

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Веб-приложение для планирования деятельности
совета студенческого самоуправления факультета**

Студент

(подпись, дата)

Новожинов Иван Алексеевич
(ФИО полностью)

Руководитель

(подпись, дата)

Балашова И.Ю.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

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

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой

(протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Новожинов Иван Алексеевич гр. 21ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Веб-приложение для планирования деятельности совета студенческого самоуправления факультета

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент, доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) *Назначение: веб-приложение для планирования деятельности совета студенческого самоуправления*

2) *Среда выполнения: Windows 10, Linux, Mac OS.*

Языковые средства разработки: TypeScript, React, C#, .NET, SQL, UML.

Инструментальные средства: Visual Studio Code, Vite, Docker, Swagger, PostgreSQL, Draw.io, Visual Paradigm 17.2.

Технологии разработки: ООП, модульный монолит, MVC.

3) *Основные функции приложения:*

а) регистрация и авторизация пользователей с разграничением ролей;

б) планирование и контроль мероприятий, задач и встреч;

в) ведение архива завершенных и отмененных мероприятий и задач;

г) формирование отчетов о проведенных мероприятиях и планов;

д) ведение базы партнеров с контактными данными;

е) просмотр и систематизация заметок с markdown-файлами.

5. Перечень подлежащих разработке вопросов:

1) <i>Анализ предметной области и постановка задачи;</i>
2) <i>Проектирование;</i>
3) <i>Реализация;</i>
4) <i>Контроль качества программного обеспечения;</i>
5) <i>Оформление пояснительной записки.</i>

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 130 листов, 22 рисунка, 10 таблиц, 22 использованных источника и 7 приложений.

СОВЕТ СТУДЕНЧЕСКОГО САМОУПРАВЛЕНИЯ, ВЕБ-ПРИЛОЖЕНИЕ, МОДУЛЬНЫЙ МОНОЛИТ, ASP.NET CORE, REACT, REST API, POSTGRESQL, EF CORE, TYPESCRIPT, REACT.

Объект исследования – веб-приложение для планирования деятельности совета студенческого самоуправления.

Цель выпускной квалификационной работы – спроектировать и разработать веб-приложение для организации планирования мероприятий, постановки и контроля выполнения задач, формирования отчетов в рамках совета студенческого самоуправления.

Технологии разработки – клиентская часть приложения реализована на языке программирования TypeScript с использованием библиотеки React, сборщика Vite и фреймворка Tailwind CSS в среде разработки Visual Studio Code. Серверная часть разработана на языке программирования C# с использованием ASP.NET Core и ORM Entity Framework Core в среде разработки Visual Studio Code. База данных спроектирована и развернута в СУБД PostgreSQL.

Среда исполнения – веб-браузер и серверное приложение ASP.NET Core, размещенное на локальном или удаленном сервере.

Результат работы – в ходе выполнения выпускной квалификационной работы спроектированы и разработаны клиентская и серверная части веб-приложения, реализована база данных, обеспечено разграничение прав доступа по ролям и организована работа с JWT-аутентификацией. Проведено тестирование программных средств, подтверждающее их работоспособность и соответствие заявленным требованиям.

					ПГУ 09.03.04 – 08BP212.21 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Новоженков И.А.			Веб-приложение для планирования деятельности совета студенческого самоуправления факультета. Пояснительная записка	Лит.	Лист
Пров.		Балашова И.Ю.					
Реценз.						4	130
Н. Контр.		Такташкин Д.В.				Группа 21ВП2	
Утв.		Козлов А.Ю.					

Содержание

Введение.....	7
1 Анализ веб-приложений для планирования деятельности совета студенческого самоуправления факультета	8
1.1 Исследование предметной области	8
1.2 Обзор и анализ веб-приложений для планирования деятельности совета студенческого самоуправления факультета	8
1.2.1 Веб-приложение «Trello»	8
1.2.2 Веб-приложение «Google Календарь»	9
1.2.3 Веб-приложение «Asana»	10
1.2.4 Веб-приложение «LeaderTask»	11
1.2.5 Сравнение и анализ веб-приложений для планирования деятельности совета студенческого самоуправления факультета	12
1.3 Анализ функциональных требований	14
1.4 Анализ нефункциональных требований	18
1.5 Планирование разработки и оценка бюджета.....	19
2 Проектирование веб-приложения для планирования деятельности совета студенческого самоуправления факультета	21
2.1 Архитектура приложения	21
2.2 Проектирование структур данных	22
2.2.1 Концептуальная модель базы данных	22
2.2.2 Логическая модель базы данных	23
2.3 Проектирование пользовательского интерфейса.....	24
2.4 Проектирование серверной части приложения	25

3 Реализация веб-приложения для планирования деятельности совета студенческого самоуправления факультета	27
3.1 Реализация базы данных.....	27
3.2 Реализация серверной части приложения	27
3.3 Реализация клиентской части приложения	28
3.4 Реализация интерфейса приложения	30
4 Контроль качества программного обеспечения	41
4.1 Выбор режима тестирования и метрик кода	41
4.2 Тестирование веб-приложения для планирования деятельности совета студенческого самоуправления факультета	42
4.2.1 Функциональное тестирование	42
4.2.2 Модульное тестирование	44
4.3 Оценка качества программного кода	46
4.3.1 Метрика Холстеда.....	46
4.3.2 Метрика Чепина	48
Заключение	50
Список используемых источников.....	52
Приложение А (справочное) Глоссарий	55
Приложение Б (обязательное) Листинг серверной части приложения	57
Приложение В (обязательное) Листинг клиентской части приложения.....	75
Приложение Г (рекомендуемое) Графическая часть.....	93
Приложение Д (рекомендуемое) Атрибуты сущностей базы данных.....	99
Приложение Е (рекомендуемое) Результаты функционального тестирования	106
Приложение Ж (рекомендуемое) Листинг модульных тестов	114

Введение

Планирование деятельности играет ключевую роль в современном мире, особенно в контексте студенческих мероприятий. Оно обеспечивает эффективность, организованность и успешную реализацию проектов, позволяя учитывать множество факторов: от бюджета и логистики до привлечения участников и оценки результатов.

В частности, процесс планирования, подготовки и проведения мероприятий предполагает осуществление студентами таких видов деятельности, как анализ ситуации, планирование, организация, контроль и рефлексия [1]. Эти действия способствуют развитию навыков самоорганизации, ответственности и командной работы [2].

Разработка веб-приложения для планирования деятельности является актуальной темой на сегодняшний день, так как такие приложения помогают эффективно организовывать рабочий процесс, управлять временем, контролировать выполнение задач и координировать командную работу.

Целью данной выпускной квалификационной работы является разработка веб-приложения для планирования деятельности ССС (совет студенческого самоуправления) факультета, которое позволит тщательно планировать мероприятия, распределять обязанности между участниками, контролировать выполнение задач и соблюдать установленные сроки.

Исходя из цели данной работы, были поставлены следующие задачи:

- провести анализ предметной области и аналогов похожего программного обеспечения, поставить задачу на разработку;
- спроектировать клиентскую и серверную часть приложения и структуры данных;
- разработать модули веб-приложения;
- провести контроль качества программного обеспечения.

Успешное выполнение поставленных задач позволит создать эффективный инструмент для планирования деятельности ССС факультета.

1 Анализ веб-приложений для планирования деятельности совета студенческого самоуправления факультета

1.1 Исследование предметной области

Программное обеспечение для управления проектами – комплексное программное обеспечение, включающее в себя приложения для планирования задач, составления расписания, управления бюджетом, распределения ресурсов, совместной работы, общения, быстрого управления, документирования и администрирования системы, которая используется совместно для управления крупными проектами. Данное программное обеспечение является веб-приложением, если доступ к нему осуществляется с помощью браузера [3].

Обязательным условием при использовании таких приложений является регистрация пользователя для сохранения информации о его деятельности.

Веб-приложение для планирования деятельности ССС факультета – это точно такое же приложение для планирования задач, составления расписания, управления бюджетом, но в рамках ССС.

Такое приложение может поспособствовать повышению прозрачности и эффективности работы студенческого самоуправления, а также развитию у студентов навыков самоорганизации и ответственности.

1.2 Обзор и анализ веб-приложений для планирования деятельности совета студенческого самоуправления факультета

1.2.1 Веб-приложение «Trello»

«Trello» – сервис для управления задачами по методу «Канбан». Работа ведется на специальных досках: у каждой доски есть колонки, отделенные специальными разделами, а колонки разбиваются на карточки-задания. Перетаскивание задач мышью между спецразделами позволяет легко и просто размещать задачи. Внутри доски можно общаться в карточках – писать комментарии, прикреплять файлы и задавать сроки исполнения задач [4]. Интерфейс приложения «Trello» представлен на рисунке 1.

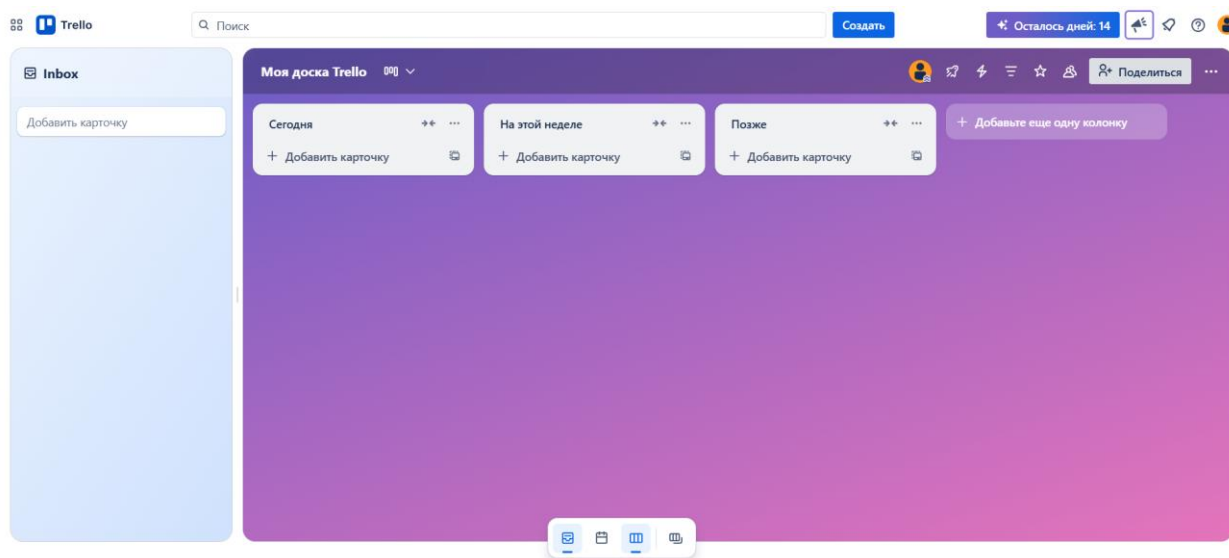


Рисунок 1 – Интерфейс приложения «Trello»

У «Trello» интуитивный интерфейс с визуальными досками, который снижает порог освоения. Приложение поддерживает неограниченное количество досок, а также позволяет организовать совместную работу без особых сложностей. Благодаря шаблонам и интеграциям Trello легко связать со сторонними календарями и мессенджерами.

К недостаткам можно отнести канбан-структуру, которая слишком проста для сложных проектов, ограничены возможности отчетности. У «Trello» нет встроенных средств для хранения всех сведений о мероприятиях, управления командой, а также отсутствует собственная система хранения заметок в системе.

1.2.2 Веб-приложение «Google Календарь»

«Google Календарь» – это бесплатный онлайн-календарь, предназначенный для планирования времени и встреч. Он позволяет создавать события, назначать дату, время и напоминания, а также делиться календарями с другими пользователями и приглашать участников на совещания. Благодаря интеграции с Gmail и другими сервисами Google (Meet, Задачи) этот инструмент удобен для синхронизации расписаний и рассылки уведомлений об изменениях [5]. Интерфейс приложения «Google Календарь» представлен на рисунке 2.

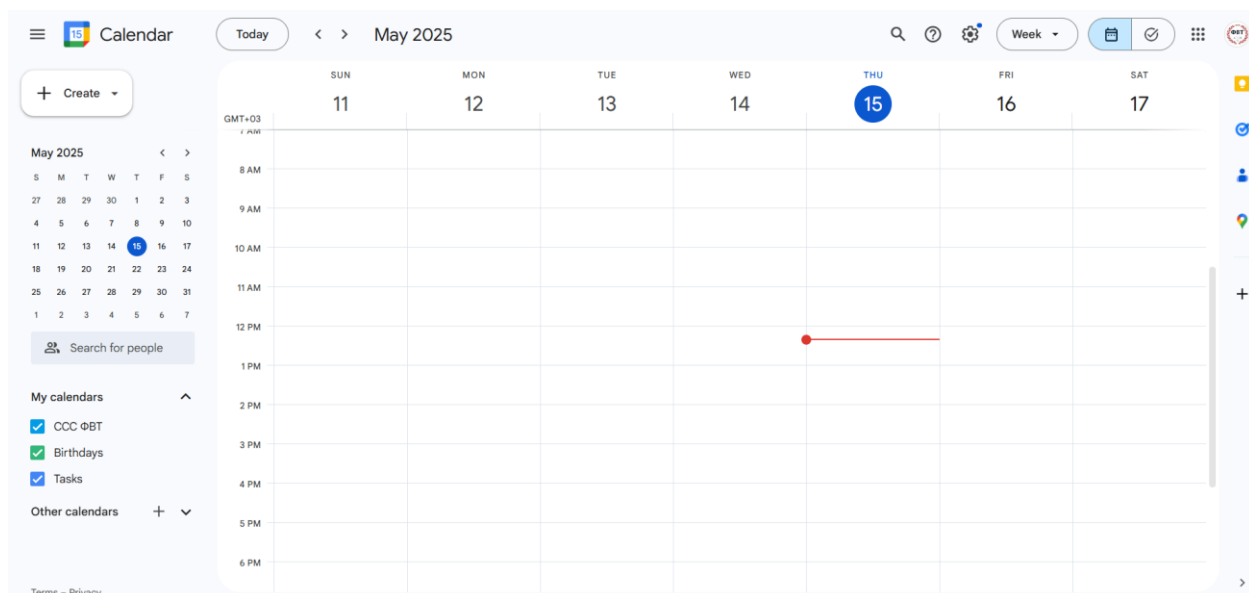


Рисунок 2 – Интерфейс приложения «Google Календарь»

Основное преимущество «Google Календаря» – это быстрое планирование встреч и дедлайнов. Интеграция с почтой позволяет мгновенно отправлять приглашения и напоминания.

Главным недостатком данного приложения является ограниченность только функцией календаря – здесь нет встроенной системы управления задачами, проектами или заметками, нет функционала для распределения ролей и ведения детальных списков дел.

1.2.3 Веб-приложение «Asana»

«Asana» – это веб-сервис управления задачами и проектами, где пользователи создают проекты и задачи с дедлайнами, подзадачи, обсуждения и приоритеты. С помощью «Asana» команда может планировать активность, назначать ответственным конкретные задачи, следить за сроками и собирать статистику. При этом инструмент описывается как цифровой ежедневник: он позволяет добавлять задачи, распределять их по проектам, назначать коллегам сроки выполнения и напоминания [6]. Интерфейс приложения «Asana» представлен на рисунке 3.

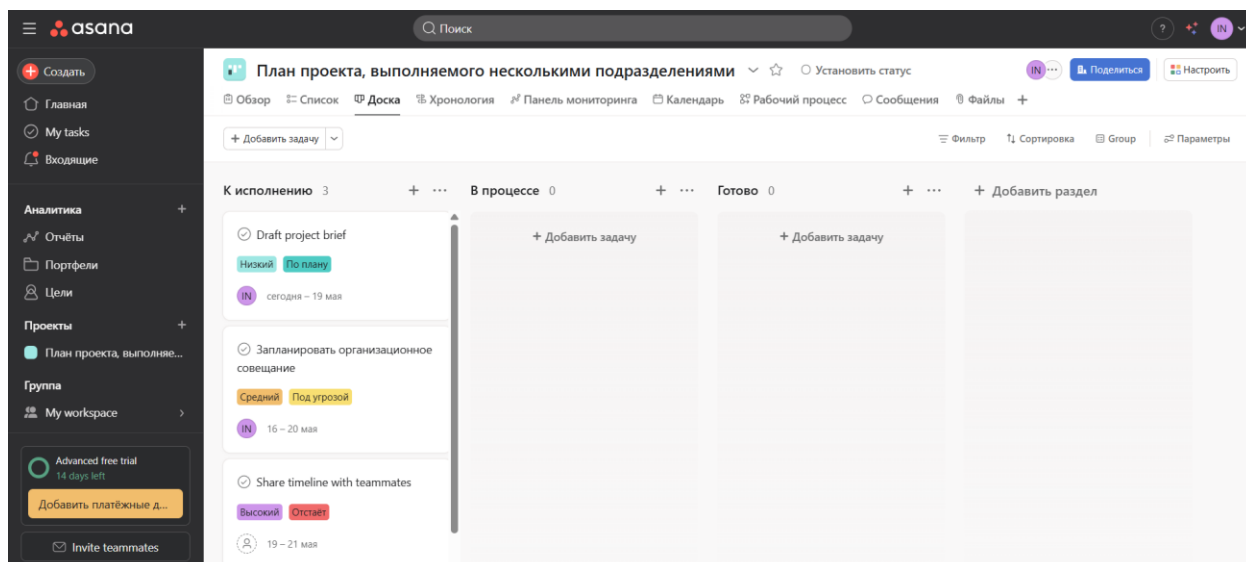


Рисунок 3 – Интерфейс приложения «Asana»

Среди достоинств данного приложения можно выделить то, что оно обладает обширным функционалом: списки задач, доски, календари, отчеты, общие проекты, в которых могут принимать участие несколько членов команды.

Недостатком этого приложения является отсутствие возможности хранения отдельных больших заметок, функционал позволяет только добавлять описание к задачам. Также приложение не обладает функционалом для планирования мероприятий с атрибутами вроде времени, места, количества участников, что приходится компенсировать вручную через задачи и комментарии.

1.2.4 Веб-приложение «LeaderTask»

«LeaderTask» – таск-менеджер и планировщик задач с командной составляющей. Он сочетает в себе личный органайзер и инструменты для совместной работы: имеется календарь с напоминаниями, работа с целями, визуальные доски и встроенный чат. В «LeaderTask» можно создавать общие проекты и распределять задачи по участникам, контролировать выполнение, обсуждать задачи и обмениваться файлами [7]. Интерфейс приложения «LeaderTask» представлен на рисунке 4.

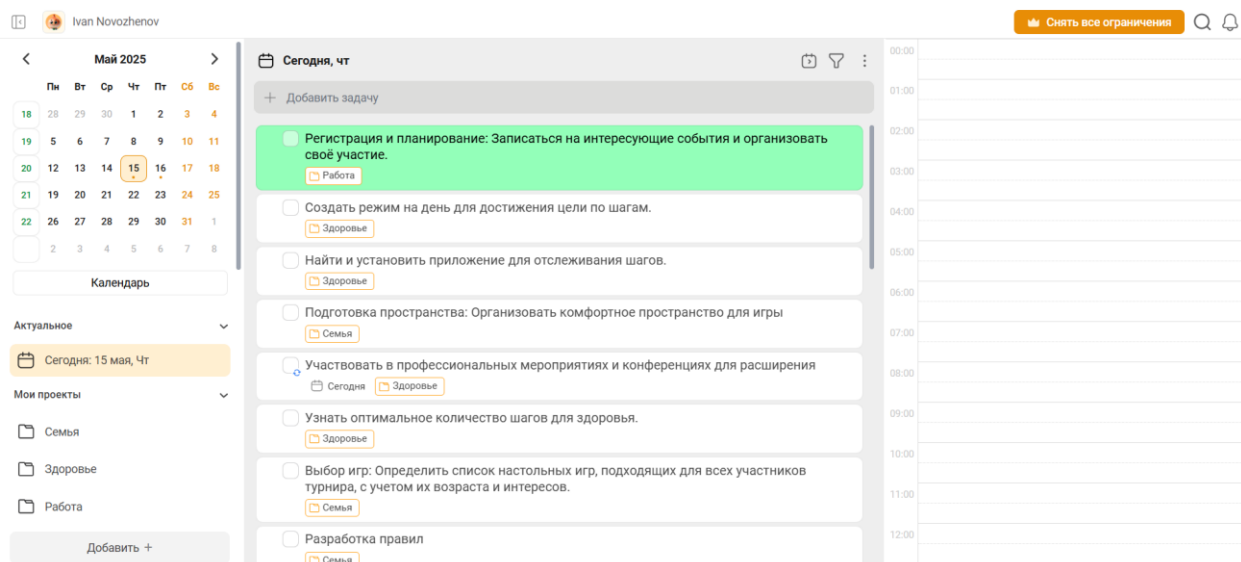


Рисунок 4 – Интерфейс приложения «LeaderTask»

Среди достоинств данного приложения можно выделить: интуитивный интерфейс, встроенный календарь с напоминаниями, возможность группировки задач по проектам и категориям.

Основными недостатками же являются ограниченный функционал для планирования мероприятий, создания заметок и генерации отчетов по мероприятиям и студентам.

1.2.5 Сравнение и анализ веб-приложений для планирования деятельности совета студенческого самоуправления факультета

Чтобы выявить какое из рассмотренных выше веб-приложений наиболее удовлетворяет ключевым функциям, проведем сравнительный анализ.

Оценивать каждое приложение будем по трехбалльной шкале. Характеристики оценки и соответствующие баллы приведены в таблице 1.

Таблица 1 – Баллы и характеристики сравнительного анализа

Балл	Характеристика оценки
0	функция полностью отсутствует
1	функция частично реализована
2	функция полностью реализована

Результаты сравнительного анализа веб-приложений представлены в таблице 2.

Таблица 2 – Сравнение веб-приложений аналогичного назначения

Приложение Критерий	«Trello»	«Google Календарь»	«Asana»	«LeaderTask»
1. Возможность добавления, редактирования и выполнения задач	2	2	2	2
2. Возможность назначения исполнителя задачи	2	0	2	2
3. Возможность планирования мероприятия (дата, время, место, количество участников)	0	2	0	0
4. Возможность генерации отчетов и планов по мероприятию и студентам	0	0	2	1
5. Возможность создания объемных заметок для мероприятий с дополнительной информацией	1	1	1	1
6. Возможность планирования встреч и видеозвонков	0	2	0	0
7. Возможность хранения информации о студентах, их задачах, мероприятиях	0	0	0	2
8. Возможность хранения информации о партнерах, их задачах и мероприятиях	1	0	1	1
Итого	6	7	8	9

Для более наглядного отображения результатов сравнения была создана диаграмма, представленная на рисунке 5.

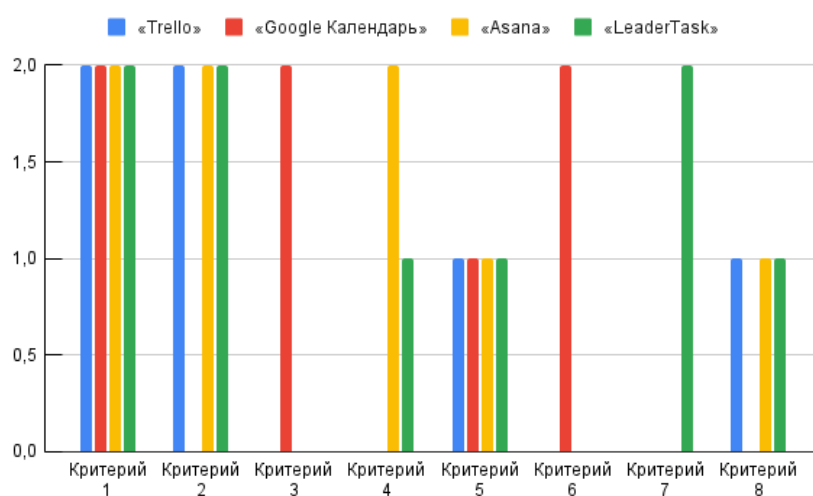


Рисунок 5 – Результаты сравнения приложений-аналогов

В результате сравнительного анализа веб-приложений для планирования деятельности ССС факультета можно заключить, что ни одно из рассмотренных решений полностью не соответствует установленным критериям. Их всех аналогичных приложений «LeaderTask» выделяется наибольшим числом реализованных функций, однако оно не предоставляет пользователям возможностей для планирования мероприятий, встреч и видеозвонков, созданий развернутых заметок, а также отображения задач в различных форматах.

В конечном итоге было принято решение о создании собственного веб-приложения. Разрабатываемое приложение для планирования деятельности ССС факультета будет включать функционал по планированию мероприятий (указание даты, времени, места проведения, назначение ответственных и распределение задач), добавлению подробных заметок к мероприятиям, планированию встреч и видеозвонков, а также формированию ежемесячных планов мероприятий, отчетов о проведенных мероприятиях и персональных отчетов по каждому пользователю.

1.3 Анализ функциональных требований

В ходе анализа задачи на разработку веб-приложения для планирования деятельности ССС факультета были выведены следующие функциональные требования:

- регистрация и авторизация пользователей с разными ролями и правами доступа;
- редактирование данных о пользователе;
- добавление мероприятия;
- изменение мероприятия;
- добавление организаторов, участников и партнеров в мероприятие;
- добавление задачи;
- привязка задачи к мероприятию;
- назначение исполнителя задачи;
- изменение задачи;

- изменение статуса задачи;
- добавление партнера;
- изменение партнера;
- добавление заметки;
- привязка задачи к мероприятию;
- изменение заметки;
- добавление встречи;
- изменение встречи;
- просмотр доступных пользователю мероприятий, задач, партнеров, заметок, встреч;
- просмотр доступных пользователю архивных мероприятий, задач, партнеров, заметок, встреч;
- генерация плана о предстоящих мероприятиях на следующий месяц;
- генерация отчета о прошедших мероприятиях за месяц;
- генерация отчета о пользователе.

