

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

*Факультет*  
вычислительной техники

*Кафедра*  
МО и ПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

**БАКАЛАВРСКАЯ РАБОТА**

на тему

**Приложение «Конструктор группового рабочего  
пространства»**

Студент \_\_\_\_\_ Малков Даниил Дмитриевич  
(подпись, дата) (ФИО полностью)

Руководитель \_\_\_\_\_ Кольчугина Е.А.  
(подпись, дата) (фамилия, инициалы)

Нормоконтролер \_\_\_\_\_ Такташкин Д.В.  
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от \_\_\_\_\_ № \_\_\_\_\_)

Заведующий кафедрой \_\_\_\_\_ Козлов А.Ю.  
(подпись) (фамилия, инициалы)

Работа защищена с отметкой \_\_\_\_\_ (протокол заседания ГЭК от \_\_\_\_\_ № \_\_\_\_\_)

Секретарь ГЭК \_\_\_\_\_ Попова Н.А.  
(подпись) (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  
«\_\_» \_\_\_\_\_ 2024 г

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент \_\_\_\_\_ Малков Даниил Дмитриевич \_\_\_\_\_ гр. 21ВИ1 \_\_\_\_\_ факультета \_\_\_\_\_ ВТ  
<Фамилия, имя, отчество полностью>

2. Тема работы \_\_\_\_\_ Приложение «Конструктор группового рабочего  
пространства» \_\_\_\_\_

Тема утверждена приказом ПГУ № \_\_\_\_\_ от "\_\_\_\_" \_\_\_\_\_ 2024 г.

3. Руководитель работы \_\_\_\_\_ профессор кафедры «МО и ПЭВМ» Кольчугина Е.А.

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

1) Разработать приложение конструктор для группового решения задач с  
возможностью расширения, управления и настройки рабочего пространства.

2) Среда и средства разработки:

Языковые средства разработки: Golang, JavaScript(React), UML.

Инструментальные средства: Visual Studio Code, Docker Desktop, Visual Paradigm,

Технологии разработки: ООП, REST API, микросервисная архитектура,  
контейнеризация с использованием Docker.

3) Среда функционирования: Клиент - Веб-браузер, Сервер – Nginx.

4) Среда виртуализации: Docker Desktop, WSL.

5) Основные функции приложения:

а) создать рабочее пространство;

б) пригласить участников в пространство;

в) управлять задачами;

г) управлять событиями;

д) управлять ссылками на источники;

е) запустить конструктор;

6) Особенности: интеграция с микросервисами, простое масштабирование

7) Требования к структуре данных: эффективность, гибкость, устойчивость

8) Требования к базе данных: надежность, безопасность, технология PostgreSQL

9) Провести функциональное приложения

5. Перечень подлежащих разработке вопросов:

|                                                   |
|---------------------------------------------------|
| 1) Анализ предметной области и постановка задачи; |
| 2) Проектирование;                                |
| 3) Реализация;                                    |
| 4) Контроль качества программного обеспечения;    |
| 5) План разработки бюджета;                       |
| 6) Оформление пояснительной записки.              |

6. Календарный график выполнения работы

| №<br>п/п | Наименование этапов<br>работы                    | Объем<br>работы | Срок выполнения         | Подпись<br>руководителя |
|----------|--------------------------------------------------|-----------------|-------------------------|-------------------------|
| 1        | Анализ предметной области<br>и постановка задачи | 10%             | 13.04.2025 – 27.04.2025 |                         |
| 2        | Проектирование                                   | 25%             | 27.04.2025 – 10.05.2025 |                         |
| 3        | Реализация                                       | 35%             | 10.05.2025 – 30.05.2025 |                         |
| 4        | Контроль качества<br>программного обеспечения    | 10%             | 31.05.2025 – 02.06.2025 |                         |
| 5        | План разработки бюджета                          | 5%              | 02.06.2025 – 03.06.2025 |                         |
| 6        | Оформление<br>пояснительной записки              | 15%             | 03.06.2025 – 14.06.2025 |                         |

Дата выдачи задания «22» \_\_\_\_\_ декабря \_\_\_\_\_ 2024 г.

Руководитель бакалаврской работы \_\_\_\_\_  
(подпись, дата)

Задание к исполнению принял студент \_\_\_\_\_  
(подпись, дата)

Работу к защите допустить  
Декан факультета \_\_\_\_\_  
(подпись, дата)

## Реферат

Пояснительная записка содержит 151 листов, 31 рисунок, 17 таблиц, 21 использованный источник, 3 приложения.

ВЕБ-ПРИЛОЖЕНИЕ, КОНСТРУКТОР РАБОЧЕГО ПРОСТРАНСТВА, ГРУППОВАЯ РАБОТА, GOLANG, REACT, UML, БАЗА ДАННЫХ, POSTGRESQL, DOCKER

Объектом разработки является конструктор группового рабочего пространства.

Цель выпускной квалификационной работы – разработка приложения «конструктор группового рабочего пространства», позволяющая управлять пространством и формировать конструктор для рабочей группы.

Технология разработки – веб-приложение с серверной частью на языке программирования Go, клиентская часть реализована с использованием библиотеки React, данные хранятся в базе данных PostgreSQL, приложение контейнеризированно с помощью Docker, среда разработки – Visual Studio Code.

В результате выпускной квалификационной работы были разработаны клиентская и серверная часть приложения «конструктор группового рабочего пространства».

|           |      |                 |         |      |                                                                                    |              |      |        |
|-----------|------|-----------------|---------|------|------------------------------------------------------------------------------------|--------------|------|--------|
|           |      |                 |         |      | ПГУ 09.03.02 – 08BP191.18 ПЗ                                                       |              |      |        |
| Изм.      | Лист | № докум.        | Подпись | Дата |                                                                                    |              |      |        |
| Разраб.   |      | Малков Д.Д.     |         |      | Приложение «Конструктор группового рабочего пространства»<br>Пояснительная записка | Лит.         | Лист | Листов |
| Провер.   |      | Кольчугина Е.А. |         |      |                                                                                    |              | 4    | 25     |
| Реценз.   |      |                 |         |      |                                                                                    | группа 21ВИ1 |      |        |
| Н. Контр. |      | Такташкин Д.В.  |         |      |                                                                                    |              |      |        |
| Утверд.   |      | Козлов А.Ю.     |         |      |                                                                                    |              |      |        |

## Содержание

|                                                                                 |    |
|---------------------------------------------------------------------------------|----|
| Введение.....                                                                   | 7  |
| 1 Анализ альтернативных решений организации групповой работы.....               | 9  |
| 1.1 Анализ предметной области и постановка задачи.....                          | 9  |
| 1.2 Обзор и анализ программного обеспечения аналогичного назначения             | 10 |
| 1.2.1 Анализ программы Notion .....                                             | 10 |
| 1.2.2 Анализ таск-менеджера Trello для групповой работы .....                   | 12 |
| 1.2.3 Анализ платформы ClickUp для кастомизации рабочих процессов               | 13 |
| 1.2.4 Сравнительный анализ аналогичных систем.....                              | 14 |
| 1.3 Анализ функциональных требований .....                                      | 15 |
| 1.4 Анализ нефункциональных требований .....                                    | 19 |
| 2 Проектирование приложения конструктора группового рабочего пространства ..... | 20 |
| 2.1 Архитектура приложения .....                                                | 20 |
| 2.2 Проектирование структур данных.....                                         | 21 |
| 2.2.1 Концептуальная модель базы данных .....                                   | 21 |
| 2.2.2 Логическая модель базы данных .....                                       | 24 |
| 2.3 Проектирование пользовательского интерфейса.....                            | 25 |
| 2.4 Проектирование структуры данных серверной части приложения...               | 25 |
| 2.5 Проектирование сервиса по принципу инверсии зависимостей .....              | 26 |
| 3 Реализация приложения конструктора группового рабочего пространства           | 28 |
| 3.1 Реализация виртуализации проекта приложения.....                            | 28 |
| 3.2 Реализация микросервисной системы группового рабочего пространства .....    | 29 |
| 3.3 Реализация базы данных.....                                                 | 31 |
| 3.4 Реализация клиентской части приложения .....                                | 33 |

|     |                                                                      |     |
|-----|----------------------------------------------------------------------|-----|
| 4   | Контроль качества программного обеспечения.....                      | 36  |
| 4.1 | Выбор режима тестирования.....                                       | 37  |
| 4.2 | Функциональное тестирование.....                                     | 38  |
| 4.3 | Метрики кода .....                                                   | 40  |
| 5   | Планирование разработки и оценка бюджета.....                        | 42  |
|     | Заключение .....                                                     | 45  |
|     | Список использованных источников .....                               | 47  |
|     | Приложение А (обязательное) Листинг серверной части приложения.....  | 50  |
|     | Приложение Б (обязательное) Листинг клиентской части приложения..... | 78  |
|     | Приложение В (обязательное) Результаты тестирования.....             | 145 |

## Введение

Сегодня все больше задач переходят в режим онлайн работы, позволяя бригадам совместно решать поставленные задачи и контролировать дедлайны. Это важно для стартапов, студенческих работ и прочих команд. Основная категория таких людей – молодое поколение, стремящееся к гибкости и адаптации задач под собственные нужды.

Большинство самых разных «таск-менеджеров» сегодня становятся все более популярными. Жесткая структура систем управления проектами используется в конкретных областях, поэтому важно определить, насколько свободным и расширяемым можно сделать приложение. Для информационных проектов, маркетинга и стартапов возникает потребность в таком приложении, где для создания персональных рабочих пространств требуется своя отдельная итоговая документация.

Ключевая идея дать участникам возможность собрать конструктор, развивая собственное рабочее пространство из предоставленных базовых компонентов, например таблицы задач, календаря событий и списка источников. Повышая отчетность управления у подобного рода проектов, улучшая визуальную структуру и целостность проектов команды.

Целью выпускной квалификационной работы является разработка приложения «конструктора группового рабочего пространства». В котором возможно будет создавать пространства, приглашать участников, добавлять задачи и закреплять общий процесс в едином конструкторе. Приложение реализуется с использованием мощного современного технологического стека: Go (серверная часть), React (клиентская часть), PostgreSQL (база данных), Docker (контейнеризация, виртуализация).

Для достижения поставленной цели были сформулированы задачи такие как:

- изучить и проанализировать существующие решения на рынке цифровых рабочих пространств;

- спроектировать архитектуру приложения и структуру данных;
- реализовать веб-приложение с возможностью настройки модулей и совместной работы;
- провести серверное и клиентское тестирование системы;
- подготовить пояснительную записку в соответствии с заданными стандартами.

Выполнение этих задач определено позволит создать приложение конструктор группового рабочего пространства.

## **1 Анализ альтернативных решений организации групповой работы**

### **1.1 Анализ предметной области и постановка задачи**

Система управления проектом – это такая система, где проект прежде всего представлен отдельной целой структурой, которая помогает участникам проекта ориентироваться в ресурсах и координировать действия. Создаваемое рабочее пространство нацелено на выполнение миссии такой системы, то есть оно и есть адаптивное управление проектом, содержащее в себе целую систему взаимодействия и документирования проекта [1].

Прямая и самая простая аналогия такой системы – это пояснительная записка. Она содержит требования к своей структуре, цель, задание, содержание, пункты реализации, вывод, используемые источники и включаемые приложения. За время учебы в университете было составлено большое множество подобных отчетов к лабораторным работам, у каждой есть общие черты с другими, такие как титульный лист, вывод, литературные источники. Имея различия лишь в конфигурации и требованиях к оформлению, например, работая в бригадах на выполнение таких работ – в титульном листе указывался перечень участников. От этого можно и отталкиваться при проектировании такого приложения группового рабочего пространства.

Так в зависимости от целей, команды и темы проекта, функциональность группового рабочего пространства различается, но только вокруг ключевых элементов: управления конфигурацией проекта, управления пользователями, системой задач, планирования событий, файлового хранилища и списком источников.

В качестве идеологической основы для проектирования разрабатываемого решения была выбрана концепция минимализма и функциональности, отраженная в книге «ReWork. Бизнес без предрассудков»,

где подчеркивается важность простых, гибких и адаптивных инструментов совместной работы [2].

Целевой аудиторией разрабатываемого приложения являются команды разработчиков, проектные группы, студенты и преподаватели, использующие совместную среду и такие системы контроля версий, как «GitHub» для управления учебными или рабочими процессами.

Как итог, приложение должно выполнять следующие функции:

- создавать и приглашать в рабочее пространство;
- управлять инструментами в пространстве;
- добавлять, изменять и удалять элементы инструментов;
- формировать конструктор по ресурсам проекта.

Таким образом, требуется разработать веб-приложение с клиентской и серверной частью, а также базой данных, которое будет предоставлять функциональную, простую и надежную платформу для группового рабочего процесса.

## **1.2 Обзор и анализ программного обеспечения аналогичного назначения**

### **1.2.1 Анализ программы Notion**

Одним из более близких аналогов является приложение для заметок Notion. Это кроссплатформенное решение, поддерживаемое на компьютерах, ноутбуках, мобильных устройствах [3]. На рисунке 1 демонстрируется окно программы Notion.

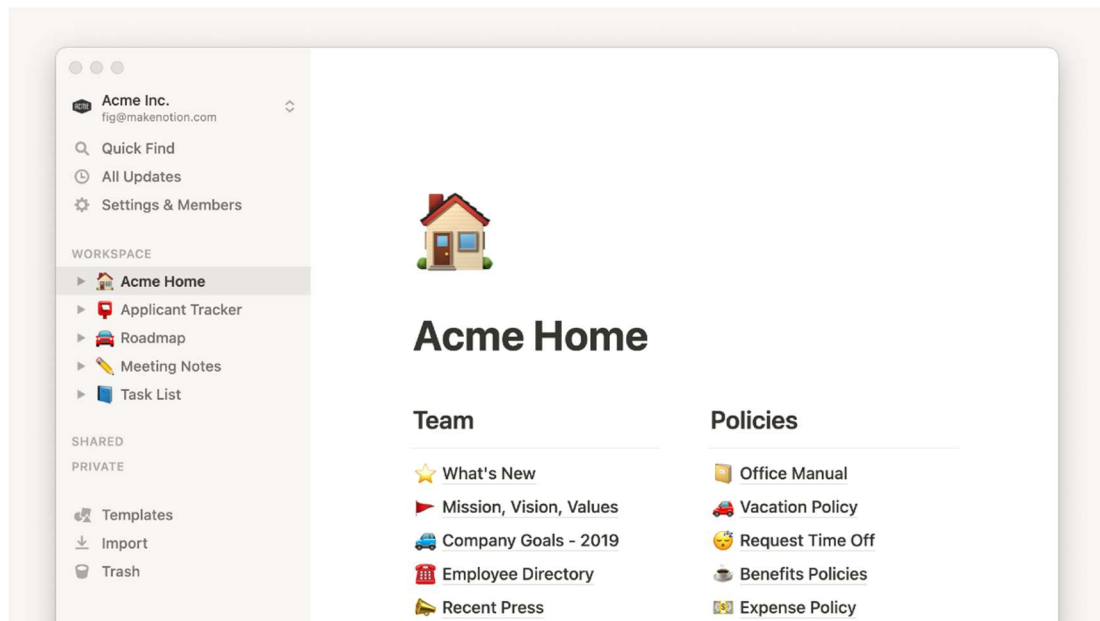


Рисунок 1 – Окно программы Notion

Notion, создававшаяся изначально, как инструмент реализации принципов «Getting Things Done», успешно отражает ряд достоинств такой системы:

- программа имеет гибкую организацию данных разных блоков, таких как текст, таблицы, списки и т. д.;
- возможность совместной работы в реальном времени с настройкой прав доступа для участников;
- интеграция с другими сервисами;
- поддержка шаблонов для быстрого создания документов, досок задач, вики-страниц;
- экспорт данных в разные форматы (PDF, Markdown, HTML, и д.р.).

Недостатком программы является отсутствие полноценной оффлайн работы для доступа к функциям, сложности начального освоения для новичков из-за множественного функционала и закрытый доступ к исходному коду проекта.

### 1.2.2 Анализ task-менеджера Trello для групповой работы

Trello – это популярный сервис для управления задачами, основанный на методологии Kanban. Он предоставляет удобный интерфейс для организации групповой работы, позволяя участникам эффективно распределять задачи и отслеживать их выполнение [4]. Интерфейс Trello представлен на рисунке 3.

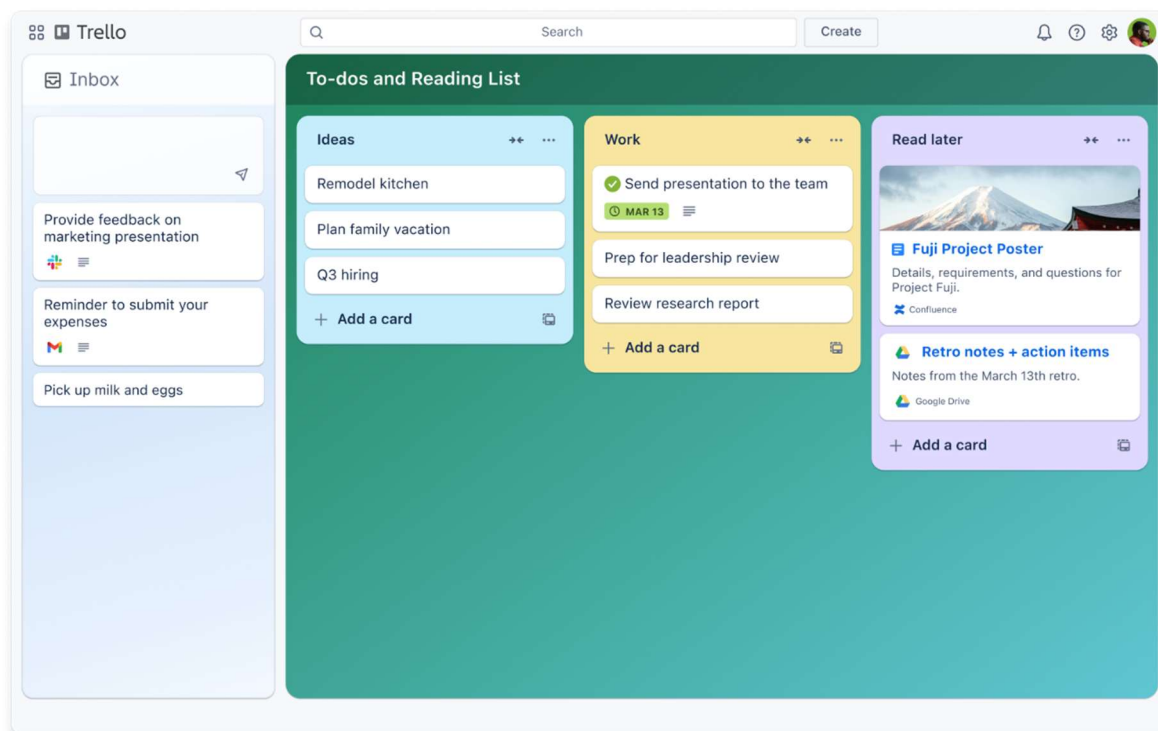


Рисунок 2 – Интерфейс task-менеджера Trello

Главное преимущество Trello заключается в его простоте и наглядности. Система использует доски, списки и карточки, что делает процесс управления проектами интуитивно понятным. Сервис поддерживает совместную работу в реальном времени, уведомления о изменениях, интеграцию с другими инструментами (Google Drive, Slack, GitHub и др.), а также мобильные приложения для работы с любого устройства.

К недостаткам можно отнести ограниченную функциональность в бесплатной версии, включая лимит на количество досок и интеграций. Также при большом количестве задач управление может стать менее удобным из-за

отсутствия сложных инструментов сортировки и фильтрации. Еще одной проблемой опять является зависимость от интернет-соединения, так как офлайн-режим имеет ограниченные возможности.

### **1.2.3 Анализ платформы ClickUp для кастомизации рабочих процессов**

ClickUp – это облачная платформа для управления проектами и задачами, которая сочетает в себе гибкость, многофункциональность и глубокие возможности кастомизации рабочих процессов. Она позволяет адаптировать инструмент под нужды различных команд – от небольших стартапов до крупных корпораций.

Гибкость системы проявляется в многоуровневой структуре организации данных. Рабочие пространства делятся на отделы, проекты, этапы работы и подзадачи. Это позволяет легко адаптировать систему под конкретные бизнес-процессы.

Платформа предлагает множество вариантов визуализации данных: канбан-доски для команд, диаграммы Ганта для долгосрочного планирования, календари для управления сроками, Mind Maps для мозговых штурмов, и т. д.

Глубокая кастомизация задач включает:

- пользовательские поля (чек-листы, метрики);
- настраиваемые статусы выполнения;
- шаблоны для повторяющихся процессов;
- систему зависимостей между задачами.

Автоматизация значительно упрощает работу. Можно настроить:

- автоматическую смену статусов;
- уведомления при изменениях;
- интеграции с популярными сервисами (Slack, GitHub, Zoom).

Аналитические возможности включают дашборды для отслеживания:

- прогресса по проектам;
- нагрузки сотрудников;

– трудозатрат.

Основные преимущества платформы имеют в себе: универсальность для разных сфер, поддержку различных методологий, удобные инструменты коллаборации, наличие бесплатного тарифа, создание отчетов по проектам.

К недостаткам можно отнести: сложность первоначального освоения, ограниченный офлайн-доступ, лимиты бесплатной версии.

#### 1.2.4 Сравнительный анализ аналогичных систем

Для проведения сравнительного анализа программных решений, были выбраны критерии, наиболее подходящие под нужды нашего сервиса [5]. В таблице 1 приведен список критериев оценки приложения и их вес.

Таблица 1 – Таблица критериев для оценки аналогов

| №     | Критерий                                   | Вес  |
|-------|--------------------------------------------|------|
| 1     | Гибкость организации данных                | 0.10 |
| 2     | Возможности управления задачами            | 0.15 |
| 3     | Совместная работа                          | 0.15 |
| 4     | Интеграции с другими сервисами             | 0.04 |
| 5     | Шаблоны и кастомизация                     | 0.10 |
| 6     | Автоматизация                              | 0.09 |
| 7     | Аналитика и отчетность                     | 0.07 |
| 8     | Мобильность и доступность                  | 0.07 |
| 9     | Простота освоения                          | 0.08 |
| 10    | Удобство создания и управления документами | 0.02 |
| 11    | Бесплатная версия                          | 0.05 |
| 12    | Производительность при масштабировании     | 0.08 |
| Сумма |                                            | 1    |

Для критериев была использована следующая формула расчета аддитивной суммы интегральной оценки для каждого приложения:

$$S = \sum_{i=1}^n (W_i * X_i)$$

где  $S$  – интегральная оценка критериев аналогов;

$n$  – количество критериев;

$W_i$  – значение, соответствующее характеристике критерию аналогов (от 1 до 10);

$X_i$  – вес важности рассматриваемого критерия (сумма всех весов равна 1).

Для расчета выбора оптимального программного решения была создана таблица 2, которая отображает результаты интегральной оценки критериев аналогичных решений управления проектами.

Таблица 2 – Оценка критериев аналогичных программ

| №  | Критерий                                   | Notion | Trello | ClickUp |
|----|--------------------------------------------|--------|--------|---------|
| 1  | Гибкость организации данных                | 10     | 6      | 9       |
| 2  | Возможности управления задачами            | 7      | 8      | 10      |
| 3  | Совместная работа                          | 9      | 9      | 9       |
| 4  | Интеграции с другими сервисами             | 8      | 8      | 9       |
| 5  | Шаблоны и кастомизация                     | 9      | 6      | 10      |
| 6  | Автоматизация                              | 6      | 7      | 10      |
| 7  | Аналитика и отчетность                     | 7      | 5      | 9       |
| 8  | Мобильность и доступность                  | 8      | 9      | 8       |
| 9  | Простота освоения                          | 6      | 10     | 7       |
| 10 | Удобство создания и управления документами | 10     | 4      | 8       |
| 11 | Бесплатная версия                          | 8      | 7      | 9       |
| 12 | Производительность при масштабировании     | 7      | 6      | 8       |
|    | Интегральная оценка                        | 7.85   | 7.39   | 9.01    |

На основе результатов таблицы можно сделать вывод, что ClickUp лидер, по интегральной оценке, и больше прочих соответствует искомым критериям и принципам нашего приложения. Однако, он имеет ограниченный тариф и не является полностью доступным для бесплатного использования. Остальные Notion и Trello различны между собой гибкостью и простотой, но при этом имеют приблизительно равные показатели.

### 1.3 Анализ функциональных требований

На основе аналогичных продуктов и поставленных задачах, были определены следующие функциональные требования:

- создание рабочего пространства;
- приглашение участников;
- управление задачами, ссылками и событиями рабочего пространства;
- компоновка результатов работы конструктора.

На рисунке 3 отображена диаграмма вариантов использования, которая демонстрирует требуемое функциональное выполнение REST API запросов [6].



Рисунок 3 – Диаграмма вариантов использования (клиент)

Диаграмма демонстрирует возможности клиента, как веб-приложения, обращаться к серверу.

Для более углубленного понимания функциональных требований приложения, были описаны порядки основных вариантов использования, изображенных на диаграмме вариантов использования.

Спецификация варианта использования «Создать рабочее пространство» представлена на рисунке 4.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Прецедент: «Создать рабочее пространство»                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| ID: 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Краткое описание: создание рабочего пространства                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Главные актеры: «Клиент»                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Основной поток: <ol style="list-style-type: none"> <li>1. Клиент выбирает создание рабочего пространства               <ol style="list-style-type: none"> <li>1.1. Клиент указывает наименование</li> </ol> </li> <li>2. Клиент отправляет запрос на создание пространства               <ol style="list-style-type: none"> <li>2.1. База данных устанавливает клиента, как участника</li> <li>2.2. База данных устанавливает клиента, как админа (создателя)</li> </ol> </li> <li>3. Сервис отвечает в соответствии с результатами валидации</li> </ol> |
| Постусловия: клиентское приложение получает ID созданного пространства и обновляет список доступных рабочих пространств пользователю                                                                                                                                                                                                                                                                                                                                                                                                                     |

Рисунок 4 – Спецификация варианта использования «Создать рабочее пространство»

Спецификация варианта использования «Создать приглашение» представлена на рисунке 5.

|                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Прецедент: «Создать приглашение»                                                                                                                                |
| ID: 2                                                                                                                                                           |
| Краткое описание: создание кода приглашения в рабочее пространство                                                                                              |
| Главные актеры: «Клиент»                                                                                                                                        |
| Основной поток: <ol style="list-style-type: none"> <li>1. Клиент создаёт код-приглашение для участника</li> <li>2. Сервер возвращает код-приглашение</li> </ol> |
| Постусловия: Клиент получает код допуска в рабочее пространство для других пользователей                                                                        |

Рисунок 5 – Спецификация варианта использования «Создать приглашение»

Спецификация варианта использования «Добавить задачу» представлена на рисунке 6.

|                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Прецедент: «Добавить задачу»                                                                                                                                                                                                                                                                                                                    |
| ID: 5                                                                                                                                                                                                                                                                                                                                           |
| Краткое описание: создание задачи для участника                                                                                                                                                                                                                                                                                                 |
| Главные актеры: «Клиент»                                                                                                                                                                                                                                                                                                                        |
| Основной поток: <ol style="list-style-type: none"> <li>1. Клиент создаёт задачу               <ol style="list-style-type: none"> <li>1.1. Клиент указывает описание</li> <li>1.2. Клиент указывает дату задачи</li> <li>1.3. Клиент назначает исполнителя задачи</li> </ol> </li> <li>2. Исполнитель помечает задачу как выполненную</li> </ol> |
| Постусловия: список выполненных задач пополняется                                                                                                                                                                                                                                                                                               |

Рисунок 6 – Спецификация варианта использования «Добавить задачу»

Спецификация варианта использования «Изменить событие» представлена на рисунке 7.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ID: 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Краткое описание: изменения события в календаре рабочего пространства                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Главные актеры: «Клиент»                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Основной поток: <ol style="list-style-type: none"> <li>1. Пользователь выбирает «Событие» из календаря               <ol style="list-style-type: none"> <li>1.1. Клиент выбирает рабочее пространство</li> <li>1.2. Клиент пишет новую дату и время</li> <li>1.3. Клиент пишет новый текст</li> <li>1.4. Клиент сохраняет изменения</li> </ol> </li> <li>2. Клиент загружает календарь за период</li> <li>3. Клиент получает изменённое событие</li> <li>4. Сервер обновляет список событий в рабочем пространстве</li> </ol> |
| Постусловия: меняется календарь рабочего пространства                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Рисунок 7 – Спецификация варианта использования «Изменить событие»

В ходе анализа функциональных требований, посредством создания диаграммы и оформления спецификации основных функций, было создано некоторое углубленное представление разрабатываемого приложения. Также важно заметить, что регистрация и авторизация не были добавлены как

требования в силу того, что за это может отвечать отдельное разрабатываемое интегрируемое приложение [7].

