МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения для арендаторов и арендодателей спортивных залов

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-305.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ Е.В. Иванова

Автор работы,
студент группы КЭ-401
_____ С.А. Лазарева

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студентке группы КЭ-401
Лазаревой Снежане Андреевне,
обучающейся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка веб-приложения для арендаторов и арендодателей спортивных залов.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Заяц А.М. Основы WEB технологий. Разработка WEB-приложений современными инструментальными средствами: учебно-методическое пособие. // А.М. Заяц. – Санкт-Петербург: СПбГЛТУ, 2021. – 116 с.
3.2. Django: The web framework for perfectionists with deadlines. Django Documentation. [Электронный ресурс] URL: https://docs.djangoproject.com/en/5.0 (дата обращения: 29.01.2024 г.).
3.3. Вейцман В.М. Проектирование информационных систем: учебное пособие. // Санкт-Петербург: Лань, 2019. – 316 с.
3.4. Флэнаган Д. JavaScript: карманный справочник, 3-е изд. // Издательство Вильямс, 2013. – 320 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить обзор open-source систем с функцией онлайн-бронирования.
4.2. Разработать техническое задание на создание веб-приложения для арендодателей и арендаторов спортивных залов.

4.3. Спроектировать веб-приложение.
4.4. Выполнить реализацию веб-приложения.
4.5. Провести тестирование веб-приложения.
5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. Е.В. Иванова

Задание принял к исполнению С.А. Лазарева

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В течение последних лет здоровый образ жизни (ЗОЖ) остается востребованной темой в обществе, в связи с чем популярность различных видов физической нагрузки, от классического фитнеса до танцев, продолжает расти [1]. Популярность ЗОЖ обеспечивает спрос на аренду площадок, предоставляющих залы для занятий спортом. Арендаторы и арендодатели спортивных залов заинтересованы в программном продукте, представляющем собой платформу, объединяющую разные площадки, где можно легко выставлять для аренды и арендовать залы, подходящие по расположению, площади и стоимости.

Веб-формат приложения предпочтителен, поскольку он обеспечивает удобный доступ к сервису из любого устройства, подключенного к интернету. Пользователи могут легко получить доступ к приложению через браузер на своем компьютере, смартфоне или планшете, без необходимости установки дополнительного программного обеспечения. Более того, веб-формат обеспечивает масштабируемость и гибкость, позволяя легко обновлять и расширять функциональность приложения в соответствии с потребностями пользователей.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения для арендаторов и арендодателей спортивных залов. Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить обзор open-source систем с функцией онлайн-бронирования;
2) разработать техническое задание на создание веб-приложения для арендодателей и арендаторов спортивных залов;
3) спроектировать веб-приложение;
4) выполнить реализацию веб-приложения;
5) провести тестирование веб-приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложения. Объем работы составляет 46 страниц, объем списка литературы – 23 источника.

В первой главе описывается предметная область и производится анализ аналогичных проектов и существующих инструментов разработки.

Вторая глава посвящена проектированию разрабатываемого приложения, включая описание требований к системе, варианты ее использования, проектирование архитектуры приложения, базы данных, алгоритмов и интерфейса.

В третьей главе перечислены выбранные программные средства разработки веб-приложения и описана реализация функциональных модулей системы и интерфейса.

В четвертой главе приведены результаты функционального тестирования веб-приложения.

В заключении описаны результаты, полученные во время выполнения выпускной квалификационной работы.

В приложении находятся таблицы, описывающие спецификации вариантов использования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ аналогичных проектов

В ходе обзора был выявлен ряд платформ, реализующих функцию онлайн-бронирования.

Avito

Avito [1] – онлайн-платформа для объявлений о продаже и покупке товаров и услуг. Avito включает широкий спектр категорий предлагаемых товаров и услуг, и аренда помещений, включая спортивные и танцевальные залы, является лишь одной из них. Основным преимуществом платформы в данном случае является количество и многообразие предлагаемых площадок, однако эта вариативность и наличие множества других категорий затрудняет поиск. Кроме того, стоимость и свободные часы зала не всегда указаны в объявлении, а для бронирования требуется лично связываться с каждым арендодателем. На рисунке 1 показан результат поиска по запросу «аренда танцевального зала» на платформе.

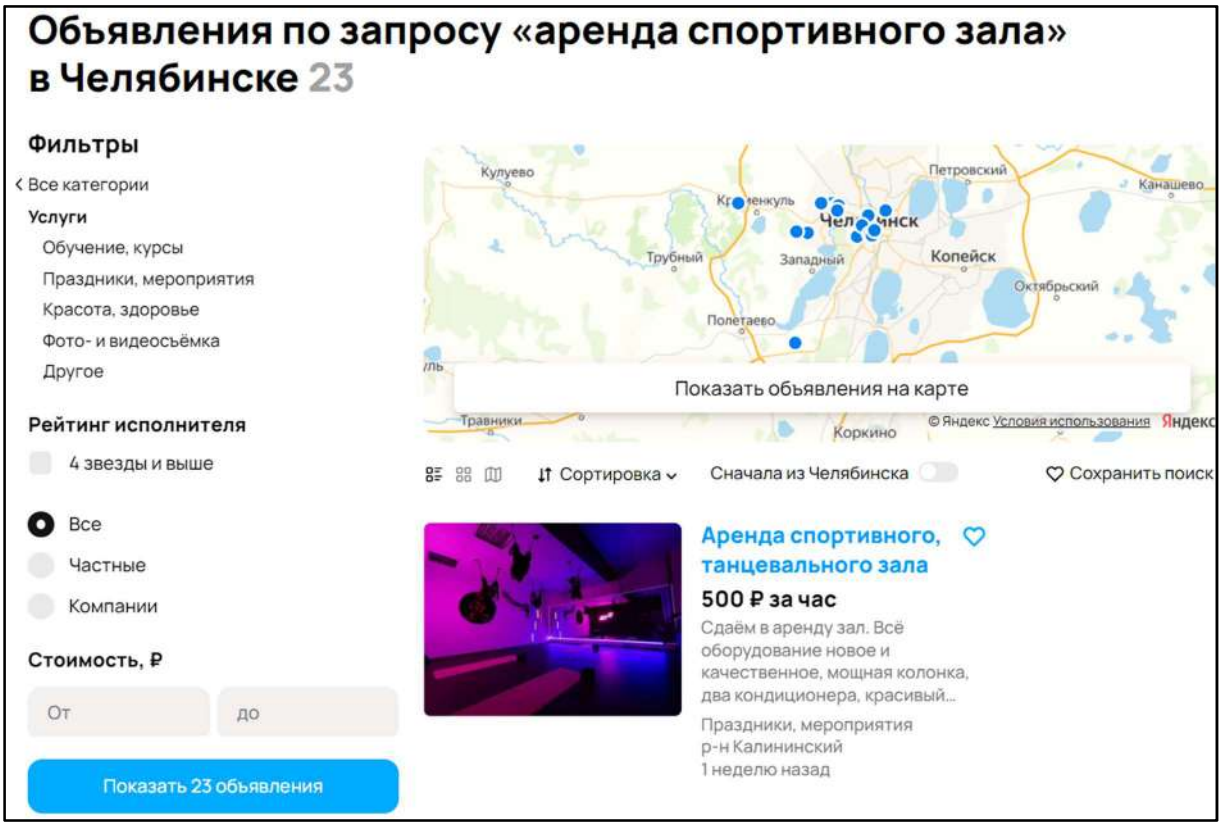

Рисунок 1 – Платформа Avito

Популярной платформой, схожей с Avito, является Юла [2]. Ее преимущества и недостатки в контексте бронирования залов аналогичны Avito.

Zalti.ru

Zalti.ru [3] – веб-приложение, реализованное на основе платформы VK Apps. Оно позволяет арендовать залы, принадлежащие одной конкретной компании. Для бронирования пользователь может либо нажать на нужный временной промежуток в календаре, либо открыть необходимую форму самостоятельно. Интерфейс Zalti представлен на рисунке 2. Приложение также позволяет вести учет оформленных броней. Одним из основных ограничений платформы является тот факт, что пользователю необходимо иметь аккаунт в социальной сети VK. С одной стороны, это автоматически интегрирует сервис в социальную сеть, позволяя арендаторам и владельцам залов получать различные уведомления о бронированиях в личных сообщениях. С другой стороны, не все являются активными пользователями VK, что ограничивает круг потенциальных клиентов Zalti.

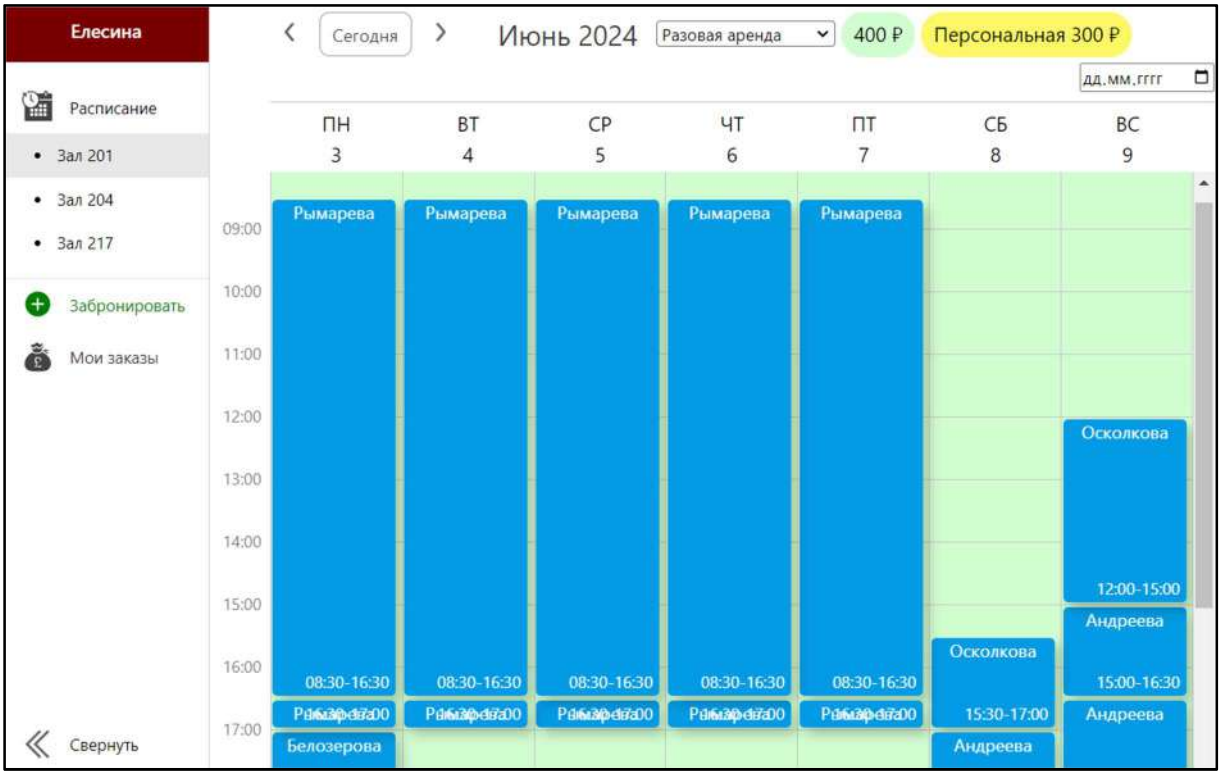

Рисунок 2 – Веб-приложение Zalti

«9 ЗАЛОВ»

Онлайн-бронирование реализовано на сайте московской студии «9 ЗАЛОВ» [4]. Аналогично Zalti, здесь можно арендовать залы, принадлежащие только одной студии. Процесс бронирования автоматизирован частично: пользовать может выбрать нужный зал и заполнить форму заявки на аренду, после чего с ним свяжутся. Отличительной чертой сайта является удобное визуальное представление ассортимента залов, где каждому помещению соответствует подробное описание с фотографиями (рисунок 3).