В соответствии с данным функциональными требованиями была разработана диаграмма вариантов использования UML. Диаграмма вариантов использования – это первый и один из важнейших этапов объектно-ориентированного анализа, поскольку они помогают: понять потребности заказчика, очертить границы проекта, спроектировать пользовательские сценарии работы с системой до перехода к технической детализации [8].

Диаграмма вариантов использования позволяет наглядно отразить взаимодействие различных ролей пользователей с функционалом системы. Это упрощает процесс согласования требований между заказчиком и разработчиком, а также формирует основу для последующего проектирования интерфейса и реализации бизнес-логики. Благодаря использованию данной диаграммы можно убедиться в полноте набора функций, определить обязательные и необязательные сценарии, а также выявить возможные точки интеграции с внешними сервисами.

Диаграмма вариантов использования представлена на рисунке 6.

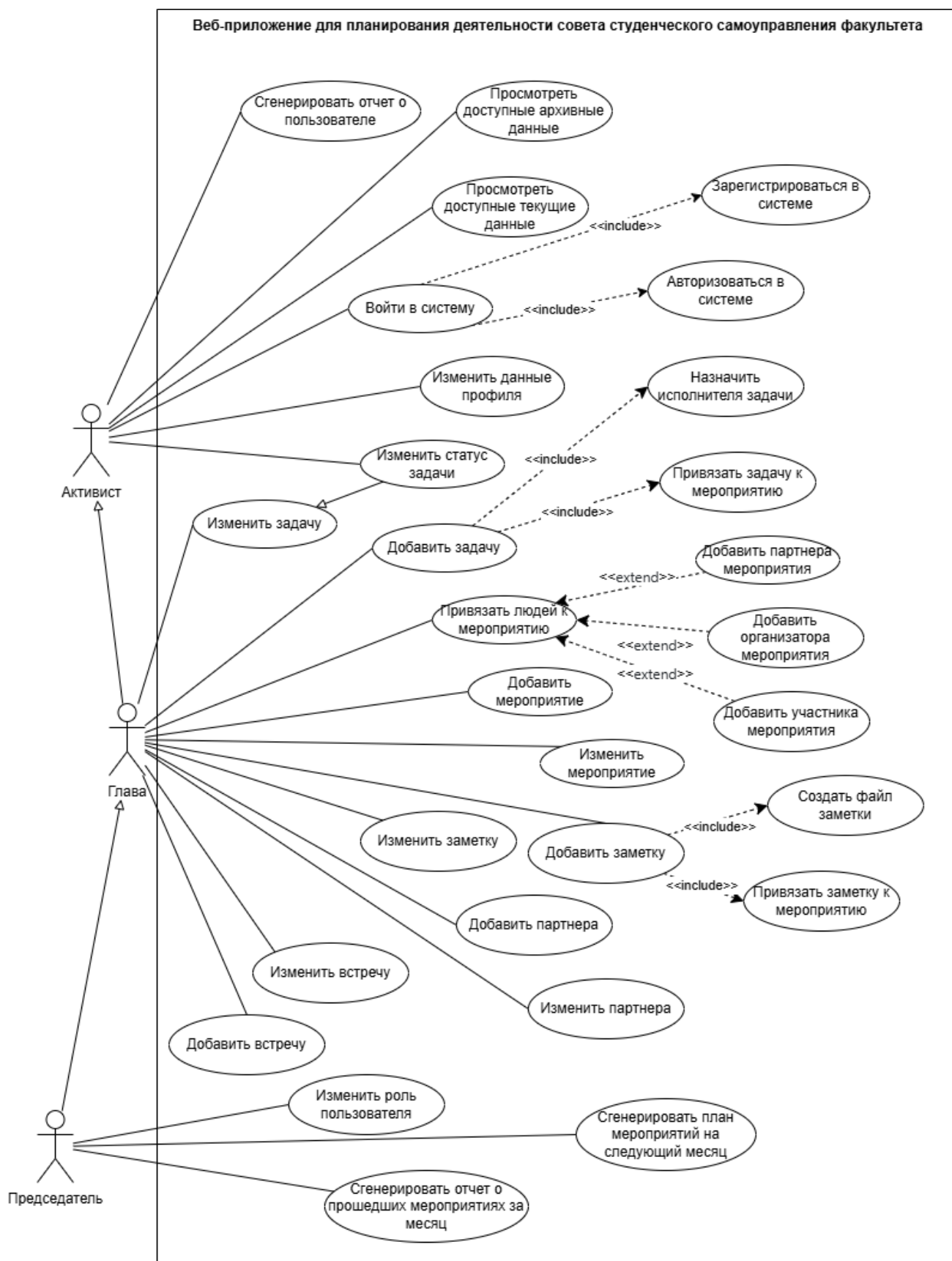


Рисунок 6 – Диаграмма вариантов использования

Ниже приведены спецификации наиболее важных вариантов использования.

Сценарий варианта использования «Добавить мероприятие» представлен на рисунке 7.

Вариант использования: Добавить мероприятие
ID: 1
Краткое описание: Пользователь добавляет мероприятие в систему
Главные актеры: Глава, Председатель
Предусловие: 1. Пользователь прошел авторизацию 2. У пользователя роль «глава» или «председатель»
Основной поток: 1. Нажать кнопку «Добавить мероприятие» 2. Заполнить данные о мероприятии 3. Сохранить
Постусловие: 1. Мероприятие создано и добавлено в систему

Рисунок 7 – Сценарий вариант использования «Добавить мероприятие»

Сценарий вариант использования «Добавить задачу» представлен на рисунке 8.

Вариант использования: Добавить задачу
ID: 2
Краткое описание: Пользователь добавляет задачу в систему
Главные актеры: Глава, Председатель
Предусловие: 1. Пользователь прошел авторизацию 2. У пользователя роль «глава» или «председатель»
Основной поток: 1. Нажать кнопку «Добавить задачу» 2. Заполнить данные о задаче 3. Назначить исполнителя задачи 4. Привязать задачу к мероприятию 5. Сохранить
Постусловие: 1. Задача создана и добавлена в систему

Рисунок 8 – Сценарий варианта использования «Добавить задачу»

Сценарий варианта использования «Сгенерировать план мероприятий на следующий месяц» представлен на рисунке 9.

Вариант использования: Сгенерировать план мероприятий на следующий месяц
ID: 3
Краткое описание: Пользователь получает документ с запланированными мероприятиями на следующий месяц
Главные актеры: Председатель
Предусловие: 1. Пользователь прошел авторизацию 2. У пользователя роль «председатель» 3. В системе существует хотя бы одно запланированное мероприятие на следующий месяц
Основной поток: 1. Нажать кнопку «Создать план на следующий месяц» 2. Система находит все запланированные мероприятия на следующий месяц 3. Система создает pdf документ с мероприятиями
Постусловие: 1. Пользователь получил pdf документ с запланированными мероприятиями на следующий месяц

Рисунок 9 – Сценарий варианта использования «Сгенерировать план мероприятий на следующий месяц»

В результате анализа функциональных требований были выделены варианты использования системы. Для основных вариантов использования в разрабатываемом веб-приложении были представлены спецификации, который отображают последовательность действий пользователя и системы.

1.4 Анализ нефункциональных требований

Анализ нефункциональных требований при разработке веб-приложения для планирования деятельности ССС факультета проводился с целью выбрать такой набор технологий и архитектурных приемов, который обеспечил бы:

- гибкость;
- масштабируемость;
- отказоустойчивость;
- производительность;
- удобство поддержки.

Эти требования выведены на основе особенностей предметной области: система будет обслуживать несколько ролей пользователей с разными правами, работать одновременно с множеством сущностей и должна сохранять целостность при высокой нагрузке и росте числа пользователей.

Опираясь на все поставленные требования, модульный монолит является наилучшим решением, сочетая в себе как особенности монолита (простота разработки и поддержки, упрощенное развертывание), так и особенности микросервисов (модульность, независимость отдельных компонентов системы).

Ниже приводится анализ каждого из нефункциональных требований и обоснование модульного монолита в качестве архитектуры веб-приложения.

Гибкость – модульный монолит позволяет вносить изменения и расширять функционал без серьезных изменений уже существующих модулей.

Масштабируемость – при росте нагрузки можно горизонтально масштабировать всю систему или отдельные модули, сохраняя единую кодовую базу.

Отказоустойчивость – модули изолированы в собственных пространствах имен и слоев, поэтому из-за сбоев в одном модуле не «падает» вся система, а обработка ошибок остается локальной.

Производительность – модульный монолит минимизирует межпроцессное взаимодействие и работает в одном процессе, что снижает накладные расходы на сереализацию и сетевые запросы.

Поддерживаемость – четкое разделение на модули, слои и единые стандарты кода упрощают навигацию и код-ревью.

1.5 Планирование разработки и оценка бюджета

Принципиальным отличием разработки программного обеспечения от других отраслей производства являются большие затраты на разработку продуктов, и относительно низкие затраты на дальнейшее тиражирование, поэтому существенное значение приобретает планирование проектов [9].

Прежде чем приступить к разработке веб-приложения, необходимо провести планирование разработки и оценку бюджета. Планирование производится с целью управления процессом разработки, при котором работа будет завершена в указанный срок. Планирование и оценка бюджета выполнены в программе Microsoft Office Project – программное обеспечение для управления проектами. Оно помогает планировать, контролировать и управлять задачами и ресурсами для успешного выполнения проектов [10].

После определения этапов разработки приложения была проведена организация процесса создания программного обеспечения и выполнен расчет предполагаемых финансовых расходов. Для визуального отображения последовательности и сроков выполнения работ применена диаграмма Ганта.

Диаграмма Ганта – это инструмент, позволяющий визуализировать и управлять проектами, структурировать их выполнение и видеть общую картину задач, как личных, так и организации [11].

Созданная диаграмма Ганта, представлена на рисунке Г.1 в приложении Г.

При планировании разработки было выделено 30 задач. За дату начала разработки было выбрано 8 апреля 2025 г. Согласно полученной диаграмме, окончание работа по проекту запланировано на 5 июня 2025 г.

Для оценки занятости трудовых ресурсов программа сгенерировала отчет, представленный на рисунке Г.2 в приложении Г.

На основании отчета по использованию трудовых ресурсов установлено, что суммарная занятость разработчика за период с 8 апреля 2025 г. по 5 июня 2025 г. при пятидневной рабочей неделе составила 344 часа.

Для расчета бюджета был сформирован отчет о совокупных расходах на реализацию проекта. Данный отчет приведен на рисунке Г.3 в приложении Г.

Согласно полученным данным, для разработки веб-приложения для планирования деятельности ССС факультета потребуется 172000 рублей.

2 Проектирование веб-приложения для планирования деятельности совета студенческого самоуправления факультета

2.1 Архитектура приложения

После проведенного анализа функциональных и нефункциональных требований веб-приложения для планирования деятельности ССС факультета была выбрана архитектура на основе модульного монолита, которая придерживается принципов MVC. Основными факторами, определившими выбор архитектуры, стали:

- необходимость четкого разделения бизнес-логики по модулям;
- единая база данных;
- отсутствие критической потребности в горизонтальном масштабировании на этапе внедрения;
- желание упростить начальное развертывание, сопровождение и отладку;
- возможность в будущем выделить отдельные модули в микросервисы при росте нагрузки.

На рисунке 10 представлена диаграмма развертывания веб-приложения, которая отображает выбранную архитектуру.

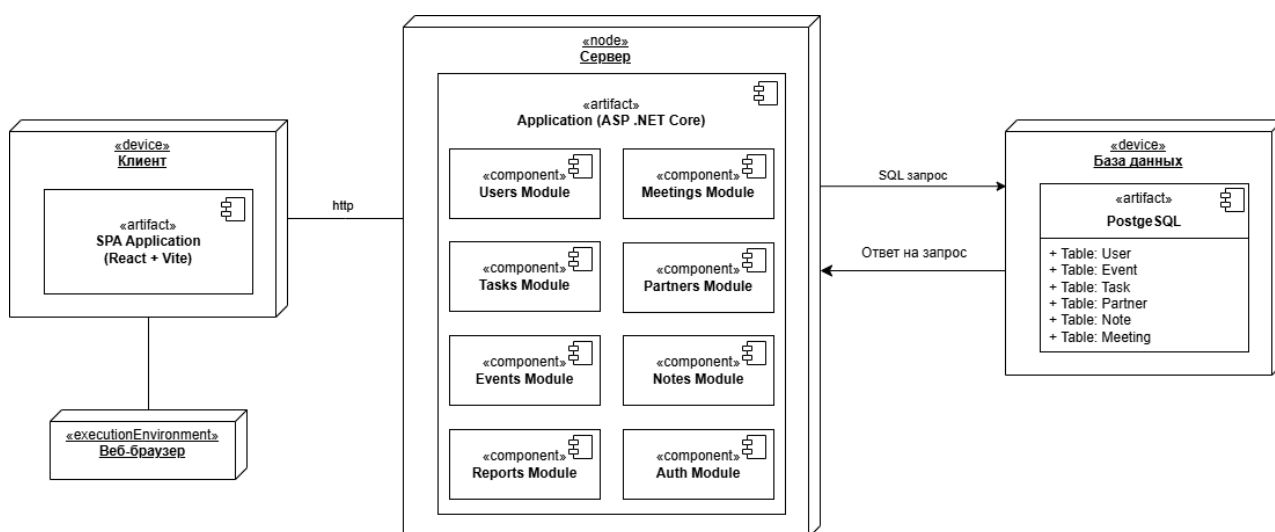


Рисунок 10 – Архитектура веб-приложения для планирования деятельности ССС факультета

Модульный монолит является отличной вариацией как подмножество однопроцессорного монолита: один процесс состоит из отдельных модулей, над каждым из которых можно работать независимо, но все они по-прежнему должны объединяться для развертывания [12].

Серверная часть приложения выполнена в виде восьми модулей, которые могут взаимодействовать между собой. Каждый модуль представлен в виде изолированного набора контроллеров, сервисов и моделей. Контроллер принимает HTTP-запросы, извлекает данные из тела запроса, передает из в сервис и формирует HTTP-ответы. Сервис выполняет бизнес-логику и общается с моделью и базой данных. Модель описывает структуру таблицы в БД (базе данных), а также содержит аннотации валидации и ограничения.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Концептуальная модель данных – это абстрактное представление предметной области, которое описывает объекты, их свойства и взаимосвязи между ними. Она не зависит от конкретной системы управления базами данных и служит основой для проектирования логической модели данных [13].

Для проектирования концептуальной модели данных необходимо выделить список всех основных сущностей предметной области разрабатываемого приложения. Описание сущностей представлено в таблице 3.

Таблица 3 – Сущности предметной области

Название сущности	Идентификатор сущности	Назначение сущности
Пользователь	User	Содержит сведения о пользователях
Мероприятие	Event	Содержит сведения о мероприятиях
Задача	Task	Содержит сведения о задачах в мероприятии
Заметка	Note	Содержит сведения о заметках в мероприятиях
Партнер	Partner	Содержит сведения о партнерах
Встреча	Meeting	Содержит сведения о встречах

Графическое представление концептуальной модели представлено на рисунке 10.

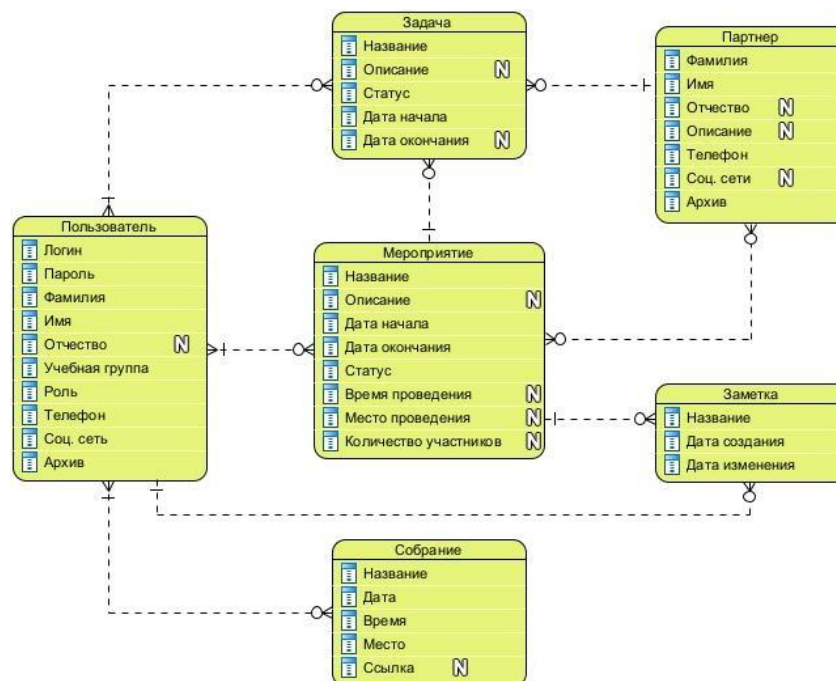


Рисунок 10 – Концептуальная модель базы данных веб-приложения для планирования деятельности CCC факультета

Данная диаграмма была построена в среде Visual Paradigm 17.2 в нотации IDEF1X.

2.2.2 Логическая модель базы данных

После построения концептуальной модели базы данных для веб-приложения для планирования деятельности CCC факультета была спроектирована логическая модель. Логическая модель данных – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила [14].

В рамках данного проектирования были окончательно определены атрибуты для каждой сущности предметной области и их характеристики (приложение Д). Также были определены первичные и внешние ключи, виды связей между сущностями.

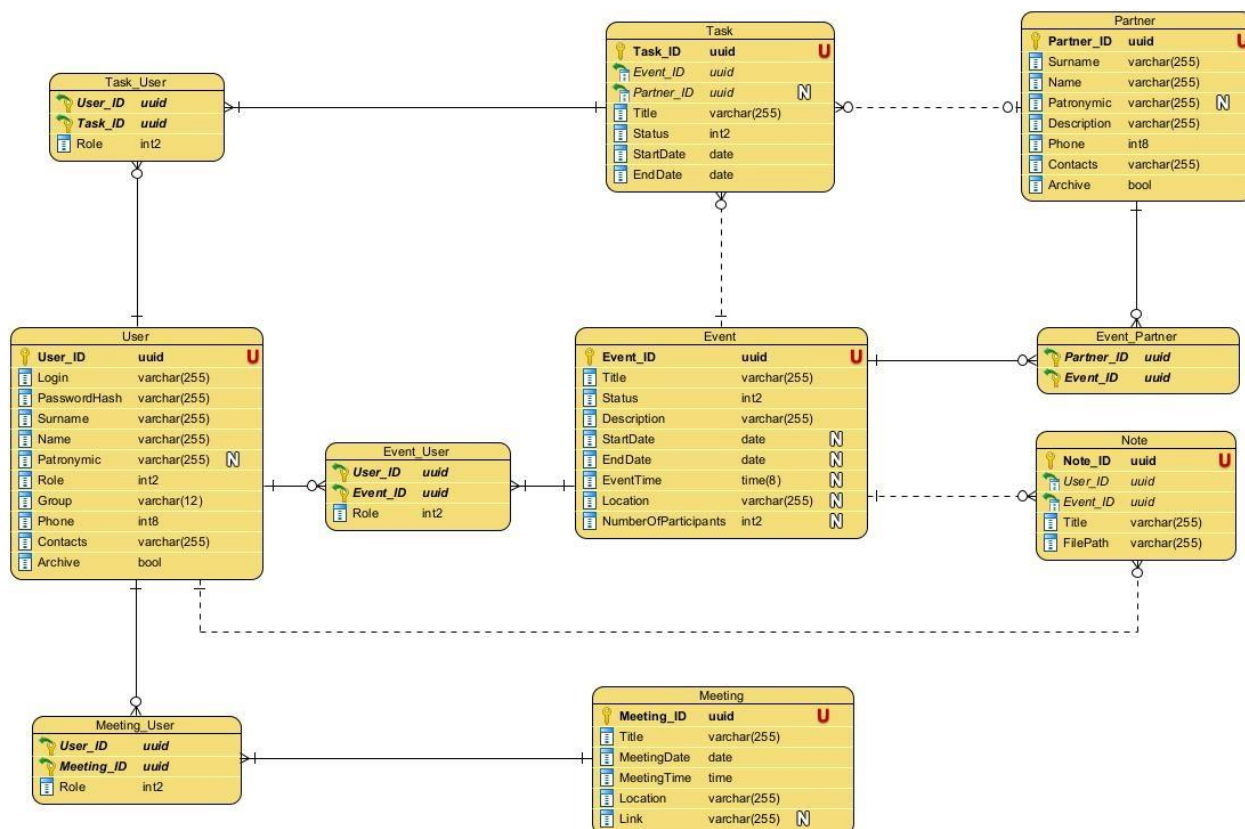


Рисунок 11 – Логическая модель базы данных веб-приложения для планирования деятельности ССС факультета

Спроектированные концептуальная и физическая модели станут основой для реализации базы данных.

2.3 Проектирование пользовательского интерфейса

Интерфейс клиентской части приложения должен быть интуитивным и единообразным. Основные его цели – обеспечить быстрый доступ ко всем ключевым функциям и при этом сохранять простоту и понятность для каждой из трех ролей.

Проектируемый интерфейс включает в себя страницу входа и регистрации, а также главную страницу. На главной странице отображаются следующие разделы:

- общая информация о пользователе, его мероприятия, задачи и встречи, а также архив его мероприятий и задач;
- разделы «пользователи», «партнеры», доступные пользователям с ролями глава или председатель, содержат информацию о всех активных и архивных пользователях и партнерах;
- раздел с кнопками для генерации отчетов мероприятий за месяц и генерации планов мероприятий на следующий месяц.

Помимо страниц и разделов в приложении необходимо реализовать компоненты взаимодействия:

- таблица или карточный вид для каждой сущности (пользователи, мероприятия, задачи, партнеры, заметки, встречи);
- модальные окна для детального просмотра, добавления или изменения информации для элементов таблиц и карточных видов;
- на странице заметок используется компонент-представление markdown-файла, который в реальном времени отображает текст пользователя.

В интерфейсе будет реализовано повторное использование элементов (кнопки, формы, таблицы, модальные окна, карточки), которые будут выполнены как переиспользуемые React-компоненты.

Таким образом, спроектированный интерфейс сочетает в себе четкую навигацию, единый стиль и высокую степень переиспользования компонентов, что позволяет быстро адаптировать и расширять систему по мере роста требований [15].

2.4 Проектирование серверной части приложения

При проектировании серверной части веб-приложения была разработана диаграмма классов по шаблону проектирования MVC.

Диаграмма классов для приложения представлена на рисунке Г.4 в приложении Г.

Краткое описание функционального назначения самых значимых классов веб-приложения приведено в таблице 4.

Таблица 4 – Компоненты диаграммы классов веб-приложения

Компонент	Назначение
GLOBAL	Точка входа в программу
Controller	Класс, реализующий обработку пользовательских запросов, взаимодействие между запросами и бизнес-логикой
ApplicationDbContext	Класс, реализующий доступ к БД, запросы и транзакции, миграции схемы БД
AuthService	Класс, предоставляющий бизнес-логику для регистрации и авторизации
UserService	Класс, содержащий методы бизнес-логики и механизмы обработки данных, связанных с пользователями
User	Модель, используемая для отображения данных о пользователе
EventService	Класс, содержащий методы бизнес-логики и механизмы обработки данных, связанных с мероприятиями
Event	Модель, используемая для отображения данных о мероприятии
TaskService	Класс, содержащий методы бизнес-логики и механизмы обработки данных, связанных с задачами
Task	Модель, используемая для отображения данных о задаче
PartnerService	Класс, содержащий методы бизнес-логики и механизмы обработки данных, связанных с партнерами
Partner	Модель, используемая для отображения данных о партнере
NoteService	Класс, содержащий методы бизнес-логики и механизмы обработки данных, связанных с заметками
Note	Модель, используемая для отображения данных о заметке
MeetingService	Класс, содержащий методы бизнес-логики и механизмы обработки данных, связанных с встречами
Meeting	Модель, используемая для отображения данных о встрече
EventReportService	Класс, предоставляющий бизнес-логику для генерации планов мероприятий и отчетов по мероприятиям и пользователям
EventUser	Модель, используемая для отображения данных о пользователе в мероприятии
TaskUser	Модель, используемая для отображения данных о задаче пользователя
EventPartner	Модель, используемая для отображения данных о партнере мероприятия
MeetingUser	Модель, используемая для отображения данных о пользователе встречи

Данные классы в полной мере реализуют все функциональные требования разрабатываемой системы.

3 Реализация веб-приложения для планирования деятельности совета студенческого самоуправления факультета

3.1 Реализация базы данных

Физическая модель БД определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне [16].

Физическая модель базы данных была построена с помощью внутренних средств инструмента pgAdmin, который предназначен для управления базами данных PostgreSQL, она представлена на рисунке Г.5 в приложении Г.

На основе физической модели была реализована база данных на платформе PostgreSQL, которая отвечает за хранение данных о пользователях, мероприятиях, задачах, партнерах, заметках и встречах.

Для создания и актуализации базы данных использовался механизм миграций из ORM EF Core.

3.2 Реализация серверной части приложения

Серверная часть приложения является ключевым компонентом системы и обеспечивает выполнение бизнес-логики, взаимодействие с базой данных, обработку пользовательских запросов и формирование ответов клиентской части.

Серверная часть приложения разработана на языке C# с использованием платформы ASP.NET Core 8.0 и реализует API по архитектурному принципу MVC с применением Entity Framework Core в качестве ORM для взаимодействия с реляционной базой данных PostgreSQL. Фрагмент кода серверной части приложения приведен в приложении Б. Полный код серверной части можно найти на GitHub [17].

Основными задачами серверной части являются:

- выполнение CRUD-операций над данными;
- реализация авторизации и аутентификации пользователей;

- генерация отчетов;
- обеспечение работы бизнес-правил системы;
- взаимодействие с клиентской частью приложения по протоколу HTTP с использованием JSON.

