УДК 004.42:378

Чашников Павел Дмитриевич

студент, Государственный университет «Дубна», Россия, Дубна

Кушнарева Марина Игоревна

студент, Государственный университет «Дубна», Россия, Дубна

РАЗРАБОТКА ВЕБ-СЕРВИСА БРОНИРОВАНИЯ АУДИТОРИЙ ДЛЯ ВНЕУЧЕБНЫХ МЕРОПРИЯТИЙ УНИВЕРСИТЕТА

Аннотация. В статье рассматривается разработка веб-сервиса бронирования аудиторий для внеучебных мероприятий университета. Актуальность обусловлена разрозненностью данных о занятости помещений, риском двойного бронирования и недостаточной прозрачностью согласования заявок. Проведён анализ предметной области, разработаны ролевая модель, логика обработки заявок, архитектура сервера, модель данных и интерфейс клиентского приложения. Практический результат — программное решение с календарём мероприятий, подбором аудиторий, оформлением и рассмотрением заявок, интеграцией с внешними источниками данных.

Ключевые слова: бронирование аудиторий; внеучебные мероприятия; веб-сервис; информационная система; REST API; клиент-серверная архитектура; модель данных; университет.

Abstract. This article examines the development of a web service for reserving classrooms for extracurricular university events. The relevance is due to fragmented data on room occupancy, the risk of double bookings, and insufficient transparency in request approval. The domain has been analyzed, and a role model, request-processing logic, server-side architecture, a data model, and the client application interface have been developed. The practical result is a software solution combining an events calendar,

classroom selection, request submission and review, and integration with external data sources.

Keywords: classroom reservation; extracurricular events; web service; information system; REST API; client-server architecture; data model; university.

Введение

В современной университетской среде аудиторный фонд используется не только для проведения учебных занятий: в аудиториях проходят конференции, открытые лекции, семинары, дополнительные занятия, встречи со специалистами, мероприятия студенческих объединений и проектные активности. Для каждого такого события необходимо подобрать помещение: оценить вместимость и оснащённость, учесть расположение корпуса и проверить отсутствие пересечений с учебным расписанием или уже подтверждёнными заявками [1].

При ручной организации бронирования возникают типовые управленческие ограничения. Сведения о занятости помещений могут находиться в разных источниках, а процесс согласования зависит от переписки, локальных таблиц и участия ответственных сотрудников. В результате увеличивается время обработки заявки, повышается вероятность ошибки и усложняется контроль изменений. Особенно критичным является риск двойного бронирования, когда одно помещение оказывается назначенным на несколько мероприятий в один и тот же временной интервал. Автоматизация распределения аудиторного фонда рассматривается как элемент электронной образовательной среды вуза [2].

Целью работы является разработка веб-сервиса бронирования аудиторий для внеучебных мероприятий Государственного университета «Дубна». В статье обобщены результаты разработки серверной и клиентской частей сервиса, выполненной в рамках связанных выпускных квалификационных работ. Основное внимание уделяется тем аспектам, которые определяют надежность системы: формализации требований [2], разграничению доступа, проверке занятости

аудиторий, структуре данных и согласованному взаимодействию пользовательского интерфейса с серверным API.

Анализ предметной области

Предметная область сервиса включает ряд взаимосвязанных сущностей: аудитории, корпуса, типы помещений, оборудование, пользователей, роли, мероприятия, документы-основания (приказы, служебные записки и пр.), заявки на бронирование, учебное расписание и историю изменения статусов. Выделение сущностей и их связей соответствует общим принципам проектирования информационных систем [2]. Аудитория в такой системе является не только физическим помещением, но и ресурсом, доступность которого зависит от календаря занятий, заявок на мероприятия и решений ответственных сотрудников.

Текущий процесс бронирования выполняется вручную и представлен на диаграмме деятельности в нотации UML (см. рис. 1) [3]. Инициатор мероприятия подбирает аудиторию, сверяясь с расписанием и данными «1С:Документооборот», но окончательная проверка занятости требует устных уточнений с ответственным сотрудником — единого интерфейса для автоматической проверки доступности не существует. При конфликте инициатор вручную ищет другой вариант; после согласования аудитория фиксируется в служебной записке, приказе или «1С:Документооборот», а участники информируются вручную.