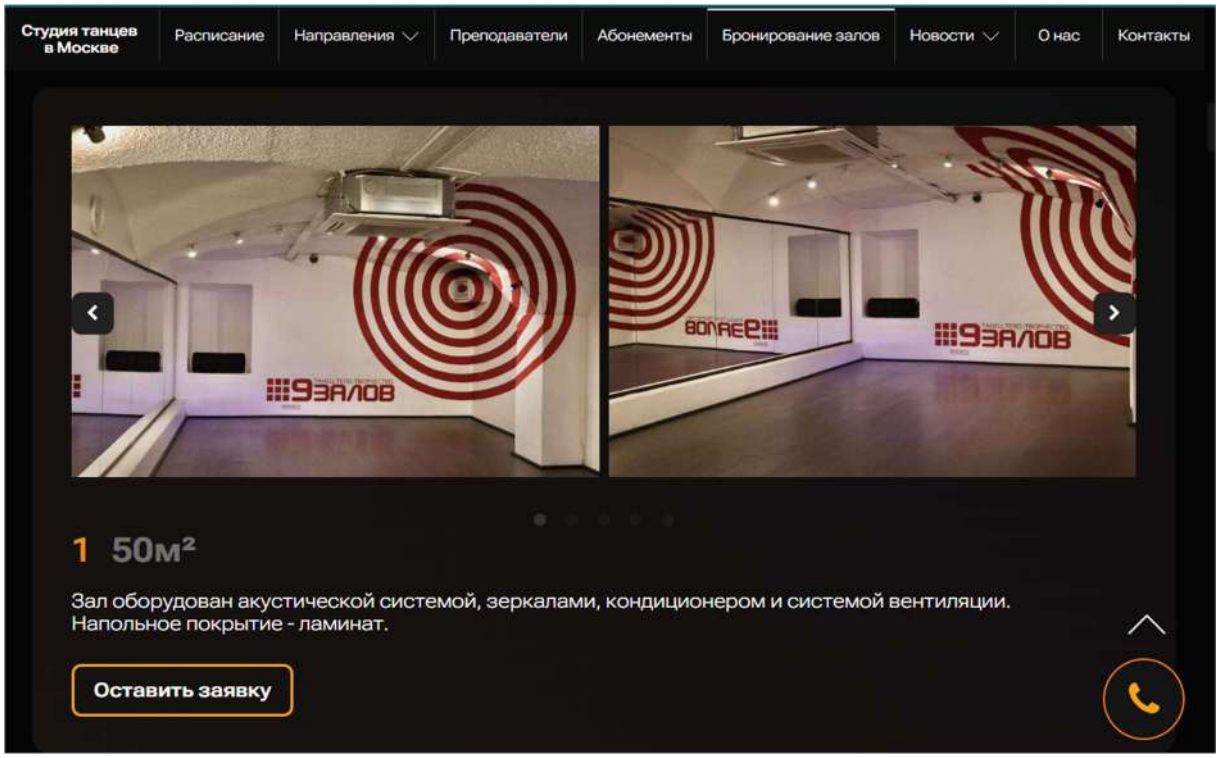

Рисунок 3 – Раздел бронирования на сайте студии «9 ЗАЛОВ»

MUSbooking

Платформа MUSbooking [5] отличается от всех вышеперечисленных тем, что включает множество творческих площадок, принадлежащих различным компаниям, и при этом процесс бронирования автоматизирован для большинства залов. Также удобно реализованы фильтры для поиска и карточки площадок (рисунок 4). Главным недостатком платформы является то, что на сегодня для бронирования доступны только залы в Москве и МО.

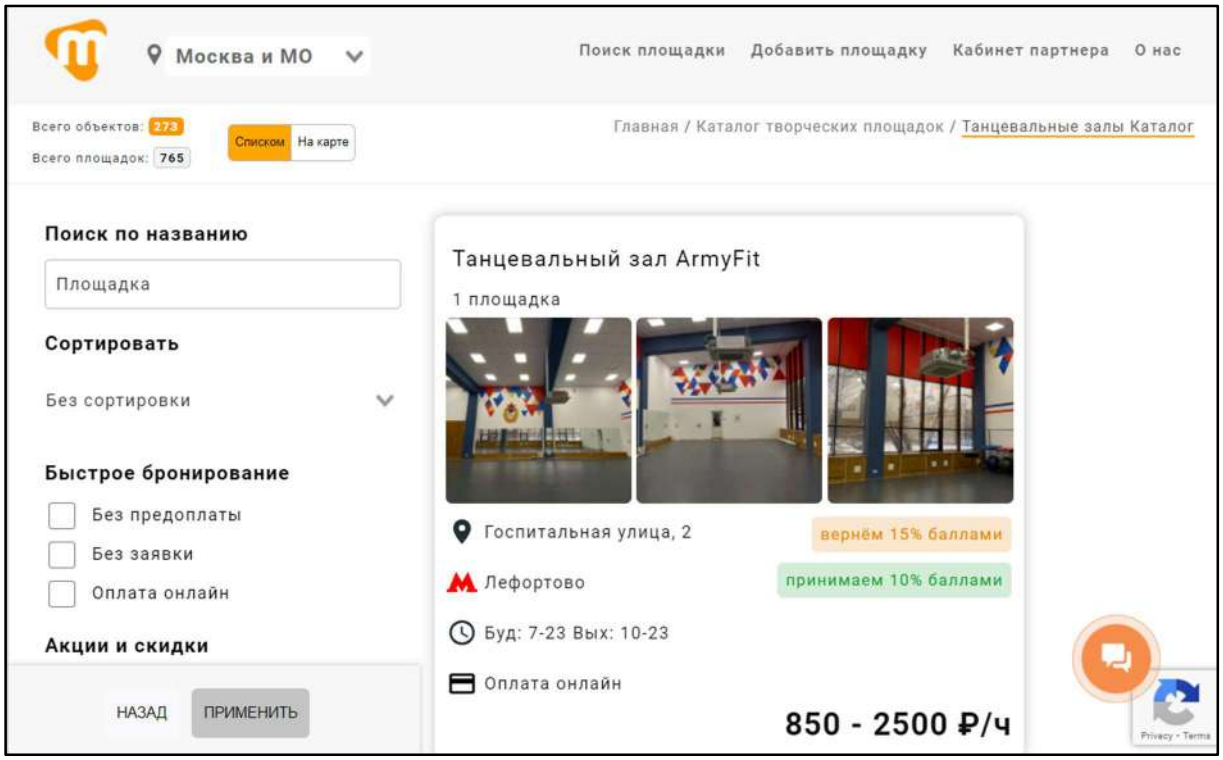

Рисунок 4 – Платформа MUSbooking

Сравнительный анализ аналогичных систем

В таблице 1 представлен сравнительный анализ рассмотренных аналогов.

Таблица 1 – Сравнительный анализ аналогов

Критерии сравнения	Avito	Zalti.ru	«9 залов»	MUSbooking
Специализация на спортивных залах	–	+	+	+/–
Автоматическое бронирование	–	+	+/–	+
Залы от разных арендодателей	+	–	–	+
Интерактивный календарь занятости зала	–	+	–	+
Галерея фотографий для каждого зала	+	–	+	+
Распространение по регионам	+	–	–	–

Сравнительный анализ показал, что наиболее соответствующей всем критериям системой является MUSbooking. Однако ее основным недостатком является отсутствия на платформе спортивных и танцевальных площадок многих регионов, в том числе Челябинской области.

1.2. Обзор существующих решений для реализации проекта

Для разработки frontend-части веб-приложения используются языки HTML [6] и CSS [7].

HTML (HyperText Markup Language) – это язык разметки, который используется для создания структуры веб-страниц. Он определяет содержимое страницы, такое как заголовки, параграфы, списки, изображения, ссылки и другие элементы, которые формируют основу веб-документа.

CSS (Cascading Style Sheets) используется для оформления и стилизации веб-страницы. Он позволяет разработчикам определять цвета, шрифты, размеры, расположение и другие свойства элементов на странице.

Совместное использование HTML и CSS позволяет создавать удобный и визуально приятный интерфейс веб-страниц, помогая улучшить опыт взаимодействия пользователей с приложением.

Одним из самых востребованных языков для backend-разработки веб-приложений является Python [8]. Во-первых, это язык с открытым исходным кодом, который имеет большое сообщество разработчиков, постоянно работающих над улучшением самого языка и его библиотек. Во-вторых, у Python простой и понятный синтаксис, что упрощает его использование, освобождая разработчиков от необходимости тратить много времени на глубокое изучение языка. В-третьих, Python обладает широкими возможностями для интеграции с другими языками программирования и технологиями. Он может легко работать с базами данных, использоваться для создания веб-серверов, а также интегрироваться с другими инструментами и сервисами.

Благодаря открытому исходному коду и широкому сообществу разработчиков, Python имеет множество open-source библиотек и фреймворков, созданных специально для облегчения процесса разработки веб-приложений. Так, некоторые из популярных фреймворков для веб-разработки на Python включают Django, Flask и Ruby on Rails.

Django [9] – это высокоуровневый веб-фреймворк, который предоставляет разработчикам гибкие инструменты для быстрой и эффективной

разработки веб-приложений. Он включает в себя множество готовых компонентов, таких как система аутентификации, ORM для работы с базами данных, а также механизмы для обработки HTTP-запросов и отображения HTML-шаблонов. Django обеспечивает высокую производительность и безопасность благодаря встроенным механизмам защиты от множества распространенных атак. Наконец, Django имеет модульную структуру, позволяющую интегрировать сторонние библиотеки и расширения для расширения функциональности веб-приложений. Благодаря активному сообществу разработчиков и обширной документации, Django позволяет создавать масштабируемые и надежные веб-приложения, поддерживаемые на протяжении долгого времени.

Flask [10] – это легкий и гибкий микрофреймворк на Python, который обеспечивает минимальный набор инструментов для быстрой разработки веб-приложений. Он предлагает простой и интуитивно понятный способ создания веб-приложений без лишнего наложения структуры. Flask позволяет разработчикам создавать веб-приложения любой сложности, начиная от небольших проектов и заканчивая большими и масштабируемыми системами. Он хорошо интегрируется с другими инструментами и библиотеками Python, что обеспечивает большую гибкость в выборе технологий. Благодаря своей простоте и модульной структуре Flask отлично подходит для создания MVP (минимально жизнеспособного продукта) и прототипов веб-приложений, а также для быстрого прототипирования и тестирования идей.

Ruby on Rails [11] – это фреймворк для разработки веб-приложений, который базируется на языке программирования Ruby. Он известен своей высокой производительностью и эффективностью разработки благодаря концепции конвенции перед конфигурацией и принципу «Don't Repeat Yourself». Фреймворк предоставляет разработчикам широкий набор готовых модулей и библиотек для быстрой разработки функциональных веб-приложений. Благодаря своей модульной архитектуре, Ruby on Rails позво-

ляет создавать масштабируемые и легко поддерживаемые приложения. Активное сообщество и обширная документация делают Ruby on Rails популярным выбором среди веб-разработчиков.

Таким образом, выбор между фреймворками зависит от конкретных потребностей проекта и уровня опыта разработчика.

Согласно рейтингу от платформы DBEngines [12], самыми востребованными реляционными СУБД на мировом рынке являются Oracle, MySQL, Microsoft SQL Server и PostgreSQL. Для российского рынка наиболее актуальными из востребованных СУБД являются MySQL [13] и PostgreSQL [14]. PostgreSQL обычно предпочтительнее для крупных и сложных приложений благодаря более расширенным возможностям SQL, более мощным механизмам управления транзакциями и блокировками, а также богатым набором типов данных. Он имеет активное сообщество разработчиков, которые постоянно расширяют его возможности и обеспечивают поддержку. MySQL может быть более подходящим для меньших проектов или тех, кто предпочитает более простое решение с меньшим набором функций.

Выводы по первой главе

В ходе анализа аналогичных проектов было обнаружено, что существует значительный интерес со стороны множества компаний к разработке систем онлайн-бронирования залов. Такие системы позволяют практически полностью автоматизировать все процессы, связанные с арендой помещений. Более того, уже были предприняты попытки локальной реализации этой концепции.

В настоящее время появляется необходимость в создании комплексной системы, которая объединила бы в себе преимущества существующих реализаций. Для этого необходимо использовать передовые инструменты разработки веб-приложений, которые были рассмотрены в процессе анализа. Такая система сможет значительно упростить и оптимизировать процессы управления залами и их бронирования, повышая эффективность бизнеса арендодателей и улучшая пользовательский опыт арендаторов.

2. ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

2.1. Определение требований к системе и вариантов ее использования

Требования к системе

По итогу проведенного анализа предметной области были выявлены следующие функциональные требования к системе.

