации механизма псевдо удаления, т.е. исключения из результатов выдачи запросов без нарушения целостности базы данных, а также возможности отмены удаления и восстановления [46].

Для взаимодействия клиентской части приложения с API используется тип данных JSON [51].

4 Реализация системы бронирования

4.1 Реализация системы бронирования

Функциональное назначение разрабатываемого веб-приложения – предоставление сервиса бронирования экскурсий и развлекательных мероприятий, а также формирование интерфейса, структуры и содержимого сайта.

Автоматизированная информационная система имеет следующее функции:

– бронирование экскурсий и мероприятий;

– управление пользователями и контроль доступа к любым разделам сайта;

– система управления структурой содержимого сайта.

Назначение системы:

– предоставление инструмента для создания приложения системы бронирования экскурсий и различных мероприятий;

– создание представительства организации в сети Интернет с возможностью формирования содержимого сайта для информирования посетителей о деятельности.

Для реализации программного продукта используется платформа Linux с веб-сервером Apache. Язык программирования PHP, желательно версии 7.2, т.к. данная версия работает быстрее и стабильнее, система управления базами данных MySQL версии 5.5 и выше для хранения данных системы. Предпочтительно использование операционной системы Linux. В основе веб-приложения используется шаблон программирования MVC. То есть модель данных, представление и бизнес-логика, разделены на отдельные компоненты.

Разрабатываемое веб-приложение «Система бронирования экскурсий и развлекательных мероприятий» представляет собой автоматизированную информационную систему на базе веб-интерфейса с интегрированной

подсистемой управления контентом, подсистемой контроля доступа и подсистемой бронирования.

Для работы с данными, хранящимися в СУБД MySQL используется ORM Eloquent. ORM от англ. Object-Relational Mapping, в переводе на русский – объектно-реляционное отображение. Eloquent – это легкая и функциональная реализация шаблона Active record для работы с базами данных с открытым исходным кодом [46]. В данной ORM каждая таблица имеет соответствующий класс-модель, наследующий родительский класс Model, который используется для работы с этой таблицей. За формирование представления отвечает шаблонизатор Twig. Его отличительными особенностями являются: высокая скорость выполнения его реализации для PHP, за счет применения компиляции и кэширования, открытый исходный код и реализации для всех популярных языков программирования значительные возможности по расширению синтаксиса и добавления своих функций [42].

Информационная система имеет следующую структуру файлов:

– Application – папка в которой хранится вся серверная часть приложения;

– assets – в папке хранятся все файлы (css, js итд) клиентской части приложения;

– config – в папке хранится конфигурационные файлы приложения, в том числе список маршрутов веб-приложения (приложение Ж);

– media – в папке хранится вся медиафайлы (иллюстрации, документы и видео) используемые в приложении;

– templates – в папке хранятся файлы шаблонов страниц, а также их вспомогательные файлы;

– storage – в папке хранятся временные файлы серверной части приложения;

– vendor – в папке хранится все зависимые библиотеки и фреймворки серверной части приложения;

– .htaccess – конфигурационный файл веб-сервера Apache (приложение В);

– composer.json – конфигурационный файл composer;

– index.php – единая точка входа в веб-приложения (приложение Г).

Основные файлы приложения хранятся в папке Application. Она имеет следующую структуру:

а) Classes – в папке хранятся файлы дополнительных классов необходимых для работы приложения;

б) Controller – в папке хранятся файлы с классами контроллеров;

в) Exception – в папке хранятся файлы исключений;

г) Model – в папке хранятся модели данных, в виде файлов моделей ORM Eloquent:

- Area.php – файл модели места проведения;
- Events.php – файл модели мероприятие;
- Navs.php – файл модели навигационное меню;
- NavItems.php – файл модели элемент навигационного меню;
- News.php – файл модели новость;
- Page.php – файл модели страница;
- Reserve.php – файл модели записи о бронировании;
- Tokens.php – файл модели место проведения;
- Users.php – файл модели пользователь;

д) Module – в папке хранится файлы модулей для подключения в приложении:

- Booking.php – файл модуля бронирования реализующий резервирование, редактирование и удаление бронирования (приложение И);
- Eloquent.php – файл инициализирующий класс Eloquent, предоставляющие доступ к базе данных посредствам ORM;
- ErrorHadler.php – файл модуля обработчика ошибок веб-приложения;
- Router.php – класс модуля маршрутизации (приложение Е);

– Twig.php – файл инициализирующий класс Twig предоставляющий доступ к функциям шаблонизатора;

– User.php – файл модуля пользователь реализующий аутентификацию, идентификацию и авторизацию пользователей (приложение К);

е) Twig – в папке хранятся файлы расширения возможностей шаблонизатора Twig;

ж) Application.php – главный файл приложения, содержит класс Application (приложение Д), служащий контейнером для всех объектов и предоставляющий доступ к объекту Router, который подгружает классы из папки Controller (компонент MVC «контроллер») ORM Eloquent (компонент MVC «модель»), модулю пользователя и шаблонизатору Twig (компонент MVC «представление»).

Веб-приложение имеет зависимости со следующими классами сторонних библиотек:

– Symfony\Component\HttpFoundation\Request – содержит информацию о запросе клиента;

– Symfony\Component\HttpFoundation\Response – содержит всю необходимую информацию, для отправки ответа пользователю, на его запрос;

– Symfony\Component\HttpFoundation\RedirectResponse – используется для отправки ответа с заголовками для переадресации запроса;

– Illuminate\Database\Capsule\Manager – инициализирует подключение к БД MySQL и запускает Eloquent;

– Illuminate\Database\Eloquent\Model – основной класс моделей Eloquent, предоставляет возможности для получения записей из СУБД MySQL в виде объектов PHP и позволяет обращаться к полям, как к свойствам объекта [46];

– \Twig_Loader_Filesystem – загружает шаблоны из файловой системы;

– \Twig_Environment – используются для хранения конфигурации и расширений и используются для загрузки шаблонов из файловой системы или других мест [42].

Компонент библиотеки Symfony HttpFoundation – позволяет взаимодействовать с HTTP через объектно-ориентированный подход.

В PHP, запрос представлен, как совокупность глобальных переменных ($_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, …), а ответ формируется с помощью некоторых функций (header(), echo(), setcookie(), …)

Компонент Symfony HttpFoundation – заменяет упомянутые глобальные переменные и функции, предоставляя объектно-ориентированный слой.

Объект Request содержит информацию о запросе клиента. Ее можно получить через определенные публичные (public) свойства [44]:

– headers: в основном содержит часть из массива «$_SERVER», например «$request->headers->get('User-Agent')»;

– query: аналог «$_GET, например «$request->query->get('name')»;

– request: содержит информацию из «$_POST»;

– cookies: содержит информацию из массива «$_COOKIE»;

– attributes: собственное свойство – используется приложением для хранения данных;

– files: аналог «$_FILES»;

– server: «$_SERVER».

Объект Response. Содержит всю необходимую информацию, для отправки ответа пользователю, на его запрос. Конструктор принимает до трех аргументов: контент, код статуса и массив с HTTP заголовками [44].

ORM Eloquent – реализация паттерна «ActiveRecord» для работы с базой данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют читать данные из таблиц и записывать данные в таблицу [46]. Все модели в папке Model расширяют класс Illuminate\Database\Eloquent\Model. В приложении М предоставлены классы моделей данных ORM Eloquent.

Шаблонизатор Twig. Twig использует центральный объект, называемый environment (класс Twig_Environment) [42]. Экземпляры этого класса используются для хранения конфигурации и расширений и используются для загрузки шаблонов из файловой системы или других мест.

Далее представлена спецификация для основных классов приложения:

а) класс Container, имплементирует интерфейс ArrayAccess и позволяет обращаться к экземпляру объекта, как к массиву, имеет следующие методы(функции) [28]:

– __construct () – определяет, существует ли заданное смещение;

– offsetSet ($offset, $value) – присваивает значение заданному смещению заданное значение, где $offset – смещение (ключ), $value – значение;

– offsetExists ($offset) – определяет, существует ли заданное смещение, где $offset – смещение (ключ);

– offsetUnset ($offset) – удаляет заданное смещение, где $offset – смещение (ключ);

– offsetGet ($offset) – возвращает заданное смещение, где $offset – смещение (ключ).

б) класс Router – реализует маршрутизатор, определяющий соответствия пути и контроллера, содержит следующие методы (приложение Е):

– __construct (Application $app) – конструктор объекта маршрутизатора, где $app – это экземпляр объекта класса Application;

– match ($url) – запускает поиск контроллера для адреса, где $url – адрес, по умолчанию используется текущий адрес;

– add ($method, $string, $action, $vars = null) – добавляет соответствие регулярному выражению и контроллера;

– addFromArray ($path) – добавляет массив регулярных выражений и контроллеров, где $path – адрес файла (приложение Ж) хранящего массив соответствий регулярных выражений пути и контроллера;

– callAction ($route) – вызывает обработчик маршрута, где $route – массив с информацией о контроллере и значениях шаблонов регулярного выражения.

– getResponse () – возвращает ответ контроллера.

в) класс Booking – модуль бронирования, содержит следующие методы (приложение И):

– get ($reserve_id) – возвращает подробную информацию записи о бронировании, где $reserve_id – идентификатор записи о бронировании;

– all ($user_id, $with_events) – возвращает список всех забронированных мест, где $user_id – идентификатор пользователя по умолчанию используется текущий пользователь, $with_events – флаг определяющий добавление описания мероприятия к каждой записи, по умолчанию описание не передается;

– create ($event_id, $qty, $options, $user_id) – создает запись о бронировании места, где $event_id – идентификатор мероприятия, $qty – количество билетов, $options – опции, $user_id – идентификатор пользователя на которого осуществляется бронирование;

– update ($reserve_id, $qty, $options, $user_id) – обновляет информации в записи о бронировании, где $reserve_id – идентификатор записи о бронировании, $qty – количество билетов, $options – опции, $user_id – идентификатор пользователя на которого осуществляется бронирование;

– remove ($reserve_id) – удаляет запись о бронировании, где $reserve_id – идентификатор записи о бронировании.

г) класс User – модуль, реализующий подсистему контроля доступа и функции, связанные с пользователями, содержит следующие методы (приложение Ж):

– getUser () – возвращает информацию о текущем авторизованном пользователе;

– login ($login, $password, $remember) – реализует авторизацию пользователя, где $login – логин пользователя, $password – пароль

пользователя, $remember – флаг запомнить на этом устройстве, по умолчанию ложь;

– logout ($key) – выполняет выход из сессии, где $key – идентификатор сессии;

– join ($data) – добавляет нового пользователя, где $data – массив информации о пользователе;

– online () – помечает текущего пользователя как находящегося в сети.

д) класс ObjectModel – модуль, реализующий функции, связанные с CRUD-действиями со всеми объектами в панели управления через API, содержит следующие методы (приложение Л):

– create () – создает новую запись в БД с помощью моделей ORM Eloquent;

– read () – возвращает данные соответствующей записи в БД с помощью моделей ORM Eloquent;

– update () – обновляет запись в БД с помощью моделей ORM Eloquent;

– delete () – удаляет запись в БД с помощью моделей ORM Eloquent;

– schema () – возвращает имена, тип данных и описание полей записи, в формате JSON Schema.

Остальные классы контроллеров, такие как: Home, Events, News, Pages, User, Panel\Home, Panel\Area, Panel\Booking, Panel\Events, Panel\Menu, Panel\News, Panel\Pages, Panel\Users имеют схожую архитектуру и выполняют обработку действий с объектами, одноименными с названием класса.

На рисунке 19 представлена диаграмма классов веб-приложения.

При запросе других маршрутов веб-приложения взаимодействие классов происходит аналогично.

Рисунок 19 – Диаграмма классов серверной части веб-приложения

За формирование интерфейса веб-приложения отвечают шаблоны, расположенные в папке templates.

Все шаблоны наследуются от базового шаблона base.html – для страниц сайта и /panel/base.html – для страниц панели управления. В файле base.html подключаются все необходимые файлы стилей и js-скриптов, а также определены основные блоки страницы, такие как хэдер, навигационное меню, область контента, футер.

Далее представлена схема взаимодействия классов при выполнении типового действия – просмотра главной страницы (рисунок 20).

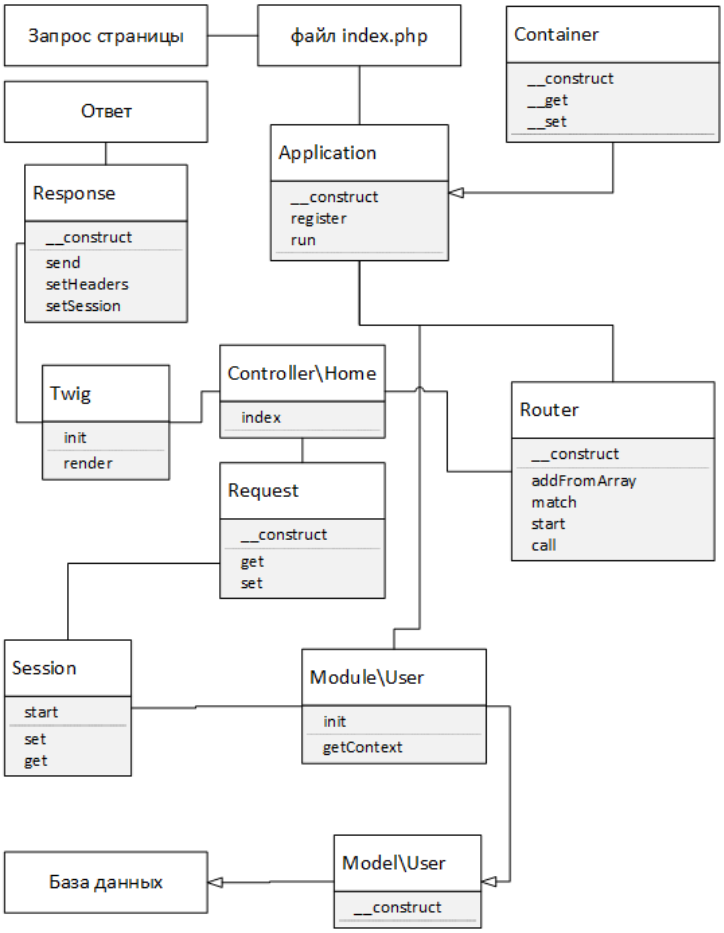

Рисунок 20 – Схема взаимодействия файлов при выполнении загрузки главной страницы

На рисунке 21 представлена главная страница веб-приложения сформированная на основе базового шаблона.

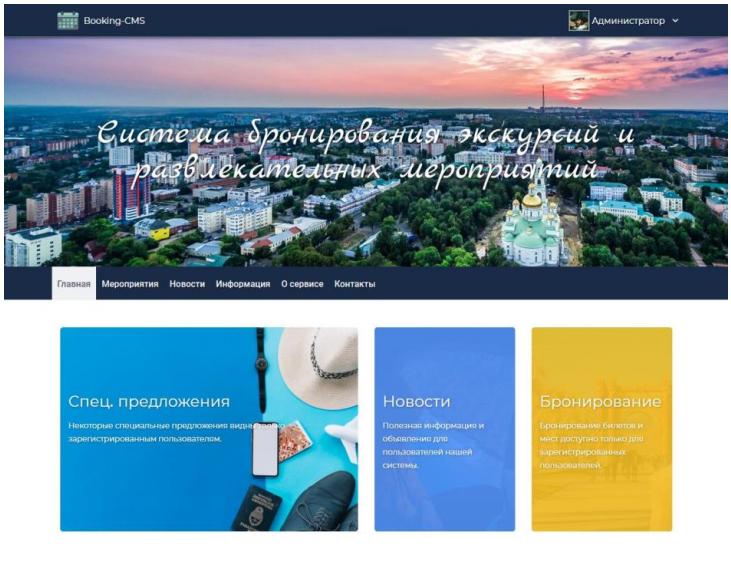

Рисунок 21 – Скриншот главной страницы

В приложении Р представлены скриншоты разработанного веб-приложения.

4.2 Развёртывание веб-приложения

Для работы веб-приложения система бронирования необходим веб-сервер с поддержкой языка программирования PHP, а также реляционная база данных, с поддержкой языка запросов SQL. Рекомендуется использование операционной системы Linux CentOS версии 7. Однако веб-приложение может также работать под Windows, macOS и любой другой операционной системой, где возможна установка веб-сервера и интерпретатора PHP.

В качестве веб-сервера рекомендуется использовать свободно распространяемый веб-сервер Apache, поскольку в веб-приложении для создания единой точки входа используется конфигурационный файл htaccess. Однако, при небольшой доработке можно использовать другой веб-сервер, например, nginx и т.д.

В качестве системы управления баз данных рекомендуется использовать MySQL версии 8, так как при разработке тестировалась именно эта СУБД и работа с другой не гарантируется.