#### **1.4 Анализ нефункциональных требований**

Нефункциональные требования определяют, как системе работать и что нельзя будет делать. На основе анализа планируется проведение тестирования в будущем, поэтому сейчас важно определить основные требования.

Анализ включает в себя следующие аспекты влияющие на качество системы:

- производительность – время отклика операций должно быть стандартным и пропускная способность для одновременной работы множества пользователей;
- надежность и отказоустойчивость – доступность системы должна быть долговременной, при ошибках участник должен получать сообщение с рекомендациями по исправлению;
- масштабируемость – архитектура должна позволять масштабироваться и использоваться для большего числа пользователей, база данных должна индексироваться;
- безопасность – гибкая система ролей, администратор должен разграничивать роли участников;
- удобство использования – минимальное число кликов до ключевых функций, понятный и гибкий веб-интерфейс;
- совместимость – поддержка ключевых браузеров Chrome, Yandex, Safari.

Таким образом, ключевые аспекты, влияющие на качество и стабильность системы, отражают что приложение должно быть: быстрым, надежным, безопасным, масштабируемым, удобным [8]. Такие требования гарантируют удобную функциональность в долгосрочном пользовании.

## 2 Проектирование приложения конструктора группового рабочего пространства

### 2.1 Архитектура приложения

Все микросервисы развернуты в «Docker-контейнерах», что обеспечивает изоляцию, простоту развертывания и масштабирования. «Docker Compose» используется для оркестрации контейнеров и управления их взаимодействием [9].

Для обеспечения безопасности система использует JWT-токены и аутентификацию через Telegram, что гарантирует надежную идентификацию пользователей и защиту от несанкционированного доступа [10].

Для наглядного представления физического размещения компонентов системы и их взаимодействия была разработана диаграмма развертывания. На диаграмме отображены все микросервисы, их размещение в «Docker-контейнерах», а также связи между ними через сетевые протоколы. Диаграмма развертывания представлена на рисунке 8.

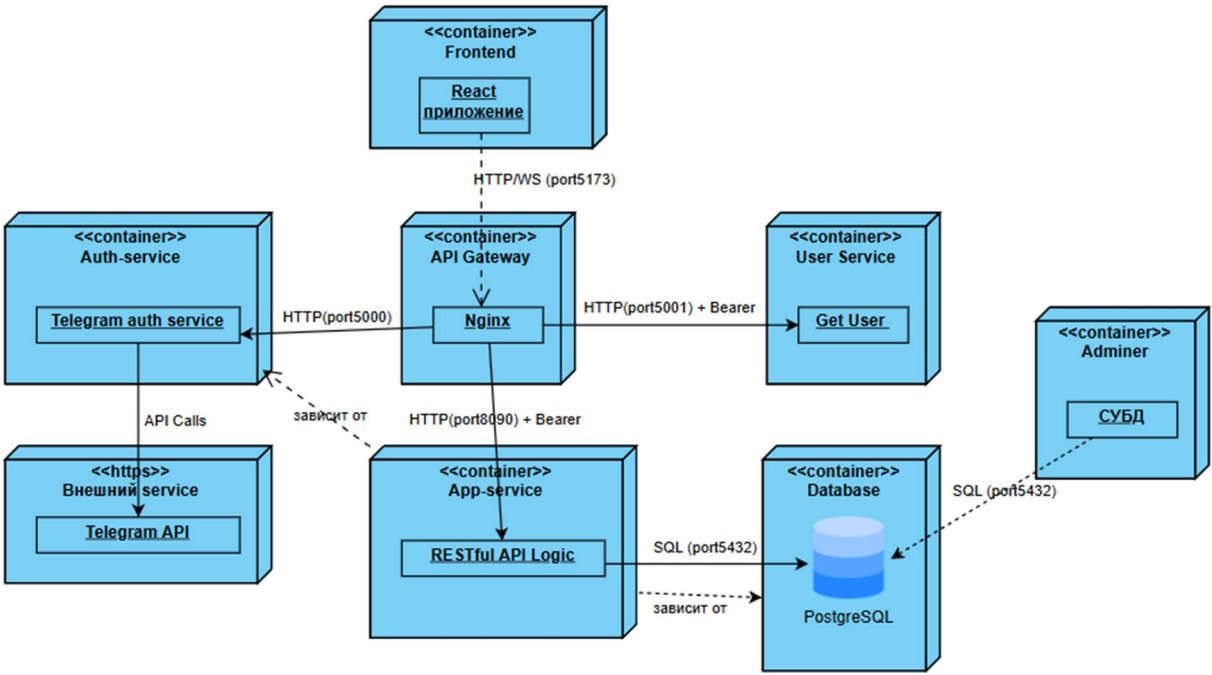


Рисунок 8 – Диаграмма развертывания

Особый интерес представляет внутренняя архитектура микросервиса «app-service», как основного компонента системы, реализующего ключевую бизнес-логику. Для детального понимания его структуры была разработана диаграмма компонентов, отражающая взаимодействие внутренних модулей, слоев и зависимостей. Диаграмма компонентов микросервиса «app-service» представлена на рисунке 17.

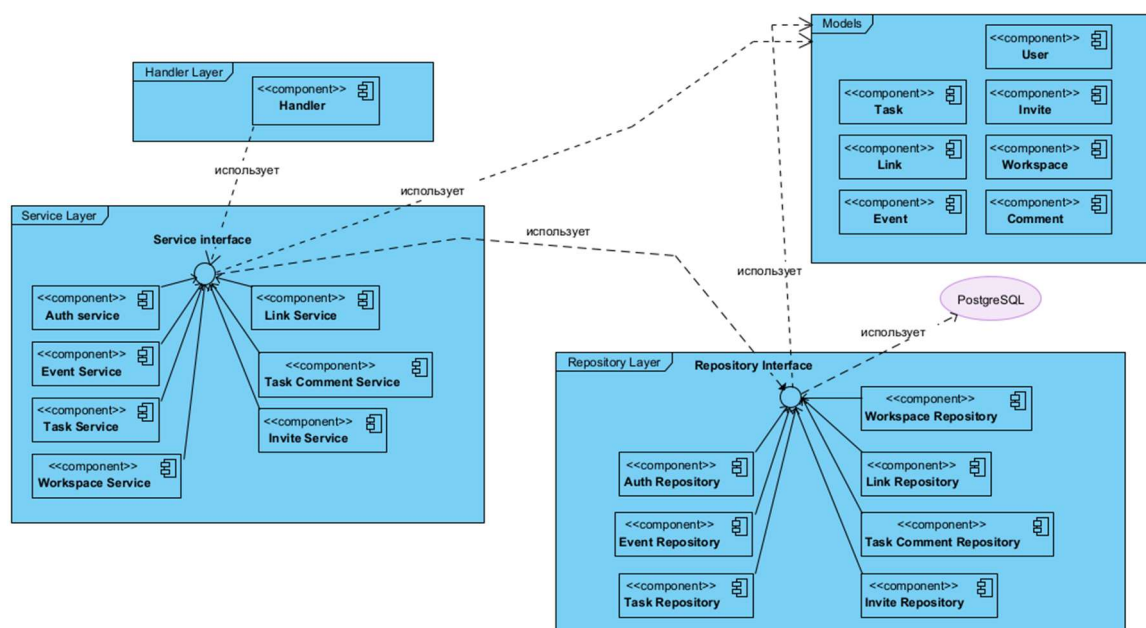


Рисунок 9 – Диаграмма компонентов

Таким образом, спроектированная микросервисная архитектура обеспечивает создание надежной, масштабируемой и легко поддерживаемой системы для организации совместной работы [11].

## 2.2 Проектирование структур данных

### 2.2.1 Концептуальная модель базы данных

Концептуальная модель данных представляет собой абстрактное описание предметной области, отражающее основные сущности системы, их характеристики и взаимосвязи между ними. Она является фундаментом для дальнейшего проектирования и разработки информационной системы, абстрагируясь от конкретных технологий реализации.

Основные цели построения концептуальной модели:

- определить ключевые сущности и их атрибуты, необходимые для функционирования системы;
- установить взаимосвязи между сущностями, отражающие бизнес-логику предметной области;
- обеспечить независимость модели от конкретных СУБД и технологий реализации;
- сформировать основу для последующего проектирования логической и физической структур базы данных.

В рамках разрабатываемой системы для совместной работы были выделены ключевые сущности, представленные в таблице 2.

Таблица 3 – Сущности концептуальной модули базы данных

| <b>Название сущности</b> | <b>Идентификатор сущности</b> | <b>Назначение сущности</b>                                                                 |
|--------------------------|-------------------------------|--------------------------------------------------------------------------------------------|
| Пользователь             | Users                         | Хранит информацию о пользователях системы, авторизованных через Telegram                   |
| Рабочее пространство     | Workspaces                    | Представляет собой контейнер для организации совместной работы                             |
| Задача                   | Tasks                         | Содержит информацию о задачах, которые необходимо выполнить в рамках рабочего пространства |
| Комментарий              | Task_comments                 | Хранит обсуждения и дополнительную информацию к задачам                                    |
| Событие                  | Events                        | Содержит информацию о запланированных мероприятиях в рабочем пространстве                  |
| Ссылка                   | Links                         | Хранит полезные ссылки и ресурсы, относящиеся к рабочему пространству                      |
| Приглашение              | Invites                       | Управляет процессом добавления новых участников в рабочее пространство                     |

Каждая сущность обладает набором атрибутов, характеризующих ее свойства. Пользователь идентифицируется через Telegram и содержит

основную информацию из профиля мессенджера. Рабочее пространство является центральной сущностью системы, объединяющей участников для совместной работы. Задачи позволяют организовать и отслеживать выполнение работ, а комментарии обеспечивают коммуникацию по конкретным задачам. События помогают планировать и координировать встречи и мероприятия. Ссылки предоставляют удобный способ хранения и доступа к внешним ресурсам. Приглашения обеспечивают контролируемый механизм расширения команды рабочего пространства.

Между сущностями установлены следующие ключевые связи:

- пользователь может администрировать несколько рабочих пространств и быть участником в других;
- рабочее пространство содержит набор задач, событий и ссылок;
- задачи могут быть назначены на конкретных пользователей и содержать комментарии;
- пользователи могут создавать приглашения для добавления новых участников в рабочее пространство.

Графическое представление концептуальной модели базы данных с отображением всех сущностей и их взаимосвязей представлено на рисунке 10.

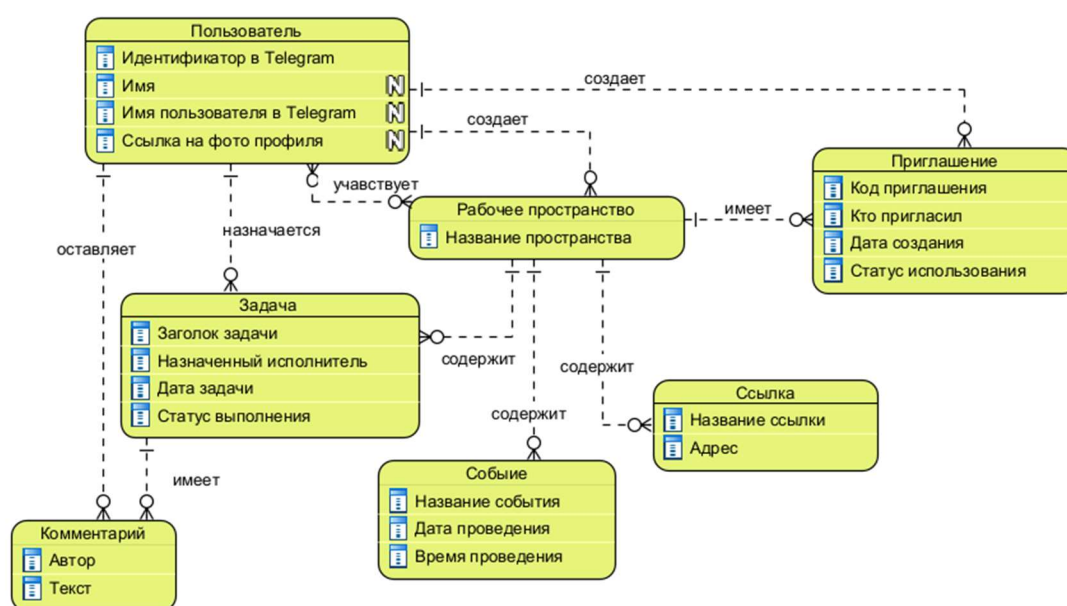


Рисунок 10 – Концептуальная модель базы данных

Разработанная концептуальная модель обеспечивает целостное представление о структуре данных системы и служит основой для дальнейшего проектирования логической и физической моделей базы данных.

### 2.2.2 Логическая модель базы данных

Логическая модель данных является техническим представлением концептуальной модели в терминах реляционной базы данных «PostgreSQL». В ходе проектирования определены конкретные таблицы, их атрибуты и типы данных, установлены связи между сущностями через механизмы первичных и внешних ключей.

Для реализации сложных связей между пользователями и рабочими пространствами применена промежуточная таблица, что позволяет гибко управлять членством пользователей в различных рабочих группах. Логическая модель проектируемой базы данных представлена на рисунке 11.

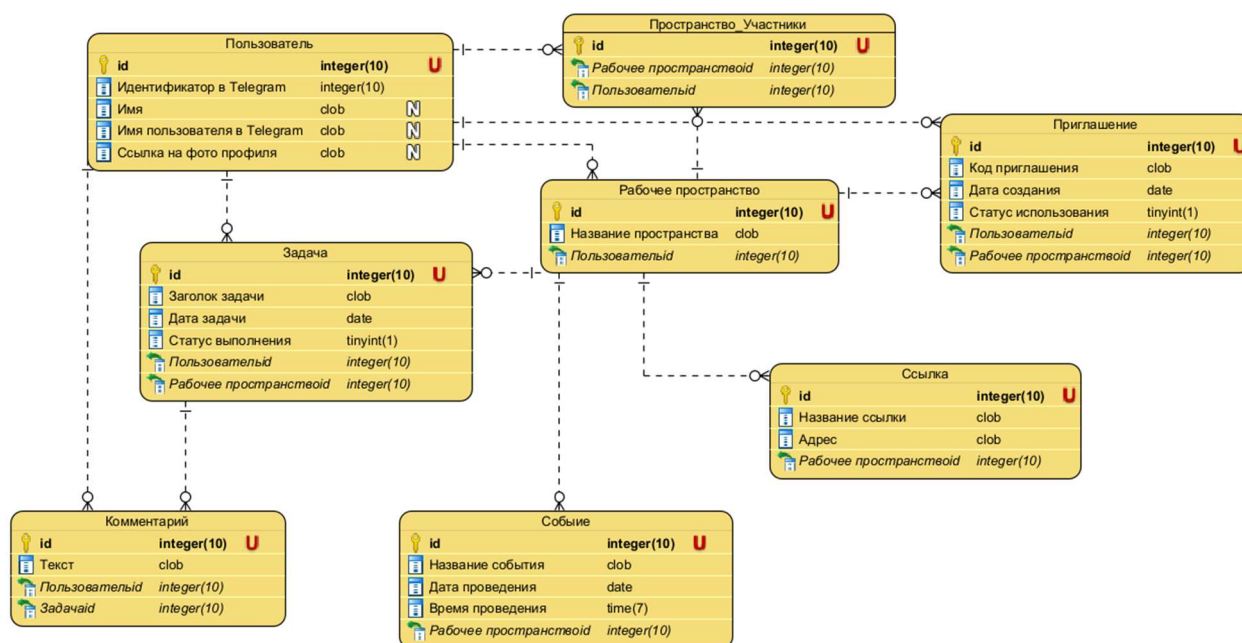


Рисунок 11 – Логическая модель базы данных

Спроектированная структура базы данных обеспечивает надежное хранение и эффективный доступ к данным в рамках разрабатываемой системы.

## 2.3 Проектирование пользовательского интерфейса

Клиентская часть разработана как современное одностраничное приложение на «React» с использованием «Vite» в качестве сборщика. Основной фокус при проектировании интерфейса сделан на простоту и удобство совместной работы пользователей в рабочих пространствах.

Ключевой функционал включает авторизацию через «Telegram», управление рабочими пространствами, работу с задачами и событиями, а также систему приглашений для новых участников. Все компоненты интерфейса спроектированы с учетом современных принципов «UI/UX» дизайна.

Для взаимодействия с сервером(бэкендом) используется «REST API» через библиотеку «axios». Все запросы проходят через «API Gateway», который распределяет их между микросервисами [12]. В приложении реализована базовая обработка ошибок и индикация состояний загрузки для улучшения пользовательского опыта.

## 2.4 Проектирование структуры данных серверной части приложения

Серверная часть построена на основе микросервисной архитектуры с использованием «Go» и фреймворка «Gin». Основной сервис приложения «app-service» спроектирован по принципам чистой архитектуры и разделен на четыре основных слоя.

Слой маршрутизации обрабатывает HTTP-запросы и определяет «API endpoints». Бизнес-логика вынесена в сервисный слой, где каждый модуль (авторизация, рабочие пространства, задачи, события) представлен отдельным интерфейсом. Доступ к данным реализован через паттерн «Repository», который инкапсулирует работу с «PostgreSQL». Слой моделей описывает структуры данных и их взаимосвязи.

Безопасность обеспечивается через аутентификацию Telegram и JWT-токены, а доступ к защищенным ресурсам контролируется «middleware». Модульная организация проекта упрощает его поддержку и развитие.

В целом, архитектура приложения обеспечивает масштабируемость, надежность и удобство поддержки. Использование современных технологий и подходов позволяет эффективно реализовать требования к совместной работе пользователей и интеграции с внешними сервисами. Такой подход способствует дальнейшему развитию системы и легкой адаптации под новые задачи.

## 2.5 Проектирование сервиса по принципу инверсии зависимостей

При проектировании сервиса был применен принцип инверсии зависимостей, являющийся частью «SOLID принципов» [13]. Этот подход позволяет создать гибкую архитектуру, где модули верхнего уровня не зависят напрямую от модулей нижнего уровня, а взаимодействуют через абстракции. В разработанном сервисе все взаимодействия между слоями осуществляются через интерфейсы, что обеспечивает слабую связанность компонентов и упрощает тестирование. Диаграмма компонентов демонстрирует описанную работу на рисунке 10.

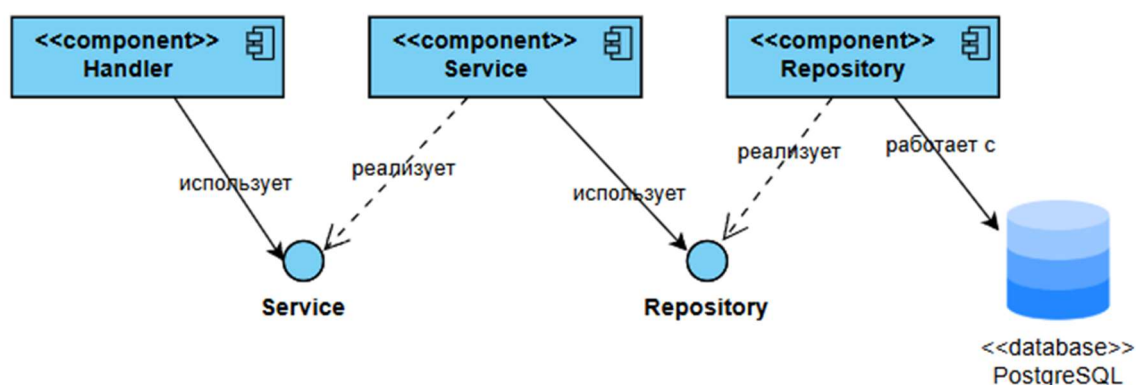


Рисунок 12 – Диаграмма компонентов сервиса с инверсией зависимостей

Архитектура сервиса разделена на три основных слоя: обработчики HTTP-запросов, бизнес-логика и работа с данными. Слой обработчиков принимает HTTP-запросы и преобразует их в вызовы методов сервисного слоя. Сервисный слой содержит всю бизнес-логику приложения и взаимодействует с хранилищем данных через интерфейсы репозитория. Слой репозитория реализует эти интерфейсы и обеспечивает работу с базой данных «PostgreSQL».

Такое разделение позволяет легко заменять компоненты системы без изменения остальных частей. Например, можно заменить «PostgreSQL» на другую базу данных, просто реализовав новый репозиторий, или добавить кэширование на уровне сервисов, не затрагивая остальные слои.

### **3 Реализация приложения конструктора группового рабочего пространства**

#### **3.1 Реализация виртуализации проекта приложения**

Виртуализация проекта реализована с использованием технологии контейнеризации «Docker». Данный подход позволяет создать изолированную среду для каждого компонента системы, что значительно упрощает процесс развертывания и обеспечивает стабильную работу приложения независимо от окружения.

В основе процесса сборки «Go-сервисов» лежит универсальный «Dockerfile», который позволяет собирать различные сервисы из единого шаблона. Сборка происходит на базе легковесного образа «golang:1.24.2-alpine», что обеспечивает оптимальный размер конечных образов. При сборке сначала устанавливаются зависимости проекта, затем копируется исходный код и выполняется компиляция приложения. Такой подход гарантирует воспроизводимость сборки и минимизирует размер конечных образов [14].

Для организации взаимодействия между контейнерами используется «Docker Compose». Он обеспечивает единую точку конфигурации всех сервисов системы и управляет их жизненным циклом. В конфигурации описаны все необходимые сервисы: API-шлюз на базе «Nginx», сервисы аутентификации и управления пользователями, основной сервис бизнес-логики, клиентское приложение и база данных «PostgreSQL». Для каждого сервиса определены необходимые параметры окружения, настройки сети и тома для хранения данных.

Особое внимание уделено надежности системы. Все сервисы работают в изолированной сети, доступ к которой строго контролируется. Настроена автоматическая проверка работоспособности базы данных и перезапуск сервисов при сбоях. Данные базы данных сохраняются в отдельном томе, что обеспечивает их сохранность при перезапуске контейнеров. Такая архитектура

позволяет легко масштабировать систему и обновлять отдельные компоненты без прерывания работы всего приложения.

### 3.2 Реализация микросервисной системы группового рабочего пространства

Система реализована на основе микросервисной архитектуры с использованием современного стека технологий. Основные компоненты системы разработаны на языке «Go», клиентская часть реализована на «React», а взаимодействие между сервисами организовано через «Nginx API Gateway».

Система включает следующие сервисы:

- «API Gateway»: обеспечивает маршрутизацию запросов и единую точку входа;
- «Auth Service»: отвечает за аутентификацию через Telegram;
- «User Service»: управляет данными пользователей;
- «App Service»: реализует основную бизнес-логику системы.

Процесс аутентификации обеспечивает базовую регистрацию пользователей в системе. После успешной регистрации пользователь получает доступ к основному функционалу системы. Методы API для аутентификации представлены в таблице 4.

Таблица 4 – Методы API для аутентификации

| URL запроса | HTTP метод | Описание                 |
|-------------|------------|--------------------------|
| /register   | POST       | Регистрация пользователя |

Каждое рабочее пространство может содержать задачи, события и ссылки. Методы API для работы с пространствами описаны в таблице 5.

Таблица 5 – Методы API для управления рабочими пространствами

| URL запроса | HTTP метод | Описание                             |
|-------------|------------|--------------------------------------|
| /workspaces | GET        | Получение списка рабочих пространств |
| /workspaces | POST       | Создание рабочего пространства       |

Продолжение таблицы 5

| URL запроса                        | HTTP метод | Описание                          |
|------------------------------------|------------|-----------------------------------|
| /workspaces/:id                    | DELETE     | Удаление рабочего пространства    |
| /workspaces/:id/members            | GET        | Получение списка участников       |
| /workspaces/:id/members/:member_id | GET        | Получение информации об участнике |
| /workspaces/:id/members/:member_id | DELETE     | Удаление участника                |

Каждая задача может иметь комментарии, что позволяет участникам обсуждать детали и ход выполнения. Методы API для работы с задачами представлены в таблице 6.

Таблица 6 – Методы API для управления задачами

| URL запроса           | HTTP метод | Описание               |
|-----------------------|------------|------------------------|
| /workspaces/:id/tasks | GET        | Получение списка задач |
| /workspaces/:id/tasks | POST       | Создание новой задачи  |
| /tasks/:id            | PUT        | Обновление задачи      |
| /tasks/:id            | DELETE     | Удаление задачи        |
| /tasks/:id/comments   | POST       | Добавление комментария |
| /tasks/:id/comments   | GET        | Получение комментариев |

События используются для организации встреч, дедлайнов и других временных меток. API методы для работы с календарем описаны в таблице 7.

Таблица 7 – Методы API для управления событиями

| URL запроса            | HTTP метод | Описание                 |
|------------------------|------------|--------------------------|
| /workspaces/:id/events | GET        | Получение списка событий |
| /workspaces/:id/events | POST       | Добавление события       |
| /events/:id            | PUT        | Обновление события       |
| /events/:id            | DELETE     | Удаление события         |

Система ссылок предоставляет возможность хранить и организовывать полезные ресурсы в рамках рабочего пространства. Методы API для работы со ссылками представлены в таблице 8.

Таблица 8 – Методы API для управления ссылками

| URL запроса           | HTTP метод | Описание                |
|-----------------------|------------|-------------------------|
| /workspaces/:id/links | GET        | Получение списка ссылок |
| /workspaces/:id/links | POST       | Добавление ссылки       |
| /links/:id            | PUT        | Обновление ссылки       |
| /links/:id            | DELETE     | Удаление ссылки         |

Система приглашений реализует механизм добавления новых участников в рабочее пространство. Методы API для работы с приглашениями описаны в таблице 9.

Таблица 9 – Методы API для управления приглашениями

| URL запроса            | HTTP метод | Описание                          |
|------------------------|------------|-----------------------------------|
| /workspaces/:id/invite | POST       | Создание приглашения              |
| /join/:invite_code     | POST       | Присоединение по коду приглашения |

Все сервисы развернуты в «Docker-контейнерах», что обеспечивает изоляцию и упрощает развертывание. Для хранения данных используется «PostgreSQL», а для удобства администрирования базы данных предусмотрен «Adminer» [15].

Система спроектирована с учетом возможности горизонтального масштабирования и легкой замены компонентов. Каждый микросервис может быть независимо обновлен или заменен без влияния на работу остальной системы.

### 3.3 Реализация базы данных

Справившись с концептуальной и логической моделью базы данных, мы не имеем проблем в реализации физической, для начала работы с данными группового рабочего пространства [16]. Физическая модель базы данных представлена на рисунке 14.

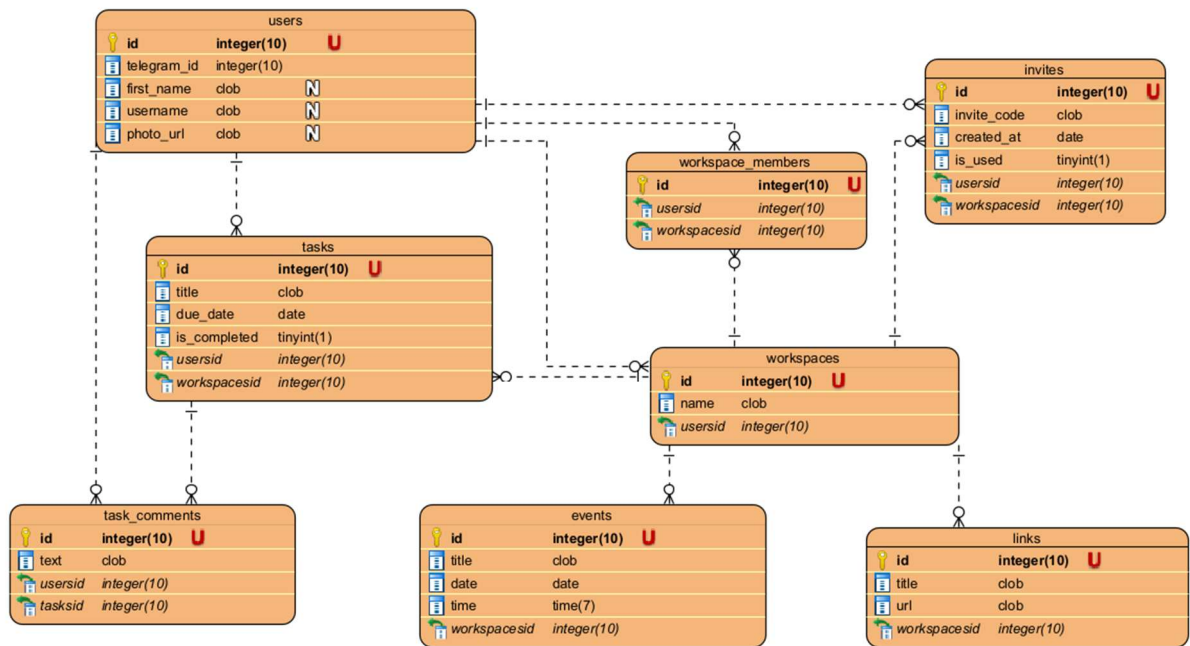


Рисунок 13 – Физическая модель базы данных

После завершения проектирования базы данных и выполнение SQL-скрипта в «Adminer», составляем схему данных в нашей СУБД, рисунок 14.

