

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**ПОДСИСТЕМА БРОНИРОВАНИЯ ПЕРЕГОВОРНЫХ
КОМНАТ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

ОРГАНИЗАЦИИ

Студент

 02.06.23
(подпись, дата)

Ходакова Д. А.
(ФИО полностью)

Руководитель

 02.06.23
(подпись, дата)

Синев М. П.
(фамилия, инициалы)

Нормоконтролёр

 02.06.23
(подпись, дата)

А.С. Бычков
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. №

14)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

14.06.2023 г. № 10)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» апреля 2023г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Ходакова Дарья Алексеевна гр. 19ВВП1 факультета вычислительной техники

2. Тема работы «Подсистема бронирования переговорных комнат информационной системы организации»

Тема утверждена приказом ПГУ № 458/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы к. н. т., Синев М. П.

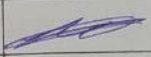
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать «Подсистему бронирования переговорных комнат информационной системы организации», которая будет способна автоматизировать процесс бронирования комнат и оптимизировать использование ресурсов организации. Система должна включать в себя функционал создания, редактирования и удаления бронирований, просмотр информации о переговорных комнатах и бронированиях пользователя и его коллег.

5. Перечень подлежащих разработке вопросов

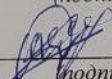
Требуется произвести анализ предметных областей: изучить аналоги существующих систем бронирования переговорных комнат, изучить способы проектирования и разработки веб-приложения, где последний поддерживает REST API методология. Также необходимо изучить функционал фреймворка Spring и системы управления базами данных PostgreSQL.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023-30.04.2023	
2.	Анализ требований	10	01.05.2023-02.05.2023	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2023-07.05.2023	
4.	Описание алгоритмов	40	08.05.2023-11.05.2023	
5.	Разработка приложения «Подсистема бронирования переговорных комнат информационной системы организации»	150	12.05.2023-29.05.2023	
6.	Оформление пояснительной записки	24	30.05.2023-31.05.2023	

Дата выдачи задания "26" апреля 2023г.

Руководитель бакалаврской работы  (подпись, дата)

Задание к исполнению принял студент  (подпись, дата)

Работу к защите допустить
Декан факультета  А.Р. Ривисов
(подпись, дата)
09.06.2023

Реферат

Бакалаврская работа содержит 82 л., 38 рис., 7 табл., 9 источн., 3 прил.
ПОДСИСТЕМА БРОНИРОВАНИЯ ПЕРЕГОВОРНЫХ КОМНАТ
ИНФОРМАЦИОННОЙ СИСТЕМЫ ОРГАНИЗАЦИИ, POSTGRESQL,
PGADMIN4, REST, API, JAVA, INTELLIJ IDEA, SPRING FRAMEWORK

Цель работы – разработать веб-приложение по бронированию переговорных комнат.

Объектом разработки является приложение, обеспечивающее выполнение следующих функций: создание, удаление, редактирование бронирований. Хранение информации о бронированиях и переговорных должно осуществляться в БД. Также необходимо реализовать *API* для взаимодействия серверной и клиентской частей.

В результате проведенной работы было разработано приложение, удовлетворяющее поставленной задаче, а именно: разработана серверная часть приложения, структура БД, создан *REST API*, реализующий необходимый функционал.

ПГУ 1.090301.25.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата
		Ходакова Д. А.		
		Синев М. П.		
		Бычков А. С.		
		Митрохин М. А.		
			Подсистема бронирования переговорных комнат информационной системы организации Пояснительная записка	
			Лит.	Лист
				2
			Листов	
			82	
ФВТ, каф. ВТ гр. 19ВВП1				

Содержание

Перечень принятых сокращений и обозначений	4
Введение.....	5
1 Подсистема бронирования переговорных комнат информационной системы организаций	6
1.1 Анализ темы.....	6
1.2 Технологии и инструменты реализации веб-приложения для бронирования переговорных комнат	17
1.3 Инструменты разработки	19
1.4 Выбор инструментов реализации	21
1.5 Разработка и реализация веб-приложения	23
1.6 Описание Rest API	27
1.7 Описание классов.....	30
1.8 Программная реализация сервера	35
1.9 Описание результата.....	37
2 Охрана труда.....	52
2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с компьютерами	52
2.2 Классификация и категорирование помещения, в котором производится разработка на компьютерах	53
2.3 Санитарно-гигиеническая характеристика.....	54
Заключение	56
Список использованных источников	57
Приложение А	58
Приложение Б.....	62
Приложение В.....	75

Перечень принятых сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – База данных

СУБД – система управления базами данных

API (Application Programming Interface) – программный интерфейс приложения

ID (Identifier) – идентификатор

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript

REST (Representational State Transfer) – передача состояния представления

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

URI (Uniform Resource Identifier) – унифицированный (единообразный) идентификатор ресурса

XML (eXtensible Markup Language) – расширяемый язык разметки

Введение

Информационные технологии стали неотъемлемой частью нашей современной жизни. Они окружают нас повсюду и влияют на различные аспекты нашего бытия. С каждым днем развитие информационных технологий ускоряется и вносит все больший вклад в нашу жизнь. Они облегчают нашу работу и повседневные задачи, увеличивают нашу мобильность, решают многие проблемы в области здравоохранения и образования, а также усиливают наши коммуникации.

Одним из направлений развития информационных технологий является создание специализированных веб-приложений для решения различных задач. Одним из таких полезных приложений является веб-приложение для бронирования переговорных комнат. Такие приложения значительно облегчают процесс бронирования комнат для встреч и конференций. Они позволяют быстро и удобно найти и забронировать необходимую комнату, выбрав нужное время и дату. Также с их помощью можно узнать информацию о стоимости услуги и о доступности различных помещений.

Особенно важной является роль веб-приложений для бронирования переговорных комнат в жизни бизнеса и компаний, которые постоянно организуют встречи с партнерами и клиентами. Такие приложения позволяют сократить время на организацию встреч и облегчить процесс взаимодействия между компаниями, упростив при этом процесс бронирования.

Веб-приложения для бронирования переговорных комнат являются необходимым и важным инструментом для любого бизнеса. Они помогают эффективно организовывать рабочее время, сокращать расходы на аренду и управлять распределением переговорных комнат. В целом, они повышают продуктивность и эффективность работы организации в целом.

1 Подсистема бронирования переговорных комнат информационной системы организаций

1.1 Анализ темы

В организованных компаниях наличие отдельных помещений для переговоров и централизованная система бронирования через календарь организации помогают избежать проблем с занятостью комнат. Однако, даже при таком идеальном подходе, часто возникают проблемы с недостаточной информированностью сотрудников о том, какая комната свободна, а какая занята, кто ей пользуется и пр. Иногда бронирование переговорных происходит только через секретаря на ресепшн, который может оказаться недоступен в любой момент. Проблемой является и то, что сотрудники иногда используют переговорные комнаты не по прямому назначению, что также может служить источником неудобств. Кроме того, в некоторых компаниях нет четкой системы и идентификации помещений для переговоров. В таких случаях сотрудники могут чувствовать себя неуверенно, не зная, какие помещения отведены под переговоры, что приводит к дискомфорту и затруднениям при организации встреч. Решением всех этих проблем является внедрение системы бронирования переговорных[1].

Система бронирования переговорных комнат решает проблему организации эффективных встреч, совещаний и презентаций в офисном пространстве. Благодаря такой системе сотрудники компаний могут быстро и легко бронировать необходимые комнаты на нужное время, без необходимости тратить время на поиск доступных переговорных и общих зон в офисе. Это помогает автоматизировать и улучшить работу коллектива, увеличить производительность и повысить эффективность решения рабочих задач. Также система бронирования переговорных помещений облегчает процесс управления офисным пространством, позволяя контролировать загрузку комнат и оптимизировать их использование.

					ПГУ 1.090301.25.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Система бронирования будет включать в себя бронирование переговорных комнат, расписание со всеми бронированиями сотрудника, где предоставляется информация о дне, времени бронирования, о переговорной, ее номере или названии, количестве вмещаемых участников, информация о них. Также в система будет предоставлять возможность просмотра доски бронирований всех переговорных или конкретного объекта на выбранный день.

Подобная система может позволить сотрудникам организации легко и удобно бронировать переговорные комнаты с помощью веб-интерфейса и мобильного приложения.

Целевая аудитория — владельцы и сотрудники организаций/компаний, желающие получить качественный удобный сервис по бронированию переговорных.

1.1.1 Сценарий использования

Сценарий использования — это техника моделирования системы, которая позволяет описывать взаимодействие пользователя с системой в контексте решения конкретной задачи или выполнения конкретного действия. Он является основой для определения требований к системе и позволяет проработать подробности взаимодействия пользователей с системой.

На диаграмме ниже (рисунок 1) представлен сценарий использования веб-приложения для бронирования переговорных комнат. Этот сценарий иллюстрирует несколько шагов, которые необходимо выполнить, чтобы забронировать переговорную комнату, удалить или редактировать бронирование, посмотреть доску бронирований и список объектов бронирования в нашем приложении.

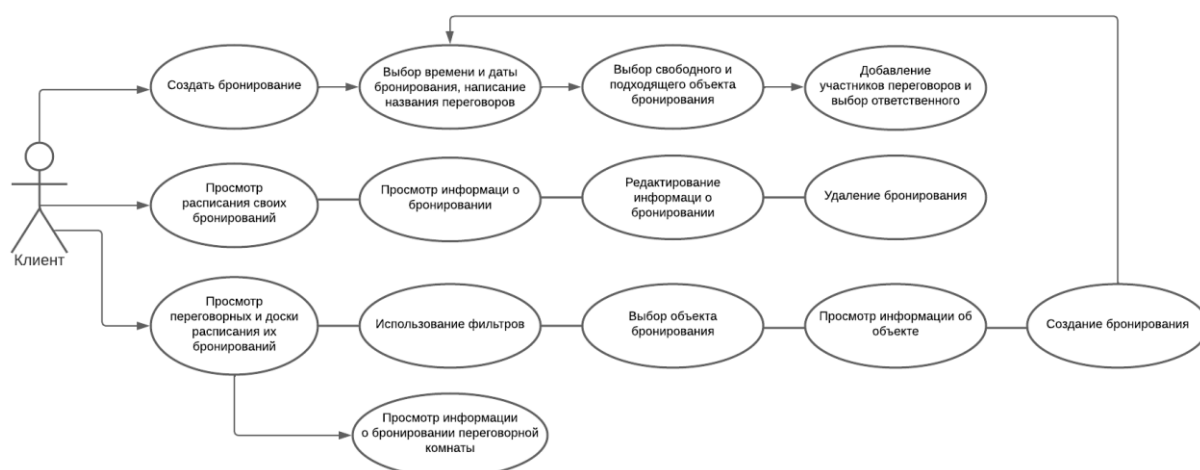


Рисунок 1 — Сценарий использования

Пользователь заходит в приложение и видит главный экран. На главном экране в меню есть кнопка "Создать", на которую пользователь нажимает, чтобы создать бронирование. Пользователь пишет название переговоров, выбирает дату, время и длительность бронирования. Далее открывается экран с доступными объектами для бронирования на указанные дату и промежуток времени. Если пользователь знает название объекта, он может воспользоваться функцией поиска. Пользователь выбирает подходящий для него объект для бронирования и нажимает на него, затем на кнопку «Далее». Открывается экран, где пользователь может выбрать людей, для которых будет проходить бронирование, и ответственного за бронирование. После внесения всех необходимых данных пользователь нажимает на кнопку "Создать".

Пользователь может просмотреть свои бронирования, нажав на соответствующую вкладку в меню на главном экране. На экране со списком бронирований пользователя он может навести на нужное бронирование. Появятся кнопки, которые отвечают за разные действия: просмотреть информацию об объекте: список участников, дату и время, информацию об объекте бронирования; отредактировать или удалить бронирование, если это необходимо.

Пользователь может просмотреть доску бронирований переговорных, нажав соответствующую вкладку в меню на главном экране. Открывается экран со списком переговорных и доской их бронирований. Пользователь может отфильтровать объекты по категории, по количеству людей и по доступности в данный момент или указав нужное время. Если пользователь знает название объекта, он может воспользоваться функцией поиска. Пользователь выбирает подходящий для него объект для бронирования и нажимает на него. Открывается экран с подробной информацией об объекте: описание, фотографии, кнопки «Закрыть» и «Забронировать». Нажав на панель бронирования на доске, пользователь может посмотреть подробную информацию о нем: название, дату, время, информацию об объекте бронирования и об участниках. Нажав на свое бронирование на доске пользователь может выбрать одно из действий: удалить или редактировать, где можно посмотреть информацию о бронировании.

Пользователь может забронировать переговорную нажав на свободное место на доске бронирований.

Таким образом, функционал приложения позволяет пользователям быстро и удобно выбирать и бронировать подходящий объект, а также управлять своими бронированиями. Он помогает пользователю сэкономить время и не тратить его на поиск и управление бронированиями вручную.

1.1.2 Анализ аналогов

В современном бизнесе переговоры являются неотъемлемой частью решения многих важных вопросов. Для удобства организации встреч многие компании используют системы бронирования переговорных комнат. Такие системы позволяют снизить время ожидания комнаты, избежать конфликтов при ее бронировании и повысить эффективность рабочего процесса. Создание новых технологий и развитие Интернета привели к появлению множества аналогов систем бронирования, каждый из которых имеет свои преимущества и недостатки.

В данном исследовании будет проанализировано несколько наиболее популярных систем бронирования переговорных комнат, с целью выявления их особенностей, преимуществ и недостатков.

UnSpot - облачный сервис «умный офис» для бронирования рабочих мест и переговорных комнат, для компаний, имеющих офис и от 100 сотрудников, работающих в гибридном формате-дом/офис.

Достоинства:

- мультиплатформенность;
- грамотный удобный и лаконичный дизайн сервиса;
- применение фильтров для поиска подходящих комнат и их бронирование (рисунок 3);
- доска расписания (рисунок 2);
- план офиса;
- управление системой;
- мониторинг загруженности помещений;
- администрирование и настройка офисов, оборудования, пользователей и доступа к технике;
- интеграции с календарями;
- планер задач;
- профиль пользователя (рисунок 4);
- настройка системы.

Недостатки:

- нет русского языка;
- неудобный планер задач (рисунок 5).

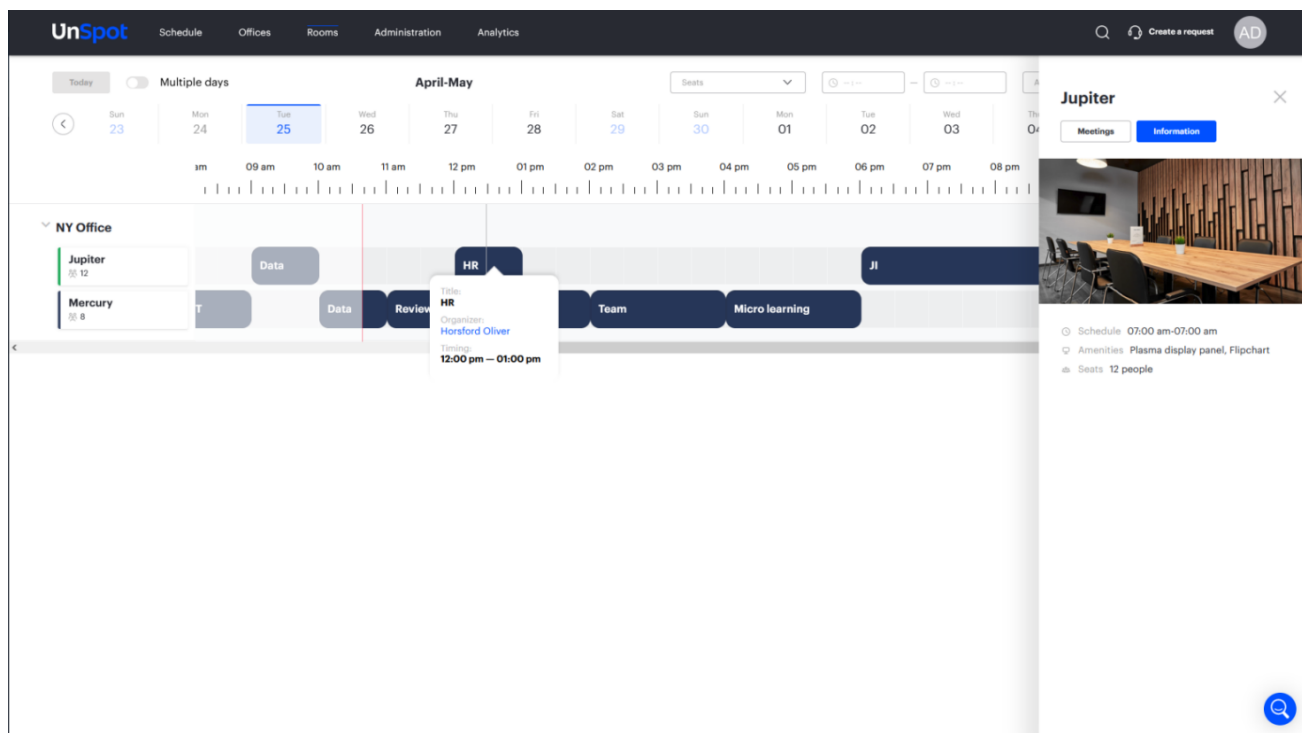


Рисунок 2 — Доска расписания бронирований переговорных

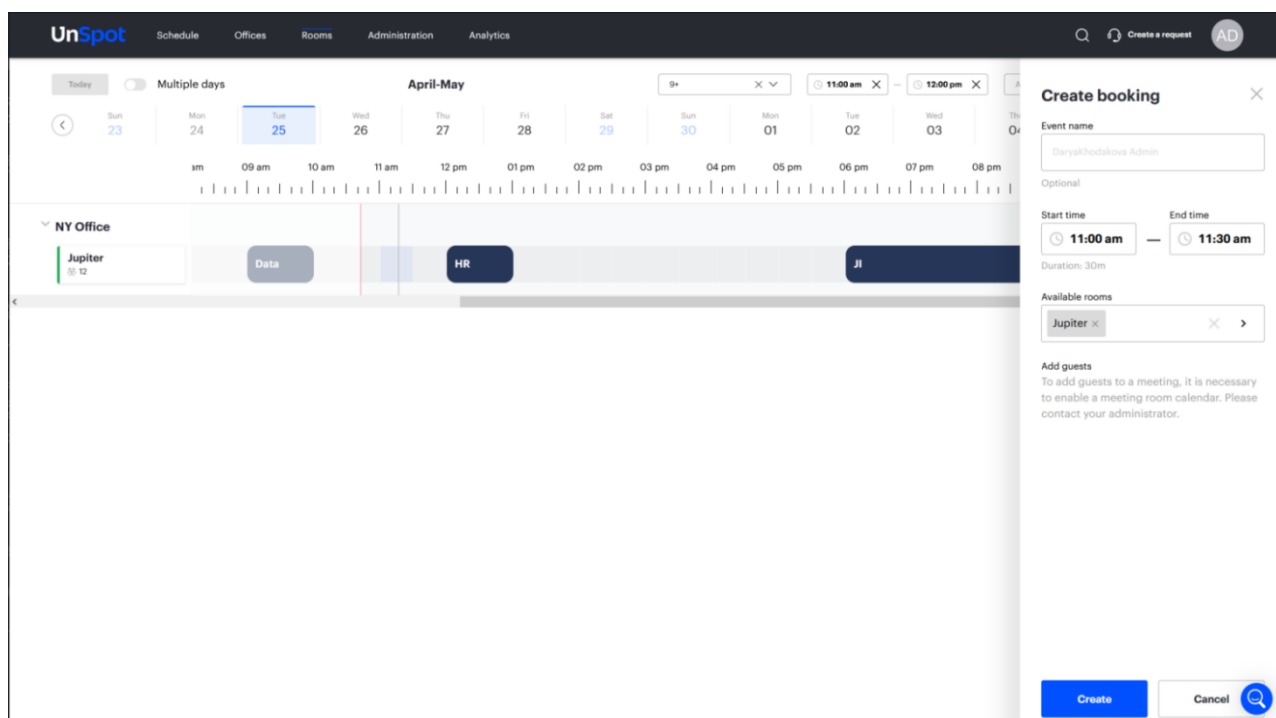


Рисунок 3 — Применение фильтров и создание бронирования

UnSpot Schedule Offices Rooms Administration Analytics

Search Create a request AD

User profile Change password

Profile

First Name: Admin Last Name: Daryakhodakova

Position: Enter your Position Department: Enter department

Manager: Enter manager Phone: Enter phone

Language: English User interface timezone: Automatically

Email: khodakova0305200157@gmail.com

Start working time: 09:00 am End working time: 06:00 pm

Save

My Bookings Calendars Reports Settings Profile Notifications Synchronizations Help Requests Help center Product updates Onboarding Sign out