В качестве клиента приложения выступает любой веб-браузер. Поддерживаются все современные версии и браузеры, то есть разработанное приложение кросс-браузерно и кроссплатформенно.

Для установки и использования системы бронирования необходимо поместить все файлы веб-приложения в папку домена веб-сервера. Настроить подключение к базе данных, а также импортировать файл с дампом базы в формате SQL.

На рисунке 22 представлена диаграмма развёртывания веб-приложения система бронирования экскурсий и развлекательных мероприятий. Основные узлы – это веб-сервер, сервер баз данных и веб-клиент, взаимодействие происходит по протоколу TCP/IP.

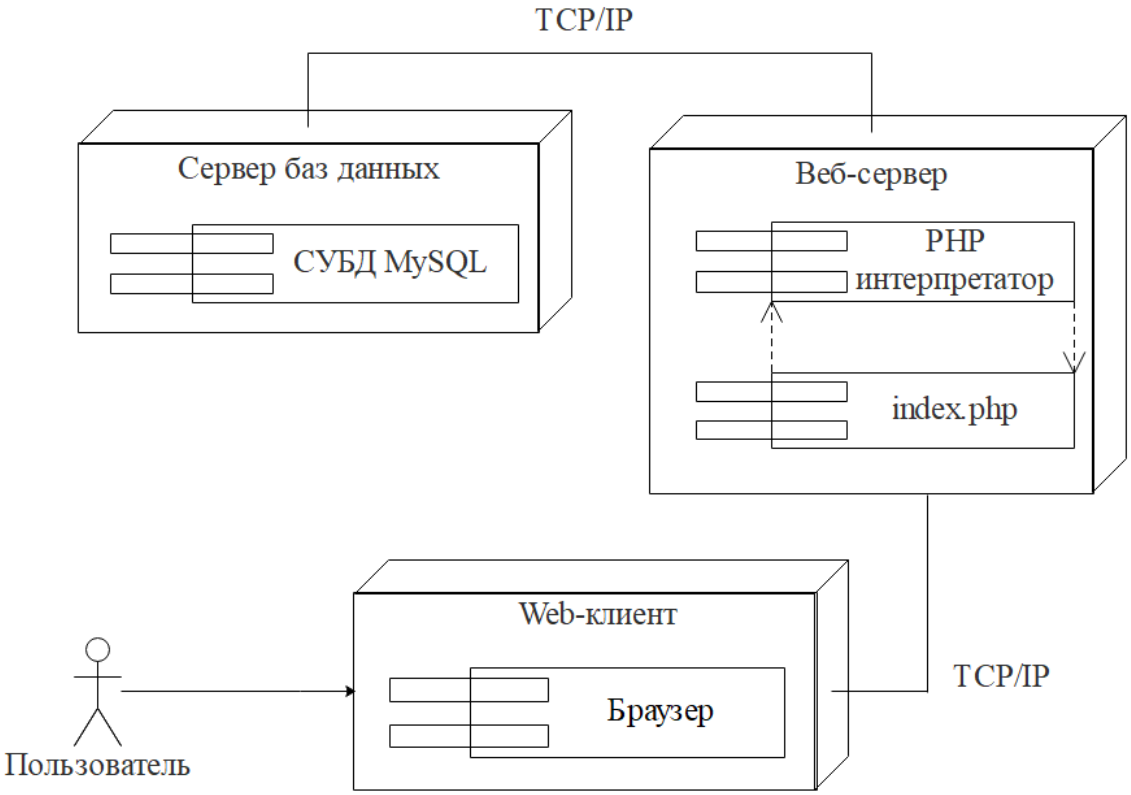

Рисунок 22 – Диаграмма развёртывания веб-приложения

На данной диаграмме развёртывания показана конфигурация обрабатывающих узлов на которых выполняется разработанное веб-приложение и его компоненты, размещенные в этих узлах.

5 Анализ качества программного продукта

5.1 Тестирование

Для проверки качества разработанного программного продукта применим три основные методологии тестирования: функциональное, структурное и модульное тестирование.

Проанализировав варианты использования и функциональные требования, был подготовлен список наиболее возможных сценариев использования сайта и системы управления содержимым сайта, который реализует требуемые от программного продукта функции:

1) проверка загрузки обычной страницы;

2) проверка загрузки несуществующей страницы;

3) проверка возможности регистрации;

4) проверка возможности авторизации;

5) проверка возможности просмотра информации о мероприятии;

6) проверка возможности бронирования мест на мероприятие авторизованным пользователем;

7) проверка входа на сайт под логином администратора;

8) проверка отображения списка мероприятий;

9) проверка возможности добавления мероприятия;

10) проверка возможности редактирования мероприятия;

11) проверка возможности отмены брони;

12) проверка загрузки файлового менеджера.

На основе этого был составлен план тестирования клиентской части веб-приложения и выполнено функциональное тестирование, результат представлен далее в таблице (таблица 18). В приложении Р представлены скриншоты, выполненные во время тестирования веб-приложения.

Таблица 18 – План тестирования веб-приложения

№	Названия тестового случая	Тест	Результат
1	Проверка загрузки обычной страницы	Будет загружена страница сайта	Была загружена страница сайта
2	Проверка загрузки несуществующей страницы	Будет загружена страница сайта с сообщением ошибки и HTTP кодом ответа 404	Была загружена страница сайта с сообщением ошибки и HTTP кодом ответа 404
3	Проверка возможности регистрации	Пользователь будет зарегистрирован	Пользователь был зарегистрирован
4	Проверка возможности авторизации	При вводе правильных данных для входа произойдет авторизация	При вводе правильных данных для входа была произведена авторизация
5	Проверка возможности просмотра информации о мероприятии	Страница с информацией будет загружена	Страница с информацией был загружена
6	Проверка возможности бронирования мест на мероприятие	Место будет забронировано	Место было забронировано
7	Проверка входа на сайт под логином администратора	При вводе правильных данных для входа произойдет авторизация	При вводе правильных данных для входа была произведена авторизация
8	Проверка отображения списка мероприятий	Должен быть отображен список мероприятий	Был отображен список мероприятий
9	Проверка возможности добавления мероприятия	Будет доступен интерфейс для изменения пунктов меню	Был доступен интерфейс для изменения пунктов меню, были изменены пункты меню
10	Проверка возможности редактирования мероприятия	Информация о мероприятие должна быть изменена	Информация о мероприятие была изменена
11	Проверка возможности отмены брони	Забронированное место должно стать доступным	Бронь отменена. Забронированное место стало доступным
12	Проверка загрузки файлового менеджера	Файловый менеджер должен загрузиться	Файловый менеджер загружается и доступен

Поскольку разрабатываемое веб-приложение состоит из модулей, то возможно выполнить автоматизированное модульное тестирование основных модулей.

Веб-приложение «Система бронирования экскурсий и развлекательных мероприятий» разрабатывается с использованием нескольких языков программирования. Можно выделить серверную и клиентскую часть. Клиентская часть работает на языке программирования JavaScript с

использованием фреймворка Vue.js. Серверная часть написана на языке программирования PHP на основе паттерна MVC с применением фреймворка Symfony и системы объектно-реляционного отображения (ORM) Eloquent.

Для тестирования разрабатываемого веб-приложения возможно использовать различные инструменты, остановимся на них подробнее.

Клиентскую часть приложения, написанную на JavaScript с применением фреймворка Vue.js эффективнее всего тестировать с использованием Vue Test Utils, поскольку она является частью фреймворка [45]. Также для тестирования клиентской части можно использовать Jest и Mocha.

Для серверной части приложения, разрабатываемой на PHP можно использовать PhpUnit. А для тестирования ответов API подходит инструмент Selenium, в частности модуль Selenium WebDriver предоставляющий возможность автоматизации действий веб-браузера.

На основе функциональных требований были выделены два основных модуля приложения для тестирования:

- подсистема контроля доступа;
- подсистема бронирования.

На основе вариантов использования для подсистемы контроля доступа был составлен список тестовых случаев:

- регистрация пользователя;
- проверка авторизации;
- авторизация на сайте с помощью логина;
- получение идентификатора для текущего пользователя;
- получение информации о авторизованном пользователе;
- получение токена текущей сессии;
- установка статуса в сети;
- выход.

Используя метод покрытия условий был составлен список тестов для вариантов использования подсистема контроля доступа, представленный далее в таблице (таблица 19).

Таблица 19 – Тесты подсистемы контроля доступа

№	Метод класса	Тест	Результат	Тест пройден
1	User::isAuth	Проверка аутентификации Для неавторизованного пользователя	Ответ отрицательный	+
2	User::login	Проверка ответа при вводе несуществующего логина	Выброшено исключение	+
3	User::login	Проверка ответа при вводе неправильного пароля	Выброшено исключение	+
4	User::login	Проверка входа при вводе верных данных	Успешный вход	+
5	User::isAuth	Проверка аутентификации Для авторизованного пользователя	Ответ положительный	+
6	User::login	Проверка авторизации с функцией запомнить	Успешный вход	+
7	User::isAuth	Проверка авторизации для пользователя с функцией запомнить при истекшей сессии PHP	Успешная авторизация	+
8	User::getUID	Запрос идентификатора для авторизованного пользователя	Был получен идентификатор пользователя	+
9	User::getToken	Запрос токена для текущей сессии пользователя	Получен токен	+
10	User::getUser	Запрос информации о пользователе	Получен массив с информацией о пользователе	+
11	User::online	Обновление статуса в сети каждые 15 минут	Обновление не требуется	+
12	User::online	Принудительное обновление статуса в сети	Установлено текущее время	+
13	User::logout	Проверка выхода из аккаунта	Пользователь успешно разлогинен	+
14	User::join	Проверка регистрации пользователя с указанием уже существующего логина	Отказ в регистрации	+
15	User::join	Проверка регистрации пользователя с указанием уже свободного логина	Новый пользователь успешно зарегистрирован	+

На основе составленных тестов было создано 15 unit-тестов на языке программирования PHP с использованием библиотеки PHPUnit, листинг кода тестов представлен в приложении Н.

После было запущено автоматизированное модульное тестирование, результаты которого представлены на рисунке 23.

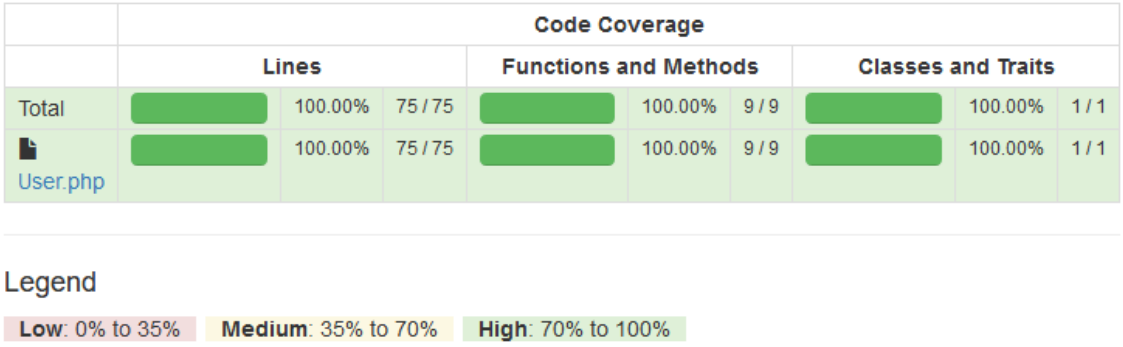

Рисунок 23 – Результат выполнения тестов подсистемы контроля доступа

А также проведен анализ покрытия, результат представлен на рисунке 24.

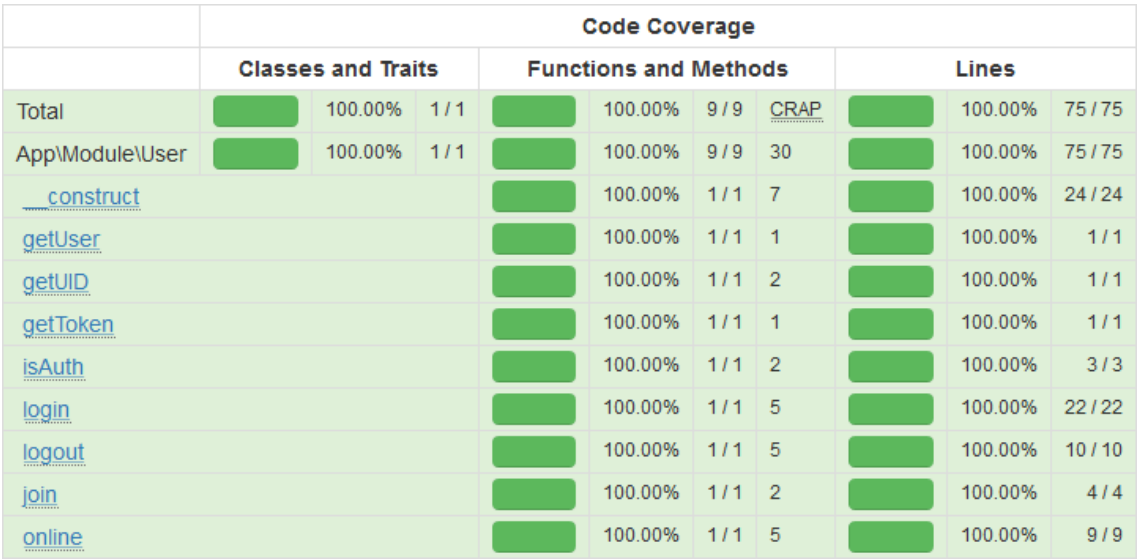

Рисунок 24 – Результат анализа покрытия кода подсистемы контроля доступа

Проанализировав полученные результаты можно сделать вывод, что были протестированы все возможные варианты использования подсистемы контроля доступа. Все функциональные требования выполнены на хорошем

уровне, о чем свидетельствует отсутствие ошибок при модульном тестировании и 100% покрытие кода тестами.

Проведем тестирование подсистемы бронирования экскурсий и различных мероприятий. На основе вариантов использования был составлен список необходимых тестов:

- бронирование мероприятия или экскурсии;
- получение списка забронированных мероприятий для пользователя;
- получение подробной информации про запись о бронировании;
- изменение опций записи о бронировании;
- отмена бронирования.

Используя метод покрытия решений и метод эквивалентных разбиений был составлен список тестов для вариантов использования подсистемы бронирования экскурсий и развлекательных мероприятий, представленный в таблице 20.

Таблица 20 – Основные тесты подсистемы бронирования

№	Метод класса	Тест	Результат	Тест пройден
1	Booking::all	Проверка запроса всех записей о бронировании для неавторизованного пользователя	Выброшено исключение	+
2	Booking::all	Проверка запроса всех записей о бронировании	Получен список мероприятий	+
3	Booking::get	Проверка запроса информации записи о бронировании для неавторизованного пользователя	Выброшено исключение	+
4	Booking::get	Проверка запроса информации для несуществующей записи о бронировании	Выброшено исключение	+
5	Booking::get	Проверка запроса информации для записи о бронировании	Успешный вывод информации	+
6	Booking::create	Проверка создания записи о бронировании для неавторизованного пользователя	Выброшено исключение	+

Продолжение таблицы 20

7	Booking::create	Проверка создания записи о бронировании для несуществующего мероприятия	Выброшено исключение	+
8	Booking::create	Проверка создания записи о бронировании	Запись успешно создана	+
9	Booking::update	Проверка обновления информации для неавторизованного пользователя	Выброшено исключение	+
10	Booking::update	Проверка обновления информации для несуществующей записи	Выброшено исключение	+
11	Booking::update	Проверка обновления записи о бронировании	Запись успешно отредактирована	+
12	Booking::remove	Проверка удаления для неавторизованного пользователя	Выброшено исключение	+
13	Booking::remove	Проверка удаления для неавторизованного пользователя	Выброшено исключение	+
14	Booking::remove	Проверка удаления для неавторизованного пользователя	Запись успешно удалена	+

На основе составленных тестов было создано 29 unit-тестов на языке программирования PHP с использованием библиотеки PHPUnit, листинг кода тестов представлен в приложении П.

После чего было запущено автоматизированное модульное тестирование, результаты которого представлены на рисунке 25.