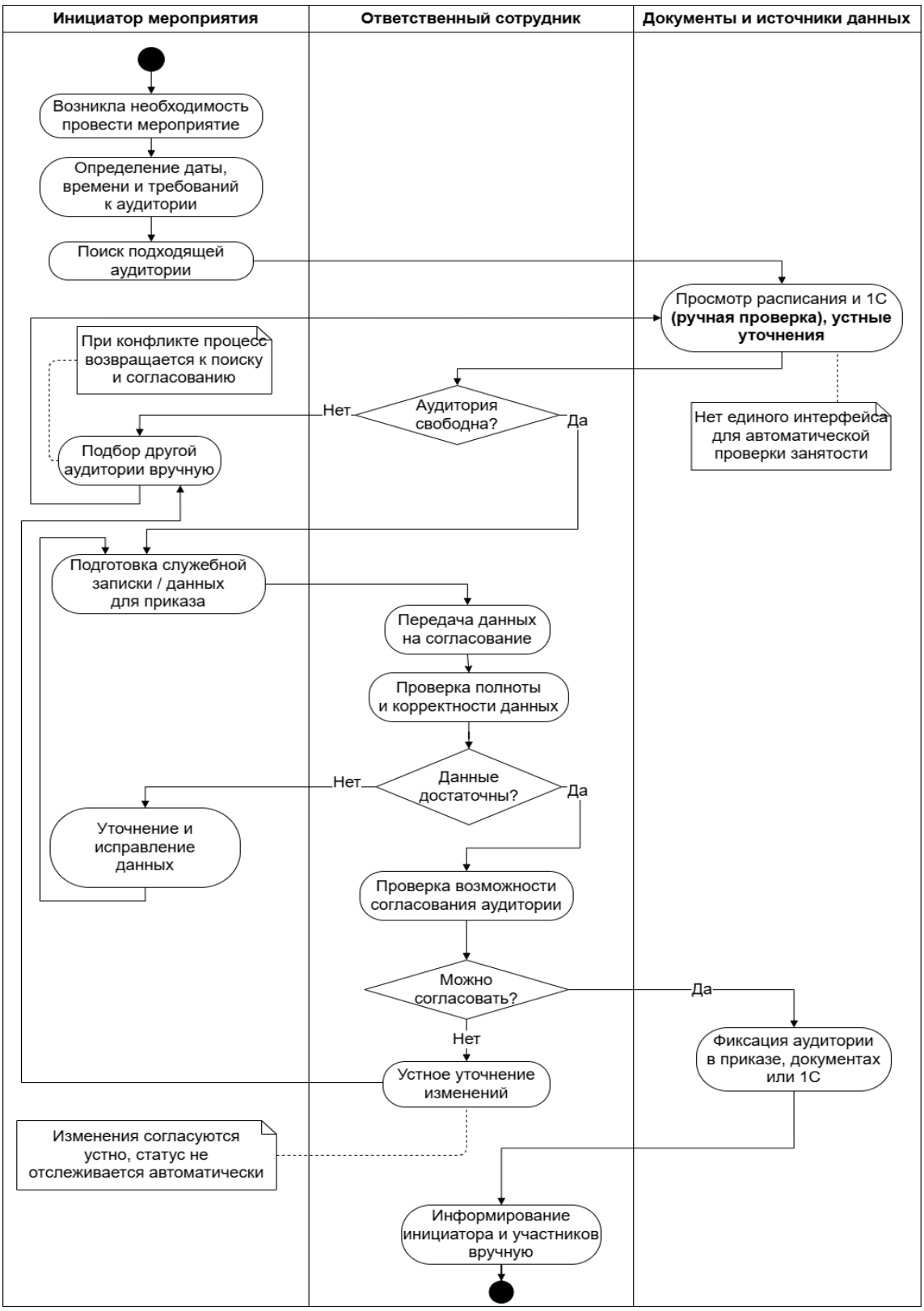

Рисунок 1. Диаграмма деятельности процесса бронирования аудиторий

Диаграмма показывает, что бронирование в текущем виде не формализовано: часть шагов требует личного взаимодействия сотрудников, изменения согласуются

устно, а статус заявки не отслеживается автоматически. Это создаёт риск двойного бронирования и увеличивает время обработки обращений [1].

Смежные задачи в университете частично закрывают отдельные системы: «1С:Документооборот» (административные документы, расписание) и Indico (календарь научных мероприятий), но ни одна из них не решает задачу бронирования аудиторий в полном объёме. Ранее в Государственном университете «Дубна» рассматривались задачи планирования научных мероприятий [4] и управления расписанием мероприятий [5], однако они не связывали между собой расписание, заявки и решения ответственных сотрудников.

Таким образом, необходим отдельный сервис, автоматизирующий не только создание записи о бронировании, но и контроль состояний заявки, проверку занятости аудиторий и уведомление участников.

Существующие решения

Для уточнения требований был выполнен обзор подходов к бронированию аудиторий в российских вузах [1]. Рассмотренные решения различаются степенью автоматизации: в Томском государственном университете бронирование регламентировано пошаговой инструкцией, но подбор помещения, подготовка документов и согласование выполняются вручную; в Московском физико-техническом институте реализован каталог аудиторий с фильтрацией по корпусу, этажу, типу и техническому оснащению; в Санкт-Петербургском государственном университете аэрокосмического приборостроения заявка подаётся через электронную форму, которая не связана с календарём в реальном времени и не проверяет пересечения, поэтому может быть отправлена на уже занятое помещение.

Отдельно рассматривались работы, выполненные ранее в Государственном университете «Дубна»: подсистема планирования научных мероприятий [4] и веб-приложение для управления расписанием мероприятий [5]. Они охватывают смежные задачи планирования и публикации событий, но не реализуют полный жизненный цикл заявки на аудиторию. Таким образом, в рассмотренных решениях

автоматизирован либо поиск помещения, либо приём заявки, но не их совместная проверка: именно отсутствие связи между подбором аудитории, учебным расписанием и решением ответственного сотрудника определило состав требований к разрабатываемому сервису.

Требования к системе и ролевая модель

Требования к сервису были сформулированы с учётом групп пользователей: внешних посетителей, которым доступны сведения об аудиториях и мероприятиях, и авторизованных участников, выполняющих операции бронирования [2]. Такой подход позволяет совместить открытость календаря внеучебной деятельности с контролем доступа к созданию, согласованию и администрированию заявок.

Сервис должен обеспечивать просмотр календаря мероприятий, поиск аудитории по параметрам, просмотр характеристик помещения, создание заявки на бронирование, обработку заявки ответственным сотрудником, работу с мероприятиями, формирование отчетов и отображение истории изменений. Сводное распределение по ролям приведено в таблице 1. Нефункциональные требования связаны с надежностью проверки доступности, защитой персональных данных, предсказуемостью API и удобством интерфейса для разных ролей [2].

Таблица 1.

Доступность функционала пользователям

Функция	Неавторизованный пользователь	Преподаватель	Ответственный сотрудник	Администратор
Просмотр календаря мероприятий	+	+	+	+
Просмотр сведений об аудитории	+	+	+	+
Поиск и фильтрация аудиторий	+	+	+	+
Выгрузка списка предстоящих мероприятий	+	+	+	+
Создание заявки на бронирование	–	+	+	+
Просмотр собственных бронирований	–	+	+	+
Отмена собственного бронирования	–	+	+	+
Просмотр заявок, требующих рассмотрения	–	–	+	+
Подтверждение заявки	–	–	+	+
Отклонение заявки	–	–	+	+
Предложение альтернативной аудитории	–	–	+	+
Создание бронирования нескольких аудиторий	–	–	+	+
Просмотр всех заявок	–	–	–	+
Изменение статуса аудитории	–	–	–	+
Создание и редактирование мероприятий	–	+	+	+
Формирование отчёта по прошедшим мероприятиям	–	+	+	+
Просмотр истории изменений бронирования	–	+	+	+

Ролевая модель включает неавторизованного пользователя, преподавателя, ответственного сотрудника и администратора. На клиентской стороне роль определяет доступные экраны и элементы управления, а на серверной стороне используется для проверки прав доступа перед выполнением операций [2]. Наследование ролей пользователей представлено на рисунке 2.