1. Система должна позволять переключаться между площадками и между залами в рамках одной площадки.
2. Система должна отображать интерактивное расписание для каждого зала.
3. Система должна позволять арендодателям добавлять собственные площадки и редактировать ранее добавленные.
4. Система должна предоставлять пользователям доступ к оформленным броням.
5. Система должна позволять арендаторам бронировать залы как через взаимодействие с интерактивным календарем, так и с помощью формы.
6. Интерфейс веб-приложения должен включать отражение информации о залах в виде карточек с фотографиями.

Также был определен ряд нефункциональных требований.

1. Система должна включать механизм верификации арендаторов, арендодателей и добавляемых арендодателями площадок.
2. Интерфейс работы с интерактивным календарем должен предусматривать риск случайных нажатий пользователем.

Варианты использования системы

На рисунке 5 представлена диаграмма вариантов использования веб-приложения. В ходе построения диаграммы было выделено три основных актера: арендатор, арендодатель и модератор.

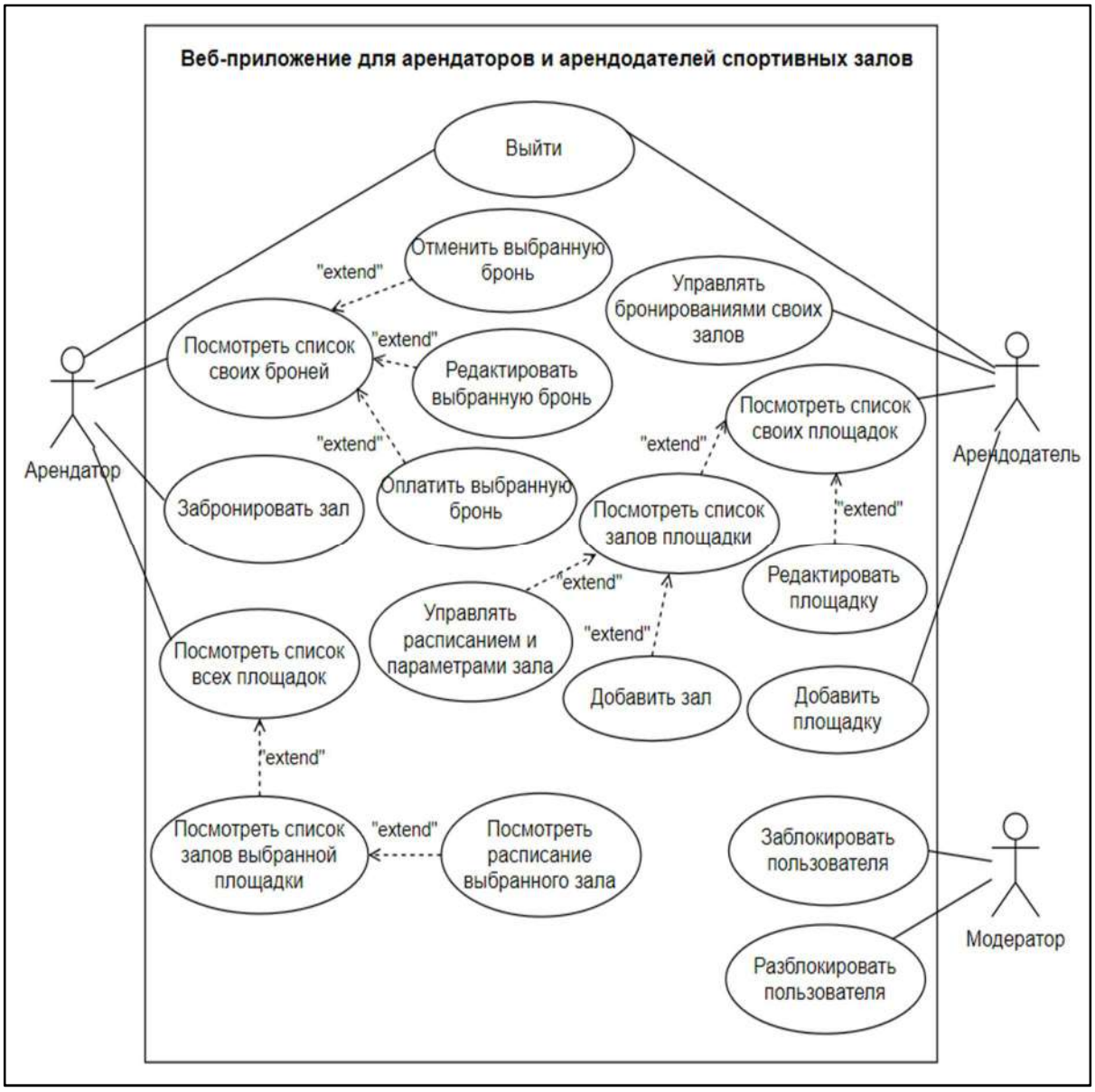

Рисунок 5 – Диаграмма вариантов использования

Арендатор может посмотреть список площадок, залов площадки и информацию о зале, забронировать зал, посмотреть список уже оформленных им бронирований и удалить, изменить или оплатить одно из них и выйти из своего аккаунта.

Арендодатель может добавить площадку или зал, при этом при добавлении новой площадки добавление зала обязательно. Также он может посмотреть список всех своих залов и оттуда перейти к расписанию конкрет-

ного зала или посмотреть список броней и арендаторов. Из расписания выбранного зала или списка броней администратор может изменить время или условия какой-либо аренды или удалить ее.

Модератор может заблокировать или разблокировать пользователя.

Подробные спецификации вариантов использования приложения описана в таблицах 1–17 приложения.

2.2. Проектирование архитектуры

На рисунке 6 представлена диаграмма компонентов разрабатываемого веб-приложения. Архитектура представляет собой структуру, состоящую из двух главных компонентов: сервера и клиента. На стороне клиента имеется компонент – веб-интерфейс, который предоставляет пользователю доступ к функциональности приложения через браузер.

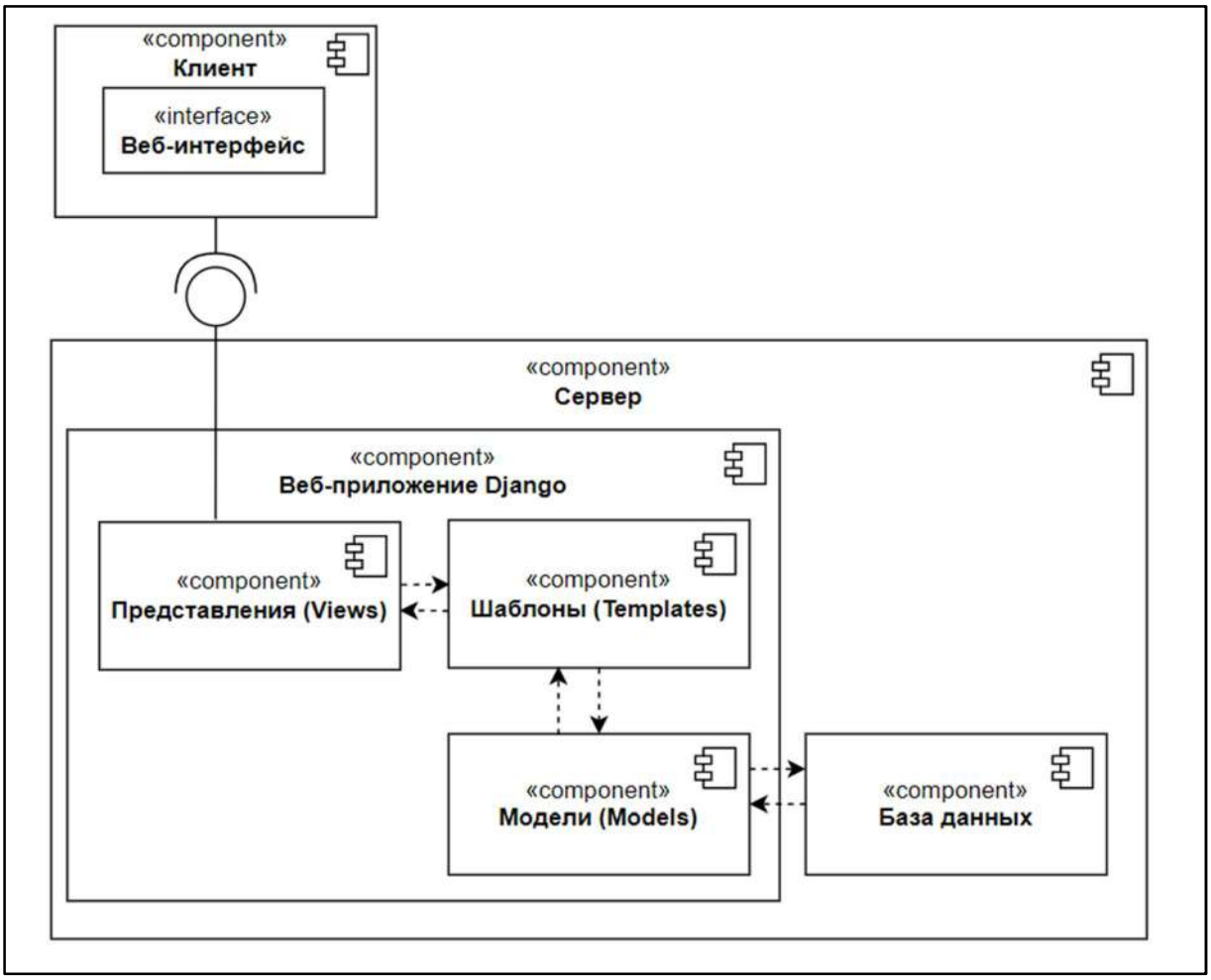

Рисунок 6 – Диаграмма компонентов системы

Сервер включает в себя два основных подкомпонента: веб-приложение Django и базу данных. Веб-приложение, в свою очередь, состоит из трех основных компонентов: компонента шаблонов, компонента представлений и компонента моделей.

Компонент шаблонов отвечает за форматирование данных и генерацию HTML-страниц для отображения пользователю. Компонент представлений управляет логикой приложения и взаимодействием с данными, он связывает компоненты сервера и клиента. Кроме того, компонент представлений взаимосвязан с компонентом шаблонов, отвечая за отображение HTML-страниц, генерируемых последним, а также с компонентом моделей.

Компонент моделей отвечает за описание структуры данных приложения в виде классов Python, которые являются моделями. Django использует модели для взаимодействия с базой данных, они определяют таблицы в базе данных и их поля, а также взаимосвязи между ними. Когда выполняются операции чтения, записи или обновления данных, компонент моделей обращается к компоненту базы данных, чтобы выполнить соответствующие SQL-запросы.

Таким образом, сервер и клиент являются основными компонентами архитектуры разрабатываемого приложения. На сервере выделяются компоненты Django веб-приложения и базы данных, тесно взаимодействующие между собой через компонент моделей. Связь компонентов шаблонов, представлений и моделей внутри веб-приложения обеспечивает целостность и эффективность приложения, а взаимосвязь компонента представлений с клиентом упрощает взаимодействие пользователя с приложением.

2.3. Проектирование базы данных

На рисунке 7 представлена схема базы данных веб-приложения. Схема базы данных состоит из нескольких таблиц, каждая из которых отвечает за определенный аспект приложения.