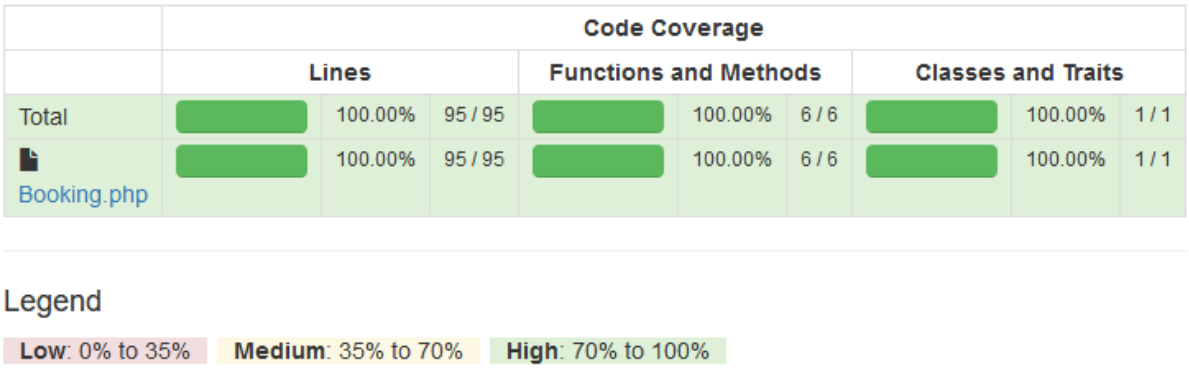

Рисунок 25 – Результат выполнения тестов для модуля бронирования

А также проведен анализ покрытия, результат представлен на рисунке 26.

	Code Coverage								
	Classes and Traits			Functions and Methods				Lines		
Total		100.00%	1 / 1		100.00%	6 / 6	CRAP		100.00%	95 / 95
App\Module\Booking		100.00%	1 / 1		100.00%	6 / 6	42		100.00%	95 / 95
__construct					100.00%	1 / 1	1		100.00%	3 / 3
get					100.00%	1 / 1	4		100.00%	9 / 9
all					100.00%	1 / 1	7		100.00%	14 / 14
create					100.00%	1 / 1	12		100.00%	30 / 30
update					100.00%	1 / 1	8		100.00%	17 / 17
remove					100.00%	1 / 1	10		100.00%	22 / 22

Рисунок 26 – Результат анализа покрытия кода модуля бронирования

Проанализировав полученные результаты тестов для подсистемы бронирования, можно сделать вывод, что были протестированы все возможные варианты использования. Все функциональные требования выполнены на хорошем уровне, о чем свидетельствует отсутствие ошибок при модульном тестировании и 100% покрытие кода тестами.

Подводя итог можно сказать, что все подсистемы веб-приложения успешно прошли тестирование и соответствуют заявленным требованиям.

5.2 Метрики кода

Серверная часть программного продукта веб-приложение «Система бронирования экскурсий и развлекательных мероприятий» разработан на языке программирования PHP, поэтому принято решение и использовать программное средство PhpMetrics для автоматизированного расчета метрик кода [54].

Для проверки качества кода был выбран показатель качества кода (Maintainability Index).

Рассчитывается метрика по следующей формуле (9):

$$\mathrm{MI} = \mathrm{MAX}\left(0, (171 - 5.2 * \ln(\mathrm{HV}) - 0.23 * \mathrm{CC} - 16.2 * \ln(\mathrm{LoC})) * \frac{100}{171}\right), \quad (9)$$

где HV – Halstead Volume, вычислительная сложность, чем больше операторов, тем больше значение этой метрики;

CC – Cyclomatic Complexity. показывает структурную сложность кода, т.е. количество различных ветвей в коде, чем больше этот показатель, тем больше тестов должно быть написано, для полного покрытия кода;

LoC – количество строк кода.

Комплексный показатель качества кода может принимать значения от 0 до 100 и показывает относительную сложность поддержки кода. Чем больше значение этой метрики, тем легче поддерживать код.

После применения автоматизированного программного средства PhpMetrics для анализа и расчёта метрик были получены следующие показатели, представленные в таблице 21.

Таблица 21 – Метрики кода для разработанного веб-приложения

Метрика	Значение
Размерно-ориентированные метрики	
Количество строк кода	2549
Логических строк кода	2322
Количество строк комментариев	227
Средний объём	411.46
Средний объём комментариев	9.88
Объектно-ориентированные метрики	
Количество классов	60
Количество методов	185
Сложность кода	
Средняя цикломатическая сложность по классу	5.53
Средневзвешенный метод подсчета по классам	8
Средняя относительная сложность системы	41.32
Средняя сложность	5.23
Средний показатель качества кода	69,15

А также получен показатель качества кода для каждого файла в веб-приложении (таблица 22). Компоненты отсортированы по частоте использования.

Таблица 22 – Показатель качества кода основных компонентов

№	Компонент	MI
1	App/Application	50,38
2	App/Exception/Exception	70,71
3	App/Exception/Auth/AuthenticationException	97,82
4	App/Controller/Panel/BasePanel	69,9
5	App/Model/Token	62,13
6	App/Module/User	53,93
7	App/Module/Twig	58,73
8	App/Module/Router	53,89
9	App/Module/ErrorHandler	53,94
10	App/Module/Eloquent	61,28
11	App/Exception/HttpException	56,71
12	App/Module/Breadcrumbs	93,23
13	App/Model/User	54,82
14	App/Model/Area	60,28
15	App/Model/Events	60,74
16	App/Model/News	72,37
17	App/Model/Page	66,96
18	App/Model/Reserve	59,01
19	App/Module/Cache/Cache	72,71
20	App/Module/Cache/FilesystemCache	62,33
21	App/Module/Booking	66,71
22	App/Exception/Booking/BookingException	95,15
23	App/Exception/Auth/UsernameNotFoundException	85,34
24	App/Controller/Panel/Auth	75,94
25	App/Controller/API/ObjectModel	60,07
26	App/Controller/Errors/Errors	65,45
27	App/Controller/Home	88,27
28	App/Exception/Auth/UsernameBusyException	85,34
29	App/Exception/Auth/AccessDeniedException	94,63
30	App/Exception/Auth/BadCredentialsException	65,65
	Средний показатель качества кода проекта	69,15

Подводя итог, можно сказать, что проект имеет средний показатель качество кода. Поддержка разработанного продукта другими разработчиками возможна.

Заключение

Проведя анализ современных сервисов и систем управления содержимым сайта с функцией бронирования, было установлено, что системы, которые имеют возможность интеграции системы бронирования, крайне сложны в обращении и имеют очень сложный интерфейс. В результате чего потенциальные пользователи таких систем будут испытывать трудности в работе с ними. Кроме того, на сегодняшний день, нет ни одной достаточно функциональной и надежной CMS, предназначенной непосредственно для работы в сфере бронирования экскурсий, что делает их еще более неудобными для ведения соответствующей деятельности. Кроме того, подавляющее большинство систем управления содержимым, обладающих необходимой для реализации системы бронирования экскурсий и развлекательных мероприятий функциональностью, являются платными и имеют фиксированный набор компонентов.

Была рассмотрена и проанализирована актуальная техническая литература по проектированию и разработке информационных систем для сети Интернет. Теоретический анализ литературы показал, что проблема создания многофункциональных систем управления содержимым сайта рассматривалась достаточно широко. В то же время целый ряд вопросов, связанных с разработкой систем бронирования мероприятий остается мало освещенным.

В ходе анализа предметной области были исследованы протекающие бизнес-процессы при использовании систем бронирования. Определены основные варианты использования, проанализированы элементы систем бронирования и их характеристики. Проведен поиск методических указаний по разработке и использованию систем бронирования. В результате чего проанализированы наиболее важные работы отечественных и зарубежных авторов, в результате чего был сделан вывод, что тема разработки системы бронирования развлекательных мероприятий освещена мало. Также стоит отметить отсутствие методических указаний по разработке таких систем, нет

сравнительного анализа инструментов и способов реализации веб-приложений систем бронирования.

После чего были изучены аналогичные программные продукты, для того, чтобы оценить необходимые характеристики подобных систем и учесть недостатки существующих систем. Были определены критерии и проведен сравнительный анализ аналогов. В результате сравнения сделан вывод, что лидерами являются системы управления содержимым сайта «Booking Core» и «Ag Booking CMS». Поэтому при разработке веб-приложения необходимо ориентироваться на данные аналоги.

После чего был проведен анализ вариантов использования, составлена диаграмма деятельности, а также диаграммы последовательности действий для прецедентов бронирование билетов на мероприятие и администрирование в панели управления

На основе вариантов использования была определена структура сайта, наиболее подходящая для выполнения поставленной задачи. Проведя анализ способов создания систем бронирования на сайте было установлено, что функция бронирования, чаще всего является частью сайта и для ведения таких сайтов используются системы управления содержимым. Были определены функциональные и дополнительные требования, а также составлено техническое задание на разработку веб-приложения.

На основе этого было проведено планирование проекта, определена длительность работ и их взаимосвязь. Составлена контекстная диаграмма бизнес-процессов и проведена их декомпозиция. На основе этого составлен план стадий и этапов проектирования и разработки веб-приложения, содержащий подробную информацию длительности этапов разработки и лицах их выполняющих. На основе плана была составлена диаграмма Ганта. Проанализировав этапы и их продолжительность было установлено, что на проектирование и разработку программного продукта потребуется 418 часов. На основе планирования был произведен расчет технологической себестоимости проекта. В процессе расчетов были определены такие статьи расходов, как амортизация, фонд заработной платы, социальные выплаты,

материальные и прочие расходы. По итогам расчетов была составлена диаграмма структуры расходов и определены основные статьи расходов, это фонд оплаты труда и социальные выплаты. Затраты на проектирование и разработку программного продукта «Веб-приложение бронирования экскурсий и развлекательных мероприятий» составляют 166 687 рублей.

На основе технического задания было проведено проектирование программных средства и структур хранения данных для разрабатываемой системы. Разработаны макеты пользовательского интерфейса для общей и административной частей приложения. Рассмотрена организация процесса обработки информации в приложении, составлена диаграмма компонентов. Были спроектированы структуры хранения данных для всех объектов базы данных приложения: страницы, меню, элементы меню, пользователи, токены доступа, мероприятия, места проведения и записи о бронировании. Описана логическая и физическая структура базы данных и составлена диаграмма классов базы данных.

На основе спроектированной модели было реализовано веб-приложения учитывающее специфику работы системы бронирования экскурсий и развлекательных мероприятий. Описана архитектура приложения, составлена спецификация основных классов и методов, таких как контейнер «ArrayAccess», маршрутизатор приложения, модуль контроля доступа, модуль бронирования и модуль осуществления CRUD-операций со всеми объектами базы данных. А также описаны все зависимости от сторонних библиотек. Разработанный программный продукт предоставляет возможность ведения сайта экскурсионной или туристической фирмы. Веб-приложение имеет дружественный пользовательский интерфейс, позволяет администратору размещать и редактировать страницы, новости, и мероприятия. Имеет подсистему бронирования, позволяющую пользователям бронировать места на экскурсии. А встроенная система контроля доступа позволяет ограничивать доступ к закрытым частям сайта.

Для проверки качества разработанного программного продукта были применены три основные методологии тестирования: функциональное,

структурное и модульное тестирование. Составлено и успешно проведено 44 модульных теста. В результате анализа полученных результатов тестов для подсистем контроля доступа и бронирования, сделан вывод, что были протестированы все возможные варианты использования. Все функциональные требования выполнены успешно, о чем свидетельствует отсутствие ошибок при модульном тестировании и 100% покрытие кода тестами. Таким образом можно сказать, что все подсистемы веб-приложения успешно прошли тестирование и соответствуют заявленным требованиям.

Подводя итог можно сказать, что поставленная цель работы достигнута. Было спроектировано и разработано веб-приложение система бронирования экскурсий и развлекательных мероприятий. В рамках данного проекта был разработан необходимый минимум, однако в системе имеется потенциал для создания дополнительных модулей и дальнейшего развития.

Список использованных источников

1 Пьюривал, С. Основы разработки веб-приложений / С. Пьюривал – СПб.: Питер, 2015. – с. 272.

2 Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 3-е издание. / Р. Никсон – СПб.: Питер, 2015. – 496 с.

3 eLIBRARY.RU - Научная электронная библиотека [Электронный ресурс] URL: https://www.elibrary.ru/. (дата обращения: 19.04.2020).

4 Чековитов, М.С. Информационная система для сервиса бронирования и покупки туристических путевок по хабаровскому краю / М.С. Чековитов, Е.А. Шеленок // Информационные технологии XXI века сборник научных трудов – Хабаровск, 2017. – с. 439-442.

5 Борсук, Н.А. Разработка модуля администрирования системы on-line бронирования / Н.А. Борсук // Международный научный журнал «Инновационная наука» – Уфа: Аэтерна, 2016. – № 5. – с. 25-27.

6 Бунина, К.А. Анализ современных компьютерных систем бронирования и резервирования / К.А. Бунина, А.Н. Ермакова // Приоритетные направления развития науки и образования № 3 – Чебоксары: Интерактив плюс, 2014. – с. 156-158.

7 Алексанина, В.О. Автоматизированные системы бронирования и резервирования / В.О. Алексанина, Л.В. Ребышева // Вузовская наука: теоретико-методологические проблемы подготовки специалистов в области экономики, менеджмента и права: – Тюмень: Вектор Бук, 2015. – с. 159-162.

8 Сарсебаева, А.М. Система бронирования и резервирования в гостиницах и ресторанах / А.М. Сарсебаева // Вестник Алматинского технологического университета. № 3 – Алматы: АТУ, 2019. – с. 109-114.

9 Куклинский, М.В. Сетевое приложение к системе предварительного бронирования авиабилетов / М.В. Куклинский, Г.В. Головня,

М.О. Лукаш // Інженерія програмного забезпечення. т. 2. № 22. Киев: НАУ, 2015. – с. 40-48.

10 Шаймерденов, Т.С. Инновации в системе бронирования для развития туризма в РК на примере системы бронирования «AMADEUS» / Т.С. Шаймерденов // Материалы международной научно-практической конференции обучающихся «Актуальные проблемы экономики и управления: теория и практика» (29 марта 2017 г.). – Гжель: ГГУ, 2017. – с. 151-153.

11 Сервис поиск и бронирования экскурсий Спутник [Электронный ресурс] URL: https://www.sputnik8.com/. (дата обращения: 19.04.2020).

12 Сервис бронирования экскурсий Weatlas [Электронный ресурс] URL: https://weatlas.com/. (дата обращения: 19.04.2020).

13 Сервис бронирования экскурсий Tripster [Электронный ресурс] URL: https://experience.tripster.ru/. (дата обращения: 19.04.2020).

14 Сервис бронирования экскурсий Excursio [Электронный ресурс] URL: https://excursio.com/ru/. (дата обращения: 19.04.2020).

15 Система онлайн записи для бронирования событий, проката и аренды [Электронный ресурс] URL: https://bukza.ru/. (дата обращения: 19.04.2020).

16 Система онлайн-записи и приложение бронирования SimplyBook.me [Электронный ресурс] URL: https://simplybook.me/ru/. (дата обращения: 19.04.2020).

17 Модули онлайн записи для бизнеса Bookio.me [Электронный ресурс] URL: https://bookio.me /. (дата обращения: 19.04.2020).

18 Модуль-сервис онлайн бронирования на сайт со встроенной CRM [Электронный ресурс] URL: https://m-bron.ru/. (дата обращения: 19.04.2020).

19 Виджет бронирования для сайта – Booking Widget [Электронный ресурс] URL: https://booking-widget.ru/. (дата обращения: 19.04.2020).

20 Система автоматизации экскурсионного туризма – Atom-S [Электронный ресурс] URL: https://start.atom-s.com/ru/. (дата обращения: 19.04.2020).

21 Booking Core – Ultimate Booking System [Электронный ресурс] URL: https://www.bookingcore.org/intro. (дата обращения: 19.04.2020).

22 Ezbus – программное обеспечение для туризма [Электронный ресурс] URL: https://ezbus.ru/products/online/. (дата обращения: 19.04.2020).

23 Ag Booking CMS [Электронный ресурс] URL: https:// onriv.com/ (дата обращения: 19.04.2020).

24 Емельянова, Н.З. Основы построения автоматизированных систем: учебное пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. – М.: ФОРУМ: ИНФРА , 2005 – 416 с.

25 Буч, Г. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. / Г. Буч, Д. Рамбо, И. Якобсон. – М.: ДМК Пресс, 2006. – 496 с.

26 Розенберг, Д. Применение объектного моделирования с использованием UML и анализпрецедентов: пер. с англ. / Д. Розенберг, К. Скотт – М.: ДМК Пресс. – 160 с

27 Спецификация HTML5. [Электронный ресурс] URL: https://www.w3.org/TR/html/ (дата обращения: 19.04.2020).

28 PHP. Руководство по PHP. [Электронный ресурс] URL: http://php.net/manual/ru/ . (дата обращения: 19.04.2020).

29 Колисниченко, Д. Движок для вашего сайта. CMS Joomla!, Slaed, PHP-Nuke. / Д. Колисниченко – Спб.: БХВ, 2008. – 352 с.

30 Apache HTTP Server Documentation [Электронный ресурс] URL: http://httpd.apache.org/docs/ (дата обращения: 15.04.2020).