Рисунок 4 — Профиль пользователя

UnSpot Schedule Offices Rooms Administration Analytics

Search Create a request AD

Current week < Apr 23 — Apr 29 > Sun 23 Mon 24 Tue 25 Wed 26 Thu 27 Fri 28 Sat 29

You AD Admin Daryakhodakova Not working Add Add Add Add Add Add Not working

0 interested in your schedule

Find your colleagues All offices Coffee Lunch Dinner

All colleagues

	Sun 23	Mon 24	Tue 25	Wed 26	Thu 27	Fri 28	Sat 29
Francesca Amati UX DESIGNER	Not working	Remote	NY Office 11	NY Office 11	NY Office 11	NY Office 11	Not working
Steve Becker IT	Not working	NY Office 16	NY Office 16	NY Office 16	NY Office 16	Remote	Not working
Kelly Bolen MARKETING MANAGER	Not working	NY Office 06	NY Office 06	Remote	NY Office 06	Remote	Not working
Lukas Fernandes FACILITY MANAGER	Not working	Remote	NY Office 25	NY Office 25	NY Office 25	NY Office 25	Not working
Melissa Green HR	Not working	Remote	NY Office 07	NY Office 07	NY Office 07	NY Office 07	Not working
Timothy Hamelton SALES MANAGER	Not working	NY Office 09	Remote	Remote	NY Office 09	NY Office 09	Not working
Oliver Horsford CMO	Not working	NY Office 21	NY Office 22	NY Office 22	Remote	Remote	Not working
Paulo Iskander COO	Not working	Remote	NY Office 23	NY Office 23	NY Office 23	NY Office 23	Not working
Halid Jamal IT MANAGER	Not working	NY Office 15	Remote	NY Office 15	NY Office 15	NY Office 15	Not working
Michael Jameson IT MANAGER	Not working	Remote	Remote	NY Office 01	NY Office 01	NY Office 01	Not working

Рисунок 5 — Планер задач.

Meru software - система управления офисом.

Достоинства:

- мультиплатформенность;

- план офиса (рисунок 6);
- возможность просмотра деталей бронирования такие, как участники бронирования, день и время.

Недостатки:

- дизайн веб-сайта и моб. приложения крайне примитивен и неудобен.

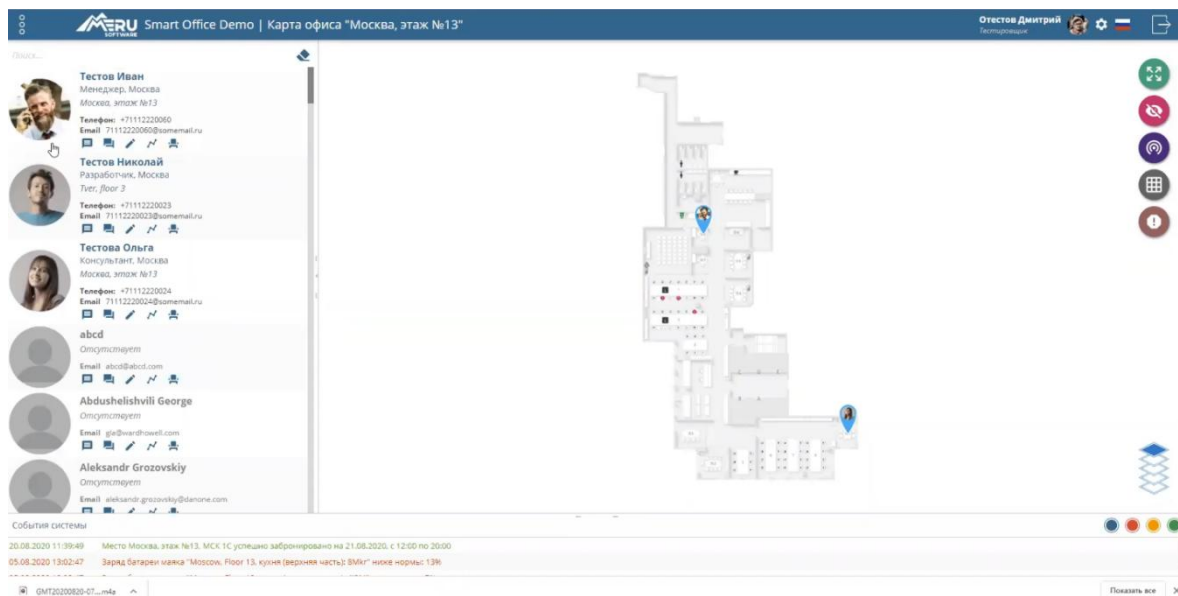


Рисунок 6 — План офиса.

UtiClock - это система управления бронированием переговорных комнат, синхронизирующаяся с корпоративным календарем и выводящая информацию о событиях на сенсорный экран.

Достоинства:

- интерфейс панели бронирования удобен;
- на экране показана информация о бронировании на несколько дней, статус комнаты (рисунок 7);
- отображение информации о текущем событии, об участниках и окончании времени бронирования;
- есть возможность настройки дизайна под стилистику компании.

Недостатки:

- в обычной версии системы можно бронировать только на сенсорной

панели, подходя к переговорной, если комната занята то придётся идти к другой;

- нет приложения по бронированию, интеграция с Битикс24 и *Microsoft Exchange* не удобна, если сотрудники компании используют разные системы, может возникнуть ошибка в бронировании или накладка.

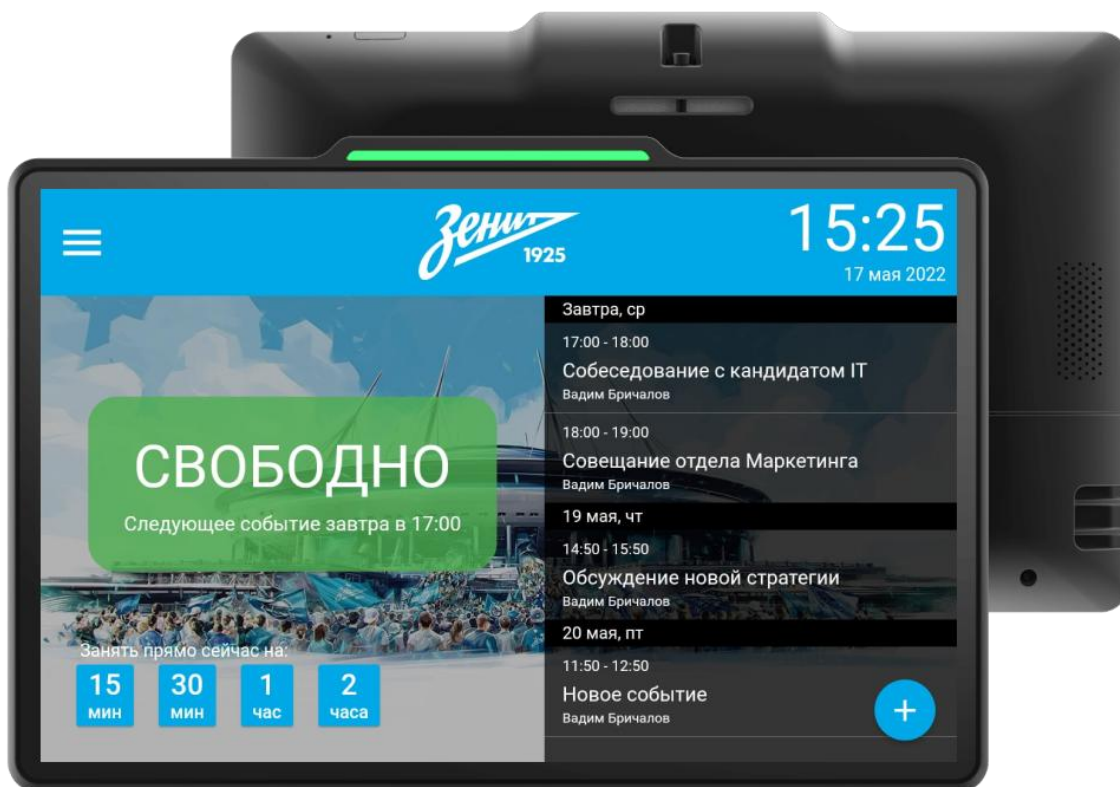


Рисунок 7 — Панель бронирования

Другие конкуренты имеют только панель бронирования, дизайн и функционал схожий.

Главный недостаток таких систем - дорогостоящее устройство, панель бронирования. В случае утраты устройства для переговорной комнаты - бизнес теряет до 300 тыс. руб.

1.1.3 Функционал веб-приложения

При разработке веб-приложения будут реализованы следующие функции:

- добавить бронирование:

- 1) просмотреть список объектов, доступных для бронирования;
 - а) отфильтровать по категории, что предоставит список конкретных объектов, которые нужны пользователю в данный момент;
 - б) поиск по названию объекта, поможет быстро найти конкретный объект;
 - с) отфильтровать по количеству людей, понадобится для пользователей знающих точное количество участников переговоров, что поможет ускорить поиск объекта бронирования;
 - д) отфильтровать по доступности в данный момент, поможет быстро найти свободный объект бронирования в данный момент;
 - е) отфильтровать по доступности на определенную дату и время, поможет быстро найти свободный объект бронирования на указанную определенную дату и время. Фильтры ускорят поиск нужного объекта для бронирования;
 - выбрать объект для бронирования;
- 2) просмотреть информацию об объекте, что поможет пользователю узнать об объекте подробнее:
 - а) выбрать время бронирования (дата, время, длительность);
 - б) выбрать людей, для которых бронируется данный объект, то есть участников переговоров, созданное бронирования отобразится для каждого участника в списке бронирований;
 - с) выбрать ответственного(по умолчанию - бронирующий);
 - просмотреть свои бронирования;
- 3) выбрать бронирование:
 - а) просмотреть информацию об объекте;
 - б) посмотреть список участников;
 - с) посмотреть дату и время;
 - удалить бронирование;
 - отредактировать бронирование (длительность, люди, ответственный).

1.1.4 Постановка задачи

Цель данной работы заключается в создании удобной и доступной на любом устройстве системы бронирования переговорных комнат, которая позволит эффективно управлять доступностью и использованием переговорных комнат. Для достижения этой цели необходимо решить следующие задачи:

- разработать дизайн и функционал пользовательского интерфейса. Разработать привлекательный и понятный пользовательский интерфейс, который позволит пользователям легко и быстро забронировать необходимые переговорные комнаты, а также легко найти информацию о доступных комнатах и дополнительной информации;

- разработать базу данных. Создать базу данных, в которой будут храниться все сведения о переговорных комнатах, включая информацию о количестве мест, стоимости, доступных удобствах, номере комнаты, расположении и дате и времени занятости комнаты;

- разработать алгоритм бронирования. Разработать алгоритм, который позволит пользователю забронировать нужную ему переговорную комнату в соответствии с выбранной датой и временем проведения мероприятия. Также нужно разработать алгоритм отмены бронирования;

- разработать серверную часть приложения. Создать серверную часть приложения, которая будет обрабатывать запросы от пользователей на бронирование, связываться с базой данных, чтобы создавать, обновлять и удалять записи о бронировании комнат.

1.1.5 Технические требования

Требования к десктопному устройству:

- процессор - минимум 4 ядра (рекомендуется 6 ядер);
- оперативная память - не менее 8 Гб (рекомендуется 12 Гб);
- только 64-разрядные серверные операционные системы, такие как Windows 7, 10, 11 XP, Vista, 8.1.

Требования к мобильному устройству:

- процессор - x64-разрядный двухъядерный процессор с тактовой частотой 3,3 ГГц и выше, с набором инструкций SSE2;
- оперативная память - ОЗУ объемом 4 Гб и более.

Требования к браузерам:

- *Google Chrome* версии 49 и выше;
- *Mozilla Firefox* версии 52 и выше;
- *Microsoft Edge* версии 14 и выше;
- *Opera* версии 36 и выше.

1.2 Технологии и инструменты реализации веб-приложения для бронирования переговорных комнат

1.2.1 Архитектура приложения

Архитектура клиент-серверного приложения предполагает разделение ролей между клиентской и серверной частями системы. Клиентская часть обеспечивает интерфейс для взаимодействия пользователя с системой и работает на компьютере пользователя, а серверная часть обеспечивает доступ к данным и другим ресурсам, необходимым для работы приложения, и запущена на сервере.

Разработка клиент-серверного приложения обычно включает несколько этапов:

- проектирование системы. Определение требований к приложению, выбор технологий и архитектуры приложения;
- разработка клиентской части. Создание пользовательского интерфейса, обработка пользовательского ввода и отправка запросов на сервер;
- разработка серверной части. Получение запросов от клиента, обработка данных и отправка ответа;

- тестирование и отладка. Проверка функциональности и производительности приложения.

Существует несколько видов архитектур клиент-серверных приложений, таких как двухуровневая, трехуровневая, n -уровневая и т.д. Для проектирования подсистемы бронирования переговорных комнат подходит двухуровневая архитектура.

На первом уровне находится клиентская часть приложения, которая представляет собой интерфейс, доступный через браузер пользователя. В данном случае, клиентской частью является форма бронирования переговорной комнаты с интерфейсом для выбора комнаты, даты и времени, добавления участников, а также отправки запроса на сервер для подтверждения брони комнаты.

На втором уровне находится серверная часть приложения, которая обрабатывает запросы от клиента, выполнение бизнес-логики, осуществляет связь с базой данных и отправляет ответ клиенту. Серверная часть приложения может быть реализована на языке программирования *Java*, *Python*, *Node.js* или любом другом серверном языке. На рисунке ниже наглядно показано взаимодействие веб-приложения с пользователем (рисунок 8).

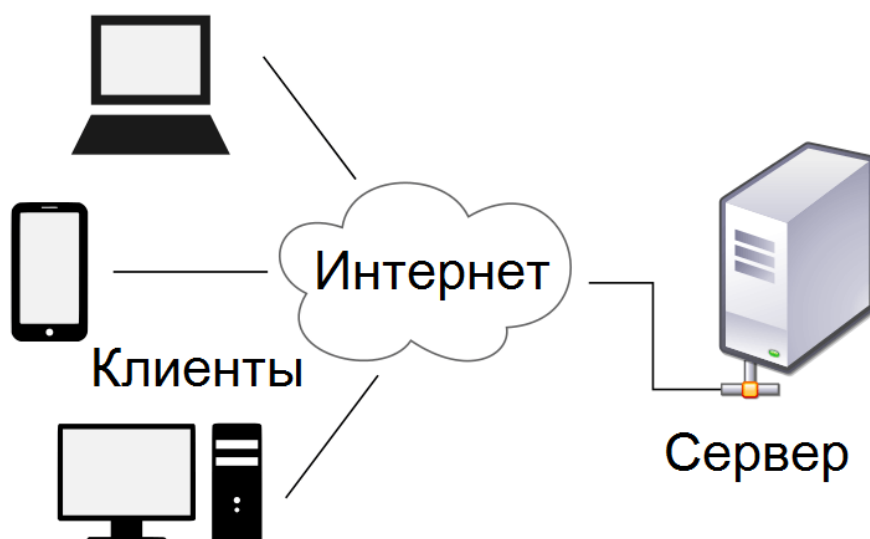


Рисунок 8 — Взаимодействие веб-приложения с пользователем

В данной архитектуре клиентской частью приложения является *web*-браузер пользователя, который отправляет запрос на серверную часть приложения по протоколу *HTTP* или *HTTPS*. Серверная часть обрабатывает запрос, выполняет операции, связанные с бронированием переговорной комнаты, и отправляет ответ на клиентский браузер. Коммуникация между серверной и клиентской частями может быть организована по протоколу *RESTful API* (рисунок 9).

В данной архитектуре серверная часть приложения может работать на любой операционной системе, такой как *Windows* и имеет доступ к базе данных с помощью системы управления базами данных (СУБД), такой как *MySQL*, *PostgreSQL* или *MongoDB*.

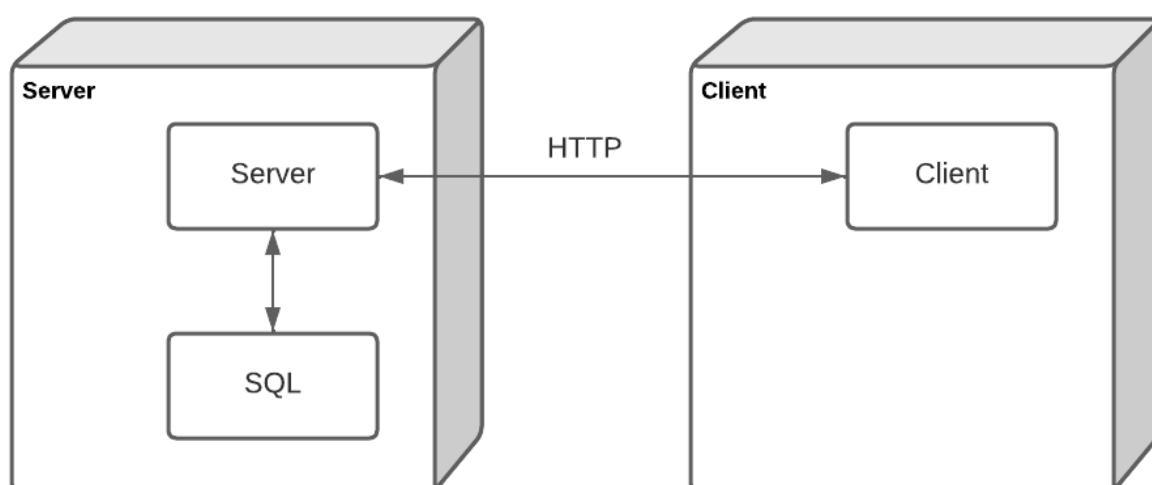


Рисунок 9 — Диаграмма структурных единиц (модулей) в архитектуре веб-приложения

Клиент-серверные приложения возможно масштабировать, добавляя дополнительные серверы и распределяя нагрузку между ними, для обеспечения более высокой производительности и устойчивости к отказам.