В серверной части приложения реализована модульная монолитная архитектура, которая объединяет все модули в рамках одного приложения, но разделяет их по папкам и пространствам имен, что упрощает сопровождение и масштабирование проекта.

Каждый модуль включает:

- модели данных, определяющие структуру сущностей приложения, используемых для формирования таблиц базы данных средствами EF Core;
- сервисы бизнес-логики, которые реализуют бизнес-логику приложения, выполняют операции с данными, валидацию и обработку ошибок;
- контроллеры API, которые представляют собой точки входа для клиентских запросов, обрабатывают запросы от клиентской части, вызывают соответствующие сервисы и возвращают данные в формате JSON;
- DTO-объекты для передачи данных.

3.3 Реализация клиентской части приложения

Клиентская часть приложения реализует пользовательский интерфейс и обеспечивает взаимодействие пользователей с серверной частью системы посредством HTTP-запросов к REST API. Она отображает данные, полученные с сервера, и позволяет пользователям выполнять различные операции – авторизацию, регистрацию, работу с задачами, мероприятиями, встречами, заметками и отчетами.

Клиентская часть построена на основе современного стека технологий:

- React – библиотека для построения пользовательских интерфейсов;
- TypeScript – язык надстройки над JavaScript с поддержкой статической типизации;
- Vite – инструмент для быстрой сборки и разработки;

– Tailwind CSS – утилитарный CSS-фреймворк для стилизации интерфейса;

– Axios – библиотека для отправки HTTP-запросов.

Фрагмент кода клиентской части приложения приведен в приложении В. Полный код клиентской части можно найти на GitHub [17].

На рисунке 12 представлена диаграмма компонентов клиентской части разрабатываемого приложения.

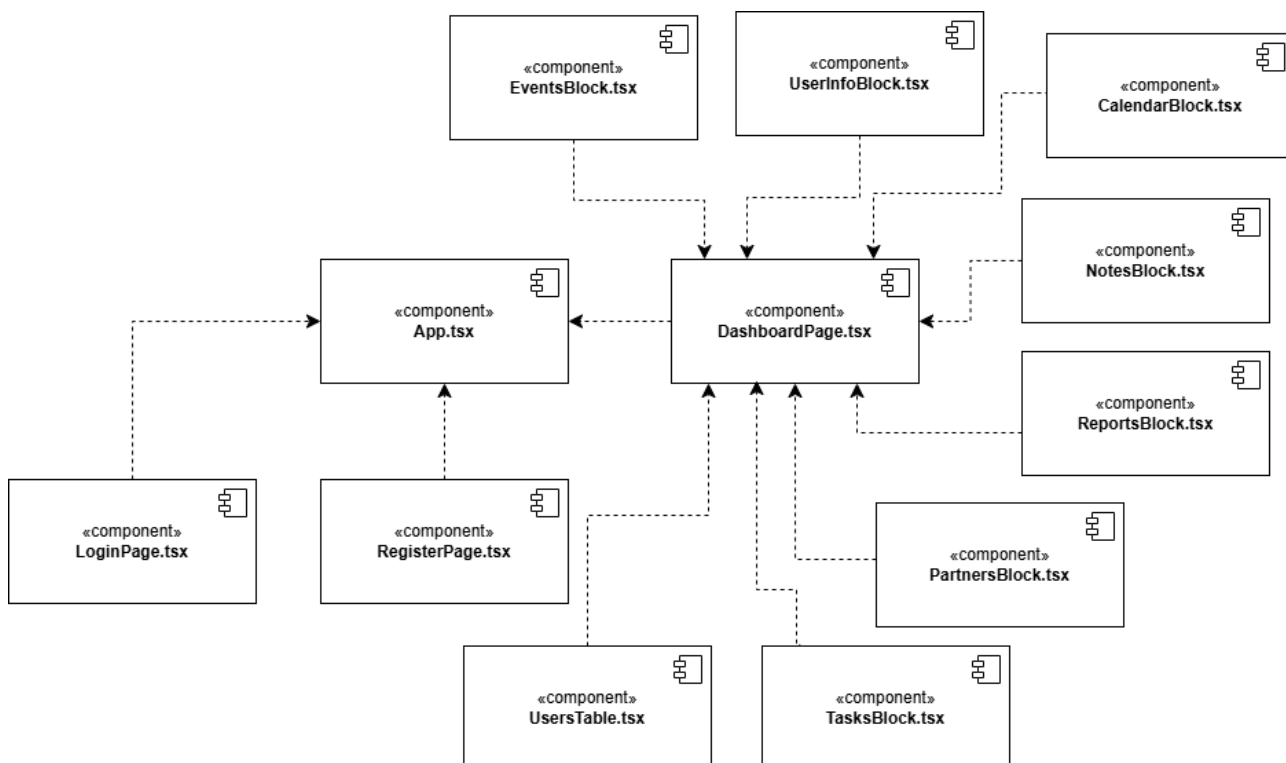


Рисунок 12 – Диаграмма компонентов клиентской части

Описание компонентов клиентской части приложения представлено в таблице 5.

Таблица 5 – Компоненты клиентской части приложения

Компонент	Назначение
App.tsx	Корневой компонент для маршрутизации между страницами
LoginPage.tsx	Компонент, реализующий функционал авторизации
RegisterPage.tsx	Компонент, реализующий функционал регистрации
DashboardPage.tsx	Основной компонент, реализующий функционал главной страницы и страницы пользователя, а также содержащий в себе блоки с другими компонентами

Продолжение таблицы 5

Компонент	Назначение
UserInfoBlock.tsx	Компонент, отображающий личную информацию авторизованного пользователя, а также реализующий функционал генерации отчета о пользователе
EventsBlock.tsx	Компонент, реализующий функционал отображения мероприятий пользователя, создания и изменения мероприятий
TasksBlock.tsx	Компонент, реализующий функционал отображения задач пользователя, создания и изменения задач
PartnersBlock.tsx	Компонент, реализующий функционал отображения партнеров, создания и изменения их
NotesBlock.tsx	Компонент, реализующий функционал отображения заметок, создания и изменения их
CalendarBlock.tsx	Компонент, реализующий функционал отображения встреч пользователя, создания и изменения встреч
ReportsBlock.tsx	Компонент, реализующий функционал генерации отчетов и планов по мероприятиям
UsersTable.tsx	Компонент, реализующий функционал отображения пользователей, изменения их ролей и добавления их в архив

Клиентская часть отправляет HTTP-запросы на серверную часть по REST API. Формат обмена – JSON.

Типовой процесс:

- пользователь выполняет действие (например, нажимает кнопку «Добавить задачу»);
- клиент отправляет POST-запрос на сервер;
- сервер обрабатывает запрос и возвращает ответ;
- клиентская часть получает ответ, обновляет состояние и перерисовывает интерфейс.

Структура проекта обеспечивает модульность, переиспользуемость компонентов и удобство масштабирования. За счет строгой типизации и выделения слоя работы с API обеспечена надежность и устойчивость клиентской части приложения.

3.4 Реализация интерфейса приложения

Пользовательский интерфейс приложения предназначен для обеспечения удобного и интуитивно понятного взаимодействия пользователя с функциональными возможностями системы.

Он отображает актуальную информацию, позволяет выполнять операции CRUD и работать с мероприятиями, задачами, партнерами, встречами, заметками и пользователями.

В соответствии с требованиями к пользовательскому интерфейсу, которые были выработаны в пункте 2.3, были реализованы следующие страницы веб-приложения:

- страница авторизации;
- страница регистрации;
- главная страница.

При открытии приложения пользователь попадает на страницу авторизации (рисунок 13).



Рисунок 13 – Страница авторизации в системе

Основные элементы страницы авторизации:

- 1 – поле для ввода логина пользователя;
- 2 – поле для ввода пароля пользователя;
- 3 – кнопка для перехода на страницу регистрации;
- 4 – кнопка для авторизации в системе.

При нажатии на кнопку «Войти» система ищет пользователя с данным логином в базе данных. Если пользователя с таким логином нет, то система выдает ошибку, иначе пароль введенный пользователь хешируется и проверяется на соответствие хешу пароля, который хранится в базе данных. Если хеши совпадают, то создается JWT-токен, пользователь авторизуется в системе и попадет в свой личный кабинет.

Если же у пользователя еще нет аккаунта в системе, то он может его создать на странице регистрации (рисунок 14)

The image shows a registration form titled "Регистрация в системе" (Registration in the system) for the "Система планирования деятельности ССС факультета" (Faculty Activity Planning System). The form contains several input fields, each with a placeholder text and a corresponding numbered callout arrow pointing to it:

- 1: Login field (placeholder: "Введите логин")
- 2: Name field (placeholder: "Введите имя")
- 3: Surname field (placeholder: "Введите фамилию")
- 4: Patronymic field (placeholder: "Введите отчество")
- 5: Academic group field (placeholder: "Введите номер учебной группы")
- 6: Phone number field (placeholder: "Введите номер телефона")
- 7: VK ID field (placeholder: "Введите VK ID")
- 8: Password field (placeholder: "Введите пароль")
- 9: Confirm password field (placeholder: "Подтвердите пароль")
- 10: Link to login page (text: "Уже есть аккаунт? [Войдите в систему](#)")
- 11: "Зарегистрироваться" (Register) button

Рисунок 14 – Страница регистрации в системе

Основные элементы страницы регистрации:

- 1 – поле для ввода логина пользователя;
- 2 – поле для ввода имени пользователя;
- 3 – поле для ввода фамилии пользователя;
- 4 – поле для ввода отчества пользователя (необязательное);
- 5 – поле для ввода учебной группы пользователя;

- 6 – поле для ввода номера телефона пользователя;
- 7 – поле для ввода VK ID пользователя (необязательное);
- 8 – поле для ввода пароля пользователя;
- 9 – поле для повторения пароля пользователя;
- 10 – кнопка для перехода на страницу авторизации;
- 11 – кнопка для регистрации в системе.

После нажатия кнопки «Зарегистрироваться» система проверяет все введенные поля на корректность, требует вести обязательные незаполненные поля, если такие есть. Затем проверяет базу данных на наличие уже зарегистрированного пользователя с таким логином. Если все проверки прошли успешно, то система создает нового пользователя и сразу же его авторизует в личный кабинет, создавая при этом JWT-токен.

Изначально все пользователи создаются с ролью «активист». Личный кабинет пользователя с ролью «активист» представлен на рисунке 15.

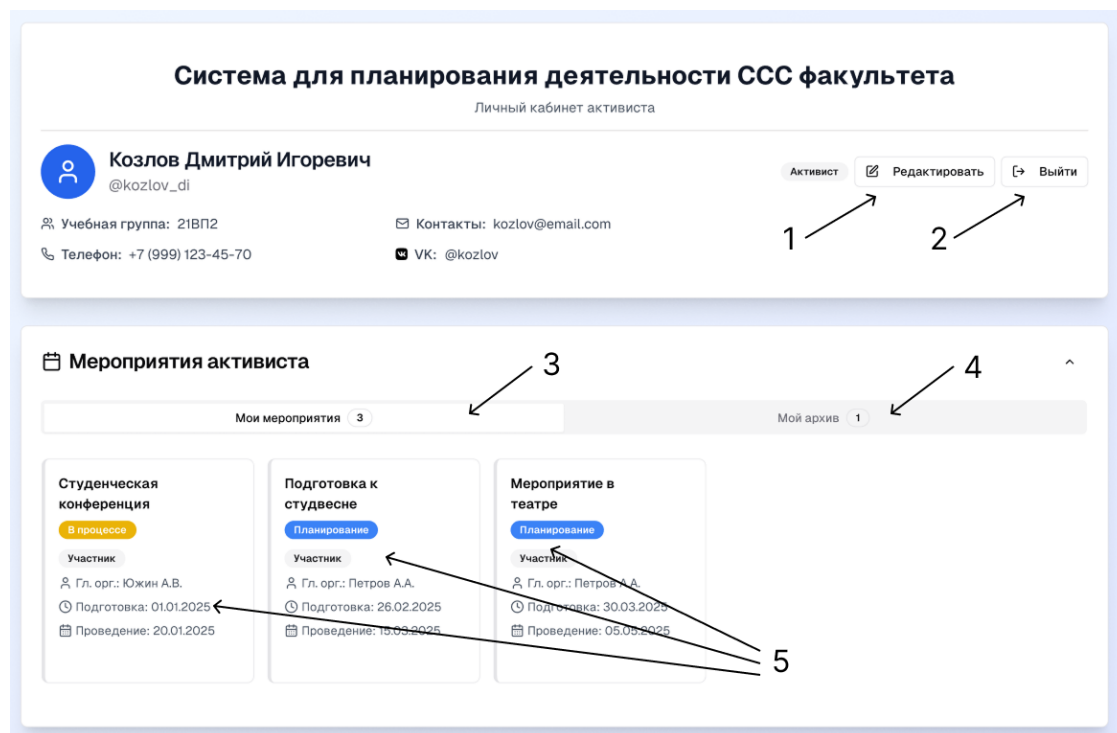


Рисунок 15 – Личный кабинет пользователя с ролью «активист»

Основные элементы страницы личного кабинета пользователя с ролью «активист»:

- 1 – кнопка редактирования личной информации пользователя;

- 2 – кнопка выхода из системы;
- 3 – таб блока мероприятий с активными мероприятиями пользователя;
- 4 – таб блока мероприятий с архивными мероприятиями пользователя;
- 5 – карточки мероприятий пользователя.

При авторизации пользователь попадает в личный кабинет, где отображается его личная информация, мероприятия, в которых он принимает участие или является организатором. При нажатии на карточку мероприятия открывается модульное окно мероприятия с более детальной информацией. Модульное окно мероприятия представлено на рисунке 16.

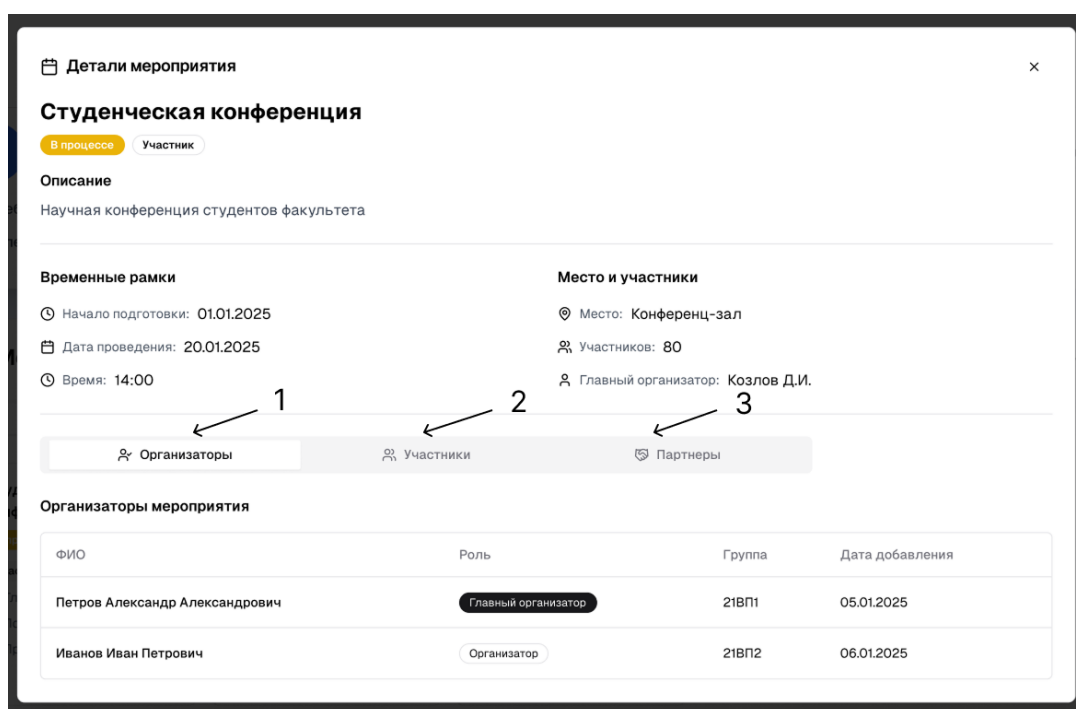


Рисунок 16 – Модульное окно мероприятия

Основные элементы модульного окна мероприятия:

- 1 – таб организаторов;
- 2 – таб участников;
- 3 – таб партнеров.

При открытии модульного окна мероприятия, пользователь может увидеть свою роль в мероприятии и его статус, всю информацию о данном мероприятии, а также всех организаторов, участников среди пользователей и партнеров

данного мероприятия. Главному организатору и организаторам мероприятия также доступен таб с задачами мероприятия.

На рисунке 17 представлено продолжение личного кабинета пользователя с ролью «активист»

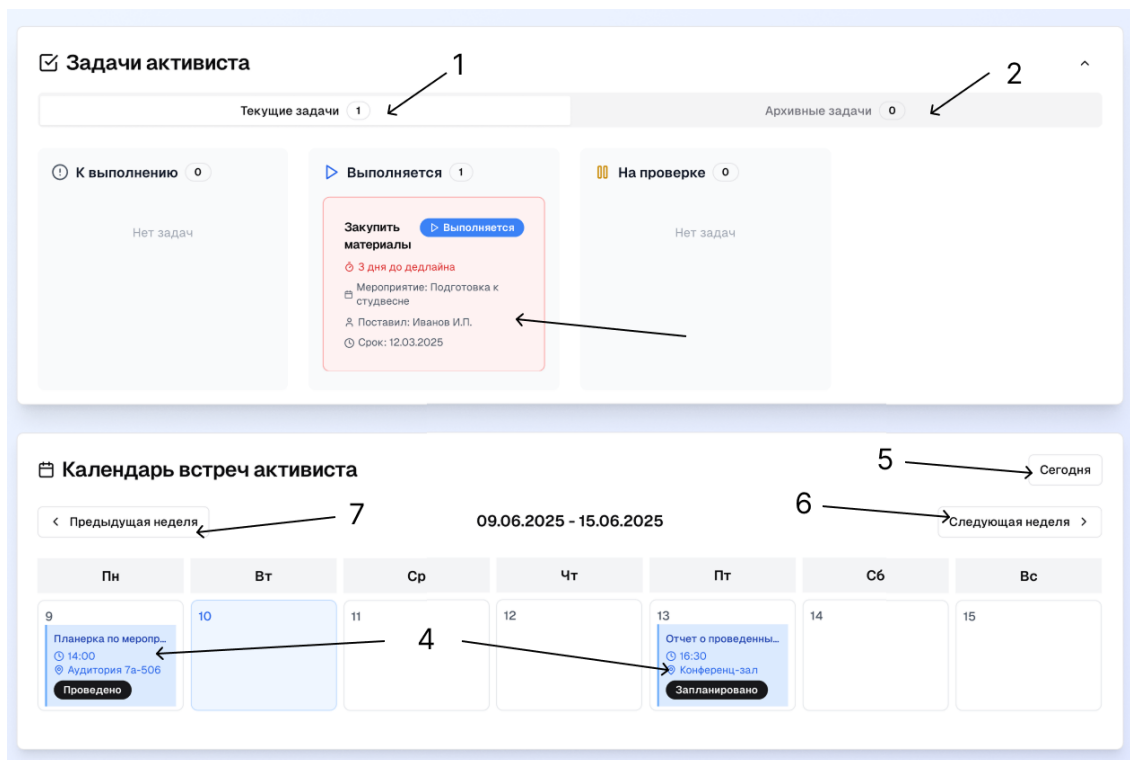


Рисунок 17 – Продолжение личного кабинета пользователя с ролью «активист»

Основные элементы продолжения личного кабинета пользователя с ролью «активист»:

- 1 – таб текущих задач;
- 2 – таб архивных задач;
- 3 – карточка задачи;
- 4 – карточка встречи;
- 5 – кнопка возвращения календаря к неделе с текущей датой;
- 6 – кнопка переключения календаря на следующую неделю;
- 7 – кнопка переключения календаря на предыдущую неделю.

В личном кабинете пользователь с ролью «активист» может просматривать назначенные ему задачи, изменять их статус между «к выполнению»,

«выполняется» и «на проверке», просматривать свои архивные задачи, просматривать свои встречи, в которые его добавили участником.

Пользователь, у которого роль «глава», также может авторизоваться в системе. Внешний вид системы будет идентичен, но будут присутствовать дополнительные блоки и элементы. На рисунке 18 представлен личный кабинет пользователя с ролью «глава».

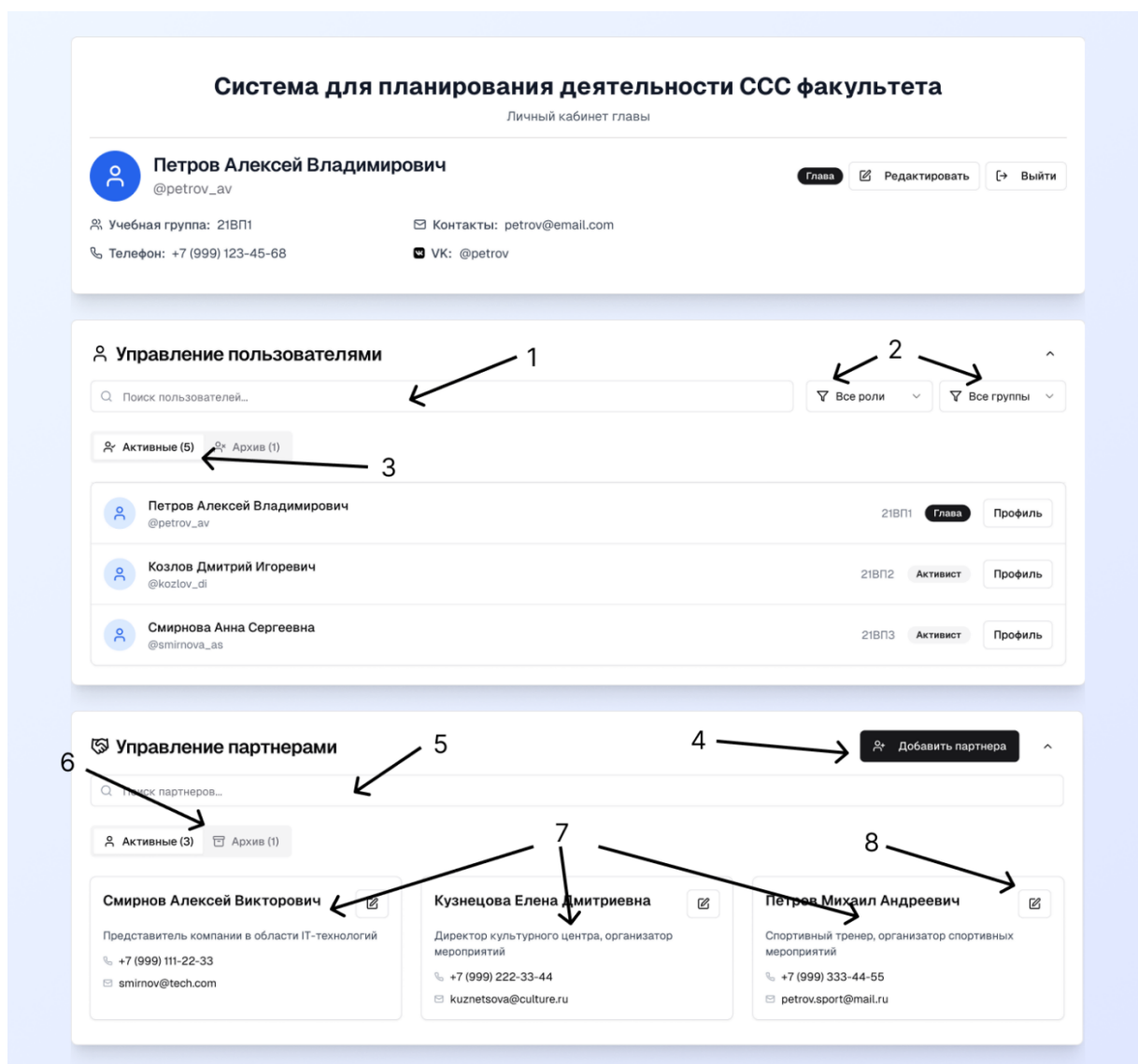


Рисунок 18 – Личный кабинет пользователя с ролью «глава»

Основные элементы личного кабинета пользователя с ролью «глава»:

- 1 – поиск пользователей по фамилии, имени, отчеству;
- 2 – фильтры для поиска пользователей по роли и группе;

- 3 – табы для переключения между активными и архивными пользователями;
- 4 – кнопка для добавления нового партнера;
- 5 – поиск партнеров по фамилии, имени, отчеству;
- 6 – табы для переключения между активными и архивными партнерами;
- 7 – карточки партнеров;
- 8 – кнопка для редактирования информации о партнере.

Основные отличия между пользователями с ролью «активист» и пользователями с ролью «глава» в том, что «главы» могут просматривать всех пользователей, создавать мероприятия и добавлять в них задачи и заметки, создавать партнеров и встречи. В связи с этим интерфейс для данных пользователей выглядит очень похожим, за исключением добавления блоков с пользователями и партнерами, а также добавления кнопок для создания и редактирования мероприятий, задач, заметок и встреч.

Модальное окно создания мероприятия изображено на рисунке 19.

Рисунок 19 – Модальное окно создания мероприятия

Основные элементы модального окна создания мероприятия:

- 1 – поле ввода названия мероприятия;

- 2 – поле ввода описания мероприятия;
- 3 – поле выбора даты начала подготовки;
- 4 – поле выбора даты проведения;
- 5 – поле выбора времени проведения;
- 6 – поле ввода места проведения;
- 7 – кнопка отмены создания мероприятия;
- 8 – кнопка подтверждения создания мероприятия.

Создание мероприятия доступно только пользователям с ролью «глава» или «председатель». После заполнения полей, система проверяет модально окно на наличие незаполненных обязательных полей, если такие есть, то выдает ошибку. Если все необходимые поля заполнены, то мероприятие создается, и его главным организатором автоматически назначается пользователь, который создал мероприятие.

Модальное окно создания задачи представлено на рисунке 20.