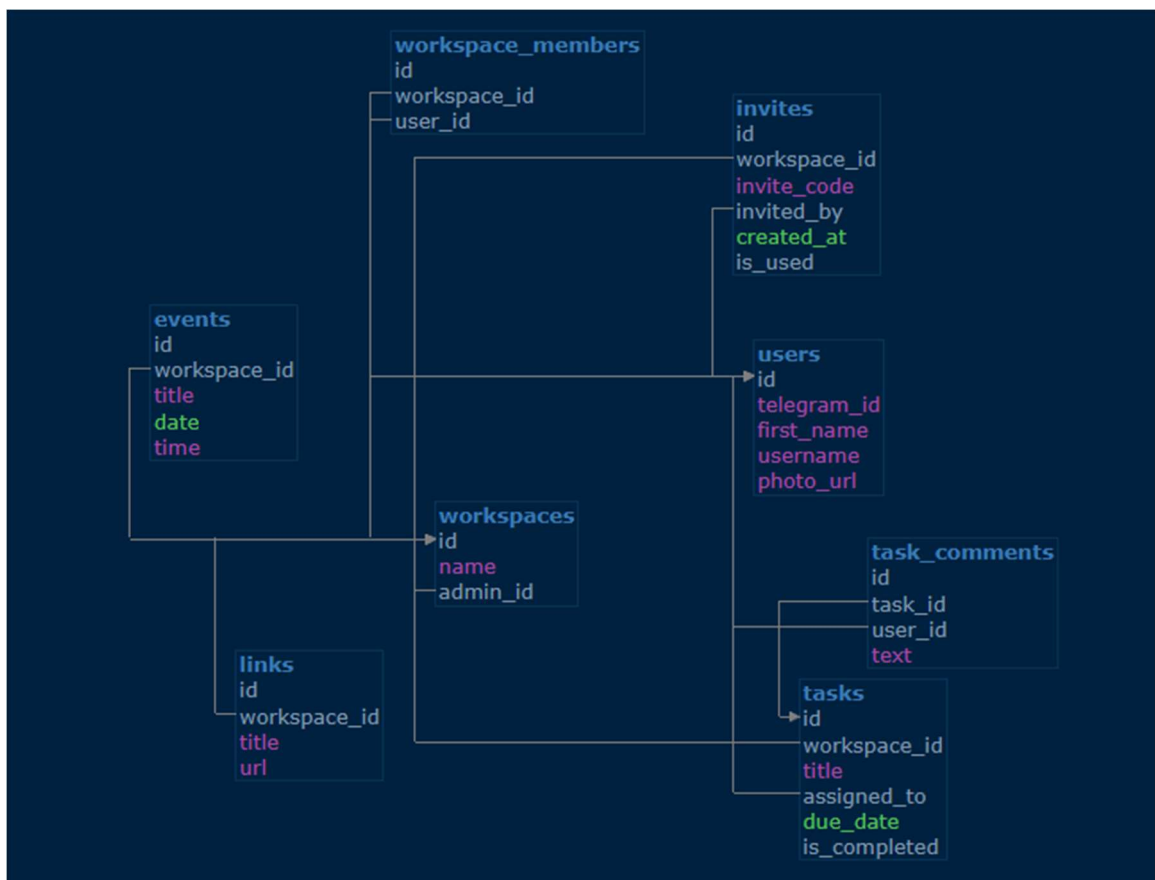


Рисунок 14 – Диаграмма схемы базы данных

Такая схема поможет оперативно анализировать требуемые запросы для получения различных типов данных.

### 3.4 Реализация клиентской части приложения

Клиентская часть системы управления рабочими пространствами отвечает за отображение пользовательского интерфейса, обработку действий пользователей и взаимодействие с серверной частью системы посредством «REST API». Ее структура спроектирована с учетом отдельных компонентов «React».

В таблице 10 приведено описание основных компонентов клиентской части системы.

Таблица 10 – Описание основных компонентов клиентской части

| Компонент | Назначение                                                      |
|-----------|-----------------------------------------------------------------|
| main.jsx  | Точка входа в приложение, инициализация React                   |
| App.jsx   | Главный компонент приложения, управление маршрутизацией и темой |
| index.css | Глобальные стили приложения                                     |
| App.css   | Стилизация основного компонента и общих элементов               |

В таблице 11 приведено описание компонентов страниц.

Таблица 11 – Описание компонентов страниц

| Компонент                | Назначение                                                 |
|--------------------------|------------------------------------------------------------|
| HomePage.jsx             | Главная страница с отображением списка рабочих пространств |
| TelegramAuthCallback.jsx | Обработка авторизации через Telegram                       |
| WorkspacePage.jsx        | Страница управления конкретным рабочим пространством       |

В таблице 12 приведено описание компонентов страниц.

Таблица 12 – Описание компонентов страниц

| Компонент                | Назначение                                                 |
|--------------------------|------------------------------------------------------------|
| HomePage.jsx             | Главная страница с отображением списка рабочих пространств |
| TelegramAuthCallback.jsx | Обработка авторизации через Telegram                       |
| WorkspacePage.jsx        | Страница управления конкретным рабочим пространством       |

В таблице 13 приведено описание компонентов интерфейса.

Таблица 13 – Описание компонентов интерфейса

| Компонент               | Назначение                                   |
|-------------------------|----------------------------------------------|
| Constructor.jsx         | Конструктор элементов рабочего пространства  |
| CreateWorkspaceForm.jsx | Форма создания нового рабочего пространства  |
| EventModal.jsx          | Модальное окно для работы с событиями        |
| LinkList.jsx            | Компонент отображения списка ссылок          |
| TaskModal.jsx           | Модальное окно управления задачами           |
| TelegramLoginButton.jsx | Кнопка авторизации через Telegram            |
| WorkspaceCard.jsx       | Карточка предпросмотра рабочего пространства |

В таблице 14 приведено описание утилит и хуков.

Таблица 14 – Описание утилит и хуков

| Компонент  | Назначение                                |
|------------|-------------------------------------------|
| useAuth.js | Хук для управления состоянием авторизации |
| api.js     | Утилиты для работы с REST API             |
| icons.jsx  | Компоненты иконок и утилиты для работы с  |

Клиентская часть системы реализована с использованием следующих технологий:

- «React.js» для построения пользовательского интерфейса;
- «React Router» для маршрутизации;
- «Bootstrap» для стилизации компонентов;
- «Vite» в качестве инструмента сборки;

- «REST API» для взаимодействия с серверной частью.

Взаимодействие с серверной частью осуществляется путем отправки HTTP-запросов к API. Полученные от сервера данные отображаются в интерфейсе с использованием React-компонентов. Система поддерживает:

- авторизацию через Telegram;
- управление рабочими пространствами;
- создание и управление задачами;
- управление событиями;
- переключение между светлой и темной темами;
- адаптивный дизайн для различных устройств.

Структура проекта организована по принципу модульности, где каждый компонент отвечает за конкретную функциональность и может быть легко модифицирован или заменен без влияния на работу остальной системы.

4      **Контроль качества программного обеспечения**

Для контроля качества мы используем функциональное тестирование, для этого нарисуем функциональную диаграмму рисунок 15.

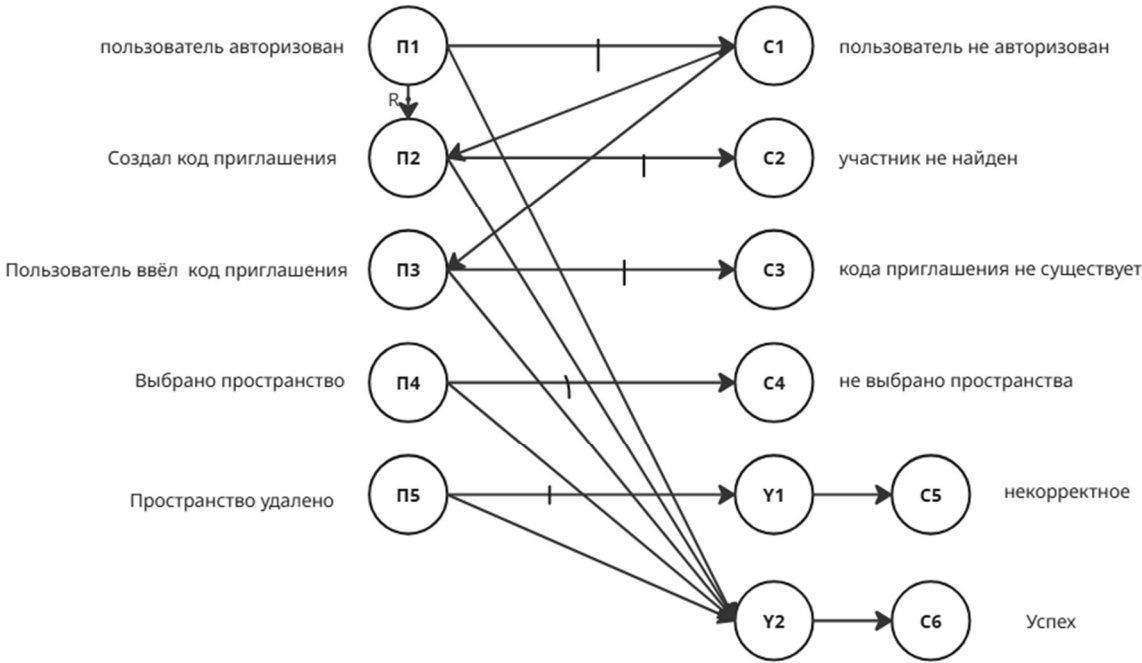


Рисунок 15 – Функциональная диаграмма приглашения участников

По реакции программы на пользовательский ввод программа должна:

- сообщить, если истек токен авторизации;
- выдать ошибку, если пространство удалено;
- выдать ошибку, если запрос идет без пространства;
- сообщить, что некорректный запрос, если он таковой;
- отобразить рабочее пространство, если успешно вошел.

На основе диаграммы составляем таблицу решений, таблица 15.

Таблица 15 – Таблица решений

|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|
| П1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1  | 0  | 1  | 0  |
| П2 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0  | 1  | 1  | 0  |
| П3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0  | 0  | 0  | 1  |

Продолжение таблицы 15

|    |   |   |   |   |   |   |   |   |   |   |   |   |   |
|----|---|---|---|---|---|---|---|---|---|---|---|---|---|
| П4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| П5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|    |   |   |   |   |   |   |   |   |   |   |   |   |   |
| С1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| С2 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| С3 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| С4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| С5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| С6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Система корректно обрабатывает все ключевые сценарии: при ошибках выдаются соответствующие сообщения, при верных данных – доступ к пространству [17]. Логика стабильна и покрывает как успешные, так и неуспешные случаи.

#### 4.1 Выбор режима тестирования

Тестирование – неотъемлемая часть разработки программного обеспечения, обеспечивающая соответствие приложения функциональным требованиям и его корректную работу [18]. В процессе тестирования выявляются дефекты, повышается надежность и качество продукта. Рассмотрим основные виды тестирования в таблице 16.

Таблица 16 – Ключевые виды тестирования

| Вид тестирования            | Описание                                                                                                                             |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Функциональное тестирование | Проверка соответствия приложения заявленным функциональным требованиям. Оценивается, выполняет ли система свои функции корректно.    |
| Интеграционное тестирование | Тестирование взаимодействия между различными модулями или компонентами системы для выявления проблем, возникающих при их интеграции. |

Продолжение таблицы 16

| <b>Вид тестирования</b>                        | <b>Описание</b>                                                                                                                                                                 |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Системное тестирование                         | Оценка работы системы в целом, включая взаимодействие с внешними компонентами, такими как базы данных и сторонние сервисы, для проверки соответствия системы общим требованиям. |
| Тестирование пользовательского интерфейса (UI) | Проверка корректности отображения и взаимодействия с элементами интерфейса, направленная на обеспечение удобства использования системы.                                         |
| Тестирование производительности                | Оценка поведения системы под нагрузкой, включая проверку скорости работы, стабильности и масштабируемости приложения.                                                           |
| Регрессионное тестирование                     | Проверка того, что новые изменения в коде не привели к появлению ошибок в ранее работавших частях системы.                                                                      |

Для приложения конструктора группового рабочего пространства подойдет функциональное тестирование.

## 4.2 Функциональное тестирование

Функциональное тестирование, известное, как тестирование черным ящиком, выполняется по ожидаемому результату и фактическому. Такое тестирование реализовано в таблице 17.

Таблица 17 – Функциональное тестирование приложения «Групповое рабочее пространство»

| <b>Тест</b> | <b>Описание</b>                           | <b>Входные данные</b>                   | <b>Ожидаемый результат</b>                             | <b>Фактический результат</b>  |
|-------------|-------------------------------------------|-----------------------------------------|--------------------------------------------------------|-------------------------------|
| 1           | Тестирование авторизации через Telegram   | Аккаунт Telegram                        | Отображение кнопки Telegram для входа                  | Все корректно.<br>Рисунок Г.1 |
| 2           | Тестирование главного меню                | Токен авторизации                       | Отображение меню, для вошедшего пользователя           | Все корректно.<br>Рисунок Г.2 |
| 3           | Тестирование входа в рабочее пространство | Токен авторизации, рабочее пространство | Пользователь успешно на странице рабочего пространства | Все корректно.<br>Рисунок Г.3 |

Продолжение таблицы 17

| <b>Тест</b> | <b>Описание</b>                          | <b>Входные данные</b>                                              | <b>Ожидаемый результат</b>                                          | <b>Фактический результат</b>   |
|-------------|------------------------------------------|--------------------------------------------------------------------|---------------------------------------------------------------------|--------------------------------|
| 4           | Тестирование выполнения задачи           | Токен авторизации, рабочее пространство, название и дата задачи    | Пользователь тестирование                                           | Все корректно.<br>Рисунок Г.4  |
| 5           | Тестирование написания комментариев      | Токен авторизации, рабочее пространство, задача, комментарий       | Отображение добавленного комментария под задачей                    | Все корректно.<br>Рисунок Г.5  |
| 6           | Тестирование добавления события          | Токен авторизации, название, дата и время события                  | Событие добавлено в календарь                                       | Все корректно.<br>Рисунок Г.6  |
| 7           | Тестирование добавления ссылки           | Токен авторизации, ссылка и название                               | Формы добавления ссылки                                             | Все корректно.<br>Рисунок Г.7  |
| 8           | Тестирование списка ссылок               | Токен авторизации, список ссылок                                   | Отображение списка ссылок                                           | Все корректно.<br>Рисунок Г.8  |
| 9           | Тестирование редактирования конструктора | Токен авторизации, задачи, комментарии, ссылки, события, участники | Открыт редактор, который сконструировал в себя рабочее пространство | Все корректно.<br>Рисунок Г.9  |
| 10          | Тестирование предпросмотра конструктора  | Токен авторизации, задачи, комментарии, ссылки, события, участники | Конструктор группового рабочего пространства отображен              | Все корректно.<br>Рисунок Г.10 |

В результате тестирования приложение показывает корректную и требуемую работу.

### 4.3 Метрики кода

В рамках разработки конструктора группового рабочего пространства была проведена предварительная оценка сложности программного обеспечения с использованием метрик Холстера. Метод Холстера позволяет определить количественные характеристики программного кода на основе анализа операторов и операндов.

Функциональность системы охватывает следующие основные модули: авторизация, создание пространств, приглашение пользователями, управление задачами и досками, работа с календарем, комментариями, метками и вложениями. Каждый из модулей включает в себя бизнес-логику, взаимодействие с API, а также работу с пользовательским интерфейсом.

Для оценки были использованы усредненные значения, полученные путем анализа типового кода, соответствующего данной архитектуре. Подсчет уникальных и общих операторов и операндов проводился ориентировочно, исходя из предполагаемой реализации.

Были получены следующие данные:

- $n1$  – количество уникальных операторов (например, =, if, for, function, {}, return): около 80;
- $n2$  – количество уникальных операндов (переменные, строки, числа): около 150 ;
- $N1$  – общее количество операторов: примерно 1000;
- $N2$  – общее количество операндов: примерно 2400.

Исходя из этих данных, были рассчитаны метрики:

Общее количество уникальных элементов (словарь)

$$n = n1 + n2 = 80 + 150 = 230$$

Общая длина программы:

$$N = N1 + N2 = 1000 + 2400 = 3400$$

Объем программы:

$$V = N \times \log_2(n) \approx 3400 \times \log_2(230) \approx 3400 \times 7.85 \approx 26,690$$

Сложность программы:

$$D = (n_1 / 2) \times (N_2 / n_2) = (80 / 2) \times (2400 / 150) = 40 \times 16 = 640$$

Усилие разработки:

$$E = D \times V = 640 \times 26,690 = 17,081,600$$

Оценка времени написания программы (в секундах):

$$T = E / 18 = 17,081,600 / 18 \approx 948,978 \text{ секунд} \approx 263.6 \text{ часов}$$

Предполагаемое количество ошибок:

$$B = V / 3000 = 26,690 / 3000 \approx 8.9$$

Таким образом, по результатам оценки можно сделать вывод, что проект имеет достаточно высокий объем и уровень сложности. Это соответствует комплексной структуре и множеству взаимодействующих компонентов, характерных для корпоративных рабочих систем [19]. Полученное значение потенциальных ошибок (около 9) также указывает на необходимость системного подхода к тестированию и контролю качества на всех этапах разработки.

5 Планирование разработки и оценка бюджета

Планирование разработки, а также оценка бюджета была оформлена в корпоративном решении Microsoft Project. Дата начала проекта указана 13 мая 2025 года. В результате планирования было выделено 18 задач [20]. Задачи проекта, а также их длительность и данные о предшествующих задачах представлены на рисунке 16.

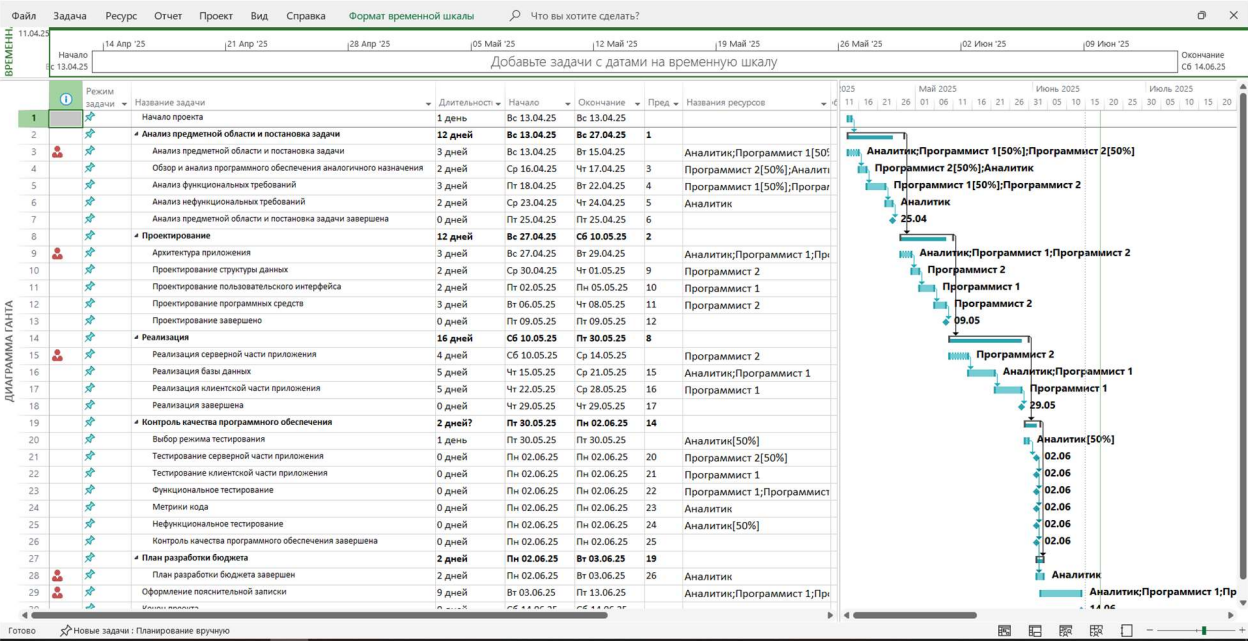


Рисунок 16 – Список задач

На рисунке 5 отображен список ресурсов, которые допускаются к выполнению задач.

|   | Название ресурса | Тип      | Единицы измерения материалов | Краткое название | Группа | Макс. единиц | Стандартная ставка | Ставка сверхурочных | Затраты на исполыз. | Начисление    | Базовый календарь |
|---|------------------|----------|------------------------------|------------------|--------|--------------|--------------------|---------------------|---------------------|---------------|-------------------|
| 1 | Программист 1    | Трудовой |                              | П                |        | 100%         | 300,00 Р/ч         | 400,00 Р/ч          | 0,00 Р              | Пропорциональ | Стандартный       |
| 2 | Программист 2    | Трудовой |                              | П                |        | 100%         | 200,00 Р/ч         | 350,00 Р/ч          | 0,00 Р              | Пропорциональ | Стандартный       |
| 3 | Аналитик         | Трудовой |                              | А                |        | 100%         | 150,00 Р/ч         | 250,00 Р/ч          | 0,00 Р              | Пропорциональ | Стандартный       |

Рисунок 17 – Список ресурсов

Диаграмма Ганта полученная в результате проектирования отображена на рисунке 17.

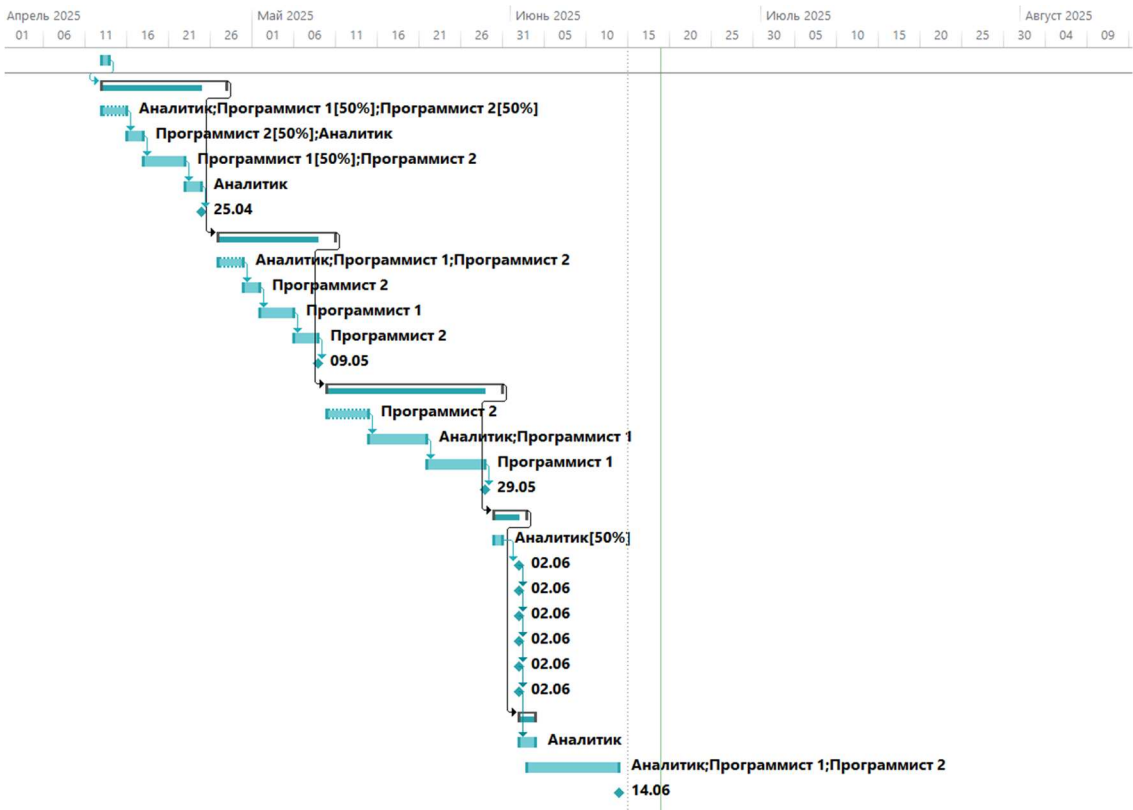


Рисунок 18 – Диаграмма Ганта

Рисунок 7 демонстрирует гистограмму затрат на задачи.

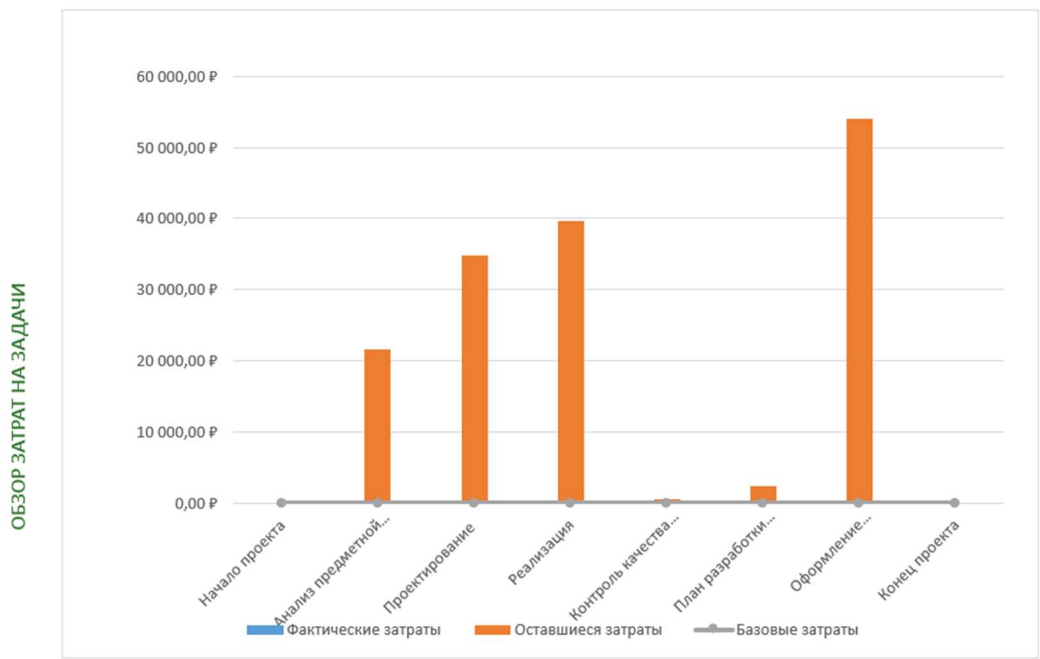


Рисунок 19 – Гистограмма затрат на задачи

Рисунок 8 отображает, как распределены временные ресурсы на проект.