31 ASP.NET – Википедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ASP.NET. (дата обращения: 15.04.2020).

32 HTML – Википедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/HTML. (дата обращения: 15.04.2020).

33 Perl – Википедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Perl. (дата обращения: 15.04.2020).

34 PHP. Руководство по PHP. [Электронный ресурс] URL: http://php.net/manual/ru/. (дата обращения: 15.04.2020).

35 Python – Википедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ Python. (дата обращения: 15.04.2020).

36 Ruby – Википедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ Ruby. (дата обращения: 15.04.2020).

37 ГОСТ 34.602.89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [Текст]. – Введ. 1990-01-01. – М.: Изд-во стандартов, 1990. – 17 с.

38 Кузнецов, М. В. MySQL 5 / М. В. Кузнецов, И. В. Симдянов – Спб.: БХВ-Петербург, 2010. – 1007 с.

39 Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов. / Д. Дакетт – М.: Эксмо, 2013. – 480 с.

40 Компания Яндекс – Принципы – Яндекс и поисковая оптимизация. [Электронный ресурс] URL: https://yandex.ru/company/rules/optimization. (дата обращения: 15.04.2020).

41 Оптимизация сайтов для поисковых систем (SEO) – Справка – Search Console [Электронный ресурс] URL: https://support.google.com/webmasters/answer/35291?hl=ru. (дата обращения: 15.04.2020).

42 Twig – The flexible, fast, and secure template engine for PHP [Электронный ресурс] URL: https://twig.sensiolabs.org/ (дата обращения: 15.04.2020).

43 Pug – robust, elegant, feature rich template engine for Node.js URL: https://github.com/pugjs/pug (дата обращения: 15.04.2020).

44 Symfony 4.1 Documentation – Symfony [Электронный ресурс] URL: https://symfony.com/doc/current/index.html (дата обращения: 15.04. 2020).

45 Vue.js Прогрессивный JavaScript-фреймворк – Vue.js [Электронный ресурс] URL: https://ru.vuejs.org/ (дата обращения: 15.04. 2020).

46 Eloquent: Getting Started – Laravel – The PHP Framework For Web Artisans [Электронный ресурс] URL: https://laravel.com/docs/5.6/eloquent. (дата обращения: 15.04.2020).

47 TinyMCE [Электронный ресурс] URL: http://www.tinymce.com/ (дата обращения: 15.04.2020).

48 elFinder file manager for web [Электронный ресурс] URL: http:// elfinder.org (дата обращения: 15.04.2020).

49 Зандстра, М. PHP. Объекты, шаблоны и методики программирования / М. Зандстра – М.: Вильямс, 2009. – 560 с.

50 Котеров, Д. В., Костарев А. Ф. PHP 5 / Д. В. Котеров, А. Ф. Костарев – Спб.: БХВ-Петербург, 2014. – 1104с.

51 Флэнаган, Д. JavaScript. Карманный справочник / Д. Флэнаган – М.: Вильямс, 2015. – 320 с.

52 Ларман, К. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание.: пер. с англ. А.Ю. Шелестова / К. Ларман – М.: ООО «ИД. Вильямс». 2013. – 736 с.

53 Фаулер, М. UML. Основы Краткое руководство по стандартному языку объектного моделирования 3-е изд.: – пер. с англ. А. Петухов / М. Фаулер – СПб. : Символ-Плюс, 2018. – 192 с.

54 PhpMetrics: static analyzys for PHP [Электронный ресурс] URL: https://www.phpmetrics.org (дата обращения: 19.04.2020).

Приложение А

Распределение трудовых ресурсов

(рекомендуемое)

Таблица А.1 – Распределение трудовых ресурсов

Название задачи	Трудозатраты
Анализ предметной области	72 ч
Описание предметной области	24 ч
Руководитель проекта	24 ч
Обзор существующих решений	24 ч
Руководитель проекта	24 ч
Анализ вариантов использования	8 ч
Руководитель проекта	8 ч
Анализ требований к системе	8 ч
Постановка задачи и составление технического задания	8 ч
Руководитель проекта	8 ч
Планирование и расчет бюджета	56 ч
Календарное планирование	24 ч
Руководитель проекта	24 ч
Рассчет бюджета	24 ч
Руководитель проекта	24 ч
Выбор программно-аппаратных средства	8 ч
Руководитель проекта	2,72 ч
Backend программист	2,64 ч
Frontend программист	2,64 ч
Проектирование	66 ч
Проектирование интерфейса	28 ч
Frontend программист	8 ч
Дизайнер	20 ч
Организация процесса обработки информации	22 ч
Backend программист	20 ч
Frontend программист	2 ч
Определение структур хранения данных	16 ч
Backend программист	16 ч
Реализация веб-приложения	136 ч
Кодирование серверной части	56 ч
Руководитель проекта	14 ч
Backend программист	42 ч
Верстка интерфейса	40 ч
Руководитель проекта	10 ч

Продолжение таблицы А.1

Frontend программист	30 ч
Кодирование клиентской части	40 ч
Руководитель проекта	10 ч
Frontend программист	30 ч
Разаработка веб-приложения завершена	0 ч
Тестирование разработанного приложения	32 ч
Выявления ошибок	8 ч
Тестировщик	8 ч
Выявление причин ошибок	8 ч
Руководитель проекта	2,72 ч
Backend программист	2,64 ч
Frontend программист	2,64 ч
Устранение ошибок	16 ч
Руководитель проекта	5,44 ч
Backend программист	5,28 ч
Frontend программист	5,28 ч
Оформление ПЗ	56 ч
Составление отчета	56 ч
Руководитель проекта	56 ч
Бумага	1 шт
Итого	418 ч

Приложение Б

Диаграмма Ганта

(рекомендуемое)

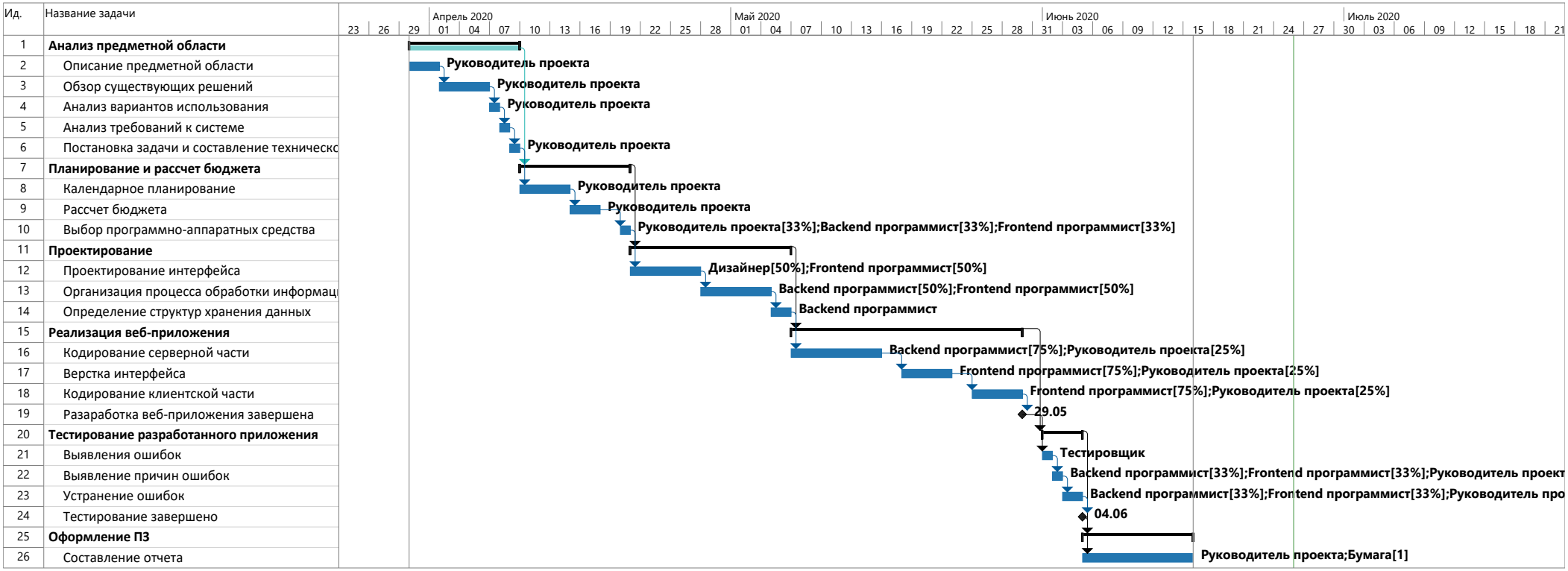

Рисунок Б – Диаграмма Ганта

Приложение В

Листинг кода конфигурационного файла htaccess

(рекомендуемое)

```
Options -Indexes
AddDefaultCharset UTF-8
php_flag error_reporting -1
php_flag display_startup_errors on
php_flag display_errors on
php_flag html_errors on
php_value error_log  ./log/PHP_errors.log

<IfModule mod_rewrite.c>
RewriteEngine on
RewriteBase /

#RewriteCond %{REQUEST_URI} ^\/public\/(.*)\.(.+)$ [OR]
RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^(.*)$ - [NC,L]
RewriteRule ^(.*)$ index.php [NC,L,QSA]
</IfModule>

DirectoryIndex index.php

<FilesMatch "\.(pdf|flv|gif|jpg|jpeg|png|svg|ico|swf|js|css)$">
  Header set Cache-Control "max-age=2592000, public"
    Header unset ETag
    FileETag None
</FilesMatch>

<IfModule mod_expires.c>
    ExpiresActive Off
    ExpiresDefault "access plus 2 day"
</IfModule>

<ifModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
application/xml application/xhtml+xml text/css text/javascript
application/javascript application/x-javascript
</ifModule>
```

Приложение Г

Листинг кода файла index.php

(рекомендуемое)

```
<?
$st = microtime(true);
date_default_timezone_set('Europe/Moscow');

require 'configs/bootstrap.php';

$app = new App\Application((bool) $_SERVER['APP_DEBUG']);
$app->run();
```

Приложение Д

Листинг кода класса Application

(рекомендуемое)

```
<?
namespace App;
use App\Exception\HttpException;
use App\Module\Breadcrumbs;
use App\Module\Container;
use App\Module\ErrorHandler;
use App\Module\Router;
use App\Module\User;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Session\Session;
class Application extends Container
{
    public $response = null;
    public $context = array();
    protected $view = array();

    public function __construct()
    {
        $this['context'] = array(
            'requestUrl' => $_SERVER['REQUEST_URI'],
            'path' => parse_url($_SERVER["REQUEST_URI"],
PHP_URL_PATH),
            'host' => ($_SERVER['SERVER_PORT'] == 443 ?
'https://' : 'http://') . $_SERVER['HTTP_HOST'],
        );
        $this->register('error', new
\App\Module\ErrorHandler($this));
        $this->register('db', new \App\Module\Eloquent($this));
        $this->register('session', new Session());
        $this->register('view', (new \App\Module\Twig($this))-
>get());
        $this->register('user', new User($this));
        $this->register('router', new Router($this));
        $this->register('breadcrumbs', new Breadcrumbs($this));
        $this['router']->addFromArray('./configs/list-
routes.php');
        $this['breadcrumbs']->add(array(
            'name' => 'Главная',
            'link' => '/',
        ));
        $this['router']->match();
        $this->handle();
    }
    public function handle()
    {
        try {
            $this['router']->callAction();
            $this->response = $this['router']->getResponse();
        } catch (HttpException $e) {
            $this->response = $this->error($e, $e-
>getStatusCode());
        } catch (\Throwable $e) {
            $this->response = $this['error']-
>handleException($e);
```

```
        }
    }
    public function run()
    {
        if ($this->response instanceof Response) {
            $this->response->send();
        }
    }
    public function register($name, &$service)
    {
        $this[$name] = $service;
    }
    public function redirect($url, $code)
    {
        return new RedirectResponse($url, $code);
    }
    public function abort($statusCode, $message = '', array
$headers = [])
    {
        throw new HttpException($statusCode, $message, null,
$headers);
    }
    public function error(\Exception $e, $code)
    {
        switch ($code) {
            case 404:
                $layout = '/view/error/404.html';
                break;
            case 500:
                $layout = '/view/error/500.html';
                break;
            default:
                $layout = '/view/error/404.html';
        }
        if ($_REQUEST['format'] == 'json') {
            return new JsonResponse(array(
                'type' => 'error',
                'path' => $_SERVER['REQUEST_URI'],
                'status' => $code,
                'msg' => $e->getMessage(),
                'code' => $e->getCode(),
                'file' => $e->getFile(),
                'line' => $e->getLine(),
            ), $code);
        } else {
            return new Response($this['view']->render($layout,
$this['context']), $code);
        }
    }
}
```

Приложение Е

Листинг кода класса маршрутизатора Router

(рекомендуемое)

```
<?
namespace App\Module;

use App\Application;
use App\Exception\NotFoundHttpException;
use App\Exception\RouteNotFoundException;
class Router
{
    private $handler = null;
    private $map = null;
    private $patterns = [
        '{all}',
        '{char}',
        '{number}',
    ];
    private $replacements = [
        '(.+)',
        '([a-zA-Z]+)',
        '([0-9]+)',
    ];
    protected $response = array();
    public function __construct(Application $app)
    {
        $this->app = $app;
    }
    public function add($method, $string, $action, $vars = null)
    {
        $method = strtoupper($method);
        $string = strtolower($string);
        if ($method == '*') {
            $this->add('GET', $string, $action, $vars);
            $this->add('POST', $string, $action, $vars);
        } else {
            if (!strstr($string, '{')) {
                $this->map[$method]['static'][$string] = [
                    'action' => $action,
                ];
                return;
            }
            $replaced = str_replace($this->patterns, $this-
>replacements, $string);
            $replaced = '#^' . $replaced . '$#';
            $this->map[$method]['regexp'][$replaced] = [
                'action' => $action,
            ];
            if ($vars) {
                $vars = (array) $vars;
                $this->map[$method]['regexp'][$replaced]['vars']
= $vars;
            }
        }
    }

    public function match($url = false)
    {
```

```
        $url = $url ? $url : explode('?',
urldecode($_SERVER['REQUEST_URI']))[0];
        $method = $_SERVER['REQUEST_METHOD'];
        $this->handler = null;
        // in static
        if (isset($this->map[$method]['static'][$url])) {
            //return $this->call($this-
>map[$method]['static'][$url]);
            return $this->handler = $this-
>map[$method]['static'][$url];
        }
        // in regex
        foreach ($this->map[$method]['regexp'] as $pattern =>
$route) {
            if (!preg_match($pattern, $url, $match)) {
                continue;
            } else {
                if ($match) {
                    $keys = array_slice($match, 1);
                    $values = $route['vars'];
                    $route['vars'] = array_combine($values,
$keys);
                } else {
                    $route['vars'] = array();
                }
                return $this->handler = $route;
            }
        }
        return $this->handler;
    }
    public function callAction($route)
    {
        $route = $route ? $route : $this->handler;
        if ($route) {
            $this->call($route);
        } else {
            throw new NotFoundHttpException(404, "Not Found");
        }
    }
    public function addFromArray($path)
    {
        $routes = include_once $path;
        if (is_array($routes)) {
            foreach ($routes as $i => $route) {
                $this->add($route[0], $route[1], $route[2],
$route[3]);
            }
        }
    }
    public function call($route)
    {
        $controller = '\\App\Controller\\' .
ucfirst($route['action']);
        list($class, $method) = explode('::', $controller, 2);
        if (class_exists($class)) {
            if (method_exists($class, $method)) {
```

```
                $controller = array(new $class(), $method);
                $arguments = array($this->app, $route['vars']);
                $this->response =
\call_user_func_array($controller, $arguments);
            } else {
                throw new RouteNotFoundException('Route
Controller method not exist');
            }
        } else {
            throw new RouteNotFoundException('Route Controller
class not exist');
        }
    }
    public function getResponse()
    {
        return $this->response;
    }
}
```

Приложение Ж

Листинг кода маршрутов веб-приложения

(рекомендуемое)

```
<?
return array(
    array(
        'GET', '/', 'home::index',
    ),
    array(
        'GET', '/page{number}', 'Page::byid', 'id',
    ),
    array(
        'GET', '/user{number}', 'user::byid', 'id',
    ),
    array(
        'GET', '/user', 'user::profile',
    ),
    array(
        '*', '/login', 'user::login',
    ),
    array(
        '*', '/logout', 'user::logout',
    ),
    array(
        'GET', '/profile', 'user::profile',
    ),
    array(
        'GET', '/news/', 'news::index',
    ),
    array(
        'GET', '/news/{number}', 'news::newsById', 'id',
    ),
    array(
        'GET', '/Test', 'Test::index',
    ),
    array(
        '*', '/api/reserve', 'API\Reserve::connect',
    ),
    array(
        '*', '/api/events', 'API\Events::connect',
    ),
    array(
        '*', '/api/news', 'API\News::connect',
    ),
    array(
        '*', '/api/object', 'API\ObjectModel::connect',
    ),
    array(
        '*', '/api/auth/user/', 'API\Auth::user',
    ),
    array(
        '*', '/api/auth/login/', 'API\Auth::login',
    ),
    array(
        '*', '/api/auth/logout/', 'API\Auth::logout',
    ),
    array(
        'GET', '/events/', 'Event::index',
    ),
```

```
    array(
        'GET', '/events/{number}', 'Event::event', ['id'],
    ),
    array(
        'GET', '/panel/', 'Panel\Panel::index',
    ),
    array(
        '*', '/test/not-exist-object', 'API\ObjectModel::index',
    ),
    array(
        '*', '/test/not-exist-method', 'API\ObjectModel::error',
    ),
);
```