The image shows a modal window titled "Создать задачу" (Create Task) with a close button (X) in the top right corner. The form contains the following fields and controls:

- 1** points to the "Название задачи *" (Task Name) input field with placeholder text "Введите название задачи".
- 2** points to the "Срок завершения *" (Completion Date) field, which includes a date input "ДД.ММ.ГГГГ" and a calendar icon.
- 3** points to the "Исполнитель *" (Executor) dropdown menu with placeholder text "Выберите исполнителя".
- 4** points to the "Партнер (необязательно)" (Partner) dropdown menu with placeholder text "Выберите партнера (если нужен)".
- 5** points to the "Отмена" (Cancel) button.
- 6** points to the "Создать задачу" (Create Task) button.

Рисунок 20 – Модальное окно создания задачи

Основные компоненты модального окна создания задачи:

- 1 – поле ввода названия задачи;
- 2 – поле выбора даты завершения;
- 3 – поле выбора исполнителя задачи;

- 4 – поле выбора партнера (необязательное);
- 5 – кнопка отмены создания задачи;
- 6 – кнопка подтверждения создания задачи.

Задача создается в контексте мероприятия, ее постановщиком может быть только главный организатор данного мероприятия (пользователи «глава» или «председатель»). После заполнения полей, система проверяет модально окно на наличие незаполненных обязательных полей, если такие есть, то выдает ошибку. Если все необходимые поля заполнены, то задача создается. Она привязывается к тому мероприятию, в контексте которого создана, постановщиком назначается создатель задачи, исполнителя выбирает пользователь. Исполнителем может быть только пользователь, который в рамках мероприятия обладает роль «организатор» или «главный организатор».

Модальные окна для создания остальных компонентов системы выполнены схожим образом.

Пользователь с ролью «председатель» обладает тем же функционалом, что и «глава». Главные отличия – «председатель» обладает дополнительными элементами в блоке управления пользователями, которые позволяют ему изменять роли у пользователей и отправлять их в архив, а также у «председателя» присутствует свой блок для генерации отчетов и планов.

Блок отчетов и планов изображен на рисунке 21.

Рисунок 21 – Блок отчетов и планов

Основные элементы блока отчетов и планов:

- 1 – поле выбора года отчета;

- 2 – поле выбора месяца отчета;
- 3 – кнопка сформировать отчет;
- 4 – кнопка сформировать план.

После выбора месяца и года составления отчета система находит все завершенные мероприятия, у которых дата завершения была в указанном месяце и году, и формирует отчет в формате pdf файла, который загружается на компьютер пользователя. Для формирования плана мероприятий на следующий месяц система находит все мероприятия, у которых дата завершения стоит в следующем месяце, и составляет план этих мероприятий в формате pdf файла, который также загружается на компьютер пользователя.

Таким образом был разработан интерфейс веб-приложения. React, TypeScript, Vite и Tailwind CSS позволили реализовать его удобным, понятным и адаптивным. Основным достоинством интерфейса являются переиспользуемые компоненты. Создание компонентов высшего порядка позволяет повторно использовать логику путем оборачивания существующих компонентов [18].

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования и метрик кода

В процессе разработки веб-приложения для планирования деятельности ССС факультета особое внимание уделяется обеспечению качества программного обеспечения. Одним из ключевых этапов является тестирование, которое позволяет выявить и устранить ошибки, а также подтвердить соответствие системы заданным требованиям. Выбор подходящих методов тестирования и метрик кода способствует повышению надежности и эффективности программного продукта.

Существует много методов тестирования программного обеспечения, каждый из которых направлен на проверку различных аспектов системы:

- модульное тестирование – тестирование отдельных методов и функций классов, компонентов и модулей, используемых в программном обеспечении;
- интеграционное тестирование – проверка насколько хорошо работают различные модули и сервисы, используемые приложением;
- функциональное тестирование – оценка соответствия работы системы функциональным требованиям;
- сквозное тестирование – копирование поведения пользователя при работе с программным обеспечением в контексте всего приложения;
- тестирование производительности – оценка работоспособности системы при определенной рабочей нагрузке для выявления надежности, скорости, масштабируемости и отзывчивости приложения [19].

Для разработки веб-приложения, предназначенного для планирования деятельности ССС факультета, бы выбраны следующие методы тестирования:

- функциональное тестирование: позволяет убедиться в том, что приложение выполняет все заявленные функции в соответствии с требованиями пользователей;

– модульное тестирование: обеспечивает возможность раннего выявления и устранения ошибок в отдельных компонентах приложения. Автоматизация модульных тестов способствует ускорению процесса разработки и повышению стабильности системы.

Для количественной оценки качества программного кода используются различные метрики, позволяющие анализировать сложность, читаемость и поддерживаемость системы.

Метрика Холстеда основана на анализе количества операторов и операндов в коде. С ее помощью можно оценить такие параметры, как объем программы, сложность и предполагаемое количество ошибок.

Метрика Чепина фокусируется на анализе использования переменных в модуле программы. Она классифицирует переменные по их роли и позволяет оценить информационную сложность модуля.

Выбор указанных метрик обусловлен их способностью предоставлять объективные количественные данные о качестве кода, что способствует принятию обоснованных решений при оптимизации и рефакторинге системы.

4.2 Тестирование веб-приложения для планирования деятельности совета студенческого самоуправления факультета

4.2.1 Функциональное тестирование

После завершения разработки веб-приложения было проведено функциональное тестирование. Рисунки с результатами тестирования приведены в приложении Е. План функционального тестирования представлен в таблице 6.

Таблица 6 – План функционального тестирования

Назначение теста	Описание теста	Результаты теста
Проверка авторизации пользователя с неверным паролем	Пользователь вводит неверный пароль при авторизации	Система выводит ошибку «Неверный логин или пароль» (рисунок Е.1)
Проверка регистрации пользователя с логином, который уже занят другим пользователем	Пользователь регистрируется с логином, который уже занят	Система выводит ошибку «Логин уже занят другим пользователем» (рисунок Е.2)

Продолжение таблицы 6

Проверка отображения личного кабинета пользователя «активист»	Пользователь авторизуется под ролью «активист»	Система отображает личный кабинет пользователя с ролью «активист» (рисунок Е.3)
Проверка отображения личного кабинета пользователя «глава»	Пользователь авторизуется под ролью «глава»	Система отображает личный кабинет пользователя с ролью «глава» (рисунок Е.4)
Проверка отображения личного кабинета пользователя «председатель»	Пользователь авторизуется под ролью «председатель»	Система отображает личный кабинет пользователя с ролью «председатель» (рисунок Е.5)
Проверка изменения информации о пользователе	Пользователь меняет информацию о себе	Система изменяет информацию о пользователе и выводит сообщение «Данные успешно изменены» (рисунок Е.6)
Проверка добавления нового мероприятия	Пользователь добавляет новое мероприятие	Система добавляет мероприятие в базу данных и выводит сообщение «Мероприятие добавлено» (рисунок Е.7)
Проверка просмотра информации о мероприятии	Пользователь просматривает подробную информацию о мероприятии	Система отображает подробную информацию о мероприятии (рисунок Е.8)
Проверка изменения информации о мероприятии	Пользователь меняет информацию о мероприятии	Система изменяет информацию о мероприятии и выводит сообщение «Данные успешно изменены» (рисунок Е.9)
Проверка добавления новой задачи	Пользователь добавляет новую задачу	Система добавляет задачу в базу данных и выводит сообщение «Задача добавлена» (рисунок Е.10)
Проверка просмотра информации о задаче	Пользователь просматривает подробную информацию о задаче	Система отображает подробную информацию о задаче (рисунок Е.11)
Проверка изменения информации о задаче	Пользователь меняет информацию о задаче	Система изменяет информацию о задаче и выводит сообщение «Данные успешно изменены» (рисунок Е.12)
Проверка добавления нового партнера	Пользователь добавляет нового партнера	Система добавляет партнера в базу данных и выводит сообщение «Партнер добавлен» (рисунок Е.13)
Проверка просмотра информации о партнере	Пользователь просматривает подробную информацию о партнере	Система отображает подробную информацию о партнере (рисунок Е.14)
Проверка изменения информации о партнере	Пользователь меняет информацию о партнере	Система изменяет информацию о партнере и выводит сообщение «Данные успешно изменены» (рисунок Е.15)
Проверка добавления новой заметки	Пользователь добавляет новую заметку	Система добавляет заметку в базу данных и выводит сообщение «Заметка добавлена» (рисунок Е.16)

Окончание таблицы 6

Проверка просмотра информации о заметке	Пользователь просматривает подробную информацию о заметке	Система отображает подробную информацию о заметке (рисунок Е.17)
Проверка изменения информации о заметке	Пользователь меняет информацию о заметке	Система изменяет информацию о заметке и выводит сообщение «Данные успешно изменены» (рисунок Е.18)
Проверка добавления новой встречи	Пользователь добавляет новую встречу	Система добавляет встречу в базу данных и выводит сообщение «Встреча добавлена» (рисунок Е.19)
Проверка просмотра информации о встрече	Пользователь просматривает подробную информацию о встрече	Система отображает подробную информацию о встрече (рисунок Е.20)
Проверка изменения информации о встрече	Пользователь меняет информацию о встрече	Система изменяет информацию о встрече и выводит сообщение «Данные успешно изменены» (рисунок Е.21)

Веб-приложение успешно прошло все функциональные тест. Из этого можно сделать вывод, о правильности работы всех функций и полном соответствии функциональным требованиям.

4.2.2 Модульное тестирование

Для проверки корректности работы отдельных модулей разработанного веб-приложения проведено модульное тестирование. Тестирование проводилось для модулей Auth, Users и Events. Тестирование осуществлялось с помощью фреймворка xUnit. xUnit – это фреймворк для написания и запуска автоматических тестов в языке программирования C#. Он предоставляет простой, но мощный интерфейс для создания тестовых наборов, проверки условий и автоматического обнаружения и запуска тестов [20].

Полный код модульных тестов приведен в приложении Ж. План модульного тестирования представлен в таблице 7.

Таблица 7 – План модульного тестирования

Тестируемый компонент	Тест	Ожидаемый результат	Результат теста
AuthService	Должен зарегистрировать нового пользователя	Пройден	Пройден

Продолжение таблицы 7

Тестируемый компонент	Тест	Ожидаемый результат	Результат теста
AuthService	Не должен регистрировать пользователя с занятым логином	Пройден	Пройден
	Должен аутентифицировать пользователя с правильным паролем	Пройден	Пройден
	Не должен аутентифицировать с неправильным паролем	Пройден	Пройден
UserService	Должен заархивировать пользователя, если это делает админ	Пройден	Пройден
	Не должен заархивировать последнего активного админа	Пройден	Пройден
	Не должен позволять обычному пользователю архивировать	Пройден	Пройден
	Должен восстановить архивированного пользователя, если это делает админ	Пройден	Пройден
	Не должен позволять обычному пользователю восстанавливать пользователя	Пройден	Пройден
	Должен менять роль пользователя, если это делает админ	Пройден	Пройден
	Не должен позволять обычному пользователю менять роль пользователя	Пройден	Пройден
	Должен обновлять пользователя, если это делает сам пользователь	Пройден	Пройден
	Не должен позволять пользователю обновлять другого пользователя	Пройден	Пройден
	Должен удалять пользователя, если это делает админ	Пройден	Пройден
	Не должен позволять обычному пользователю удалять другого пользователя	Пройден	Пройден
EventService	Должен вернуть список событий	Пройден	Пройден
	Должен вернуть событие по id	Пройден	Пройден
	Должен вернуть null, если id не найдено	Пройден	Пройден
	Должен создать событие	Пройден	Пройден
	Должен обновить событие	Пройден	Пройден
	Должен добавить участника	Пройден	Пройден
	Не должен добавлять существующего участника	Пройден	Пройден
	Должен удалять участника	Пройден	Пройден

Окончание таблицы 7

Тестируемый компонент	Тест	Ожидаемый результат	Результат теста
EventService	Не должен удалять несуществующего участника	Пройден	Пройден
	Должен добавлять организатора	Пройден	Пройден
	Не должен добавлять существующего организатора	Пройден	Пройден
	Должен удалять организатора	Пройден	Пройден
	Не должен удалять несуществующего организатора	Пройден	Пройден

Результаты модульного тестирования представлены на рисунке 22.

```

StudetCouncilPlannerAPI успешно выполнено (0,3 с) → D:\Projects\StudetCouncilPlanner\StudetCouncilPlannerAPI\bin\Debug\net8.0\StudetCouncilPlannerAPI.dll
StudentCouncilPlannerAPI.Tests успешно выполнено (0,6 с) → bin\Debug\net9.0\StudentCouncilPlannerAPI.Tests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.8.2+699d445a1a (64-bit .NET 9.0.6)
[xUnit.net 00:00:00.06]   Discovering: StudentCouncilPlannerAPI.Tests
[xUnit.net 00:00:00.08]   Discovered: StudentCouncilPlannerAPI.Tests
[xUnit.net 00:00:00.09]   Starting: StudentCouncilPlannerAPI.Tests
[xUnit.net 00:00:01.18]   Finished: StudentCouncilPlannerAPI.Tests
StudentCouncilPlannerAPI.Tests (тест) успешно выполнено (1,9 с)

Сводка теста: всего: 28; сбой: 0; успешно: 28; пропущено: 0; длительность: 1,9 с
Сборка успешно выполнено через 3,7 с

```

Рисунок 22 – Результаты модульного тестирования

Веб-приложение для планирования деятельности CCC факультета успешно прошло все этапы модульного тестирования.

4.3 Оценка качества программного кода

4.3.1 Метрика Холстеда

Для анализа сложности и трудоемкости кода в проекте была использована метрика Холстеда. В качестве примера рассмотрена функция `GetUsersAsync` сервиса `UserService`, реализующая получение и фильтрацию пользователей по определенным параметрам. Код данной функции приведен в листинге Б.4 приложения Б.

Для вычисления метрик Холстеда необходимо определить количество уникальных операторов $n1$ и операндов $n2$, а также общее количество операторов $N1$ и операндов $N2$ в рассматриваемой функции [21].

Операторы функции `GetUsersAsync` и число их вхождений приведены в таблице 8.

Таблица 8 – Операторы функции `GetUsersAsync`

№	Оператор	Число вхождений
1	=	9
2	new	1
3	if	4
4	!	3
5	string.IsNullOrEmpty	2
6	. (доступ к члену)	20
7	Where	4
8	Contains	4
9	==	2
10	!=	1
11	HasValue	3
12	OrderBy	1
13	ThenBy	1
14	Skip	1
15	Take	1
16	ToListAsync	1
17	await	1
18	return	1
19	Select	1
20	ToList	1
21	- (вычитание)	1
22	* (умножение)	1
23		3
24	&&	1
25	$n1 = 23$	$N1 = 44$

Операнды функции `GetUsersAsync` и число их вхождений приведены в таблице 9.

Таблица 9 – Операнды функции `GetUsersAsync`

№	Оператор	Число вхождений
1	dbContext	1
2	Users	1
3	usersQuery	7
4	query	10
5	Search	3
6	Surname	3

Продолжение таблицы 9

№	Оператор	Число вхождений
7	Name	3
8	Patronymic	2
9	Group	4
10	Role	2
11	Archive	2
12	Page	1
13	PageSize	2
14	skip	2
15	u	8
16	ToUserDto	1
17	n2 = 16	N2 = 51

На основе полученных данных были рассчитаны значения метрик Холстеда. Результаты расчетов сведены в таблицу 10.

Таблица 10 – Результаты вычислений по метрике Холстеда

Значение	Результат
Длина реализации	95
Словарь программы	40
Объем программы	505
Трудоемкость	38.3
Усилие	19 352

Анализ полученных значений показывает, что функция `GetUsersAsync` имеет средний уровень сложности и трудоемкости. Объем программы не превышает критические значения, что свидетельствует о приемлемой сложности и поддерживаемости данного участка кода.

4.3.2 Метрика Чепина

Для оценки сложности кода в веб-приложении использована метрика Чепина. В качестве объекта анализа выбран сервис управления мероприятиями (`EventService`), исходный код которого приведен в листинге Б.2 приложения Б.

Для расчета метрики Чепина выделяются все переменные, используемые в компоненте, и они разбиваются на четыре функциональные группы:

- вводимые переменные (Р);
- модифицируемые внутри программы переменные (М);

- переменные, участвующие в управлении работой программного модуля (С);
- временные переменные, не влияющие на поток данных, но присутствующие в коде (Т) [22].

Каждая переменная может одновременно относиться к нескольким группам, а при подсчете учитывается в каждой соответствующей категории. Метрика Чепина рассчитывается по формуле:

$$Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T,$$

где a_1 , a_2 , a_3 , a_4 – весовые коэффициенты для каждой группы переменных.

Согласно автору методики, наибольший вес имеет группа С ($a_3 = 3$), поскольку управляющие переменные наиболее сильно влияют на сложность программы. Для остальных групп используются следующие коэффициенты: $a_1 = 1$, $a_2 = 2$, $a_4 = 0.5$.

В сервисе управления мероприятиями были выделены следующие группы переменных:

- шесть вводимых переменных;
- девять модифицируемых переменных;
- ноль управляющих переменных;
- шесть временных переменных.

Таким образом, формула для расчета метрики Чепина для данного сервиса принимает вид:

$$Q = 1 * 6 + 2 * 9 + 3 * 0 + 0.5 * 6 = 27.$$

Полученное значение метрики Чепина показывает, что сервис EventService обладает средней степенью сложности. Это свидетельствует о приемлемом уровне структурированности и поддерживаемости данного компонента, а также об отсутствии избыточных или чрезмерно сложных фрагментов кода.

Заключение

В рамках выполнения выпускной квалификационной работы разработано веб-приложение для планирования деятельности совета студенческого самоуправления факультета.

На этапе анализа требований проведено исследование предметной области, а также сравнительный анализ существующих веб-приложений, аналогичных по назначению. Выявленные преимущества и недостатки рассмотренных решений позволили определить функциональные и нефункциональные требования к создаваемому веб-приложению.

Произведено планирование разработки и затраченного бюджета. Итоговые денежные затраты на разработку проекта составили 172000 рублей, а затраты трудовых ресурсов составили 344 часов. Начало работы поставлено на 8 апреля, а конец на 5 июня текущего года.

На стадии проектирования определены:

- архитектура приложения на основе модульного монолита, реализующего принципы MVC;
- концептуальная и логическая модели базы данных;
- пользовательский интерфейс, включающий страницы регистрации, авторизации, личного кабинета для разных ролей пользователей.

Реализация веб-приложения включала разработку:

- серверной части на языке C# с использованием ASP.NET Core 8.0 и Entity Framework Core;
- клиентской части на TypeScript с React, Vite и Tailwind CSS;
- базы данных в PostgreSQL.

Разработка выполнялась с учетом принципов многослойной архитектуры, где серверная часть разделена на контроллеры, сервисы и модели. Было организовано подключение аутентификации по JWT-токенам, формирование отчетов в формате PDF и работа с файлами Markdown.

На этапе тестирования проведено функциональное и модульное тестирование приложения. Полученные результаты подтвердили корректность работы системы, соответствие функциональным требованиям и приемлемый уровень качества программного кода, что подтверждено расчетом метрик Холстеда и Чепина.

В результате выполненной работы создано современное, масштабируемое, отказоустойчивое и удобное в использовании веб-приложение для планирования деятельности совета студенческого самоуправления факультета. Оно позволяет:

- регистрировать пользователей с различным ролями;
- планировать мероприятия, задачи, встречи;
- вести заметки;
- хранить информацию о партнерах;
- контролировать статус задач;
- формировать планы и отчеты по деятельности.

Разработанное программное обеспечение соответствует заявленным требованиям, реализовано с применением актуальных технологий и отвечает современным требованиям к информационным системам. Приложение может быть успешно использовано студенческими объединениями для планирования и контроля деятельности.

Список используемых источников

1. Бужинская Н.В. Развитие навыков самоорганизации студентов в процессе планирования и проведения мероприятий / Н.В. Бужинская, Е.С. Васева // Профессиональное образование в России и за рубежом. – 2024 – № 3 (55). – С. 57 – 59.
2. Организация студенческих инициатив и мероприятий // РосДиплом [Электронный ресурс]. URL: https://blog.rosdiplom.ru/rd/pubdiplom/view.aspx?id=2683&utm_source=chatgpt.com (Дата обращения: 14.05.2025).
3. Программное обеспечение для управления проектами // Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Программное_обеспечение_для_управления_проектами (Дата обращения: 14.05.2025).
4. Знакомство с Trello // Trello [Электронный ресурс]. URL: <https://trello.com/ru/tour> (Дата обращения: 14.05.2025).
5. Что можно делать с помощью Календаря // Центр обучения Google Workspace [Электронный ресурс]. URL: <https://support.google.com/a/users/answer/9302892?hl=ru> (Дата обращения: 14.05.2025).
6. Все функции Asana // Asana guide [Электронный ресурс]. URL: <https://asana.com/ru/guide/help/faq/all-asana-features> (Дата обращения: 14.05.2025).
7. Возможности таск-менеджера LeaderTask // LeaderTask [Электронный ресурс]. URL: <https://www.leadertask.ru/features> (Дата обращения: 14.05.2025).
8. Арлоу Д. UML 2 и Унифицированный процесс: практический объектно-ориентированный анализ и проектирование / Д. Арлоу, А. Нейштадт – Символ-Плюс, 2007. – 113 с.

9. Полицын С.А. Система планирования проектов разработки программного обеспечения / С.А. Полицын // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2013 – №1. – С. 1.

10. Microsoft Office Project: инструмент для планирования // Microsoft [Электронный ресурс]. URL: <https://www.microsoft.com/ru-ru/microsoft-365/planner/microsoft-planner?market=ru> (дата обращения: 14.05.2025).

11. Все о диаграмме Ганта: как успешно и вовремя завершать проекты // GanttPRO [Электронный ресурс]. URL: <https://blog.ganttpro.com/ru/diagramma-gantta-gantt-chart/> (Дата обращения 14.05.2025).

12. Ньюмен С. От монолита к микросервисам / С. Ньюмен – СПб.: БХВ-Петербург, 2021 – 272 с.

13. Моделирование данных: основные понятия и методы. Что такое модель данных // Денвик [Электронный ресурс]. URL: <https://bi.denvic.ru/blog/modelirovanie-dannykh-osnovnye-ponyatiya-i-metody-cto-takoe-model-dannykh/> (дата обращения 14.05.2025).

14. Моделирование данных: обзор // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/556790/> (дата обращения: 14.05.2025).

15. Тиленс Т.М. React в действии / Т.М. Тиленс – СПб.: Питер, 2019 – 368 с.

16. Физическая модель базы данных // StudFiles [Электронный ресурс]. URL: <https://studfile.net/preview/6337934/> (дата обращения 14.05.2025).

17. StudentCouncilPlannerAPI // GitHub [Электронный ресурс]. URL: <https://github.com/Ivan-Novozhenov-73/StudetCouncilPlannerAPI> (дата обращения 14.05.2025).

18. Как организовать переиспользование компонентов в React? // MediaSoft [Электронный ресурс]. URL: <https://academy.mediasoft.team/article/kak-organizovat-pereispolzovanie-komponentov-v-react-sovety-razrabotchikov-mediasoft/> (дата обращения 14.05.2025).

19. Различные виды тестирования ПО // Atlassian [Электронный ресурс].
URL: <https://www.atlassian.com/ru/continuous-delivery/software-testing/types-of-software-testing> (дата обращения 14.05.2025).
20. Что такое xUnit в C#? // itProger [Электронный ресурс].
URL: <https://itproger.com/spravka/csharp/xunit> (дата обращения 14.05.2025).
21. Аверьянов А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Изв. вузов. Приборостроение. – 2019 – Т. 62, № 11. – С. 970 - 975.
22. Бахтизин В.В. Метрология, стандартизация и сертификация в информационных технологиях / В.В. Бахтизин, Л.А. Глухова, С.Н. Неборский. – Минск:БГУИР, 2013. – 60 с.

Приложение А
(справочное)
Глоссарий

Совет студенческого самоуправления – это выборный орган внутри вуза или факультета, состоящий из студентов, который занимается организацией внеучебной деятельности, поддержкой студенческих инициатив, защитой прав и интересов студентов, а также координацией мероприятий и проектов.

Веб-приложение – это программное обеспечение, доступное пользователям через веб-браузер и работающее по технологии клиент-сервер.

MVC – это архитектурный паттерн проектирования, разделяющий приложение на три компонента: Model, View, Controller.

Entity Framework Core – это ORM-фреймворк для .NET, обеспечивающий взаимодействие между объектами приложения и реляционной базой данных.

JWT – способ безопасной передачи утверждений между участниками в виде JSON-объекта с цифровой подписью.

REST API – это архитектурный стиль взаимодействия компонентов информационных систем, использующий стандартный HTTP-запросы.

ORM – подход к работе с реляционными базами данных, при котором таблицы отображаются на объекты программного кода.

CRUD – это аббревиатура четырех базовых операций, выполняемых над данными в информационных системах: Create, Read, Update, Delete.

JSON – текстовый формат обмена данными между сервером и клиентом, удобный для человека и легко обрабатываемый программами.

DTO – специальный объект, предназначенный для передачи данных между слоями приложения или между клиентом и сервером.