1.3 Инструменты разработки

1.3.1 Выбор языка программирования для серверной части

Языки программирования, которые могут быть использованы для создания серверной части веб-приложения, включают в себя *Java*, *C*, *Python*, *Ruby*, *PHP* и многие другие.

Java является одним из наиболее популярных языков программирования для создания серверной части веб-приложения. Его популярность связана с тем, что *Java* предоставляет мощные настройки и инструменты разработки, а также является очень надежным и безопасным языком программирования[2].

Один из факторов, который делает *Java* хорошим выбором для создания серверной части веб-приложения, заключается в его возможности использования многопоточности, что позволяет быстро обрабатывать большое количество запросов. Кроме того, *Java* очень популярен в корпоративной среде, что означает, что он имеет хорошую поддержку и множество библиотек, что упрощает и ускоряет процесс разработки[3].

В общем, *Java* является отличным выбором для создания серверной части веб-приложения, благодаря своей надежности, безопасности, мощным настройкам и инструментам разработки, а также многопоточности и хорошей поддержке в корпоративной среде.

1.3.2 Выбор фреймворка для серверной части

Существует несколько фреймворков для написания серверной части веб-приложения на языке *Java*:

- *Spring*. Это фреймворк для создания масштабируемых и эффективных веб-приложений. Он предоставляет различные инструменты для работы с базами данных, взаимодействия с пользовательским интерфейсом, обработки запросов и многое другое[4];

- *Hibernate*. Это *ORM*-фреймворк, который упрощает работу с базами данных, позволяя переключаться между различными базами данных без изменения кода;

- *Struts*. Это фреймворк для создания веб-приложений, который

основан на шаблоне проектирования *Model-View-Controller (MVC)*.

Несмотря на то, что все эти фреймворки предоставляют множество инструментов и возможностей для создания веб-приложений, Spring является одним из наиболее популярных и распространенных фреймворков для создания веб-приложений на языке *Java*[5]. *Spring* предоставляет широкий спектр инструментов, включая *Spring Boot*, *Spring Security*, *Spring Data* и многое другое, что облегчает разработку веб-приложений. *Spring* также позволяет интегрироваться с другими фреймворками, такими как *Hibernate* и *Struts*, что делает его еще более универсальным и гибким[6].

1.3.3 Выбор языка программирования для базы данных

Для работы с базами данных подходит структурированный язык запросов *SQL*.

SQL является стандартным языком для управления и манипулирования данными в реляционных базах данных (РБД)[7]. Он имеет широкий спектр операторов и встроенных функций для работы с данными, что делает его мощным инструментом для написания запросов к базам данных.

1.4 Выбор инструментов реализации

1.4.1 Среда разработки веб-приложения

Для написания серверной и клиентской части веб-приложения по бронированию переговорных комнат будет использоваться *IntelliJ IDEA*.

Среда разработки *IntelliJ IDEA* отличается высокой производительностью, интеграцией со многими технологиями и фреймворками, также предоставляет широкие возможности по автоматическому построению и тестированию проектов. Для создания вебприложения на *Java Spring* и *React JS* *IntelliJ IDEA* является лучшим выбором благодаря следующим причинам:

- интеграция с *Java Spring* и *React JS*. Одним из главных

преимуществ среды разработки *IntelliJ IDEA* является интеграция с *Java Spring* и *React JS*, позволяющая значительно упростить процесс разработки, а также быстро находить и устранять ошибки;

- мощные инструменты для разработки *Java*-приложений. *IntelliJ IDEA* предлагает мощные инструменты для разработки *Java*-приложений, которые позволяют быстро создавать, отлаживать и тестировать код;

- встроенная система контроля версий. *IntelliJ IDEA* предоставляет встроенную систему контроля версий, которая позволяет легко управлять внесенными изменениями и синхронизировать код с другими разработчиками;

- широкие возможности по автоматизации процесса разработки. *IntelliJ IDEA* имеет широкие возможности по автоматизации процесса

разработки, включая автоматическое построение проектов, автоматизированные тесты и интеграцию с системами непрерывной интеграции.

Сравнивая *IntelliJ IDEA* с конкурентами, можно отметить, что другие среды разработки, такие как *Eclipse* или *NetBeans*, также имеют интеграцию с *Java Spring* и *React JS*, но не настолько удобны и функциональны, как *IntelliJ IDEA*. Кроме того, *IntelliJ IDEA* предоставляет ряд эксклюзивных функций и инструментов, которые повышают производительность и качество разработки.

1.4.2 Система управления базами данных

PostgreSQL является сильным конкурентом *MySQL* и *SQLite* в мире систем управления баз данных. Он предлагает множество расширенных функций и возможностей, обеспечивающих высокую производительность, надежность и безопасность.

Некоторые причины, почему *PostgreSQL* для системы бронирования переговорных комнат предпочтительнее *MySQL* или *SQLite*:

- открытый исходный код. *PostgreSQL* является *open-source* проектом,

что означает, что его исходный код доступен для изменения и распространения. Это делает его более гибким и динамичным, чем проприетарные системы типа *MySQL*;

- расширенные возможности. *PostgreSQL* имеет множество расширенных функций, таких как оконные функции, *common table expressions*, *JSON* и *XML* поддержку, массивы и множество других. Эти функции позволяют разработчикам написать более сложные запросы и процедуры;

- надежность. *PostgreSQL* имеет высокий уровень надежности и безопасности данных. Он поддерживает транзакции для обеспечения целостности данных и восстановления данных в случае сбоев;

- расширяемость. *PostgreSQL* поддерживает расширяемые модули, которые могут быть написаны на различных языках программирования, таких как *Python*, *C ++* и *Java*. Это делает *PostgreSQL* более гибким для интеграции с другими приложениями и библиотеками;

- лицензия. *PostgreSQL* имеет лицензию распространения *BSD*, что означает, что он может быть свободно использован как для коммерческих, так и для некоммерческих проектов[8].

Таким образом, *PostgreSQL* является высокопроизводительной и надежной системой управления базами данных, которая обладает множеством расширенных функций и возможностей. Он является хорошим выбором для проектов любой сложности.

1.5 Разработка и реализация веб-приложения

Процесс разработки веб-приложения по бронированию переговорных комнат можно разделить на следующие этапы:

- разработка структуры БД;
- проектирование интерфейса веб-приложения;
- разработка *REST API*;
- разработка программы.

1.5.1 Разработка базы данных веб-приложения

При разработке сервиса по бронированию переговорных комнат была использована СУБД *PostgreSQL*. База данных необходима для хранения исходных данных об объектах бронирования, то есть переговорных комнат, их категориях и времени работы, расписании бронирований, а также сведений о забронированных переговорных комнат. Кроме этого в базу данных необходимо записывать сведения о пользователях, которые, такие как *id*, фамилия, имя, отчество, роль.

Далее представлена схема структуры разработанной БД (рисунок 10).

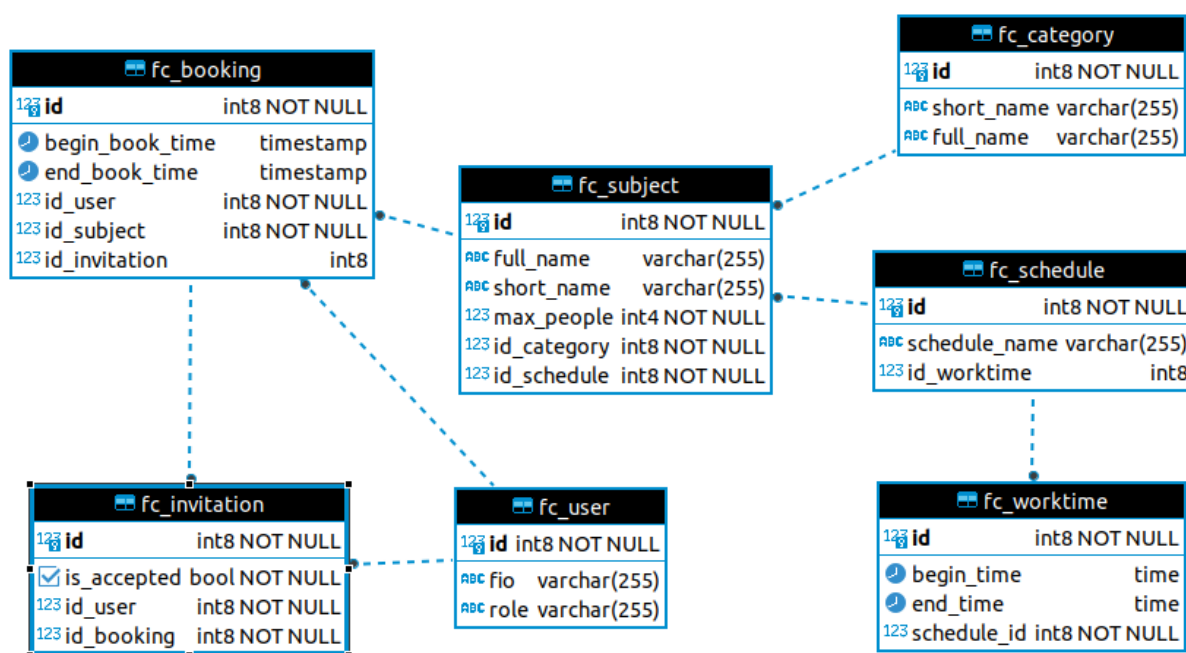


Рисунок 10 — Схема базы данных

Для реализации требований к приложению необходимо разработать 7 таблиц:

Таблица 1 содержит описание структуры таблицы «Бронирование (*fc_booking*)».

Таблица 1 — Структура таблицы «Бронирование (*fc_booking*)»

Название поля	Описание поля	Тип данных
id	Идентификатор бронирования	Int8
begin_book_time	Время начала бронирования	TimeStamp
end_book_time	Время конца бронирования	TimeStamp
Id_user	Идентификатор пользователя	Int8
Id_subject	Идентификатор переговорной	Int8
Id_invitation	Идентификатор уведомления	Int8

Таблица 2 содержит описание структуры таблицы «Объект бронирования(*fc_subject*)».

Таблица 2 — Структура таблицы «Объект бронирования (*fc_subject*)»

Название поля	Описание поля	Тип данных
id	Идентификатор объекта бронирования	Int8
full_name	Полное название переговорной	VarChar(255)
Short_name	Короткое название переговорной	VarChar(255)
Max_people	Максимальное количество пользователей	Int4
Id_category	Идентификатор категории	Int8
Id_schedule	Идентификатор расписания	Int8

Таблица 3 содержит описание структуры таблицы «Категория (*fc_category*)».

Таблица 3 — Структура таблицы «Категория (*fc_category*)»

Название поля	Описание поля	Тип данных
id	Идентификатор категории	Int8
full_name	Полное название категории	VarChar(255)
Short_name	Короткое название категории	VarChar(255)

Таблица 4 содержит описание структуры таблицы «Расписание (*fc_schedule*)».

Таблица 4 — Структура таблицы «Расписание(*fc_schedule*)»

Название поля	Описание поля	Тип данных
id	Идентификатор расписания	Int8
shedule_name	Название расписания	VarChar(255)
Id_worktime	Идентификатор времени работы	Int8

Таблица 5 содержит описание структуры таблицы «Время работы(*fc_worktime*)».

Таблица 5 — Структура таблицы «Время работы(*fc_worktime*)»

Название поля	Описание поля	Тип данных
id	Идентификатор времени работы	Int8
begin_time	Время начала работы	time
end_time	Время окончания работы	time
shedule_id	Идентификатор расписания	Int8

Таблица 6 содержит описание структуры таблицы «Пользователь(*fc_user*)».

Таблица 6 — Структура таблицы «Пользователь(*fc_user*)»

Название поля	Описание поля	Тип данных
id	Идентификатор пользователя	Int8
fio	Фамилия Имя Отчество пользователя	VarChar(255)
role	Роль	VarChar(255)

Таблица 7 содержит описание структуры таблицы «Уведомления(*fc_invitation*)».

Таблица 7 — Структура таблицы «Уведомления(*fc_incitation*)»

Название поля	Описание поля	Тип данных
id	Идентификатор уведомления	Int8
Is_accepted	Логическое значение просмотра уведомления	bool
Id_user	Идентификатор пользователя	time
Id_booking	Идентификатор бронирования	Int8

Листинг SQL-файла, формирующего БД находится в приложении Б.

1.6 Описание Rest API

REST API - это стандарт программного интерфейса, который определяет, как веб-приложения должны взаимодействовать друг с другом. При разработке *REST API* необходимо придерживаться определенных правил и структуры запросов.

Для разработки *REST API* необходимо определить структуру запросов и ответов, которые будут использоваться в приложении. В *REST API* используются определенные правила и форматы запросов и ответов, основанные на протоколах *HTTP* и *HTTPS*[9].

Разработка *REST API* для веб-приложения по бронированию переговорных должна включать в себя такие функции, как:

- получение списка доступных переговорных комнат и их характеристик;
- просмотр информации об объектах бронирования;
- создание нового бронирования переговорной комнаты;
- просмотр информации о бронированиях своих и других пользователей;
- редактирование существующего бронирования;
- удаление существующего бронирования.

Полное описание *REST API* для работы с пользователями находится в приложении Б.

Для примера рассмотрим запрос на создание пользователя. При создании объекта в *REST API* необходимо использовать *HTTP* метод *POST* (листинг 2). Также имеется требование к структуре *URI* запроса, которая должна содержать информацию о названии микросервиса, названии сущности и операции или идентификаторе объекта. Тело запроса представляет собой поля, необходимые для создания пользователя.

После успешного создания объекта сервер возвращает код 200 и объект, который содержит информацию об созданном пользователе, включая

переданные при создании поля и новые поля, добавленные после создания объекта.

Листинг 1 — Запрос получения категорий

```
Метод: GET
URI: /categories
Тело запроса:
Ответ:

200      OK
        {
          "content": [
            {
              "categoryId": "1703a",
              "fullName": "Leather chairs",
              "shortName": "Chairs"
            }
          ],
          "first": true,
          "last": true,
          "number": 0,
          "numberOfElements": 0,
          "size": 0,
          "totalElements": 0,
          "totalPages": 0
        }

400      Bad Request
        { "message": "string" }
401      Not authorized
        { "message": "string" }
403      Forbidden
        { "message": "string" }
500      Server error
        { "message": "string" }
```

Листинг 2 — Запрос на создание категории

```
Метод: POST
URI: /categories
Тело запроса:
{
  "fullName": "Leather chairs",
  "shortName": "Chairs"
}
Ответ:
```

```

200      OK
        {
          "categoryId": "1703a",
          "fullName": "Leather chairs",          "shortName":
          "Chairs"
        }
400      Bad Request
        { "message": "string" }
401      Not authorized
        { "message": "string" }
403      Forbidden
        { "message": "string" }
500      Server error
        { "message": "string" }

```

Листинг 3 — Запрос поиска категории

Метод: GET
 URI: /categories/{categoryId}
 Тело запроса:
 Ответ:

```

200      OK
        {
          "categoryId": "1703a",
          "fullName": "Leather chairs",          "shortName":
          "Chairs"
        }
400      Bad Request
        { "message": "string" }
401      Not authorized
        { "message": "string" }
403      Forbidden
        { "message": "string" }
500      Server error
        { "message": "string" }

```

Листинг 4 — Запрос удаления категории

Метод: DELETE
 URI: /categories/{categoryId}
 Тело запроса:
 Ответ:

```

200      OK
400      Bad Request
        { "message": "string" }
401      Not authorized

```

```

403      { "message": "string" }
Forbidden
500      { "message": "string" }
Server error
         { "message": "string" }

```

Ли
ст

инг 5 — Запрос редактирования категории

Метод: PUT

URI: /categories/{categoryId}

Тело запроса:

```

{
  "fullName": "Leather chairs",
  "shortName": "Chairs"
}

```

Ответ:

```

200      OK
         {
           "categoryId": "1703a",
           "fullName": "Leather chairs",      "shortName":
           "Chairs"
         }
400      Bad Request
         { "message": "string" }
401      Not authorized
         { "message": "string" }

403      Forbidden
         { "message": "string" }
500      Server error
         { "message": "string" }

```

1.7 Описание классов

Для разработки веб-приложения по бронированию переговорных комнат, включающего взаимодействие с СУБД и *REST API*, разделим архитектуру на следующие основные функциональные блоки:

- модели данных или классы сущностей. Описывают структуру данных, которые будут передаваться между клиентом и сервером. Например, для взаимодействия с пользователями могут быть созданы классы User,

Address, Phone и т.д.;

- классы сервисов или классы, реализующие бизнес-логику. Отвечают за обработку запросов от клиента и взаимодействие с базой данных. Например, для работы с пользователями может быть создан класс *UserService*, который содержит методы для выполнения операций над пользователями, например, методы для создания, чтения, обновления и удаления (*CRUD*) пользователей;

- классы контроллеров. Обеспечивают взаимодействие между клиентской частью приложения и серверной частью через *REST API*. Контроллеры обычно содержат обработчики запросов, которые вызывают соответствующие методы сервисов и возвращают данные клиенту в формате *JSON*. Например, если клиентское приложение отправляет запрос на создание нового пользователя, то соответствующий обработчик запроса на сервере вызывает метод *UserService* для создания нового пользователя и возвращает ответ с информацией о созданном пользователе в формате *JSON*;

- классы баз данных или классы репозитория. Предоставляются для взаимодействия с базой данных. Они обеспечивают сохранение и извлечение данных из базы в соответствующих моделях данных. Например, для работы с пользователями может быть создан класс *UserRepository*, который содержит методы для выполнения запросов к базе данных для создания, чтения, обновления и удаления (*CRUD*) пользователей;

- классы конфигурации и классы утилит. Содержат настройки приложения, параметры подключения к базе данных, настройки авторизации и аутентификации, а также утилиты для обработки запросов и ответов. Например, для обеспечения безопасной передачи данных между клиентом и сервером могут использоваться классы утилит для шифрования и дешифрования данных;

- классы обработки ошибок. Обрабатывают возможные ошибки, которые могут произойти при взаимодействии с *REST API* и СУБД, и

возвращают уведомления о возникших проблемах.

На примере сущности «Бронирование» рассмотрим взаимодействие всех классов.

Класс «*Booking*» содержит методы, которые позволяют получить и установить свойства объекта бронирования, такие как дата и время начала и конца бронирования, имя клиента, который забронировал комнату, название или номер переговорной, ее описание, вместимость. Также описывается отношение сущности к остальным сущностям (сущности «Объект бронирования», «Пользователь», «Уведомления»).

UML-диаграмма классов сущности «*Booking*» представлена на рисунке 11.