Приложение И

Листинг кода класса Booking

(рекомендуемое)

```
<?php

namespace App\Module;

use App\Application;
use App\Exception\Reserve\ReserveException;

class Booking
{
    protected $user;
    protected $token;
    protected $session;
    protected $cookies;
    protected $request;
    protected $response;

    public function __construct(Application $app)
    {
        $this->app = $app;
        $time = microtime(true);

        return $this->user;
    }

    public function get($rid)
    {
        $user = $this->app['user']->getUser();
        if ($user) {
            if ($rid) {
                $reserve = \App\Model\Reserve::find($rid);
                if ($reserve) {
                    return $reserve;
                } else {
                    throw new ReserveException("Reserve not
exist", "reserve/not-exist");
                }
            } else {
                throw new ReserveException("No arguments
passed", "reserve/not-arguments");
            }

        } else {
            throw new ReserveException("User not authorized",
"reserve/user-not-authorized");
        }
    }

    public function all($user_id = null, $events = false)
    {
        $mode = 'user';
        $user = $this->app['user']->getUser();
        if ($user_id && $user_id != $user['id']) {
            $mode = 'admin';
            if ($user['access'] >= 7) {
                $user = \App\Model\User::find($user_id);
                if (!$user) {
```

```
                    throw new ReserveException("User not exist",
'reserve/user-not-exist');
                }
            } else {
                throw new ReserveException("Permission denied",
'reserve/permission-denied');
            }

        }

        if ($user) {
            $user_id = $user['id'];
            $reserve = $events ?
\App\Model\Reserve::where('user_id', $user_id)->with('event')-
>get() : \App\Model\Reserve::where('user_id', $user_id)->get();
            if ($reserve) {
                return $reserve;
            } else {
                throw new ReserveException("Reserve not exist",
"reserve/not-exist");
            }

        } else {
            throw new ReserveException("User not authorized",
"reserve/user-not-authorized");
        }
    }

    public function create(int $event_id, $qty = 1, $options
= [], $user_id = null)
    {
        $mode = 'user';
        $user = $this->app['user']->getUser();
        if ($user_id && $user_id != $user['id']) {
            $mode = 'admin';
            if ($user['access'] >= 7) {
                $user = \App\Model\User::find($user_id);
                if (!$user) {
                    throw new ReserveException("User not exist",
'reserve/user-not-exist');
                }
            } else {
                throw new ReserveException("Permission denied",
'reserve/permission-denied');
            }
        }

        if ($user) {
            $user_id = $user['id'];
            if ($event_id) {
                $event = \App\Model\Events::find($event_id);
                if ($event) {
                    $reserved = \App\Model\Reserve::
                        where('event_id', $event_id)
                        ->count();
```

```
                    if ($event->places && $reserved < $event-
>places) {
                        $data = array(
                            'event_id' => (int) $event_id,
                            'user_id' => (int) $user_id,
                            'options' => $options,
                        );
                        $reserve = new
\App\Model\Reserve($data);
                        $r = $reserve->save();
                        if ($r) {
                            return [] + $data + ['mode' =>
$mode];
                        } else {
                            throw new ReserveException("Unknown
Error");
                        }
                    } else {
                        throw new ReserveException("Places are
booked", 'reserve/places-are-occupied');
                    }

                } else {
                    throw new ReserveException("Event not
exist", 'reserve/event-not-exist');
                }
            } else {
                throw new ReserveException("No arguments
passed");
            }
        } else {
            throw new ReserveException("User not authorized",
'reserve/user-not-authorized');
        }
        return $response;
    }

    public function update(int $reserve_id, $qty = null,
$options = [], $user_id = null)
    {
        $user = $this->app['user']->getUser();
        if ($user) {
            if ($rid) {
                $reserve = \App\Model\Reserve::find($rid);
                if ($reserve) {
                    return $reserve;
                } else {
                    throw new ReserveException("Reserve not
exist", "reserve/not-exist");
                }
            } else {
                throw new ReserveException("No arguments
passed", "reserve/not-arguments");
            }

        } else {
```

```
            throw new ReserveException("User not authorized",
"reserve/user-not-authorized");
        }
    }

    public function remove($reserve_id)
    {
        $mode = 'user';
        $user = $this->app['user']->getUser();
        if ($user_id && $user_id != $user['id']) {
            $mode = 'admin';
            if ($user['access'] >= 7) {
                $user = \App\Model\User::find($user_id);
                if (!$user) {
                    throw new ReserveException("User not exist",
'reserve/user-not-exist');
                }
            } else {
                throw new ReserveException("Permission denied",
'reserve/permission-denied');
            }

        }

        if ($reserve_id && $user) {
            if (is_array($reserve_id)) {
                if ($user['access'] >= 7) {
                    $reserves = \App\Model\Reserve::
                        whereIn('id', $reserve_id)
                        ->get(['id']);
                } else {
                    $reserves = \App\Model\Reserve::
                        whereIn('id', $reserve_id)
                        ->where('user_id', $user['id'])
                        ->get(['id']);
                }

                if ($reserves) {
                    $deleted_ids = $reserves->pluck('id');
                    $deleted = \App\Model\Reserve::
                        whereIn('id', $deleted_ids)
                        ->delete();

                    if ($deleted) {
                        return $reserve_id;
                    } else {
                        throw new ReserveException("Unknown
Error");
                    }
                } else {
                    throw new ReserveException("Reserve not
exist", 'reserve/reserves-not-exist');
                }
            } else {
                if ($user['access'] >= 7) {
                    $reserve = \App\Model\Reserve::
```

```
                        where('id', $reserve_id)
                        ->first(['id']);
                } else {
                    $reserve = \App\Model\Reserve::
                        where('id', (int) $post['rid'])
                        ->where('user_id', $user['id'])
                        ->first(['id']);
                }
                if ($reserve) {
                    $deleted = \App\Model\Reserve::
                        where('id', $reserve->id)
                        ->delete();

                    if ($deleted) {
                        return $reserve_id;
                    } else {
                        throw new ReserveException("Unknown
Error");
                    }
                } else {
                    throw new ReserveException("Reserve not
exist");
                }
            }
        } else {
            throw new ReserveException("No arguments passed");
        }
    }
}
```

Приложение К

Листинг кода класса User

(рекомендуемое)

```
<?php

namespace App\Module;

use App\Exception\Auth\BadCredentialsException;
use App\Exception\Auth\UsernameNotFoundException;
use Symfony\Component\HttpFoundation\Cookie;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

class User
{
    protected $user;
    protected $token;
    protected $session;
    protected $cookies;
    protected $request;
    protected $response;

    public function __construct($app)
    {
        $this->app = $app;
        $time = microtime(true);
        $this->request = Request::createFromGlobals();
        $this->response = new Response();
        $this->cookies = $this->request->cookies;
        $this->session = $this->app['session'];
        $this->token = $this->session->get('key') ? $this-
>session->get('key') : $this->cookies->get('key');

        if ($this->session->get('user')) {
            if ($this->session->get('key') == $this->token) {
                $user = $this->session->get('user');
                $user['token'] = $this->token;
            } else {
                $this->session->remove('user');
            }
        } else {
            if ($this->token) {
                $token = \App\Model\Token::where('token', $this-
>token)->first();
                if ($token) {
                    $user = $token->user->first()->toArray();
                    $user['token'] = $this->token;
                    $this->session->set('key', $this->token);
                    $this->online();
                    $this->session->set('user', $user);
                }
            }
        }
        $this->user = isset($user) ? $user : null;
        $app['context'] = array_merge($app['context'], array(
            'user' => $this->getUser(),
        ));
        return $this->user;
    }
```

```
    public function getUser()
    {
        return $this->user;
    }

    public function getUID()
    {
        return $this->user ? $this->user['id'] : null;
    }

    public function getToken()
    {
        return $this->token;
    }

    public function isAuth()
    {
        if ($this->user) {
            return true;
        } else {
            return false;
        }
    }

    public function login($login, $password, $remember = false)
    {
        $ans = false;
        $this->token = bin2hex(openssl_random_pseudo_bytes(16));
        $user = \App\Model\User::where('login', $login)-
>first();

        if ($user != false && !empty($user)) {
            if ($user['password'] == $password) {
                $token = \App\Model\Token::create(array(
                    'token' => $this->token,
                    'user_id' => $user->id,
                    'ip' => $_SERVER['REMOTE_ADDR'],
                    'user_agent' => $_SERVER['HTTP_USER_AGENT'],
                    'last_visit' => time(),
                ));
                $this->user = $user->toArray();
                $this->session->set('user', $this->user);
                $this->session->set('key', $this->token);
                if ($remember) {
                    $this->response->headers->setCookie(new
Cookie('key', $this->token, strtotime('now + 2 year')));
                } else {
                    $this->response->headers->setCookie(new
Cookie('key', $this->token, 0));
                }
                $this->response->sendHeaders();
                $user = $this->user;
            } else {
                throw new BadCredentialsException();
            }
```

```
        } else {
            throw new UsernameNotFoundException($login);
        }
        return $user;
    }

    public function logout($key = false)
    {
        if (!$key) {
            $key = $this->token;
        }
        if ($key) {
            $token = \App\Model\Token::where('token', $key)-
>first();
            $this->session->remove('key');
            $this->response->sendHeaders();
            if ($token) {
                $deleted = $token->delete();
            }
        }
        return $deleted;
    }

    public function join($data = array())
    {
        if ($data) {
            $isset = \App\Model\User::where('login',
$data['login'])->first();

            if (!$isset) {
            $user = new \App\Model\User($data);
            return $user->save();
            } else {
throw new UsernameBusyException($login);
$user = false;
            }
            return !$isset;
        } else {
            return false;
        }
    }

    public function online($force = false)
    {
        $ans = false;
        if ($this->token) {
            $token = \App\Model\Token::where('token', $this-
>token)->first();
            if ($token) {
                if ($force || $token->last_visit + 60 * 15 <=
time()) {
                    $token->update(array('last_visit' =>
time()));
                    $ans = true;
                } else {
                    $ans = true;
```

```
                    }
                }
            }
            return $ans;
        }

    }
```

Приложение Л

Листинг кода класса ObjectModel

(рекомендуемое)

```
<?
namespace App\Controller\API;

use App\Application;
use App\Controller\Panel\Auth;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Response;

class ObjectModel
{
    use Auth;

    protected function connect(Application $app)
    {
        global $st;
        $model_name = $_REQUEST['object'];
        $action_name = $_REQUEST['action'];

        $r = array(
            'request' => array(
                'object' => $model_name,
                'action' => $action_name,
            ),
        );
        $r['request'] += $_REQUEST;

        if ($action_name == 'read') {
            $r += self::read($app);
        } elseif ($action_name == 'all') {
            $r += self::all($app);
        } elseif ($action_name == 'create') {
            $r += self::create($app);
        } elseif ($action_name == 'update') {
            $r += self::update($app);
        } elseif ($action_name == 'updateItems') {
            $r += self::updateItems($app);
        } elseif ($action_name == 'delete') {
            $r += self::delete($app);
        } elseif ($action_name == 'ids') {
            $r += self::ids($app);
        } else {
            $r += self::schema($app);
        }

        if ($_REQUEST['format'] == 'json') {
            $response = new JsonResponse(
                $r
            );
        } else {
            $response = new Response(
                json_encode($r, JSON_PRETTY_PRINT)
            );
        }
        $response->headers->set('X-Data-render_time',
(microtime(true) - $st));
        return $response;
```

```
    }

    protected static function read(Application $app)
    {
        $model_name = $_REQUEST['object'];
        if (file_exists(STORAGE . 'schemes/' .
ucfirst($model_name) . '.json')) {
            $schema = json_decode(file_get_contents(STORAGE .
'schemes/' . ucfirst($model_name) . '.json'));
        } else {
            $schema = '{}';
        }

        $post = array(
            'pk' => (int) $_REQUEST['pk'],
        );
        $model = '\\App\Model\\' . ucfirst($model_name);
        if (class_exists($model)) {
            $object = $model::
                where('id', $post['pk'])
                ->first();
            if ($object) {
                $response = array(
                    'response' => array(
                        'state' => 'success',
                        'values' => $object->toArray(),
                        'schema' => $schema,
                        'post' => $post,
                    ),
                );
            } else {
                $response = array(
                    "response" => array(
                        "state" => 'error',
                        'error' => 'object/record-not-exist',
                        'msg' => 'Запись с таким id не
существует',
                    ),
                );
            }
        } else {
            $response = array(
                "response" => array(
                    "state" => 'error',
                    'error' => 'object/model-not-exist',
                    'msg' => 'Модели с таким классом не
существует',
                ),
            );
        }
        return $response;
    }

    protected static function create(Application $app)
    {
        global $st;
```

```
        $model_name = $_REQUEST['object'];
        $values = $_REQUEST['data'];
        $values['created_by'] = $app['user']->getUID();
        $values['created_at'] =
strtotime($_REQUEST['values']['created_at']) ?
$_REQUEST['values']['created_at'] : new \DateTime('now');

        $model = '\\App\Model\\' . ucfirst($model_name);
        $exist = $model::
            where('id', $values['id'])
            ->first();

        if (!$exist) {
            $model = '\\App\Model\\' . ucfirst($model_name);
            $object = new $model($values);
            $result = $object->save();
            if ($result) {
                $response = array(
                    'response' => array(
                        'state' => 'success',
                        'msg' => 'Успешно добавлено',
                        'values' => $object->toArray(),
                    ),
                );
            } else {
                $response = array(
                    "response" => array(
                        "state" => 'error',
                        'error_code' => 11,
                        'msg' => 'При добавлении произошла
ошибка',
                    ),
                );
            }
        } else {
            $response = array(
                "response" => array(
                    "state" => 'error',
                    'error_code' => 12,
                    'msg' => 'Запись с таким id уже существует',
                ),
            );
        }

        return $response;
    }

    protected static function update(Application $app)
    {
        $model_name = $_REQUEST['object'];
        $pk = (int) $_REQUEST['pk'];

        $values = $_REQUEST['data'];
        $values['updated_by'] =
$_REQUEST['values']['updated_by'] ?
$_REQUEST['values']['updated_by'] : $app['user']->getUID();
```

```
        $values['updated_at'] =
strtotime($_REQUEST['values']['updated_at']) ?
$_REQUEST['values']['updated_at'] : new \DateTime('now');

        $model = '\\App\Model\\' . ucfirst($model_name);
        $object = $model::
            where('id', '=', $pk)
            ->first();

        if ($object) {

            $result = $object->update($values);

            if ($result) {
                $response = array(
                    'response' => array(
                        'state' => 'success',
                        'msg' => 'Запись успешно обновлена',
                        'values' => $object->toArray(),
                    ),
                );
            } else {
                $response = array(
                    "response" => array(
                        "state" => 'error',
                        'error_code' => 21,
                        'msg' => 'При обновлении данных
произошла ошибка',
                    ),
                );
            }
        } else {
            $response = array(
                "response" => array(
                    "state" => 'error',
                    'error_code' => 22,
                    'msg' => 'Запись с таким id не существует',
                ),
            );
        }

        return $response;
    }

    protected static function updateItems(Application $app)
    {

        $items = $_REQUEST['items'];
        if ($items) {
            $objects = false;
            $items_id = [];
            $model_name = $_REQUEST['object'];

            foreach ($items as $key => $object) {
                array_push($items_id, $object['pk']);
            }
```

```
            $model = '\\App\Model\\' . ucfirst($model_name);
            $objects = $model::
                //where('id','=', $pk)
                whereIn('id', $items_id)
                ->get();

            if ($objects) {
                $result = true;

                $objects->map(function ($object) use ($items,
$result) {
                    if ($items[$object->id]['values']) {

                        try {
                            $object->update($items[$object-
>id]['values']);
                            $result = true;

                        } catch (Exception $e) {
                            $result = false;
                        }

                        $result = $object->save();
                    }
                });

                if ($result) {
                    $response = array(
                        'response' => array(
                            'state' => 'success',
                            'msg' => 'Запись успешно обновлена',
                            'items' => $objects->toArray(),
                        ),
                    );
                } else {
                    $response = array(
                        "response" => array(
                            "state" => 'error',
                            'error_code' => 21,
                            'msg' => 'При обновлении данных
произошла ошибка',
                        ),
                    );
                }
            } else {
                $response = array(
                    "response" => array(
                        "state" => 'error',
                        'error_code' => 22,
                        'msg' => 'Запись с таким id не
существует',
                    ),
                );
            }
```

```
        } else {
            $response = array(
                "response" => array(
                    "state" => 'error',
                    'error_code' => 24,
                    'msg' => 'Не переданы объекты для
обновления',
                ),
            );
        }

        return $response;
    }

    protected static function schema(Application $app)
    {
        $model_name = $_REQUEST['object'];
        if (file_exists(STORAGE . 'schemes/' .
ucfirst($model_name) . '.json')) {
            $schema = json_decode(file_get_contents(STORAGE .
'schemes/' . ucfirst($model_name) . '.json'));
        } else {
            $schema = '{}';
        }

        $response = array(
            "response" => array(
                "state" => 'success',
                "schema" => $schema,
            ),
        );
        return $response;
    }

    protected static function all(Application $app)
    {
        $model_name = 'OrderStatus';

        $order_id = (int) $_REQUEST['data']['order_id'];

        if ($order_id) {
            $model = '\\App\Model\\' . ucfirst($model_name);
            $values = $model::
                where('order_id', $order_id)
                ->get();

            if ($values) {

                $response = array(
                    "response" => array(
                        "state" => 'success',
                        "values" => $values->toArray(),
                    ),
                );
            } else {
                $response = array(
```

```
                    "response" => array(
                        "state" => 'error',
                        'error_code' => 54,
                        'msg' => 'Нет данных',
                    ),
                );
            }
        } else {
            $response = array(
                "response" => array(
                    "state" => 'error',
                    'error_code' => 51,
                    'msg' => 'Не передан id заказа',
                ),
            );
        }
        return $response;
    }

    protected static function ids(Application $app)
    {
        $model_name = $_REQUEST['object'];
        $key_title = $_REQUEST['title'] ? $_REQUEST['title'] :
'title';

        if ($model_name) {
            $model = '\\App\Model\\' . ucfirst($model_name);

            $object = $model::
                get(['id', $key_title])
                ->keyBy('id');

            if ($object) {
                $response = array(
                    'response' => array(
                        'state' => 'success',
                        'values' => $object->toArray(),
                    ),
                );
            } else {
                $response = array(
                    "response" => array(
                        "state" => 'error',
                        'error_code' => 1,
                        'msg' => 'Объекта ' . $model_name . 'не
существует',
                    ),
                );
            }
        } else {
            $response = array(
                "response" => array(
                    "state" => 'error',
                    'error_code' => 2,
                    'msg' => 'Не передана модель объекта',
                ),
```

```
                );
            }
            return $response;
        }
    }
```