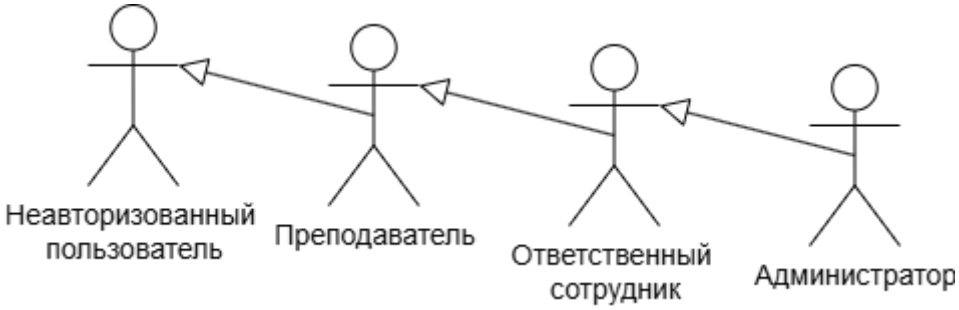

Рисунок 2. Иерархия пользователей

Разграничение ролей позволяет отделить публичные информационные сценарии от операций, влияющих на состояние системы, что снижает вероятность несанкционированного изменения данных [2].

Архитектура и модель данных

Сервис реализован как клиент-серверное веб-приложение, взаимодействие между частями которого осуществляется через документированное REST API. Общая архитектура серверной части показана на рисунке 3.

Серверная часть системы построена на основе многоуровневой архитектуры. Клиентское приложение отправляет HTTP-запрос на сервер, после чего запрос направляется в уровень маршрутизации. Маршруты, реализованные с помощью Express.js, определяют, какой метод контроллера должен обработать входящий запрос.

Контроллер получает данные запроса, выполняет первоначальную обработку параметров и передает их на уровень сервисной логики. На этом уровне выполняются основные операции системы: валидация данных, обработка событий, запросов на аудитории и бронирования, проверка доступности аудиторий и обновление статусов запросов.

Для взаимодействия с базой данных используется Prisma ORM. С его помощью серверная часть выполняет запросы к PostgreSQL, где хранится информация о пользователях, ролях, аудиториях, оборудовании, мероприятиях, бронированиях, документах и расписании занятий [6].

После выполнения необходимых операций сервер генерирует ответ и возвращает его клиентскому приложению (см. рис. 3)

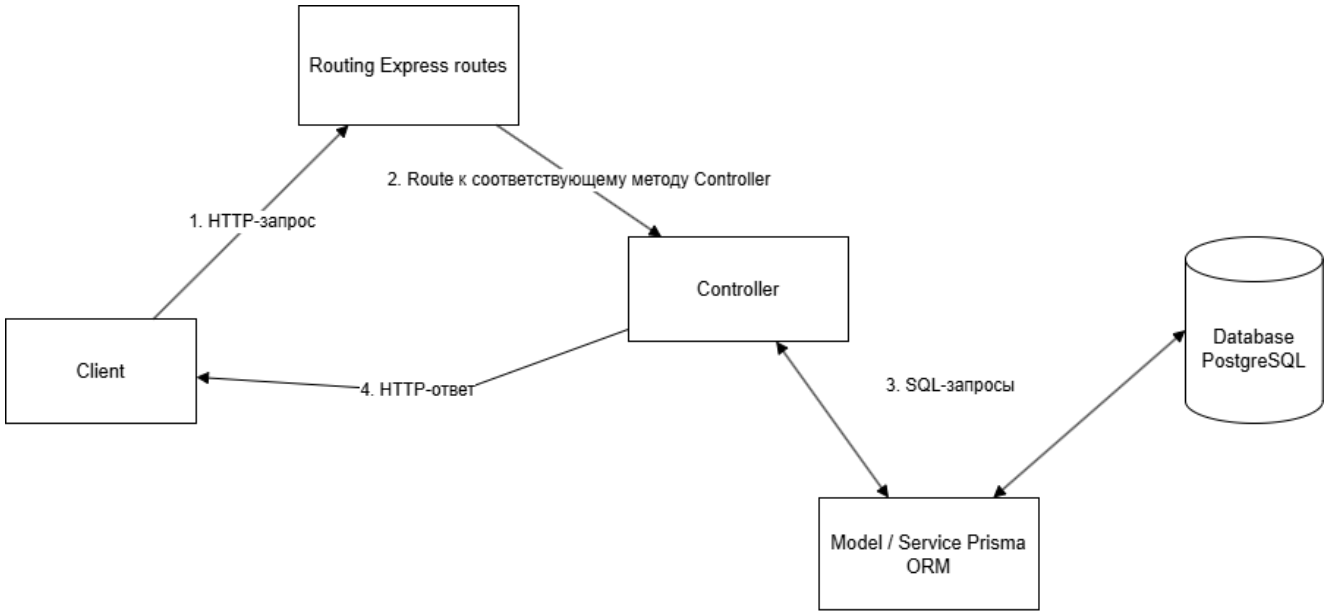

Рисунок 3. Архитектура серверной части сервиса

Модель данных имеет реляционную основу. Центральными сущностями являются мероприятия и бронирования. Мероприятие описывает содержательную часть события, а бронирование фиксирует факт резервирования конкретной аудитории на заданный период. Отдельные сущности используются для аудиторий, оборудования, документов, пользователей, ролей, учебного расписания, внешних мероприятий Indico и истории изменений. Фрагмент физической модели данных приведен на рисунке 4.

Использование PostgreSQL и Prisma ORM позволяет описывать структуру данных в виде формализованной схемы и выполнять миграции при развитии проекта [6]. История изменений выделена в отдельные таблицы, поскольку для

процесса бронирования важны не только текущие значения, но и последовательность действий, приведших к итоговому статусу заявки.