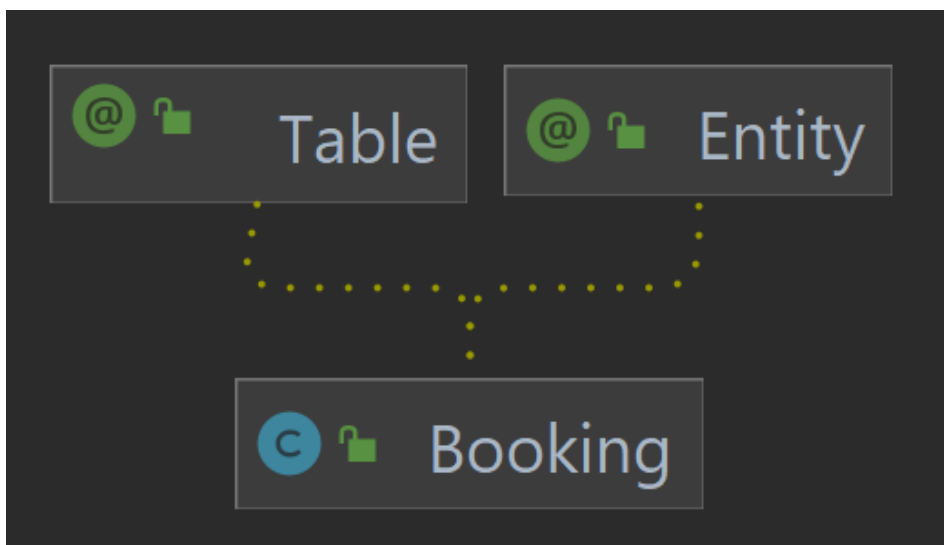


Рисунок 11 — UML-диаграмма классов сущности «Бронирование»

Класс-контроллер «*BookingController*» отвечает за обработку входящих запросов, валидацию входных данных, вызов методов класса-сервиса («*BookingService*») и возврат результата.

UML-диаграмма, показывающая структуру класса-контроллера «*BookingController*», представлена на рисунке 12.

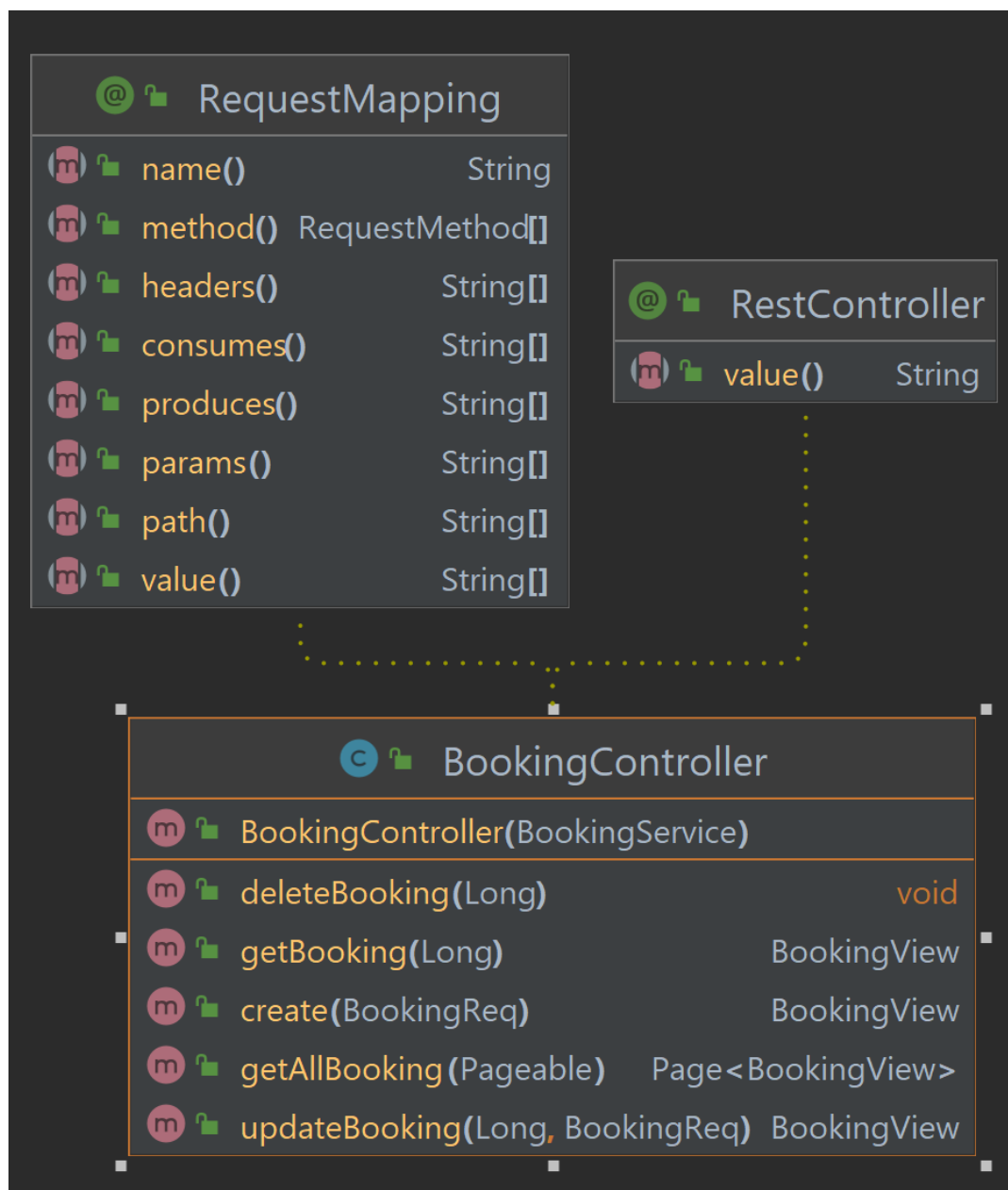


Рисунок 12 — UML-диаграмма класса-контроллера
«*BookingController*»

Класс-репозиторий «*BookingRepo*» используется для выполнения операций с базой данных «Бронирования» (рисунок 13). Репозиторий будет использоваться для извлечения данных из таблицы "Бронирования". Репозиторий «*BookingRepo*» содержит методы для создания бронирования, чтения информации о существующих бронированиях, обновления данных о бронировании и удаления бронирования, при необходимости.

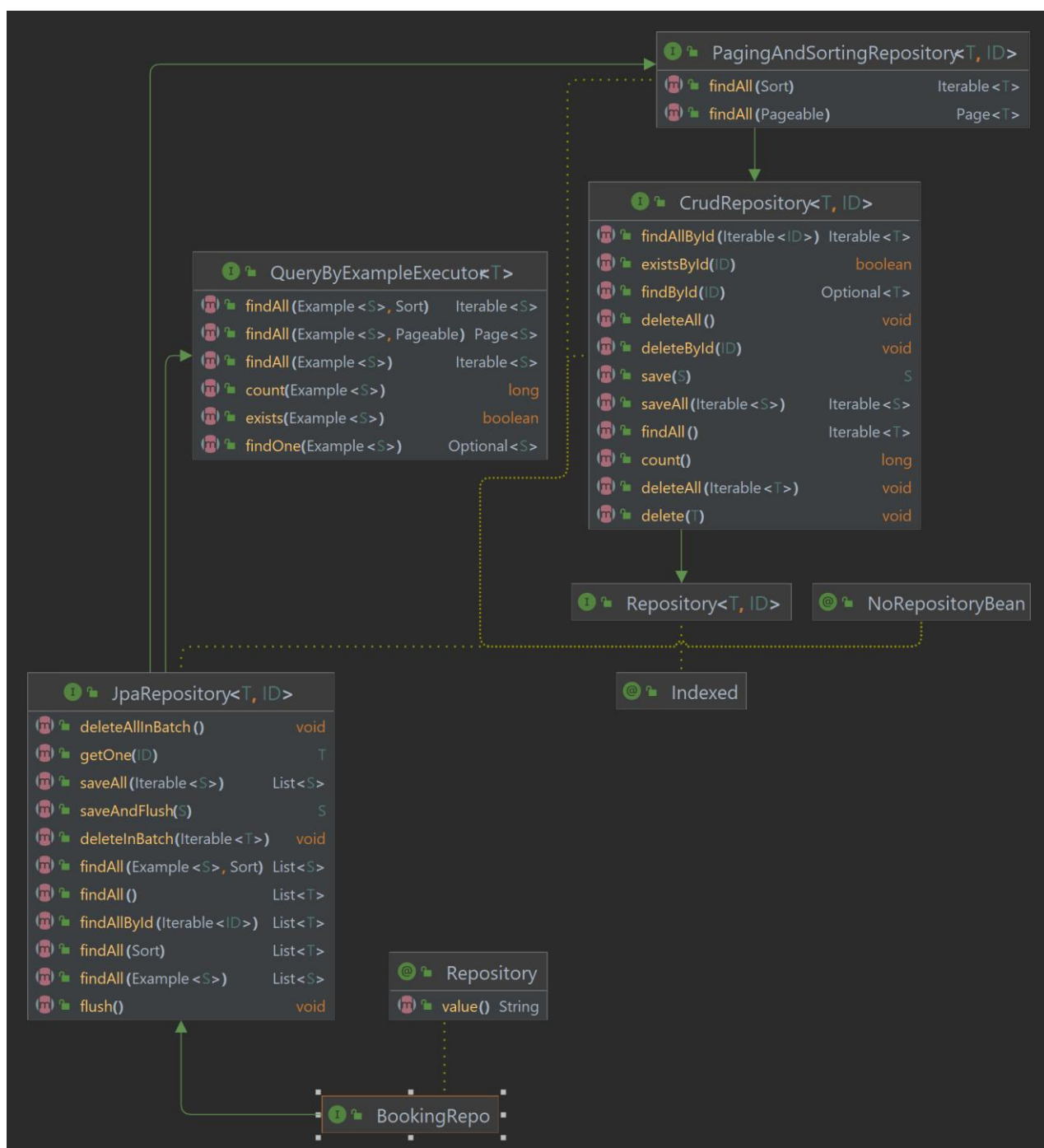


Рисунок 13 — UML-диаграмма класса-контроллера «UserRepo»

Функции класса «BookingService» позволяют работать с объектами, описывающими пользователей, производя подготовку объектов к передаче для обработки в класс-репозиторий. На UML-диаграмме класса-репозитория «BookingService» (рисунок 14) показаны методы, которые используются для обработки бронирований переговорных комнат.

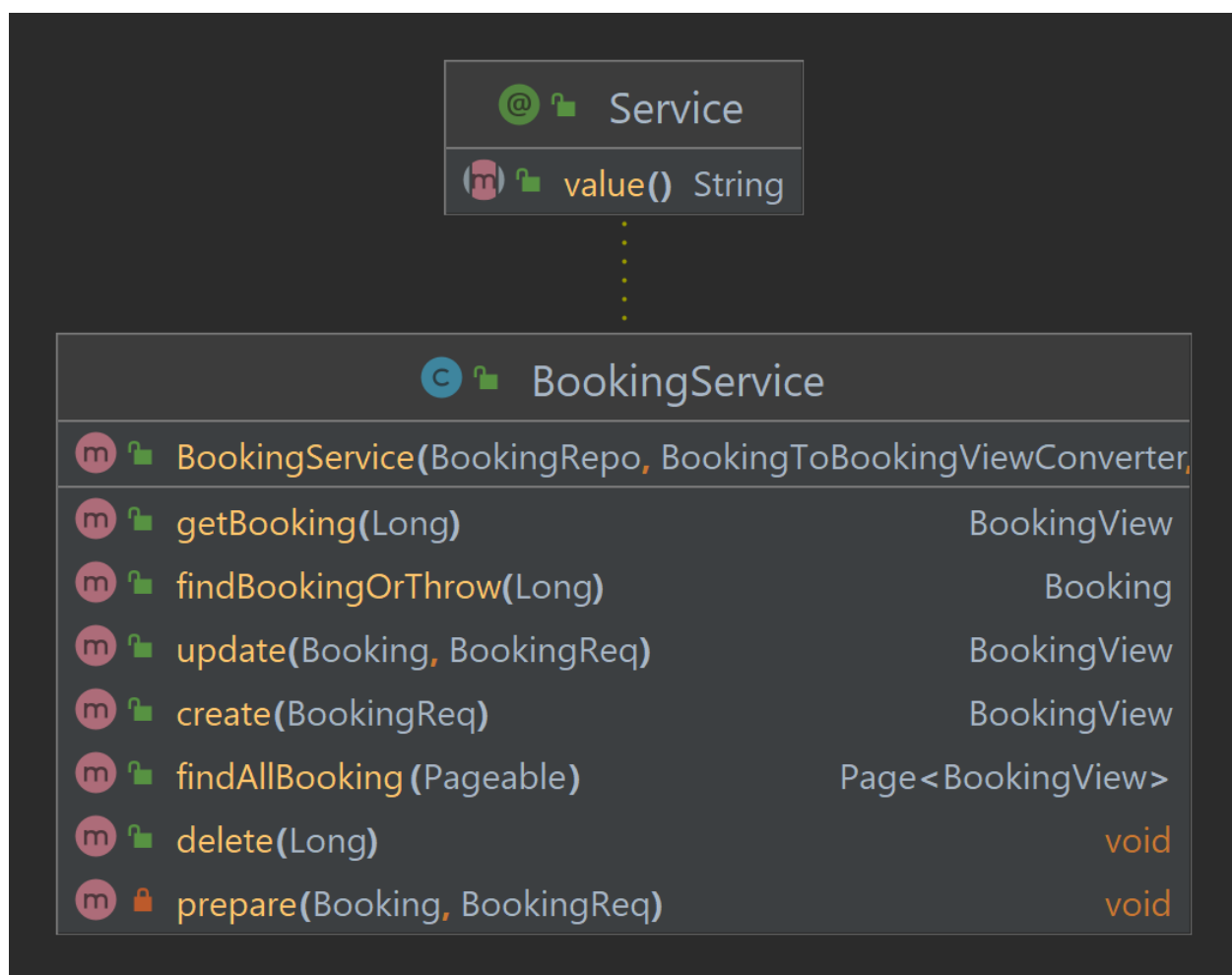


Рисунок 14 — UML-диаграмма класса-контроллера «*BookingService*»

1.8 Программная реализация сервера

Для реализации приложения по бронированию переговорных комнат используется язык программирования *Java* и фреймворк *Spring*. Был реализован *API* для работы с пользователями, бронированием, переговорными комнатами и другими сущностями. Исходный код классов, описанный в предыдущем разделе, приведен в приложении Б.

На UML-диаграмме (рисунок 15) показана последовательность взаимодействия классов при обработке запроса на получение бронирования по *ID*, учитывая структуру приложения по бронированию переговорных комнат.

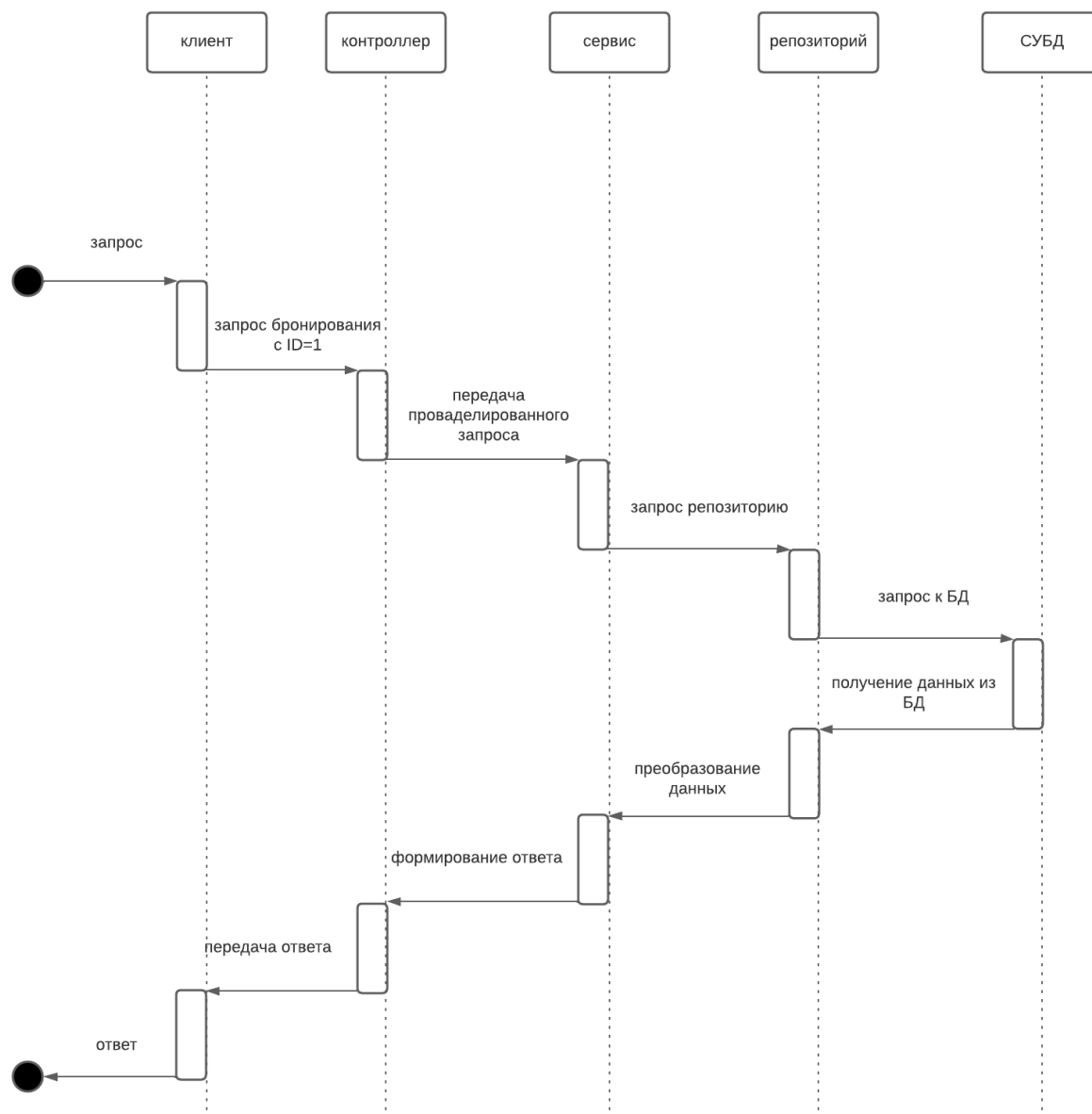


Рисунок 15 — Последовательность взаимодействия классов при обработке запроса «Получение бронирования по ID»

Через поисковую строку клиентского приложения пользователь запрашивает данные, которые генерируют запрос к классу-контроллеру на сервере для получения информации о бронировании по ID. Класс-контроллер проверяет запрос на соответствие требованиям валидации и, в случае успеха, передает его в класс-сервис. Класс-сервис готовит запрос для взаимодействия с объектами класса-репозитория, который отправляет запрос к базе данных. После получения необходимых данных, класс-репозиторий преобразует их в

объекты программы и передает их в класс-сервис. Класс-сервис формирует ответ для класса-контроллера, который передает результат обратно в клиентское приложение для отображения пользователю. Исходный код на языке Java для класса-сущности «Бронирование», класса-контроллера, сервиса и репозитория приведен в Приложении Б.

1.9 Описание результата

В результате было разработано веб приложение, адаптированное под десктопные и мобильные устройства.

1.9.1 Создание бронирования

1.9.1.1 На десктопном устройстве

Создать бронирование можно разными способами: нажать кнопку «Создать», в окне «Объекты бронирования» нажать на доску бронирований объектов. Рассмотрим создание по нажатию кнопки «Создать» в окне «Мои бронирования». На рисунке 16 пользователь навел на свое бронирование в списке, появившиеся иконки отвечают за подробную информацию, редактирование и удаление соответственно.