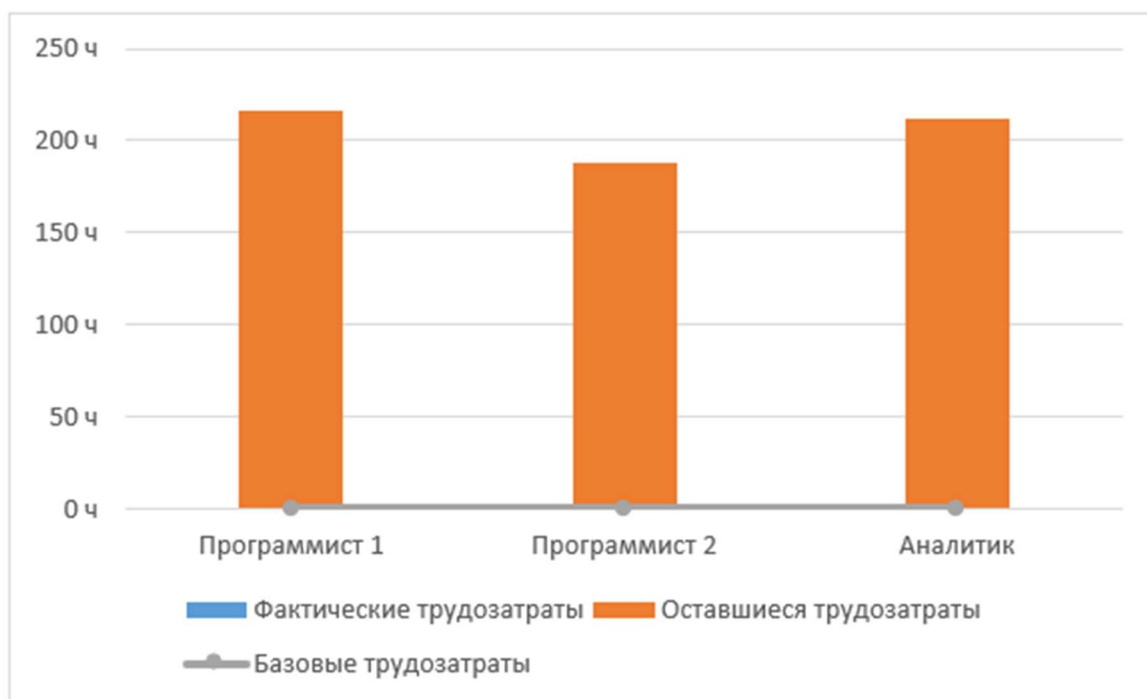


Рисунок 20 – Гистограмма загрузки ресурсов

В результате планировании разработки, выяснилось, что человеческих ресурсов понадобится на 616 часов, и оплата временных ресурсов потребует бюджета в 153 тыс. рублей. В условиях, указанных при планировании, разработка займет 62 дня из которых 43 рабочих дня.

## Заключение

В рамках выпускной квалификационной работы разработано и реализовано веб-приложение – конструктор группового рабочего пространства, предназначенное для поддержки совместной деятельности пользователей в рамках задач и событий единого проекта.

На этапе предварительного анализа была изучена предметная область коллективной работы, проанализированы существующие решения и выделены ключевые функциональные потребности целевой аудитории. Сформированы требования к архитектуре, интерфейсу, безопасности и масштабируемости приложения.

Для реализации системы была выбрана микросервисная архитектура, что позволило обеспечить независимость компонентов, гибкость масштабирования и простоту сопровождения. Основными технологиями стали Golang (серверная логика), React (интерфейс пользователя), PostgreSQL (система управления базами данных) и Docker (средство виртуализации и контейнеризации). Все компоненты разработаны и протестированы в кроссплатформенной среде с использованием WSL и Docker Desktop.

В процессе проектирования разработаны физическая и логическая модели базы данных, диаграммы компонентов, вариантов использования и развертывания с применением языка UML. Проведено функциональное тестирование ключевых пользовательских сценариев, включая создание рабочего пространства, добавление участников, управление задачами, событиями и ссылками.

Клиентская и серверная части приложения интегрированы через REST API, что обеспечивает надежную передачу данных и удобство расширения. Также реализованы базовые механизмы безопасности и устойчивости данных, соответствующие современным требованиям к веб-приложениям.

Результаты разработки демонстрируют достижение поставленных целей: создано гибкое, модульное и расширяемое решение для управления

групповыми рабочими пространствами. Приложение обладает понятным пользовательским интерфейсом, хорошо документированной архитектурой и потенциалом для дальнейшей доработки и внедрения в реальных условиях командной работы.

### Список использованных источников

1. Тихомирова О. Г. Управление проектом: комплексный подход и системный анализ. – Издательство: НИЦ ИНФРА-М. 2024. – 256 с.
2. Фрайд Джейсон, Хайнемайер Хенссон. «ReWork» Бизнес без предрассудков. Издательство: Манн, Иванов и Фербер (МИФ). 2014. – 208 с.
3. Аллен Дэвид. Как привести дела в порядок: Искусство продуктивности без стресса. Издательство: Манн, Иванов и Фербер (МИФ). 2024. – 416 с.
4. Кон Майк. Scrum: гибкая разработка ПО. Издательство: Вильямс, 2017. – 576 с.
5. Веста Д. В. Многокритериальный метод принятия управленческих решений / Гимнастика ума, 19.03.2023. – [Электронный ресурс]. – URL: <https://www.dekanblog.ru/2023/03/gimnastika-uma/mnogokriterialnyj-metod-prinyatiya-upravlencheskix-reshenij/> (дата обращения: 01.06.2025)
6. Ларман К. Применение UML 2.0 и шаблонов проектирования: введение в объектно-ориентированный анализ, проектирование и итеративную разработку / К. Ларман; [пер. с англ. и ред. А. Ю. Шелестова]. – 3-е изд. – Москва: Вильямс, 2013. – 727 с.
7. Зими́на К. И., Лапо́нина О. Р. Механизмы межсервисной аутентификации в приложениях с микросервисной архитектурой / К. И. Зими́на, О. Р. Лапо́нина. – International Journal of Open Information Technologies, 2023. – 10 с.
8. Стандарт ISO/IEC 9126 Международный стандарт – характеристики качества. – [Электронный ресурс]. – URL: [https://ru.wikipedia.org/wiki/ISO/IEC\\_9126](https://ru.wikipedia.org/wiki/ISO/IEC_9126) (дата обращения: 05.06.2025)
9. Gaël Thomas. Руководство по Docker Compose для начинающих / Хабр. – 2 мая 2019. – [Электронный ресурс]. – URL: <https://habr.com/ru/companies/ruvds/articles/450312/> (дата обращения: 06.06.2025)

10. Подробно про JWT / Хабр. – 9 сентября 2024. – [Электронный ресурс]. – URL: <https://habr.com/ru/articles/842056/> (дата обращения: 06.06.2025)
11. Сюй А. System Design Interview: Инсайты от экс-инженера Amazon / А. Сюй. – М.: Бомбора, 2022. – 352 с.
12. Миграция микросервисной архитектуры на API Gateway / Habr. – 20 января 2024. – [Электронный ресурс]. – URL: <https://habr.com/ru/articles/765944/> (дата обращения: 05.06.2025).
13. Web-Creator. «Принципы SOLID: принцип инверсии зависимостей» / Web-Creator.ru – [Электронный ресурс]. – URL: [https://web-creator.ru/articles/solid\\_the\\_dependency\\_inversion\\_principle](https://web-creator.ru/articles/solid_the_dependency_inversion_principle) (дата обращения: 07.06.2025).
14. Марков О. Работа с Docker-контейнерами в Go / PurpleSchool. – 05.06.2025. – [Электронный ресурс]. – URL: <https://purpleschool.ru/knowledge-base/article/working-with-docker-containers> (дата обращения: 07.06.2025).
15. Марсель. Adminer, Apache, PostgreSQL. Полная настройка нового сервера. / Teletype – 5 сентября 2024 – [Электронный ресурс]. – URL: [https://teletype.in/@marssak/adminer\\_instruction](https://teletype.in/@marssak/adminer_instruction) (дата обращения: 07.06.2025).
16. Физическая модель данных / AppMaster. – 31 октября 2023. – [Электронный ресурс]. – URL: <https://appmaster.io/ru/glossary/fizicheskaiamodel-dannykh> (дата обращения: 06.06.2025)
17. Баранцева Н. Decision Table – что это и как применять / Software-Testing.ru. – 20.04.2021. – [Электронный ресурс]. – URL: <https://software-testing.ru/forum/topic/40091-decision-table—chto-eto-i-kak-primeniat/> (дата обращения: 05.06.2025)
18. Кем Кейнер, Джеймс Бах, Брет Петтикорд. Тестирование программного обеспечения: контекстно ориентированный подход. – СПб.: Питер, 2025. – 352 с.
19. Метрики Холстеда и количественные метрики программного кода / Программный код и его метрики / Хабр / Habr.com. – 2022. – [Электронный

ресурс]. – URL: <https://habr.com/ru/companies/intel/articles/106082/> (дата обращения: 08.06.2025)

20. Иванов В. Ресурсно-бюджетное планирование в MS Project / MicrosoftProject.ru. – 14.01.2021. – [Электронный ресурс]. – URL: <https://www.microsoftproject.ru/2021/01/budgetinmsproject.html> (дата обращения: 08.06.2025)

**Приложение А**  
**(обязательное)**

**Листинг серверной части приложения**

Листинг А.1 - /group\_construct/app-service/app/cmd/main.go  
package main

```
import (
    "os"

    "app-service/pkg/handler"
    "app-service/pkg/repository"
    "app-service/pkg/service"

    "github.com/joho/godotenv"
    _ "github.com/lib/pq"
    "github.com/sirupsen/logrus"
    "github.com/spf13/viper"
)

func main() {
    logrus.SetFormatter(new(logrus.JSONFormatter))

    if err := initConfig(); err != nil {
        logrus.Fatalf("error initializing configs: %s",
err.Error())
    }

    if err := godotenv.Load(".env"); err != nil {
        logrus.Fatalf("Error      godotenv.Load():      %s",
err.Error())
    }

    db, err := repository.NewPostgresDB(repository.Config{
        Host:      viper.GetString("db.host"),
        Port:      viper.GetString("db.port"),
        Username:  viper.GetString("db.username"),
        Password:  os.Getenv("DB_PASSWORD"),
        DBName:    viper.GetString("db.dbname"),
        SSLMod:    viper.GetString("db.sslmode"),
    })

    if err != nil {
        logrus.Fatalf("failed to initialize db: %s",
err.Error())
    }

    repos := repository.NewRepository(db)
    services := service.NewService(repos)
    handlers := handler.NewHandler(services)

    srv := new(Server)
    if err := srv.Run(viper.GetString("port"),
handlers.InitRoutes()); err != nil {
        logrus.Fatalf("error http server: %s", err.Error())
    }
}
```

```

}

func initConfig() error {
    viper.AddConfigPath("configs")
    viper.SetConfigName("config")
    return viper.ReadInConfig()
}

```

Листинг A.2 - /group\_construct/app-service/app/cmd/server.go  
package main

```

import (
    "context"
    "errors"
    "net/http"
    "time"
)

type Server struct {
    httpServer *http.Server
}

func (s *Server) Run(port string, handler http.Handler) error
{
    s.httpServer = &http.Server{
        Addr:           ":", + port,
        Handler:        handler,
        MaxHeaderBytes: 1 << 20, // 1mb
        ReadTimeout:   10 * time.Second,
        WriteTimeout:  10 * time.Second,
    }
    return s.httpServer.ListenAndServe()
}

func (s *Server) Shutdown(ctx context.Context) error {
    if s.httpServer == nil {
        return errors.New("сервер не был запущен")
    }
    return s.httpServer.Shutdown(ctx)
}

```

Листинг A.3 - /group\_construct/app-service/app/configs/config.yml

```

jwt_secret: "supersecret"

port: "8090"

db:
    username: "postgres"

```

```
password: "qwerty"
host: "db"
port: "5432"
dbname: "postgres"
sslmode: "disable".
```

```
.
Листинг A.4 - /group_construct/app-
service/app/pkg/handler/handler.go
package handler

import (
    "app-service/pkg/service"

    "github.com/gin-gonic/gin"
)

type Handler struct {
    services *service.Service
}

func NewHandler(services *service.Service) *Handler {
    return &Handler{services: services}
}

func (h *Handler) InitRoutes() *gin.Engine {
    router := gin.New()

    router.POST("/register", h.registerUser)

    // Protected routes
    api := router.Group("/")
    api.Use(h.userIdentity)
    {
        // Workspaces
        api.POST("/workspaces", h.createWorkspace)
        api.GET("/workspaces", h.getAllWorkspaces)
        api.DELETE("/workspaces/:id", h.deleteWorkspace)
        api.DELETE("/workspaces/:id/members/:member_id",
h.deleteMember)
        api.GET("/workspaces/:id/members", h.getMembers)
        api.GET("/workspaces/:id/members/:member_id",
h.getMember)

        // Tasks
        api.POST("/workspaces/:id/tasks", h.createTask)
        api.GET("/workspaces/:id/tasks", h.getTasks)
        api.PUT("/tasks/:id", h.updateTask)
        api.DELETE("/tasks/:id", h.deleteTask)

        // Task comments
        api.POST("/tasks/:id/comments",
h.createTaskComment)
```

```

    api.GET("/tasks/:id/comments", h.getTaskComments)

    // Events
    api.POST("/workspaces/:id/events", h.createEvent)
    api.GET("/workspaces/:id/events", h.getEvents)
    api.PUT("/events/:id", h.updateEvent)
    api.DELETE("/events/:id", h.deleteEvent)

    // Links
    api.POST("/workspaces/:id/links", h.createLink)
    api.GET("/workspaces/:id/links", h.getLinks)
    api.PUT("/links/:id", h.updateLink)
    api.DELETE("/links/:id", h.deleteLink)

    // Invite
    api.POST("/workspaces/:id/invite", h.createInvite)
    api.POST("/join/:invite_code", h.joinByInvite)
}
return router
}
.
.
Листинг A.5 - /group_construct/app-
service/app/pkg/handler/list.go
package handler

import (
    "app-service/pkg/models"
    "net/http"
    "strconv"
    "strings"

    "github.com/gin-gonic/gin"
)

// --- WORKSPACE ---

func (h *Handler) createWorkspace(c *gin.Context) {
    var input struct {
        Name string `json:"name"
binding:"required"`
        AdminTelegramID string `json:"admin_telegram_id"
binding:"required"`
    }
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
        return
    }
    id, err := h.services.Workspace.Create(input.Name,
input.AdminTelegramID)
    if err != nil {

```

```

        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"id": id})
}
func (h *Handler) deleteMember(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    memberIDStr := c.Param("member_id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }
    memberID, err := strconv.Atoi(memberIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid member id"})
        return
    }
    err = h.services.Workspace.DeleteMember(workspaceID,
memberID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.Status(http.StatusOK)
}

func (h *Handler) getMembers(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }
    members, err :=
h.services.Workspace.GetMembers(workspaceID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, members)
}

func (h *Handler) getMember(c *gin.Context) {
    workspaceID, err := strconv.Atoi(c.Param("id"))
    if err != nil {

```

```

        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }

    memberID, err := strconv.Atoi(c.Param("member_id"))
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid member id"})
        return
    }

    member, err :=
h.services.Workspace.GetMember(workspaceID, memberID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, member)
}

func (h *Handler) getAllWorkspaces(c *gin.Context) {
    userID := getUserId(c)
    workspaces, err := h.services.Workspace.GetAll(userID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, workspaces)
}

func (h *Handler) deleteWorkspace(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }
    userID := getUserId(c)
    err = h.services.Workspace.Delete(workspaceID, userID)
    if err != nil {
        c.JSON(http.StatusForbidden, gin.H{"error":
err.Error()})
        return
    }
    c.Status(http.StatusOK)
}

```

```

// --- TASK ---

func (h *Handler) createTask(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }

    var input struct {
        Title    string `json:"title" binding:"required"`
        DueDate  string `json:"due_date"
binding:"required"`
    }
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
        return
    }

    userID := getUserId(c)
    task := models.Task{
        Title:      input.Title,
        AssignedTo: userID,
        DueDate:    input.DueDate,
    }

    id, err := h.services.Task.Create(workspaceID, task)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"id": id})
}

func (h *Handler) getTasks(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }
    tasks, err := h.services.Task.GetAll(workspaceID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
}

```

```

        c.JSON(http.StatusOK, tasks)
    }

    func (h *Handler) updateTask(c *gin.Context) {
        taskIDStr := c.Param("id")
        taskID, err := strconv.Atoi(taskIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid task id"})
            return
        }
        var input models.UpdateTaskInput
        if err := c.ShouldBindJSON(&input); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
            return
        }
        err = h.services.Task.Update(taskID, input)
        if err != nil {
            c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
            return
        }
        c.Status(http.StatusOK)
    }

    func (h *Handler) deleteTask(c *gin.Context) {
        taskIDStr := c.Param("id")
        taskID, err := strconv.Atoi(taskIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid task id"})
            return
        }
        err = h.services.Task.Delete(taskID)
        if err != nil {
            c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
            return
        }
        c.Status(http.StatusOK)
    }

    // --- TASK COMMENT ---

    func (h *Handler) createTaskComment(c *gin.Context) {
        taskIDStr := c.Param("id")
        taskID, err := strconv.Atoi(taskIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid task id"})
            return
        }

```

```

    }
    var input struct {
        Text string `json:"text" binding:"required"`
    }
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
        return
    }
    userID, _ := c.Get("userId")
    comment := models.Comment{
        Text:    input.Text,
        UserID:  toInt(userID),
    }
    id, err := h.services.TaskComment.Create(taskID,
comment)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"id": id})
}

func (h *Handler) getTaskComments(c *gin.Context) {
    taskIDStr := c.Param("id")
    taskID, err := strconv.Atoi(taskIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid task id"})
        return
    }
    comments, err := h.services.TaskComment.GetAll(taskID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, comments)
}

// --- EVENT ---

func (h *Handler) createEvent(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }
    var input struct {

```

```

        Title string `json:"title" binding:"required"`
        Date  string `json:"date" binding:"required"`
        Time  string `json:"time" binding:"required"`
    }
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
        return
    }
    event := models.Event{
        Title: input.Title,
        Date:  input.Date,
        Time:  input.Time,
    }
    id, err := h.services.Event.Create(workspaceID, event)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"id": id})
}

func (h *Handler) getEvents(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
        return
    }
    start := c.Query("start")
    end := c.Query("end")
    events, err := h.services.Event.GetAll(workspaceID,
start, end)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, events)
}

func (h *Handler) updateEvent(c *gin.Context) {
    eventIDStr := c.Param("id")
    eventID, err := strconv.Atoi(eventIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid event id"})
        return
    }
    var input models.UpdateEventInput

```

```

        if err := c.ShouldBindJSON(&input); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
            return
        }
        err = h.services.Event.Update(eventID, input)
        if err != nil {
            c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
            return
        }
        c.Status(http.StatusOK)
    }

    func (h *Handler) deleteEvent(c *gin.Context) {
        eventIDStr := c.Param("id")
        eventID, err := strconv.Atoi(eventIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid event id"})
            return
        }
        err = h.services.Event.Delete(eventID)
        if err != nil {
            c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
            return
        }
        c.Status(http.StatusOK)
    }

    // --- LINK ---

    func (h *Handler) createLink(c *gin.Context) {
        workspaceIDStr := c.Param("id")
        workspaceID, err := strconv.Atoi(workspaceIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
            return
        }
        var input struct {
            Title string `json:"title" binding:"required"`
            URL   string `json:"url" binding:"required"`
        }
        if err := c.ShouldBindJSON(&input); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
            return
        }
        link := models.Link{
            Title: input.Title,

```

```

        URL:    input.URL,
    }
    id, err := h.services.Link.Create(workspaceID, link)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"id": id})
}

func (h *Handler) getLinks(c *gin.Context) {
    workspaceIDStr := c.Param("id")
    workspaceID, err := strconv.Atoi(workspaceIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest,      gin.H{"error":
"invalid workspace id"})
        return
    }
    links, err := h.services.Link.GetAll(workspaceID)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, links)
}

func (h *Handler) updateLink(c *gin.Context) {
    linkIDStr := c.Param("id")
    linkID, err := strconv.Atoi(linkIDStr)
    if err != nil {
        c.JSON(http.StatusBadRequest,      gin.H{"error":
"invalid link id"})
        return
    }
    var input models.UpdateLinkInput
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest,      gin.H{"error":
err.Error()})
        return
    }
    err = h.services.Link.Update(linkID, input)
    if err != nil {
        c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
        return
    }
    c.Status(http.StatusOK)
}

func (h *Handler) deleteLink(c *gin.Context) {

```

```

        linkIDStr := c.Param("id")
        linkID, err := strconv.Atoi(linkIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid link id"})
            return
        }
        err = h.services.Link.Delete(linkID)
        if err != nil {
            c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
            return
        }
        c.Status(http.StatusOK)
    }

    // --- AUTH ---

    func (h *Handler) registerUser(c *gin.Context) {
        tokenString := c.GetHeader("Authorization")
        if tokenString == "" || !strings.HasPrefix(tokenString,
"Bearer ") {
            c.JSON(http.StatusUnauthorized, gin.H{"error":
"missing or invalid Authorization header"})
            return
        }
        user, err :=
h.services.Authorisation.ParseAndRegisterUser(tokenString)
        if err != nil {
            c.JSON(http.StatusUnauthorized, gin.H{"error":
err.Error()})
            return
        }
        c.JSON(http.StatusOK, gin.H{"id": user.ID})
    }

    // --- INVITE ---

    func (h *Handler) createInvite(c *gin.Context) {
        workspaceIDStr := c.Param("id")
        workspaceID, err := strconv.Atoi(workspaceIDStr)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error":
"invalid workspace id"})
            return
        }
        userID := getUserId(c)
        inviteCode, err :=
h.services.Invite.CreateInvite(workspaceID, userID)
        if err != nil {
            c.JSON(http.StatusInternalServerError,
gin.H{"error": err.Error()})
            return
        }
    }

```

```

    }
    c.JSON(http.StatusOK, gin.H{"invite_code": inviteCode})
}

func (h *Handler) joinByInvite(c *gin.Context) {
    inviteCode := c.Param("invite_code")
    userID := getUserId(c)
    err := h.services.Invite.JoinByInvite(inviteCode,
userID)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "joined
workspace"})
}

// --- helpers ---

func toInt(val interface{}) int {
    switch v := val.(type) {
    case float64:
        return int(v)
    case float32:
        return int(v)
    case int:
        return v
    case int64:
        return int(v)
    case string:
        i, _ := strconv.Atoi(v)
        return i
    default:
        return 0
    }
}

func getUserId(c *gin.Context) int {
    id, _ := c.Get("userId")
    return toInt(id)
}
.
.

```

Листинг A.6 - /group\_construct/app-  
service/app/pkg/handler/middleware.go

```

package handler

import (
    "github.com/gin-gonic/gin"
)

```

```
func (h *Handler) userIdentity(c *gin.Context) {
    h.services.Authorisation.UserIdentity(c)
}
```

```
·
·
```

Листинг A.7 — /group\_construct/app-  
service/app/pkg/models/models.go

```
package models
```

```
type Workspace struct {
    ID      int    `db:"id"`
    Name    string `db:"name"`
    AdminID int    `db:"admin_id"`
}
```

```
type Task struct {
    ID          int    `db:"id"`
    Title       string `db:"title"`
    AssignedTo  int    `db:"assigned_to"`
    DueDate     string `db:"due_date"`
    IsCompleted bool   `db:"is_completed"`
}
```

```
type UpdateTaskInput struct {
    IsCompleted *bool `json:"is_completed"`
    AssignedTo  *int  `json:"assigned_to"`
}
```

```
type Comment struct {
    ID      int    `db:"id"`
    Text    string `db:"text"`
    UserID  int    `db:"user_id"`
}
```

```
type Event struct {
    ID      int    `db:"id"`
    Title   string `db:"title"`
    Date    string `db:"date"`
    Time    string `db:"time"`
}
```

```
type UpdateEventInput struct {
    Title *string `json:"title"`
    Time  *string `json:"time"`
}
```

```
type Link struct {
    ID      int    `db:"id"`
    Title   string `db:"title"`
    URL     string `db:"url"`
}
```

```

type UpdateLinkInput struct {
    Title *string `json:"title"`
}

```

```

// --- INVITE ---

```

```

type Invite struct {
    ID          int      `db:"id"`
    WorkspaceID int      `db:"workspace_id"`
    InviteCode  string  `db:"invite_code"`
    InvitedBy   int      `db:"invited_by"`
    CreatedAt   string  `db:"created_at"`
    IsUsed      bool    `db:"is_used"`
}

```

Листинг A.8 - /group\_construct/app-service/app/pkg/models/User.go

```

package models

```

```

type User struct {
    ID          int      `json:"id" db:"id"`
    TelegramID  string  `json:"telegram_id" db:"telegram_id"
binding:"required"`
    FirstName   string  `json:"first_name" db:"first_name"
binding:"required"`
    Username    string  `json:"username" db:"username"
binding:"required"`
    PhotoURL    string  `json:"photo_url" db:"photo_url"`
}

```

Листинг A.9 - /group\_construct/app-service/app/pkg/repository/auth\_postgres.go

```

package repository

```

```

import (
    "app-service/pkg/models"
    "fmt"

    "github.com/jmoiron/sqlx"
)

type AuthPostgres struct {
    db *sqlx.DB
}

func NewAuthPostgres(db *sqlx.DB) *AuthPostgres {
    return &AuthPostgres{db: db}
}

func (r *AuthPostgres) RegisterUser(user models.User) (int,
error) {

```

```

        var id int
        // Check if user exists by telegram_id
        err := r.db.Get(&id, fmt.Sprintf("SELECT id FROM %s
WHERE telegram_id=$1", userTable), user.TelegramID)
        if err == nil {
            return id, nil // already exists
        }
        // Insert new user
        query := fmt.Sprintf("INSERT INTO %s (telegram_id,
first_name, username, photo_url) VALUES ($1, $2, $3, $4) RETURNING
id", userTable)
        err = r.db.QueryRow(query, user.TelegramID,
user.FirstName, user.Username, user.PhotoURL).Scan(&id)
        return id, err
    }
    .
    .
    Листинг A.10 - /group_construct/app-
service/app/pkg/repository/event_postgres.go
package repository

import (
    "app-service/pkg/models"
    "fmt"

    "github.com/jmoiron/sqlx"
)

func NewEventPostgres(db *sqlx.DB) *EventPostgres {
    return &EventPostgres{db: db}
}

func (r *EventPostgres) Create(workspaceID int, event
models.Event) (int, error) {
    var id int
    query := fmt.Sprintf("INSERT INTO %s (workspace_id,
title, date, time) VALUES ($1, $2, $3, $4) RETURNING id",
eventTable)
    err := r.db.QueryRow(query, workspaceID, event.Title,
event.Date, event.Time).Scan(&id)
    return id, err
}

func (r *EventPostgres) GetAll(workspaceID int, start, end
string) ([]models.Event, error) {
    var events []models.Event
    query := fmt.Sprintf("SELECT id, title, date, time FROM
%s WHERE workspace_id=$1 AND date >= $2 AND date <= $3",
eventTable)
    err := r.db.Select(&events, query, workspaceID, start,
end)
    return events, err
}

```

```

    }

    func (r *EventPostgres) Update(eventID int, input
models.UpdateEventInput) error {
        setQuery := ""
        args := []interface{}{}
        argId := 1

        if input.Title != nil {
            setQuery += fmt.Sprintf("title=%d,", argId)
            args = append(args, *input.Title)
            argId++
        }
        if input.Time != nil {
            setQuery += fmt.Sprintf("time=%d,", argId)
            args = append(args, *input.Time)
            argId++
        }
        if len(setQuery) == 0 {
            return nil
        }
        setQuery = setQuery[:len(setQuery)-1]
        args = append(args, eventID)
        query := fmt.Sprintf("UPDATE %s SET %s WHERE id=%d",
eventTable, setQuery, argId)
        _, err := r.db.Exec(query, args...)
        return err
    }

    func (r *EventPostgres) Delete(eventID int) error {
        query := fmt.Sprintf("DELETE FROM %s WHERE id=%d",
eventTable)
        _, err := r.db.Exec(query, eventID)
        return err
    }
    .
    .
    Листинг A.11 - /group_construct/app-
service/app/pkg/repository/invite_postgres.go
package repository

import (
    "app-service/pkg/models"
    "crypto/rand"
    "encoding/hex"
    "fmt"

    "github.com/jmoiron/sqlx"
)

type InvitePostgres struct {
    db *sqlx.DB