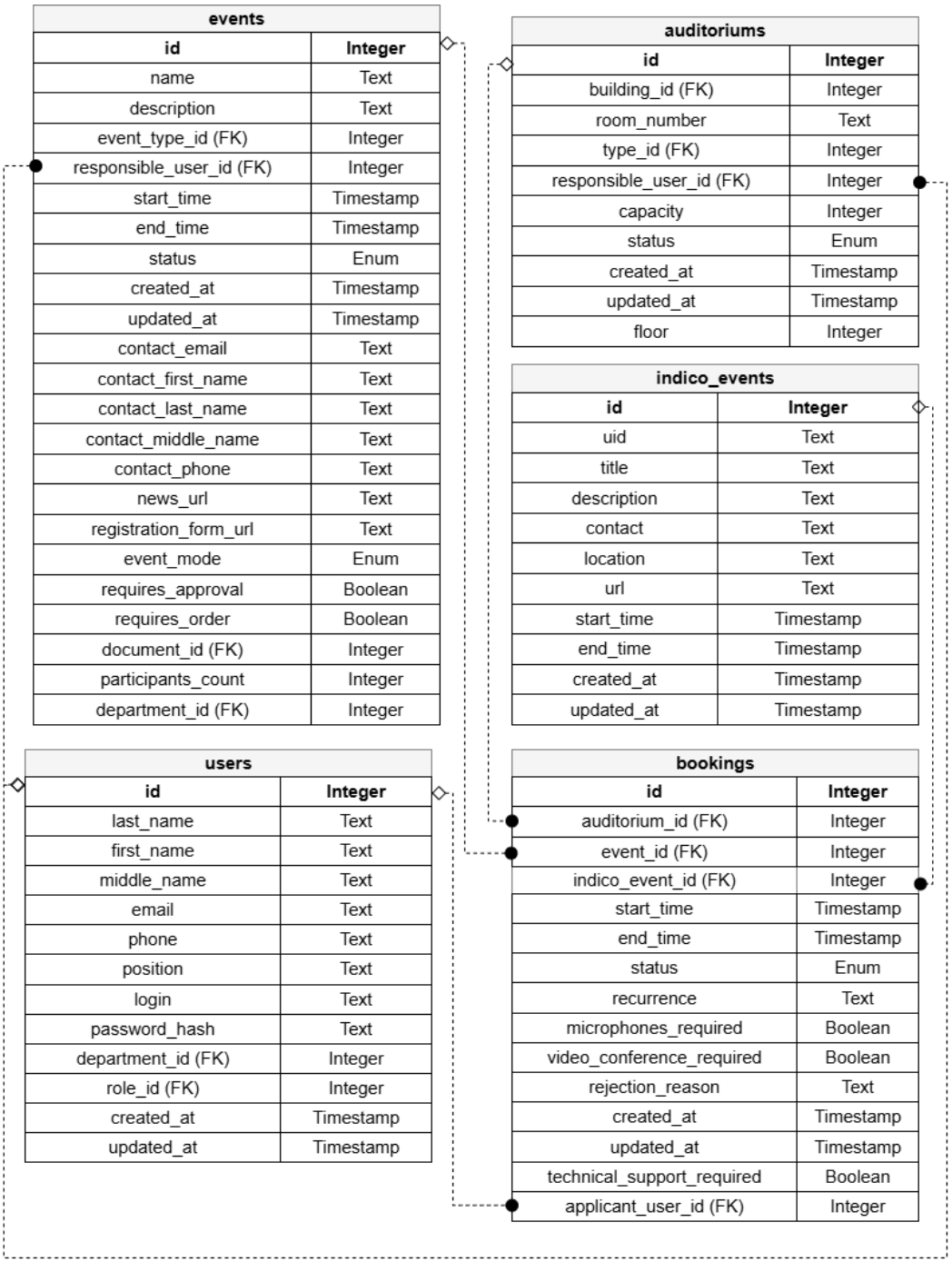

Рисунок 4. Фрагмент физической модели данных сервиса

Технологический стек сервиса приведён в таблице 2.

Таблица 2.

Технологический стек сервиса

Компонент	Стек
Клиентская часть	Next.js, React, TypeScript, Tailwind CSS, shadcn/ui
Состояние и работа с API	Redux Toolkit, RTK Query
Календарь мероприятий	FullCalendar
Серверная часть	Node.js, Express.js, TypeScript
База данных	PostgreSQL, Prisma ORM
Внешние системы	«1С:Документооборот», Indico
Документирование	Swagger

Логика обработки заявки на бронирование

Центральным бизнес-процессом сервиса является создание и проверка заявки на бронирование. Пользователь указывает параметры поиска (дату, время, требования к аудитории) и получает список свободных вариантов; после выбора аудитории и отправки формы сервер повторно проверяет пересечения с уже созданными бронированиями и учебным расписанием, на этот раз в рамках транзакции создания заявки.

Такая двойная проверка необходима, поскольку клиентское приложение не может быть источником окончательного решения: между моментом просмотра календаря и отправкой формы состояние базы данных может измениться. Окончательная проверка объединена с созданием заявки в одну транзакцию и выполняется в сервисном слое: из базы данных отбираются бронирования и занятия учебного расписания, попадающие в указанный интервал; при обнаружении пересечения заявка не сохраняется, а клиент получает ответ с ошибкой конфликта. Последовательность взаимодействия компонентов при создании заявки, включая оба этапа проверки занятости и обработку конфликта, показана на рисунке 5.