Приложение М

Листинг кода классов моделей Eloquent

(рекомендуемое)

```
<?
namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Area extends Model
{

    protected $table = 'Area';
    protected $with = [
    ];
    protected $casts = [
        'places' => 'json',
    ];

    protected $fillable = ['title', 'places'];
    protected $dateFormat = "Y-m-d H:i:s";
    public $timestamps = true;

    public function seats()
    {
        return $this->hasMany('App\Model\Seat', 'area_id',
'id');
    }

    public function event()
    {
        return $this->belongsToMany('App\Model\Event', 'id',
'area_id');
    }

}

class Events extends Model
{

    protected $table = 'Events';

    protected $fillable = ['area_id', 'image', 'title', 'desc',
'started_at', 'finished_at'];
    protected $dateFormat = "Y-m-d H:i:s";
    public $timestamps = true;

    public function area()
    {
        return $this->belongsTo('App\Model\Area', 'area_id',
'id');
    }

    public function reserve()
    {
        return $this->belongsTo('App\Model\Reserve', 'event_id',
'id');
    }

}
```

```
class Nav extends Model
{
    protected $table = 'Nav';
    protected $casts = [
        'params' => 'json',
    ];

    public $timestamps = true;
    protected $fillable = ['name', 'title', 'description',
'access', 'params'];

    public function getItemsAttribute()
    {
        return $this->getRelationValue('navItems')->keyBy('id');
    }

    public function navItems()
    {
        return $this->hasMany('App\Model\navItems', 'menu_id',
'id')->orderBy('ordering', 'asc');
    }

    public function items()
    {
        return $this->hasMany('App\Model\NavItems', 'nav_id',
'id')->orderBy('id', 'asc')->orderBy('ordering', 'asc');
    }

}

class NavItems extends Model
{
    protected $table = 'NavItems';
    protected $casts = [
        'params' => 'json',
    ];
    public $timestamps = false;
    protected $fillable = ['nav_id', 'parent_id', 'ordering',
'name', 'link', 'description', 'published', 'access', 'params'];


    public function items()
    {
        return $this->belongsTo('App\Model\Nav, 'nav_id', 'id');
    }

}
class News extends Model
{
    protected $table = 'News';
    public $timestamps = true;

    protected $fillable = ['title', 'image', 'content',
'created_by', 'updated_by', 'published_at'];
```

```
}
class Pages extends Model
{
    protected $table = 'Pages';
    protected $casts = [
        'params' => 'json',
    ];
    public $timestamps = true;
    protected $fillable = ['type', 'parent_id', '_lft', '_rgt',
'title', 'url', 'header', 'content', 'access', 'published_at'];
    protected $guarded = array('id');

}
class Reserve extends Model
{

    protected $table = 'Reserve';
    protected $with = [
    ];
    protected $casts = [
        'options' => 'json',
    ];

    protected $fillable = ['user_id', 'event_id', 'seat_id',
'row', 'seat', 'price', 'status'];

    protected $dateFormat = "Y-m-d H:i:s";
    public $timestamps = true;

    public function event()
    {
        return $this->belongsTo('App\Model\Events', 'event_id',
'id');
    }

    public function user()
    {
        return $this->belongsTo('App\Model\User', 'user_id',
'id');
    }

}
class Token extends Model
{
    protected $table = 'Tokens';
    public $timestamps = true;
    protected $fillable = ['user_id', 'token', 'ip',
'user_agent', 'last_visit'];

    public function user()
    {
        return $this->hasMany('App\Model\User', 'id',
'user_id');
    }

    public function tokens()
```

```
    {
        return $this->belongsTo('App\Model\User', 'id',
'user_id');
    }

}
class User extends Model
{
    protected $table = 'Users';
    public $timestamps = true;

    protected $hidden = ['password'];

    protected $fillable = ['login', 'email', 'password',
'first_name', 'patronymic', 'last_name', 'phone', 'state',
'access', 'group'];

    public function user()
    {
        return $this->belongsTo('App\Model\Token', 'id',
'user_id');
    }

    public function tokens()
    {
        return $this->hasMany('App\Model\Token', 'user_id',
'id');
    }

    public function events()
    {
        return $this->hasMany('App\Model\Events', 'created_by',
'id');
    }

    public function reserve()
    {
        return $this->hasMany('App\Model\Reserve', 'user_id',
'id');
    }
}
```

Приложение Н

Листинг кода unit-тестов модуля контроля доступа

(рекомендуемое)

```
<?php
use PHPUnit\Framework\TestCase;
$_SERVER['DOCUMENT_ROOT'] = dirname(__FILE__) . "/../";
require 'config/bootstrap.php';

class UserTest extends TestCase
{
  protected $app;
  protected $user;

  public function setUp()
  {
    $this->app = new App\Application();
    $this->user = $this->app['user'];
  }

  public function testIsAuth()
  {
    $result = $this->user->isAuth();
    $this->assertFalse($result);
  }
  public function testLoginBadCredentialsException()
  {
    $this-
>expectException(App\Exception\Auth\BadCredentialsException::cla
ss);
    $user = $this->user->login('admin', 'error');
  }

  public function testLoginUsernameNotFoundException()
  {
    $this-
>expectException(App\Exception\Auth\UsernameNotFoundException::c
lass);
    $user = $this->user->login('error', 'admin');
  }
  public function testLoginSuccess()
  {
    $user = $this->user->login('admin', 'admin');

    $result = $user['login'];

    $this->assertEquals('admin', $result);
  }

  public function testLoginRememberSuccess()
  {
    $user = $this->user->login('admin', 'admin', true);

    $result = $user['login'];

    $this->assertEquals('admin', $result);
  }

  public function testIsAuthTrue()
  {
```

```
    $result = $this->user->isAuth();

    $this->assertTrue($result);
  }

  public function testIsAuthCookie()
  {
    $this->app['session']->remove('user');
    $this->app['session']->remove('user');

    $result = $this->user->isAuth();

    $this->assertTrue($result);
  }

  public function testGetUID()
  {
    $result = $this->user->getUID();

    $this->assertEquals(1, $result);
  }

  public function testGetToken()
  {
    $result = $this->user->getToken();

    $this->assertRegExp('/^[a-f0-9]{32}$/i', $result);
  }

  public function testGetUser()
  {
    $user = $this->user->getUser();

    $result = $user['login'];

    $this->assertEquals('admin', $result);
  }

  public function testSetOnline()
  {
    $result = $this->user->online();

    $this->assertTrue($result);
  }

  public function testSetOnlineForce()
  {
    $result = $this->user->online(true);

    $this->assertTrue($result);
  }

  public function testLogout()
  {
    $result = $this->user->logout();
```

```
    $this->assertEquals(true, $result);
  }

  public function testJoinFail()
  {
    $result = $this->user->join();

    $this->assertEquals(false, $result);
  }

  public function testJoinSuccess()
  {
    $result = $this->user->join([
      'login' => 'admin2',
      'password' => 'admin2',
    ]);

    $this->assertEquals(true, $result);
  }

}
```

Приложение П

Листинг кода unit-тестов модуля бронирования

(рекомендуемое)

```
<?php
use App\Exception\Booking\AccessDeniedException;
use App\Exception\Booking\ArgumentsEmptyException;
use App\Exception\Booking\BookingException;
use App\Exception\Booking\EventNotExistException;
use App\Exception\Booking\NotExistException;
use App\Exception\Booking\PlacesBookedException;
use App\Exception\Booking\UserNotAuthorizedException;
use App\Exception\Booking\UserNotExistException;
use PHPUnit\Framework\TestCase;
use \App\Module\Booking;
use \App\Module\User;

$_SERVER['DOCUMENT_ROOT'] = dirname(__FILE__) . "/../";
require 'configs/bootstrap.php';

class UserTest extends TestCase
{

    protected $app;
    protected $user;
    protected $reserve;

    public function setUp()
    {
        $this->app = new \App\Application();
        $this->reserve = new Booking($this->app);
    }
    public function tearDown()
    {
        $_COOKIE = [];
        $_SESSION = [];
        $this->app['user'] = null;
    }

    public function testBookingAllExceptionAuth()
    {
        $this-
>expectException(UserNotAuthorizedException::class);
        $this->reserve->all();
    }

    public function testBookingAll()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $result = $this->reserve->all();
        $this->assertNotEmpty($result);
    }

    public function testBookingAllExceptionUserNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));
```

```
        $this->expectException(UserNotExistException::class);
        $result = $this->reserve->all(15);
    }

    public function testBookingAllExceptionAccessDenied()
    {
        $_COOKIE['key'] = 'ebd2ff0d54aaaea6249f4eda17e1f1cb';
        $this->app->register('user', new User($this->app));

        $this->expectException(AccessDeniedException::class);
        $this->reserve->all(1);
    }

    public function testBookingGet()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $reserve = $this->reserve->get(32);
        $user_id = $this->app['user']->getUID();

        if ($user_id == 1 && $reserve['event_id'] == 1) {
            $r = true;
        } else {
            $r = false;
        }

        $this->assertEquals(true, $r);
    }

    public function testBookingGetExceptionNotAuth()
    {
        $this->expectException(BookingException::class);
        $this->reserve->get(32);
    }

    public function testBookingGetExceptionNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(NotExistException::class);
        $this->reserve->get(1);
    }

    public function testBookingGetExceptionNotArguments()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(ArgumentsEmptyException::class);
        $this->reserve->get();
    }

    public function testBookingCreateExceptionNotAuth()
    {
```

```
        $this-
>expectException(UserNotAuthorizedException::class);
        $this->reserve->create(1);
    }

    public function testBookingCreateExceptionArgmuentsEmpty()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(ArgumentsEmptyException::class);
        $this->reserve->create();
    }

    public function testBookingCreateExceptionAccessDenied()
    {
        $_COOKIE['key'] = 'ebd2ff0d54aaaea6249f4eda17e1f1cb';
        $this->app->register('user', new User($this->app));

        $this->expectException(AccessDeniedException::class);
        $this->reserve->create(1, 1, null, 1);
    }

    public function testBookingCreateExceptionUserNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(UserNotExistException::class);
        $this->reserve->create(1, 1, null, 15);
    }

    public function testBookingCreateExceptionEventNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(EventNotExistException::class);
        $this->reserve->create(1050, 1, null, 2);
    }

    public function testBookingCreateExceptionPlacesAreBooked()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(PlacesBookedException::class);
        $this->reserve->create(1);
    }

    public function testBookingCreateExceptionUnknownError()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(BookingException::class);
```

```
        $this->reserve->create(2, null, null, null, true);
    }

    public function testBookingCreate()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $reserve = $this->reserve->create(2);

        if ($reserve['event_id'] === 2) {
            $r = true;
        } else {
            $r = false;
        }

        $this->assertEquals(true, $r);
    }

    public function testBookingUpdateExceptionNotAuth()
    {
        $this-
>expectException(UserNotAuthorizedException::class);
        $this->reserve->update(1);
    }

    public function testBookingUpdateExceptionArgmuentsEmpty()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(ArgumentsEmptyException::class);
        $this->reserve->update();
    }

    public function testBookingUpdateExceptionAccessDenied()
    {
        $_COOKIE['key'] = 'ebd2ff0d54aaaea6249f4eda17e1f1cb';
        $this->app->register('user', new User($this->app));

        $this->expectException(AccessDeniedException::class);
        $this->reserve->update(1, 1, null, 1);
    }

    public function testBookingUpdateExceptionUserNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(UserNotExistException::class);
        $this->reserve->update(1, 1, null, 15);
    }

    public function testBookingUpdateExceptionNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
```

```
        $this->app->register('user', new User($this->app));

        $this->expectException(NotExistException::class);
        $this->reserve->update(1050, 1, null, 2);
    }

    public function testBookingUpdate()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $reserve = $this->reserve->update(32, 2);

        if ($reserve['id'] === 32 && $reserve['qty'] === 1) {
            $r = true;
        } else {
            $r = false;
        }

        $this->assertEquals(true, $r);
    }

    public function testBookingRemoveExceptionNotAuth()
    {
        $this-
>expectException(UserNotAuthorizedException::class);
        $this->reserve->remove(32);
    }

    public function testBookingRemoveExceptionArgmuentsEmpty()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(ArgumentsEmptyException::class);
        $this->reserve->remove();
    }

    public function testBookingRemoveExceptionNotExist()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(NotExistException::class);
        $this->reserve->remove(1050);
    }

    public function testBookingRemoveExceptionUnknownError()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $this->expectException(BookingException::class);
        $this->reserve->remove(32, true);
    }
```

```
    public function testBookingRemove()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $deleted = $this->reserve->remove(32);

        $this->assertEquals(true, $deleted);
    }

    public function testBookingRemoveArray()
    {
        $_COOKIE['key'] = 'f0c384a20e9c6588db4e05aecc554192';
        $this->app->register('user', new User($this->app));

        $deleted = $this->reserve->remove([32, 34, 36]);
        if ([32, 34, 36] == $deleted) {
            $r = true;
        } else {
            $r = false;
        }
        $this->assertEquals(true, $r);
    }

    public function testBookingRemoveAsUser()
    {
        $_COOKIE['key'] = 'ebd2ff0d54aaaea6249f4eda17e1f1cb';
        $this->app->register('user', new User($this->app));

        $this->expectException(NotExistException::class);
        $deleted = $this->reserve->remove(32);
        $this->assertEquals(true, $deleted);
    }

}
```

Приложение Р

Тестирование веб-приложения

(рекомендуемое)

Функциональное тестирование клиентской части веб-приложения

1) Загрузка обычной страницы (рисунок Р.1).