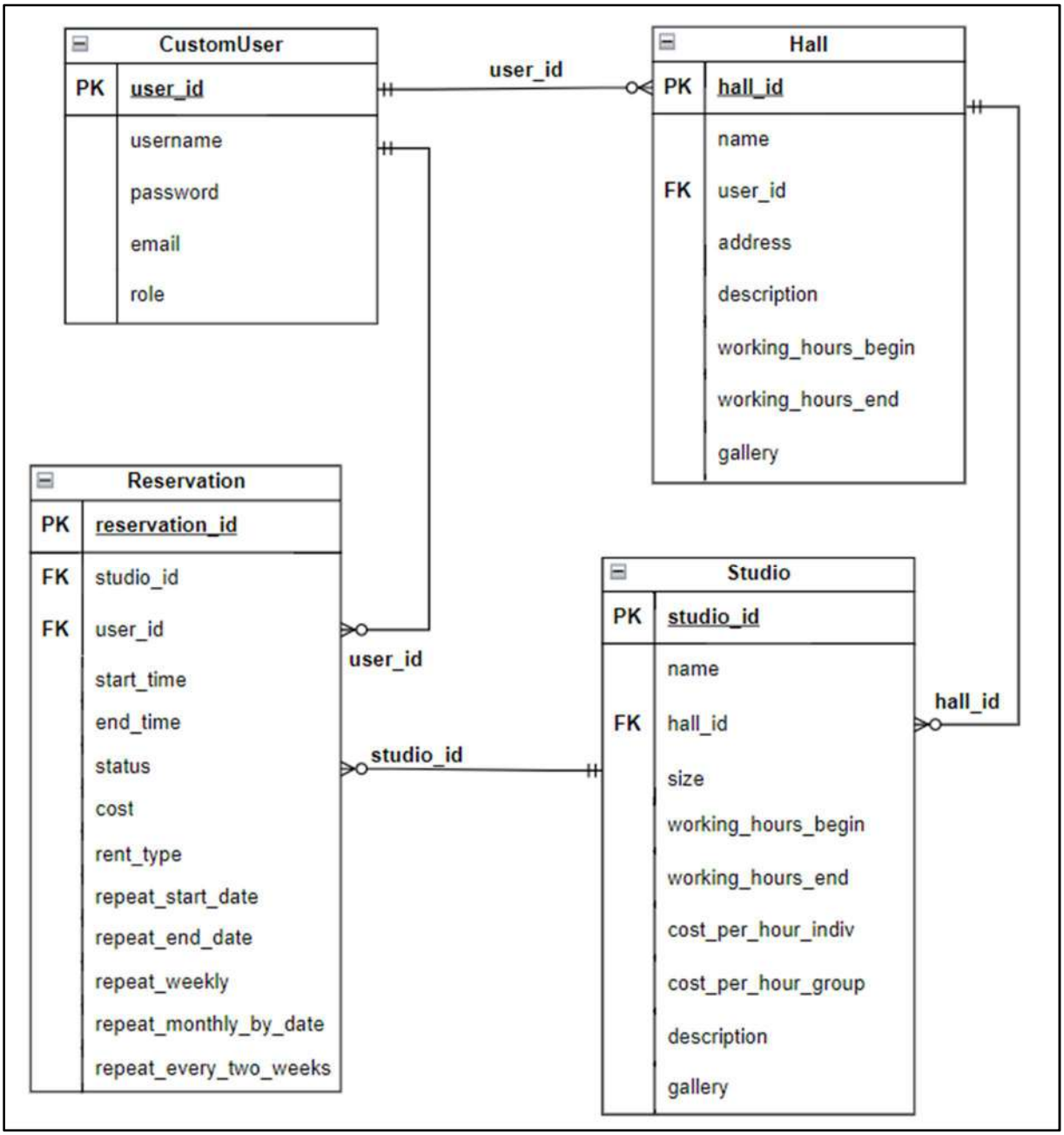

Рисунок 7 – Схема базы данных

Таблица `CustomUser` содержит информацию о пользователях. Она включает атрибуты `username`, `password`, `email`, `phone_number` для хранения имени пользователя, его пароля, адреса электронной почты и номера телефона соответственно, а также атрибут `role`, определяющий, является ли пользователь арендодателем. Первичным ключом является `user_id`, использующийся для связи `CustomUser` с таблицами `Hall` и `Reservation`.

Таблица Hall хранит информацию о доступных для аренды площадках. Ее поля name, address, description, working_hours_begin и working_hours_end описывают название, адрес, описание площадки, время начала и окончания ее работы соответственно. В поле gallery можно будет хранить изображения площадки, например, фотографии локации для облегчения поиска входа и т.п. Атрибут user_id является внешним ключом, связывающим таблицу площадок с таблицей пользователей. Первичным ключом является hall_id.

Таблица Studio содержит информацию о залах, доступных на каждой площадке. Поля name, size, description, working_hours_begin и working_hours_end хранят данные о названии, размере, описании и времени работы зала, а в поле gallery будут храниться изображения зала. Поля cost_per_hour_indiv и cost_per_hour_group. Hall_id является внешним ключом, связывающим зал с площадкой. Первичным ключом является studio_id.

Таблица Reservation отвечает за бронирования залов. Она включает внешние ключи studio_id, hall_id и user_id, связывающие каждое бронирование с залом, площадкой и арендатором. Атрибуты start_time, end_time и cost описывают время начала и окончания, а также стоимость аренды соответственно. Status может принимать значение из списка: «EP» (в ожидании оплаты), «2P» (оплачено), «C» (отмена). Rent_type принимает одно из следующих значений: «IND» (индивидуальная тренировка), «GROUP» (групповая тренировка). Поля repeat_start_date, repeat_end_date, repeat_weekly, repeat_monthly_by_date и repeat_every_two_weeks отвечают за параметры периодически повторяющихся бронирований. Первичным ключом является reservation_id.

Таким образом, представленная схема базы данных веб-приложения для управления площадками и их бронированием позволяет эффективно

хранить и управлять информацией о пользователях, площадках, залах и бронированиях. Связи между таблицами определяют, что каждая площадка принадлежит одному администратору, каждый зал принадлежит только одной площадке, а каждое бронирование связано с одной студией, одним залом и с одним пользователем-арендатором.

2.4. Проектирование интерфейса

На рисунке 8 представлен интерфейс главной страницы сайта для арендаторов. На ней будет отображаться список площадок с их названиями, адресами и соответствующими им фотографиями.

Рисунок 8 – Макет главной страницы сайта

На рисунке 9 показан пример макет страницы интерактивного расписания некоторого зала. Календарь отображается по неделям, но можно переключить отображение на дни. Слева отображается список всех залов, относящихся к этой площадке, пользователь может переключаться между ними. В календаре отображаются бронирования, причем особым цветом выделяются брони пользователя, просматривающего расписание. В верхней панели добавлена кнопка «Добавить бронь».

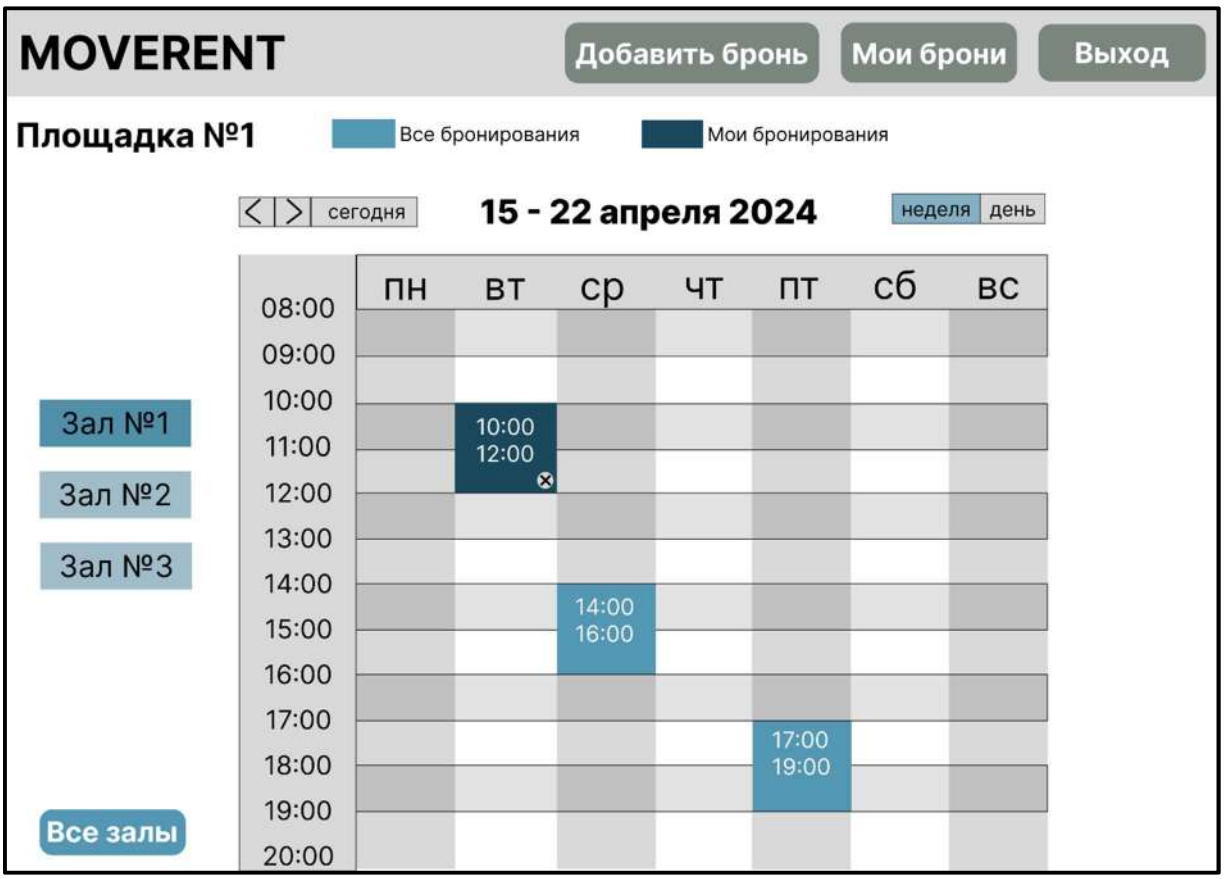

Рисунок 9 – Макет страницы расписания зала для арендатора

На рисунке 10 показан макет страницы расписания зала от лица арендодателя, управляющего площадкой, к которой относится зал. Общий интерфейс календаря идентичен интерфейсу для арендатора. Добавляется кнопка «Редактировать зал». В верхней панели кнопка «Добавить зал» позволяет перейти к форме добавления зала для открытой площадки, кнопка «Бронирования» – перейти к списку всех бронирований, относящихся к площадкам арендодателя. При нажатии на одну из броней в календаре открывается всплывающее окно с информацией о данном бронировании с возможностью редактировать его.

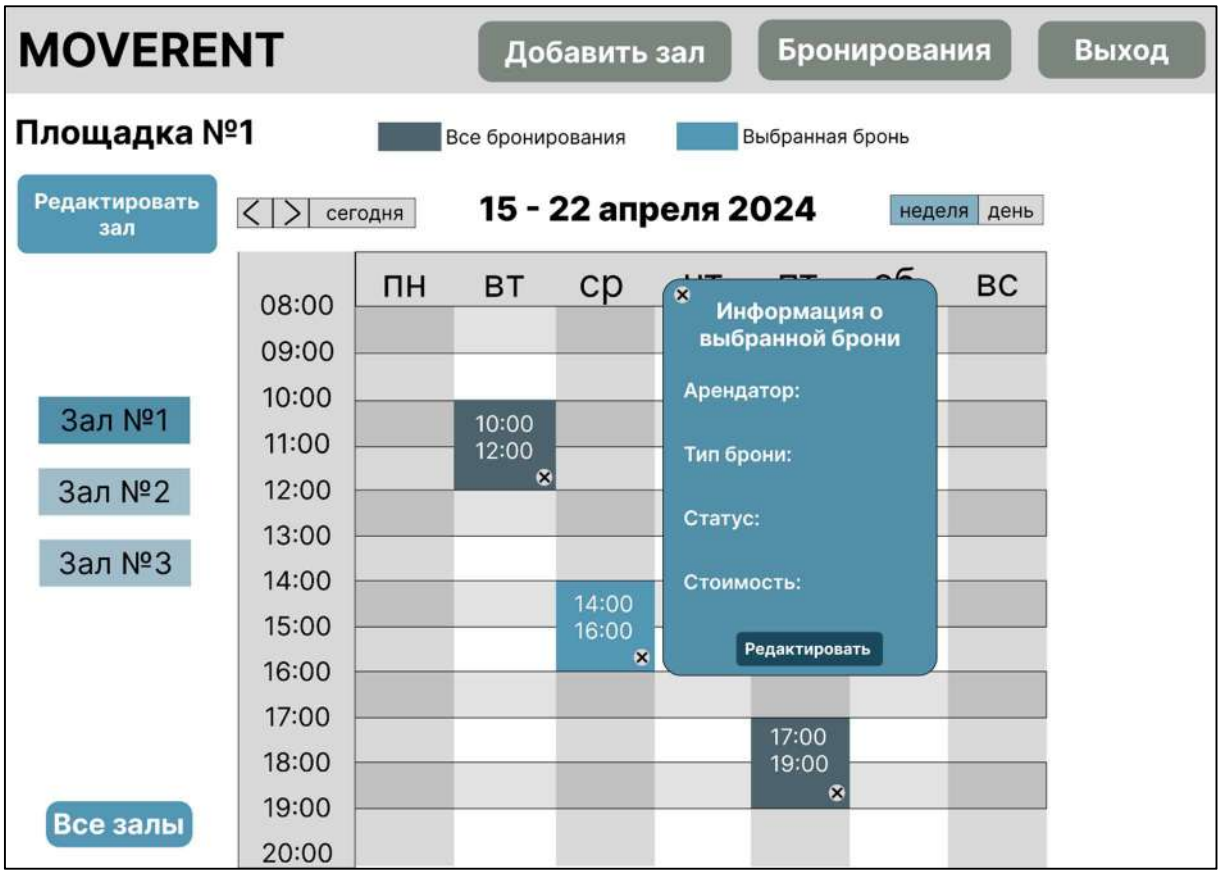

Рисунок 10 – Макет страницы расписания зала для арендодателя

Выводы по второй главе

В результате проектирования были определены функциональные и нефункциональные требования к системе и составлена диаграмма вариантов использования, включающая возможные действия для четырех актеров. С помощью диаграммы компонентов была спроектирована архитектура веб-приложения, отражающее структуру клиентской и серверной части приложения. Также была построена схема базы данных, описывающая четыре таблицы: `CustomUser`, `Hall`, `Studio` и `Reservation`. Наконец, был спроектирован интерфейс ключевых страниц веб-приложения.