Приложение Б
(обязательное)
Листинг серверной части приложения

Листинг Б.1 – Файл Program.cs

```

using Microsoft.EntityFrameworkCore;
using StudetCouncilPlannerAPI.Data;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using StudetCouncilPlannerAPI.Services;
using Microsoft.OpenApi.Models;
using QuestPDF.Infrastructure;

QuestPDF.Settings.License = LicenseType.Community;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "Student Council
API", Version = "v1" });
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        Description = "JWT Authorization header using the Bearer
scheme. Example: 'Bearer {token} '",
        Name = "Authorization",
        In = ParameterLocation.Header,
        Type = SecuritySchemeType.Http,
        Scheme = "bearer"
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
            },
            []
        }
    });
});

builder.Services.AddRouting(options =>
{
    options.LowercaseUrls = true;
});

builder.Services.AddControllers();

```

```

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddScoped<AuthService>();
builder.Services.AddScoped<UserService>();
builder.Services.AddScoped<EventService>();
builder.Services.AddScoped<TaskService>();
builder.Services.AddScoped<PartnerService>();
builder.Services.AddScoped<NoteService>();
builder.Services.AddScoped<MeetingService>();
builder.Services.AddScoped<EventReportService>();

// JWT
var jwtSettings = builder.Configuration.GetSection("Jwt");
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new
TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = jwtSettings["Issuer"],
        ValidAudience = jwtSettings["Audience"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["Key"])),
        RoleClaimType =
"http://schemas.microsoft.com/ws/2008/06/identity/claims/role"
    };
});
Console.WriteLine($"JWT KEY = {jwtSettings["Key"]}");

var app = builder.Build();

using (var scope = app.Services.CreateScope())
{
    var db =
scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();
    db.Database.Migrate();
}

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{

```

```

app.UseSwagger();
app.UseSwaggerUI(c =>
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "Student
Council API V1");
    c.RoutePrefix = "swagger"; // Swagger будет доступен по
/swagger
});
}

app.UseHttpsRedirection();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

app.Run();

```

Листинг Б.2 – Файл EventService.cs

```

using StudetCouncilPlannerAPI.Models.Entities;
using StudetCouncilPlannerAPI.Models.Dtos;
using Microsoft.EntityFrameworkCore;
using StudetCouncilPlannerAPI.Data;
using StudetCouncilPlannerAPI.Interfaces;

namespace StudetCouncilPlannerAPI.Services
{
    public class EventService(ApplicationDbContext context) :
    IEventService
    {
        // Получить список мероприятий (с фильтрацией, пагинацией,
        сортировкой)
        public async Task<List<EventShortDto>>
        GetEventsAsync(EventListQueryDto query)
        {
            var eventsQuery = context.Events.AsQueryable();

            if (!string.IsNullOrEmpty(query.Search))
                eventsQuery = eventsQuery.Where(e =>
e.Title.Contains(query.Search) ||
e.Description.Contains(query.Search));

            if (query.Status.HasValue)
                eventsQuery = eventsQuery.Where(e => e.Status ==
query.Status.Value);

            if (query.IsActual.HasValue)
            {
                if (query.IsActual.Value)
                    eventsQuery = eventsQuery.Where(e => e.Status
>= 0 && e.Status <= 3);
                else
                    eventsQuery = eventsQuery.Where(e => e.Status
== 4 || e.Status == 5);
            }
        }
    }
}

```

```

    }

    eventsQuery = eventsQuery.OrderBy(e => e.EndDate);

    var events = await eventsQuery
        .Skip((query.Page - 1) * query.PageSize)
        .Take(query.PageSize)
        .ToListAsync();

    // Преобразование в DTO
    return events.Select(e => new EventShortDto
    {
        EventId = e.EventId,
        Title = e.Title,
        Status = e.Status,
        Description = e.Description,
        StartDate = e.StartDate,
        EndDate = e.EndDate,
        EventTime = e.EventTime,
        NumberOfParticipants = e.NumberOfParticipants
    }).ToList();
}

// Получить мероприятие по id
public async Task<EventDetailDto?> GetEventByIdAsync(Guid
eventId)
{
    var ev = await context.Events
        .Include(e => e.Tasks)
        .Include(e => e.Notes)
        .Include(e => e.EventUsers).ThenInclude(eu =>
eu.User)
        .Include(e => e.EventPartners).ThenInclude(ep =>
ep.Partner)
        .FirstOrDefaultAsync(e => e.EventId == eventId);

    if (ev == null) return null;

    // Разделение участников и организаторов по роли
    var organizers = ev.EventUsers
        .Where(eu => eu.Role == 1) // 1 – организатор
        .Select(eu => new UserShortDto
        {
            UserId = eu.UserId,
            FullName = eu.User.Surname + " " +
eu.User.Name,
            Role = eu.Role
        }).ToList();

    var participants = ev.EventUsers
        .Where(eu => eu.Role == 0) // 0 – участник
        .Select(eu => new UserShortDto
        {

```

```

        UserId = eu.UserId,
        FullName = eu.User.Surname + " " +
eu.User.Name,
        Role = eu.Role
    }).ToList();

    return new EventDetailDto
    {
        EventId = ev.EventId,
        Title = ev.Title,
        Status = ev.Status,
        Description = ev.Description,
        StartDate = ev.StartDate,
        EndDate = ev.EndDate,
        EventTime = ev.EventTime,
        Location = ev.Location,
        NumberOfParticipants = ev.NumberOfParticipants,
        Tasks = ev.Tasks.Select(t => new EventTaskShortDto
{ TaskId = t.TaskId, Title = t.Title }).ToList(),
        Notes = ev.Notes.Select(n => new EventNoteShortDto
{ NoteId = n.NoteId, Title = n.Title }).ToList(),
        Organizers = organizers,
        Participants = participants,
        Partners = ev.EventPartners.Select(ep => new
EventPartnerShortDto
        {
            PartnerId = ep.PartnerId,
            Name = ep.Partner.Name + " " +
ep.Partner.Surname
        }).ToList()
    });
}

// Создать мероприятие
public async Task<Guid> CreateEventAsync(EventCreatedDto
dto, Guid creatorUserId)
{
    var ev = new Event
    {
        EventId = Guid.NewGuid(),
        Title = dto.Title,
        Status = 0, // Входящее по умолчанию
        Description = dto.Description,
        StartDate = dto.StartDate,
        EndDate = dto.EndDate,
        EventTime = dto.EventTime,
        Location = dto.Location,
        NumberOfParticipants = dto.NumberOfParticipants,
        EventUsers = new List<EventUser>
        {
            new()
            {
                UserId = creatorUserId,

```

```

        Role = 2 // 2 – главный организатор
    }
    };

    context.Events.Add(ev);
    await context.SaveChangesAsync();
    return ev.EventId;
}

// Изменить мероприятие
public async Task<bool> UpdateEventAsync(Guid eventId,
EventUpdateDto dto)
{
    var ev = await context.Events.FirstOrDefaultAsync(e =>
e.EventId == eventId);
    if (ev == null) return false;

    ev.Title = dto.Title;
    ev.Description = dto.Description;
    ev.Status = dto.Status;
    ev.StartDate = dto.StartDate;
    ev.EndDate = dto.EndDate;
    ev.EventTime = dto.EventTime;
    ev.Location = dto.Location;
    ev.NumberOfParticipants = dto.NumberOfParticipants;

    await context.SaveChangesAsync();
    return true;
}

// Добавить участника
public async Task<bool> AddParticipantAsync(Guid eventId,
Guid userId)
{
    var exists = await context.EventUsers.AnyAsync(eu =>
eu.EventId == eventId && eu.UserId == userId);
    if (exists) return false;

    context.EventUsers.Add(new EventUser
    {
        EventId = eventId,
        UserId = userId,
        Role = 0 // участник
    });
    await context.SaveChangesAsync();
    return true;
}

// Удалить участника
public async Task<bool> RemoveParticipantAsync(Guid
eventId, Guid userId)
{

```

```

        var eu = await
context.EventUsers.FirstOrDefaultAsync(eu => eu.EventId == eventId
&& eu.UserId == userId && eu.Role == 0);
        if (eu == null) return false;

        context.EventUsers.Remove(eu);
        await context.SaveChangesAsync();
        return true;
    }

    // Добавить организатора
    public async Task<bool> AddOrganizerAsync(Guid eventId,
Guid userId)
    {
        var exists = await context.EventUsers.AnyAsync(eu =>
eu.EventId == eventId && eu.UserId == userId);
        if (exists) return false;

        context.EventUsers.Add(new EventUser
        {
            EventId = eventId,
            UserId = userId,
            Role = 1 // организатор
        });
        await context.SaveChangesAsync();
        return true;
    }

    // Удалить организатора
    public async Task<bool> RemoveOrganizerAsync(Guid eventId,
Guid userId)
    {
        var eu = await
context.EventUsers.FirstOrDefaultAsync(eu => eu.EventId == eventId
&& eu.UserId == userId && eu.Role == 1);
        if (eu == null) return false;

        context.EventUsers.Remove(eu);
        await context.SaveChangesAsync();
        return true;
    }

    // Добавить партнера
    public async Task<bool> AddPartnerAsync(Guid eventId, Guid
partnerId)
    {
        var exists = await context.EventPartners.AnyAsync(ep
=> ep.EventId == eventId && ep.PartnerId == partnerId);
        if (exists) return false;

        context.EventPartners.Add(new EventPartner
        {
            EventId = eventId,

```

```

        PartnerId = partnerId
    });
    await context.SaveChangesAsync();
    return true;
}

// Удалить партнера
public async Task<bool> RemovePartnerAsync(Guid eventId,
Guid partnerId)
{
    var ep = await
context.EventPartners.FirstOrDefaultAsync(ep => ep.EventId ==
eventId && ep.PartnerId == partnerId);
    if (ep == null) return false;

    context.EventPartners.Remove(ep);
    await context.SaveChangesAsync();
    return true;
}

// Получить мероприятия пользователя
public async Task<List<EventShortDto>>
GetUserEventsAsync(Guid userId)
{
    var events = await context.EventUsers
        .Where(eu => eu.UserId == userId)
        .Select(eu => eu.Event)
        .OrderByDescending(e => e.StartDate)
        .ToListAsync();

    return events.Select(e => new EventShortDto
    {
        EventId = e.EventId,
        Title = e.Title,
        Status = e.Status,
        Description = e.Description,
        StartDate = e.StartDate,
        EndDate = e.EndDate,
        EventTime = e.EventTime,
        NumberOfParticipants = e.NumberOfParticipants
    }).ToList();
}

// Получить список организаторов мероприятия
public async Task<List<UserShortDto>>
GetOrganizersAsync(Guid eventId)
{
    var organizers = await context.EventUsers
        .Where(eu => eu.EventId == eventId && eu.Role ==
1)
        .Include(eu => eu.User)
        .Select(eu => new UserShortDto
        {

```

```

        UserId = eu.UserId,
        FullName = eu.User.Surname + " " +
eu.User.Name,
        Role = eu.Role
    })).ToListAsync();

    return organizers;
}

// Получить список участников мероприятия
public async Task<List<UserShortDto>>
GetParticipantsAsync(Guid eventId)
{
    var participants = await context.EventUsers
        .Where(eu => eu.EventId == eventId && eu.Role ==
0)
        .Include(eu => eu.User)
        .Select(eu => new UserShortDto
        {
            UserId = eu.UserId,
            FullName = eu.User.Surname + " " +
eu.User.Name,
            Role = eu.Role
        })).ToListAsync();

    return participants;
}

// Получить список партнеров мероприятия
public async Task<List<EventPartnerShortDto>>
GetPartnersAsync(Guid eventId)
{
    var partners = await context.EventPartners
        .Where(ep => ep.EventId == eventId)
        .Include(ep => ep.Partner)
        .Select(ep => new EventPartnerShortDto
        {
            PartnerId = ep.PartnerId,
            Name = ep.Partner.Name + " " +
ep.Partner.Surname
        })).ToListAsync();

    return partners;
}
}
}

```

Листинг Б.3 – TaskService.cs

```

using Microsoft.EntityFrameworkCore;
using StudetCouncilPlannerAPI.Data;

```

```

using StudetCouncilPlannerAPI.Interfaces;
using StudetCouncilPlannerAPI.Models.DTOs;
using StudetCouncilPlannerAPI.Models.Entities;

namespace StudetCouncilPlannerAPI.Services
{
    public class TaskService(ApplicationDbContext context) :
    ITaskService
    {
        // 1. Создание задачи
        public async Task<Guid> CreateTaskAsync(TaskCreateDto dto,
        Guid creatorUserId)
        {
            var task = new Models.Entities.Task
            {
                TaskId = Guid.NewGuid(),
                Title = dto.Title,
                Status = 0,
                StartDate = dto.StartDate,
                EndDate = dto.EndDate,
                EventId = dto.EventId,
                PartnerId = dto.PartnerId,
                TaskUsers = new List<TaskUser>
                {
                    new TaskUser { UserId = creatorUserId, Role =
0 }, // 0 - постановщик
                    new TaskUser { UserId = dto.ExecutorUserId,
Role = 1 } // 1 - исполнитель
                }
            };
            context.Tasks.Add(task);
            await context.SaveChangesAsync();
            return task.TaskId;
        }

        // 2. Изменение задачи (для постановщика)
        public async Task<bool> UpdateTaskAsync(Guid taskId,
        TaskUpdateDto dto, Guid userId)
        {
            var task = await context.Tasks
                .Include(t => t.TaskUsers)
                .FirstOrDefaultAsync(t => t.TaskId == taskId);
            if (task == null) return false;

            var isCreator = task.TaskUsers.Any(tu => tu.UserId ==
userId && tu.Role == 0);
            if (!isCreator) return false;

            task.Title = dto.Title;
            task.StartDate = dto.StartDate;
            task.EndDate = dto.EndDate;
            task.Status = dto.Status;
            task.PartnerId = dto.PartnerId;
        }
    }
}

```

```

        // смена исполнителя
        var executor = task.TaskUsers.FirstOrDefault(tu =>
tu.Role == 1);
        if (executor != null && executor.UserId !=
dto.ExecutorUserId)
        {
            task.TaskUsers.Remove(executor);
            task.TaskUsers.Add(new TaskUser { UserId =
dto.ExecutorUserId, Role = 1, TaskId = taskId });
        }
        await context.SaveChangesAsync();
        return true;
    }

    // 3. Изменение статуса задачи (для исполнителя)
    public async Task<bool> UpdateTaskStatusAsync(Guid taskId,
TaskStatusUpdatedDto dto, Guid userId)
    {
        var task = await context.Tasks
            .Include(t => t.TaskUsers)
            .FirstOrDefaultAsync(t => t.TaskId == taskId);
        if (task == null) return false;

        var isExecutor = task.TaskUsers.Any(tu => tu.UserId ==
userId && tu.Role == 1);
        if (!isExecutor) return false;

        // Исполнитель может только статусы 0, 1, 2
        if (dto.Status < 0 || dto.Status > 2) return false;

        task.Status = dto.Status;
        await context.SaveChangesAsync();
        return true;
    }

    // 4. Назначить/удалить партнера (только постановщик)
    public async Task<bool> SetPartnerAsync(Guid taskId, Guid?
partnerId, Guid userId)
    {
        var task = await context.Tasks
            .Include(t => t.TaskUsers)
            .FirstOrDefaultAsync(t => t.TaskId == taskId);
        if (task == null) return false;

        var isCreator = task.TaskUsers.Any(tu => tu.UserId ==
userId && tu.Role == 0);
        if (!isCreator) return false;

        task.PartnerId = partnerId;
        await context.SaveChangesAsync();
        return true;
    }

```

```

        // 5. Получить задачу по ID (с деталями)
        public async Task<TaskDetailDto?> GetTaskByIdAsync(Guid
taskId)
        {
            var task = await context.Tasks
                .Include(t => t.TaskUsers).ThenInclude(tu =>
tu.User)
                .Include(t => t.Partner)
                .Include(t => t.Event)
                .FirstOrDefaultAsync(t => t.TaskId == taskId);

            if (task == null) return null;

            var creator = task.TaskUsers.FirstOrDefault(tu =>
tu.Role == 0)?.User;
            var executor = task.TaskUsers.FirstOrDefault(tu =>
tu.Role == 1)?.User;

            return new TaskDetailDto
            {
                TaskId = task.TaskId,
                EventId = task.EventId,
                EventTitle = task.Event.Title,
                Title = task.Title,
                StartDate = task.StartDate,
                EndDate = task.EndDate,
                Status = task.Status,
                CreatorUserId = creator?.UserId ?? Guid.Empty,
                CreatorUserFullName = creator != null ?
$"{{creator.Surname}} {{creator.Name}}" : "",
                ExecutorUserId = executor?.UserId ?? Guid.Empty,
                ExecutorUserFullName = executor != null ?
$"{{executor.Surname}} {{executor.Name}}" : "",
                PartnerId = task.PartnerId,
                PartnerName = task.Partner != null ?
$"{{task.Partner.Surname}} {{task.Partner.Name}}" : null
            };
        }

        // 6. Получить задачи пользователя с фильтрами
        public async Task<List<TaskShortDto>>
GetUserTasksAsync(TaskFilterDto filter)
        {
            var query = context.TaskUsers
                .Where(tu => (!filter.UserId.HasValue || tu.UserId
== filter.UserId.Value)
                    && (!filter.UserRole.HasValue || tu.Role
== filter.UserRole.Value))
                .Select(tu => tu.Task)
                .Distinct()
                .AsQueryable();

```

```

        if (filter.Statuses != null && filter.Statuses.Any())
            query = query.Where(t =>
filter.Statuses.Contains(t.Status));
        if (filter.EventId.HasValue)
            query = query.Where(t => t.EventId ==
filter.EventId.Value);
        if (filter.PartnerId.HasValue)
            query = query.Where(t => t.PartnerId ==
filter.PartnerId.Value);

        query = filter.SortByEndDateAsc == true
            ? query.OrderBy(t => t.EndDate)
            : query.OrderByDescending(t => t.EndDate);

        if (filter.Page.HasValue && filter.PageSize.HasValue)
            query = query.Skip((filter.Page.Value - 1) *
filter.PageSize.Value).Take(filter.PageSize.Value);

        return await query.Select(t => new TaskShortDto
        {
            TaskId = t.TaskId,
            Title = t.Title,
            Status = t.Status,
            EndDate = t.EndDate
        }).ToListAsync();
    }

    // 7. Получить задачи мероприятия
    public async Task<List<TaskShortDto>>
GetEventTasksAsync(TaskFilterDto filter)
    {
        var query = context.Tasks.Where(t => t.EventId ==
filter.EventId);

        if (filter.Statuses != null && filter.Statuses.Any())
            query = query.Where(t =>
filter.Statuses.Contains(t.Status));

        query = filter.SortByEndDateAsc == true
            ? query.OrderBy(t => t.EndDate)
            : query.OrderByDescending(t => t.EndDate);

        if (filter.Page.HasValue && filter.PageSize.HasValue)
            query = query.Skip((filter.Page.Value - 1) *
filter.PageSize.Value).Take(filter.PageSize.Value);

        return await query.Select(t => new TaskShortDto
        {
            TaskId = t.TaskId,
            Title = t.Title,
            Status = t.Status,
            EndDate = t.EndDate
        }).ToListAsync();
    }

```

```

    }

    // 8. Получить задачи партнера
    public async Task<List<TaskShortDto>>
    GetPartnerTasksAsync(TaskFilterDto filter)
    {
        var query = context.Tasks.Where(t => t.PartnerId ==
filter.PartnerId);

        if (filter.Statuses != null && filter.Statuses.Any())
            query = query.Where(t =>
filter.Statuses.Contains(t.Status));

        query = filter.SortByEndDateAsc == true
            ? query.OrderBy(t => t.EndDate)
            : query.OrderByDescending(t => t.EndDate);

        if (filter.Page.HasValue && filter.PageSize.HasValue)
            query = query.Skip((filter.Page.Value - 1) *
filter.PageSize.Value).Take(filter.PageSize.Value);

        return await query.Select(t => new TaskShortDto
        {
            TaskId = t.TaskId,
            Title = t.Title,
            Status = t.Status,
            EndDate = t.EndDate
        }).ToListAsync();
    }
}

```

Листинг Б.4 – UserService.cs

```

using Microsoft.EntityFrameworkCore;
using StudetCouncilPlannerAPI.Data;
using StudetCouncilPlannerAPI.Interfaces;
using StudetCouncilPlannerAPI.Models.DTOs;
using StudetCouncilPlannerAPI.Models.Entities;

namespace StudetCouncilPlannerAPI.Services
{
    public class UserService(ApplicationDbContext dbContext) :
    IUserService
    {
        // Получить список пользователей с фильтрацией и
        пагинацией
        public async Task<List<UserDto>>
        GetUsersAsync(UserListQueryDto query)
        {
            var usersQuery = dbContext.Users.AsQueryable();

            if (!string.IsNullOrEmpty(query.Search))
            {

```

```

        usersQuery = usersQuery.Where(u =>
            u.Surname.Contains(query.Search) ||
            u.Name.Contains(query.Search) ||
            (u.Patronymic != null &&
u.Patronymic.Contains(query.Search)) ||
            u.Group.Contains(query.Search)
        );
    }

    if (!string.IsNullOrEmpty(query.Group))
        usersQuery = usersQuery.Where(u => u.Group ==
query.Group);

    if (query.Role.HasValue)
        usersQuery = usersQuery.Where(u => u.Role ==
query.Role);

    if (query.Archive.HasValue)
        usersQuery = usersQuery.Where(u => u.Archive ==
query.Archive);

    var skip = (query.Page - 1) * query.PageSize;

    var users = await usersQuery
        .OrderBy(u => u.Surname)
        .ThenBy(u => u.Name)
        .Skip(skip)
        .Take(query.PageSize)
        .ToListAsync();

    return users.Select(ToUserDto).ToList();
}

// Получить пользователя по id
public async Task<UserDto?> GetUserByIdAsync(Guid userId)
{
    var user = await dbContext.Users.FindAsync(userId);
    return user != null ? ToUserDto(user) : null;
}

// Обновить пользователя
public async Task<bool> UpdateUserAsync(Guid userId,
UserUpdateDto dto, User currentUser)
{
    var user = await dbContext.Users.FindAsync(userId);
    if (user == null)
        return false;

    // Роль 0 и 1 могут менять только себя, 2 — всех
    if (currentUser.Role < 2 && currentUser.UserId !=
userId)
        return false;

```

```

        user.Surname = dto.Surname ?? user.Surname;
        user.Name = dto.Name ?? user.Name;
        user.Patronymic = dto.Patronymic ?? user.Patronymic;
        user.Group = dto.Group ?? user.Group;
        user.Phone = dto.Phone ?? user.Phone;
        user.Contacts = dto.Contacts ?? user.Contacts;

        await dbContext.SaveChangesAsync();
        return true;
    }

    // Архивировать пользователя (роль 2)
    public async Task<bool> ArchiveUserAsync(Guid userId, User
currentUser)
    {
        if (currentUser.Role != 2)
            return false;

        if (currentUser.UserId == userId)
        {
            // Проверка: нельзя архивировать себя, если ты
единственный неархивированный админ
            int activeAdmins = await
dbContext.Users.CountAsync(u => u.Role == 2 && !u.Archive);
            if (activeAdmins <= 1)
                return false;
        }

        var user = await dbContext.Users.FindAsync(userId);
        if (user == null || user.Archive)
            return false;

        user.Archive = true;
        await dbContext.SaveChangesAsync();
        return true;
    }

    // Восстановить пользователя из архива (роль 2)
    public async Task<bool> RestoreUserAsync(Guid userId, User
currentUser)
    {
        if (currentUser.Role != 2)
            return false;

        var user = await dbContext.Users.FindAsync(userId);
        if (user == null || !user.Archive)
            return false;

        user.Archive = false;
        await dbContext.SaveChangesAsync();
        return true;
    }

```

```

        // Изменить роль пользователя (роль 2, с защитой от потери
последнего админа)
        public async Task<bool> ChangeUserRoleAsync(Guid userId,
short newRole, User currentUser)
        {
            if (currentUser.Role != 2)
                return false;

            var user = await dbContext.Users.FindAsync(userId);
            if (user == null || user.Role == newRole)
                return false;

            // Защита: нельзя понизить себя, если ты последний
админ
            if (user.UserId == currentUser.UserId && user.Role ==
2 && newRole != 2)
            {
                int activeAdmins = await
dbContext.Users.CountAsync(u => u.Role == 2 && !u.Archive);
                if (activeAdmins <= 1)
                    return false;
            }

            user.Role = newRole;
            await dbContext.SaveChangesAsync();
            return true;
        }

        // Маппинг User -> UserDto
        private static UserDto ToUserDto(User u) => new()
        {
            UserId = u.UserId,
            Login = u.Login,
            Surname = u.Surname,
            Name = u.Name,
            Patronymic = u.Patronymic,
            Role = u.Role,
            Group = u.Group,
            Phone = u.Phone,
            Contacts = u.Contacts,
            Archive = u.Archive
        };
    }
}

```

Приложение В
(обязательное)
Листинг клиентской части приложения

Листинг В.1 – Файл login-page.tsx

```

"use client"

import type React from "react"

import { useState } from "react"
import { Button } from "@components/ui/button"
import { Input } from "@components/ui/input"
import { Label } from "@components/ui/label"
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@components/ui/card"
import { Checkbox } from "@components/ui/checkbox"
import { LogIn, User, Lock, Eye, EyeOff } from "lucide-react"

interface LoginData {
  login: string
  password: string
  rememberMe: boolean
}

export default function LoginPage() {
  const [formData, setFormData] = useState<LoginData>({
    login: "",
    password: "",
    rememberMe: false,
  })

  const [isLoading, setIsLoading] = useState(false)
  const [showPassword, setShowPassword] = useState(false)
  const [errors, setErrors] = useState<Partial<LoginData>>({})
  const [loginError, setLoginError] = useState("")

  const handleInputChange = (field: keyof LoginData, value: string | boolean) => {
    setFormData((prev) => ({
      ...prev,
      [field]: value,
    }))

    // Очистка ошибок при изменении поля
    if (errors[field]) {
      setErrors((prev) => ({
        ...prev,
        [field]: "",
      }))
    }

    if (loginError) {
      setLoginError("")
    }
  }
}