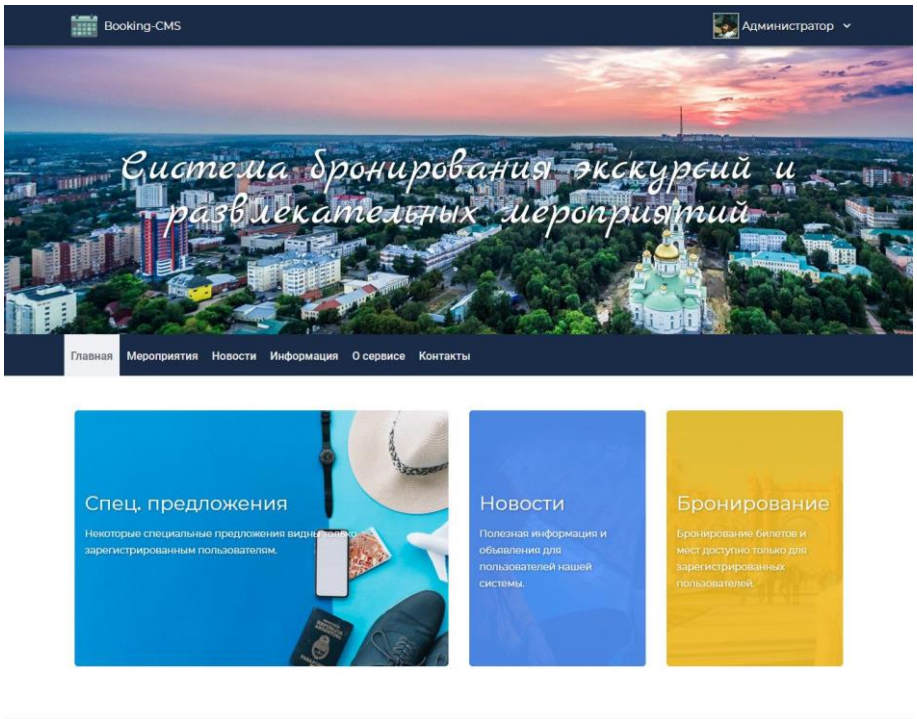

Рисунок Р.1 – Скриншот страницы мероприятий

2) Загрузка несуществующей страницы (рисунок Р.2).

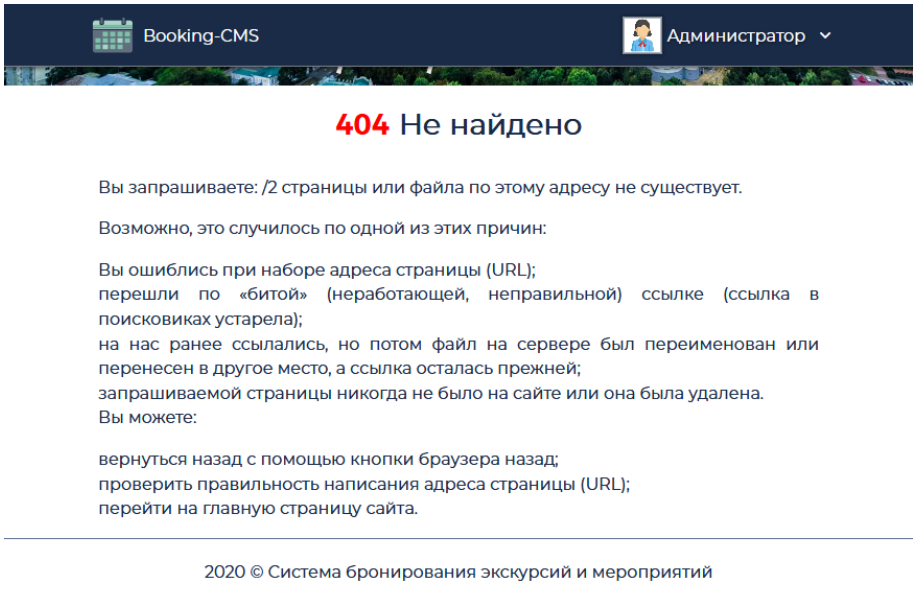

Рисунок Р.2 – Скриншот запроса несуществующей страницы

3) Загрузка страницы регистрации (рисунок Р.3).

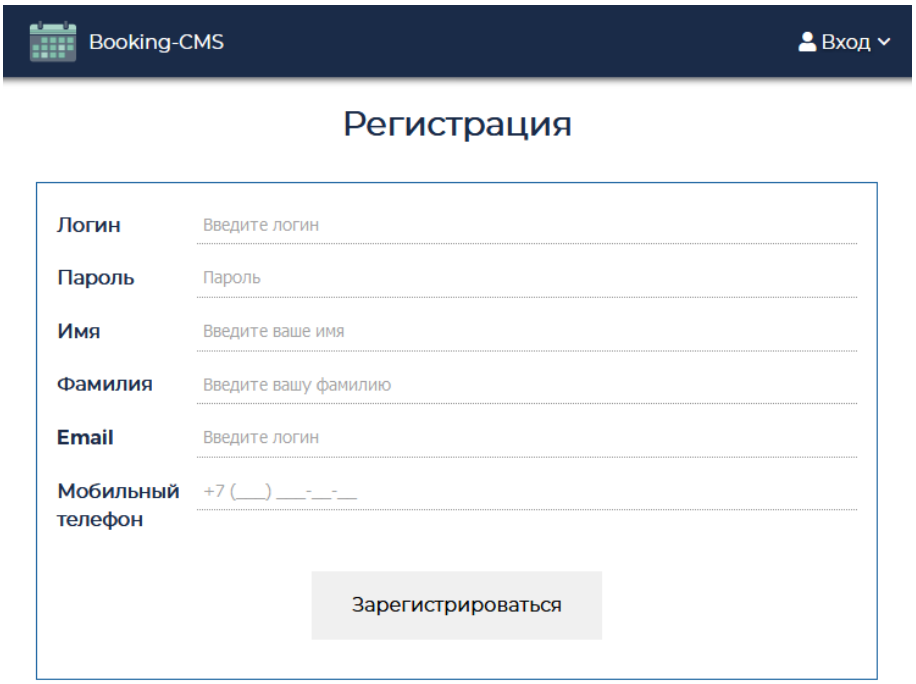

Рисунок Р.3 – Скриншот страницы мероприятий

4) Авторизация (рисунок Р.4).

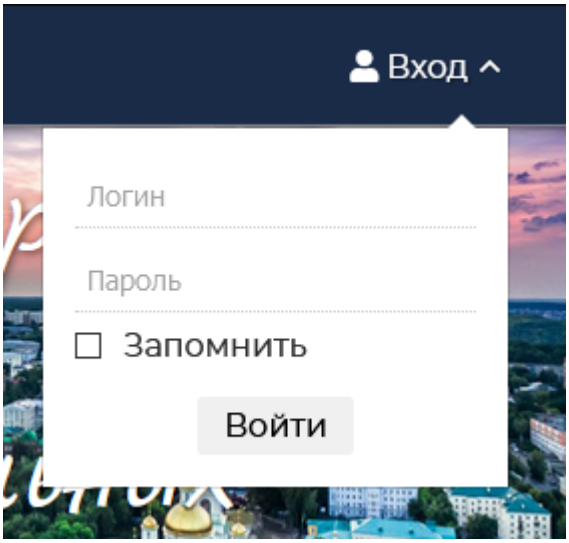

Рисунок Р.4 – Скриншот формы авторизации

5) Просмотр информации о мероприятии (рисунки Р.5, Р.6, Р.7).

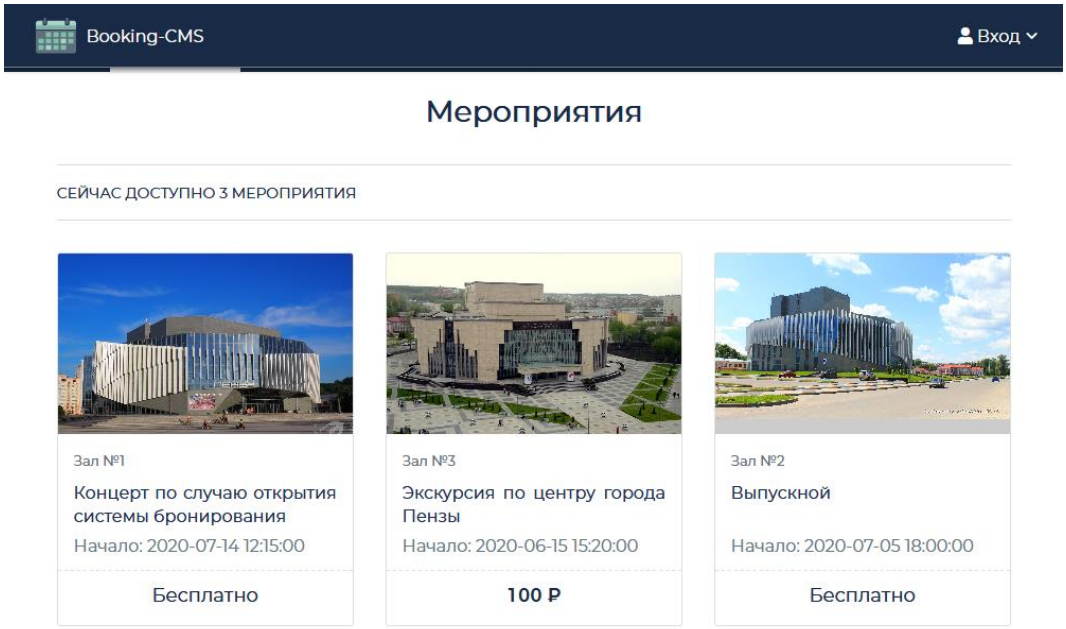

Рисунок Р.5 – Скриншот страницы мероприятий

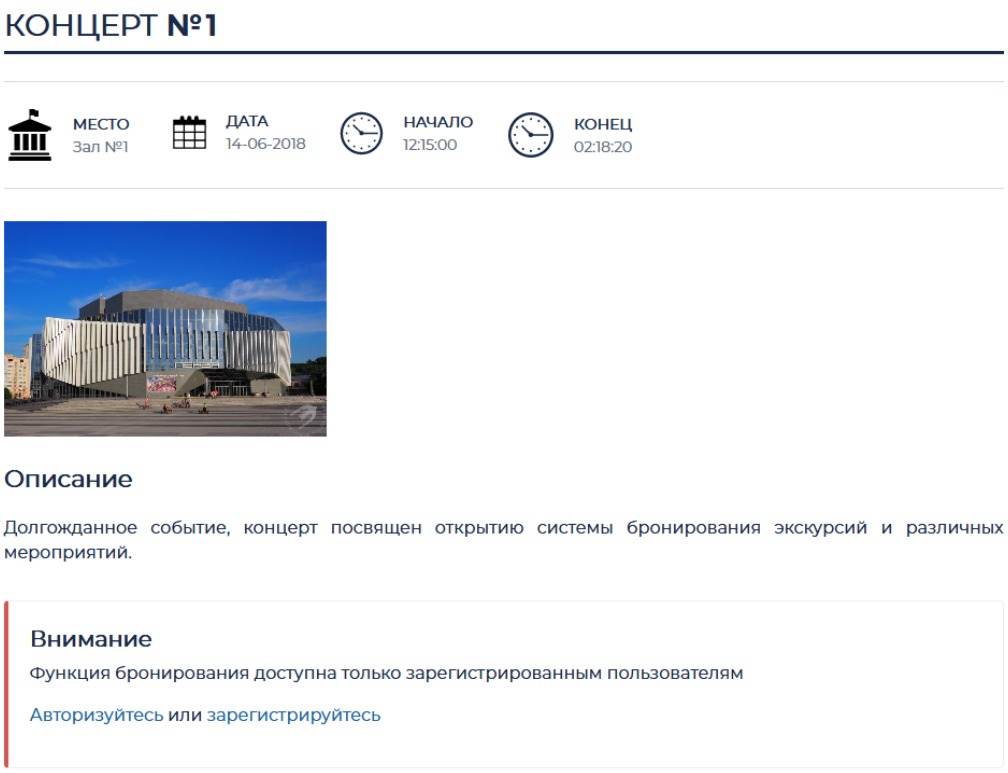

Рисунок Р.6 – Скриншот страницы мероприятия для неавторизованного пользователя

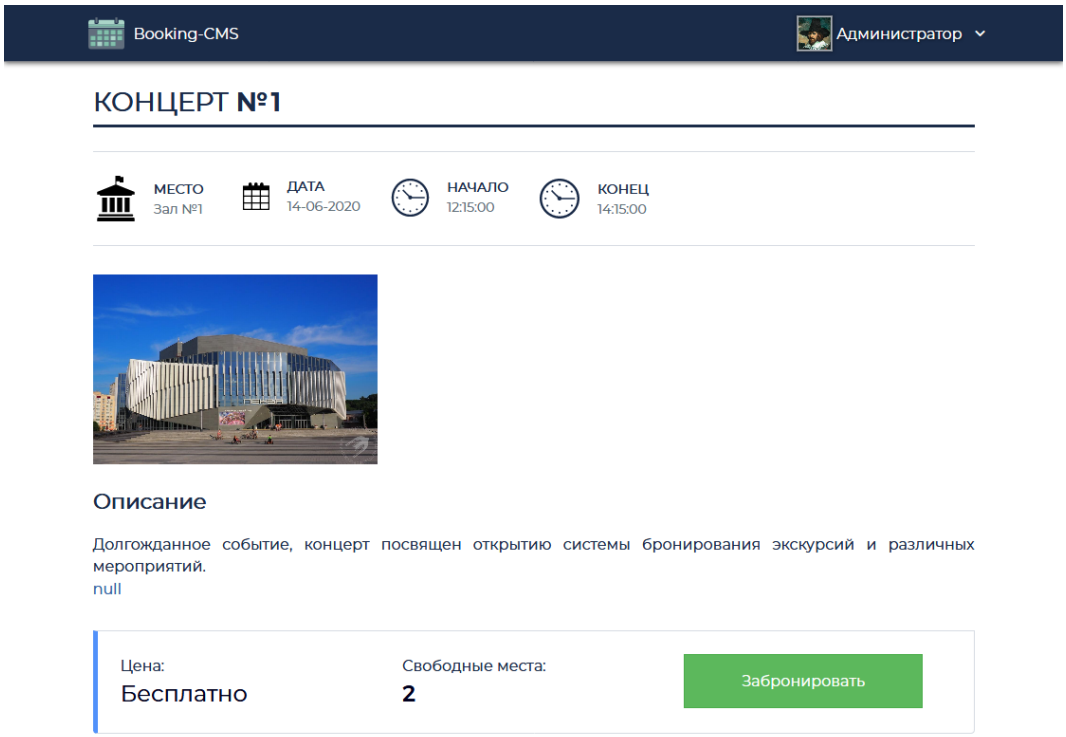

Рисунок Р.7 – Скриншот страницы мероприятия для авторизованного пользователя

6) Просмотр информации о забронированных билетах в профиле пользователя (рисунок Р.8).

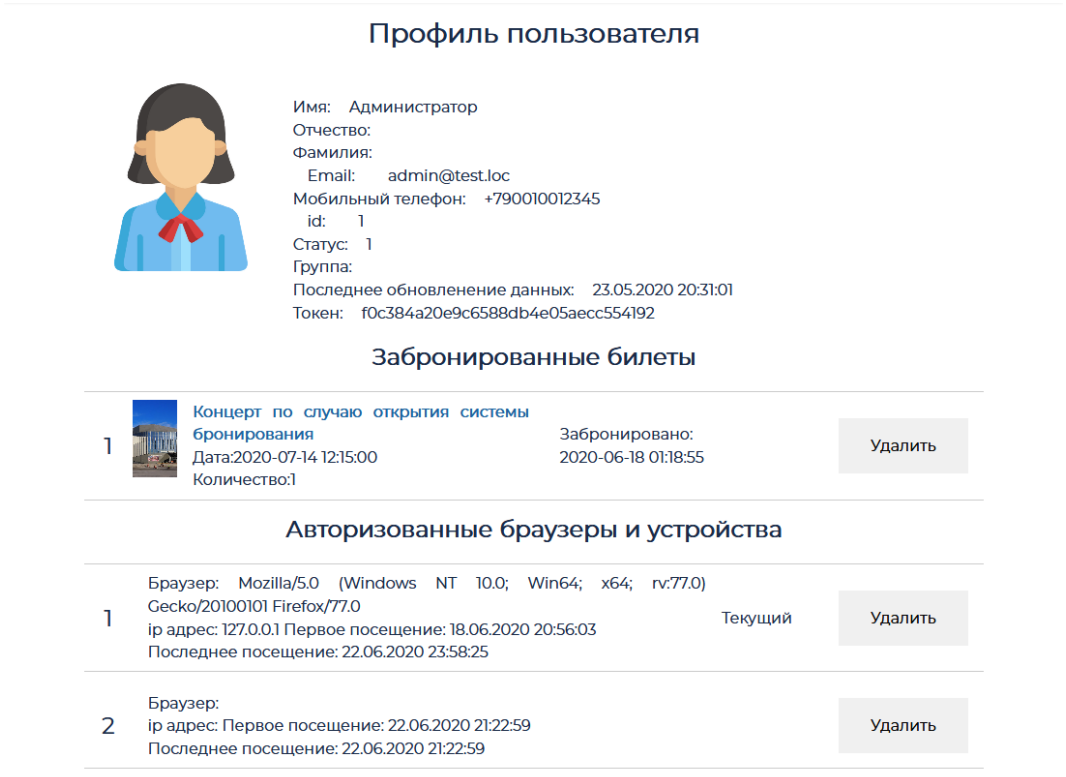

Рисунок Р.8 – Скриншот страницы профиля пользователя

7) Вход под логином администратора – загружена страница панель управления (рисунок Р.9).