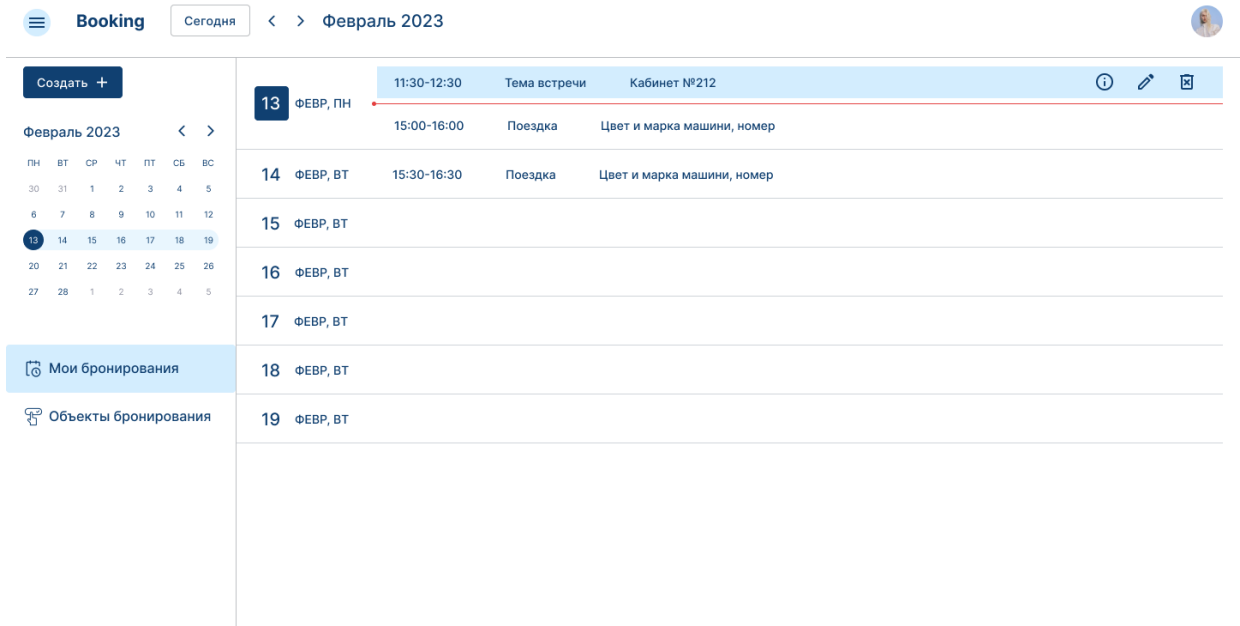


Рисунок 16 — Главный экран «Мои бронирования»

После нажатия на кнопку «Создать» появляется окно, где пользователь вводит название встречи, выбирает дату и время в соответствующих полях (рисунок 17).

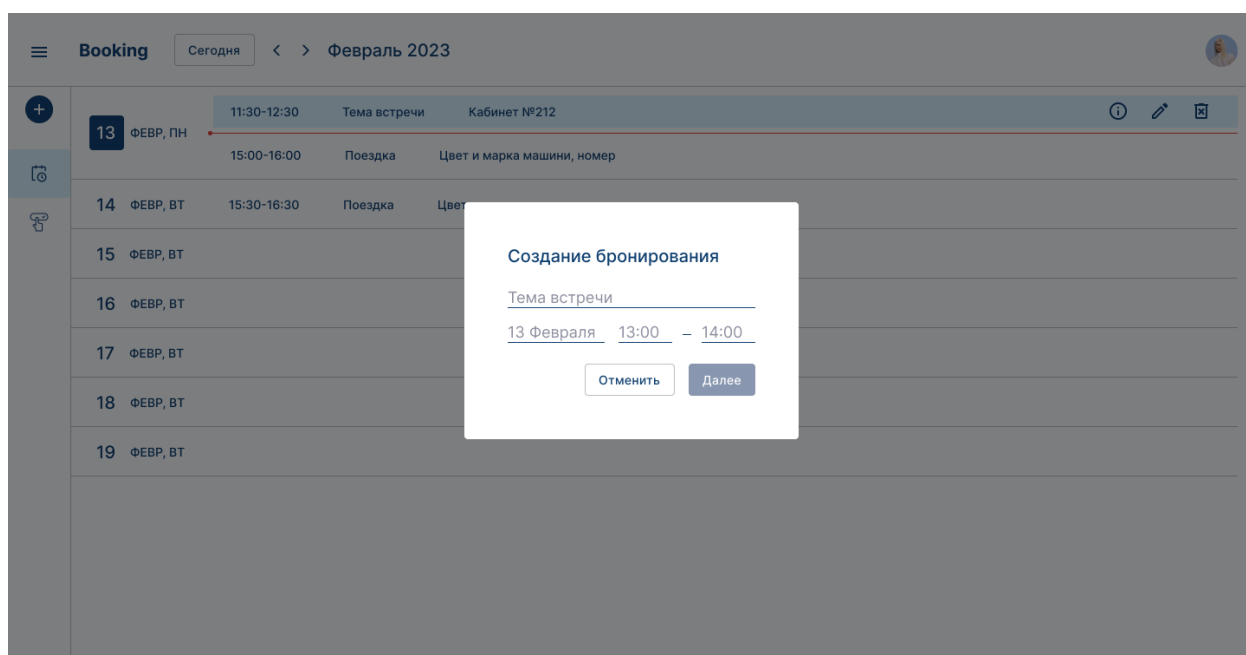


Рисунок 17 — Создание бронирования при нажатии кнопки «Создать»

При нажатии на кнопку «Далее» окно ввода основной информации меняется на окно выбора объекта бронирования (рисунок 18).

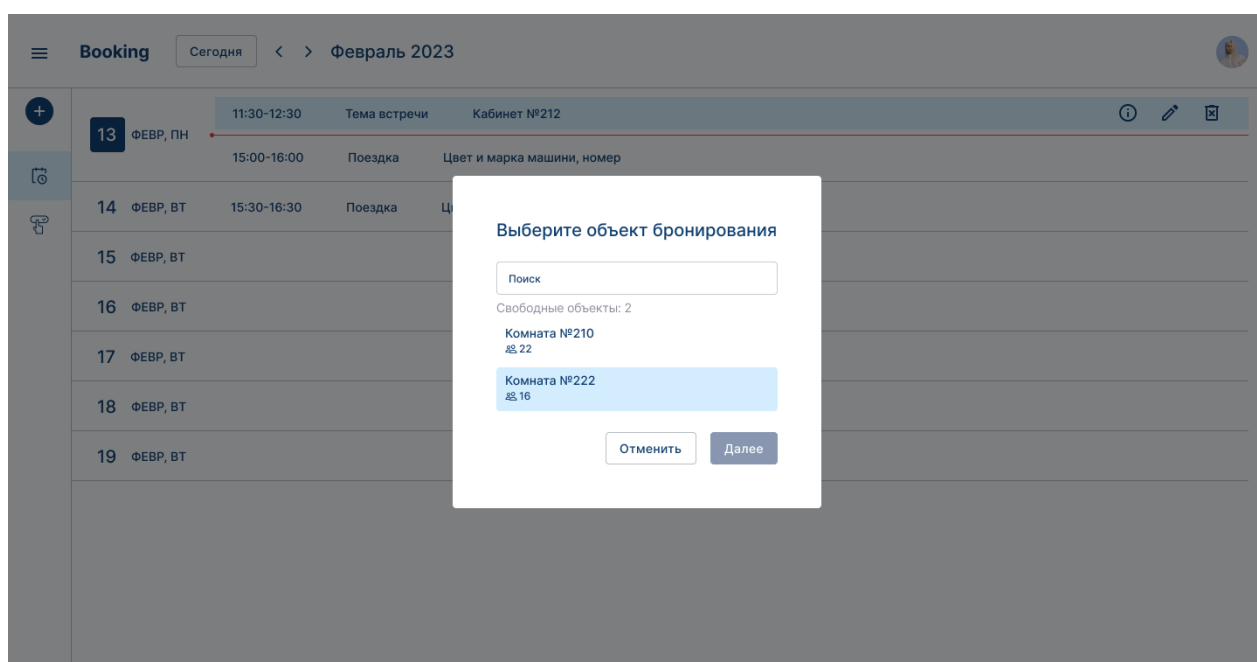


Рисунок 18 — Выбор объекта бронирования

При нажатии на кнопку «Далее» появляется окно добавления участников (рисунок 19).

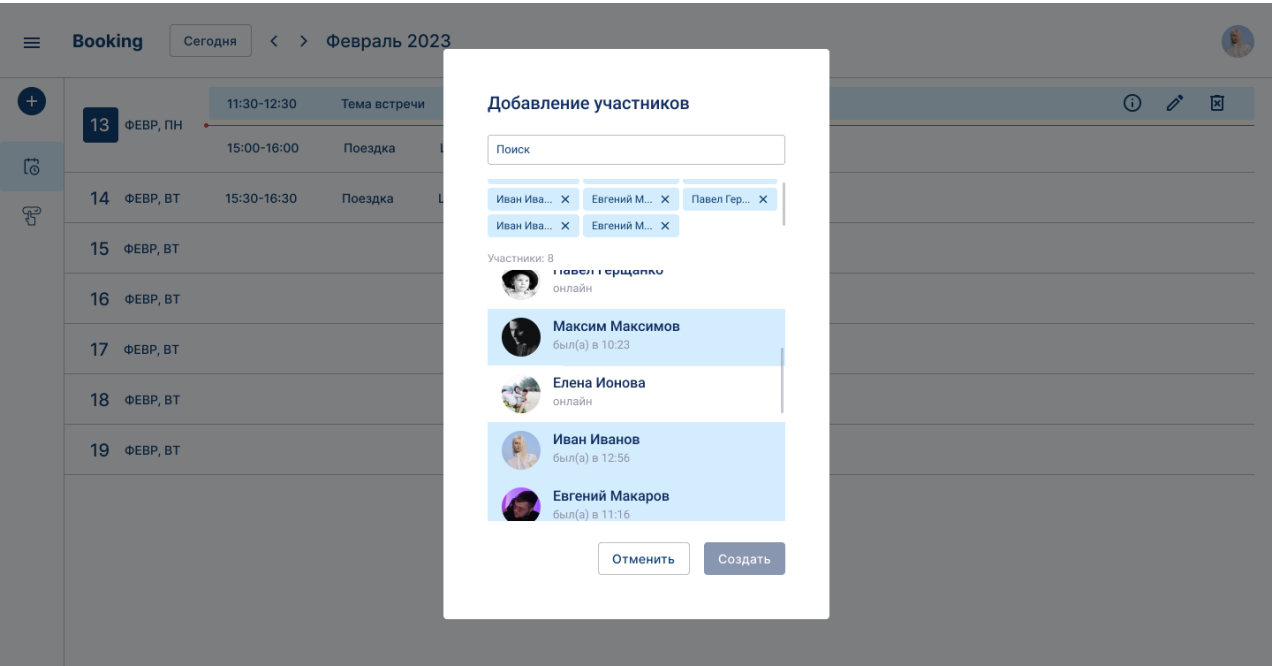


Рисунок 19 — Добавление участников бронирования

Нажав на кнопку «Создать» приложение показывает окно «Мои бронирования» с созданным бронированием (рисунок 20).

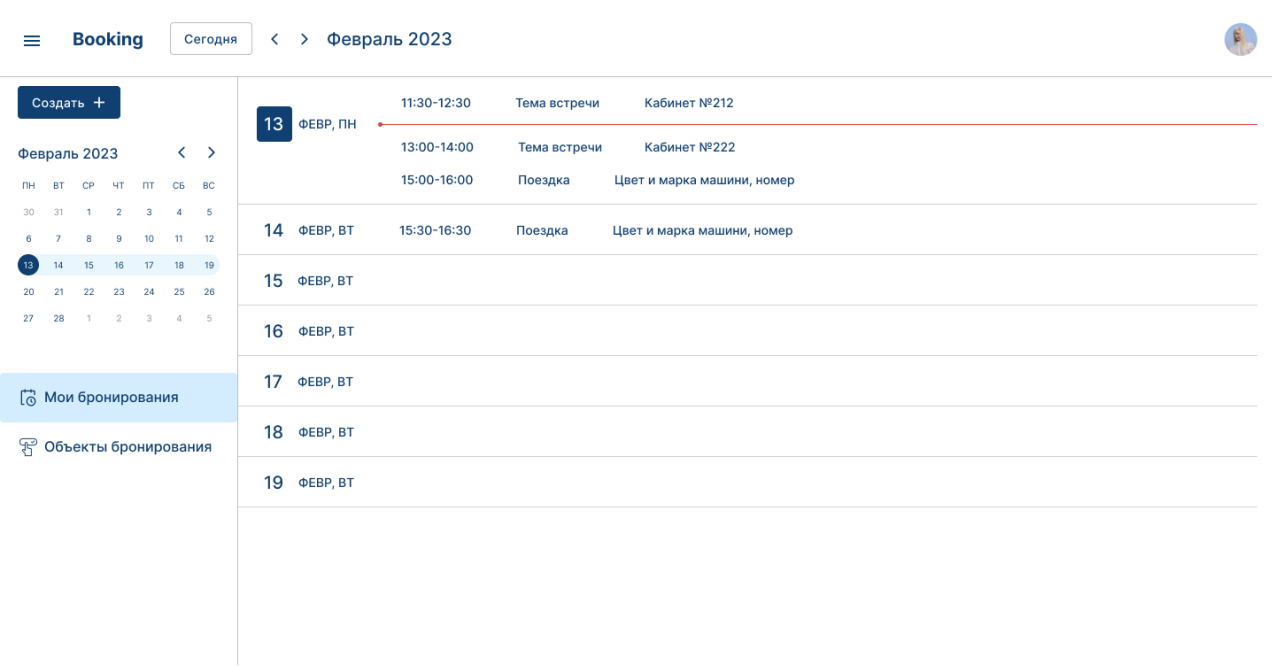


Рисунок 20 — Созданное бронирование в окне «Мои бронирования»

1.9.1.2 На мобильном устройстве

Создание бронирования на мобильном устройстве так же осуществляется разными способами: выбирая объект бронирования и нажимая на кнопку «Забронировать» в окне «Объекты бронирования», в окнах «Информация об объекте»и «Бронирования объекта» нажав на иконку меню и выбрав из пунктов «Забронировать».

Рассмотрим способ бронирования в окне «Объекты бронирования» (рисунок 21).

В окне «Объекты бронирования» находится список переговорных комнат, используя фильтры можно упростить поиск подходящего объекта, указав дату и точное время мероприятия, тип объекта, вместимость и статус (рисунок 22-23).

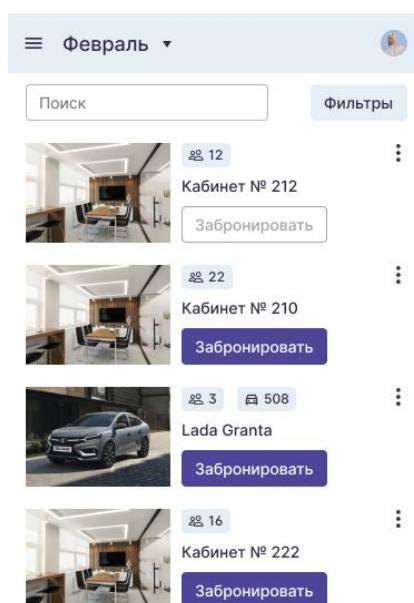


Рисунок 21 — Окно «Объекты бронирования»

← Фильтры

Февраль 2023

< >

ПН

ВТ

СР

ЧТ

ПТ

СБ

ВС

13

14

15

16

17

18

19

Время мероприятия

11:30

–

12:30

×

Тип объекта

☒ Переговорная

☐ Машина

Количество человек

До 12

×

Статус объекта

☒ Не выбрано

☐ Свободно

☐ Занято

Применить

Рисунок 22 — Фильтры

≡

Февраль

▼

Поиск

Фильтры

12

Кабинет № 212

Забронировать

16

Кабинет № 222

Забронировать

< 0

□

переговорная

12

i

своi

>

Рисунок 23 — Примененные фильтры

					ПГУ 1.090301.25.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

При нажатии на кнопку «Забронировать» открывается окно «Бронирование» (рисунок 24-25).

✕ Бронирование ✓

Тема встречи

Февраль 2023 < >

ПН	ВТ	СР	ЧТ	ПТ	СБ	ВС
13	14	15	16	17	18	19

Время мероприятия

11:30 – 12:30 ✕

Место проведения ✎

Кабинет № 222

Тип: переговорная

Адрес: ул. Кураева 12

Вместимость: 16 человек

Участники

👤 Добавить участника

🔍 Поиск по участникам

2 участника

👤 Пользователь

Организатор

👤 Пользователь

⋮

Рисунок 24 — Окно «Бронирование»

При нажатии на кнопку «Добавить участников» открывается окно «Добавление пользователей» (рисунок 25).

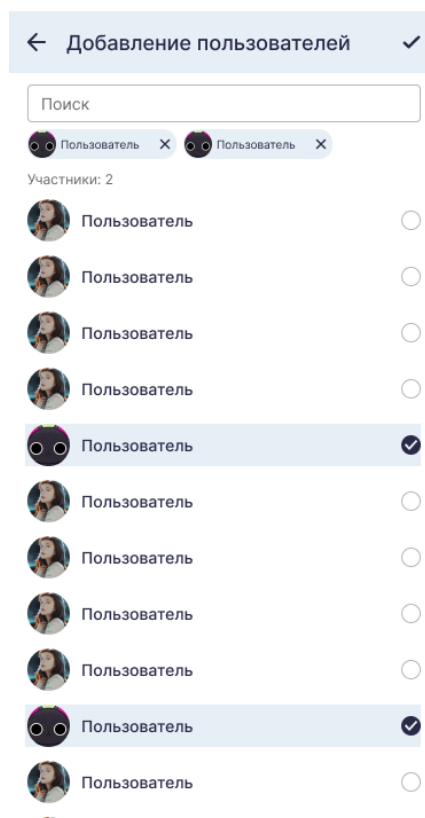


Рисунок 25 — Нажата кнопка «Добавить участников»

Нажав на иконку «Глалочка» в окне «Бронирование» создается бронирование.

1.9.2 Редактирование бронирования

1.9.2.1 На десктопном устройстве

Редактировать бронирование можно нажав на иконку «Карандаш» при наведении на бронирование в окне «Мои бронирования». Другой способ заключается в том, что в окне «Объекты бронирования» нужно нажать на свое бронирование на доске бронирований и выбрать из выпадающего меню пункт «Редактировать».

В окне редактирования можно выбрать что конкретно нужно изменить: основную информацию, объект бронирования, участников (рисунок 26-27). Окна редактирования информации будут идентичны окнам из создания бронирования.