```

```

    }

    func NewInvitePostgres(db *sqlx.DB) *InvitePostgres {
        return &InvitePostgres{db: db}
    }

    func generateInviteCode() (string, error) {
        b := make([]byte, 8)
        _, err := rand.Read(b)
        if err != nil {
            return "", err
        }
        return hex.EncodeToString(b), nil
    }

    func (r *InvitePostgres) CreateInvite(workspaceID int,
        invitedBy int) (string, error) {
        code, err := generateInviteCode()
        if err != nil {
            return "", err
        }
        query := "INSERT INTO invites (workspace_id,
        invite_code, invited_by) VALUES ($1, $2, $3) RETURNING
        invite_code"
        var inviteCode string
        err = r.db.QueryRow(query, workspaceID, code,
        invitedBy).Scan(&inviteCode)
        return inviteCode, err
    }

    func (r *InvitePostgres) GetInviteByCode(inviteCode string)
    (*models.Invite, error) {
        var invite models.Invite
        query := "SELECT id, workspace_id, invite_code,
        invited_by, created_at, is_used FROM invites WHERE invite_code=$1"
        err := r.db.Get(&invite, query, inviteCode)
        if err != nil {
            return nil, err
        }
        return &invite, nil
    }

    func (r *InvitePostgres) JoinByInvite(inviteCode string,
        userID int) error {
        invite, err := r.GetInviteByCode(inviteCode)
        if err != nil {
            return fmt.Errorf("invalid invite code")
        }
        if invite.IsUsed {
            return fmt.Errorf("invite already used")
        }
        // Check if user is already a member
    }

```

```

        var exists bool
        checkQuery := "SELECT EXISTS(SELECT 1 FROM
workspace_members WHERE workspace_id=$1 AND user_id=$2)"
        err = r.db.Get(&exists, checkQuery, invite.WorkspaceID,
userID)
        if err != nil {
            return err
        }
        if exists {
            return fmt.Errorf("user is already a member")
        }
        // Add user to workspace_members
        insertQuery := "INSERT INTO workspace_members
(workspace_id, user_id) VALUES ($1, $2)"
        _, err = r.db.Exec(insertQuery, invite.WorkspaceID,
userID)
        if err != nil {
            return err
        }
        // Optionally, mark invite as used (if one-time)
        // _, _ = r.db.Exec("UPDATE invites SET is_used=TRUE
WHERE id=$1", invite.ID)
        return nil
    }
    .
    .
Листинг A.12 - /group_construct/app-
service/app/pkg/repository/link_postgres.go
package repository

import (
    "app-service/pkg/models"
    "fmt"

    "github.com/jmoiron/sqlx"
)

func NewLinkPostgres(db *sqlx.DB) *LinkPostgres {
    return &LinkPostgres{db: db}
}

func (r *LinkPostgres) Create(workspaceID int, link
models.Link) (int, error) {
    var id int
    query := fmt.Sprintf("INSERT INTO %s (workspace_id,
title, url) VALUES ($1, $2, $3) RETURNING id", linkTable)
    err := r.db.QueryRow(query, workspaceID, link.Title,
link.URL).Scan(&id)
    return id, err
}

```

```

func (r *LinkPostgres) GetAll(workspaceID int)
([]models.Link, error) {
    var links []models.Link
    query := fmt.Sprintf("SELECT id, title, url FROM %s WHERE
workspace_id=$1", linkTable)
    err := r.db.Select(&links, query, workspaceID)
    return links, err
}

func (r *LinkPostgres) Update(linkID int, input
models.UpdateLinkInput) error {
    setQuery := ""
    args := []interface{}{}
    argId := 1

    if input.Title != nil {
        setQuery += fmt.Sprintf("title=%d,", argId)
        args = append(args, *input.Title)
        argId++
    }
    if len(setQuery) == 0 {
        return nil
    }
    setQuery = setQuery[:len(setQuery)-1]
    args = append(args, linkID)
    query := fmt.Sprintf("UPDATE %s SET %s WHERE id=%d",
linkTable, setQuery, argId)
    _, err := r.db.Exec(query, args...)
    return err
}

func (r *LinkPostgres) Delete(linkID int) error {
    query := fmt.Sprintf("DELETE FROM %s WHERE id=$1",
linkTable)
    _, err := r.db.Exec(query, linkID)
    return err
}
.
.
.
Листинг A.13 - /group_construct/app-
service/app/pkg/repository/postgres.go
package repository

import (
    "fmt"

    "github.com/jmoiron/sqlx"
)

type Config struct {
    Host      string
    Port      string

```

```

        Username string
        Password string
        DBName    string
        SSLMod    string
    }

    const (
        userTable          = "users"
        todoListTable      = "todo_lists"
        userListTable      = "users_lists"
        todoItemsTable     = "todo_items"
        listsItemsTable    = "lists_items"
        workspaceTable     = "workspaces"
        workspaceMemberTable = "workspace_members"
        taskTable          = "tasks"
        taskCommentTable   = "task_comments"
        eventTable         = "events"
        fileTable          = "files"
        fileCommentTable   = "file_comments"
        linkTable          = "links"
    )

    func NewPostgresDB(cfg Config) (*sqlx.DB, error) {
        db, err := sqlx.Open("postgres", fmt.Sprintf("host=%s
port=%s user=%s dbname=%s password=%s sslmode=%s",
        cfg.Host,    cfg.Port,    cfg.Username,    cfg.DBName,
        cfg.Password, cfg.SSLMod))
        if err != nil {
            return nil, err
        }
        err = db.Ping()
        if err != nil {
            return nil, err
        }

        return db, nil
    }

    type WorkspacePostgres struct {
        db *sqlx.DB
    }

    type TaskPostgres struct {
        db *sqlx.DB
    }

    type TaskCommentPostgres struct {
        db *sqlx.DB
    }

    type EventPostgres struct {
        db *sqlx.DB
    }

```

```

    }

    type LinkPostgres struct {
        db *sqlx.DB
    }

    type FilePostgres struct {
        db *sqlx.DB
    }

    func NewFilePostgres(db *sqlx.DB) *FilePostgres {
        return &FilePostgres{db: db}
    }
    .
    .
    Листинг A.14 - /group_construct/app-
service/app/pkg/repository/repository.go
package repository

import (
    "app-service/pkg/models"

    "github.com/jmoiron/sqlx"
)

type Authorisation interface {
    RegisterUser(user models.User) (int, error)
}

type Workspace interface {
    Create(name string, adminTelegramID string) (int, error)
    InviteMember(workspaceID int, telegramID string) error
    DeleteMember(workspaceID, memberID int) error
    GetMembers(workspaceID int) ([]models.User, error)
    GetAll(userID int) ([]models.Workspace, error)
    Delete(workspaceID int) error
    IsWorkspaceMember(workspaceID, userID int) (bool, error)
    GetMember(workspaceID, memberID int) (models.User,
error)
    IsAdmin(workspaceID, userID int) (bool, error)
    GetByID(workspaceID int) (*models.Workspace, error)
}

type Task interface {
    Create(workspaceID int, task models.Task) (int, error)
    GetAll(workspaceID int) ([]models.Task, error)
    Update(taskID int, input models.UpdateTaskInput) error
    Delete(taskID int) error
}

type TaskComment interface {
    Create(taskID int, comment models.Comment) (int, error)

```

```

        GetAll(taskID int) ([]models.Comment, error)
    }

    type Event interface {
        Create(workspaceID int, event models.Event) (int, error)
        GetAll(workspaceID int, start, end string)
        ([]models.Event, error)
        Update(eventID int, input models.UpdateEventInput) error
        Delete(eventID int) error
    }

    type Link interface {
        Create(workspaceID int, link models.Link) (int, error)
        GetAll(workspaceID int) ([]models.Link, error)
        Update(linkID int, input models.UpdateLinkInput) error
        Delete(linkID int) error
    }

    // --- INVITE ---
    type Invite interface {
        CreateInvite(workspaceID int, invitedBy int) (string,
error)
        GetInviteByCode(inviteCode string) (*models.Invite,
error)
        JoinByInvite(inviteCode string, userID int) error
    }

    type Repository struct {
        Authorisation
        Workspace
        Task
        TaskComment
        Event
        Link
        Invite
    }

    func NewRepository(db *sqlx.DB) *Repository {
        return &Repository{
            Authorisation: NewAuthPostgres(db),
            Workspace:     NewWorkspacePostgres(db),
            Task:          NewTaskPostgres(db),
            TaskComment:   NewTaskCommentPostgres(db),
            Event:         NewEventPostgres(db),
            Link:          NewLinkPostgres(db),
            Invite:        NewInvitePostgres(db),
        }
    }

```

```

package repository

import (
    "app-service/pkg/models"
    "fmt"

    "github.com/jmoiron/sqlx"
)

func NewTaskCommentPostgres(db *sqlx.DB)
*TaskCommentPostgres {
    return &TaskCommentPostgres{db: db}
}

func (r *TaskCommentPostgres) Create(taskID int, comment
models.Comment) (int, error) {
    var id int
    query := fmt.Sprintf("INSERT INTO %s (task_id, user_id,
text) VALUES ($1, $2, $3) RETURNING id", taskCommentTable)
    err := r.db.QueryRow(query, taskID, comment.UserID,
comment.Text).Scan(&id)
    return id, err
}

func (r *TaskCommentPostgres) GetAll(taskID int)
([]models.Comment, error) {
    var comments []models.Comment
    query := fmt.Sprintf("SELECT id, text, user_id FROM %s
WHERE task_id=$1", taskCommentTable)
    err := r.db.Select(&comments, query, taskID)
    return comments, err
}
.
.
Листинг A.16 - /group_construct/app-
service/app/pkg/repository/task_postgres.go
package repository

import (
    "app-service/pkg/models"
    "fmt"

    "github.com/jmoiron/sqlx"
)

func NewTaskPostgres(db *sqlx.DB) *TaskPostgres {
    return &TaskPostgres{db: db}
}

func (r *TaskPostgres) Create(workspaceID int, task
models.Task) (int, error) {
    tx, err := r.db.Begin()

```

```

        if err != nil {
            return 0, err
        }

        // Check if assigned user exists and is a member of the
workspace
        var exists bool
        err = r.db.Get(&exists, fmt.Sprintf(`
            SELECT EXISTS (
                SELECT 1 FROM %s
                WHERE user_id = $1 AND workspace_id = $2
            )`, workspaceMemberTable), task.AssignedTo,
workspaceID)
        if err != nil {
            tx.Rollback()
            return 0, err
        }
        if !exists {
            tx.Rollback()
            return 0, fmt.Errorf("assigned user is not a member
of this workspace")
        }

        var id int
        query := fmt.Sprintf("INSERT INTO %s (workspace_id,
title, assigned_to, due_date, is_completed) VALUES ($1, $2, $3,
$4, $5) RETURNING id", taskTable)
        err = tx.QueryRow(query, workspaceID, task.Title,
task.AssignedTo, task.DueDate, task.IsCompleted).Scan(&id)
        if err != nil {
            tx.Rollback()
            return 0, err
        }

        return id, tx.Commit()
    }

    func (r *TaskPostgres) GetAll(workspaceID int)
([]models.Task, error) {
        var tasks []models.Task
        query := fmt.Sprintf("SELECT id, title, assigned_to,
due_date, is_completed FROM %s WHERE workspace_id=$1", taskTable)
        err := r.db.Select(&tasks, query, workspaceID)
        return tasks, err
    }

    func (r *TaskPostgres) Update(taskID int, input
models.UpdateTaskInput) error {
        setQuery := ""
        args := []interface{}{}
        argId := 1

```

```

        if input.IsCompleted != nil {
            setQuery += fmt.Sprintf("is_completed=%d,",
argId)
            args = append(args, *input.IsCompleted)
            argId++
        }
        if input.AssignedTo != nil {
            setQuery += fmt.Sprintf("assigned_to=%d,", argId)
            args = append(args, *input.AssignedTo)
            argId++
        }
        if len(setQuery) == 0 {
            return nil
        }
        setQuery = setQuery[:len(setQuery)-1] // remove last
comma
        args = append(args, taskID)
        query := fmt.Sprintf("UPDATE %s SET %s WHERE id=%d",
taskTable, setQuery, argId)
        _, err := r.db.Exec(query, args...)
        return err
    }

    func (r *TaskPostgres) Delete(taskID int) error {
        query := fmt.Sprintf("DELETE FROM %s WHERE id=%d",
taskTable)
        _, err := r.db.Exec(query, taskID)
        return err
    }

```

**Приложение Б**  
**(обязательное)**  
**Листинг клиентской части приложения**

```

Листинг A.1 - /group_construct/frontend/app/eslint.config.js
import js from '@eslint/js'
import globals from 'globals'
import reactHooks from 'eslint-plugin-react-hooks'
import reactRefresh from 'eslint-plugin-react-refresh'

export default [
  { ignores: ['dist'] },
  {
    files: ['**/*.{js,jsx}'],
    languageOptions: {
      ecmaVersion: 2020,
      globals: globals.browser,
      parserOptions: {
        ecmaVersion: 'latest',
        ecmaFeatures: { jsx: true },
        sourceType: 'module',
      },
    },
    plugins: {
      'react-hooks': reactHooks,
      'react-refresh': reactRefresh,
    },
    rules: {
      ...js.configs.recommended.rules,
      ...reactHooks.configs.recommended.rules,
      'no-unused-vars': ['error', { varsIgnorePattern: '^[_A-Z_]' }],
      'react-refresh/only-export-components': [
        'warn',
        { allowConstantExport: true },
      ],
    },
  },
]

```

```

Листинг A.2 - /group_construct/frontend/app/index.html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Group Construct</title>
    <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script
>
  </head>
  <body>
    <div id="root"></div>

```

```

    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>

```

Листинг A.4 - /group\_construct/frontend/app/package.json

```

{
  "name": "app",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "bootstrap": "^5.3.7",
    "github-markdown-css": "^5.8.1",
    "jquery": "^3.7.1",
    "marked": "^15.0.12",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-router-dom": "^7.6.2"
  },
  "devDependencies": {
    "@eslint/js": "^9.25.0",
    "@types/react": "^19.1.2",
    "@types/react-dom": "^19.1.2",
    "@vitejs/plugin-react": "^4.4.1",
    "eslint": "^9.25.0",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.19",
    "globals": "^16.0.0",
    "vite": "^6.3.5"
  }
}

```

Листинг A.5 - /group\_construct/frontend/app/vite.config.js

```

import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import path from 'path'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    host: '0.0.0.0',
    port: 5173,
    strictPort: true,

```

```

        allowedHosts:      ['localhost',      '.ngrok-free.app',
'.loca.lt', 'api.local'],
        watch: {
            usePolling: true,
        },

    },
    resolve: {
        alias: {
            '~bootstrap':      path.resolve(__dirname,
'node_modules/bootstrap'),
            'react-calendar':    path.resolve(__dirname,
'node_modules/react-calendar'),
        },
    },
    optimizeDeps: {
        include: ['prop-types'],
    },
})

```

Листинг А.6 - /group\_construct/frontend/app/src/App.css

```

:root {
    --tg-primary: #0088cc;
    --tg-bg: #f5f7fa;
    --tg-bg-dark: #181c23;
    --tg-card: #fff;
    --tg-card-dark: #23272f;
    --tg-text: #213547;
    --tg-text-dark: #e9e9e9;
    --tg-border: #e3e3e3;
    --tg-border-dark: #23272f;
}

body.theme-dark {
    background: var(--tg-bg-dark) !important;
    color: var(--tg-text-dark) !important;
}

.theme-dark {
    background-color: #181c23 !important;
    color: #e6eaf0 !important;
}

body.theme-dark .card {
    background: var(--tg-card-dark) !important;
    color: var(--tg-text-dark) !important;
    border: 1px solid var(--tg-border-dark) !important;
}

.theme-dark .card {
    background: #232b38 !important;
    color: #e6eaf0 !important;
    border: 1px solid #232b38;
}

```

```

}

body.theme-dark .navbar, body.theme-dark .bg-white {
  background: #23272f !important;
  color: var(--tg-text-dark) !important;
}
.theme-dark .navbar {
  background-color: #232b38 !important;
  color: #e6eaf0 !important;
  border-bottom: 1px solid #232b38;
}

body.theme-dark .alert-success {
  background: #1a222b !important;
  color: #b2f2ff !important;
  border-color: #0088cc !important;
}
.theme-dark .alert-success {
  background: #23303e;
  color: #6ab6ec;
  border-color: #23303e;
}

body.theme-dark .btn-outline-danger {
  color: #ff6b6b;
  border-color: #ff6b6b;
}
body.theme-dark .btn-outline-danger:hover {
  background: #ff6b6b;
  color: #fff;
}
.theme-dark .btn-outline-danger {
  color: #ff6b6b;
  border-color: #ff6b6b;
}
.theme-dark .btn-outline-danger:hover {
  background: #ff6b6b;
  color: #fff;
}

body.theme-dark .navbar-brand, body.theme-dark .fw-bold {
  color: #40a7e3 !important;
}
.theme-dark .navbar .navbar-brand, .theme-dark .navbar .fw-
bold {
  color: #6ab6ec !important;
}

body.theme-dark .btn {
  background: #23272f;
  color: #e9e9e9;
}

```

```

.theme-dark .btn {
  background: #232b38;
  color: #e6eaf0;
  border-color: #3a4250;
}

body.theme-dark .btn:hover {
  background: #0088cc;
  color: #fff;
}

.card.telegram-login {
  border-radius: 20px;
  background: linear-gradient(135deg, var(--tg-primary) 0%,
var(--tg-card) 100%);
  border: none;
  box-shadow: 0 4px 24px 0 rgba(0,0,0,0.10);
  transition: background 0.3s;
}

body.theme-dark .card.telegram-login {
  background: linear-gradient(135deg, #0088cc 0%, #23272f
100%);
  background: #232b38 !important;
  color: #e6eaf0 !important;
  border: 1px solid #232b38;
}

.card.telegram-login h4 {
  color: var(--tg-primary);
  font-weight: 700;
}

body.theme-dark .card.telegram-login h4 {
  color: #40a7e3;
}

.card.telegram-login p {
  color: #888;
}

body.theme-dark .card.telegram-login p {
  color: #b2b2b2;
}

#root {
  text-align: center;
  width: 100%;
}

.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: filter 300ms;
}

```

```

}
.logo:hover {
  filter: drop-shadow(0 0 2em #646cffaa);
}
.logo.react:hover {
  filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

@media (prefers-reduced-motion: no-preference) {
  a:nth-of-type(2) .logo {
    animation: logo-spin infinite 20s linear;
  }
}

.card {
  padding: 2em;
}

.read-the-docs {
  color: #888;
}

.theme-dark .bg-light {
  background: #181c23 !important;
}

.theme-dark .form-control {
  background: #232b38;
  color: #e6eaf0;
  border-color: #3a4250;
}

.theme-dark .form-control:focus {
  background: #232b38;
  color: #e6eaf0;
  border-color: #6ab6ec;
  box-shadow: 0 0 0 0.2rem rgba(106,182,236,.25);
}

/* Calendar styles */
.calendar-container {
  margin-bottom: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

```

```
.calendar-grid {
  border: 1px solid #dee2e6;
  border-radius: 8px;
  overflow: hidden;
}

.calendar-day {
  min-height: 120px;
  padding: 8px;
  transition: background-color 0.2s;
}

.calendar-day.empty {
  min-height: 120px;
}

.calendar-day.today {
  background-color: #e8f4ff;
}

.calendar-day-header {
  font-weight: 500;
  text-align: right;
  padding: 4px;
  margin-bottom: 8px;
  border-radius: 4px;
}

.today .calendar-day-header {
  background-color: #0d6efd;
  color: white;
  font-weight: bold;
  display: inline-block;
  padding: 2px 8px;
  border-radius: 12px;
}

.calendar-day-events {
  display: flex;
  flex-direction: column;
  gap: 4px;
}

.calendar-event {
  background-color: #0d6efd;
  color: white;
  padding: 4px 8px;
  border-radius: 4px;
  cursor: pointer;
  font-size: 0.85rem;
  transition: transform 0.2s, box-shadow 0.2s;
```

```

}

.calendar-event:hover {
  transform: translateY(-1px);
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
}

.event-time {
  font-size: 0.75rem;
  opacity: 0.9;
}

.event-title {
  font-weight: 500;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}

/* Responsive calendar */
@media (max-width: 768px) {
  .calendar-day {
    min-height: 100px;
  }

  .calendar-event {
    font-size: 0.75rem;
  }

  .event-time {
    font-size: 0.7rem;
  }
}

/* Links Component Styles */
.links-container {
  background: white;
  border-radius: 8px;
  padding: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  margin: 20px 0;
}

.links-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.links-header h3 {
  margin: 0;
}

```

```

    color: #333;
}

.add-link-button {
  background-color: #4CAF50;
  color: white;
  border: none;
  padding: 8px 16px;
  border-radius: 4px;
  cursor: pointer;
  font-size: 14px;
}

.add-link-button:hover {
  background-color: #45a049;
}

.add-link-form {
  display: flex;
  flex-direction: column;
  gap: 10px;
  margin-bottom: 20px;
  padding: 15px;
  border: 1px solid #ddd;
  border-radius: 4px;
}

.add-link-form input {
  padding: 8px;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-size: 14px;
}

.form-buttons {
  display: flex;
  gap: 10px;
}

.form-buttons button {
  padding: 8px 16px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 14px;
}

.form-buttons button[type="submit"] {
  background-color: #4CAF50;
  color: white;
}

```

```

.form-buttons button[type="button"] {
  background-color: #f44336;
  color: white;
}

.links-list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.link-item {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 4px;
  background-color: #f9f9f9;
}

.link-item a {
  color: #2196F3;
  text-decoration: none;
  font-size: 14px;
}

.link-item a:hover {
  text-decoration: underline;
}

.delete-link-button {
  background-color: transparent;
  color: #f44336;
  border: none;
  cursor: pointer;
  font-size: 14px;
  padding: 4px 8px;
}

.delete-link-button:hover {
  text-decoration: underline;
}

.btn-close{
  background: red !important;
}

.admin-label {
  color: #213547;
}

body.theme-dark .admin-label {

```

```

    color: #6ab6ec !important;
  }.
.
Листинг А.7 - /group_construct/frontend/app/src/App.jsx
import { BrowserRouter as Router, Routes, Route, Navigate }
from 'react-router-dom';
import HomePage from './pages/HomePage';
import      TelegramAuthCallback      from
'./pages/TelegramAuthCallback';
import { WorkspaceList } from './components/WorkspaceList';
import      {      CreateWorkspaceForm      }      from
'./components/CreateWorkspaceForm';
import { WorkspacePage } from './pages/WorkspacePage';
import './App.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'jquery';
import { useEffect, useState } from 'react';
import { useAuth } from './hooks/useAuth';

function App() {
  const [theme, setTheme] = useState(() => {
    if      (localStorage.getItem('theme'))      return
localStorage.getItem('theme');
    if      (window.matchMedia('(prefers-color-scheme:
dark)').matches) return 'dark';
    return 'light';
  });
  const { isAuth } = useAuth();

  useEffect(() => {
    if (theme === 'dark') {
      document.body.classList.add('theme-dark');
    } else {
      document.body.classList.remove('theme-dark');
    }
    localStorage.setItem('theme', theme);
  }, [theme]);

  const toggleTheme = () => setTheme(theme === 'dark' ?
'light' : 'dark');

  return (
    <Router>
      <div className={`App min-vh-100${theme === 'dark' ? '
theme-dark bg-dark' : ' bg-light'}}>
        <Routes>
          <Route path="/" element={
            isAuth ? <HomePage toggleTheme={toggleTheme}
theme={theme} /> : <Navigate to="/login" />
          } />
          <Route path="/login" element={

```

```

        !isAuth ? <HomePage toggleTheme={toggleTheme}
theme={theme} /> : <Navigate to="/" />
      } />
      <Route path="/telegram-callback"
element={<TelegramAuthCallback />} />
      <Route path="/workspace/create" element={
        isAuth ? <CreateWorkspaceForm /> : <Navigate
to="/login" />
      } />
      <Route path="/workspace/:workspaceId" element={
        isAuth ? <WorkspacePage /> : <Navigate
to="/login" />
      } />
    </Routes>
  </div>
</Router>
);
}

```

```
export default App;
```

```
.
.
```

Листинг А.8 - /group\_construct/frontend/app/src/index.css

```
/*
```

Унификация тем: используем только :root и body.theme-dark для переменных и цветов.

Все переменные для тем только в этих двух местах. Удалены дублирующие переменные и стили.

```

*/
:root {
  --tg-bg: #f5f7fa;
  --tg-bg-dark: #181c23;
  --tg-text: #213547;
  --tg-text-dark: #e9e9e9;
  --tg-primary: #0088cc;
  --tg-card: #fff;
  --tg-card-dark: #23272f;
  --tg-border: #e3e3e3;
  --tg-border-dark: #23272f;
  font-family: system-ui, Avenir, Helvetica, Arial, sans-
serif;
  line-height: 1.5;
  font-weight: 400;
  color-scheme: light dark;
  color: var(--tg-text);
  background-color: var(--tg-bg);
  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

```

```

body.theme-dark {
  --tg-bg: #181c23;
  --tg-card: #23272f;
  --tg-text: #e9e9e9;
  --tg-border: #23272f;
  background: var(--tg-bg);
  color: var(--tg-text);
}

body {
  margin: 0;
  min-width: 320px;
  min-height: 100vh;
  background: var(--tg-bg);
  color: var(--tg-text);
}

.App.theme-dark {
  background: var(--tg-bg) !important;
  color: var(--tg-text) !important;
}

.card.telegram-login {
  background: var(--tg-card);
  border: 1px solid var(--tg-border);
}

body.theme-dark .card.telegram-login {
  background: var(--tg-card);
  border: 1px solid var(--tg-border);
  color: var(--tg-text);
}

.navbar, .navbar-brand, .fw-bold {
  background: var(--tg-card);
  color: var(--tg-primary) !important;
}

body.theme-dark .navbar, body.theme-dark .navbar-brand,
body.theme-dark .fw-bold {
  background: var(--tg-card);
  color: var(--tg-primary) !important;
}

button, .btn, .btn-primary, .btn-outline-danger, .btn-
outline-secondary {
  border-radius: 8px;
  border: 1px solid var(--tg-border);
  padding: 0.6em 1.2em;
  font-size: 1em;
  font-weight: 500;
  font-family: inherit;
  background: var(--tg-card);
  color: var(--tg-text);
}

```

```

        cursor: pointer;
        transition: border-color 0.25s, background 0.25s;
    }
    body.theme-dark button, body.theme-dark .btn, body.theme-dark
    .btn-primary, body.theme-dark .btn-outline-danger, body.theme-
    dark .btn-outline-secondary {
        background: var(--tg-card);
        color: var(--tg-text);
        border-color: var(--tg-border);
    }

    .btn:hover, .btn-primary:hover, .btn-outline-danger:hover,
    .btn-outline-secondary:hover, button:hover {
        background: var(--tg-primary);
        color: #fff;
    }

    a {
        font-weight: 500;
        color: #646cff;
        text-decoration: inherit;
    }
    a:hover {
        color: #535bf2;
    }

```

Листинг А.9 - /group\_construct/frontend/app/src/main.jsx

```

import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)

```

Листинг А.10

```

/group_construct/frontend/app/src/components/Constructor.jsx
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import { marked } from 'marked';
import 'github-markdown-css/github-markdown.css';
import {
  fetchWorkspaceMembers,
  fetchWorkspaceTasks,
  fetchTaskComments,
  fetchWorkspaceEvents,
  fetchWorkspaceLinks,

```

```

    fetchWorkspaces,
  } from '../utils/api';

const Constructor = () => {
  const { workspaceId } = useParams();
  const [workspaceData, setWorkspaceData] = useState({
    workspace: null,
    members: [],
    tasks: [],
    taskComments: {},
    events: [],
    links: [],
    loading: true,
    error: null,
  });
  const [editableContent, setEditableContent] = useState('');
  const [isPreview, setIsPreview] = useState(false);

  useEffect(() => {
    const fetchAllData = async () => {
      try {
        const today = new Date();
        const threeMonthsFromNow = new Date(today);
        threeMonthsFromNow.setMonth(today.getMonth() + 3);

        const [workspaces, members, tasks, events, links] =
await Promise.all([
          fetchWorkspaces(),
          fetchWorkspaceMembers(workspaceId),
          fetchWorkspaceTasks(workspaceId),
          fetchWorkspaceEvents(
            workspaceId,
            today.toISOString().split('T')[0],
            threeMonthsFromNow.toISOString().split('T')[0]
          ),
          fetchWorkspaceLinks(workspaceId),
        ]);
        console.log(members);

        const workspace = workspaces.find(w
w.ID.toString() === workspaceId);

        const commentsPromises = tasks.map(task =>
fetchTaskComments(task.ID));
        const commentsResults = await
Promise.all(commentsPromises);
        const taskComments = {};
        tasks.forEach((task, idx) => {
          taskComments[task.ID]
Array.isArray(commentsResults[idx]) ? commentsResults[idx] : [];
        });
      }
    };
  });