3. РЕАЛИЗАЦИЯ

3.1. Программные средства

Реализация веб-приложения для арендаторов и арендодателей спортивных залов производилась с использованием следующих программных средств.

1. Python 3.12 – высокоуровневый язык программирования, используемый для разработки приложения и реализации его функциональности.
2. HTML 5, CSS 3 – языки разметки и стилей для создания пользовательского интерфейса и визуального оформления веб-страниц.
3. JavaScript [16] – язык программирования, используется для создания интерактивных элементов на веб-страницах и скриптов.
4. Django 5.1 – фреймворк для разработки веб-приложений на языке программирования Python.
5. Requests 28.2.2 [17] – библиотека для работы с HTTP-запросами в языке программирования Python, которая предоставляет удобные методы для отправки запросов на сервер и получения ответов.
6. PostgreSQL 16 – свободная объектно-реляционная система управления базами данных.
7. JetBrains PyCharm Community Edition 2022.2:1 [18] – это интегрированная среда разработки (IDE) для языка программирования Python.
8. DBeaver 24.0.0 [19] – инструмент для управления базами данных с открытым исходным кодом.
9. FullCalendar 5.10.2 [20] – это JavaScript библиотека для создания интерактивных и настраиваемых календарей на веб-страницах.
10. jQuery 3.6.0 [21] – библиотека JavaScript, которая упрощает работу с HTML-документами, манипуляции с элементами страницы, обработку событий и анимацию.
11. Moment.js 2.29.1 [22] – библиотека JavaScript для работы с датами и временем, предоставляющая удобные методы для парсинга, валидации, манипулирования и форматирования дат.

3.2. Реализация архитектуры веб-приложения в Django

При разработке на Django применяется шаблон проектирования MTV (Model-Template-View), который обеспечивает четкое разделение обязанностей, что делает код более модульным, легко поддерживаемым и масштабируемым. Этот подход разделяет приложение на три основных компонента, связанных между собой механизмами маршрутизации.

1. Модели (Model). Модели определяют структуру данных приложения, включая таблицы базы данных, их связи и логику работы с данными. В Django модели описываются в файлах `models.py` каждого приложения и могут включать поля, методы и связи с другими моделями.

2. Шаблоны (Template). Шаблоны отвечают за визуальное представление данных, полученных из представлений. Они создаются в файлах HTML с вставками переменных и логики, определенной в шаблонных тегах и фильтрах. В проекте Django шаблоны, как правило, находятся в папке `/templates` каждого приложения.

3. Представления (View). Представления обрабатывают запросы от клиента, взаимодействуют с моделями для получения или изменения данных и передают эти данные в шаблоны для отображения. В Django представления определяются в файлах `views.py` каждого приложения и могут содержать логику обработки запросов и подготовки данных для передачи в шаблоны.

Модульная структура Django проекта обеспечивается созданием отдельных приложений в рамках одного проекта. Такое стимулирует организацию кода в более структурированном и масштабируемом виде и предоставляет ряд преимуществ.

1. Каждое приложение отвечает за определенную функциональность, что делает код более организованным и понятным.

2. Каждое приложение можно использовать в других проектах.

3. Каждое приложение можно легко масштабировать независимо от других приложений проекта.

4. Каждое приложение имеет свой собственный набор моделей, представлений, шаблонов и статических файлов, что делает поиск и исправление ошибок более простым и эффективным.

В рамках реализации разрабатываемого веб-приложения было создано три приложения, использующих шаблон MTV:

1) `owners_app`, реализующее функционал стороны арендодателей;

2) `renters_app`, реализующее функционал стороны арендаторов;

3) `users_app`, реализующее механизм регистрации, авторизации и валидации пользователей.

3.3. Реализация приложения для арендаторов

Процесс реализации приложения для арендаторов включал в себя разработку двух ключевых аспектов функционала всего веб-приложения: интерактивного календаря и процесса бронирования.

3.3.1. Реализация процесса бронирования

Система бронирования реализуется с помощью модели `Reservation`, структура которой представлена в листинге 1. Помимо полей данной модели, в листинге 1 приведен код функции `set_cost`, которая используется для автоматического расчета стоимости брони по времени аренды и стоимости зала за час при сохранении бронирования в базу.

Листинг 1 – Код модели Reservation

```
class Reservation(models.Model):
    STATUSES = {
        "EP": "В ожидании оплаты",
        "P": "Оплачено",
        "C": "Отмена",
    }

    RENT_TYPE = {
        "IND": "Индивидуальная",
        "GROUP": "Групповая"
    }

    WEEK_DAYS = {
        "ПН": "Понедельник",
        "ВТ": "Вторник",
        "СР": "Среда",
```

```
        "ЧТ": "Четверг",
        "ПТ": "Пятница",
        "СБ": "Суббота",
        "ВС": "Воскресенье"
    }
    studio = models.ForeignKey(Studio, on_delete=models.CASCADE, re-
lated_name='rented_studio')
    renter = models.ForeignKey(user.MyUser, on_delete=models.CASCADE)
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()
    status = models.CharField(max_length=2, choices=STATUSES)
    cost = models.SmallIntegerField(blank=True)
    rent_type = models.CharField(max_length=6, choices=RENT_TYPE)
    repeat_start_date = models.DateField(blank=True, null=True)
    repeat_end_date = models.DateField(blank=True, null=True)
    repeat_weekly = models.CharField(max_length=2, choices=WEEK_DAYS,
blank=True, null=True)
    repeat_monthly_by_date = models.SmallIntegerField(blank=True,
null=True)
    repeat_every_two_weeks = models.CharField(max_length=2,
choices=WEEK_DAYS, blank=True, null=True)


@receiver(pre_save, sender=Reservation)
def set_cost(sender, instance, *args, **kwargs):
    time_difference = instance.end_time - instance.start_time
    time_difference_hours = time_difference.total_seconds() / 3600
    if instance.rent_type == "IND":
        instance.cost = time_difference_hours * instance.stu-
dio.cost_per_hour_indiv
    else:
        instance.cost = time_difference_hours * instance.stu-
dio.cost_per_hour_group
```

Данная модели затем используется в форме для добавления бронирования `ReservationForm`, код которой приведен в листинге 2.

Листинг 2 – Код формы для добавления брони

```
class ReservationForm(forms.ModelForm):
    repeat_booking = forms.BooleanField(label='Повторять бронь', re-
quired=False)

    class Meta:
        model = Reservation
        fields = ['start_time', 'end_time', 'rent_type', 're-
peat_start_date', 'repeat_end_date', 'repeat_weekly', 're-
peat_monthly_by_date',
                  'repeat_every_two_weeks']
        widgets = {
            'start_time': forms.DateTimeInput(attrs={'type': 'datetime-lo-
cal'}),
            'end_time': forms.DateTimeInput(attrs={'type': 'datetime-lo-
cal'}),
            'repeat_start_date': forms.DateInput(attrs={'type': 'date'}),
            'repeat_end_date': forms.DateInput(attrs={'type': 'date'}),

        }
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
```

```
        self.fields['repeat_start_date'].required = False
        self.fields['repeat_end_date'].required = False
        self.fields['repeat_weekly'].required = False
        self.fields['repeat_monthly_by_date'].required = False
        self.fields['repeat_every_two_weeks'].required = False
        for field_name in ['repeat_start_date', 'repeat_end_date', 're-
peat_weekly', 'repeat_monthly_by_date',
                           'repeat_every_two_weeks']:
            self.fields[field_name].widget.attrs['class'] = 'repeat-fields'
```

`ReservationForm` наследуется от класса `ModelForm`, предоставляемого фреймворком, и определяет следующие дополнительные поля и метаданные.

1. Поле `repeat_booking` – дополнительное поле логического типа, которое позволяет пользователю указать, будет ли бронирование регулярным.

2. `Meta` – внутренний класс, который связывает форму с моделью `Reservation` и указывает, какие поля формы будут отображаться. Также в нем определяются виджеты для полей `start_time`, `end_time`, `repeat_start_date` и `repeat_end_date`.

Метод `__init__` переопределяется для настройки параметров полей после их инициализации. Все поля, отвечающие за настройки повторяющегося бронирования, по умолчанию отмечаются как необязательные.

Метод `clean`, код которого представлен в листинге 3, переопределяется для выполнения пользовательской валидации данных формы. Выполняется проверка времени: время окончания брони должно быть больше времени начала. Также выполняется проверка заполнения условий повторяющегося бронирования: если `repeat_booking` установлено в `True`, проверяется, что одно из полей `repeat_weekly`, `repeat_monthly_by_date` или `repeat_every_two_weeks` заполнено. При этом может быть заполнено не более одного поля, отвечающего за настройки повторения.

Листинг 3 – Код метода `clean` формы добавления бронирования

```
def clean(self):
        cleaned_data = super().clean()
        start_time = cleaned_data.get('start_time')
        end_time = cleaned_data.get('end_time')
        if start_time and end_time:
```

```
            if end_time <= start_time:
                raise forms.ValidationError('Конечное время должно быть
позже начального времени.')
        repeat_booking = cleaned_data.get('repeat_booking')
        repeat_start_date = cleaned_data.get('repeat_start_date')
        repeat_end_date = cleaned_data.get('repeat_end_date')
        repeat_weekly = cleaned_data.get('repeat_weekly')
        repeat_monthly_by_date = cleaned_data.get('repeat_monthly_by_date')
        repeat_every_two_weeks = cleaned_data.get('repeat_every_two_weeks')

        if repeat_booking:
            # Проверка наличия заполненного поля повторения
            if not any([repeat_weekly, repeat_monthly_by_date, re-
peat_every_two_weeks]):
                raise forms.ValidationError('Выберите хотя бы одно поле для
повторения брони.')
            # Проверка на заполнение только одного поля повторения
            repeat_fields = [repeat_weekly, repeat_monthly_by_date, re-
peat_every_two_weeks]
            filled_count = sum(1 for field in repeat_fields if field)
            if filled_count != 1:
                raise forms.ValidationError('Выберите только одно поле для
повторения брони.')

        return cleaned_data
```

При отображении формы `ReservationForm` на HTML-странице поля, относящиеся к repeat-fields, изначально скрыты от пользователя и появляются после отметки чекбокса «Повторять бронь».

Если пользователь корректно заполнил все поля формы, то новый объект бронирования сохраняется в базу с помощью следующего кода, представленного в листинге 4.

Листинг 4 – Обработка полей формы бронирования перед сохранением

```
if request.method == 'POST':
        form = ReservationForm(request.POST, request.FILES)
        if form.is_valid():
            new_reservation = form.save(commit=False)
            studio = studio
            new_reservation.studio = studio
            new_reservation.renter = request.user
            new_reservation.status = "EP"
            new_reservation.save()
            return redirect('studio', id)
    else:
        form = ReservationForm()
```

Поля, связывающие бронирование с залом и арендатором, заполняются автоматически на основе имеющихся данных. Полю статуса по умолчанию присваивается значение «Ожидается оплата».

3.3.2. Реализация функционала интерактивного календаря

Интерактивный календарь был реализован с помощью библиотеки FullCalendar для JavaScript. В сочетании с библиотеками jQuery и Moment.js она позволяет динамически генерировать календарь, загружать события и взаимодействовать с пользователем. В листинге 5 приведена инициализация событий в календаре.