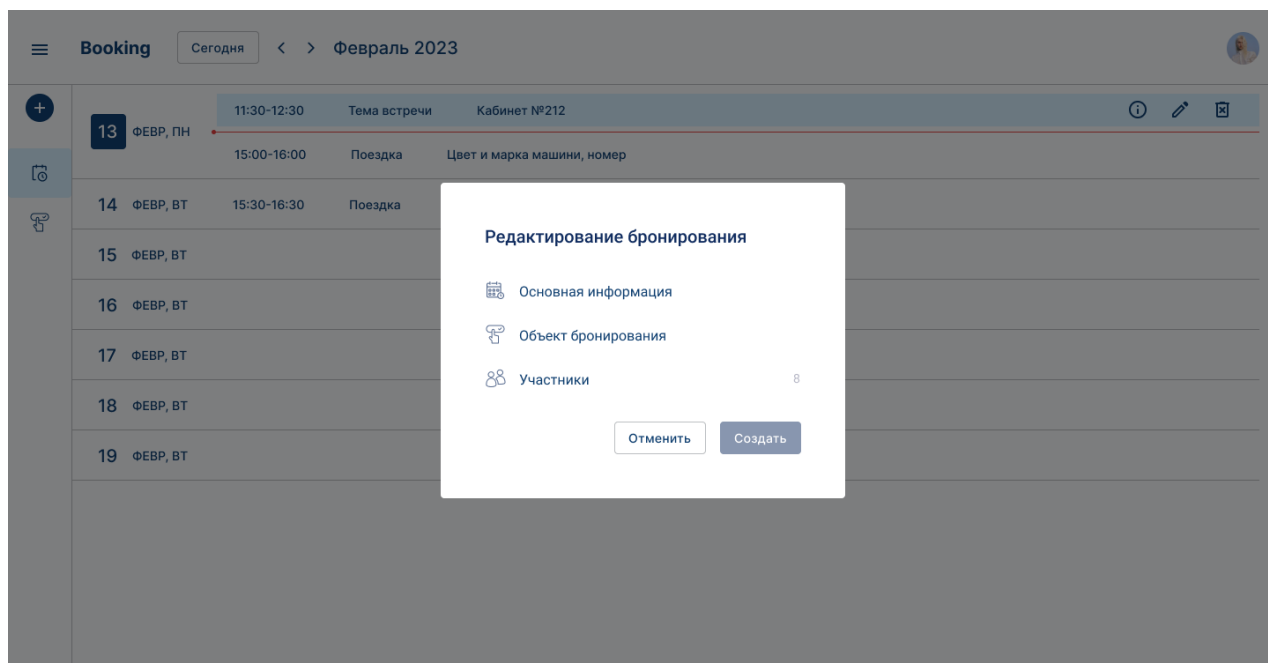


Рисунок 26 — Окно редактирования бронирования

1.9.2.2 На мобильном устройстве

×

Редактирования встречи

✓

Тема встречи

Февраль 2023

ПН

13

ВТ

14

СР

15

ЧТ

16

ПТ

17

СБ

18

ВС

19

Время мероприятия

11:30

—

12:30

×

Место проведения

Кабинет № 212

Тип: переговорная

Адрес: ул. Кураева 12

Вместимость: 12 человек

Участники

+

Добавить участника

Поиск по участникам

2 участника

Пользователь

Организатор

Пользователь

Рисунок 27 — Окно редактирования бронирования

1.9.3 Просмотр информации своего бронирования

1.9.3.1 На десктопном устройстве

Посмотреть подробную информацию бронирования можно нажав на иконку «Информация» при наведении на бронирование в окне «Мои бронирования», есть возможность редактировать (рисунок 28-32). Другой способ заключается в том, что в окне «Объекты бронирования» нужно нажать на свое бронирование на доске бронирований и выбрать из выпадающего меню пункт «Редактировать».

При просмотре информации о бронировании другого пользователя возможности редактирования нет.

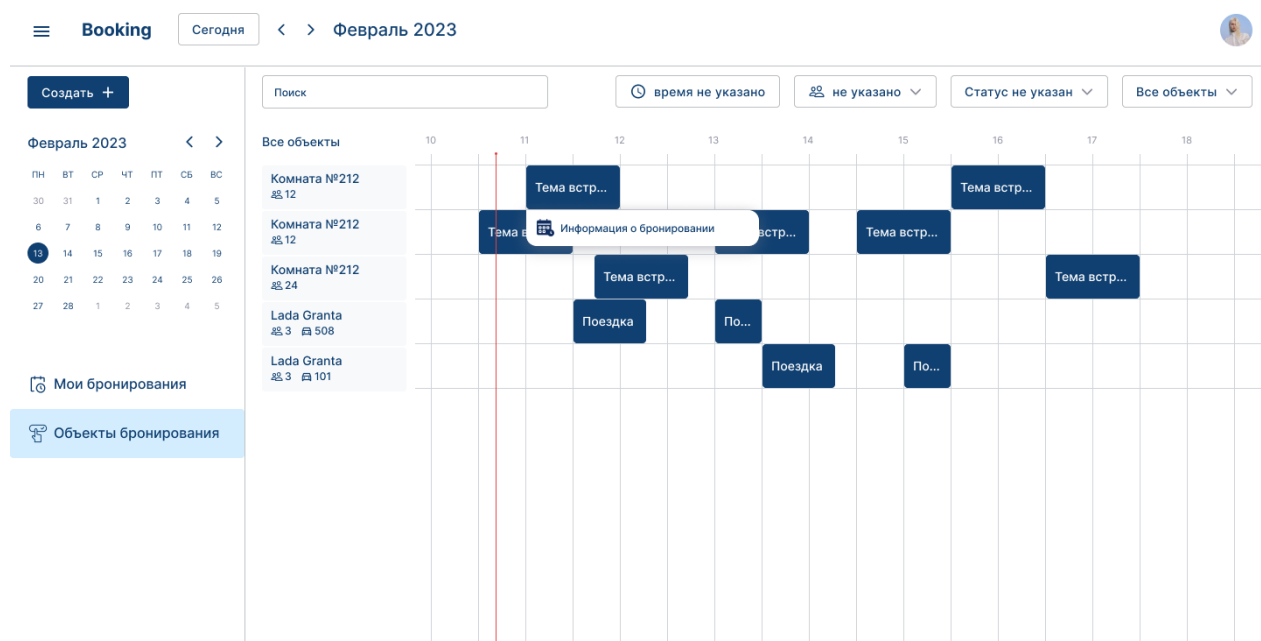


Рисунок 28 — Окно «Объекты бронирования» при нажатии на бронирование другого пользователя

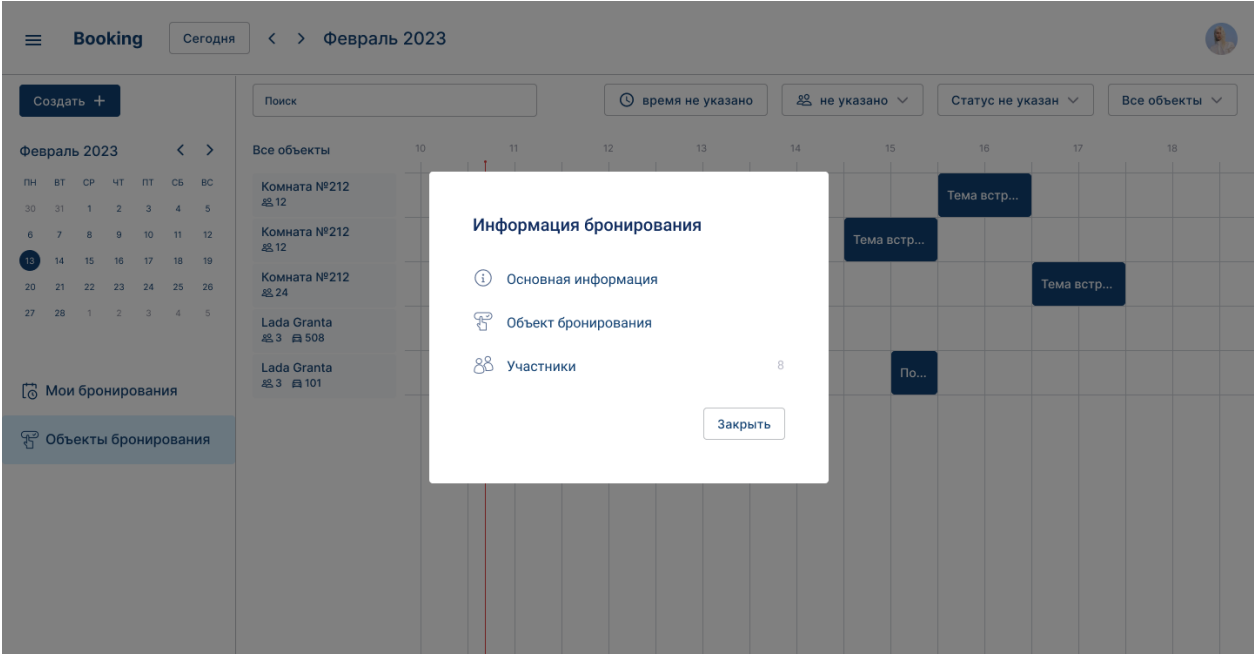


Рисунок 29 — Окно информации о бронировании другого пользователя

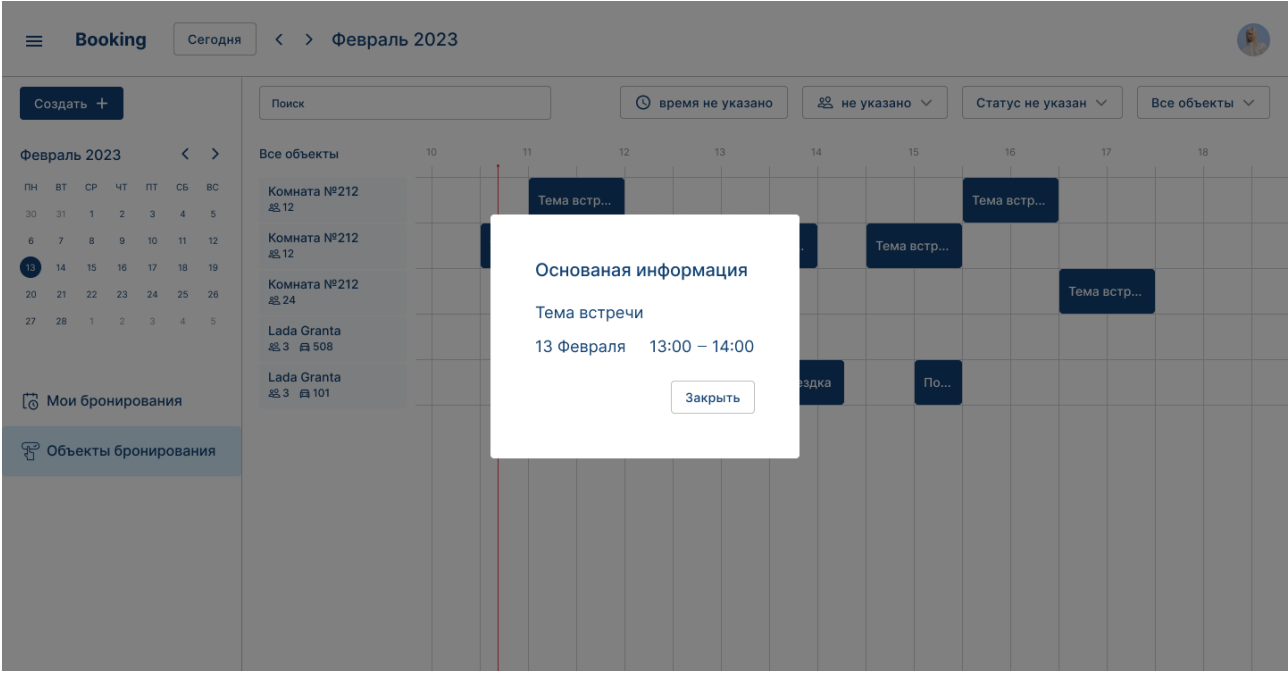


Рисунок 30 — Основная информация о бронировании другого пользователя

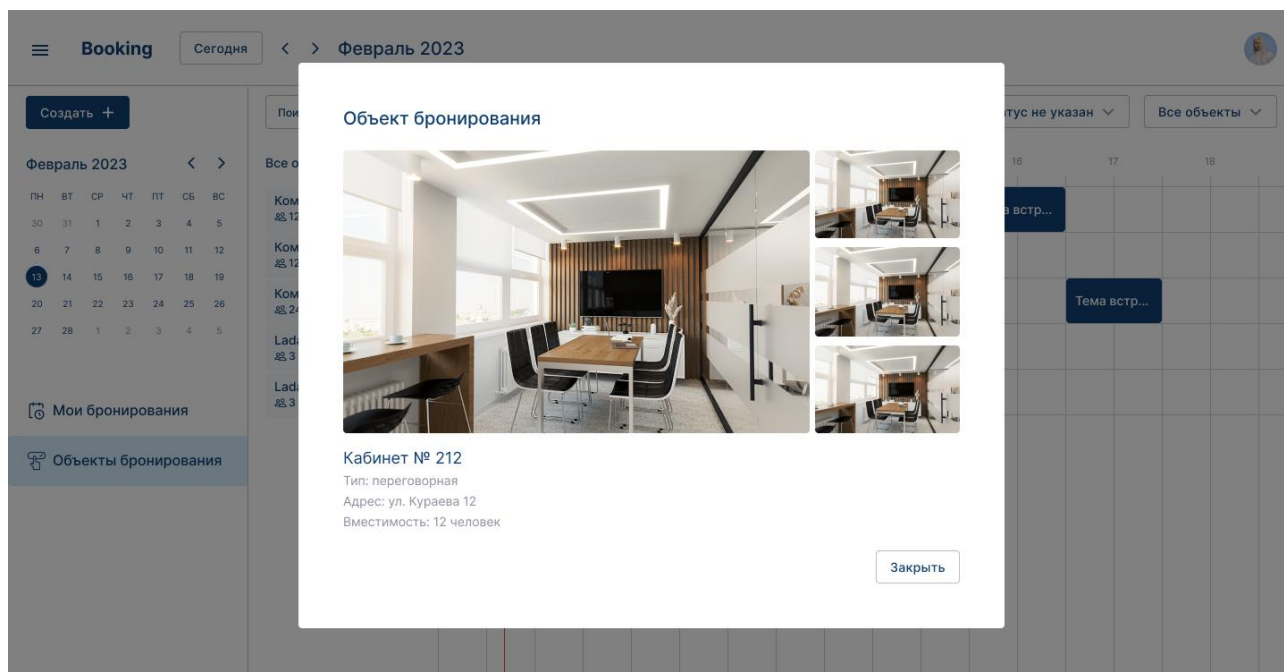


Рисунок 31 — Информация об объекте бронирования другого пользователя

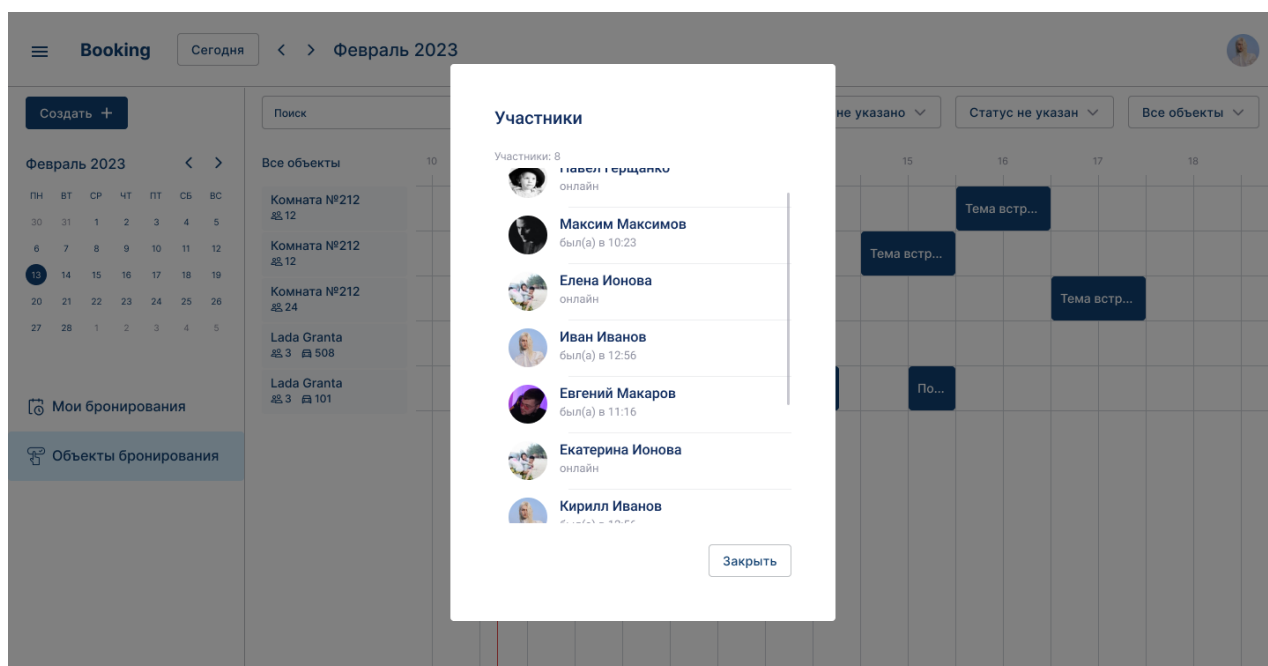


Рисунок 32 — Информация об участниках бронирования другого пользователя

1.9.3.2 На мобильном устройстве

Посмотреть подробную информацию бронирования можно нажав на

свое бронирование в окне «Мои бронирования», есть возможность редактирования при нажатии на иконку «Меню» и выбрав пункт в выпадающем меню «Редактировать» (рисунок 33). Также в окне «Объекты бронирования» нужно нажать на свое бронирование на доске бронирований и выбрать из выпадающего меню пункт «Бронирования объекта» и нажать на бронирование из списка (рисунок 34-35).

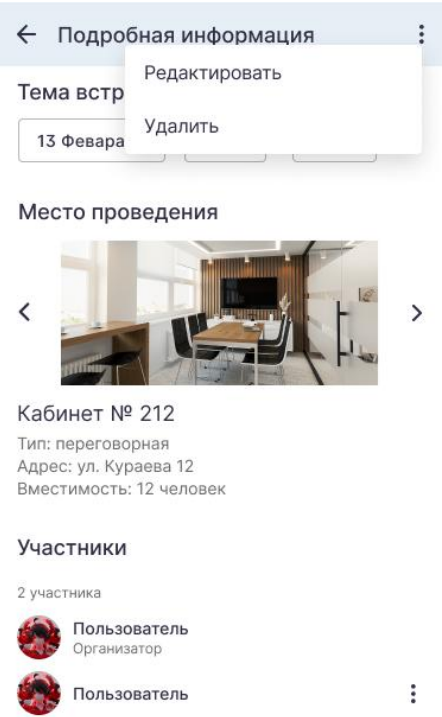


Рисунок 33 — Окно «Детальная информация»

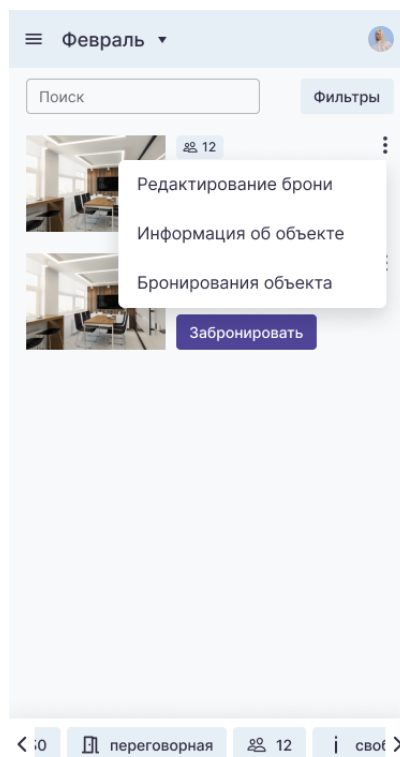


Рисунок 34 — Окно «Мои бронирования» при нажатии на иконку «Меню»

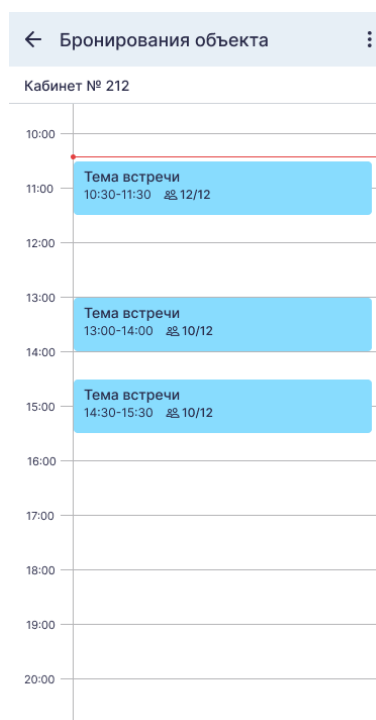


Рисунок 35 — Окно «Бронирования объекта»

1.9.4 Просмотр информации об объекте бронирования

Посмотреть информацию о переговорной комнате в десктопной версии можно в окне информации о бронировании (рисунок 36), окне объектов бронирования нажав на комнату в списке.