```

```

        const newData = {
            workspace,
            members,
            tasks,
            taskComments,
            events,
            links,
            loading: false,
            error: null,
        };

        setWorkspaceData(newData);

setEditableContent(generateMarkdownContent(newData));
    } catch (error) {
        console.error('Error loading workspace data:',
error);
        setWorkspaceData(prev => ({
            ...prev,
            loading: false,
            error: error.message,
        }));
    }
};

    fetchAllData();
}, [workspaceId]);

const formatDate = (dateString) => {
    if (!dateString) return 'Дата не указана';
    const date = new Date(dateString);
    return date instanceof Date && !isNaN(date)
        ? date.toLocaleDateString('ru-RU')
        : 'Некорректная дата';
};

const formatTime = (timeString) => {
    return timeString ? timeString.substring(0, 5) : '';
};

const getCommentAuthor = (userId, members) => {
    const user = members.find(m => String(m.id) ===
String(userId));
    if (!user) {
        return 'Неизвестный пользователь';
    }
    return `${user.first_name}${user.username} ?`
    `(@${user.username})` : ''`;
};

const generateMarkdownContent = (data) => {
    if (!data.workspace) return '';

```

```

        return `# ${data.workspace.Name} || 'Рабочее
пространство'

    ## Участники
    ${data.members.map(member => `-${member.first_name}
    ${member.last_name} || ''} ${member.username} ?
    `(@${member.username})` : ''}`).join('\n')}

    ## Задачи
    ${data.tasks.map(task => {
        const taskComments = data.taskComments[task.ID] || [];
        return `
        \n### ${task.Title}\n- Статус:
        ${task.IsCompleted ? '✅ Выполнено' : '⌚ В процессе'}\n- Срок:
        ${formatDate(task.DueDate)}\n${taskComments.length > 0 ? `
        \n###
        Комментарии к ${task.Title}:
        \n${taskComments
            .filter(comment => comment && typeof comment ===
            'object' && comment.Text)
            .map(comment => {
                const author =
                getCommentAuthor(comment.UserID, data.members);
                return `-${comment.Text} _(${author})_`;
            })
            .join('\n')}` : ''}
        `).join('\n')}

    ## События
    ${data.events.map(event => `
    \n### ${event.Title}\n- Дата:
    ${formatDate(event.Date)} ${event.Time} ?
    ${formatTime(event.Time)}` : ''}`).join('\n')}

    ## Полезные ссылки
    ${data.links.map(link => `
    [${link.Title}](${link.URL})`
    }).join('\n')}

    if (workspaceData.loading) {
        return <div>Loading...</div>;
    }

    if (workspaceData.error) {
        return <div>Error: {workspaceData.error}</div>;
    }

    return (
        <div className="constructor-container" style={{ padding:
        '20px' }}>
            <div style={{ marginBottom: '20px', display: 'flex',
            gap: '10px' }}>
                <button
                    onClick={() => setIsPreview(prev => !prev)}
                    style={{

```

```

padding: '8px 16px',
backgroundColor: isPreview ? '#6c757d' :
'#007bff',

color: 'white',
border: 'none',
borderRadius: '4px',
cursor: 'pointer'
}}
>
{isPreview ? 'Редактировать' : 'Предпросмотр'}
</button>
</div>
{isPreview ? (
  <div
    className="markdown-body"
    style={{
      width: '100%',
      minHeight: '500px',
      padding: '16px',
      textAlign: 'left',
      border: '1px solid #ccc',
      borderRadius: '4px',
      overflowX: 'auto'
    }}
    dangerouslySetInnerHTML={{__html:
marked.parse(editableContent) }}
  />
) : (
  <textarea
    value={editableContent}
    onChange={ (e) =>
setEditableContent(e.target.value)}
    style={{
      width: '100%',
      minHeight: '500px',
      padding: '16px',
      fontFamily: 'monospace',
      fontSize: '14px',
      lineHeight: '1.5',
      border: '1px solid #ccc',
      borderRadius: '4px',
      resize: 'vertical'
    }}
  />
)}
</div>
);
};

export default Constructor;.
.
```

ЛИСТИНГ

A.11

-

```

/group_construct/frontend/app/src/components/CreateWorkspaceForm
.jsx
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { createWorkspace } from '../utils/api';
import { useAuth } from '../hooks/useAuth';

export function CreateWorkspaceForm() {
  const [name, setName] = useState('');
  const [error, setError] = useState(null);
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();
  const { user } = useAuth();

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    setError(null);

    try {
      if (!name.trim()) {
        throw new Error('Название не может быть пустым');
      }

      await createWorkspace({
        name: name.trim(),
        admin_telegram_id: String(user.id)
      });

      navigate('/');
    } catch (err) {
      setError(err.message || 'Произошла ошибка при создании
рабочего пространства');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="container pt-4">
      <div className="row justify-content-center">
        <div className="col-md-6">
          <div className="card">
            <div className="card-body">
              <h2 className="card-title text-center mb-
4">Создание рабочего пространства</h2>

              {error && (
                <div className="alert alert-danger"
role="alert">
                  {error}

```

```

    </div>
  ) }

  <form onSubmit={handleSubmit}>
    <div className="mb-3">
      <label htmlFor="name" className="form-
label">Название</label>
      <input
        type="text"
        className="form-control"
        id="name"
        value={name}
        onChange={ (e) =>
setName(e.target.value) }
        placeholder="Введите название рабочего
пространства"
        disabled={loading}
      />
    </div>

    <div className="d-grid gap-2">
      <button
        type="submit"
        className="btn btn-primary"
        disabled={loading}
      >
        {loading ? (
          <>
            <span className="spinner-border
spinner-border-sm me-2" role="status" aria-hidden="true"></span>
            Создание...
          </>
        ) : (
          'Создать'
        ) }
      </button>
      <button
        type="button"
        className="btn btn-outline-secondary"
        onClick={ () => navigate('/') }
        disabled={loading}
      >
        Отмена
      </button>
    </div>
  </form>
</div>
</div>
</div>
</div>
</div>
);

```

```

    }.
    .
    ЛИСТИНГ A.12 -
    /group_construct/frontend/app/src/components/EventModal.jsx
    import React, { useState, useEffect } from 'react';
    import { Icons } from '../utils/icons';

    export function EventModal({
      show,
      onHide,
      onSubmit,
      event = null
    }) {
      const [title, setTitle] = useState('');
      const [date, setDate] = useState('');
      const [time, setTime] = useState('');

      useEffect(() => {
        if (event) {
          setTitle(event.Title);
          // Convert UTC date to local date
          if (event.Date) {
            const utcDate = new Date(event.Date);
            const localDate = new Date(utcDate.getTime() -
            utcDate.getTimezoneOffset() * 60000);
            setDate(localDate.toISOString().split('T')[0]);
          } else {
            setDate('');
          }
          setTime(event.Time || '');
        } else {
          setTitle('');
          setDate('');
          setTime('');
        }
      }, [event]);

      const handleSubmit = (e) => {
        e.preventDefault();

        onSubmit({
          title,
          date: date,
          time
        });
      };

      if (!show) return null;

      return (
        <div className="modal d-block" tabIndex="-1" style={{
        backgroundColor: 'rgba(0,0,0,0.5)' }}>

```

```

<div className="modal-dialog">
  <div className="modal-content">
    <div className="modal-header">
      <h5      className="modal-title">{event      ?
'Редактировать событие' : 'Создать событие'}</h5>
      <button
        type="button"
        className="btn-close"
        onClick={onHide}
        aria-label="Close"
      />
    </div>
    <form onSubmit={handleSubmit}>
      <div className="modal-body">
        <div className="mb-3">
          <label      htmlFor="title"      className="form-
label">Название</label>
          <input
            type="text"
            className="form-control"
            id="title"
            value={title}
            onChange={(e) => setTitle(e.target.value)}
            required
          />
        </div>
        <div className="mb-3">
          <label      htmlFor="date"      className="form-
label">Дата</label>
          <input
            type="date"
            className="form-control"
            id="date"
            value={date}
            onChange={(e) => setDate(e.target.value)}
            required
          />
        </div>
        <div className="mb-3">
          <label      htmlFor="time"      className="form-
label">Время</label>
          <input
            type="time"
            className="form-control"
            id="time"
            value={time}
            onChange={(e) => setTime(e.target.value)}
            required
          />
        </div>
      </div>
      <div className="modal-footer">

```

```

        <button
          type="button"
          className="btn btn-secondary"
          onClick={onHide}
        >
          Отмена
        </button>
        <button
          type="submit"
          className="btn btn-primary"
        >
          {event ? 'Сохранить' : 'Создать'}
        </button>
      </div>
    </form>
  </div>
</div>
</div>

```

```
);
```

```
} .
```

```
.
```

```
Листинг
```

```
A.13
```

```
-
```

```
/group_construct/frontend/app/src/components/LinkList.jsx
```

```
import { useState, useEffect } from 'react';
```

```
import { api } from '../utils/api';
```

```
export function LinkList({ workspaceId, links, setLinks }) {
  const [newLink, setNewLink] = useState({ Title: '', URL: ''
});
```

```
  const [isAdding, setIsAdding] = useState(false);
```

```
  const [isLoading, setIsLoading] = useState(true);
```

```
  const [error, setError] = useState(null);
```

```
  const fetchLinks = async () => {
```

```
    if (!workspaceId) return;
```

```
    setIsLoading(true);
```

```
    try {
```

```
      const response =
```

```
await
```

```
api.get(`/api/app/workspaces/${workspaceId}/links`);
```

```
    setLinks(response || []);
```

```
  } catch (error) {
```

```
    setError('Failed to load links');
```

```
  } finally {
```

```
    setIsLoading(false);
```

```
  }
```

```
};
```

```
useEffect(() => {
```

```
  fetchLinks();
```

```
}, [workspaceId]);
```

```

const handleAddLink = async (e) => {
  e.preventDefault();
  try {
    await
api.post(`/api/app/workspaces/${workspaceId}/links`, {
    Title: newLink.Title,
    URL: newLink.URL,
  });
  setNewLink({ Title: '', URL: '' });
  setIsAdding(false);
  fetchLinks();
} catch (error) {
  setError('Failed to add link');
}
};

const handleDeleteLink = async (linkId) => {
  try {
    await api.delete(`/api/app/links/${linkId}`);
    fetchLinks();
  } catch (error) {
    setError('Failed to delete link');
  }
};

return (
  <div className="links-container">
    <div className="links-header">
      <h3>Полезные ссылки</h3>
      <button
        className="add-link-button"
        onClick={() => setIsAdding(true)}
      >
        Добавить ссылку
      </button>
    </div>

    {isAdding && (
      <form onSubmit={handleAddLink} className="add-link-
form">
        <input
          type="text"
          placeholder="Название ссылки"
          value={newLink.Title}
          onChange={(e) => setNewLink({ ...newLink, Title:
e.target.value })}
          required
        />
        <input
          type="url"
          placeholder="URL"
          value={newLink.URL}

```

```

        onChange={(e) => setNewLink({ ...newLink, URL:
e.target.value })}
        required
      />
      <div className="form-buttons">
        <button type="submit">Сохранить</button>
        <button type="button" onClick={() =>
setIsAdding(false)}>Отмена</button>
      </div>
    </form>
  )}

  {error && <div className="error-
message">{error}</div>}

  {isLoading ? (
    <div>Загрузка ссылок...</div>
  ) : (
    <div className="links-list">
      {links && links.length > 0 ? (
        links.map((link) => (
          <div key={link.ID} className="link-item">
            <a href={link.URL} target="_blank"
rel="noopener noreferrer">
              {link.Title}
            </a>
            <button
              className="delete-link-button"
              onClick={() => handleDeleteLink(link.ID)}
            >
              Удалить
            </button>
          </div>
        ))
      ) : (
        <div>Нет доступных ссылок</div>
      )}
    </div>
  )}
};
}.
```

Листинг

A.14

```

/group_construct/frontend/app/src/components/TaskModal.jsx
import React, { useState, useEffect } from 'react';
import { Icons } from '../utils/icons';

export function TaskModal({
  show,
  onHide,
  onSubmit,

```

```

    task = null,
    members = []
  }) {
    const [title, setTitle] = useState('');
    const [assignedTo, setAssignedTo] = useState('');
    const [dueDate, setDueDate] = useState('');
    const [isCompleted, setIsCompleted] = useState(false);

    useEffect(() => {
      if (task) {
        setTitle(task.Title);
        setAssignedTo(task.AssignedTo ? task.AssignedTo.toString() : '');
        setDueDate(task.DueDate ? task.DueDate.split('T')[0] : '');
        setIsCompleted(task.IsCompleted || false);
      } else {
        setTitle('');
        setAssignedTo('');
        setDueDate('');
        setIsCompleted(false);
      }
    }, [task]);

    const handleSubmit = (e) => {
      e.preventDefault();
      onSubmit({
        Title: title,
        AssignedTo: assignedTo ? parseInt(assignedTo) : null,
        DueDate: dueDate || null,
        IsCompleted: isCompleted
      });
    };

    if (!show) return null;

    return (
      <div className="modal d-block" tabIndex="-1" style={{
        backgroundColor: 'rgba(0,0,0,0.5)' }}>
        <div className="modal-dialog">
          <div className="modal-content">
            <div className="modal-header">
              <h5 className="modal-title">{task
                ? 'Редактировать задачу' : 'Создать задачу'}</h5>
              <button
                type="button"
                className="btn-close"
                onClick={onHide}
                aria-label="Close"
              >
            </button>
          </div>
        </div>
      </div>
    );
  }
}

```

```

    <form onSubmit={handleSubmit}>
      <div className="modal-body">
        <div className="mb-3">
          <label      htmlFor="title"      className="form-
label">Описание</label>
          <input
            type="text"
            className="form-control"
            id="title"
            value={title}
            onChange={(e) => setTitle(e.target.value)}
            required
          />
        </div>
        <div className="mb-3">
          <label htmlFor="assignedTo" className="form-
label">Назначить</label>
          <select
            className="form-select"
            id="assignedTo"
            value={assignedTo}
            onChange={(e)                                =>
setAssignedTo(e.target.value)}
          >
            <option                                value="">Выберите
участника</option>
            {members.map(member => (
              <option                                key={`member-option-
${member.ID}`} value={member.ID}>
                {member.first_name}
              </option>
            ))}
          </select>
        </div>
        <div className="mb-3">
          <label      htmlFor="dueDate"      className="form-
label">Срок</label>
          <input
            type="date"
            className="form-control"
            id="dueDate"
            value={dueDate}
            onChange={(e)                                =>
setDueDate(e.target.value)}
          />
        </div>
        {task && (
          <div className="mb-3 form-check">
            <input
              type="checkbox"
              className="form-check-input"
              id="isCompleted"

```

```

        checked={isCompleted}
        onChange={ (e) =>
setIsCompleted(e.target.checked) }
    />
    <label          className="form-check-label"
htmlFor="isCompleted">
        Выполнено
    </label>
</div>
    ) }
</div>
<div className="modal-footer">
    <button
        type="button"
        className="btn btn-secondary"
        onClick={onHide}
    >
        Отмена
    </button>
    <button
        type="submit"
        className="btn btn-primary"
    >
        {task ? 'Сохранить' : 'Создать'}
    </button>
</div>
</form>
</div>
</div>
</div>
);
} .

```

Листинг

A.15

-

```

/group_construct/frontend/app/src/components/TelegramLoginButton
.jsx

```

```

import { useEffect } from 'react';

export default function TelegramLoginButton() {
    useEffect(() => {
        const btn = document.getElementById('telegram-login-
btn');
        if (!btn) return;
        const script = document.createElement('script');
        script.async = true;
        script.src = 'https://telegram.org/js/telegram-
widget.js?22';
        script.setAttribute('data-telegram-login',
'namalka_bot');
        script.setAttribute('data-size', 'large');
        script.setAttribute('data-auth-url', '/telegram-
callback');
    });
}

```

```

        script.setAttribute('data-request-access', 'write');
        btn.appendChild(script);
        return () => {
            btn.innerHTML = '';
        };
    }, []);

    return (
        <div className="d-flex justify-content-center align-items-center" style={{ minHeight: '60vh' }}>
            <div className="card telegram-login shadow p-4">
                <div className="text-center mb-3">
                    
                    <h4 className="mb-2" style={{ color: '#0088cc' }}>Войти через Telegram</h4>
                    <p style={{ fontSize: 15 }}>Быстрый и безопасный
вход</p>
                </div>
                <div id="telegram-login-btn" className="d-flex justify-content-center"></div>
            </div>
        </div>
    );
}

```

Листинг

A.16

```

/group_construct/frontend/app/src/components/WorkspaceCard.jsx
import React, { useEffect, useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { deleteWorkspace, fetchWorkspaceMember } from
'../utils/api';
import { Icons } from '../utils/icons';

function WorkspaceCard({ workspace, onDelete, currentUserId
}) {
    const navigate = useNavigate();
    const [adminInfo, setAdminInfo] = useState(null);
    const [loadingAdmin, setLoadingAdmin] = useState(false);

    useEffect(() => {
        let mounted = true;
        async function loadAdmin() {
            if (!workspace.AdminID) return;
            setLoadingAdmin(true);
            try {
                const info = await fetchWorkspaceMember(workspace.ID,
workspace.AdminID);
                if (mounted) setAdminInfo(info);
            } catch {
                if (mounted) setAdminInfo(null);
            }
        }
    });
}

```

```

        } finally {
            if (mounted) setLoadingAdmin(false);
        }
    }
    loadAdmin();
    return () => { mounted = false; };
}, [workspace.ID, workspace.AdminID]);

const membersCount = Array.isArray(workspace.Members) ?
workspace.Members.length : (workspace.MembersCount || 0);
const tasksCount = Array.isArray(workspace.Tasks) ?
workspace.Tasks.length : (workspace.TasksCount || 0);
const isAdmin = currentUserId === workspace.AdminID;

const handleDelete = async () => {
    if (!window.confirm('Вы уверены, что хотите удалить это
рабочее пространство?')) {
        return;
    }
    try {
        await deleteWorkspace(workspace.ID);
        if (onDelete) {
            onDelete(workspace.ID);
        }
    } catch (error) {
        console.error('Error deleting workspace:', error);
        alert('Ошибка при удалении рабочего пространства');
    }
};

return (
    <div className="card mb-3">
        <div className="card-body">
            <h5 className="card-title d-flex justify-content-
between align-items-center">
                {workspace.Name}
                {isAdmin && <span className="badge bg-primary ms-
2">Admin</span>}
            </h5>
            <div className="mb-2 d-flex align-items-center"
style={{ fontSize: 14 }}>
                <span className="me-2 admin-label">Админ:</span>
                {loadingAdmin ? (
                    <span>Загрузка...</span>
                ) : adminInfo ? (
                    <>
                        {adminInfo.photo_url && (
                            <img
                                src={adminInfo.photo_url}
                                alt={adminInfo.first_name} style={{ width: 28, height: 28,
borderRadius: '50%', marginRight: 8 }} />
                        )}
                    </>
                ) : null}
            </div>
        </div>
    </div>
);

```

```

        <span                                className="fw-
bold">{adminInfo.first_name}</span>
        {adminInfo.username && <span className="ms-2
text-secondary">@{adminInfo.username}</span>}
        </>
    ) : (
        <span>Неизвестно</span>
    )}
</div>
<div className="row mb-3">
    <div className="col">
        <div className="d-flex align-items-center">
            <Icons.People className="me-2" size={20} />
            <span>{membersCount} участников</span>
        </div>
    </div>
    <div className="col">
        <div className="d-flex align-items-center">
            <Icons.Tasks className="me-2" size={20} />
            <span>{tasksCount} задач</span>
        </div>
    </div>
</div>
<div className="d-flex gap-2 mt-2">
    <Link
        to={` /workspace/${workspace.ID}`}
        className="btn btn-primary d-flex align-items-
center px-3 shadow-sm"
        style={{ fontWeight: 500 }}
    >
        <Icons.Link className="me-2" size={18} />Открыть
    </Link>
    {isAdmin && (
        <button
            className="btn btn-danger d-flex align-items-
center px-3 shadow-sm"
            style={{ fontWeight: 500 }}
            onClick={handleDelete}
        >
            <Icons.Delete      className="me-2"      size={18}
/>Удалить
        </button>
    )}
</div>
</div>
</div>
);
}

export { WorkspaceCard };
.

```

ЛИСТИНГ

A.17

-

```

/group_construct/frontend/app/src/components/WorkspaceList.jsx
import React, { useState, useEffect, useCallback } from
'react';
import { useNavigate } from 'react-router-dom';
import { fetchWorkspaces, fetchWorkspaceMembers,
fetchWorkspaceTasks, registerUser, getCurrentUserId } from
'../utils/api';
import { WorkspaceCard } from '../WorkspaceCard';
import { useAuth } from '../hooks/useAuth';

export function WorkspaceList() {
  const [workspaces, setWorkspaces] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [currentUserId, setCurrentUserId] = useState(null);
  const { user, logout } = useAuth();
  const navigate = useNavigate();

  const handleWorkspaceDelete = useCallback((workspaceId) =>
{
  setWorkspaces(prevWorkspaces =>
    prevWorkspaces.filter(w => w.ID !== workspaceId)
  );
}, []);

  useEffect(() => {
    let mounted = true;

    const loadData = async () => {
      if (!user) {
        return;
      }

      try {
        // Get current user ID first
        const userId = await getCurrentUserId();
        if (mounted) {
          setCurrentUserId(userId);
        }

        const data = await fetchWorkspaces();
        if (!data || !Array.isArray(data)) {
          setWorkspaces([]);
          setError(null);
          setLoading(false);
          return;
        }
        // Для каждого workspace подгружаем участников и
задачи

        const withDetails = await Promise.all(
          data.map(async (ws) => {

```

```

        let Members = [];
        let Tasks = [];
        try {
            Members = await fetchWorkspaceMembers(ws.ID);
        } catch {}
        try {
            Tasks = await fetchWorkspaceTasks(ws.ID);
        } catch {}
        return { ...ws, Members, Tasks };
    })
);
if (mounted) {
    setWorkspaces(withDetails);
    setError(null);
}
} catch (error) {
    console.error('Error loading data:', error);
    if (mounted) {
        if (error.message.includes('invalid token')) {
            logout();
            navigate('/login');
        } else {
            setError('Ошибка при загрузке данных');
        }
    }
} finally {
    if (mounted) {
        setLoading(false);
    }
}
};

loadData();

return () => {
    mounted = false;
};
}, [user?.id, logout, navigate]);

if (loading && !user) {
    return (
        <div className="container pt-4">
            <div className="d-flex justify-content-center">
                <div className="spinner-border text-primary"
role="status">
                    <span className="visually-
hidden">Loading...</span>
                </div>
            </div>
        </div>
    );
}

```

```

    if (error) {
      return (
        <div className="container pt-4">
          <div className="alert alert-danger" role="alert">
            {error}
          </div>
        </div>
      );
    }

    return (
      <div className="container pt-4">
        <div className="d-flex justify-content-between align-items-center mb-4">
          <h2>Мои рабочие пространства</h2>
          <button
            className="btn btn-primary"
            onClick={() => navigate('/workspace/create')}
          >
            Создать пространство
          </button>
        </div>

        {workspaces.length === 0 ? (
          <div className="alert alert-info">
            У вас пока нет рабочих пространств. Создайте новое
            или примите приглашение.
          </div>
        ) : (
          <div className="row">
            {workspaces.map(workspace => (
              <div key={workspace.ID} className="col-md-6 col-
lg-4">
                <WorkspaceCard
                  workspace={workspace}
                  currentUserId={currentUserId}
                  onDelete={handleWorkspaceDelete}
                />
              </div>
            ))}
          </div>
        )}
      </div>
    );
  }.

```

Листинг

A.18

—

/group\_construct/frontend/app/src/hooks/useAuth.js

```

import { useEffect, useState, useCallback } from 'react';
import { fetchCurrentUser } from '../utils/api';

```

```

export function useAuth() {
  const [isAuth, setIsAuth] = useState(() =>
!!localStorage.getItem('token'));
  const [user, setUser] = useState(() => {
    const userData = localStorage.getItem('user');
    return userData ? JSON.parse(userData) : null;
  });

  const logout = useCallback(() => {
    localStorage.removeItem('token');
    localStorage.removeItem('user');
    setIsAuth(false);
    setUser(null);
  }, []);

  useEffect(() => {
    const onStorage = (e) => {
      if (e.key === 'token') {
        setIsAuth(!!e.newValue);
      } else if (e.key === 'user') {
        setUser(e.newValue ? JSON.parse(e.newValue) : null);
      }
    };

    window.addEventListener('storage', onStorage);
    return () => window.removeEventListener('storage',
onStorage);
  }, []);

  useEffect(() => {
    let mounted = true;

    const initUser = async () => {
      if (isAuth && !user) {
        try {
          const data = await fetchCurrentUser();
          if (mounted) {
            setUser(data);
            localStorage.setItem('user',
JSON.stringify(data));
          }
        } catch (error) {
          console.error('Error fetching user:', error);
          if (mounted) {
            setUser(null);
            localStorage.removeItem('user');
            if (error.message.includes('invalid token')) {
              logout();
            }
          }
        }
      }
    }
  })
}

```

```

    };

    initUser();

    return () => {
        mounted = false;
    };
}, [isAuth, logout]);

return { isAuth, user, logout };
}
.
.

```

ЛИСТИНГ A.19 -

```

/group_construct/frontend/app/src/pages/HomePage.jsx
import { useAuth } from '../hooks/useAuth';
import TelegramLoginButton from
'../components/TelegramLoginButton';
import { WorkspaceList } from '../components/WorkspaceList';
import { useState } from 'react';
import { joinByInviteCode } from '../utils/api';

export default function HomePage({ toggleTheme, theme }) {
    const { isAuth, user, logout } = useAuth();
    const [inviteCode, setInviteCode] = useState('');
    const [joinStatus, setJoinStatus] = useState(null);
    const [joinLoading, setJoinLoading] = useState(false);

    if (!isAuth) return <TelegramLoginButton />;

    const handleJoin = async (e) => {
        e.preventDefault();
        setJoinStatus(null);
        setJoinLoading(true);
        try {
            await joinByInviteCode(inviteCode.trim());
            setJoinStatus({ success: true, message: 'Вы успешно
присоединились к рабочему пространству!' });
            setInviteCode('');
            window.location.reload();
        } catch (error) {
            setJoinStatus({ success: false, message: error.message
|| 'Ошибка при присоединении' });
        } finally {
            setJoinLoading(false);
        }
    };

    return (
        <div>

```

```

        <nav className={`navbar navbar-expand-lg shadow-sm mb-
4 ${theme === 'dark' ? 'theme-dark bg-dark navbar-dark' : 'navbar-
light bg-white'}}`>
        <div className="container-fluid">
        <span className="navbar-brand d-flex align-items-
center">
            {user?.photo_url && (
                <img
                    src={user.photo_url}
                    alt="avatar"
                    style={{ width: 40, height: 40, borderRadius:
'50%', marginRight: 10 }}
                />
            )}
            <span className="fw-bold" style={{ color:
'#0088cc' }}>{user?.username}</span>
        </span>
        <div className="d-flex align-items-center gap-2">
            {/* Theme toggle button */}
            <button
                className="btn btn-outline-secondary"
                title={theme === 'dark' ? 'Светлая тема' :
'Темная тема'}
                onClick={toggleTheme}
                style={{ borderRadius: '50%', width: 38,
height: 38, display: 'flex', alignItems: 'center', justifyContent:
'center' }}
            >
                {theme === 'dark' ? (
                    <svg xmlns="http://www.w3.org/2000/svg"
width="20" height="20" fill="currentColor" viewBox="0 0 24 24"
role="img" aria-label="Темная тема" title="Темная тема" > <path
d="M21.75 15.5a9 9 0 0 1-11.25-11.25A9 9 0 1 0 21.75 15.5z" />
</svg>
                ) : (
                    <svg xmlns="http://www.w3.org/2000/svg"
width="20" height="20" fill="currentColor" viewBox="0 0 24 24"
role="img" aria-label="Светлая тема" title="Светлая тема" > <path
d="M12 4a1 1 0 0 1 1-1h0a1 1 0 1 1-2 0h0a1 1 0 0 1 1 1zm0 16a1 1
0 1 1-1 1h0a1 1 0 0 1 1-1zm8-8a1 1 0 0 1 1 1h0a1 1 0 1 1-2 0h0a1
1 0 0 1 1-1zm-16 0a1 1 0 0 1 1 1h0a1 1 0 1 1-2 0h0a1 1 0 0 1 1-
1zm11.31-5.31a1 1 0 0 1 1.42 1.42h0a1 1 0 1 1-1.42-1.42zm-9.9 9.9a1
1 0 0 1 1.42 1.42h0a1 1 0 1 1-1.42-1.42zm12.02 1.42a1 1 0 0 1-
1.42-1.42h0a1 1 0 1 1 1.42 1.42zm5.69 6.1a1 1 0 1 1 1.42 1.42h0a1
1 0 1 1-1.42-1.42zM12 7a5 5 0 1 1 0 10 5 5 0 0 1 0-10z" /> </svg>
                )}
            </button>
            <button className="btn btn-outline-danger ms-2"
onClick={logout}>
                Выйти
            </button>
        </div>

```

```

        </div>
      </nav>
      <div className="container-fluid">
        { /* Invite code join form */ }
        <div className="m-auto" style={{ maxWidth: 400 }}>
          <form      onSubmit={handleJoin}      className="d-flex
flex-column gap-2">
            <label                                htmlFor="invite-code-input"
className="form-label fw-bold">Присоединиться по пригласительному
коду</label>
            <div className="input-group">
              <input
                id="invite-code-input"
                type="text"
                className="form-control"
                placeholder="Введите код..."
                value={inviteCode}
                onChange={e =>
setInviteCode(e.target.value)}
                disabled={joinLoading}
                required
              />
              <button      className="btn      btn-primary"
type="submit" disabled={joinLoading || !inviteCode.trim()}>
                {joinLoading ? '...' : 'Присоединиться'}
              </button>
            </div>
            {joinStatus && (
              <div      className={ `alert      mt-2
${joinStatus.success      ?      'alert-success'      :      'alert-
danger' } ` }>{joinStatus.message}</div>
            )}
          </form>
        </div>
        <WorkspaceList />
      </div>
    </div>
  );
}

```

·  
·

ЛИСТИНГ

A.20

—

```

/group_construct/frontend/app/src/pages/TelegramAuthCallback.jsx
import { useEffect, useState } from 'react';
import { useLocation, useNavigate } from 'react-router-dom';
import { telegramAuthCallback } from '../utils/api';

export default function TelegramAuthCallback() {
  const location = useLocation();
  const navigate = useNavigate();
  const [status, setStatus] = useState('loading');
  const [error, setError] = useState('');

```

```

useEffect(() => {
  const params = new URLSearchParams(location.search);
  const data = {};
  params.forEach((value, key) => {
    data[key] = value;
  });

  telegramAuthCallback(data)
    .then((json) => {
      localStorage.setItem('token', json.token);
      setStatus('success');
      navigate('/');
    })
    .catch((e) => {
      setStatus('error');
      setError(e.message);
    });
}, [location, navigate]);

if (status === 'loading') return <div>Авторизация через
Telegram...</div>;
if (status === 'error') return <div>Ошибка авторизации:
{error}</div>;
return null;
}
.
.