```

```

const validateForm = (): boolean => {
  const newErrors: Partial<LoginData> = {}

  if (!formData.login.trim()) {
    newErrors.login = "Логин обязателен"
  }

  if (!formData.password.trim()) {
    newErrors.password = "Пароль обязателен"
  }

  setErrors(newErrors)
  return Object.keys(newErrors).length === 0
}

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault()

  if (!validateForm()) {
    return
  }

  setIsLoading(true)
  setLoginError("")

  try {
    console.log("Данные авторизации:", formData)

    await new Promise((resolve) => setTimeout(resolve, 1500))

    if (formData.login === "admin" && formData.password ===
"password") {
      alert("Авторизация успешна!")
    } else {
      setLoginError("Неверный логин или пароль")
    }
  } catch (error) {
    console.error("Ошибка авторизации:", error)
    setLoginError("Произошла ошибка при авторизации. Попробуйте
еще раз.")
  } finally {
    setIsLoading(false)
  }
}

const togglePasswordVisibility = () => {
  setShowPassword(!showPassword)
}

return (
  <div className="min-h-screen flex items-center justify-center
bg-gradient-to-br from-blue-50 to-indigo-100 p-4">
    <Card className="w-full max-w-md shadow-xl">

```

```

    <CardHeader className="text-center space-y-2">
      <div className="mx-auto w-12 h-12 bg-blue-600 rounded-
full flex items-center justify-center">
        <LogIn className="w-6 h-6 text-white" />
      </div>
      <CardTitle className="text-2xl font-bold text-gray-
900">Вход в систему планирования CCC</CardTitle>
      <CardDescription className="text-gray-600">
        Система для планирования деятельности CCC факультета
      </CardDescription>
    </CardHeader>

    <CardContent>
      <form onSubmit={handleSubmit} className="space-y-6">
        {/* Сообщение об ошибке авторизации */}
        {loginError && (
          <div className="p-3 text-sm text-red-600 bg-red-50
border border-red-200 rounded-md">{loginError}</div>
        )}

        {/* Поле логина */}
        <div className="space-y-2">
          <Label htmlFor="login">
            <div className="flex items-center gap-2">
              <User className="w-4 h-4" />
              Логин
            </div>
          </Label>
          <Input
            id="login"
            type="text"
            placeholder="Введите логин"
            value={formData.login}
            onChange={(e) => handleInputChange("login",
e.target.value)}
            className={errors.login ? "border-red-500" : ""}
            autoComplete="username"
          />
          {errors.login && <p className="text-sm text-red-
500">{errors.login}</p>}
        </div>

        {/* Поле пароля */}
        <div className="space-y-2">
          <Label htmlFor="password">
            <div className="flex items-center gap-2">
              <Lock className="w-4 h-4" />
              Пароль
            </div>
          </Label>
          <div className="relative">
            <Input
              id="password"

```

```

        type={showPassword ? "text" : "password"}
        placeholder="Введите пароль"
        value={formData.password}
        onChange={(e) => handleInputChange("password",
e.target.value)}
        className={errors.password ? "border-red-500 pr-
10" : "pr-10"}
        autoComplete="current-password"
      />
      <button
        type="button"
        onClick={togglePasswordVisibility}
        className="absolute right-3 top-1/2 transform -
translate-y-1/2 text-gray-500 hover:text-gray-700"
      >
        {showPassword ? <EyeOff className="w-4 h-4" /> :
<Eye className="w-4 h-4" />}
      </button>
    </div>
    {errors.password && <p className="text-sm text-red-
500">{errors.password}</p>}
  </div>

  {/* Запомнить меня */}
  <div className="flex items-center space-x-2">
    <Checkbox
      id="rememberMe"
      checked={formData.rememberMe}
      onChange={(checked) =>
handleInputChange("rememberMe", checked as boolean)}
    />
    <Label
      htmlFor="rememberMe"
      className="text-sm font-medium leading-none peer-
disabled:cursor-not-allowed peer-disabled:opacity-70"
    >
      Запомнить меня
    </Label>
  </div>

  {/* Кнопка входа */}
  <Button type="submit" className="w-full h-12 text-lg
font-medium" disabled={isLoading}>
    {isLoading ? "Вход..." : "Войти"}
  </Button>

  {/* Дополнительные ссылки */}
  <div className="space-y-4 text-center">
    <div className="text-sm text-gray-600">
      Нет аккаунта?{" "}
      <a href="#" className="text-slate-600 hover:text-
slate-800 font-medium hover:underline">
        Зарегистрироваться

```

```

        </a>
      </div>
    </div>
  </form>
</CardContent>
</Card>
</div>
)
}

```

Листинг В.2 – Файл register-page.tsx

```

"use client"

import type React from "react"

import { useState } from "react"
import { Button } from "@components/ui/button"
import { Input } from "@components/ui/input"
import { Label } from "@components/ui/label"
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@components/ui/card"
import { Separator } from "@components/ui/separator"
import { User, Lock, Phone, Users, Mail, UserCheck } from "lucide-react"

interface RegistrationData {
  login: string
  password: string
  lastName: string
  firstName: string
  middleName: string
  studyGroup: string
  phoneNumber: string
  contacts: string
}

export default function LoginPage() {
  const [formData, setFormData] = useState<RegistrationData>({
    login: "",
    password: "",
    lastName: "",
    firstName: "",
    middleName: "",
    studyGroup: "",
    phoneNumber: "",
    contacts: "",
  })

  const [isLoading, setIsLoading] = useState(false)
  const [errors, setErrors] =
    useState<Partial<RegistrationData>>({})

```

```

const handleInputChange = (field: keyof RegistrationData, value:
string) => {
  setFormData((prev) => ({
    ...prev,
    [field]: value,
  }))

  // Очистка ошибки при изменении поля
  if (errors[field]) {
    setErrors((prev) => ({
      ...prev,
      [field]: "",
    }))
  }
}

const validateForm = (): boolean => {
  const newErrors: Partial<RegistrationData> = {}

  if (!formData.login.trim()) {
    newErrors.login = "Логин обязателен"
  } else if (formData.login.length < 3) {
    newErrors.login = "Логин должен содержать минимум 3 символа"
  }

  if (!formData.password.trim()) {
    newErrors.password = "Пароль обязателен"
  } else if (formData.password.length < 6) {
    newErrors.password = "Пароль должен содержать минимум 6
СИМВОЛОВ"
  }

  if (!formData.lastName.trim()) {
    newErrors.lastName = "Фамилия обязательна"
  }

  if (!formData.firstName.trim()) {
    newErrors.firstName = "Имя обязательно"
  }

  if (!formData.studyGroup.trim()) {
    newErrors.studyGroup = "Учебная группа обязательна"
  }

  if (!formData.phoneNumber.trim()) {
    newErrors.phoneNumber = "Номер телефона обязателен"
  } else if (!/^[\+?[-
9]\d{1,14}$/.test(formData.phoneNumber.replace(/\s/g, ""))) {
    newErrors.phoneNumber = "Введите корректный номер телефона"
  }

  if (!formData.contacts.trim()) {
    newErrors.contacts = "Контакты обязательны"
  }
}

```

```

    }

    setErrors(newErrors)
    return Object.keys(newErrors).length === 0
  }

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault()

  if (!validateForm()) {
    return
  }

  setIsLoading(true)

  try {
    console.log("Данные регистрации:", formData)

    await new Promise((resolve) => setTimeout(resolve, 2000))

    alert("Регистрация успешно завершена!")

    // Очистка формы после успешной регистрации
    setFormData({
      login: "",
      password: "",
      lastName: "",
      firstName: "",
      middleName: "",
      studyGroup: "",
      phoneNumber: "",
      contacts: "",
    })
  } catch (error) {
    console.error("Ошибка регистрации:", error)
    alert("Произошла ошибка при регистрации. Попробуйте еще раз.")
  } finally {
    setIsLoading(false)
  }
}

return (
  <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-blue-50 to-indigo-100 p-4">
    <Card className="w-full max-w-2xl shadow-xl">
      <CardHeader className="text-center space-y-2">
        <div className="mx-auto w-12 h-12 bg-blue-600 rounded-full flex items-center justify-center">
          <UserCheck className="w-6 h-6 text-white" />
        </div>
        <CardTitle className="text-2xl font-bold text-gray-900">Регистрация в системе планирования CCC</CardTitle>
      </CardHeader>
    </Card>
  </div>
)

```

```

    <CardDescription className="text-gray-600">
      Система для планирования деятельности CCC факультета
    </CardDescription>
  </CardHeader>

  <CardContent>
    <form onSubmit={handleSubmit} className="space-y-6">
      {/* Учетные данные */}
      <div className="space-y-4">
        <div className="flex items-center gap-2 text-sm
font-medium text-gray-700">
          <Lock className="w-4 h-4" />
          Учетные данные
        </div>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-
4">
          <div className="space-y-2">
            <Label htmlFor="login">Логин *</Label>
            <Input
              id="login"
              type="text"
              placeholder="Введите логин"
              value={formData.login}
              onChange={(e) => handleInputChange("login",
e.target.value)}
              className={errors.login ? "border-red-500" :
""}
            />
            {errors.login && <p className="text-sm text-red-
500">{errors.login}</p>}
          </div>

          <div className="space-y-2">
            <Label htmlFor="password">Пароль *</Label>
            <Input
              id="password"
              type="password"
              placeholder="Введите пароль"
              value={formData.password}
              onChange={(e) => handleInputChange("password",
e.target.value)}
              className={errors.password ? "border-red-500"
: ""}
            />
            {errors.password && <p className="text-sm text-
red-500">{errors.password}</p>}
          </div>
        </div>

        <Separator />

```

```

    { /* Личные данные */ }
    <div className="space-y-4">
      <div className="flex items-center gap-2 text-sm
font-medium text-gray-700">
        <User className="w-4 h-4" />
        Личные данные
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-
4">
        <div className="space-y-2">
          <Label htmlFor="lastName">Фамилия *</Label>
          <Input
            id="lastName"
            type="text"
            placeholder="Введите фамилию"
            value={formData.lastName}
            onChange={(e) => handleInputChange("lastName",
e.target.value)}
            className={errors.lastName ? "border-red-500"
: ""}
          />
          {errors.lastName && <p className="text-sm text-
red-500">{errors.lastName}</p>}
        </div>

        <div className="space-y-2">
          <Label htmlFor="firstName">Имя *</Label>
          <Input
            id="firstName"
            type="text"
            placeholder="Введите имя"
            value={formData.firstName}
            onChange={(e) =>
handleInputChange("firstName", e.target.value)}
            className={errors.firstName ? "border-red-500"
: ""}
          />
          {errors.firstName && <p className="text-sm text-
red-500">{errors.firstName}</p>}
        </div>
      </div>

      <div className="space-y-2">
        <Label htmlFor="middleName">Отчество</Label>
        <Input
          id="middleName"
          type="text"
          placeholder="Введите отчество (необязательно)"
          value={formData.middleName}
          onChange={(e) => handleInputChange("middleName",
e.target.value)}
        />
      </div>
    </div>

```

```

    </div>
  </div>

  <Separator />

  { /* Учебная информация и контакты */ }
  <div className="space-y-4">
    <div className="flex items-center gap-2 text-sm
font-medium text-gray-700">
      <Users className="w-4 h-4" />
      Учебная информация и контакты
    </div>

    <div className="space-y-4">
      <div className="space-y-2">
        <Label htmlFor="studyGroup">Учебная группа
*</Label>
        <Input
          id="studyGroup"
          type="text"
          placeholder="Например: 21ВП2"
          value={formData.studyGroup}
          onChange={ (e) =>
handleInputChange("studyGroup", e.target.value)}
          className={errors.studyGroup ? "border-red-
500" : ""}
        />
        {errors.studyGroup && <p className="text-sm
text-red-500">{errors.studyGroup}</p>}
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2
gap-4">
        <div className="space-y-2">
          <Label htmlFor="phoneNumber">
            <div className="flex items-center gap-2">
              <Phone className="w-4 h-4" />
              Номер телефона *
            </div>
          </Label>
          <Input
            id="phoneNumber"
            type="tel"
            placeholder="+7 (999) 123-45-67"
            value={formData.phoneNumber}
            onChange={ (e) =>
handleInputChange("phoneNumber", e.target.value)}
            className={errors.phoneNumber ? "border-red-
500" : ""}
          />
          {errors.phoneNumber && <p className="text-sm
text-red-500">{errors.phoneNumber}</p>}
        </div>

```

```

        <div className="space-y-2">
          <Label htmlFor="contacts">
            <div className="flex items-center gap-2">
              <Mail className="w-4 h-4" />
              Контакты *
            </div>
          </Label>
          <Input
            id="contacts"
            type="text"
            placeholder="Email, Telegram, VK и т.д."
            value={formData.contacts}
            onChange={ (e) =>
handleInputChange("contacts", e.target.value)}
            className={errors.contacts ? "border-red-
500" : ""}
          />
          {errors.contacts && <p className="text-sm
text-red-500">{errors.contacts}</p>}
        </div>
      </div>
    </div>
    <Button type="submit" className="w-full h-12 text-lg
font-medium" disabled={isLoading}>
      {isLoading ? "Регистрация..." :
"Зарегистрироваться"}
    </Button>

    <div className="text-center text-sm text-gray-600">
      <p>Поля, отмеченные *, обязательны для
заполнения</p>
    </div>

    <div className="text-center text-sm text-gray-600">Уже
есть аккаунт? Войти</div>
  </form>
</CardContent>
</Card>
</div>
)
}

```

Листинг Б.3 – Файл profile-page.tsx

```

"use client"

import { useState } from "react"
import { Button } from "@components/ui/button"
import { Card, CardContent, CardDescription, CardHeader, CardTitle
} from "@components/ui/card"

```

```

import { Badge } from "@components/ui/badge"
import {
  User,
  Phone,
  Mail,
  Users,
  Edit,
  Logout,
  AlertCircle,
  Calendar,
  Clock,
  FileText,
  Archive,
  BarChart3,
} from "lucide-react"
import UserManagementSection from "../user-management-section"
import ReportsSection from "../reports-section"
import RoleBasedEventsSection from "../role-based-events-section"
import RoleBasedTasksKanban from "../role-based-tasks-kanban"
import RoleBasedMeetingsCalendar from "../role-based-meetings-
calendar"
import EditUserModal from "../edit-user-modal"
import CreateEventModal from "../create-event-modal"
import CreateMeetingModal from "../create-meeting-modal"
import Breadcrumbs from "../components/breadcrumbs"

interface UserProfile {
  id: string
  login: string
  firstName: string
  lastName: string
  middleName?: string
  studyGroup: string
  phoneNumber: string
  contacts: string
  role: "activist" | "head" | "chairman"
}

export default function ProfilePageChairman() {
  const [isEditModalOpen, setIsEditModalOpen] = useState(false)
  const [isCreateEventModalOpen, setIsCreateEventModalOpen] =
    useState(false)
  const [isCreateMeetingModalOpen, setIsCreateMeetingModalOpen] =
    useState(false)

  const getRoleLabel = (role: string) => {
    switch (role) {
      case "activist":
        return "Активист"
      case "head":
        return "Глава"
      case "chairman":
        return "Председатель"
    }
  }

```

```

        default:
            return "Пользователь"
    }
}

const getRoleBadgeVariant = (role: string) => {
    switch (role) {
        case "chairman":
            return "destructive"
        case "head":
            return "default"
        default:
            return "secondary"
    }
}

const handleUserUpdated = (updatedUser: UserProfile) => {
    setUser(updatedUser)
    setIsEditModalOpen(false)
}

const handleEventCreated = (newEvent: any) => {
    console.log("Создано мероприятие:", newEvent)
    setIsCreateEventModalOpen(false)
}

const handleMeetingCreated = (newMeeting: any) => {
    console.log("Создана встреча:", newMeeting)
    setIsCreateMeetingModalOpen(false)
}

return (
    <div className="min-h-screen bg-gradient-to-br from-blue-50
to-indigo-100 p-4">
        <div className="max-w-7xl mx-auto space-y-8">
            {/* Заголовок системы */}
            <div className="text-center py-4">
                <h1 className="text-3xl font-bold text-gray-900">Система
для планирования деятельности CCC факультета</h1>
                <p className="text-gray-600 mt-2">Личный кабинет
председателя</p>
            </div>

            {/* Хлебные крошки */}
            <Breadcrumbs items={[{ label: "Личный кабинет" }]} />

            {/* Быстрые действия */}
            <Card className="shadow-lg">
                <CardHeader>
                    <CardTitle className="text-xl">Быстрые
действия</CardTitle>
                </CardHeader>
                <CardContent>

```

```

        <div className="grid grid-cols-2 md:grid-cols-3
lg:grid-cols-5 gap-4">
            <Button className="h-16 flex flex-col gap-2"
onClick={() => setIsCreateEventModalOpen(true)}>
                <Calendar className="w-6 h-6" />
                <span className="text-xs">Создать
мероприятие</span>
            </Button>
            <Button
                className="h-16 flex flex-col gap-2"
                variant="outline"
                onClick={() => setIsCreateMeetingModalOpen(true)}
            >
                <Clock className="w-6 h-6" />
                <span className="text-xs">Запланировать
встречу</span>
            </Button>
            <Button className="h-16 flex flex-col gap-2"
variant="outline" onClick={() => console.log("Отчеты")}>
                <FileText className="w-6 h-6" />
                <span className="text-xs">Отчеты</span>
            </Button>
            <Button className="h-16 flex flex-col gap-2"
variant="outline" onClick={() => console.log("Архив")}>
                <Archive className="w-6 h-6" />
                <span className="text-xs">Архив</span>
            </Button>
            <Button className="h-16 flex flex-col gap-2"
variant="outline" onClick={() => console.log("Статистика")}>
                <BarChart3 className="w-6 h-6" />
                <span className="text-xs">Статистика</span>
            </Button>
        </div>
    </CardContent>
</Card>

{/* Профиль пользователя */}
<Card className="shadow-xl">
    <CardHeader className="pb-4">
        <div className="flex items-center justify-between">
            <div className="flex items-center gap-4">
                <div className="w-16 h-16 bg-red-600 rounded-full
flex items-center justify-center">
                    <User className="w-8 h-8 text-white" />
                </div>
                <div>
                    <CardTitle className="text-2xl text-gray-900">
                        {user.lastName} {user.firstName}
{user.middleName}
                    </CardTitle>
                    <CardDescription className="text-
lg">@{user.login}</CardDescription>
                </div>
            </div>

```

```

        </div>
        <div className="flex items-center gap-2">
            <Badge
variant={getRoleBadgeVariant(user.role)}>{getRoleLabel(user.role)}
</Badge>
            <Button variant="outline" size="sm" onClick={() =>
setIsEditModalOpen(true)}>
                <Edit className="w-4 h-4 mr-2" />
                Редактировать
            </Button>
            <Button variant="outline" size="sm">
                <LogOut className="w-4 h-4 mr-2" />
                Выйти
            </Button>
        </div>
    </div>
</CardHeader>

    <CardContent>
        <div className="grid grid-cols-1 md:grid-cols-3 gap-
6">
            <div className="space-y-3">
                <div className="flex items-center gap-2 text-gray-
700">
                    <Users className="w-4 h-4" />
                    <span className="font-medium">Учебная
группа:</span>
                    <span>{user.studyGroup}</span>
                </div>
                <div className="flex items-center gap-2 text-gray-
700">
                    <Phone className="w-4 h-4" />
                    <span className="font-medium">Телефон:</span>
                    <span>{user.phoneNumber}</span>
                </div>
            </div>
            <div className="md:col-span-2">
                <div className="flex items-center gap-2 text-gray-
700">
                    <Mail className="w-4 h-4" />
                    <span className="font-medium">Контакты:</span>
                    <span>{user.contacts}</span>
                </div>
            </div>
        </div>

        {/* Информация о правах председателя */}
        <div className="mt-6 p-4 bg-red-50 rounded-lg border
border-red-200">
            <div className="flex items-start gap-3">
                <AlertCircle className="w-5 h-5 text-red-600 mt-
0.5" />
                <div>

```

```

        <h4 className="font-semibold text-red-900 mb-
2">Права председателя</h4>
        <ul className="text-sm text-red-800 space-y-1">
            <li>• Все права главы и активиста</li>
            <li>• Управление ролями пользователей</li>
            <li>• Генерация отчетов о мероприятиях</li>
            <li>• Создание планов мероприятий</li>
            <li>• Архивирование пользователей</li>
            <li>• Полный административный доступ</li>
        </ul>
    </div>
</div>
</div>
</CardContent>
</Card>

{ /* Статистика и аналитика */ }
<Card className="shadow-lg">
    <CardHeader>
        <CardTitle className="flex items-center gap-2">
            <BarChart3 className="w-6 h-6" />
            Статистика системы
        </CardTitle>
    </CardHeader>
    <CardContent>
        <div className="grid grid-cols-2 md:grid-cols-4 gap-
4">
            <div className="text-center p-4 bg-blue-50 rounded-
lg">
                <div className="text-2xl font-bold text-blue-
600">15</div>
                <div className="text-sm text-gray-600">Активных
пользователей</div>
            </div>
            <div className="text-center p-4 bg-green-50 rounded-
lg">
                <div className="text-2xl font-bold text-green-
600">8</div>
                <div className="text-sm text-gray-600">Текущих
мероприятий</div>
            </div>
            <div className="text-center p-4 bg-yellow-50
rounded-lg">
                <div className="text-2xl font-bold text-yellow-
600">23</div>
                <div className="text-sm text-gray-600">Активных
задач</div>
            </div>
            <div className="text-center p-4 bg-purple-50
rounded-lg">
                <div className="text-2xl font-bold text-purple-
600">12</div>

```

```

        <div className="text-sm text-gray-
600">Запланированных встреч</div>
    </div>
</div>
</CardContent>
</Card>

{/* Управление пользователями */}
<UserManagementSection userRole="chairman" />

{/* Отчеты и планирование */}
<ReportsSection />

{/* Мероприятия */}
<RoleBasedEventsSection userRole="chairman"
userId={user.id} />

{/* Канбан доска задач */}
<RoleBasedTasksKanban userRole="chairman" userId={user.id}
/>

{/* Календарь встреч */}
<RoleBasedMeetingsCalendar userRole="chairman"
userId={user.id} />

{/* Модальные окна */}
<EditUserModal
    isOpen={isEditModalOpen}
    onClose={() => setIsEditModalOpen(false)}
    user={user}
    onUserUpdated={handleUserUpdated}
/>

<CreateEventModal
    isOpen={isCreateEventModalOpen}
    onClose={() => setIsCreateEventModalOpen(false)}
    onEventCreated={handleEventCreated}
/>

<CreateMeetingModal
    isOpen={isCreateMeetingModalOpen}
    onClose={() => setIsCreateMeetingModalOpen(false)}
    onMeetingCreated={handleMeetingCreated}
/>
</div>
</div>
)
}

```

Приложение Г
(рекомендуемое)
Графическая часть

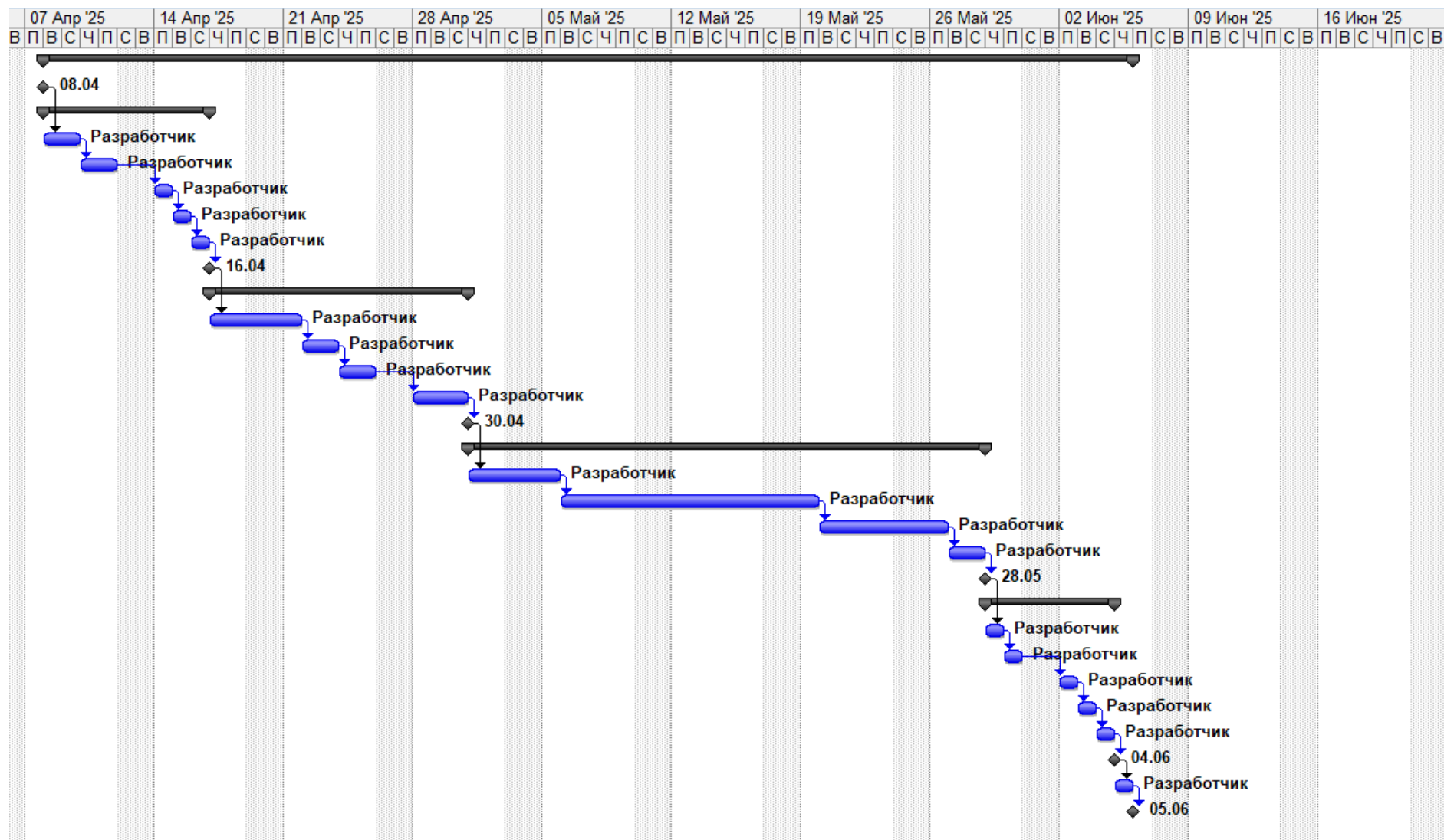


Рисунок Г.1 – Диаграмма Ганта задач с распределением ресурсов

Использование ресурсов от Сб 14.06.25
Веб-приложение ССС

	07.04.25	14.04.25	21.04.25	28.04.25	05.05.25	12.05.25	19.05.25	26.05.25	02.06.25	09.06.25	Итого
Разработчик	32 ч	40 ч	40 ч	40 ч	40 ч	40 ч	40 ч	40 ч	32 ч		344 ч
Исследование предметной области	16 ч										16 ч
Обзор и анализ приложений аналогов	16 ч										16 ч
Анализ функциональных требований		8 ч									8 ч
Анализ нефункциональных требований		8 ч									8 ч
Планирование разработки и оценка бюджета		8 ч									8 ч
Проектирование архитектуры приложения		16 ч	8 ч								24 ч
Проектирование структур данных			16 ч								16 ч
Проектирование пользовательского интерфейса			16 ч								16 ч
Проектирование серверной части приложения				24 ч							24 ч
Реализация базы данных				16 ч	8 ч						24 ч
Реализация серверной части приложения					32 ч	40 ч	8 ч				80 ч
Реализация клиентской части приложения							32 ч	8 ч			40 ч
Реализация интерфейса								16 ч			16 ч
Выбор методов тестирования и метрик кода								8 ч			8 ч
Функциональное тестирование								8 ч			8 ч
Модульное тестирование									8 ч		8 ч
Оценка кода по метрике Холстеда									8 ч		8 ч
Оценка кода по метрике Чепина									8 ч		8 ч
Оформление отчета									8 ч		8 ч
Итого	32 ч	40 ч	40 ч	40 ч	40 ч	40 ч	40 ч	40 ч	32 ч		344 ч