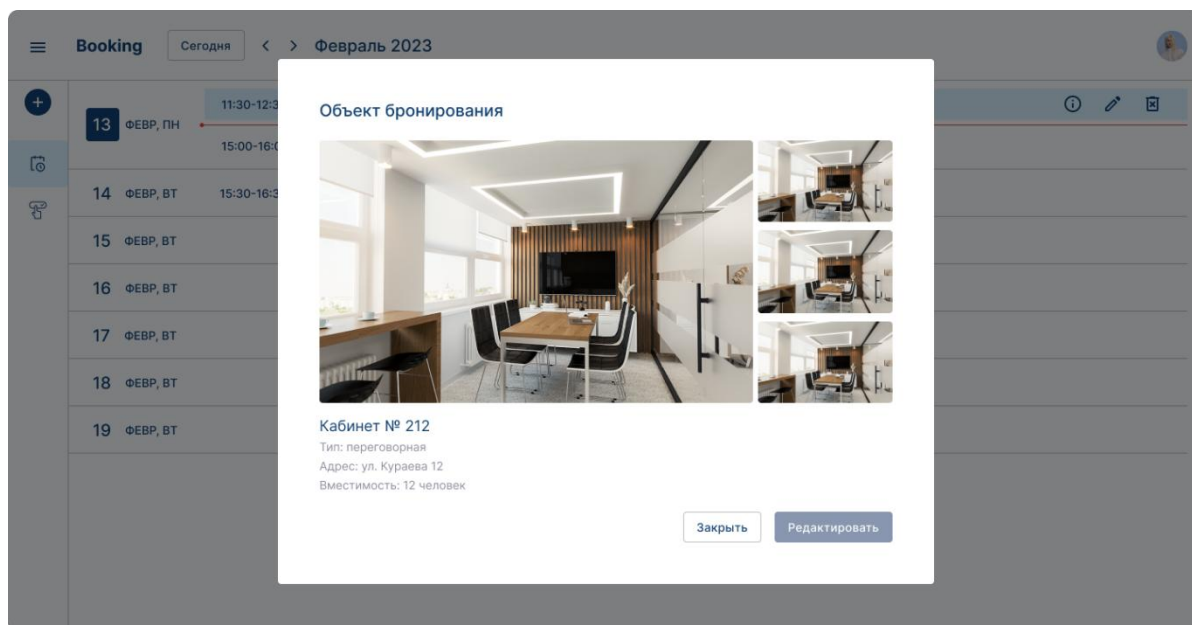


Рисунок 36 — Окно «Информация об объекте бронирования» из информации о своем бронировании

Для просмотра информации об объекте бронирования в мобильной версии нужно в окне «Объекты бронирования» нажать на иконку «Меню» объекта бронирования и нажать из выпадающего меню на пункт «Информация об объекте» (рисунок 37-38).

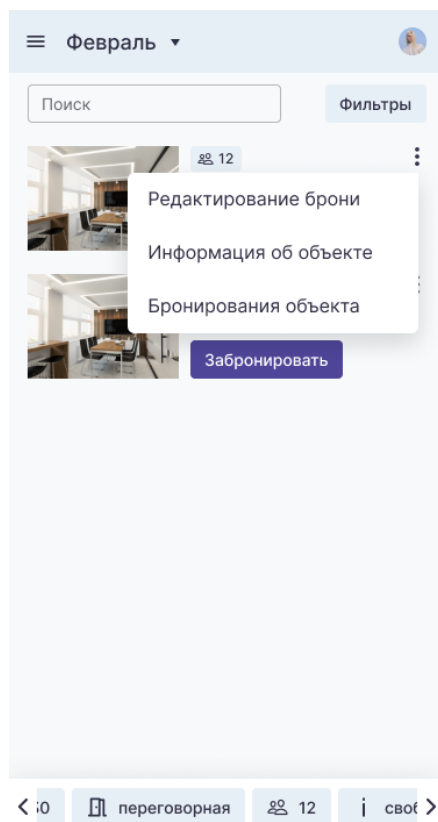


Рисунок 37 — Окно «Мои бронирования» при нажатии на иконку «Меню»



Рисунок 38 — Окно «Информация об объекте»

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.25.001 ПЗ

Лист
51

2 Охрана труда

В данном разделе излагаются условия, которые обеспечивают безопасность и сохранность здоровья для пользователей ПЭВМ, занятых созданием и управлением программного обеспечения, и подробно описываются способы уменьшения негативного влияния на организм.

2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с компьютерами

Работа с компьютерами является неотъемлемой частью повседневной жизни в условиях цифрового технологического прогресса. Однако, существует опасность возникновения негативных последствий для здоровья и общего благополучия человека. Рассмотрим подробнее некоторые потенциальные опасности и вредности, которые могут возникнуть при работе с компьютерами:

- компьютерный вид деятельности. При работе за компьютером значительную часть рабочего времени человек проводит в одной позиции, что создает однообразные и монотонные движения и ведет к проблемам со здоровьем, таким как различные заболевания позвоночника, мышечно-связочного аппарата, а также снижение работоспособности и общего состояния здоровья;

- защитные меры информационно-коммуникационных технологий. Работа с компьютером может быть опасной, когда применяются несертифицированные и нелегальные программы, технически некачественное и устаревшее оборудование либо проводится работа в условиях несоблюдения требований безопасности;

- стихийные бедствия. Аварии на электропроводах, серверных шкафах и других вторичных объектах информационно-коммуникационных технологий могут привести к чрезвычайным ситуациям, которые в свою

очередь могут представлять угрозу для здоровья человека;

- информационная безопасность. Утечка информации, хакерские атаки, злоупотребление персональными данными, а также другие угрозы и риски, связанные с некачественной работой информационных систем, могут привести к серьезным последствиям для здоровья, финансов и репутации человека.

2.2 Классификация и категорирование помещения, в котором производится разработка на компьютерах

Классификация и категорирование помещения, в котором производится разработка на компьютерах, имеет большое значение для обеспечения безопасности работников и сохранения эффективности работы в течение длительных периодов времени. Основными факторами, которые необходимо учитывать при определении классификации и категорирования помещения, являются:

- относительная площадь. Размер помещения является главным фактором, который влияет на классификацию и категорирование помещения для разработки на компьютерах. Исходя из этого фактора, можно получить ценный показатель грандиозности естественного света;

- естественное освещение. Наличие естественного освещения в помещении, обеспечивающего комфортные условия для работы на компьютере, также влияет на классификацию и категорирование помещения;

- искусственное освещение. Наличие и качество искусственного освещения в помещении являются факторами, которые влияют на эффективность работы с компьютером, и принимаются во внимание при определении категории помещения;

- уровень шума. Шум в помещении может негативно сказаться на производительности и концентрации работников, поэтому уровень шума имеет большое значение при определении категории помещения;

- условия вентиляции. Наличие вентиляционной системы, обеспечивающей безопасную среду для работы в помещении на компьютере, является фактором, важным при определении классификации и категорирования помещения.

2.3 Санитарно-гигиеническая характеристика

Санитарно-гигиеническая характеристика - это описание условий и окружающей среды, в которых осуществляется работа с компьютером, с целью обеспечения здоровья и безопасности работников. Основными элементами санитарно-гигиенической характеристики являются:

- качество воздуха. Воздух в помещении, где происходит работа на компьютере, должен быть достаточно чистым и свежим, поэтому необходимо обеспечивать достаточную вентиляцию и удалять излишки пыли и грязи;

- освещение. Освещение должно обеспечивать комфортные условия для работы на компьютере и должно соответствовать нормам и требованиям безопасности и гигиены;

- шум. Для работы на компьютере необходимо обеспечить условия приемлемого уровня шума, чтобы работники не испытывали сильную усталость и стресс;

- эргономика. Рабочее место для работы на компьютере должно быть оборудовано соответствующим образом, чтобы работники могли поддерживать правильную позу и избегать различных видов напряжения и монотонности;

- хранение и утилизация отходов. Отходы, связанные с использованием компьютеров, должны храниться и утилизироваться в соответствии с нормами и требованиями безопасности и гигиены;

- информационная безопасность. Необходимо соблюдать соответствующие меры по защите конфиденциальной и личной информации,

обрабатываемой на компьютере, чтобы исключить возможность утечки данных.

Кроме того, необходимо своевременно производить превентивные меры и регулярно проводить проверки для устранения любых негативных последствий, связанных со сбоями или ухудшением состояния системы.

					ПГУ 1.090301.25.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения данной работы была изучена методология проектирования веб-приложения с клиент-серверной архитектурой и приобретены навыки работы с фрейворком *Spring*. При реализации поставленной задачи были получены следующие результаты:

- разработана структура базы данных для СУБД *PostgreSQL*;
- разработан прикладной интерфейс для взаимодействия серверной и клиентской частей (*REST API*);
- разработана серверная часть приложения.

Список использованных источников

1. Автоматизация предприятий, <https://unspot.ru/meeting-room/>, дата обращения: 26.04.2023.
2. "Java 8 in Action" by Raoul-Gabriel Urma, Mario Fusco, and Alan Mycroft - издана в 2014 году издательством Manning Publications.
3. "Java Persistence with Hibernate" (2nd edition) by Christian Bauer and Gavin King - издана в 2016 году издательством Manning Publications.
4. "Spring Boot in Action" by Craig Walls - издана в 2020 году издательством Manning Publications.
5. "Java Web Services: Up and Running" by Martin Kalin, 2009, O'Reilly Media.
6. Spring Boot Documentation - <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>.
7. SQL Tutorial - <https://www.sqltutorial.org/>.
8. Официальная документация SQL - <https://dev.mysql.com/doc/> или <https://www.postgresql.org/docs/>.
9. "RESTful Web Services" by Leonard Richardson, Sam Ruby, and David Heinemeier Hansson - издана в 2007 году издательством O'Reilly Media.

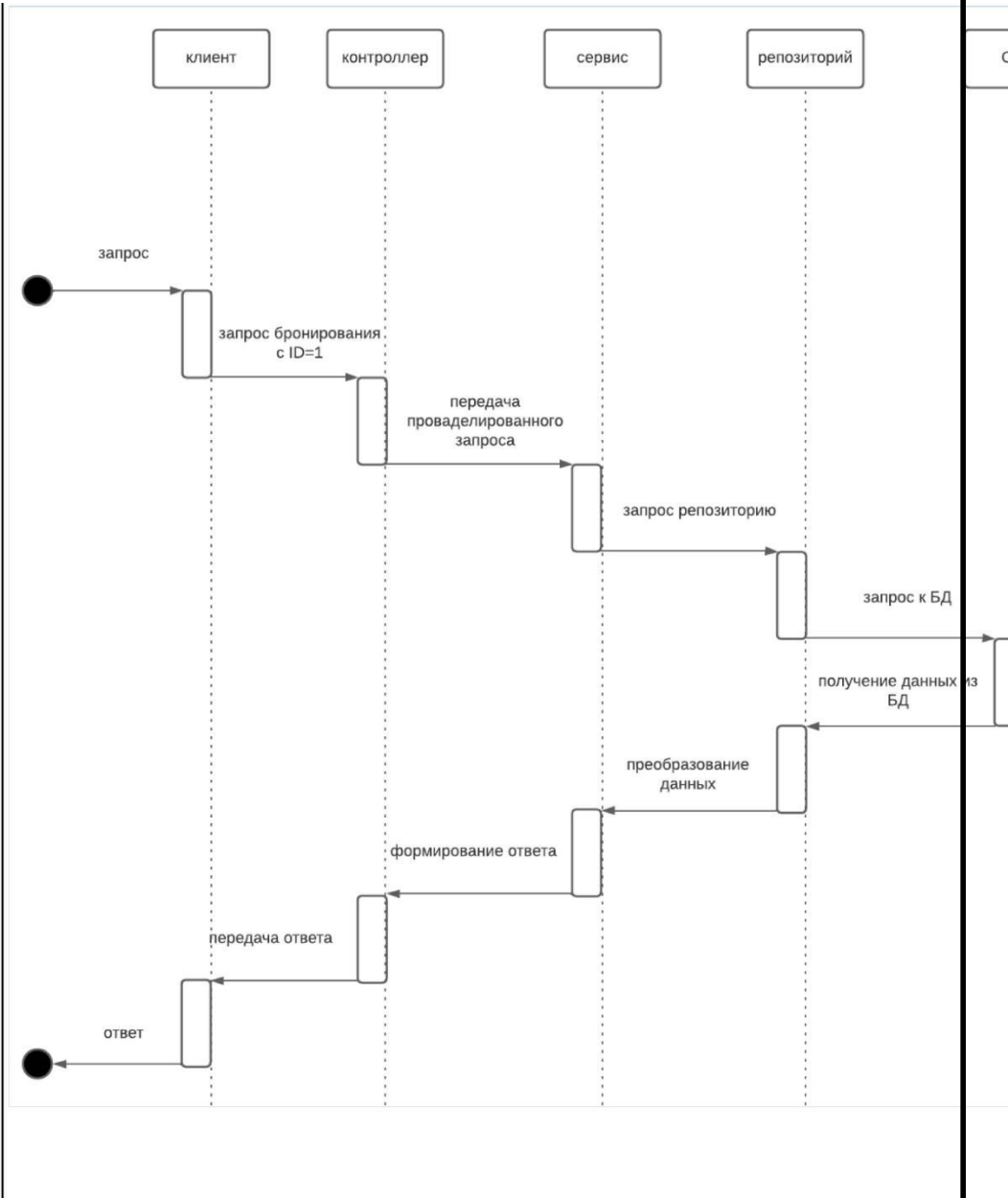
Приложение А

(Обязательное)

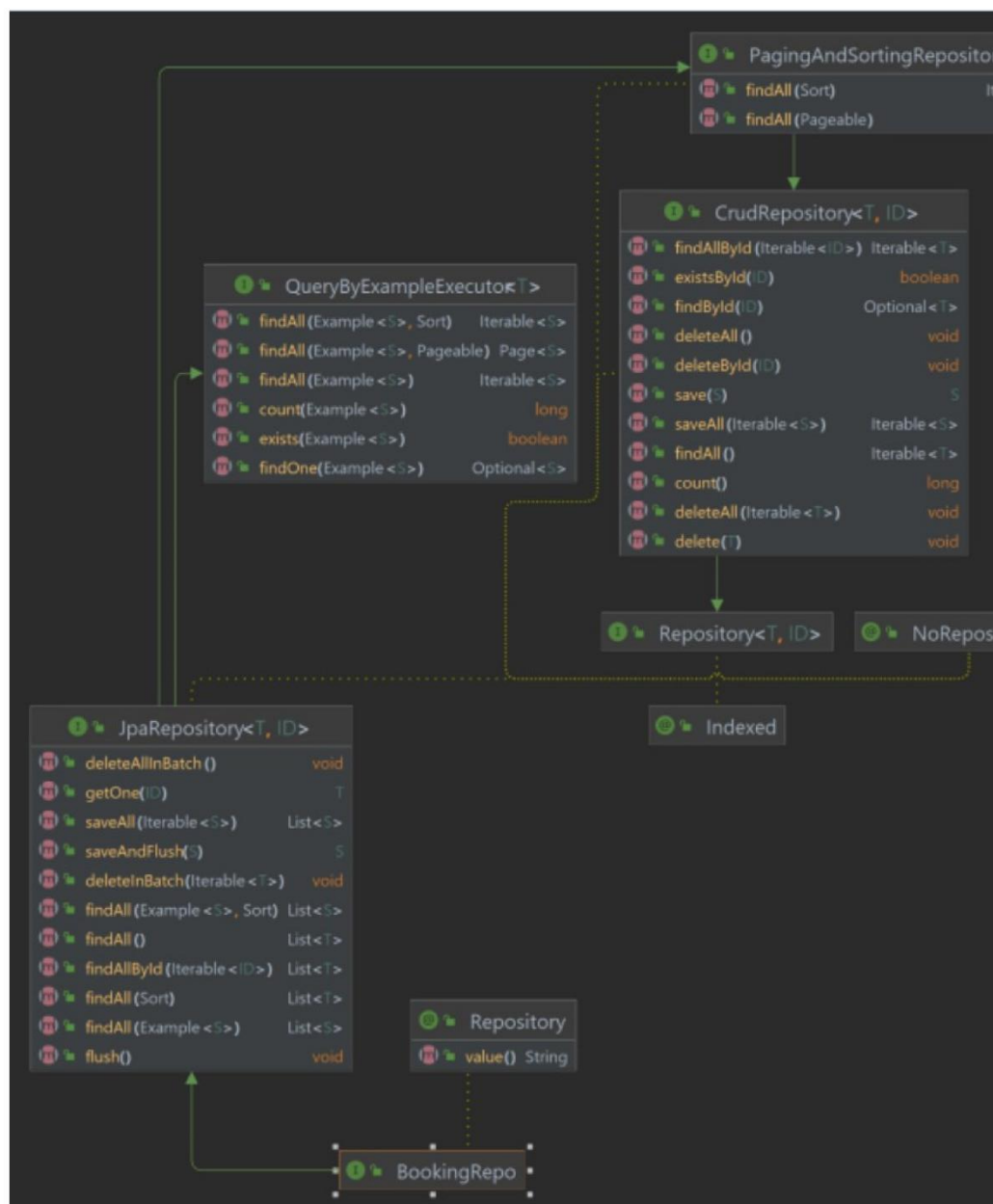
Подсистема бронирования переговорных комнат информационной системы
организации
UML-диаграммы

					ПГУ 1.090301.25.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 — Диаграмма последовательностей



Приложение А.2 — Диаграмма развертывания



Приложение А.4 — Диаграмма прецедентов

Приложение Б — Листинг программы на Java

ПГУ 1.090301.25.001 ПЗ

Приложение Б.1 — Файл «Booking.java»

```
package com.example.booking.core.booking;
import com.example.booking.core.invitation.Invitation;
import com.example.booking.core.subject.Subject;
import com.example.booking.core.user.User;
import org.hibernate.annotations.GenericGenerator;
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
```

@Entity

						Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Table(name = "fc_booking")
public class Booking {

    @Id
    @Column(name = "id")
    @GenericGenerator(
        name = "booking_id_seq",
        strategy
=
"org.hibernate.id.enhanced.SequenceStyleGenerator",
        parameters = {
            @org.hibernate.annotations.Parameter(name
=
"sequence_name", value = "booking_id_seq"),
            @org.hibernate.annotations.Parameter(name=
"INCREMENT", value = "1"),
            @org.hibernate.annotations.Parameter(name
=
"MINVALUE", value = "1"),
            @org.hibernate.annotations.Parameter(name
=
"MAXVALUE", value = "2147483647"),
            @org.hibernate.annotations.Parameter(name
=
"CACHE", value = "1")
        }
    )
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator
= "booking_id_seq")
    private long id;