```

Листинг

A.21

—

```

/group_construct/frontend/app/src/pages/WorkspacePage.jsx
import React, { useState, useEffect, useCallback } from
'react';
import { useParams, useNavigate } from 'react-router-dom';
import {
  fetchWorkspaceMembers,
  fetchWorkspaceTasks,
  fetchWorkspaceEvents,
  fetchWorkspaceLinks,
  fetchWorkspaces,
  fetchTaskComments,
  addTaskComment,
  createTask,
  updateTask,
  deleteTask,
  createEvent,
  updateEvent,
  deleteEvent,
  createInvite,
  removeMember,
  fetchWorkspaceMember
} from '../utils/api';
import { useAuth } from '../hooks/useAuth';

```

```

import { TaskModal } from '../components/TaskModal';
import { EventModal } from '../components/EventModal';
import { LinkList } from '../components/LinkList';
import { Icons } from '../utils/icons';
import Constructor from '../components/Constructor';

export function WorkspacePage() {
  const { workspaceId } = useParams();
  const navigate = useNavigate();
  const { user, logout } = useAuth();
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [workspace, setWorkspace] = useState(null);
  const [members, setMembers] = useState([]);
  const [tasks, setTasks] = useState([]);
  const [events, setEvents] = useState([]);
  const [links, setLinks] = useState([]);
  const [activeTab, setActiveTab] = useState('tasks');
  const [showTaskModal, setShowTaskModal] = useState(false);
  const [showEventModal, setShowEventModal] =
useState(false);
  const [selectedTask, setSelectedTask] = useState(null);
  const [selectedEvent, setSelectedEvent] = useState(null);
  const [inviteCode, setInviteCode] = useState(null);
  const [showInviteModal, setShowInviteModal] =
useState(false);
  const [newLinkTitle, setNewLinkTitle] = useState('');
  const [newLinkUrl, setNewLinkUrl] = useState('');
  const [taskComments, setTaskComments] = useState({});
  const [newComment, setNewComment] = useState('');
  const [selectedTaskForComment, setSelectedTaskForComment]
= useState(null);
  const [commentUsers, setCommentUsers] = useState({});
  const [currentMonth, setCurrentMonth] = useState(new
Date());
  const [taskAssignees, setTaskAssignees] = useState({});
  const [expandedComments, setExpandedComments] =
useState({});

  // Функция для загрузки комментариев
  const loadTaskComments = async (tasksData) => {
    if (!Array.isArray(tasksData) || tasksData.length === 0)
return;

    try {
      const commentsPromises = tasksData.map(task =>
fetchTaskComments(task.ID));
      const commentsResults = await
Promise.all(commentsPromises);
      const commentsMap = {};
      tasksData.forEach((task, index) => {
        commentsMap[task.ID] = commentsResults[index] || [];
      });
    }
  }
}

```

```

});
setTaskComments(commentsMap);

// Собираем уникальные ID пользователей из комментариев
const userIds = new Set();
Object.values(commentsMap).flat().forEach(comment => {
  if (comment && comment.UserID) {
    userIds.add(comment.UserID);
  }
});

// Загружаем информацию о пользователях
const usersPromises = Array.from(userIds).map(userId =>
  fetchWorkspaceMember(workspaceId, userId)
);
const users = await Promise.all(usersPromises);
const usersMap = {};
users.forEach(user => {
  if (user) {
    usersMap[user.id] = user;
  }
});
setCommentUsers(usersMap);
} catch (error) {
  console.error('Error loading comments:', error);
}
};

// Функция для загрузки информации о пользователях, которым
назначены задачи
const loadTaskAssignees = async (tasksData) => {
  if (!Array.isArray(tasksData) || tasksData.length === 0)
    return;

  try {
    // Собираем уникальные ID пользователей из задач
    const userIds = new Set(tasksData.map(task =>
      task.AssignedTo).filter(id => id !== null));

    // Загружаем информацию о пользователях
    const usersPromises = Array.from(userIds).map(userId =>
      fetchWorkspaceMember(workspaceId, userId)
    );
    const users = await Promise.all(usersPromises);

    // Создаем мапу пользователей
    const usersMap = {};
    users.forEach(user => {
      if (user) {
        usersMap[user.id] = user;
      }
    });
  }
};

```

```

        setTaskAssignees(usersMap);
    } catch (error) {
        console.error('Error loading task assignees:', error);
    }
};

// Функция для загрузки всех данных рабочего пространства
const loadWorkspaceData = async () => {
    try {
        // Получаем список всех рабочих пространств
        const workspaces = await fetchWorkspaces();
        const currentWorkspace = workspaces.find(w =>
w.ID.toString() === workspaceId);

        if (!currentWorkspace) {
            setError('Рабочее пространство не найдено');
            navigate('/');
            return;
        }

        setWorkspace(currentWorkspace);

        // Загружаем остальные данные последовательно
        try {
            const membersData = await
fetchWorkspaceMembers(workspaceId);
            setMembers(membersData || []);
        } catch (error) {
            console.error('Error loading members:', error);
            setMembers([]);
        }

        try {
            const tasksData = await
fetchWorkspaceTasks(workspaceId);
            setTasks(tasksData || []);
            // Загружаем информацию о пользователях после
получения задач
            await loadTaskAssignees(tasksData);
            // Загружаем комментарии сразу после получения задач
            await loadTaskComments(tasksData);
        } catch (error) {
            console.error('Error loading tasks:', error);
            setTasks([]);
        }

        try {
            const startDate = new Date();
            const endDate = new Date();
            endDate.setDate(endDate.getDate() + 30);

            // Convert dates to UTC for API request

```

```

        const startUTC = new Date(startDate.getTime() +
startDate.getTimezoneOffset() * 60000);
        const endUTC = new Date(endDate.getTime() +
endDate.getTimezoneOffset() * 60000);

        const eventsData = await fetchWorkspaceEvents(
            workspaceId,
            startUTC.toISOString().split('T')[0],
            endUTC.toISOString().split('T')[0]
        );
        setEvents(eventsData || []);
    } catch (error) {
        console.error('Error loading events:', error);
        setEvents([]);
    }

    try {
        const linksData = await
fetchWorkspaceLinks(workspaceId);
        setLinks(linksData || []);
    } catch (error) {
        console.error('Error loading links:', error);
        setLinks([]);
    }

    setError(null);
} catch (error) {
    console.error('Error loading workspace:', error);
    if (error.message.includes('invalid token')) {
        logout();
        navigate('/login');
        return;
    }
    setError('Ошибка при загрузке данных рабочего
пространства');
} finally {
    setLoading(false);
}
};

useEffect(() => {
    loadWorkspaceData();
}, [workspaceId]);

const handleCreateTask = async (taskData) => {
    try {
        await createTask(workspaceId, {
            Title: taskData.Title,
            AssignedTo: taskData.AssignedTo,
            DueDate: taskData.DueDate,
            IsCompleted: taskData.IsCompleted
        });
    }

```

```

        await loadWorkspaceData(); // Перезагружаем все данные
        setShowTaskModal(false);
    } catch (error) {
        console.error('Error creating task:', error);
        setError('Ошибка при создании задачи');
    }
};

const handleUpdateTask = async (taskId, taskData) => {
    try {
        await updateTask(taskId, {
            Title: taskData.Title,
            AssignedTo: taskData.AssignedTo,
            DueDate: taskData.DueDate,
            IsCompleted: taskData.IsCompleted
        });
        await loadWorkspaceData(); // Перезагружаем все данные
        setShowTaskModal(false);
        setSelectedTask(null);
    } catch (error) {
        console.error('Error updating task:', error);
        setError('Ошибка при обновлении задачи');
    }
};

const handleDeleteTask = async (taskId) => {
    if (!window.confirm('Вы уверены, что хотите удалить эту задачу?')) {
        return;
    }

    try {
        await deleteTask(taskId);
        setTasks(prevTasks => prevTasks.filter(task => task.ID !== taskId));
    } catch (error) {
        console.error('Error deleting task:', error);
        setError('Ошибка при удалении задачи');
    }
};

const handleCreateEvent = async (eventData) => {
    try {
        await createEvent(workspaceId, {
            ...eventData,
            date: eventData.date
        });
        await loadWorkspaceData();
        setShowEventModal(false);
    } catch (error) {
        console.error('Error creating event:', error);
        setError('Ошибка при создании события');
    }
};

```

```

    }
  };

  const handleUpdateEvent = async (eventId, eventData) => {
    try {
      // Convert the date to UTC before sending to the server
      const localDate = new Date(eventData.date +
'T00:00:00');
      const utcDate = new Date(localDate.getTime() +
localDate.getTimezoneOffset() * 60000);

      await updateEvent(eventId, {
        ...eventData,
        date: utcDate.toISOString().split('T')[0]
      });
      await loadWorkspaceData();
      setShowEventModal(false);
      setSelectedEvent(null);
    } catch (error) {
      console.error('Error updating event:', error);
      setError('Ошибка при обновлении события');
    }
  };

  const handleDeleteEvent = async (eventId) => {
    if (window.confirm('Вы уверены, что хотите удалить это
событие?')) {
      try {
        await deleteEvent(eventId);
        setEvents(prevEvents => prevEvents.filter(event =>
event.ID !== eventId));
      } catch (error) {
        console.error('Error deleting event:', error);
        if (error.message.includes('invalid token')) {
          logout();
          navigate('/login');
        } else {
          setError('Ошибка при удалении события');
        }
      }
    }
  };

  const handleCreateInvite = async () => {
    try {
      const response = await createInvite(workspaceId);
      setInviteCode(response.invite_code);
      setShowInviteModal(true);
      // Удаляем предыдущий backdrop, если он есть
      const backdrop = document.querySelector('.modal-
backdrop');
      if (backdrop) {

```

```

        backdrop.remove();
    }
} catch (error) {
    console.error('Error creating invite:', error);
    setError('Ошибка при создании приглашения');
}
};

const handleCreateLink = async (e) => {
    e.preventDefault();
    if (!newLinkTitle.trim() || !newLinkUrl.trim()) return;

    try {
        const newLink = await createLink(workspaceId, {
            title: newLinkTitle.trim(),
            url: newLinkUrl.trim()
        });
        setLinks(prevLinks => [...prevLinks, newLink]);
        setNewLinkTitle('');
        setNewLinkUrl('');
    } catch (error) {
        console.error('Error creating link:', error);
        setError('Ошибка при создании ссылки');
    }
};

const handleDeleteLink = async (linkId) => {
    if (!window.confirm('Вы уверены, что хотите удалить эту ссылку?')) {
        return;
    }

    try {
        await deleteLink(linkId);
        setLinks(prevLinks => prevLinks.filter(link => link.ID !== linkId));
    } catch (error) {
        console.error('Error deleting link:', error);
        setError('Ошибка при удалении ссылки');
    }
};

const handleDeleteMember = async (memberId) => {
    if (!window.confirm('Вы уверены, что хотите удалить этого участника?')) {
        return;
    }

    try {
        await removeMember(workspaceId, memberId);
        setMembers(prevMembers => prevMembers.filter(member => member.ID !== memberId));
    }

```

```

    } catch (error) {
      console.error('Error removing member:', error);
      setError('Ошибка при удалении участника');
    }
  };

const handleToggleTaskComplete = async (task) => {
  try {
    await updateTask(task.ID, {
      is_completed: !task.IsCompleted,
      assigned_to: task.AssignedTo
    });

    // После успешного обновления задачи перезагружаем все
данные
    await loadWorkspaceData();
  } catch (error) {
    console.error('Error updating task:', error);
    setError('Ошибка при обновлении задачи');
  }
};

const handleAddComment = async (taskId) => {
  if (!newComment.trim()) return;

  try {
    const response = await addTaskComment(taskId, { text:
newComment });
    const comments = await fetchTaskComments(taskId);
    setTaskComments(prev => ({ ...prev, [taskId]: comments
    }));

    // Загружаем информацию о новом пользователе, если его
нет в списке и если ID определен
    if (response?.UserID && !commentUsers[response.UserID]) {
      try {
        const user = await
fetchWorkspaceMember(workspaceId, response.UserID);
        if (user) {
          setCommentUsers(prev => ({ ...prev, [user.id]:
user }));
        }
      } catch (error) {
        console.error('Error fetching comment user:',
error);
        // Continue execution even if we fail to fetch user
info
      }
    }

    setNewComment('');
  }
};

```

```

        setSelectedTaskForComment(null);
    } catch (error) {
        console.error('Error adding comment:', error);
        setError('Ошибка при добавлении комментария');
    }
};

const renderTaskList = (tasks, isCompleted) => {
    if (!Array.isArray(tasks)) {
        return null;
    }

    return (
        <div className="list-group mb-4">
            {tasks
                .filter(task => task && typeof task === 'object' &&
task.IsCompleted === isCompleted)
                .map(task => {
                    const comments = taskComments[task.ID] || [];
                    const isExpanded = expandedComments[task.ID];
                    const showExpand = comments.length > 1 &&
!isExpanded;
                    const visibleComments = isExpanded ? comments :
comments.slice(0, 1);
                    return (
                        <div key={task.ID} className="list-group-
item">
                            <div className="d-flex justify-content-
between align-items-center mb-1">
                                <h6
                                    className={`mb-1
flex-grow-
1${task.IsCompleted ? ' text-decoration-line-through text-muted'
: ''}`}
                                    style={{ wordBreak: 'break-word' }}
                                >
                                    {task.Title}
                                </h6>
                                <input
                                    type="checkbox"
                                    className="form-check-input ms-2"
                                    checked={task.IsCompleted}
                                    onChange={ () =>
handleToggleTaskComplete(task) }
                                    style={{ width: 22, height: 22 }}
                                />
                                </div>
                                <div className="small text-muted mb-1 text-
start">
                                    {task.DueDate && (
                                        <span>
                                            Дата:
                                            {new
Date(task.DueDate).toLocaleDateString()}

```

```

        </span>
      )}
    </div>
    {task.AssignedTo
taskAssignees[task.AssignedTo] && (
      <div className="d-flex align-items-center
mt-1 text-start">

{taskAssignees[task.AssignedTo].photo_url && (
  <img

src={taskAssignees[task.AssignedTo].photo_url}

alt={taskAssignees[task.AssignedTo].first_name}
  className="rounded-circle me-2"
  style={{ width: '24px', height:
'24px' }}

  />
  )}
  <span className="small">
    Выполняет:
{taskAssignees[task.AssignedTo].first_name}
  </span>
</div>
)}
<div className="d-flex gap-2 mt-3">
  <button
    className="btn btn-sm btn-outline-
danger"
    onClick={ () =>
handleDeleteTask(task.ID) }
  >
    Удалить
  </button>
</div>
{/* Комментарии */}
<div className="mt-3">
  <div className="d-flex gap-2 mb-2">
    <input
      type="text"
      className="form-control form-control-
sm"
      placeholder="Добавить комментарий..."
      value={selectedTaskForComment ===
task.ID ? newComment : ''}
      onChange={ (e) => {
        setSelectedTaskForComment(task.ID);
        setNewComment(e.target.value);
      }}
    />
    <button
      className="btn btn-sm btn-primary"

```

```

                                onClick={() =>
handleAddComment(task.ID) }
                                disabled={selectedTaskForComment !==
task.ID || !newComment.trim()}
                                >
                                    Отправить
                                </button>
                                </div>
                                {visibleComments.map(comment => (
                                    <div key={comment.ID} className="border-
start border-2 ps-2 mb-2 text-start">
                                        <small className="text-muted">

{commentUsers[comment.UserID]?.first_name || 'Пользователь'}:
                                        </small>
                                        <div
                                            className="text-
break">{comment.Text}</div>
                                        </div>
                                    )})
                                {showExpand && (
                                    <button
                                        className="btn btn-link btn-sm px-0"
                                        style={{ fontSize: '13px' }}
                                        onClick={() =>
setExpandedComments(prev => ({ ...prev, [task.ID]: true })))
                                    >
                                        Развернуть
                                        все
                                        комментарии
                                    </button>
                                )}
                                {isExpanded && comments.length > 1 && (
                                    <button
                                        className="btn btn-link btn-sm px-0"
                                        style={{ fontSize: '13px' }}
                                        onClick={() =>
setExpandedComments(prev => ({ ...prev, [task.ID]: false })))
                                    >
                                        Свернуть
                                    </button>
                                )}
                                </div>
                            </div>
                        );
                    }}}
                </div>
            );
        };

const renderTasks = () => (
    <div className="row">
        <div className="col-12 mb-4">

```

```

        <div className="d-flex justify-content-between
align-items-center">
          <h3>Задачи</h3>
          <button
            className="btn btn-primary d-flex align-items-
center"
            onClick={() => {
              setSelectedTask(null);
              setShowTaskModal(true);
            }}
          >
            <Icons.Plus className="me-2" size={20} />
            Создать задачу
          </button>
        </div>
      </div>
      <div className="col-md-6">
        <h5 className="mb-3">Невыполненные задачи</h5>
        {renderTaskList(tasks, false)}
      </div>
      <div className="col-md-6">
        <h5 className="mb-3">Выполненные задачи</h5>
        {renderTaskList(tasks, true)}
      </div>
    </div>
  );

  const renderCalendar = () => {
    const startOfMonth = new Date(currentMonth.getFullYear(),
currentMonth.getMonth(), 1);
    const endOfMonth = new Date(currentMonth.getFullYear(),
currentMonth.getMonth() + 1, 0);

    const days = [];
    const currentDate = new Date(startOfMonth);

    while (currentDate <= endOfMonth) {
      days.push(new Date(currentDate));
      currentDate.setDate(currentDate.getDate() + 1);
    }

    const dayEvents = days.map(day => {
      const dayStr = day.toISOString().split('T')[0];
      return {
        date: day,
        events: events.filter(event => {
          const eventDate = new Date(event.Date);
          eventDate.setDate(eventDate.getDate() - 1); //
Вычитаем один день
          return eventDate.toISOString().split('T')[0] ===
dayStr;
        })
      });
    });
  };

```

```

    };
  });

  const handlePrevMonth = () => {
    setCurrentMonth(new Date(currentMonth.getFullYear(),
currentMonth.getMonth() - 1));
  };

  const handleNextMonth = () => {
    setCurrentMonth(new Date(currentMonth.getFullYear(),
currentMonth.getMonth() + 1));
  };

  return (
    <div className="calendar-container">
      <div className="d-flex justify-content-between
align-items-center mb-4">
        <div className="d-flex align-items-center">
          <h3 className="mb-0 me-4">
            {currentMonth.toLocaleString('ru', { month:
'long', year: 'numeric' })}
          </h3>
          <div className="btn-group">
            <button
              className="btn btn-outline-secondary"
              onClick={handlePrevMonth}
            >
              ←
            </button>
            <button
              className="btn btn-outline-secondary"
              onClick={() => setCurrentMonth(new Date())}
            >
              Сегодня
            </button>
            <button
              className="btn btn-outline-secondary"
              onClick={handleNextMonth}
            >
              →
            </button>
          </div>
        </div>
        <button
          className="btn btn-primary d-flex align-items-
center"
          onClick={() => {
            setSelectedEvent(null);
            setShowEventModal(true);
          }}
        >
          <Icons.Plus className="me-2" size={20} />

```



```

column">
time">{event.Time}</div>
title">{event.Title}</div>
text-danger p-0 ms-2"
<div className="d-flex flex-
  <div className="event-
    <div className="event-
  </div>
  <button
    className="btn btn-sm btn-link
    onClick={ (e) => {
      e.stopPropagation();
      handleDeleteEvent(event.ID);
    }}
    style={{ fontSize: '12px' }}
  >
    <Icons.Delete size={14} />
  </button>
</div>
  )}
</div>
</div>
</div>
  );
  )}
</div>
</div>
</div>
  );
};

const renderLinks = () => (
  <div className="workspace-section">
    <LinkList workspaceId={workspaceId} links={links}
setLinks={setLinks} />
  </div>
);

if (loading) {
  return (
    <div className="container pt-4">
      <div className="d-flex justify-content-center">
        <div className="spinner-border text-primary"
role="status">
          <span className="visually-
hidden">Loading...</span>
        </div>
      </div>
    </div>
  );
}

```

```

    if (error) {
      return (
        <div className="container pt-4">
          <div className="alert alert-danger" role="alert">
            {error}
            <button
              className="btn btn-outline-danger ms-3"
              onClick={() => navigate('/') }
            >
              Вернуться на главную
            </button>
          </div>
        </div>
      );
    }

    return (
      <div className="container pt-4">
        <div className="d-flex justify-content-between align-items-center mb-4">
          <h2>Рабочее пространство</h2>
          <button
            className="btn btn-secondary d-flex align-items-center"
            onClick={() => navigate('/') }
          >
            <Icons.Back className="me-2" size={20} />
            Назад
          </button>
        </div>

        <ul className="nav nav-tabs mb-4">
          <li className="nav-item">
            <button
              className={`nav-link ${activeTab === 'tasks' ? 'active' : ''} d-flex align-items-center`}
              onClick={() => setActiveTab('tasks')}
            >
              <Icons.Tasks className="me-2" size={20} />
              Задачи ({tasks?.length || 0})
            </button>
          </li>
          <li className="nav-item">
            <button
              className={`nav-link ${activeTab === 'calendar' ? 'active' : ''} d-flex align-items-center`}
              onClick={() => setActiveTab('calendar')}
            >
              <Icons.Calendar className="me-2" size={20} />
              Календарь ({events?.length || 0})
            </button>
          </li>
        </ul>
      </div>
    );
  }
}

```

```

    </li>
    <li className="nav-item">
      <button
        className={`nav-link ${activeTab === 'members' ?
'active' : ''} d-flex align-items-center`}
        onClick={() => setActiveTab('members')}
      >
        <Icons.People className="me-2" size={20} />
        Участники ({members?.length || 0})
      </button>
    </li>
    <li className="nav-item">
      <button
        className={`nav-link ${activeTab === 'links' ?
'active' : ''} d-flex align-items-center`}
        onClick={() => setActiveTab('links')}
      >
        <Icons.Link className="me-2" size={20} />
        Ссылки ({links?.length || 0})
      </button>
    </li>
    <li className="nav-item">
      <button
        className={`nav-link ${activeTab ===
'constructor' ? 'active' : ''} d-flex align-items-center`}
        onClick={() => setActiveTab('constructor')}
      >
        <Icons.Constructor className="me-2" size={20} />
        Конструктор
      </button>
    </li>
  </ul>

  {activeTab === 'tasks' && renderTasks()}
  {activeTab === 'calendar' && renderCalendar()}

  {activeTab === 'members' && (
    <div className="members-container">
      <div className="d-flex justify-content-between
align-items-center mb-3">
        <h3>Участники</h3>
        <button
          className="btn btn-primary"
          onClick={handleCreateInvite}
        >
          Пригласить участника
        </button>
      </div>

      {members?.length === 0 ? (
        <div className="alert alert-info">

```

В этом рабочем пространстве пока нет участников.

```

    </div>
  ) : (
    <div className="list-group">
      {members.map(member => (
        <div
          key={`member-${member.ID}`}
className="list-group-item d-flex justify-content-between align-items-center">
          <div
            className="d-flex
            align-items-center">
            {member.photo_url && (
              <img
                src={member.photo_url}
                alt={member.first_name}
                className="rounded-circle me-2"
                style={{ width: '32px', height:
'32px' }}
              />
            )}
            <div>
              <h6
                className="mb-0">{member.first_name}</h6>
              <small>@{member.username}</small>
            </div>
            </div>
            {workspace?.AdminID === user?.id &&
member.ID !== user?.id && (
              <button
                className="btn btn-sm btn-outline-danger"
                onClick={() =>
handleDeleteMember(member.ID)}
              >
                Удалить
              </button>
            )}
            </div>
          )}
        </div>
      )}
    </div>
  )}

{activeTab === 'links' && renderLinks()}
{activeTab === 'constructor' && <Constructor />}

{showTaskModal && (
  <TaskModal
    show={showTaskModal}
    onHide={() => {
      setShowTaskModal(false);
    }}
  >

```

```

        setSelectedTask(null);
    }}
    onSubmit={selectedTask ? handleUpdateTask :
handleCreateTask}
    task={selectedTask}
    members={members}
  />
)}

{showEventModal && (
  <EventModal
    show={showEventModal}
    onHide={() => {
      setShowEventModal(false);
      setSelectedEvent(null);
    }}
    onSubmit={selectedEvent ? handleUpdateEvent :
handleCreateEvent}
    event={selectedEvent}
    members={members}
  />
)}

{showInviteModal && inviteCode && (
  <div className="modal show d-block" tabIndex="-1">
    <div className="modal-dialog">
      <div className="modal-content">
        <div className="modal-header">
          <h5
приглашения</h5>
            className="modal-title">Код

          <button
            type="button"
            className="btn-close"
            onClick={() => {
              setShowInviteModal(false);
              setInviteCode(null);
            }}
          ></button>
        </div>
        <div className="modal-body">
          <p>Поделитесь этим кодом с участником,
которого хотите пригласить:</p>
          <div className="alert alert-info">
            {inviteCode}
          </div>
        </div>
        <div className="modal-footer">
          <button
            type="button"
            className="btn btn-secondary"
            onClick={() => {
              setShowInviteModal(false);

```

```

        setInviteCode(null);
    }}
    >
    Закрыть
  </button>
</div>
</div>
</div>
</div>
  )}
</div>
);
}.
.