Листинг 5 – Инициализация событий календаря

```
var events = [
    {% for reservation in all_reservations %} {
     title: 'Забронировано',
     start: '{{ reservation.start_time|date:"Y-m-d\\TH:i" }}',
     end: '{{ reservation.end_time|date:"Y-m-d\\TH:i" }}',
     className: 'all-reservation'
    },
    {% endfor %}
    {% for reservation in user_reservations %} {
     title: 'Ваша бронь',
     start: '{{ reservation.start_time|date:"Y-m-d\\TH:i" }}',
     end: '{{ reservation.end_time|date:"Y-m-d\\TH:i" }}',
     className: 'user-reservation'
    },
    {% endfor %}
];
```

События делятся на `all_reservations` – бронирования всех пользователей, за исключением текущего – и `user_reservations` – бронирования пользователя, просматривающего страницу. Данные передаются в шаблон на уровне контроллера уже разделенные по категориям.

Код инициализации самого календаря приведен в листинге 6.

Листинг 6 – Инициализация интерактивного календаря

```
var calendar = new FullCalendar.Calendar(calendarEl, {
    initialView: 'timeGridWeek',
    headerToolbar: {
        left: 'prev,next today',
        center: 'title',
        right: 'timeGridWeek,timeGridDay'
    },
    events: events,
    allDaySlot: false,
    buttonText: {
        today: 'Сегодня',
        week: 'Неделя',
        day: 'День'
    },
    locale: 'ru',
    firstDay: 1,
    slotMinTime: studioWorkingHoursBegin,
```

```
    slotMaxTime: studioWorkingHoursEnd,
    slotLabelFormat: {
        hour: '2-digit',
        minute: '2-digit',
        hour12: false,
    },
    eventTimeFormat: {
        hour: '2-digit',
        minute: '2-digit',
        hour12: false // 24-hour format
    },
    expandRows: true,
    height: 'auto',
    dateClick: function(info) {
        var start = info.dateStr;
        var end = moment(info.date).add(1,'hour').format('YYYY-MM-
DDTHH:mm:ss');
        window.location.href = "{% url 'rent' studio_id %}?start=" + start
+ "&end=" + end; },
    select: function(info) {
        var start = info.startStr;
        var end = info.endStr;
        window.location.href = "{% url 'rent' studio_id %}?start=" + start
+ "&end=" + end;
    },
    selectable: true,
});
```

Календарь настраивается с использованием различных параметров, включая плагины, начальный вид, локализацию и обработчики событий.

1. С помощью параметра `plugins` подключаются плагины `timeGrid` и `interaction`, необходимые для отображения календаря в виде сетки и обработки взаимодействия с пользователем.

2. Параметр `initialView` устанавливает начальный вид календаря – `timeGridWeek`.

3. Параметр `headerToolbar` определяет элементы управления в заголовке календаря, такие как кнопки навигации и переключения вида.

4. В качестве параметра `events` передаются события, сформированные ранее.

5. Параметры `slotMinTime` и `slotMaxTime` устанавливают границы отображаемого времени в календаре на основе рабочего времени студии.

6. Параметр `allDaySlot` управляет отображением слота для целого дня.

7. С помощью параметров `locale` и `buttonText` настраиваются локализация и текст кнопок для соответствия русскому языку.

8. Параметры `dateClick` и `select` – обработчики событий для клика по дате и выбора диапазона времени. Эти обработчики перенаправляют пользователя на страницу добавления бронирования с предзаполненными параметрами времени при нажатии на свободную ячейку календаря.

3.4. Реализация интерфейса

Основная часть интерфейса была реализована путем создания шаблонов HTML и применения стилей CSS. Исключением является интерактивный календарь: в его случае код HTML был сгенерирован автоматически благодаря библиотеке FullCalendar для JavaScript.

На рисунках 11–13 показан интерфейс страниц, которые последовательно проходит арендатор для бронирования зала. При входе на сайт пользователь видит список всех площадок, зарегистрированных в базе.

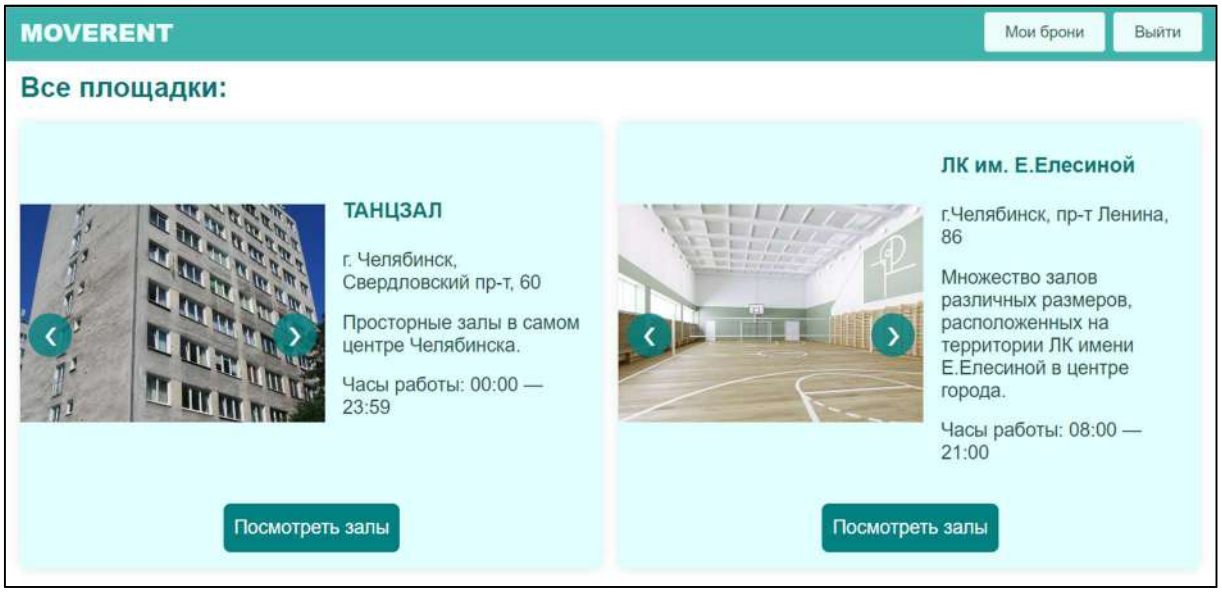

Рисунок 11 – Главная страница сайта со всеми площадками

Нажав на кнопку «Посмотреть залы» для какой-либо площадки, можно перейти к списку залов, относящихся к ней.

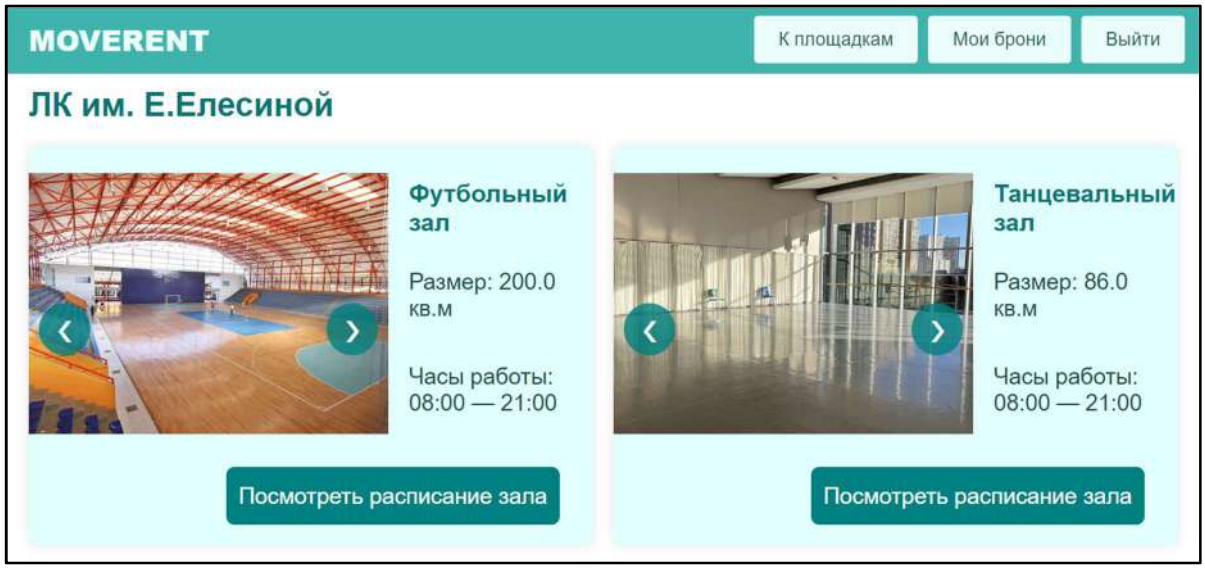

Рисунок 12 – Страница списка залов выбранной площадки

Нажав на «Посмотреть расписание зала» для одного из предложенных залов, пользователь перейдет к странице расписания выбранного зала. Здесь он может посмотреть занятость выбранного зала, переключаться между залами площадки и перейти к форме добавления брони. Интерактивный календарь предоставляет возможность переключаться между неделями или днями в зависимости от выбранного представления, а также мгновенно переключаться на текущую дату нажатием на кнопку «Сегодня».

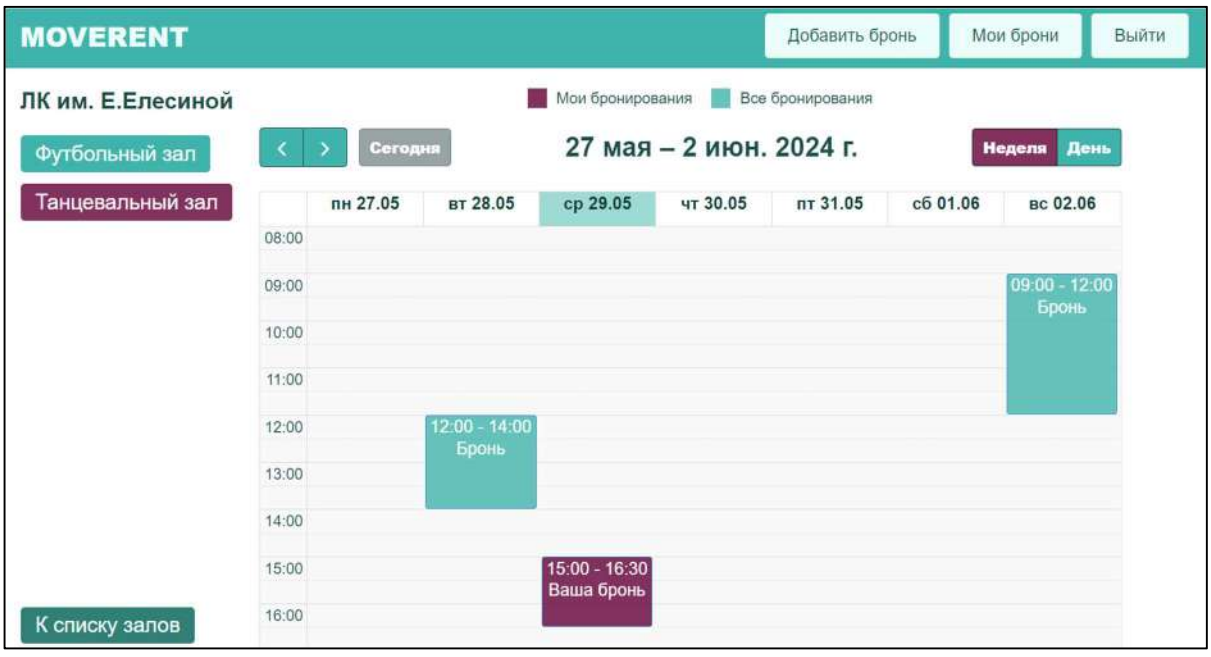

Рисунок 13 – Страница расписания выбранного зала для арендатора

Для арендатора информация о бронированиях других пользователей недоступна и ограничивается временным промежутком. Арендодатели же могут получить полную информацию о брони с возможностью ее редактирования или удаления, нажав на нее в календаре.

На рисунке 14 показан полный интерфейс формы для бронирования выбранного зала, включая поля, относящиеся к повторяющимся броням.