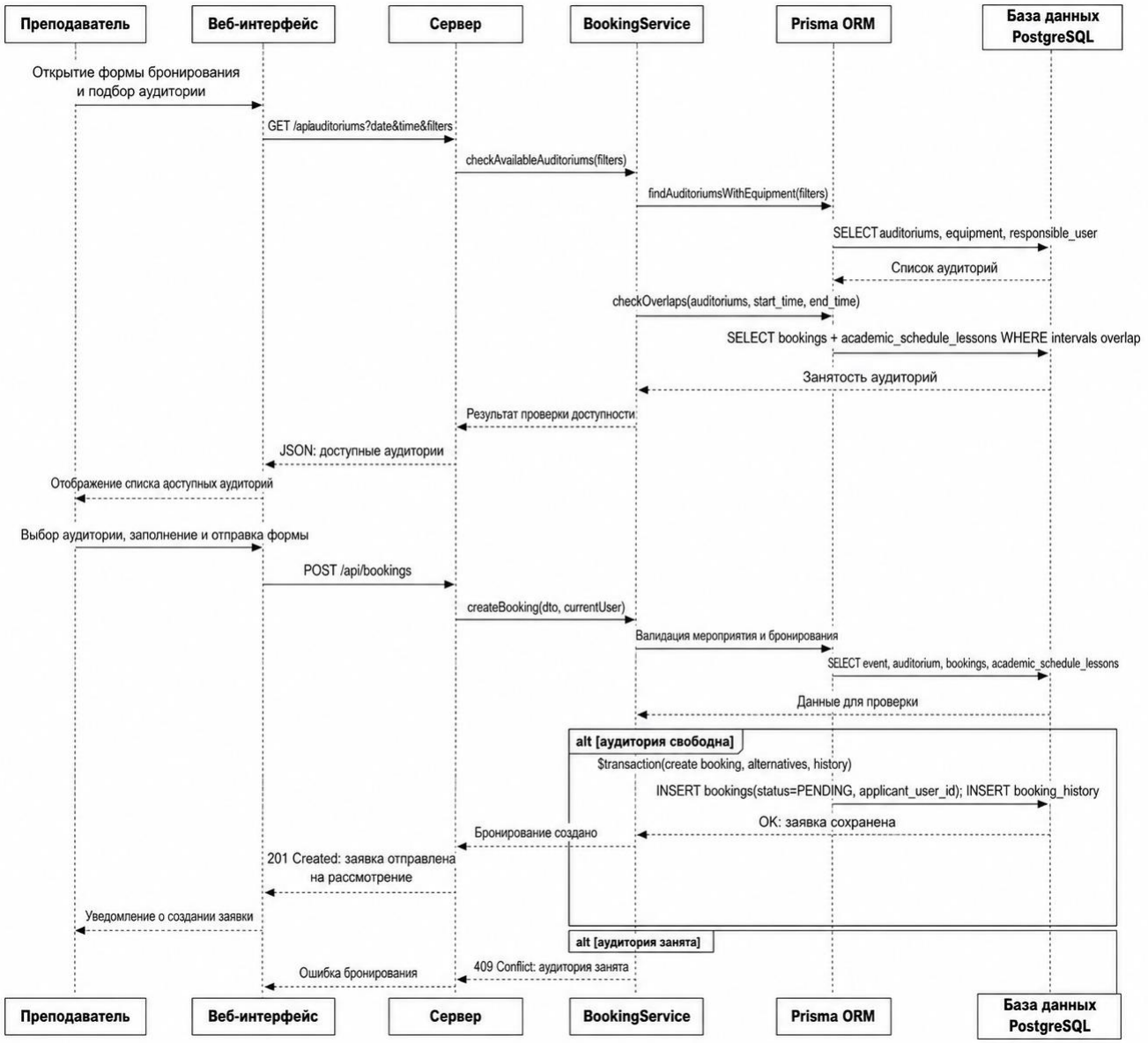

Рисунок 5. Диаграмма последовательности создания заявки на бронирование

Реализация клиентской части и интерфейсных сценариев

Клиентское приложение построено на Next.js и React с использованием компонентного подхода: элементы интерфейса (кнопки, карточки, формы, диалоговые окна) вынесены в переиспользуемые компоненты [7; 9]. Работа с серверным API организована через сервисы RTK Query, а обращения к серверной части выполняются не напрямую, а через промежуточные API-обработчики Next.js, что позволяет скрыть служебные заголовки и ключ доступа к публичным маршрутам [7]. Для повышения надёжности разработки используется TypeScript

[8]. Роль пользователя определяет набор доступных экранов и элементов управления.

Интерфейс должен не только отображать сведения, но и направлять пользователя по корректному сценарию: сначала выбрать аудиторию или мероприятие, затем указать параметры бронирования, после чего отправить заявку и получить информацию о её статусе. Одним из ключевых экранов является календарь мероприятий (см. рис. 6). Для университета такой календарь выполняет информационную функцию, а инициаторам мероприятий помогает предварительно оценить загруженность аудиторного фонда.