Рисунок Г.2 – Отчет об использовании трудовых ресурсов

Отчет о бюджете от Сб 14.06.25
Веб-приложение ССС

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты	Базовые	Отклонение
18	Реализация серверной части прило	0,00 Р	Пропорциональное	40 000,00 Р	0,00 Р	40 000,00 Р
19	Реализация клиентской части прило	0,00 Р	Пропорциональное	20 000,00 Р	0,00 Р	20 000,00 Р
11	Проектирование архитектуры прило	0,00 Р	Пропорциональное	12 000,00 Р	0,00 Р	12 000,00 Р
14	Проектирование серверной части пр	0,00 Р	Пропорциональное	12 000,00 Р	0,00 Р	12 000,00 Р
17	Реализация базы данных	0,00 Р	Пропорциональное	12 000,00 Р	0,00 Р	12 000,00 Р
4	Исследование предметной области	0,00 Р	Пропорциональное	8 000,00 Р	0,00 Р	8 000,00 Р
5	Обзор и анализ приложений аналог	0,00 Р	Пропорциональное	8 000,00 Р	0,00 Р	8 000,00 Р
12	Проектирование структур данных	0,00 Р	Пропорциональное	8 000,00 Р	0,00 Р	8 000,00 Р
13	Проектирование пользовательского	0,00 Р	Пропорциональное	8 000,00 Р	0,00 Р	8 000,00 Р
20	Реализация интерфейса	0,00 Р	Пропорциональное	8 000,00 Р	0,00 Р	8 000,00 Р
6	Анализ функциональный требовани	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
7	Анализ нефункциональных требова	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
8	Планирование разработки и оценка	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
23	Выбор методов тестирования и мет	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
24	Функциональное тестирование	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
25	Модульное тестирование	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
26	Оценка кода по метрике Холстеда	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
27	Оценка кода по метрике Чепина	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
29	Оформление отчета	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
2	Начало проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
9	Анализ завершен	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
15	Проектирование завершено	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
21	Реализация завершена	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
28	Тестирование завершено	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
30	Конец проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
		0,00 Р		172 000,00 Р	0,00 Р	172 000,00 Р

Рисунок Г.3 – Отчет об общих затратах проекта

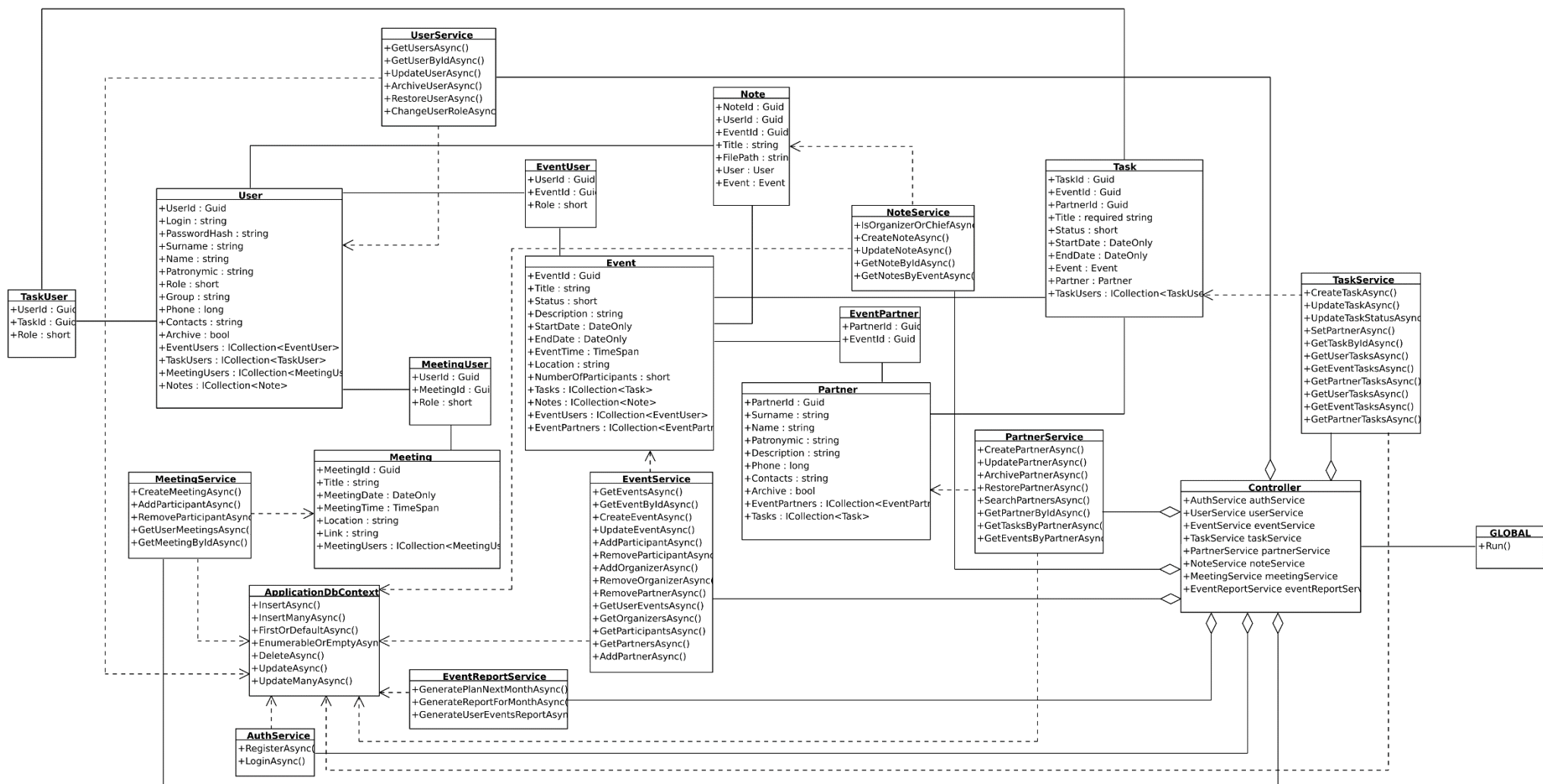


Рисунок Г.4 – Диаграмма классов веб-приложения

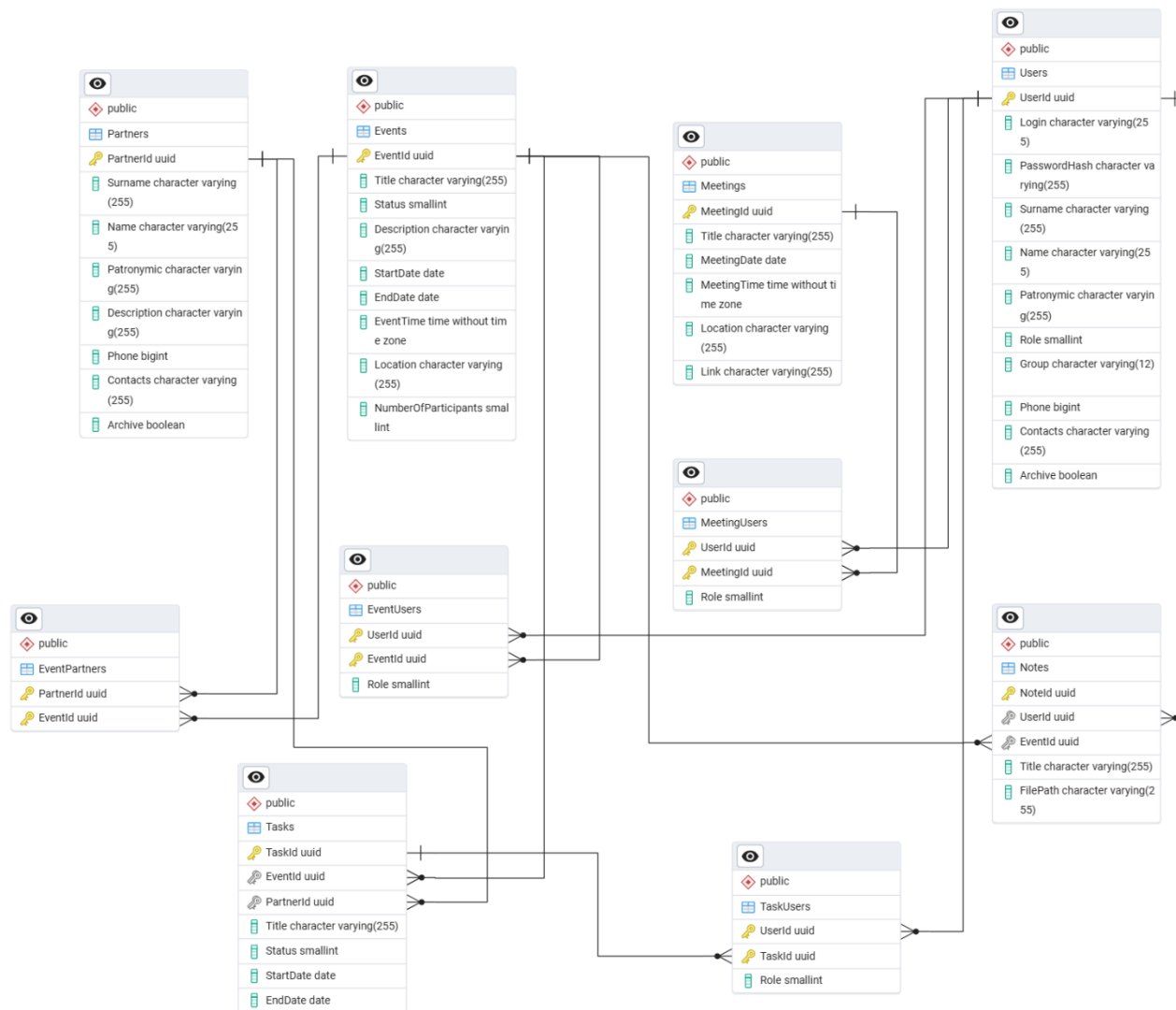


Рисунок Г.5 – Физическая модель базы данных

Приложение Д
(рекомендуемое)
Атрибуты сущностей базы данных

Таблица Д.1 – Атрибуты сущности «User»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
User_ID	Уникальный идентификатор пользователя	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Login	Логин пользователя	varchar	255	a-Я a-Z 0-9	нет		ivannov73
PasswordHash	Пароль пользователя	varchar	255		нет		dn883bsdmvhs
Surname	Фамилия пользователя	varchar	255	a-Я	нет		Иванов
Name	Имя пользователя	varchar	255	a-Я	нет		Иван
Patronymic	Отчество пользователя	varchar	255	a-Я	да		Иванович
Role	Одна из 3 возможных ролей	int2	2	0, 1, 2	нет		2
Group	Учебная группа	varchar	12	a-Я 0-9	нет		21ВП1
Phone	Номер телефона	int8	8	0-1	нет		89272895686
Contacts	Контакты пользователя	varchar	255		нет		https://vk.com/id123456789
Archive	Статус пользователя (активный или нет)	bool	1	True, False	нет		True

Таблица Д.2 – Атрибуты сущности «Event»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Event_ID	Уникальный идентификатор мероприятия	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Title	Название мероприятия	varchar	255	a-Я a-Z 0-9	нет		Студенческая весна 2025
Status	Статус мероприятия	int2	2	0, 1, 2, 3, 4, 5	нет		0

Продолжение таблицы Д.2

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Description	Описание мероприятия	varchar	255	а-Я а-Z 0-9	нет		Главное культурно-массовое соревнование между факультетами
StartDate	Дата начала организации мероприятия	date	4	0001-01-01-9999-12-31	да		2025-04-05
EndDate	Дата окончания и проведения мероприятия	date	4	0001-01-01-9999-12-31	да		2025-04-05
EventTime	Время проведения мероприятия	time	8	00:00:00-24:00:00	да		13:15:00
Location	Место проведения мероприятия	varchar	255	а-Я а-Z 0-9	да		ККЗ 5 корпуса
NumberOfParticipants	Количество участников	int2	2	0-9	да		154

Таблица Д.3 – Атрибуты сущности «Task»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Task_ID	Уникальный идентификатор задачи	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Event_ID	Идентификатор мероприятия	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	5faaff21-b936-926n-l3m4-cbdf4744

Продолжение таблицы Д.3

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Partner_ID	Идентификатор партнера задачи	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	да	FK	5faaff21-b936-926n-l3m4-cbdf4744
Title	Название задачи	varchar	255	a-Я a-Z 0-9	нет		Заказать сувенирную продукцию
Status	Статус задачи	int2	2	0, 1, 2, 3, 4	нет		
StartDate	Дата начала исполнения задачи	date	4	0001-01-01-9999-12-31	нет		2025-04-05
EndDate	Дата завершения исполнения задачи	date	4	0001-01-01-9999-12-31	нет		2025-04-05

Таблица Д.4 – Атрибуты сущности «Partner»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Partner_ID	Уникальный идентификатор партнера	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Surname	Фамилия пользователя	varchar	255	a-Я	нет		Иванов
Name	Имя пользователя	varchar	255	a-Я	нет		Иван
Patronymic	Отчество пользователя	varchar	255	a-Я	да		Иванович
Description	Описание партера	varchar	255	a-Я a-Z 0-9	нет		Световик звукач
Phone	Номер телефона	int8	8	0-1	нет		89272895686
Contacts	Контакты пользователя	varchar	255		нет		https://vk.com/id123456789
Archive	Статус партнера (активный или нет)	bool	1	True, False	нет		True

Таблица Д.5 – Атрибуты сущности «Note»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Note_ID	Уникальный идентификатор заметки	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
User_ID	Идентификатор автора заметки	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	5faaff21-b936-926n-l3m4-cbdf4744
Event_ID	Идентификатор мероприятия	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	5faaff21-b936-926n-l3m4-cbdf4744
Tilte	Название заметки	varchar	255	а-Я а-Z 0-9	нет		Список гостей
FilePath	Путь к markdown файлу	varchar	255		нет		files/notes/task_001.m d

Таблица Д.6 – Атрибуты сущности «Meeting»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Meeting_ID	Уникальный идентификатор встречи	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Title	Название встречи	varchar	255	а-Я а-Z 0-9	нет		
MeetingDate	Дата встречи	date	4	0001-01-01-9999-12-31	нет		2025-04-05
MeetingTime	Время встречи	time	8	00:00:00-24:00:00	нет		13:15:00
Location	Место встречи	varchar	255		нет		Аудитория 7а-506
Link	Ссылка на встречу	varchar	255		да		https://meet.google.com/yzatopwuuw

Таблица Д.7 – Атрибуты сущности «Event_User»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
User_ID	Уникальный идентификатор пользователя	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Event_ID	Уникальный идентификатор мероприятия	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Role	Роль человека в мероприятии	int2	2	0, 1, 2	нет		0

Таблица Д.8 – Атрибуты сущности «Task_User»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
User_ID	Уникальный идентификатор пользователя	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Task_ID	Уникальный идентификатор задачи	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Role	Роль человека в задаче	int2	2	0, 1	нет		0

Таблица Д.9 – Атрибуты сущности «Meeting_User»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
User_ID	Уникальный идентификатор пользователя	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744

Продолжение таблицы Д.9

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Meeting_ID	Уникальный идентификатор встречи	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Role	Роль человека на встрече	int2	2	0, 1	нет		0

Таблица Д.10 – Атрибуты сущности «Event_Partner»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Partner_ID	Уникальный идентификатор партнера	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744
Event_ID	Уникальный идентификатор мероприятия	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	5faaff21-b936-926n-l3m4-cbdf4744

Приложение Е
(рекомендуемое)
Результаты функционального тестирования

Вход в систему
Система планирования деятельности ССС факультета

Неверный логин или пароль

Логин: user

Пароль:

Нет аккаунта в системе? [Зарегистрируйтесь](#)

Войти

Рисунок Е.1 – Авторизация пользователя с неверным паролем

Регистрация в системе
Система планирования деятельности ССС факультета

Логин уже занят другим пользователем

Логин: user1

Имя: Дмитрий

Фамилия: Козлов

Отчество: Игоревич

Учебная группа: 21ВП2

Номер телефона: +7 (999) 123-45-70

VK: @kozlov

Пароль:

Пароль ещё раз:

Уже есть аккаунт? [Войдите в систему](#)

Зарегистрироваться

Рисунок Е.2 – Регистрация с логином, который уже занят

Система для планирования деятельности ССС факультета
Личный кабинет активиста

Козлов Дмитрий Игоревич
@kozlov_di

Активен | Редактировать | Выйти

Учебная группа: 21ВП2 | Контакты: kozlov@email.com | Телефон: +7 (999) 123-45-70 | VK: @kozlov

Мероприятия активиста

Мои мероприятия: 3 | Мой архив: 1

Судебная конференция	Подготовка к студесне	Мероприятие в театре
Участник	Участник	Участник
Д. Гл. орг.: Южи А.В.	Д. Гл. орг.: Петров А.А.	Д. Гл. орг.: Петров А.А.
Подготовка: 01.01.2025	Подготовка: 26.02.2025	Подготовка: 30.03.2025
Проведение: 20.01.2025	Проведение: 15.03.2025	Проведение: 05.05.2025

Рисунок Е.3 – Отображение личного кабинета пользователя «активист»

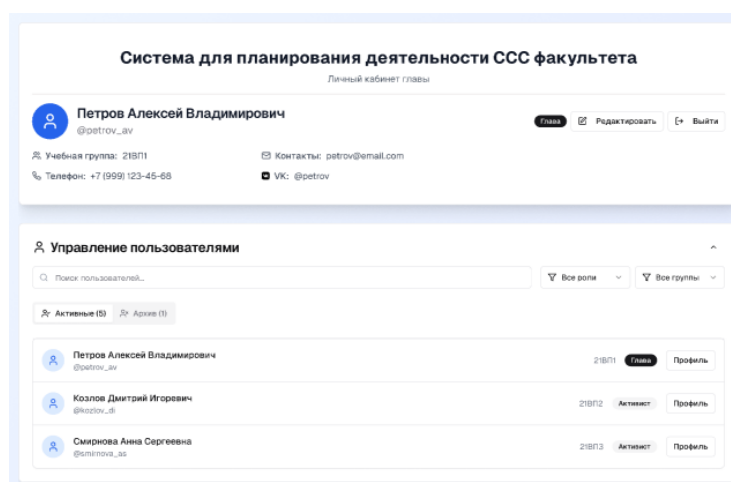


Рисунок Е.4 – Отображение личного кабинета пользователя «глава»

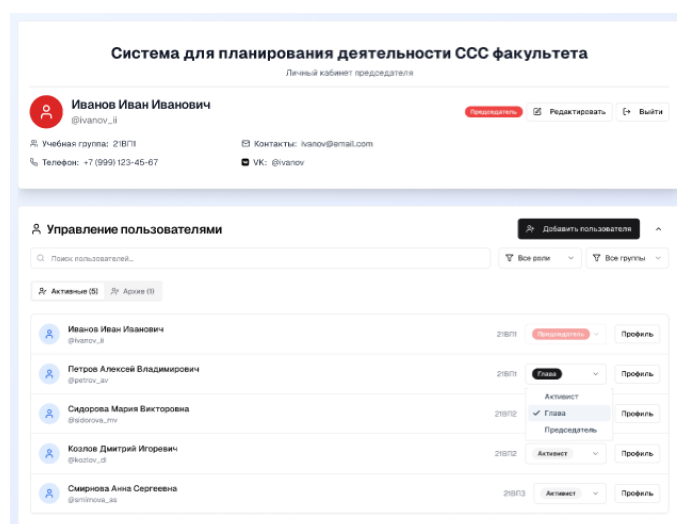


Рисунок Е.5 – Отображение личного кабинета пользователя «председатель»

Редактировать профиль

Данные успешно изменены

Фамилия: Иванов

Имя: Дмитрий

Отчество: Игоревич

Учебная группа: 218П2

Номер телефона: +7 (999) 123-45-70

Email: kozlov@email.com

VK: @kozlov

Отмена Сохранить изменения

Рисунок Е.6 – Изменение данных пользователя

Создать мероприятие

Мероприятие добавлено

Название мероприятия
День знаний

Описание
Торжественный концерт в честь начала учебного года

Дата начала подготовки
01.08.2025

Дата проведения
01.09.2025

Время проведения
10:00

Место проведения
Стадион "Темп"

Отмена Создать мероприятие

Рисунок Е.7 – Добавление нового мероприятия

Детали мероприятия

Студенческая конференция

Описание
Научная конференция студентов факультета

Временные рамки
Начало подготовки: 01.01.2025
Дата проведения: 20.01.2025
Время: 14:00

Место и участники
Место: Конференц-зал
Участников: 80
Главный организатор: Козлов Д.И.

Организаторы мероприятия

ФИО	Роль	Группа	Дата добавления
Петров Александр Александрович	Главный организатор	218П1	05.01.2025
Иванов Иван Петрович	Организатор	218П2	06.01.2025

Рисунок Е.8 – Просмотр подробной информации о мероприятии

Редактировать мероприятие

Данные успешно изменены

Название мероприятия
Студенческая конференция

В процессе

Описание
Научная конференция студентов факультета

Дата начала подготовки
01.01.2025

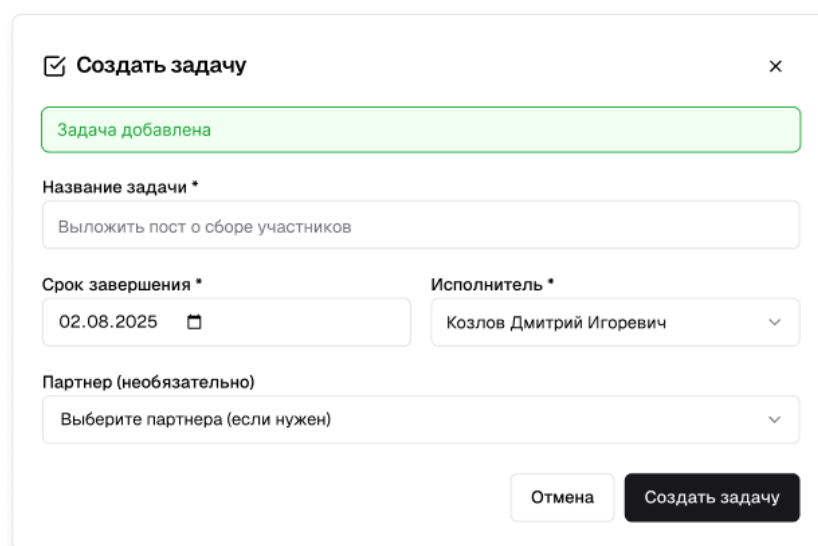
Дата проведения
30.01.2025

Время проведения
15:00

Место проведения
Конференц-зал

Отмена Сохранить изменения

Рисунок Е.9 – Изменение данных мероприятия



Создать задачу

Задача добавлена

Название задачи *

Выложить пост о сборе участников

Срок завершения *

02.08.2025

Исполнитель *

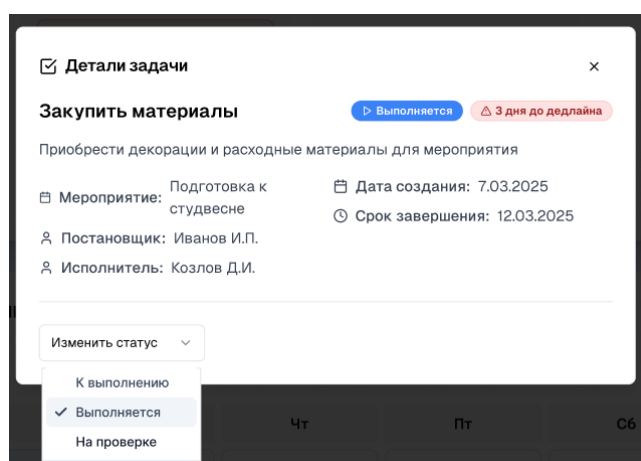
Козлов Дмитрий Игоревич

Партнер (необязательно)

Выберите партнера (если нужен)

Отмена Создать задачу

Рисунок Е.10 – Добавление новой задачи



Детали задачи

Закупить материалы

Выполняется 3 дня до дедлайна

Приобрести декорации и расходные материалы для мероприятия

Мероприятие: Подготовка к студвесне

Постановщик: Иванов И.П.

Исполнитель: Козлов Д.И.