    @Column(name = "begin_book_time")
    private LocalDateTime beginBookTime;

    @Column(name = "end_book_time")
    private LocalDateTime endBookTime;

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name = "id_user")
    private User user;

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name = "id_subject")
    private Subject subject;

    @OneToMany(mappedBy = "booking", cascade=CascadeType.ALL)
    private List<Invitation> participant =new ArrayList<>();
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public LocalDateTime getBeginBookTime() {
        return beginBookTime;
    }
    public void setBeginBookTime(LocalDateTime beginBookTime) {
        this.beginBookTime = beginBookTime;
    }

```

ПГУ 1.090301.25.001 ПЗ

						Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }
    public LocalDateTime getEndBookTime() {
        return endBookTime;
    }
    public void setEndBookTime(LocalDateTime endBookTime) {
        this.endBookTime = endBookTime;
    }
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
    public Subject getSubject() {
        return subject;
    }
    public void setSubject(Subject subject) {
        this.subject = subject;
    }
    public List<Invitation> getParticipant() {
        return participant;
    }
    public void setParticipant(List <Invitation> participant) {
        this.participant = participant;
    }
}

```

Приложение Б.2 — Файл «BookingController.java»

```

package com.example.booking.core.booking.web;

import com.example.booking.core.booking.Booking;
import com.example.booking.core.booking.BookingService;
import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

@RestController
@RequestMapping("/booking")
public class BookingController {

    private final BookingService service;
    public BookingController(BookingService service) {
        this.service = service;
    }

```

ПГУ 1-090301-25.001 ПЗ

```

    @GetMapping("/{id}")
    @ResponseBody
    public BookingView getBooking(@PathVariable Long id) {
        return service.getBooking(id);
    }

    @GetMapping
    @ResponseBody
    public Page<BookingView> getAllBooking(@PageableDefault(sort
= "id", direction = Sort.Direction.ASC) Pageable pageable) {
        return service.findAllBooking(pageable);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public BookingView create(@RequestBody @Valid BookingReq req)
{
        return service.create(req);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteBooking(@PathVariable Long id){
        service.delete(id);
    }

    @PutMapping("/{id}")
    public BookingView updateBooking(@PathVariable(name = "id")
Long id,
                                   @RequestBody @Valid BookingReq
req) {
        Booking booking = service.findBookingOrThrow(id);
        return service.update(booking, req);
    }
}

```

Приложение Б.3 — Файл «BookingRepo.java»

```

package com.example.booking.core.booking;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface BookingRepo extends JpaRepository<Booking, Long>
{
}

```

Приложение Б.4 — Файл «BookingService.java»

```

package com.example.booking.core.booking;

import
com.example.booking.core.booking.converter.BookingToBookingViewC
onverter;
import com.example.booking.core.booking.web.BookingView;
import com.example.booking.core.booking.web.BookingReq;
import com.example.booking.core.user.UserRepo;
import com.example.booking.core.subject.SubjectRepo;
import com.example.booking.error.EntityNotFoundException;
import com.example.booking.util.MessageUtil;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageImpl;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.List;
import java.util.stream.Collectors;

@Service
public class BookingService {

    private final BookingRepo bookingRepo;
    private final BookingToBookingViewConverter
bookingToBookingViewConverter;
    private final UserRepo userRepo;
    private final SubjectRepo subjectRepo;
    private final MessageUtil messageUtil;

    public BookingService(BookingRepo bookingRepo,
                          BookingToBookingViewConverter
bookingToBookingViewConverter,
                          UserRepo userRepo,
                          SubjectRepo subjectRepo,
                          MessageUtil messageUtil) {
        this.bookingRepo = bookingRepo;
        this.bookingToBookingViewConverter
bookingToBookingViewConverter;
        this.userRepo = userRepo;
        this.subjectRepo = subjectRepo;
        this.messageUtil = messageUtil;
    }

    public Booking findBookingOrThrow(Long id) {
        return bookingRepo.findById(id)
            .orElseThrow(() -> new
EntityNotFoundException(messageUtil.getMessage("booking.NotFound
", id)));
    }

    public BookingView getBooking(Long id) {

```

```

        Booking booking = findBookingOrThrow(id);
        return bookingToBookingViewConverter.convert(booking);
    }

    public BookingView create(BookingReq req) {
        Booking booking = new Booking();
        this.prepare(booking, req);
        Booking bookingSave = bookingRepo.save(booking);
        return bookingToBookingViewConverter.convert(bookingSave);
    }

    public Page<BookingView> findAllBooking(Pageable pageable) {
        Page <Booking> bookings = bookingRepo.findAll(pageable);
        List <BookingView> bookingViews = bookings.stream()
            .map(bookingToBookingViewConverter::convert)
            .collect(Collectors.toList());
        return new PageImpl<>(bookingViews, pageable,
bookings.getTotalElements());
    }

    @Transactional
    public void delete(Long id) {
        try {
            bookingRepo.deleteById(id);
        } catch (EmptyResultDataAccessException e) {
            throw new
EntityNotFoundException(messageUtil.getMessage("booking.NotFound
", id));
        }
    }

    public BookingView update(Booking booking, BookingReq req) {
        this.prepare(booking, req);
        booking = bookingRepo.save(booking);

        return bookingToBookingViewConverter.convert(booking);
    }

    private void prepare(Booking booking, BookingReq req) {
        booking.setBeginBookTime(req.getBeginBookTime());
        booking.setEndBookTime(req.getEndBookTime());
        booking.setUser(userRepo.findOne(req.getUserId()));

        booking.setSubject(subjectRepo.findOne(req.getSubjectId()));
        booking.setParticipant(booking.getParticipant());
    }
}

```

Приложение Б.5 — SQL-файл

```
CREATE DATABASE booking;
```

					ПГУ 1.090301.25.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

т. ADD CONSTRAINT tc category

Приложение Б.6 — Описание REST API

Приложение Б.6.1 — Получение бронирования

Метод: GET

URI: /booking

Тело запроса:

Ответ:

200

OK

```
{
  "content": [
    {
      "bookingId": "1711i",
      "subject": {
        "subjectId": "1704b",
        "fullName": "Leather chair in the room №110",
        "shortName": "Boss chair №110",
        "maxPeople": 70,
        "categoryId": "1705c",
        "scheduleId": "1706d"
      },
      "beginBookTime": "2022-02-02 20:00",
      "endBookTime": "2022-02-02 21:00",
      "responserId": "1713k",
      "responserFio": "Ivanov Ivan Ivanovich",
      "invitedUsers": [
        {
          "invitationId": "1714l",
          "participant": {
            "userId": "1710h",
            "fio": "Ivanov Ivan Ivanovich",
            "role": 0
          }
        }
      ],
      "bookingId": "1716n",
      "isAccepted": true
    }
  ],
  "first": true,
  "last": true,
  "number": 0,
  "numberOfElements": 0,
  "size": 0,
  "totalElements": 0,
  "totalPages": 0
}
```

400

Bad Request

```
{ "message": "string" }
```

401

Not authorized

```
{ "message": "string" }
```

ПГУ 1.090301.25.001 ПЗ

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.25.001 ПЗ

Лист

70

```
403      Forbidden
        { "message": "string" }
500      Server error
        { "message": "string" }
```

Приложение Б.6.2 — Создание бронирования.

Метод: POST

URI: /booking

Тело запроса:

```
{
  "subject": {
    "subjectId": "1704b",
    "fullName": "Leather chair in the room №110",
    "shortName": "Boss chair №110",
    "maxPeople": 70,
    "categoryId": "1705c",
    "scheduleId": "1706d"
  },
  "beginBookTime": "2022-02-02 20:00",
  "endBookTime": "2022-02-02 21:00",
  "responserId": "1713k",
  "responserFio": "Ivanov Ivan Ivanovich",
  "invitedUsers": [
    {
      "invitationId": "1714l",
      "participant": {
        "userId": "1710h",
        "fio": "Ivanov Ivan Ivanovich",
        "role": 0
      },
      "bookingId": "1716n",
      "isAccepted": true
    }
  ]
}
```

Ответ:

```
200      OK
        {
          "bookingId": "1711i",
          "subject": {
            "subjectId": "1704b",
            "fullName": "Leather chair in the room №110",
            "shortName": "Boss chair №110",
            "maxPeople": 70,
            "categoryId": "1705c",
            "scheduleId": "1706d"
          },
          "beginBookTime": "2022-02-02 20:00",
          "endBookTime": "2022-02-02 21:00",
          "responserId": "1713k",
          "responserFio": "Ivanov Ivan Ivanovich",

```

ПГУ 1.090301.25.001 ПЗ

						Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

        "invitedUsers": [
        {
        "invitationId": "1714l",
        "participant": {
        "userId": "1710h",
        "fio": "Ivanov Ivan Ivanovich",          "role": 0
        },
        "bookingId": "1716n",
        "isAccepted": true
        }
        ]
    }
400    Bad Request
    { "message": "string" }
401    Not authorized
    { "message": "string" }
403    Forbidden
    { "message": "string" }
500    Server error
    { "message": "string" }

```

Приложение Б.6.3 — Получение информации о бронировании

Метод: GET

URI: /booking/{bookingId}

Тело запроса:

Ответ:

```

200    OK
    {
    "bookingId": "1711i",
    "subject": {
    "subjectId": "1704b",
    "fullName": "Leather chair in the room №110",
    "shortName": "Boss chair №110",          "maxPeople":
    70,
    "categoryId": "1705c",
    "scheduleId": "1706d"
    },
    "beginBookTime":          "2022-02-02          20:00",
    "endBookTime":          "2022-02-02          21:00",
    "responserId": "1713k",
    "responserFio":          "Ivanov          Ivan          Ivanovich",
    "invitedUsers": [
    {
    "invitationId": "1714l",
    "participant": {
    "userId": "1710h",
    "fio": "Ivanov Ivan Ivanovich",          "role": 0
    },
    "bookingId": "1716n",
    "isAccepted": true
    }
    ]
    }

```

ПГУ, 1.090301.25.001 ПЗ

```

    ]
  }
400    Bad Request
      { "message": "string" }
401    Not authorized
      { "message": "string" }
403    Forbidden
      { "message": "string" }
500    Server error
      { "message": "string" }

```

Приложение Б.6.4 — Удаление бронирования

Метод: DELETE

URI: /booking/{bookingId}

Тело запроса:

Ответ:

```

200    OK
400    Bad Request
      { "message": "string" }
401    Not authorized
      { "message": "string" }
403    Forbidden
      { "message": "string" }
500    Server error
      { "message": "string" }

```

Приложение Б.6.5 — Редактирование бронирования

Метод: PUT

URI: /booking/{bookingId}

Тело запроса:

```

{
  "subject": {
    "subjectId": "1704b",
    "fullName": "Leather chair in the room №110",
    "shortName": "Boss chair №110",
    "maxPeople": 70,
    "categoryId": "1705c",
    "scheduleId": "1706d"
  },
  "beginBookTime": "2022-02-02 20:00",
  "endBookTime": "2022-02-02 21:00",
  "responserId": "1713k",
  "responserFio": "Ivanov Ivan Ivanovich",
  "invitedUsers": [
    {
      "invitationId": "1714l",
      "participant": {
        "userId": "1710h",
        "fio": "Ivanov Ivan Ivanovich"
      }
    }
  ]
}

```

ПГУ 1.090301.25.001 ПЗ

ПГУ 1.090301.25.001 ПЗ

ПГУ 1.090301.25.001 ПЗ

Изм.	Лист	№ докум.	Подпись	Дата

Лист

73

```

        "role": 0
    },
    "bookingId": "1716n",
    "isAccepted": true
}
]
}
Ответ:
200      OK
        {
            "bookingId": "1711i",
            "subject": {
                "subjectId": "1704b",
                "fullName": "Leather chair in the room №110",
                "shortName": "Boss chair №110",
                "maxPeople": 70,
                "categoryId": "1705c",
                "scheduleId": "1706d"
            },
            "beginBookTime": "2022-02-02 20:00",
            "endBookTime": "2022-02-02 21:00",
            "responserId": "1713k",
            "responserFio": "Ivanov Ivan Ivanovich",
            "invitedUsers": [
                {
                    "invitationId": "1714l",
                    "participant": {
                        "userId": "1710h",
                        "fio": "Ivanov Ivan Ivanovich",
                        "role": 0
                    },
                    "bookingId": "1716n",
                    "isAccepted": true
                }
            ]
        }
400      Bad Request
        { "message": "string" }
401      Not authorized
        { "message": "string" }
403      Forbidden
        { "message": "string" }
500      Server error
        { "message": "string" }

```

ПГУ 1.090301.25.001 ПЗ

Приложение В

(Обязательное)

Подсистема бронирования переговорных комнат информационной системы
организации

Презентация

Подсистема бронирования переговорных комнат информационной системы организации

Автор работы: студентка группы 19ВВП1 Ходакова Д. А.

Научный руководитель: к.т.н., Синев М. П.

ПГУ 1.090301.25.001 ПЗ

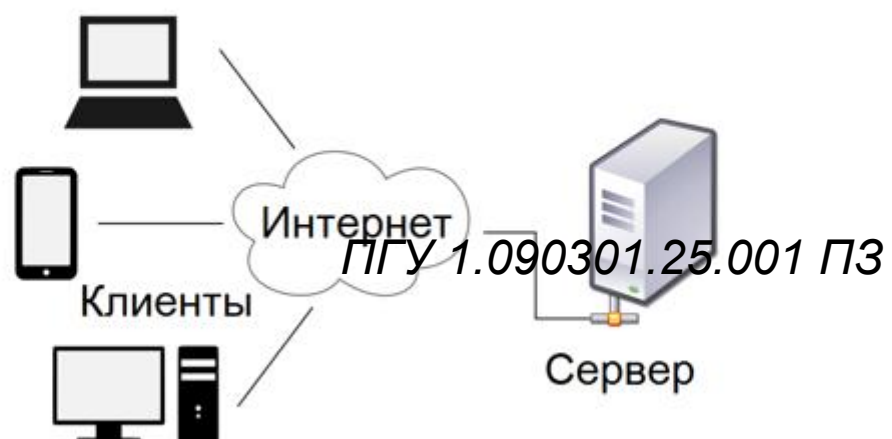
						Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

Цель и объект работы

Цель работы: разработать веб-приложение бронирования переговорных комнат

Объектом разработки является приложение, обеспечивающее выполнение следующих функций: создание, редактирование, удаление, просмотр информации бронирования переговорных комнат. Хранение информации о переговорных комнатах, бронированиях, пользователях должно осуществляться в БД. Также необходимо реализовать API для взаимодействия серверной и клиентской частями.

Клиент-серверная архитектура



Стек технологий

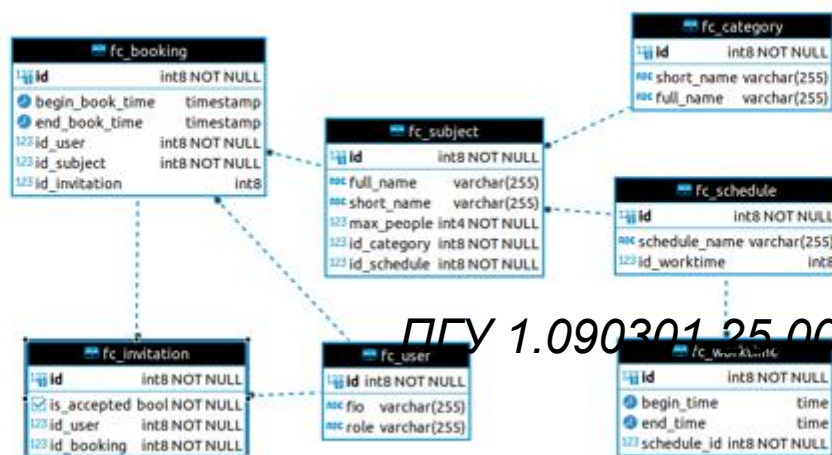


PostgreSQL



REST API

Структура БД

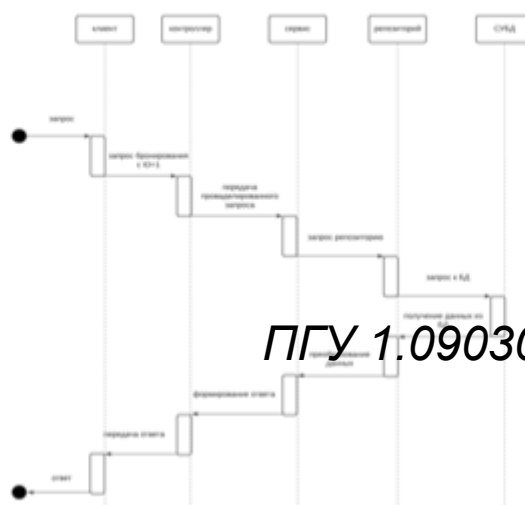


ПГУ 1.090301-25-001 ПЗ

Диаграмма класса-репозитория «Бронирования»

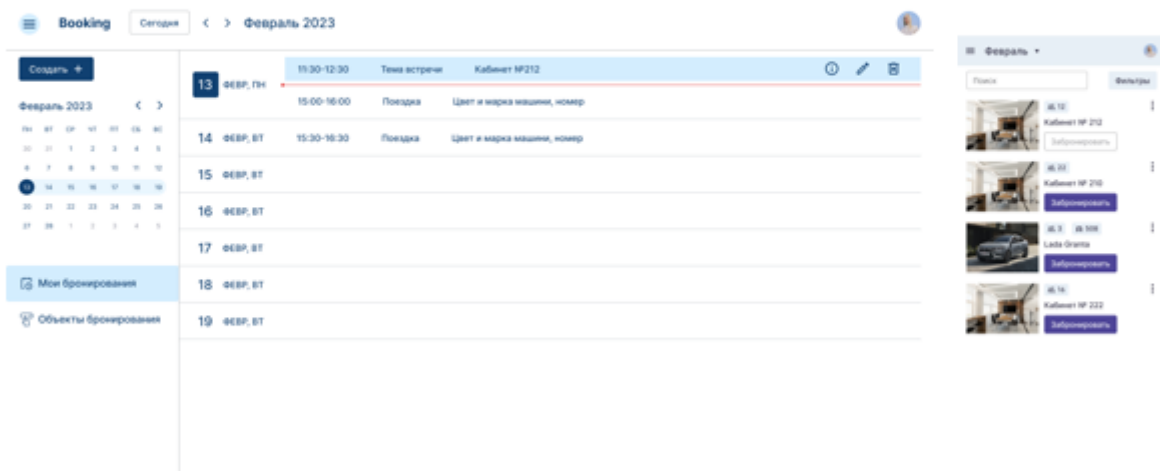


Последовательность взаимодействия классов при обработке запроса «Получение бронирования по ID»



ПГУ 1.090301.25.001 ПЗ

Результат разработки



Заключение

- разработана структура базы данных для СУБД *PostgreSQL*
- спроектирована архитектура приложения
- разработан прикладной интерфейс для взаимодействия с микросервисами, входящими в состав веб-приложения, согласно архитектурному стилю REST API

ПГУ 1.090301.25.001 ПЗ

Благодарю за внимание!

ПГУ 1.090301.25.001 ПЗ

						Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		