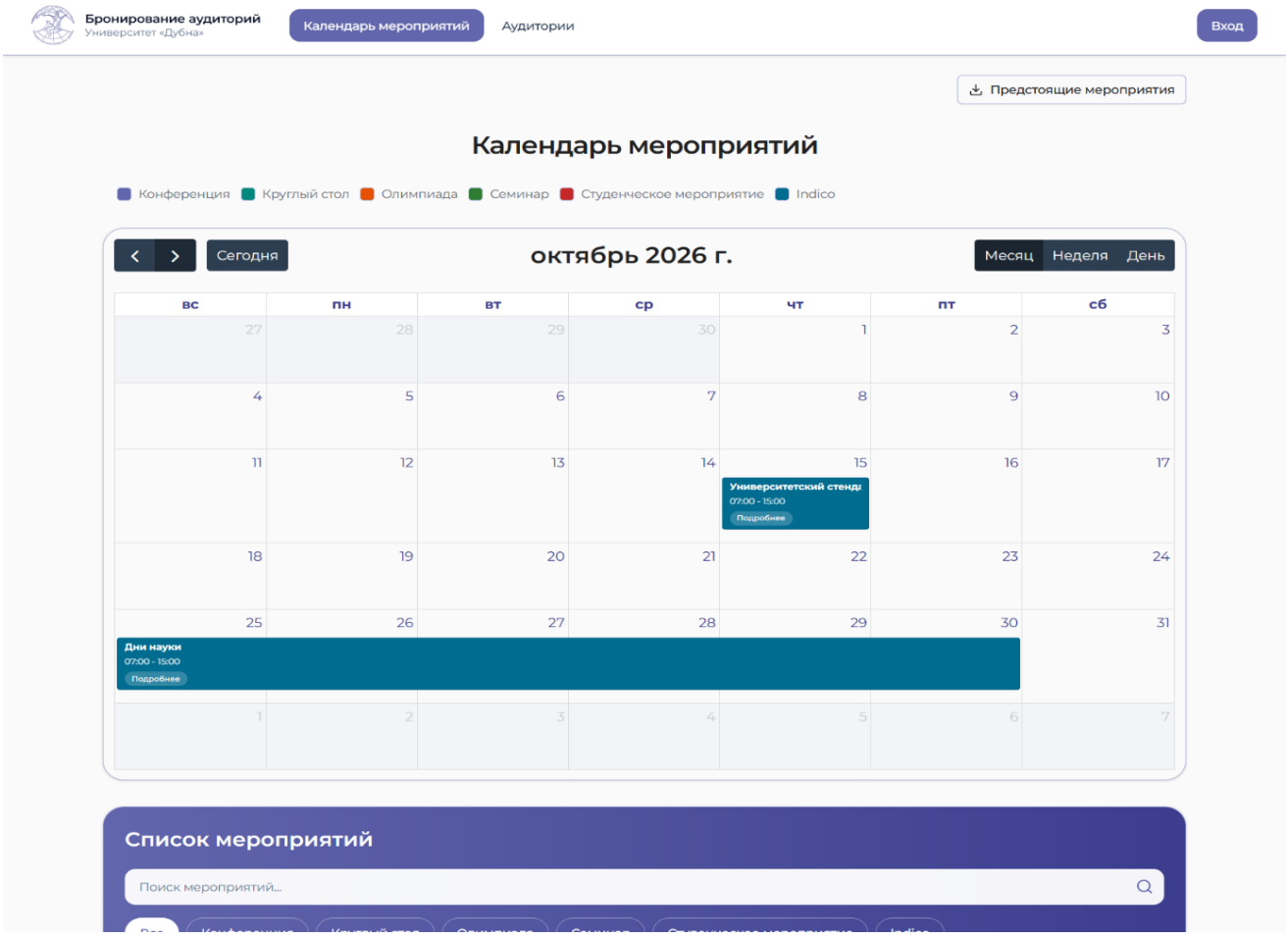

Рисунок 6. Календарь мероприятий

Экран подбора аудитории реализует фильтрацию по параметрам помещения: корпусу, типу, вместимости и оснащению (см. рис. 7). Наличие такого механизма

снижает количество некорректных заявок, поскольку пользователь заранее видит помещения, соответствующие организационным требованиям мероприятия [1].

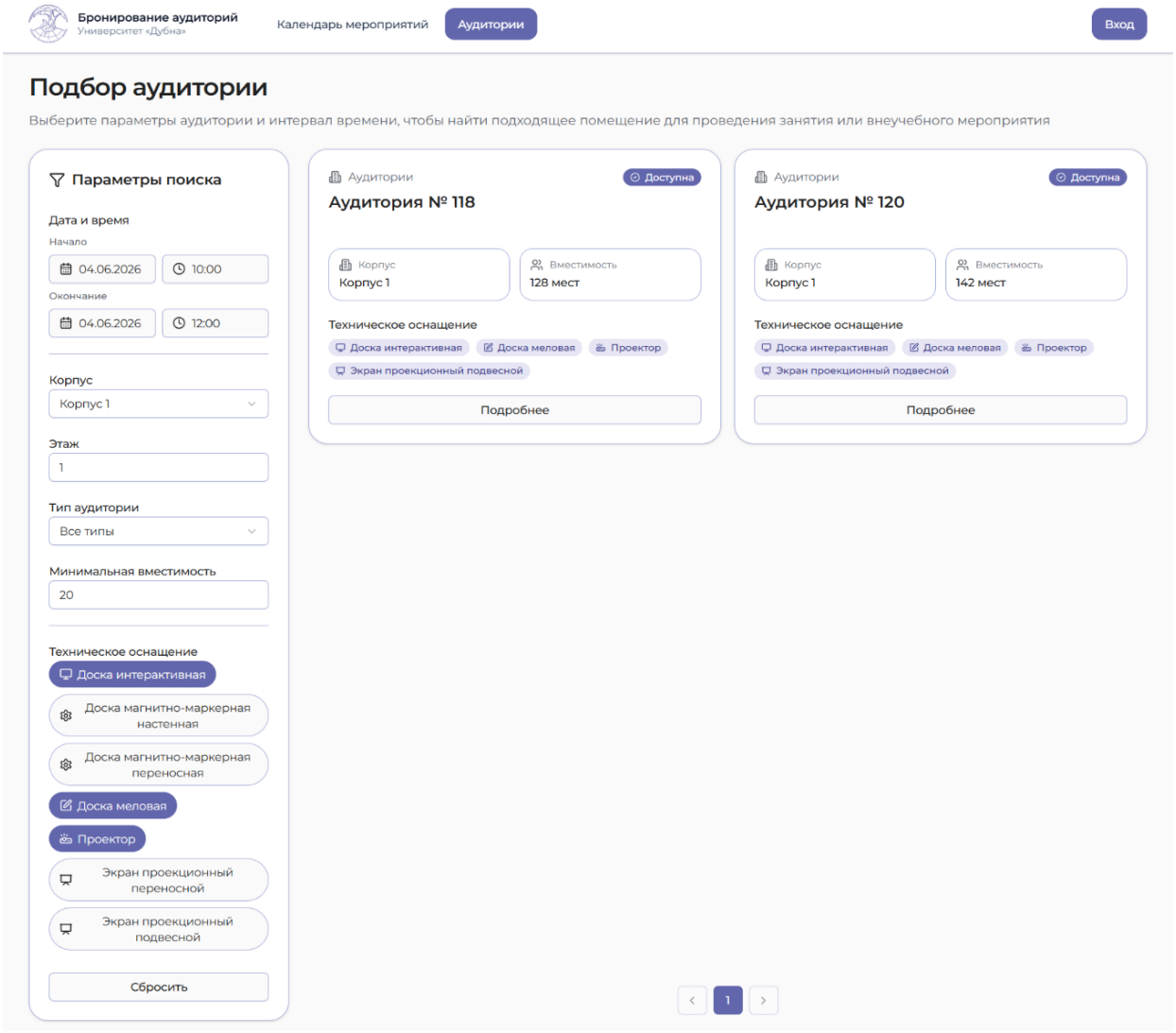

Рисунок 7. Страница подбора аудитории

Форма создания заявки разделена на логические блоки (см. рис. 8). Пользователь последовательно указывает сценарий бронирования, аудиторию, временной интервал, сведения о мероприятии и дополнительные требования. Такая структура уменьшает когнитивную нагрузку и делает процесс оформления заявки более предсказуемым.