Дата создания: 7.03.2025

Срок завершения: 12.03.2025

Изменить статус

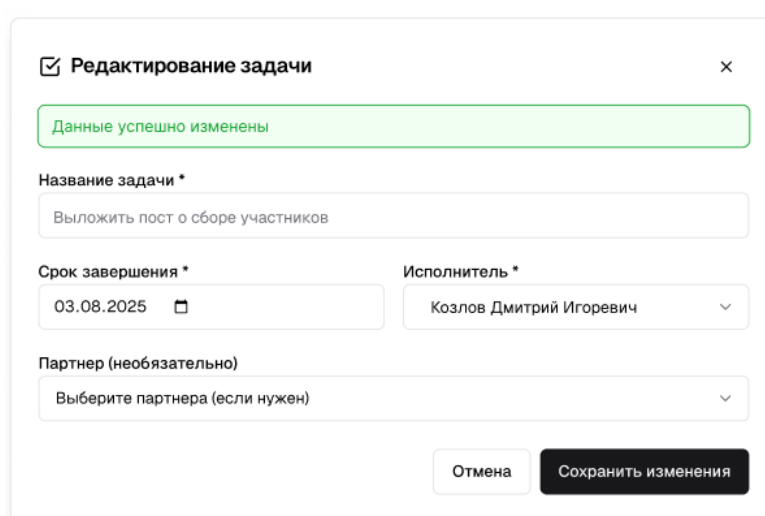
К выполнению

✓ Выполняется

На проверке

Чт Пт Сб

Рисунок Е.11 – Просмотр подробной информации о задаче



Редактирование задачи

Данные успешно изменены

Название задачи *

Выложить пост о сборе участников

Срок завершения *

03.08.2025

Исполнитель *

Козлов Дмитрий Игоревич

Партнер (необязательно)

Выберите партнера (если нужен)

Отмена Сохранить изменения

Рисунок Е.12 – Изменение данных задачи

Добавить партнера ✕

Партнер добавлен

Фамилия: Кокорин Имя: Михаил Отчество: Михайлович

Описание *: Постановщик инструментальных номеров, sound-продюсер

Телефон *: +7 (987) 654-32-11 Контакты: kokorinmm@main.ru

Отмена Создать партнера

Рисунок Е.13 – Добавление нового партнера

Кузнецова Елена Дмитриевна ✎

Директор культурного центра, организатор мероприятий

☎ +7 (999) 222-33-44

✉ kuznetsova@culture.ru

Рисунок Е.14 – Просмотр подробной информации о партнере

Добавить партнера ✕

Данные успешно изменены

Фамилия: Смирнов Имя: Алексей Отчество: Николаевич

Описание *: Представитель компании в области IT-технологий

Телефон *: +7 (999) 111-22-33 Контакты: smirnov@tech.com

Отмена Сохранить изменения

Рисунок Е.15 – Изменение данных партнера

Создать заметку

Заметка добавлена

Название заметки

Список необходимого оборудования

Содержание

Редактирование Предпросмотр

Оборудование:

1. микрофон 2 шт.
2. колонки 2 шт.
3. пульт
4. провода

Отмена Создать заметку

Рисунок Е.16 – Добавление новой заметки

План подготовки

Редактировать

А. Петров А.А.

План подготовки мероприятия

Этапы:

1. Подготовка сценария
2. Бронирование зала
3. Приглашение гостей

Рисунок Е.17 – Просмотр подробной информации о заметке

Редактирование заметки

Данные успешно изменены

Название заметки

План подготовки

Содержание

Редактирование Предпросмотр

План подготовки мероприятия

Этапы:

1. Подготовка сценария
2. Бронирование зала
3. Приглашение гостей
4. Поиск наградной группы

Отмена Сохранить заметку

Рисунок Е.18 – Изменение данных заметки

Запланировать встречу ×

Встреча добавлена

Название встречи *

Встреча по 1 сентября

Дата встречи * Время встречи *

01.08.2025 📅 12:00 ⌚

Место встречи *

Аудитория 7а-506

Ссылка на онлайн-встречу

Необязательно, для онлайн-встреч

Участники *

🔍 Найти участников...

Отмена Создать встречу

Рисунок Е.19 – Добавление новой встречи

Детали встречи ✎ ×

Встреча по церемонии награждения

📍 Место встречи: Аудитория 7а-506 📅 Дата встречи: 01.05.2025

👤 Организатор встречи: Иванов И.И. ⌚ Время встречи 12:00

ФИО	Группа
Козлов Дмитрий Игоревич	21ВП2
Смирнова Анна Сергеевна	21ВП3

Добавить участников

Рисунок Е.20 – Просмотр подробной информации о встрече

Редактировать встречу ×

Данные успешно обновлены

Название встречи *

Планерка по мероприятиям

Дата встречи * Время встречи *

14.08.2025 📅 --:-- ⌚

Место встречи *

Аудитория 7а-506

Ссылка на онлайн-встречу

Необязательно, для онлайн-встреч

Участники *

🔍 Найти участников...

Отмена Сохранить изменения

Рисунок Е.21 – Изменение данных встречи

Приложение Ж
(рекомендуемое)
Листинг модульных тестов

Листинг Ж.1 – Файл UserServiceTests.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Xunit;
using Microsoft.EntityFrameworkCore;
using StudetCouncilPlannerAPI.Models.Entities;
using StudetCouncilPlannerAPI.Models.DTOs;
using StudetCouncilPlannerAPI.Services;
using StudetCouncilPlannerAPI.Data;

namespace StudetCouncilPlannerAPI.Tests
{
    public class UserServiceTests
    {
        private ApplicationDbContext GetDbContext(string dbName)
        {
            var options = new
DbContextOptionsBuilder<ApplicationDbContext>()
                .UseInMemoryDatabase(databaseName: dbName)
                .Options;
            return new ApplicationDbContext(options);
        }

        private User CreateAdmin(string login = "admin", bool
archive = false)
        {
            return new User
            {
                UserId = Guid.NewGuid(),
                Login = login,
                PasswordHash = "hash",
                Surname = "Admin",
                Name = "User",
                Role = 2,
                Group = "A",
                Phone = 1234567890,
                Contacts = "admin@site",
                Archive = archive
            };
        }

        private User CreateStudent(string login = "student", bool
archive = false)
        {
            return new User
            {
                UserId = Guid.NewGuid(),
                Login = login,
                PasswordHash = "hash",
                Surname = "Student",

```

```

        Name = "User",
        Role = 1,
        Group = "B",
        Phone = 9876543210,
        Contacts = "student@site",
        Archive = archive
    };
}

[Fact]
public async System.Threading.Tasks.Task
GetUsersAsync_WithFilters_ReturnsFilteredUsers()
{
    var db =
GetDbContext(nameof(GetUsersAsync_WithFilters_ReturnsFilteredUsers
));

    db.Users.Add(CreateAdmin());
    db.Users.Add(CreateStudent());
    db.SaveChanges();

    var service = new UserService(db);
    var query = new UserListQueryDto { Role = 2 };

    var result = await service.GetUsersAsync(query);

    Assert.Single(result);
    Assert.Equal(2, result[0].Role);
}

[Fact]
public async System.Threading.Tasks.Task
GetUserByIdAsync_ReturnsUserDto_IfExists()
{
    var db =
GetDbContext(nameof(GetUserByIdAsync_ReturnsUserDto_IfExists));
    var user = CreateAdmin();
    db.Users.Add(user);
    db.SaveChanges();

    var service = new UserService(db);
    var result = await
service.GetUserByIdAsync(user.UserId);

    Assert.NotNull(result);
    Assert.Equal(user.UserId, result!.UserId);
}

[Fact]
public async System.Threading.Tasks.Task
UpdateUserAsync_AdminCanEditOtherUser()
{
    var db =
GetDbContext(nameof(UpdateUserAsync_AdminCanEditOtherUser));

```

```

        var admin = CreateAdmin();
        var student = CreateStudent();
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);
        var updateDto = new UserUpdateDto { Name = "Changed"
};

        var updated = await
service.UpdateUserAsync(student.UserId, updateDto, admin);

        Assert.True(updated);
        Assert.Equal("Changed",
db.Users.Find(student.UserId)!.Name);
    }

    [Fact]
    public async System.Threading.Tasks.Task
UpdateUserAsync_UserCannotEditOtherUser()
    {
        var db =
GetDbContext(nameof(UpdateUserAsync_UserCannotEditOtherUser));
        var user1 = CreateStudent("user1");
        var user2 = CreateStudent("user2");
        db.Users.Add(user1);
        db.Users.Add(user2);
        db.SaveChanges();

        var service = new UserService(db);
        var updateDto = new UserUpdateDto { Name = "Changed"
};

        var updated = await
service.UpdateUserAsync(user2.UserId, updateDto, user1);

        Assert.False(updated);
    }

    [Fact]
    public async System.Threading.Tasks.Task
ArchiveUserAsync_AdminCanArchiveUser()
    {
        var db =
GetDbContext(nameof(ArchiveUserAsync_AdminCanArchiveUser));
        var admin = CreateAdmin();
        var student = CreateStudent();
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);

```

```

        var result = await
service.ArchiveUserAsync(student.UserId, admin);

        Assert.True(result);
        Assert.True(db.Users.Find(student.UserId)!.Archive);
    }

    [Fact]
    public async System.Threading.Tasks.Task
ArchiveUserAsync_AdminCannotArchiveLastActiveAdmin()
    {
        var db =
GetDbContext(nameof(ArchiveUserAsync_AdminCannotArchiveLastActiveA
dmin));

        var admin = CreateAdmin();
        db.Users.Add(admin);
        db.SaveChanges();

        var service = new UserService(db);
        var result = await
service.ArchiveUserAsync(admin.UserId, admin);

        Assert.False(result);
        Assert.False(db.Users.Find(admin.UserId)!.Archive);
    }

    [Fact]
    public async System.Threading.Tasks.Task
ArchiveUserAsync_UserCannotArchive()
    {
        var db =
GetDbContext(nameof(ArchiveUserAsync_UserCannotArchive));
        var admin = CreateAdmin();
        var student = CreateStudent();
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);

        var result = await
service.ArchiveUserAsync(admin.UserId, student);

        Assert.False(result);
    }

    [Fact]
    public async System.Threading.Tasks.Task
RestoreUserAsync_AdminCanRestoreArchivedUser()
    {
        var db =
GetDbContext(nameof(RestoreUserAsync_AdminCanRestoreArchivedUser))
;

```

```

        var admin = CreateAdmin();
        var student = CreateStudent(archive: true);
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);

        var result = await
service.RestoreUserAsync(student.UserId, admin);

        Assert.True(result);
        Assert.False(db.Users.Find(student.UserId)!.Archive);
    }

    [Fact]
    public async System.Threading.Tasks.Task
RestoreUserAsync_UserCannotRestore()
    {
        var db =
GetDbContext(nameof(RestoreUserAsync_UserCannotRestore));
        var admin = CreateAdmin(archive: true);
        var student = CreateStudent();
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);

        var result = await
service.RestoreUserAsync(admin.UserId, student);

        Assert.False(result);
    }

    [Fact]
    public async System.Threading.Tasks.Task
ChangeUserRoleAsync_AdminCanChangeRole()
    {
        var db =
GetDbContext(nameof(ChangeUserRoleAsync_AdminCanChangeRole));
        var admin = CreateAdmin();
        var student = CreateStudent();
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);

        var result = await
service.ChangeUserRoleAsync(student.UserId, 2, admin);

        Assert.True(result);
    }

```

```

        Assert.Equal(2, db.Users.Find(student.UserId)!.Role);
    }

    [Fact]
    public async System.Threading.Tasks.Task
    ChangeUserRoleAsync_UserCannotChangeRole()
    {
        var db =
        GetDbContext(nameof(ChangeUserRoleAsync_UserCannotChangeRole));
        var admin = CreateAdmin();
        var student = CreateStudent();
        db.Users.Add(admin);
        db.Users.Add(student);
        db.SaveChanges();

        var service = new UserService(db);

        var result = await
        service.ChangeUserRoleAsync(admin.UserId, 1, student);

        Assert.False(result);
        Assert.Equal(2, db.Users.Find(admin.UserId)!.Role);
    }
}

```

Листинг Ж.2 – Файл AuthServiceTests.cs

```

using System;
using System.Threading.Tasks;
using Xunit;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using StudetCouncilPlannerAPI.Data;
using StudetCouncilPlannerAPI.Models.Entities;
using StudetCouncilPlannerAPI.Models.DTOs;
using StudetCouncilPlannerAPI.Services;

namespace StudetCouncilPlannerAPI.Tests
{
    public class AuthServiceTests
    {
        private ApplicationDbContext GetDbContext(string dbName)
        {
            var options = new
            DbContextOptionsBuilder<ApplicationDbContext>()
                .UseInMemoryDatabase(databaseName: dbName)
                .Options;
            return new ApplicationDbContext(options);
        }

        private IConfiguration GetConfig()
        {
            var settings = new Dictionary<string, string>

```

```

        {
            {"Jwt:Key",
"test_key_12345678901234567890123456789012"},
            {"Jwt:Issuer", "test_issuer"},
            {"Jwt:Audience", "test_audience"}
        };
        return new
ConfigurationBuilder().AddInMemoryCollection(settings).Build();
    }

    [Fact]
    public async System.Threading.Tasks.Task
RegisterAsync_RegistersNewUser()
    {
        var db =
GetDbContext(nameof(RegisterAsync_RegistersNewUser));
        var service = new AuthService(db, GetConfig());

        var dto = new UserRegisterDto
        {
            Login = "user1",
            Password = "Password123",
            Surname = "Ivanov",
            Name = "Ivan",
            Patronymic = "Ivanovich",
            Group = "A1",
            Phone = 1234567890,
            Contacts = "tg"
        };

        var result = await service.RegisterAsync(dto);

        Assert.NotNull(result);
        Assert.Equal("user1", result.Login);
    }

    [Fact]
    public async System.Threading.Tasks.Task
RegisterAsync_DoesNotRegisterDuplicateLogin()
    {
        var db =
GetDbContext(nameof(RegisterAsync_DoesNotRegisterDuplicateLogin));
        var user = new User
        {
            UserId = Guid.NewGuid(),
            Login = "user1",
            PasswordHash = "hash",
            Surname = "Ivanov",
            Name = "Ivan",
            Patronymic = "Ivanovich",
            Group = "A1",
            Phone = 1234567890,
            Contacts = "tg",

```

```

        Role = 0
    };
    db.Users.Add(user);
    db.SaveChanges();

    var service = new AuthService(db, GetConfig());

    var dto = new UserRegisterDto
    {
        Login = "user1",
        Password = "Password123",
        Surname = "Petrov",
        Name = "Petr",
        Patronymic = "Petrovich",
        Group = "B2",
        Phone = 9876543210,
        Contacts = "tg"
    };

    var result = await service.RegisterAsync(dto);

    Assert.Null(result);
}

[Fact]
public async System.Threading.Tasks.Task
LoginAsync_ReturnsTokenForValidCredentials()
{
    var db =
        GetDbContext(nameof(LoginAsync_ReturnsTokenForValidCredentials));
    var service = new AuthService(db, GetConfig());

    var password = "Password123";
    var user = new User
    {
        UserId = Guid.NewGuid(),
        Login = "user1",
        PasswordHash =
            BCrypt.Net.BCrypt.HashPassword(password),
        Surname = "Ivanov",
        Name = "Ivan",
        Patronymic = "Ivanovich",
        Group = "A1",
        Phone = 1234567890,
        Contacts = "tg",
        Role = 0
    };
    db.Users.Add(user);
    db.SaveChanges();

    var dto = new UserLoginDto
    {
        Login = "user1",

```

```

        Password = password
    };

    var result = await service.LoginAsync(dto);

    Assert.NotNull(result);
    Assert.False(string.IsNullOrEmpty(result.Token));
    Assert.NotNull(result.User);
    Assert.Equal("user1", result.User.Login);
}

[Fact]
public async System.Threading.Tasks.Task
LoginAsync_ReturnsNullForInvalidPassword()
{
    var db =
        GetDbContext(nameof(LoginAsync_ReturnsNullForInvalidPassword));
    var service = new AuthService(db, GetConfig());

    var user = new User
    {
        UserId = Guid.NewGuid(),
        Login = "user1",
        PasswordHash =
            BCrypt.Net.BCrypt.HashPassword("Password123"),
        Surname = "Ivanov",
        Name = "Ivan",
        Patronymic = "Ivanovich",
        Group = "A1",
        Phone = 1234567890,
        Contacts = "tg",
        Role = 0
    };
    db.Users.Add(user);
    db.SaveChanges();

    var dto = new UserLoginDto
    {
        Login = "user1",
        Password = "WrongPassword"
    };

    var result = await service.LoginAsync(dto);

    Assert.Null(result);
}
}
}

```

Листинг Ж.3 – Файл EventServiceTests.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Threading.Tasks;
using Xunit;
using Microsoft.EntityFrameworkCore;
using StudetCouncilPlannerAPI.Models.Entities;
using StudetCouncilPlannerAPI.Models.Dtos;
using StudetCouncilPlannerAPI.Services;
using StudetCouncilPlannerAPI.Data;

namespace StudetCouncilPlannerAPI.Tests
{
    public class EventServiceTests
    {
        private ApplicationDbContext GetDbContext(string dbName)
        {
            var options = new
DbContextOptionsBuilder<ApplicationDbContext>()
                .UseInMemoryDatabase(databaseName: dbName)
                .Options;
            return new ApplicationDbContext(options);
        }

        private Event CreateEvent()
        {
            return new Event
            {
                EventId = Guid.NewGuid(),
                Title = "Event",
                Status = 0,
                Description = "Desc",
                StartDate =
DateOnly.FromDateTime(DateTime.UtcNow),
                EndDate =
DateOnly.FromDateTime(DateTime.UtcNow.AddDays(1)),
                EventTime = TimeSpan.FromHours(10),
                Location = "Room 1",
                NumberOfParticipants = 0
            };
        }

        [Fact]
        public async System.Threading.Tasks.Task
GetEventsAsync_ReturnsEvents()
        {
            var db =
GetDbContext(nameof(GetEventsAsync_ReturnsEvents));
            db.Events.Add(CreateEvent());
            db.Events.Add(CreateEvent());
            db.SaveChanges();

            var service = new EventService(db);

            var result = await service.GetEventsAsync(new
EventListQueryDto());

```

```

        Assert.Equal(2, result.Count);
    }

    [Fact]
    public async System.Threading.Tasks.Task
    GetEventByIdAsync_ReturnsEvent()
    {
        var db =
        GetDbContext(nameof(GetEventByIdAsync_ReturnsEvent));
        var ev = CreateEvent();
        db.Events.Add(ev);
        db.SaveChanges();

        var service = new EventService(db);

        var result = await
        service.GetEventByIdAsync(ev.EventId);

        Assert.NotNull(result);
        Assert.Equal(ev.EventId, result.EventId);
    }

    [Fact]
    public async System.Threading.Tasks.Task
    GetEventByIdAsync_ReturnsNullIfNotFound()
    {
        var db =
        GetDbContext(nameof(GetEventByIdAsync_ReturnsNullIfNotFound));
        var service = new EventService(db);

        var result = await
        service.GetEventByIdAsync(Guid.NewGuid());

        Assert.Null(result);
    }

    [Fact]
    public async System.Threading.Tasks.Task
    CreateEventAsync_CreatesEvent()
    {
        var db =
        GetDbContext(nameof(CreateEventAsync_CreatesEvent));
        var service = new EventService(db);

        var dto = new EventCreatedDto
        {
            Title = "New Event",
            Description = "Description",
            StartDate = DateOnly.FromDateTime(DateTime.Today),
            EndDate =
            DateOnly.FromDateTime(DateTime.Today.AddDays(1)),
            EventTime = TimeSpan.FromHours(12),

```

```

        Location = "Main Hall",
        NumberOfParticipants = 20
    };

    var creatorId = Guid.NewGuid();

    var eventId = await service.CreateEventAsync(dto,
creatorId);

    var ev = db.Events.Include(e =>
e.EventUsers).FirstOrDefault(e => e.EventId == eventId);
    Assert.NotNull(ev);
    Assert.Equal("New Event", ev.Title);
    Assert.Single(ev.EventUsers.Where(eu => eu.UserId ==
creatorId));
    }

    [Fact]
    public async System.Threading.Tasks.Task
UpdateEventAsync_UpdatesEvent()
    {
        var db =
GetDbContext(nameof(UpdateEventAsync_UpdatesEvent));
        var ev = CreateEvent();
        db.Events.Add(ev);
        db.SaveChanges();

        var service = new EventService(db);

        var updateDto = new EventUpdateDto
        {
            Title = "Updated",
            Description = "Updated Desc",
            Status = 1,
            StartDate = ev.StartDate,
            EndDate = ev.EndDate,
            EventTime = ev.EventTime,
            Location = ev.Location,
            NumberOfParticipants = 42
        };

        var result = await
service.UpdateEventAsync(ev.EventId, updateDto);

        Assert.True(result);
        var updated = db.Events.Find(ev.EventId);
        Assert.Equal("Updated", updated.Title);
        Assert.Equal(1, updated.Status);
    }

    [Fact]
    public async System.Threading.Tasks.Task
AddParticipantAsync_AddsParticipant()

```

```

    {
        var db =
        GetDbContext(nameof(AddParticipantAsync_AddsParticipant));
        var ev = CreateEvent();
        db.Events.Add(ev);
        db.SaveChanges();

        var service = new EventService(db);

        var userId = Guid.NewGuid();

        var result = await
        service.AddParticipantAsync(ev.EventId, userId);

        Assert.True(result);
        Assert.Contains(db.EventUsers, eu => eu.EventId ==
        ev.EventId && eu.UserId == userId && eu.Role == 0);
    }

    [Fact]
    public async System.Threading.Tasks.Task
    AddParticipantAsync_DoesNotAddDuplicate()
    {
        var db =
        GetDbContext(nameof(AddParticipantAsync_DoesNotAddDuplicate));
        var ev = CreateEvent();
        var userId = Guid.NewGuid();
        db.Events.Add(ev);
        db.EventUsers.Add(new EventUser { EventId =
        ev.EventId, UserId = userId, Role = 0 });
        db.SaveChanges();

        var service = new EventService(db);

        var result = await
        service.AddParticipantAsync(ev.EventId, userId);

        Assert.False(result);
    }

    [Fact]
    public async System.Threading.Tasks.Task
    RemoveParticipantAsync_RemovesParticipant()
    {
        var db =
        GetDbContext(nameof(RemoveParticipantAsync_RemovesParticipant));
        var ev = CreateEvent();
        var userId = Guid.NewGuid();
        db.Events.Add(ev);
        db.EventUsers.Add(new EventUser { EventId =
        ev.EventId, UserId = userId, Role = 0 });
        db.SaveChanges();
    }

```

```

        var service = new EventService(db);

        var result = await
service.RemoveParticipantAsync(ev.EventId, userId);

        Assert.True(result);
        Assert.DoesNotContain(db.EventUsers, eu => eu.EventId
== ev.EventId && eu.UserId == userId && eu.Role == 0);
    }

    [Fact]
    public async System.Threading.Tasks.Task
RemoveParticipantAsync_NoSuchParticipant()
    {
        var db =
GetDbContext(nameof(RemoveParticipantAsync_NoSuchParticipant));
        var ev = CreateEvent();
        db.Events.Add(ev);
        db.SaveChanges();

        var service = new EventService(db);

        var result = await
service.RemoveParticipantAsync(ev.EventId, Guid.NewGuid());

        Assert.False(result);
    }

    [Fact]
    public async System.Threading.Tasks.Task
AddOrganizerAsync_AddsOrganizer()
    {
        var db =
GetDbContext(nameof(AddOrganizerAsync_AddsOrganizer));
        var ev = CreateEvent();
        db.Events.Add(ev);
        db.SaveChanges();

        var service = new EventService(db);

        var userId = Guid.NewGuid();

        var result = await
service.AddOrganizerAsync(ev.EventId, userId);

        Assert.True(result);
        Assert.Contains(db.EventUsers, eu => eu.EventId ==
ev.EventId && eu.UserId == userId && eu.Role == 1);
    }

    [Fact]
    public async System.Threading.Tasks.Task
AddOrganizerAsync_DoesNotAddDuplicate()

```

```

        {
            var db =
                GetDbContext(nameof(AddOrganizerAsync_DoesNotAddDuplicate));
            var ev = CreateEvent();
            var userId = Guid.NewGuid();
            db.Events.Add(ev);
            db.EventUsers.Add(new EventUser { EventId =
                ev.EventId, UserId = userId, Role = 1 });
            db.SaveChanges();

            var service = new EventService(db);

            var result = await
                service.AddOrganizerAsync(ev.EventId, userId);

            Assert.False(result);
        }

        [Fact]
        public async System.Threading.Tasks.Task
            RemoveOrganizerAsync_RemovesOrganizer()
        {
            var db =
                GetDbContext(nameof(RemoveOrganizerAsync_RemovesOrganizer));
            var ev = CreateEvent();
            var userId = Guid.NewGuid();
            db.Events.Add(ev);
            db.EventUsers.Add(new EventUser { EventId =
                ev.EventId, UserId = userId, Role = 1 });
            db.SaveChanges();

            var service = new EventService(db);

            var result = await
                service.RemoveOrganizerAsync(ev.EventId, userId);

            Assert.True(result);
            Assert.DoesNotContain(db.EventUsers, eu => eu.EventId
                == ev.EventId && eu.UserId == userId && eu.Role == 1);
        }

        [Fact]
        public async System.Threading.Tasks.Task
            RemoveOrganizerAsync_NoSuchOrganizer()
        {
            var db =
                GetDbContext(nameof(RemoveOrganizerAsync_NoSuchOrganizer));
            var ev = CreateEvent();
            db.Events.Add(ev);
            db.SaveChanges();

            var service = new EventService(db);

```

```
        var result = await
service.RemoveOrganizerAsync(ev.EventId, Guid.NewGuid());

        Assert.False(result);
    }
}
```