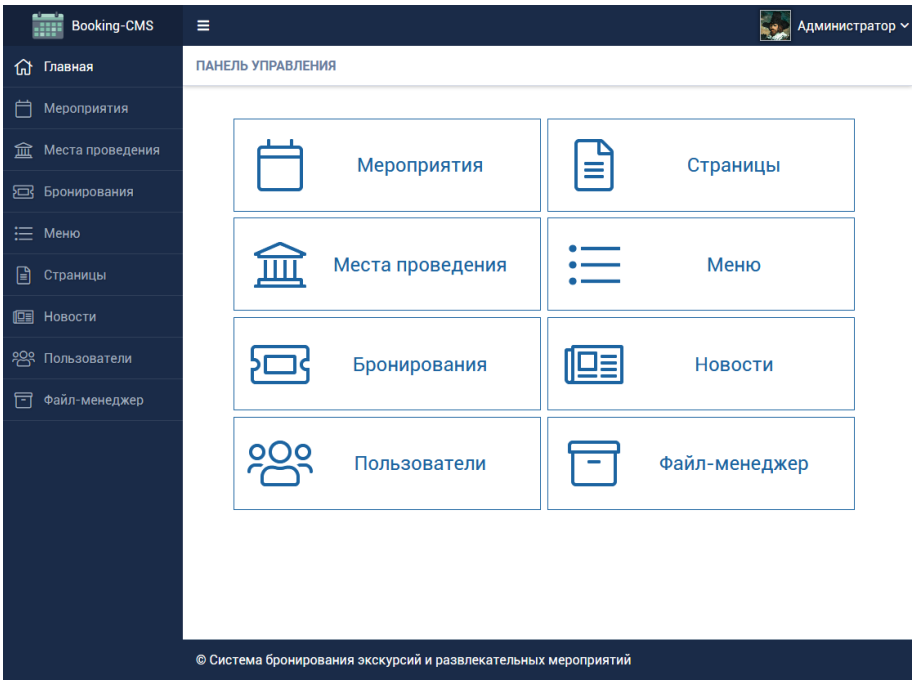

Рисунок Р.9 – Скриншот главной страницы панели управления

8) Просмотр списка мероприятий в панели управления (рисунок Р.10).

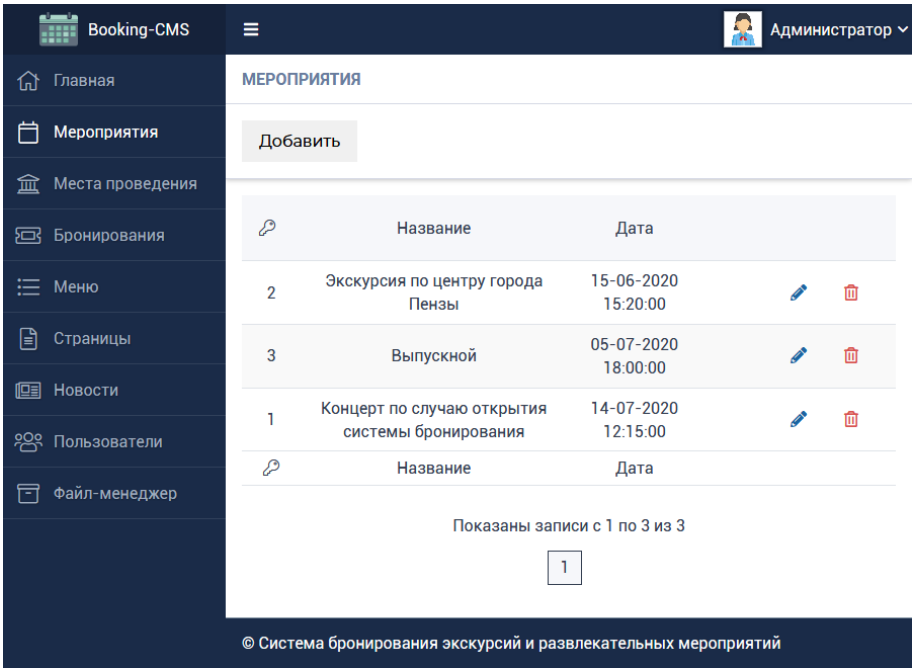

Рисунок Р.10 – Скриншот списка мероприятий в панели управления

9) Добавления мероприятия в панели управления (рисунок Р.11).

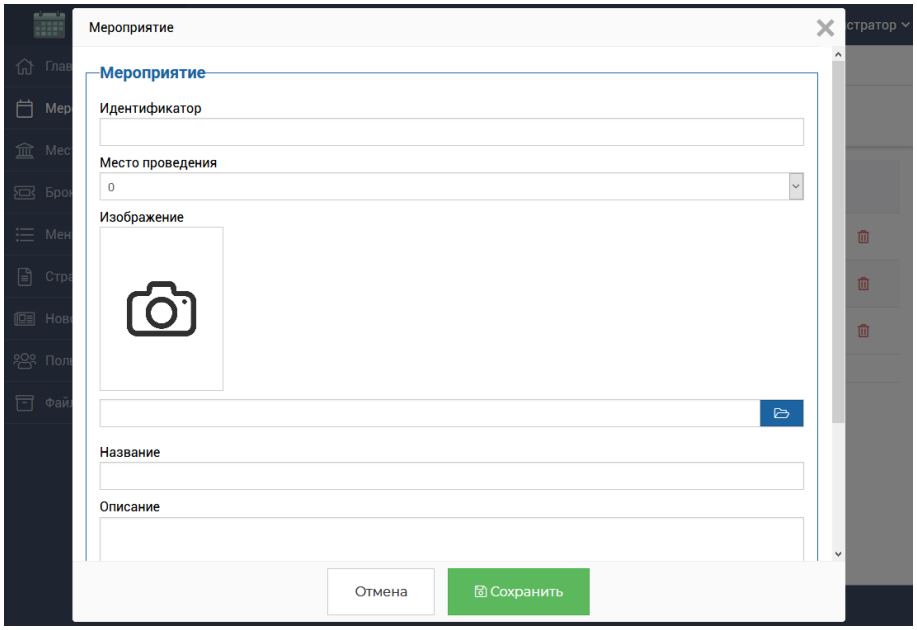

Рисунок Р.11 – Скриншот добавления мероприятия в панели управления

10) Редактирование мероприятия в панели управления (рисунок Р.12).

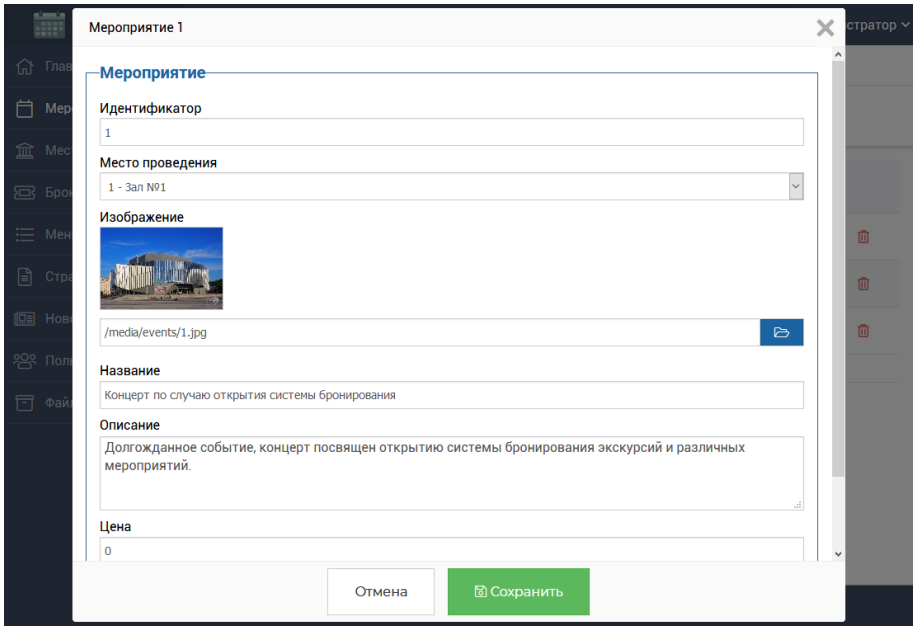

Рисунок Р.12 – Скриншот редактирование созданного мероприятия в панели управления

11) Удаление записи о бронировании в панели управления (рисунки Р.13, Р.14, Р.15).

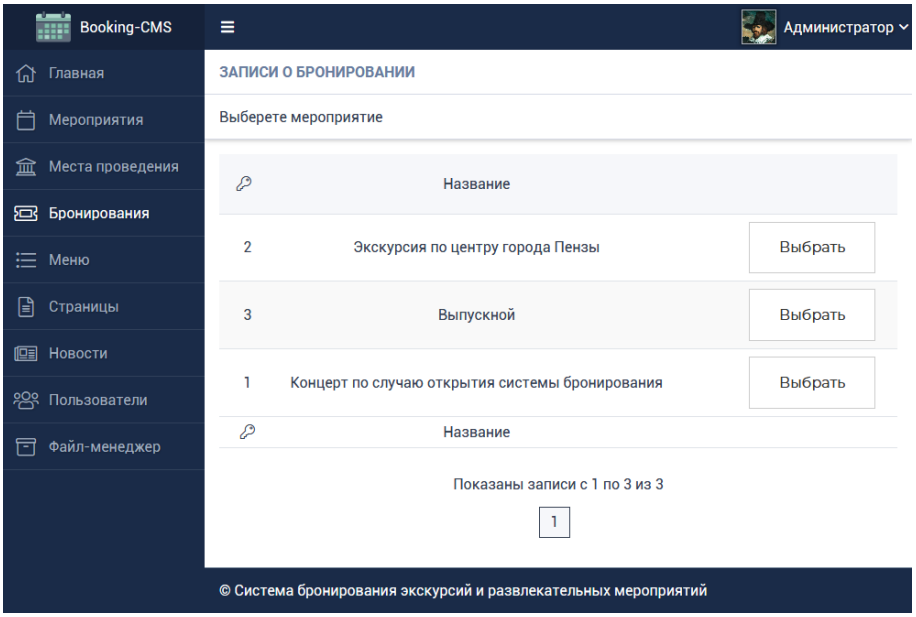

Рисунок Р.13 – Скриншот страницы выбора мероприятия для просмотра забронированных билетов

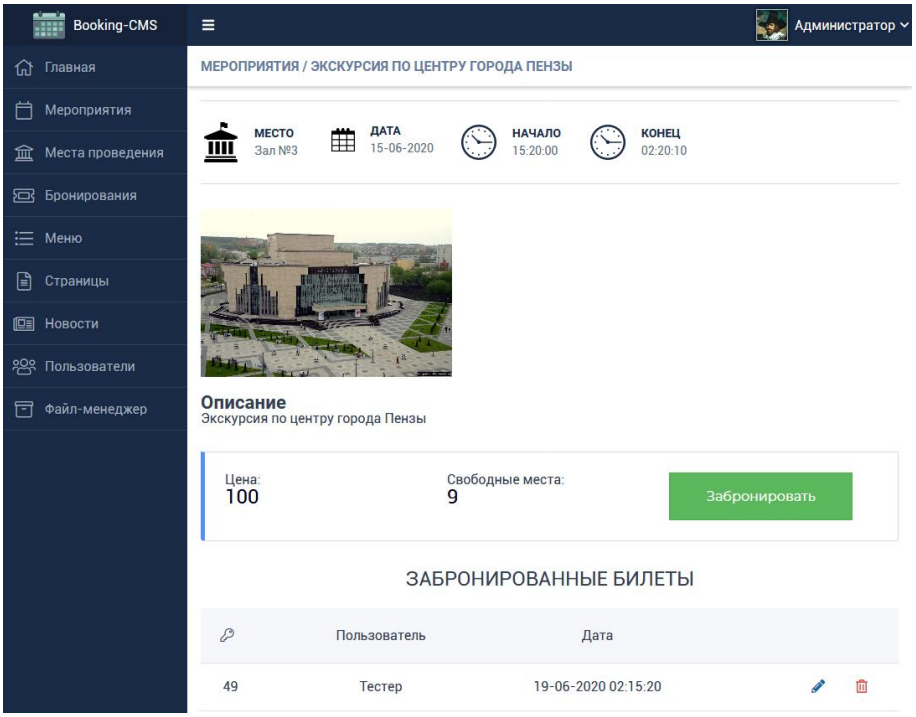

Рисунок Р.14 – Скриншот страницы просмотра забронированных билетов на мероприятие

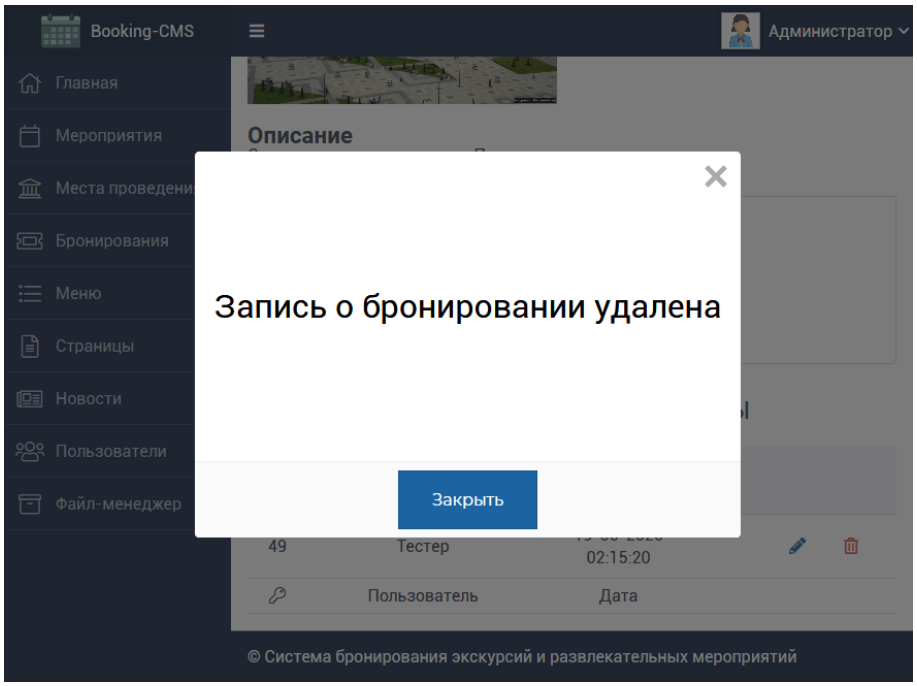

Рисунок Р.15 – Скриншот сообщеня об удалении записи о бронировании билета на мероприятие в панели управления

12) Файловый менеджер в панели управления (рисунок Р.16).

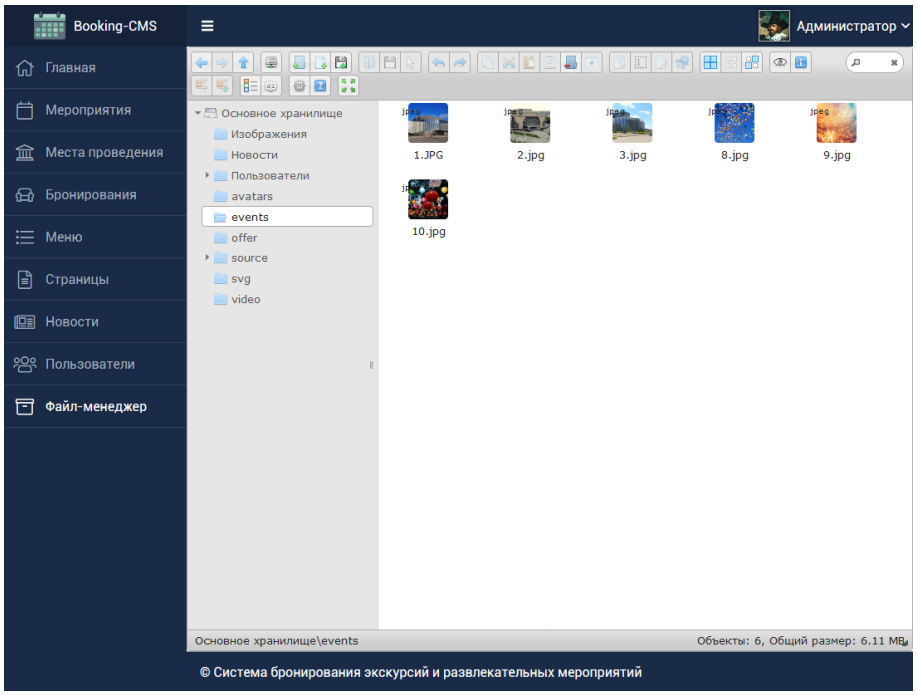

Рисунок Р.16 – Скриншот страницы файлового менеджера в панели управления