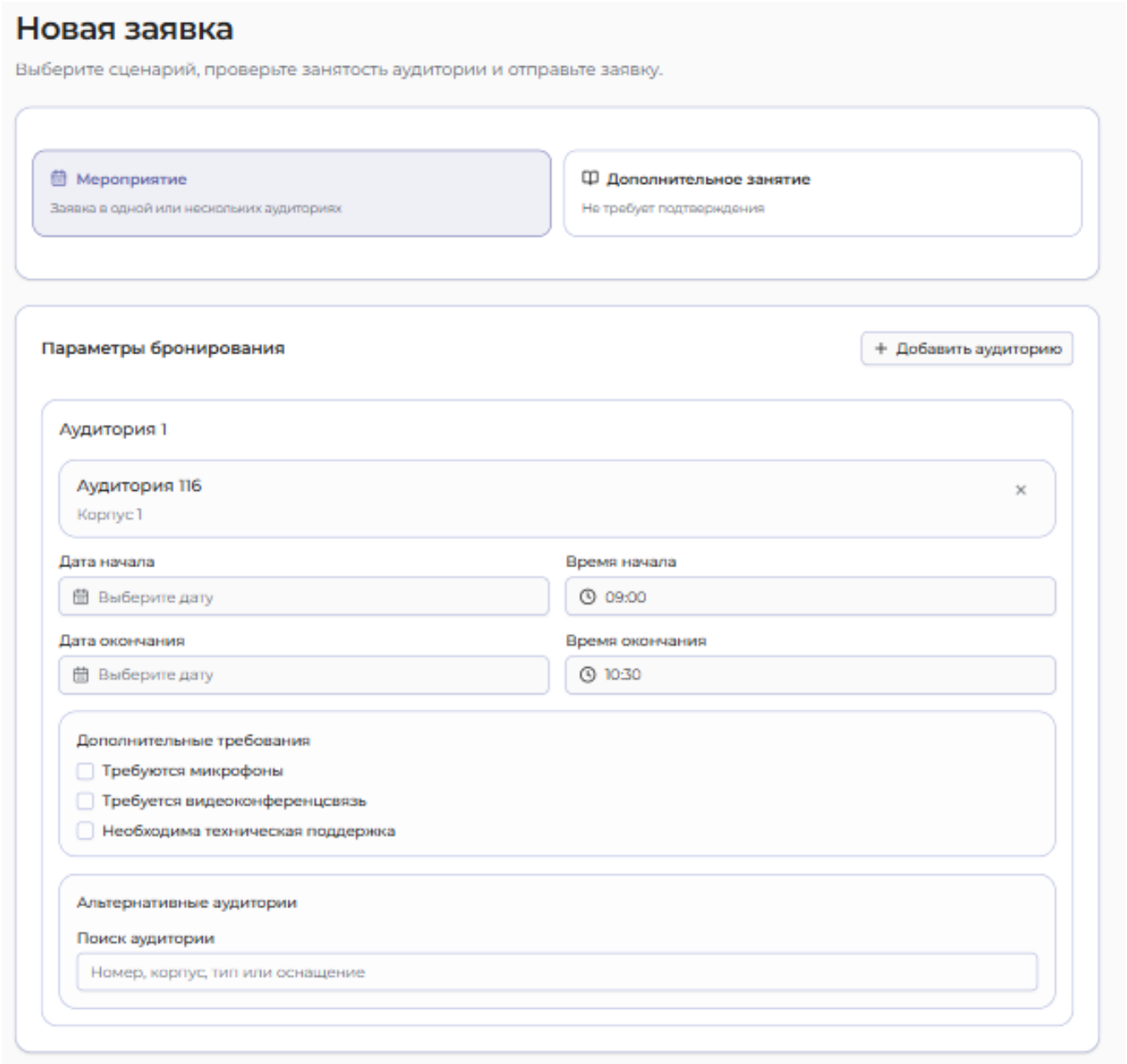

Рисунок 8. Форма создания заявки на бронирование

Дальнейшая работа с заявкой сосредоточена на странице бронирований. Для инициатора она содержит список собственных заявок с фильтрацией по статусу и возможностью отмены, а для ответственного сотрудника и администратора дополняется действиями подтверждения, отклонения с указанием причины и замены аудитории. В карточке заявки отображается история изменения статусов, благодаря чему процесс рассмотрения остаётся прослеживаемым для обеих сторон (см. рис. 9).