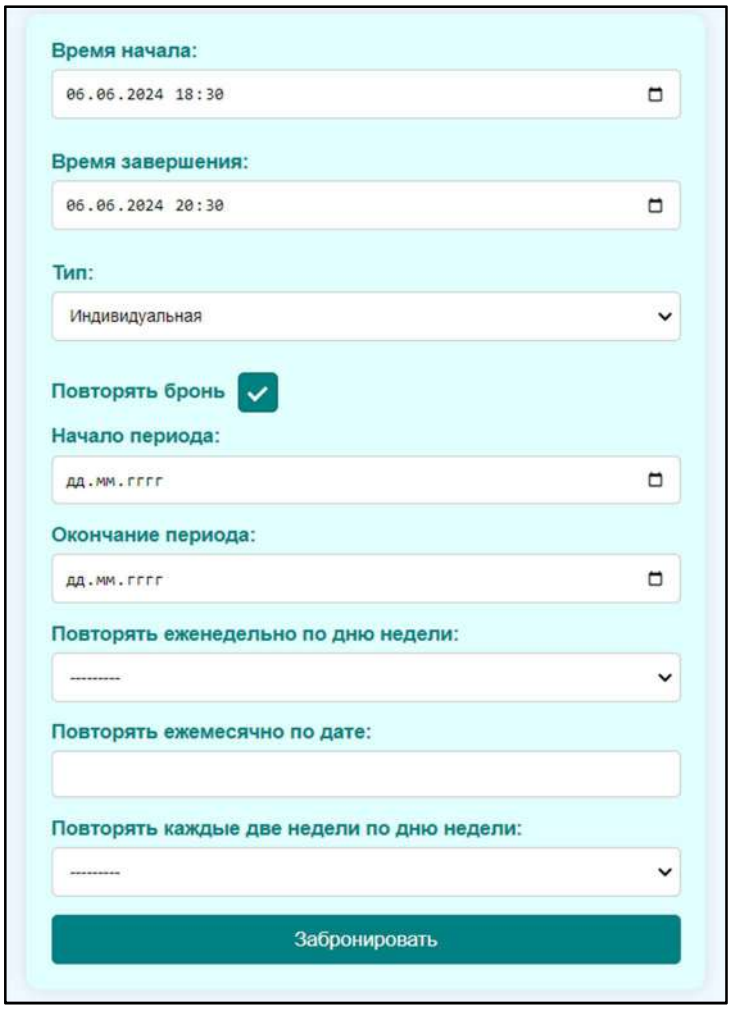

Рисунок 14 – Форма для бронирования зала

Выводы по третьей главе

Использованный при реализации приложения фреймворк Django предоставляет мощные инструменты для работы с моделями, формами и аутентификацией, что было необходимо для создания функциональных компонентов приложения, таких как система бронирования залов, управления галереями изображений, добавления площадок и залов и структурирования их системы.

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование

Функциональное тестирование [23] – это процесс проверки программного продукта на соответствие его функциональным требованиям. Оно включает в себя тестирование функций, взаимодействия с пользователем, обработки данных и других аспектов, чтобы убедиться, что программа работает правильно и выполняет запланированные задачи. Результаты проведения функционального тестирования представлены в таблице 2.

Таблица 2 – Результаты функционального тестирования

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1.	Регистрация арендодателя	1. На главной странице нажать «Зарегистрироваться» 2. Заполнить форму регистрации с выбором «Зарегистрироваться как арендодатель» 3. Нажать «Зарегистрироваться»	Арендодатель видит главную страницу сайта, где пока нет добавленных им площадок.	Да
2.	Регистрация арендатора	1. На главной странице нажать «Зарегистрироваться» 2. Заполнить форму регистрации без выбора «Зарегистрироваться как арендодатель» 3. Нажать «Зарегистрироваться»	Арендатор видит главную страницу сайта со всеми площадками, добавленными в базу.	Да
3.	Вход арендодателя на сайт	1. На главной странице нажать «Войти» 2. Заполнить форму авторизации 3. Нажать «Войти»	Арендодатель видит главную страницу сайта со всеми своими площадками.	Да
4.	Вход арендатора на сайт	1. На главной странице нажать «Войти» 2. Заполнить форму авторизации 3. Нажать «Войти»	Арендатор видит главную страницу сайта со всеми площадками, добавленными в базу.	Да
4.	Добавление площадки арендодателем	1. Войти как арендодатель 2. На главной странице нажать «Добавить площадку» 3. Заполнить форму добавления площадки 4. Нажать «Добавить»	Арендодатель видит главную страницу сайта со всеми его площадками, включая только что добавленную.	Да

Продолжение таблицы 2

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
5.	Добавление зала арендодателем	1. Войти как арендодатель 2. На главной странице нажать «Посмотреть залы» для одной из площадок 3. Нажать «Добавить зал» 4. Заполнить форму добавления зала 5. Нажать «Добавить»	Арендодатель видит страницу со списком залов площадки, список включает только что добавленный зал.	Да
6.	Просмотр арендодателем бронирований его залов	1. Войти как арендодатель 2. На главной странице нажать «Бронирования»	Арендодатель видит страницу со списком всех броней, относящихся к его площадкам.	Да
7.	Просмотр всех своих бронирований арендатором	1. Войти как арендатор 2. На главной странице нажать «Мои брони»	Арендатор видит страницу со списком всех своих бронирований.	Да
8.	Просмотр расписания зала арендатором	1. Войти как арендатор 2. На главной странице нажать «Посмотреть залы» для одной из площадок 3. На странице площадки нажать «Посмотреть расписание зала» для одного из залов	Арендатор видит страницу с интерактивным календарем, отражающим занятость выбранного зала, при этом его брони выделяются темно-малиновым цветом.	Да
9.	Добавления брони арендатором	1. Войти как арендатор 2. На главной странице нажать «Посмотреть залы» для одной из площадок 3. На странице площадки нажать «Посмотреть расписание зала» для одного из залов 4. Нажать «Добавить бронь» 5. Заполнить форму добавления брони 6. Нажать «Забронировать»	Арендатор видит страницу с расписанием зала, добавленная им бронь отображается и выделяется темно-малиновым цветом.	Да
10.	Взаимодействие арендатора со свободной ячейкой календаря	1. Войти как арендатор 2. На главной странице нажать «Посмотреть залы» для одной из площадок 3. На странице площадки нажать «Посмотреть расписание зала» для одного из залов 4. Нажать на свободную ячейку в календаре	Арендатор будет перенаправлен на страницу с формой бронирования с предварительно заполненными полями времени начала и окончания брони.	Да

Окончание таблицы 2

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
11.	Взаимодействие арендодателя с занятой ячейкой календаря	1. Войти как арендодатель 2. На главной странице нажать «Посмотреть залы» для одной из площадок 3. На странице площадки нажать «Посмотреть расписание зала» для одного из залов 4. Нажать на ячейку с бронированием	На экране арендодателя отобразится всплывающее окно с подробной информацией о бронировании, на которое он нажал.	Да
12.	Взаимодействие незарегистрированного пользователя со свободной ячейкой календаря-	1. На главной странице нажать «Посмотреть залы» для одной из площадок 2. На странице площадки нажать «Посмотреть расписание зала» для одного из залов 3. Нажать на свободную ячейку в календаре	На экране пользователя отобразится всплывающее окно с сообщением: «Для бронирования залов необходимо авторизоваться».	Да
13.	Удаление арендатором бронирования	1. Войти как арендатор 2. На главной странице нажать «Мои брони» 3. Рядом с нужной бронью в отразившемся списке нажать «Удалить». 4. Во всплывающем окне нажать «ОК».	На экране арендатора отобразится сообщение об успешном удалении бронирования.	Да

Выводы по четвертой главе

В ходе проведения тестирования были проверены основные функции веб-приложения. Были проведены наборы тестов для каждого функционального требования веб-приложения. Ошибок, влияющих на работоспособность системы, не выявлено.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано веб-приложение для арендаторов и арендодателей спортивных залов. При этом были решены следующие задачи.

1. Произведен обзор литературы и существующих аналогичных open-source проектов по предметной области.

2. Разработано техническое задание на создание веб-приложения.

3. Спроектировано веб-приложение. В процессе проектирования были определены требования к разрабатываемой системе, построена ее архитектура, включая структуру базы данных, разработаны алгоритмы и интерфейс.

4. Выполнена реализация веб-приложения с использованием языков программирования Python, JavaScript, HTML, CSS, а также фреймворка Django и системы управления СУБД PostgreSQL.

5. Проведено функциональное тестирование веб-приложения.

В дальнейшем планируется расширение функционала веб-приложения. Приоритетными являются следующие задачи:

1) добавление поисковых фильтров для площадок и залов;

2) адаптация интерфейса под мобильные устройства;

3) добавление кастомизации правил редактирования и удаления бронирований для каждой площадки;

4) интеграция внешних сервисов для оплаты бронирований;

5) добавление возможности регистрироваться на платформе через социальные сети;

6) реализация возможности прямой коммуникации арендаторов и арендодателей внутри приложения.

Исходный код веб-приложения доступен по адресу https://github.com/Lazarevasn/web-studio-renting.

ЛИТЕРАТУРА

1. Аналитический центр НАФИ: Как изменились образ жизни и здоровье россиян в 2023 году. [Электронный ресурс] URL: https://nafi.ru/analytics/kak-izmenilis-obraz-zhizni-i-zdorove-rossiyan-v-2023-godu/ (дата обращения: 04.04.2024 г.).

2. Авито: недвижимость, транспорт, работа, услуги, вещи. [Электронный ресурс] URL: https://avito.ru (дата обращения: 10.02.2024 г.).

3. Юла – доска объявлений. [Электронный ресурс] URL: https://youla.ru (дата обращения: 10.02.2024 г.).

4. Zalti. [Электронный ресурс] URL: https://zalti.ru (дата обращения: 10.02.2024 г.).

5. Студия танцев в Москве «9 ЗАЛОВ». [Электронный ресурс] URL: https://9zalov.ru (дата обращения: 10.02.2024 г.).

6. Платформа MUSbooking. [Электронный ресурс] URL: https://musbooking.com (дата обращения: 10.02.2024 г.).

7. HTML: HyperText Markup Language. [Электронный ресурс] URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата обращения: 10.02.2024 г.).

8. CSS: Cascading Style Sheets. [Электронный ресурс] URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата обращения: 10.02.2024 г.).

9. Python. [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 10.02.2024 г.).

10. Django: The web framework for perfectionists with deadlines. [Электронный ресурс] URL: https://www.djangoproject.com/ (дата обращения: 11.02.2024 г.).

11. Welcome to Flask – Flask Documentation (2.2.x). [Электронный ресурс] URL: https://flask.palletsprojects.com/en/2.2.x/ (дата обращения: 11.02.2024 г.).

12. Ruby on Rails. [Электронный ресурс] URL: https://rubyonrails.org/ (дата обращения: 11.02.2024 г.).

13. DB-Engines. Knowledge Base of Relational and NoSQL Database Management Systems. [Электронный ресурс] URL: https://db-engines.com/en/ranking (дата обращения: 04.04.2024 г.).

14. MySQL: The world’s most popular open source database. [Электронный ресурс] URL: https://www.mysql.com/ (дата обращения: 11.02.2024 г.).

15. PostgreSQL: The world's most advanced open source database. [Электронный ресурс] URL: https://www.postgresql.org/ (дата обращения: 11.02.2024 г.).

16. Флэнаган Д. JavaScript: карманный справочник, 3-е изд. // Издательство Вильямс, 2013 г. – С. 320.

17. Requests. [Электронный ресурс] URL: https://pypi.org/project/requests/ (дата обращения: 02.04.2024 г.).

18. JetBrains PyCharm IDE. [Электронный ресурс] URL: https://www.jetbrains.com/pycharm/ (дата обращения: 02.04.2024 г.).

19. DBeaver: Free Universal Database Tool. [Электронный ресурс] URL: https://dbeaver.io/ (дата обращения: 02.04.2024 г.).

20. FullCalendar. [Электронный ресурс] URL: https://fullcalendar.io/ (дата обращения: 10.04.2024 г.).

21. jQuery. The Write Less, Do More, JavaScript Library. [Электронный ресурс] URL: https://jquery.com/ (дата обращения: 10.04.2024 г.).

22. Moment.js. [Электронный ресурс] URL: https://momentjs.com/ (дата обращения: 10.04.2024 г.).

23. Юрченко А.Н. «Функциональное тестирование как одна из методологий тестирования программного обеспечения». Вестник современных исследований. [Электронный ресурс] URL: https://www.elibrary.ru/item.asp?id=32535053 (дата обращения: 20.05.2024 г.).