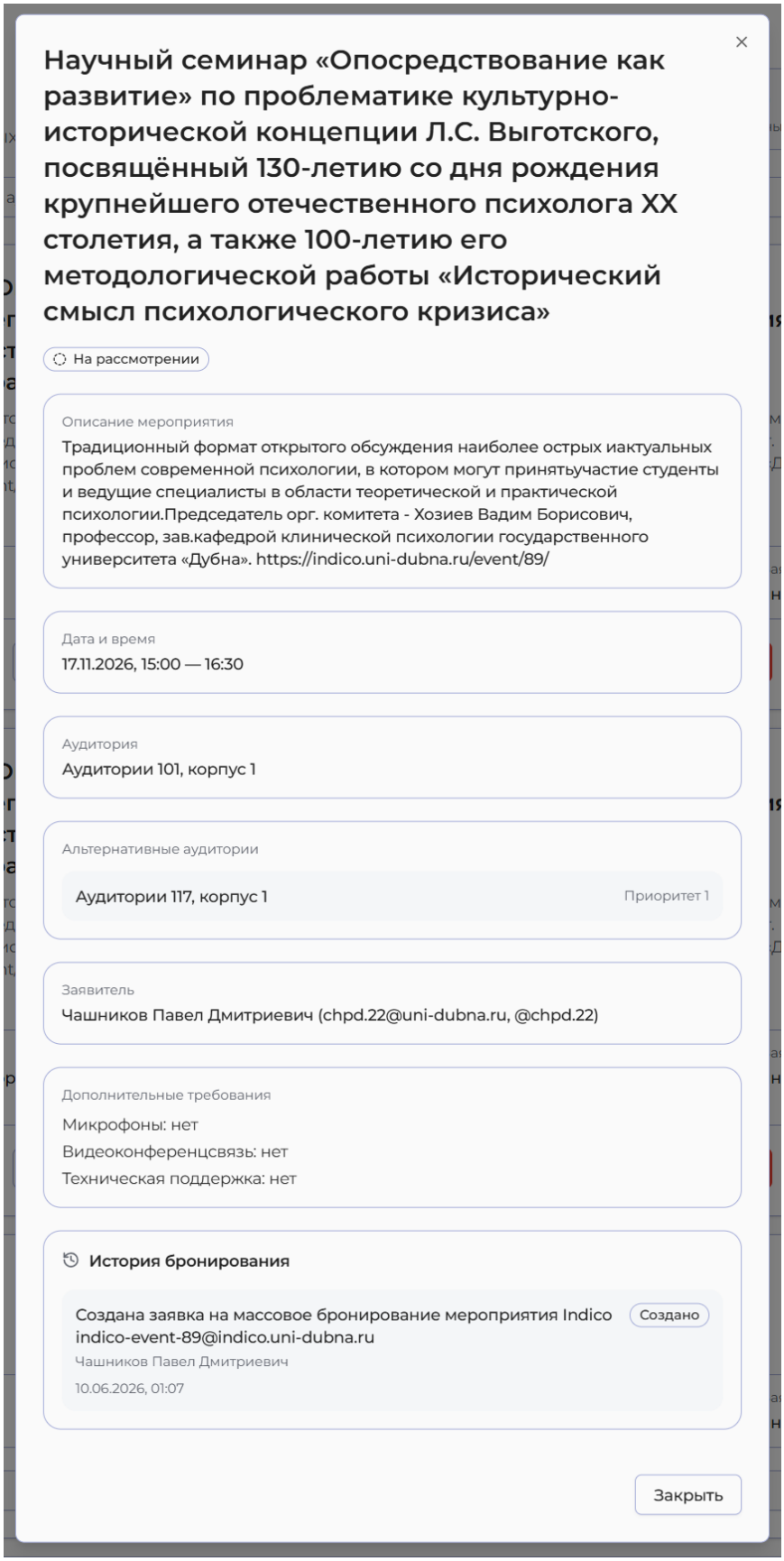

Рисунок 9. Подробная информация о бронировании

Интеграции, безопасность и тестирование

Практическая ценность сервиса повышается за счёт интеграции с внешними источниками данных. Из системы «1С:Документооборот» поступают сведения об учебном расписании, структуре университета и распорядительных документах. Интеграция с Indico обеспечивает получение сведений о научных мероприятиях, которые отображаются в общем календаре отдельной категорией. Синхронизация выполняется по расписанию: еженедельно для данных «1С:Документооборот» и ежечасно для Indico, а также может быть запущена вручную.

Дополнительно реализовано формирование отчётов в формате XLSX по прошедшим мероприятиям с фильтрацией по ответственному сотруднику, типу мероприятия, кафедре и диапазону дат, выгрузка списка предстоящих мероприятий, а также уведомления по электронной почте.

Функциональное тестирование выполнялось вручную: серверная часть проверялась запросами через Swagger-документацию, клиентская — прохождением пользовательских сценариев в браузере. Для серверной части подготовлено и выполнено 102 тест-кейса, охватывающих разграничение доступа по ролям, работу со справочниками, создание и фильтрацию мероприятий, создание заявок и проверку конфликтов, подтверждение, отклонение и отмену бронирования. Для клиентской части выполнено 35 тест-кейсов, проверяющих переходы между страницами, валидацию форм, отображение состояний загрузки, ошибок и пустых результатов, работу модальных окон и уведомлений, а также состав доступных действий для каждой роли [2].

Заключение

В рамках проведённой работы спроектирован и реализован веб-сервис бронирования аудиторий для внеучебных мероприятий университета. Сервис формализует процесс подачи и рассмотрения заявок, разграничивает доступ между ролями, хранит сведения об аудиториях, мероприятиях и бронированиях и

выполняет проверку занятости помещений с учётом как ранее созданных заявок, так и учебного расписания.

На основе анализа предметной области и существующих решений выявлены ключевые организационные проблемы: разрозненность информации о доступности аудиторий, ручной характер согласования, риск двойного бронирования и высокая нагрузка на ответственных сотрудников. Разграничение доступа реализовано одновременно на уровне интерфейса и на уровне серверных проверок.

Практическая значимость разработки заключается в создании основы для дальнейшей цифровизации процесса управления аудиторным фондом. Направлениями развития являются авторизация через единую учётную запись университета, интеграция с личными календарями пользователей, расширение состава отчётов и аналитических показателей загруженности аудиторий, а также разработка мобильного приложения.

Список литературы

1. Рабовская М. Я., Козлов В. Р. Автоматизация процесса бронирования аудиторий // International Journal of Advanced Studies. 2021. Т. 11, № 4. С. 43–52.
2. Савватеева Т. П., Миловидова А. А., Кудрявцева Д. В. Технологии проектирования информационных систем: учебное пособие. Дубна: Государственный университет «Дубна», 2016. 120 с.
3. Моисеев А. Н., Литовченко М. И. Основы языка UML: учебное пособие. Томск: Томский государственный университет, 2023. 96 с.
4. Кукин А. А. Разработка подсистемы планирования научных мероприятий Государственного университета «Дубна»: бакалаврская работа. Дубна: Государственный университет «Дубна», 2025. 83 с.
5. Карпов Т. Ю. Веб-приложение для управления расписанием мероприятий в университете «Дубна»: бакалаврская работа. Дубна: Государственный университет «Дубна», 2025. 45 с.

6. PostgreSQL Documentation // PostgreSQL. URL: https://www.postgresql.org/docs/ (дата обращения: 10.05.2026).
7. Next.js Documentation // Next.js. URL: https://nextjs.org/docs (дата обращения: 10.05.2026).
8. TypeScript Documentation // TypeScript. URL: https://www.typescriptlang.org/docs/ (дата обращения: 10.05.2026).
9. React Documentation // React. URL: https://react.dev/ (дата обращения: 10.05.2026).