ПРИЛОЖЕНИЕ. Спецификация вариантов использования

Таблица 1 – Спецификация прецедента «Посмотреть список всех площадок»

Прецедент: Посмотреть список всех площадок
ID: 1
Аннотация: Просмотр списка всех площадок платформы
Главные актеры: Неавторизованный пользователь или арендатор
Второстепенные актеры: Нет
Предусловия: Нет
Основной поток: 1. Вариант использования начинается, когда пользователь открывает главную страницу сайта. 2. Система отображает список всех имеющихся в базе данных площадок.
Постусловия: Пользователь видит список всех доступных площадок.
Альтернативные потоки: Нет

Таблица 2 – Спецификация прецедента «Посмотреть список залов выбранной площадки»

Прецедент: Посмотреть список залов выбранной площадки
ID: 2
Аннотация: Просмотр списка залов выбранной площадки
Главные актеры: Неавторизованный пользователь или арендатор
Второстепенные актеры: Нет
Предусловия: Пользователь находится на странице со списком площадок
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Посмотреть список залов» для одной из площадок. 2. Система отображает список залов, относящихся к данной площадке.
Постусловия: Пользователь видит список залов, относящихся к выбранной площадке.
Альтернативные потоки: Нет

Таблица 3 – Спецификация прецедента «Посмотреть расписание выбранного зала»

Прецедент: Посмотреть расписание выбранного зала
ID: 3
Аннотация: Просмотр расписания выбранного зала
Главные актеры: Неавторизованный пользователь или арендатор
Второстепенные актеры: Нет
Предусловия: Пользователь находится на странице со списком залов площадки
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Посмотреть расписание зала» для одного из залов 2. Система отображает страницу с расписанием зала в виде интерактивного календаря.
Постусловия: Пользователь видит расписание интересующего его зала.
Альтернативные потоки: Нет

Таблица 4 – Спецификация прецедента «Забронировать зал»

Прецедент: Забронировать зал
ID: 4
Аннотация: Бронирование выбранного зала
Главные актеры: Арендатор
Второстепенные актеры: Нет
Предусловия: Пользователь находится на странице с расписанием какого-либо зала.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Добавить бронь» или на свободную ячейку в календаре. 2. Система отображает страницу с формой добавления бронирования. 3. Если пользователь нажимает галочку на «Повторять бронь», то система отображает дополнительные поля для повторяющихся броней. 4. Пользователь заполняет поля формы. 5. Пользователь нажимает «Забронировать».
Постусловия: Новое бронирование успешно добавлено в базу данных.
Альтернативные потоки: Пользователь не заполнил поле со временем начала или окончания бронирования.

Таблица 5 – Спецификация прецедента «Посмотреть список оформленных броней»

Прецедент: Посмотреть список оформленных броней
ID: 5
Аннотация: Просмотр списка ранее оформленных бронирований
Главные актеры: Арендатор
Второстепенные актеры: Нет
Предусловия: Нет
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Мои брони». 2. Система отображает страницу со списком всех бронирований, оформленных пользователем.
Постусловия: Пользователь видит список всех своих бронирований, если они есть.
Альтернативные потоки: Нет

Таблица 6 – Спецификация прецедента «Удалить выбранную бронь»

Прецедент: Удалить выбранную бронь
ID: 6
Аннотация: Удаление ранее оформленного бронирования
Главные актеры: Арендатор
Второстепенные актеры: Нет
Предусловия: Арендатор просматривает страницу со списком всех своих бронирований.

Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Удалить» возле одной из своих броней. 2. Система проверяет, осталось ли до времени аренды более 24 часов. 3. Если осталось больше 24 часов 3.1. Система отображает всплывающее окно, требующее повторное подтверждение удаления. 3.2 Если пользователь нажимает «ОК» 3.2.1. Система удаляет соответствующую бронь из базы данных. 3.3. Если пользователь нажимает «Отмена», переход к шагу 5. 4. Если осталось менее 24 часов, система отображает уведомление об ошибке. 5. Система отображает список броней пользователя.
Постусловия: Пользователь успешно удалил бронь.
Альтернативные потоки: 1. Пользователь нажал «Отмена», бронь не была удалена. 2. До бронирования осталось менее 24 часов, удаление недоступно.

Таблица 7 – Спецификация прецедента «Редактировать выбранную бронь»

Прецедент: Редактировать выбранную бронь
ID: 7
Аннотация: Редактирование ранее оформленного бронирования
Главные актеры: Арендатор
Второстепенные актеры: Нет
Предусловия: Арендатор просматривает страницу со списком всех своих бронирований.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Редактировать» возле одной из своих броней. 2. Система проверяет, осталось ли более 24 часов до начала времени аренды. 3. Если осталось более 24 часов 3.1. Система отображает форму для редактирования брони. 3.2. Пользователь меняет необходимые ему параметры. 3.3. Если пользователь нажимает «Сохранить изменения», система вносит изменения в соответствующую бронь в базе данных. 3.4. Если пользователь нажимает «Отмена», переход к шагу 5. 4. Если осталось менее 24 часов, система отображает ошибку. 5. Система отображает список броней пользователя.
Постусловия: Изменения бронирования успешно сохранены в базу данных.
Альтернативные потоки: 1. Пользователь нажал «Отмена», изменения не были сохранены. 2. До бронирования осталось менее 24 часов, редактирование не доступно.

Таблица 8 – Спецификация прецедента «Выйти»

Прецедент: Выйти
ID: 8
Аннотация: Выход пользователя из системы
Главные актеры: Арендатор или арендодатель

Второстепенные актеры: Нет
Предусловия: Пользователь был авторизован.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Выйти». 2. Система заканчивает текущую сессию пользователя. 3. Система отображает главную страницу сайта.
Постусловия: Пользователь успешно вышел из системы.
Альтернативные потоки: Нет

Таблица 9 – Спецификация прецедента «Заблокировать пользователя»

Прецедент: Заблокировать пользователя
ID: 9
Аннотация: Блокировка пользователя
Главные актеры: Модератор
Второстепенные актеры: Нет
Предусловия: Модератор вошел в систему как суперпользователь.
Основной поток: 1. Вариант использования начинается, когда модератор открывает список пользователей. 2. Модератор открывает информацию о конкретном пользователе. 3. Модератор снимает галочку с поля «Active». 3. Если модератор нажимает «Save» 3.1. Система сохраняет изменения. 3.2. Система отображает список пользователей. 4. Если модератор нажимает «Save and continue editing» 4.1. Система сохраняет изменения.
Постусловия: Модератор успешно заблокировал пользователя.
Альтернативные потоки: Модератор покинул страницу пользователя, не сохранив изменения.

Таблица 10 – Спецификация прецедента «Посмотреть список своих площадок»

Прецедент: Посмотреть список своих площадок
ID: 10
Аннотация: Просмотр арендодателем площадок, которыми он может управлять
Главные актеры: Арендодатель
Второстепенные актеры: Нет
Предусловия: Пользователь вошел в систему как арендодатель
Основной поток: 1. Вариант использования начинается, когда арендодатель открывает главную страницу. 2. Система находит в базе данных площадки, относящиеся к данному арендодателю. 3. Система отображает список найденных площадок.
Постусловия: Арендодатель видит список всех своих площадок.
Альтернативные потоки: Нет

Таблица 11 – Спецификация прецедента «Разблокировать пользователя».

Прецедент: Разблокировать пользователя
ID: 11
Аннотация: Разблокировка пользователя
Главные актеры: Модератор
Второстепенные актеры: Нет
Предусловия: Модератор вошел в систему как суперпользователь.
Основной поток: 1. Вариант использования начинается, когда модератор открывает список пользователей. 2. Модератор открывает информацию о конкретном пользователе, который ранее был заблокирован. 3. Модератор ставит галочку на поле «Active». 3. Если модератор нажимает «Save» 3.1. Система сохраняет изменения. 3.2. Система отображает список пользователей. 4. Если модератор нажимает «Save and continue editing» 4.1. Система сохраняет изменения.
Постусловия: Модератор успешно разблокировал пользователя.
Альтернативные потоки: Модератор покинул страницу пользователя, не сохранив изменения.

Таблица 12 – Спецификация прецедента «Добавить площадку»

Прецедент: Добавить площадку
ID: 12
Аннотация: Добавление площадки арендодателем
Главные актеры: Арендодатель
Второстепенные актеры: Нет
Предусловия: Пользователь вошел в систему как арендодатель и открыл главную страницу.
Основной поток: 1. Вариант использования начинается, когда арендодатель нажимает «Добавить площадку». 2. Система отображает форму для добавления площадки. 3. Пользователь заполняет форму. 4. Пользователь нажимает «Добавить». 5. Система сохраняет площадку в базе данных. 6. Система отображает страницу со списком площадок арендодателя.
Постусловия: Площадка успешно добавлена в базу.
Альтернативные потоки: Пользователь не заполнил одно из полей формы, система отобразила ошибку.

Таблица 13 – Спецификация прецедента «Добавить зал»

Прецедент: Добавить зал
ID: 13
Аннотация: Добавление зала

Главные актеры: Арендодатель
Второстепенные актеры: Нет
Предусловия: Арендодатель открыл список залов какой-либо своей площадки
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Добавить зал». 2. Система отображает форму для добавления зала. 3. Пользователь заполняет поля формы. 4. Пользователь нажимает «Добавить». 5. Система сохраняет зал в базе данных. 6. Система отображает страницу со списком залов арендодателя, относящихся к данной площадке.
Постусловия: Зал успешно добавлен в список залов данной площадки.
Альтернативные потоки: Пользователь не заполнил одно из полей формы, система отобразила ошибку.

Таблица 14 – Спецификация прецедента «Удалить аренду из расписания»

Прецедент: Удалить аренду из расписания
ID: 14
Аннотация: Удаление бронирования пользователя арендодателем
Главные актеры: Арендодатель
Второстепенные актеры: Нет
Предусловия: Арендодатель просматривает страницу со списком всех бронирований своих залов.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Удалить» возле одного из бронирований. 2. Система отображает всплывающее окно, требующее повторное подтверждение удаления. 3. Если пользователь нажимает «ОК» 3.1. Система удаляет соответствующую бронь из базы данных. 3.2. Система отображает обновленный список броней пользователя. 4. Если пользователь нажимает «Отмена», переход к шагу 4. 5. Система отображает список броней пользователя.
Постусловия: Пользователь успешно удалил бронь.
Альтернативные потоки: Пользователь нажал «Отмена», бронь не была удалена.

Таблица 15 – Спецификация прецедента «Посмотреть список броней и арендаторов»

Прецедент: Посмотреть список броней и арендаторов
ID: 15
Аннотация: Просмотр списка бронирований залов, относящихся к площадкам арендодателя
Главные актеры: Арендодатель
Второстепенные актеры: Нет

Предусловия: Нет
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Бронирования» 2. Система отображает страницу со списком всех бронирований, относящихся к площадкам пользователя.
Постусловия: Пользователь видит список всех бронирований его залов.
Альтернативные потоки: Нет

Таблица 16 – Список прецедента «Изменить параметры зала»

Прецедент: Изменить параметры зала
ID: 16
Аннотация: Редактирование параметров зала
Главные актеры: Арендодатель
Второстепенные актеры: Нет
Предусловия: Арендодатель просматривает страницу расписания своего зала.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Редактировать зал». 2. Пользователь меняет необходимые ему параметры. 3. Если пользователь нажимает «Сохранить изменения» система вносит изменения в соответствующий зал в базе данных. 4. Система отображает страницу расписания данного зала.
Постусловия: Пользователь успешно изменил параметры зала.
Альтернативные потоки: Пользователь нажал «Отмена», изменения не были сохранены.

Таблица 17 – Спецификация прецедента «Изменить условия аренды»

Прецедент: Изменить условия аренды
ID: 17
Аннотация: Удаление бронь из списка
Главные актеры: Арендодатель
Второстепенные актеры: Нет
Предусловия: Арендодатель просматривает страницу со списком броней его залов.
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Редактировать» возле одной из броней. 2. Система отображает форму редактирования бронирования. 3. Пользователь меняет необходимые ему параметры. 4. Если пользователь нажимает «Сохранить изменения». система вносит изменения в соответствующую бронь в базе данных. 5. Система отображает список броней пользователя.
Постусловия: Пользователь успешно изменил бронирование.
Альтернативные потоки: Пользователь нажал «Отмена», изменения не были сохранены.