```

Листинг

A.22

```

/group_construct/frontend/app/src/utils/api.js
export async function apiFetch(endpoint, options = {}) {
  const token = localStorage.getItem('token');
  const headers = {
    'Content-Type': 'application/json',
    ...(token && { Authorization: `Bearer ${token}` }),
    ...options.headers,
  };

  try {
    const res = await fetch(endpoint, {
      ...options,
      headers,
    });

    if (!res.ok) {
      const errorText = await res.text();
      const error = new Error(errorText || 'Ошибка запроса');
      error.status = res.status;
      throw error;
    }

    const text = await res.text();
    if (!text) {
      return null;
    }
    return JSON.parse(text);
  } catch (error) {
    if (error.status === 401) {
      throw new Error('invalid token');
    }
    throw error;
  }
}

export async function fetchCurrentUser() {
  return apiFetch('/api/users/me');
}

```

```

    }

    export async function telegramAuthCallback(data) {
      return apiFetch('/auth/telegram/callback', {
        method: 'POST',
        body: JSON.stringify(data),
      });
    }

    export async function registerUser() {
      return apiFetch('/api/app/register', {
        method: 'POST'
      });
    }

    export async function fetchWorkspaces() {
      return apiFetch('/api/app/workspaces');
    }

    export async function createWorkspace(data) {
      return apiFetch('/api/app/workspaces', {
        method: 'POST',
        body: JSON.stringify(data),
      });
    }

    export async function deleteWorkspace(workspaceId) {
      return apiFetch(`/api/app/workspaces/${workspaceId}`, {
        method: 'DELETE',
      });
    }

    export async function inviteMember(workspaceId, data) {
      return
      apiFetch(`/api/app/workspaces/${workspaceId}/members`, {
        method: 'POST',
        body: JSON.stringify(data),
      });
    }

    export async function removeMember(workspaceId, memberId) {
      return
      apiFetch(`/api/app/workspaces/${workspaceId}/members/${memberId}
      `, {
        method: 'DELETE',
      });
    }

    export async function fetchWorkspaceMember(workspaceId,
    memberId) {

```

```

        return
    apiFetch(`/api/app/workspaces/${workspaceId}/members/${memberId}
`);
    }

    export async function fetchWorkspaceMembers(workspaceId) {
        return
    apiFetch(`/api/app/workspaces/${workspaceId}/members`);
    }

    export async function createInvite(workspaceId) {
        return
    apiFetch(`/api/app/workspaces/${workspaceId}/invite`, {
        method: 'POST',
    });
    }

    export async function joinByInviteCode(code) {
        return apiFetch(`/api/app/join/${code}`, {
            method: 'POST',
        });
    }

    export async function createTask(workspaceId, data) {
        return
    apiFetch(`/api/app/workspaces/${workspaceId}/tasks`, {
        method: 'POST',
        body: JSON.stringify({
            title: data.Title,
            assigned_to: data.AssignedTo,
            due_date: data.DueDate
        }),
    });
    }

    export async function updateTask(taskId, data) {
        const requestBody = {};
        if (data.hasOwnProperty('Title'))    requestBody.title    =
data.Title;
        if (data.hasOwnProperty('AssignedTo'))
requestBody.assigned_to = data.AssignedTo;
        if (data.hasOwnProperty('DueDate')) requestBody.due_date =
data.DueDate;
        if (data.hasOwnProperty('is_completed'))
requestBody.is_completed = data.is_completed;
        if (data.hasOwnProperty('assigned_to'))
requestBody.assigned_to = data.assigned_to;

        return apiFetch(`/api/app/tasks/${taskId}`, {
            method: 'PUT',
            body: JSON.stringify(requestBody),
        });
    }

```

```

    }

    export async function deleteTask(taskId) {
      return apiFetch(`/api/app/tasks/${taskId}`, {
        method: 'DELETE',
      });
    }

    export async function fetchWorkspaceTasks(workspaceId) {
      return
    apiFetch(`/api/app/workspaces/${workspaceId}/tasks`);
    }

    export async function addTaskComment(taskId, data) {
      return apiFetch(`/api/app/tasks/${taskId}/comments`, {
        method: 'POST',
        body: JSON.stringify(data),
      });
    }

    export async function fetchTaskComments(taskId) {
      return apiFetch(`/api/app/tasks/${taskId}/comments`);
    }

    export async function createEvent(workspaceId, data) {
      return
    apiFetch(`/api/app/workspaces/${workspaceId}/events`, {
        method: 'POST',
        body: JSON.stringify(data),
      });
    }

    export async function updateEvent(eventId, data) {
      return apiFetch(`/api/app/events/${eventId}`, {
        method: 'PUT',
        body: JSON.stringify(data),
      });
    }

    export async function deleteEvent(eventId) {
      return apiFetch(`/api/app/events/${eventId}`, {
        method: 'DELETE',
      });
    }

    export async function fetchWorkspaceEvents(workspaceId,
start, end) {
      return
    apiFetch(`/api/app/workspaces/${workspaceId}/events?start=${start}&end=${end}`);
    }

```

```

    export async function createLink(workspaceId, data) {
      return
    }
    apiFetch(`/api/app/workspaces/${workspaceId}/links`, {
      method: 'POST',
      body: JSON.stringify(data),
    });
  }

  export async function updateLink(linkId, data) {
    return apiFetch(`/api/app/links/${linkId}`, {
      method: 'PUT',
      body: JSON.stringify(data),
    });
  }

  export async function deleteLink(linkId) {
    return apiFetch(`/api/app/links/${linkId}`, {
      method: 'DELETE',
    });
  }

  export async function fetchWorkspaceLinks(workspaceId) {
    return
  }
  apiFetch(`/api/app/workspaces/${workspaceId}/links`);
}

export async function getCurrentUserId() {
  const res = await apiFetch('/api/app/register', {
    method: 'POST',
  });
  return res?.id;
}

// API client object that wraps the individual functions
export const api = {
  get: async (endpoint) => {
    return apiFetch(endpoint);
  },
  post: async (endpoint, data) => {
    return apiFetch(endpoint, {
      method: 'POST',
      body: JSON.stringify(data),
    });
  },
  put: async (endpoint, data) => {
    return apiFetch(endpoint, {
      method: 'PUT',
      body: JSON.stringify(data),
    });
  },
  delete: async (endpoint) => {
    return apiFetch(endpoint, {

```

```

        method: 'DELETE',
      });
    },
  };
};

.
ЛИСТИНГ A.23
/group_construct/frontend/app/src/utils/icons.jsx
import React from 'react';

export const Icons = {
  Plus: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path d="M8 4a.5.5 0 0 1 .5.5v3h3a.5.5 0 0 1 0 1h-
3v3a.5.5 0 0 1-1 0v-3h-3a.5.5 0 0 1 0-1h3v-3A.5.5 0 0 1 8 4z"/>
    </svg>
  ),

  People: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path d="M15 14s1 0 1-1-1-4-5-4-5 3-5 4 1 1 1 1h8Zm-
7.978-1A.261.261 0 0 1 7 12.996c.001-.264.167-1.03.76-1.72C8.312
10.629 9.282 10 11 10c1.717 0 2.687.63 3.24 1.276.593.69.758
1.457.76 1.721-.008.002a.274.274 0 0 1-.014.002H7.022ZM11 7a2 2 0
1 0 0-4 2 2 0 0 0 0 4Zm3-2a3 3 0 1 1-6 0 3 3 0 0 1 6 0ZM6.936
9.28a5.88 5.88 0 0 0-1.23-.247A7.35 7.35 0 0 0 5 9c-4 0-5 3-5 4 0
.667.333 1 1 1h4.216A2.238 2.238 0 0 1 5 13c0-1.01.377-2.042 1.09-
2.904.243-.294.526-.569.846-.816ZM4.92 10A5.493 5.493 0 0 0 4
13H1c0-.26.164-1.03.76-1.724.545-.636 1.492-1.256 3.16-1.275ZM1.5
5.5a3 3 0 1 1 6 0 3 3 0 0 1-6 0Zm3-2a2 2 0 1 0 0 4 2 2 0 0 0 0-
4Z"/>
    </svg>
  ),

  Tasks: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path d="M14.5 3a.5.5 0 0 1 .5.5v9a.5.5 0 0 1-.5.5h-
13a.5.5 0 0 1-.5-.5v-9a.5.5 0 0 1 .5-.5h13zm-13-1A1.5 1.5 0 0 0 0
3.5v9A1.5 1.5 0 0 0 1.5 14h13a1.5 1.5 0 0 0 1.5-1.5v-9A1.5 1.5 0
0 0 14.5 2h-13z"/>
      <path d="M7 5.5a.5.5 0 0 1 .5-.5h5a.5.5 0 0 1 0 1h-
5a.5.5 0 0 1-.5-.5zm-1.496-.854a.5.5 0 0 1 .707 0l1.414 1.414a.5.5
0 1 1-.707.707L5.5 5.5V7a.5.5 0 0 1-1 0V5.5l-.793.793a.5.5 0 1 1-
.707-.707l1.414-1.414a.5.5 0 0 1 0-.707zM7 9.5a.5.5 0 0 1 .5-
.5h5a.5.5 0 0 1 0 1h-5a.5.5 0 0 1-.5-.5zm-1.496-.854a.5.5 0 0 1
.707 0l1.414 1.414a.5.5 0 1 1-.707.707L5.5 9.5V11a.5.5 0 0 1-1
0V9.5l-.793.793a.5.5 0 1 1-.707-.707l1.414-1.414z"/>

```

```

    </svg>
  ),

  Calendar: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path d="M3.5 0a.5.5 0 0 1 .5.5V1h8V.5a.5.5 0 0 1 1
0V1h1a2 2 0 0 1 2 2v11a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2V3a2 2 0 0 1
2-2h1V.5a.5.5 0 0 1 .5-.5zM1 4v10a1 1 0 0 0 1 1h12a1 1 0 0 0 1-
1V4H1z"/>
    </svg>
  ),

  Edit: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path d="M12.854.146a.5.5 0 0 0-.707 0L10.5 1.793
14.207 5.511.647-1.646a.5.5 0 0 0 0-.708l-3-3zm.646 6.061L9.793
2.5 3.293 9H3.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.5h.5a.5.5
0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.207l6.5-6.5zm-7.468 7.468A.5.5
0 0 1 6 13.5V13h-.5a.5.5 0 0 1-.5-.5V12h-.5a.5.5 0 0 1-.5-.5V11h-
.5a.5.5 0 0 1-.5-.5V10h-.5a.499.499 0 0 1-.175-.032l-.179.178a.5.5
0 0 0-.11.168l-2 5a.5.5 0 0 0 .65.65l5-2a.5.5 0 0 0 .168-.111.178-
.178z"/>
    </svg>
  ),

  Delete: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5
0 0 1 .5-.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1
.5-.5zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0 0 1 0V6z"/>
      <path fillRule="evenodd" d="M14.5 3a1 1 0 0 1-1 1H13v9a2
2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h-.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-
1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1 1v1zM4.118 4 4
4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882 4H4.118zM2.5
3V2h11v1h-11z"/>
    </svg>
  ),

  Back: ({ size = 24, className = '' }) => (
    <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
      <path fillRule="evenodd" d="M15 8a.5.5 0 0 0-.5-
.5H2.707l3.147-3.146a.5.5 0 1 0-.708-.708l-4 4a.5.5 0 0 0 0 .708l4
4a.5.5 0 0 0 .708-.708L2.707 8.5H14.5A.5.5 0 0 0 15 8z"/>
    </svg>
  )

```

```

),

Close: ({ size = 24, className = '' }) => (
  <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
    <path d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293l2.646-
2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8
8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5
0 0 1 0-.708z"/>
  </svg>
),

Link: ({ size = 24, className = '' }) => (
  <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
    <path d="M6.354 5.5H4a3 3 0 0 0 0 6h3a3 3 0 0 0 2.83-
4H9c-.086 0-.17.01-.25.031A2 2 0 0 1 7 10.5H4a2 2 0 1 1 0-
4h1.535c.218-.376.495-.714.82-1z"/>
    <path d="M9 5.5a3 3 0 0 0 0-2.83 4h1.098A2 2 0 0 1 9
6.5h3a2 2 0 1 1 0 4h-1.535a4.02 4.02 0 0 1-.82 1H12a3 3 0 1 0 0-
6H9z"/>
  </svg>
),

Constructor: ({ size = 24, className = '' }) => (
  <svg xmlns="http://www.w3.org/2000/svg" width={size}
height={size} fill="currentColor" className={className}
viewBox="0 0 16 16">
    <path d="M1.5 0A1.5 1.5 0 0 0 0 1.5v7A1.5 1.5 0 0 0 1.5
10H6v1H1a1 1 0 0 0-1 1v3a1 1 0 0 0 1 1h14a1 1 0 0 0 1-1v-3a1 1 0
0 0-1-1h-5v-1h4.5A1.5 1.5 0 0 0 16 8.5v-7A1.5 1.5 0 0 0 14.5 0h-
13Zm0 1h13a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-.5.5h-13a.5.5 0 0 1-.5-
.5v-7a.5.5 0 0 1 .5-.5ZM12 12.5a.5.5 0 1 1 0 .5.5 0 0 1-1 0Zm2
0a.5.5 0 1 1 1 0 .5.5 0 0 1-1 0ZM1.5 12h5a.5.5 0 0 1 0 1h-5a.5.5
0 0 1 0-1ZM1 14.25a.25.25 0 0 1 .25-.25h5.5a.25.25 0 1 1 0 .5h-
5.5a.25.25 0 0 1-.25-.25Z"/>
  </svg>
)
};
.

```

**Приложение В**  
**(обязательное)**  
**Результаты тестирования**

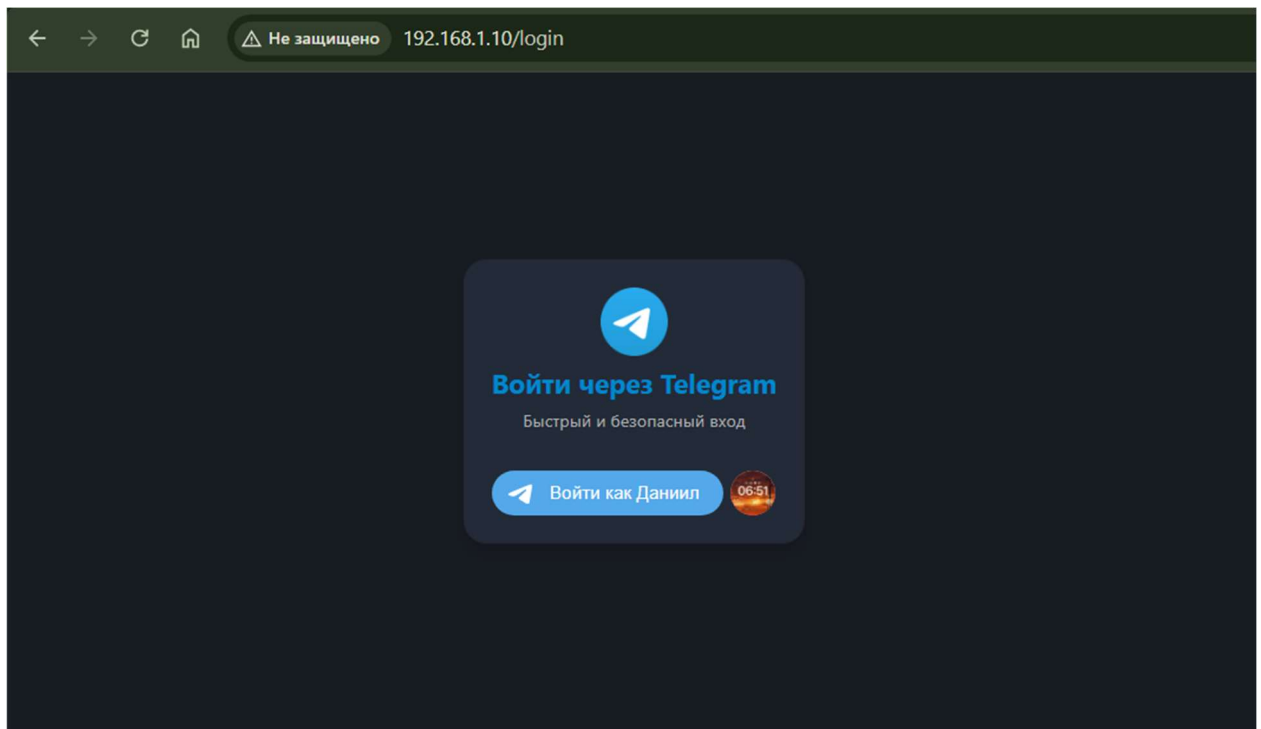


Рисунок Г.1 – Тестирование авторизации через Telegram

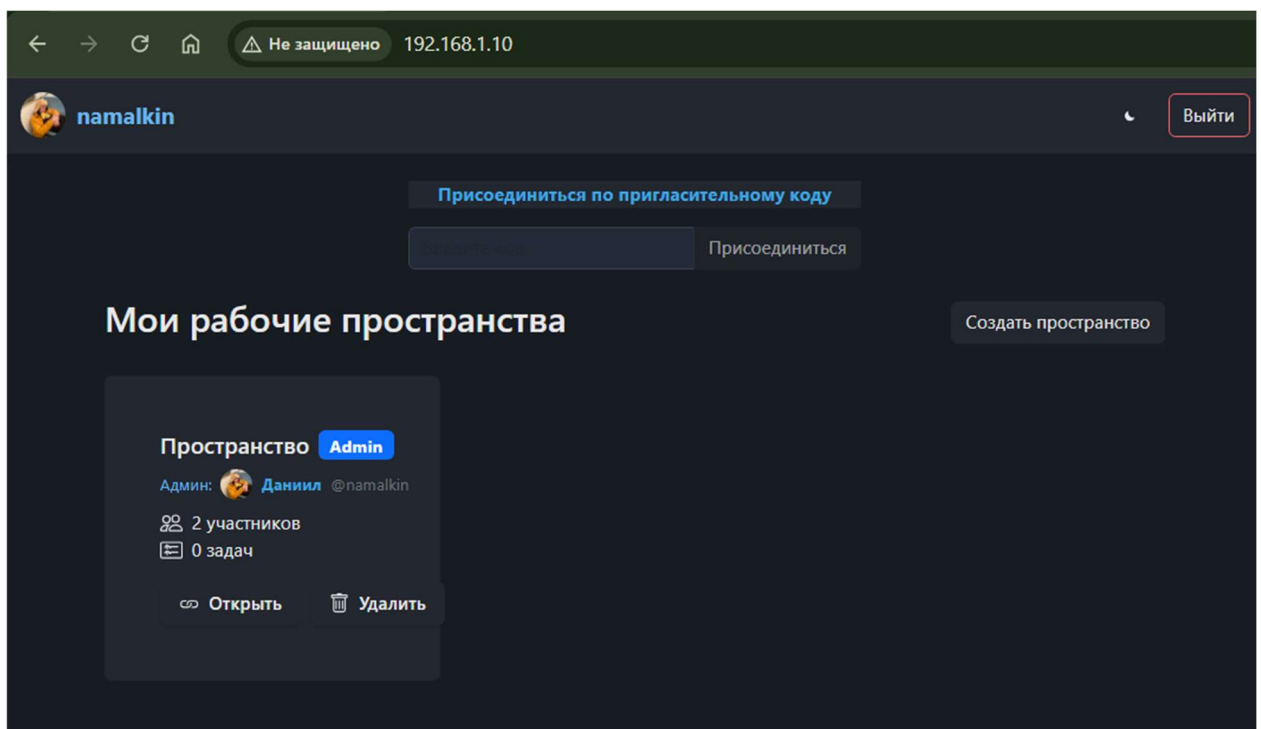


Рисунок Г.2 – Тестирование главного меню

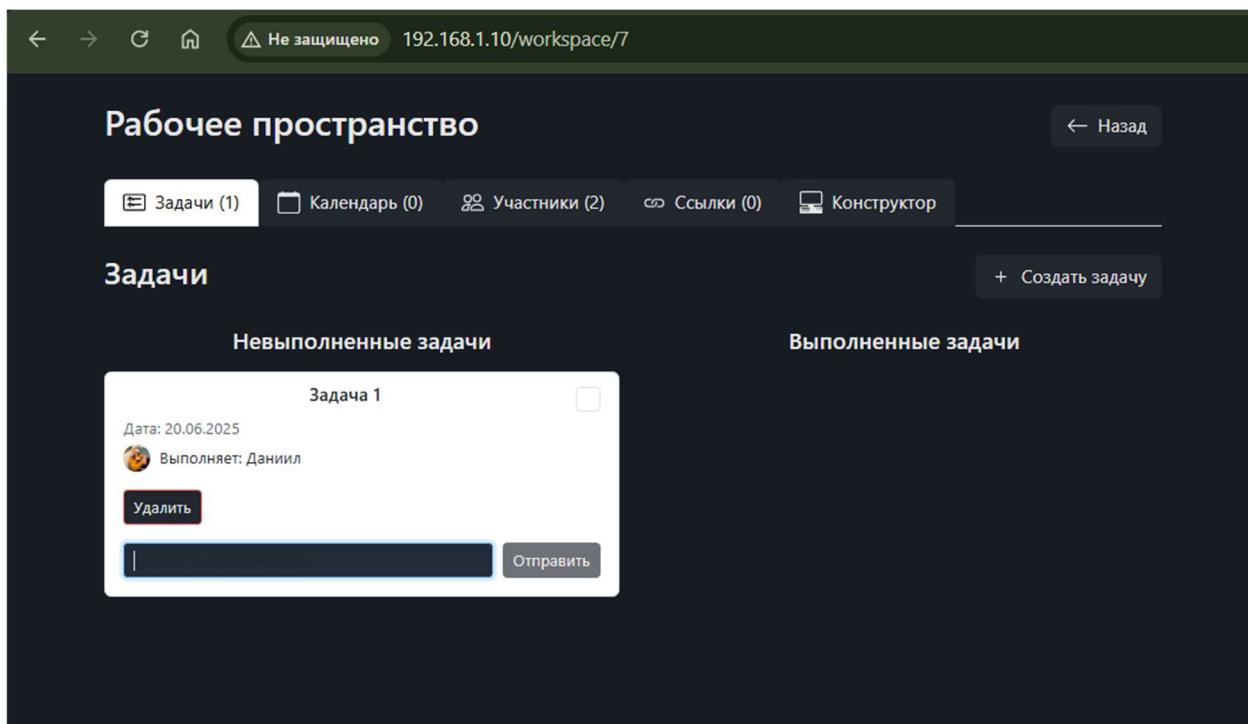


Рисунок Г.3 – Тестирование входа в рабочее пространство

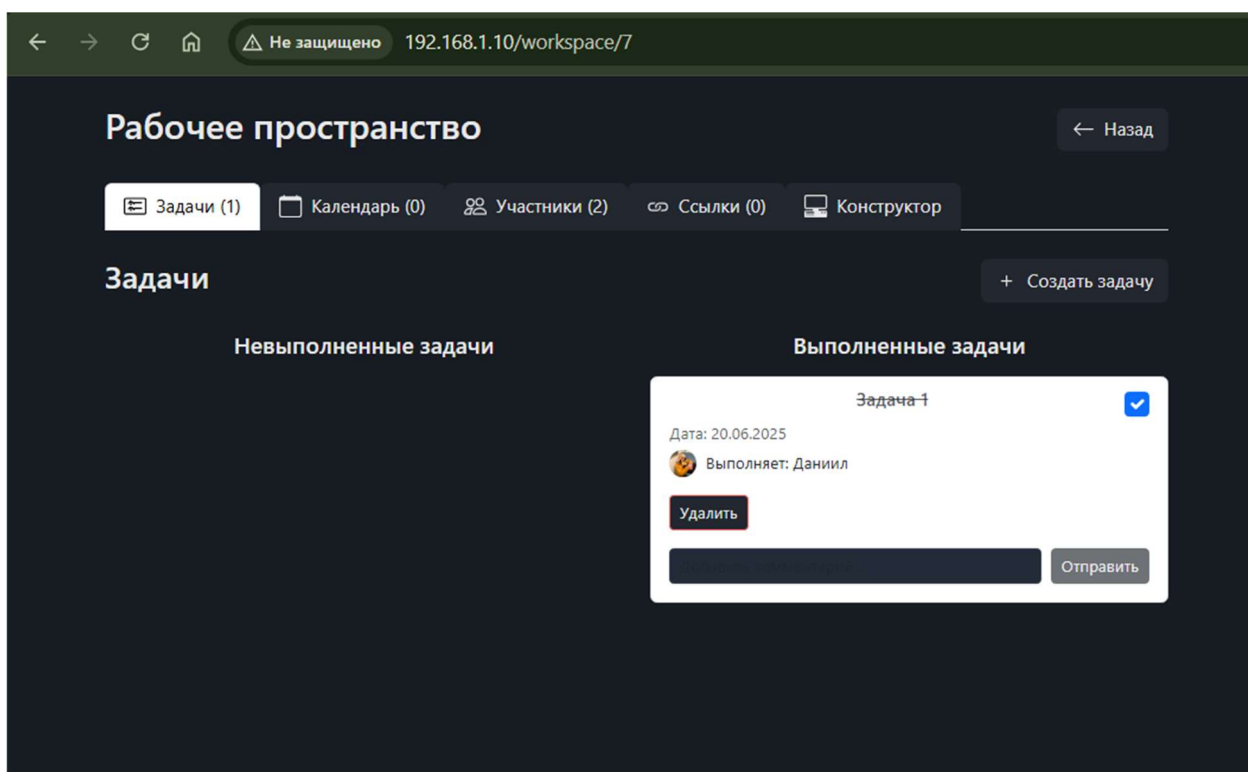


Рисунок Г.4 – Тестирование выполнение задачи

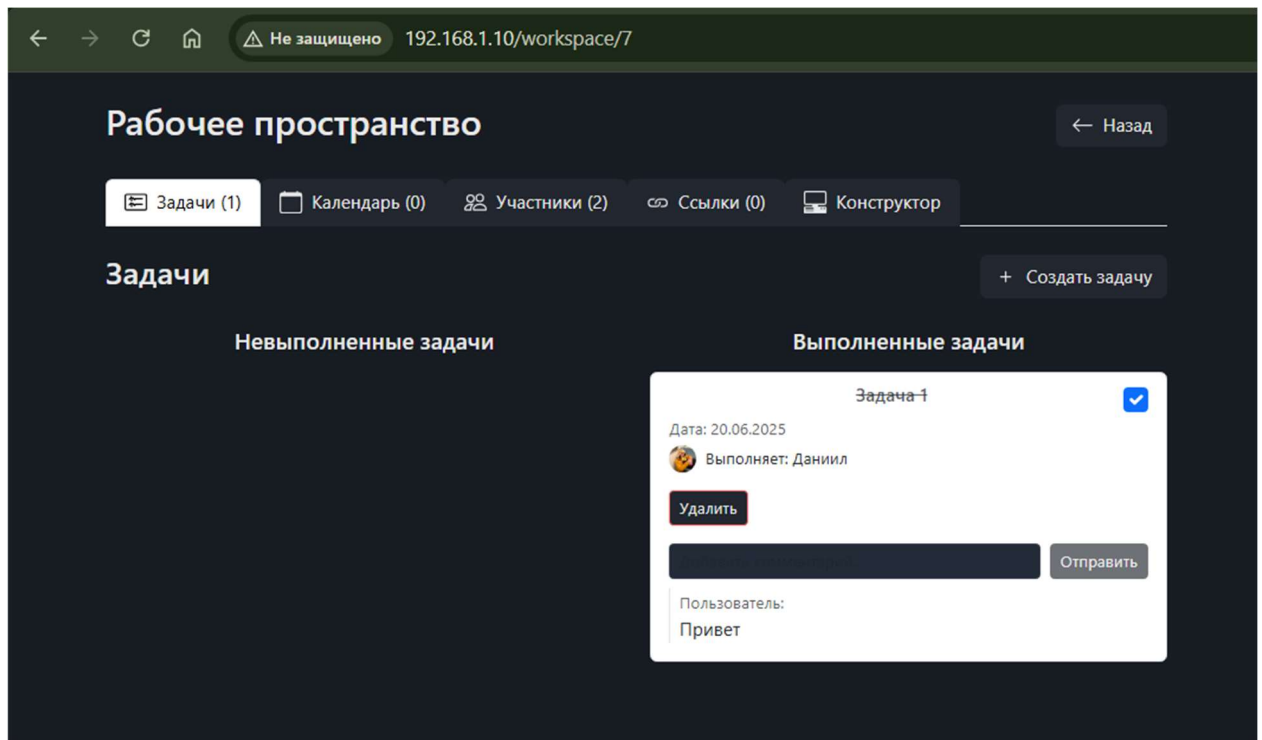


Рисунок Г.5 – Тестирование написания комментария

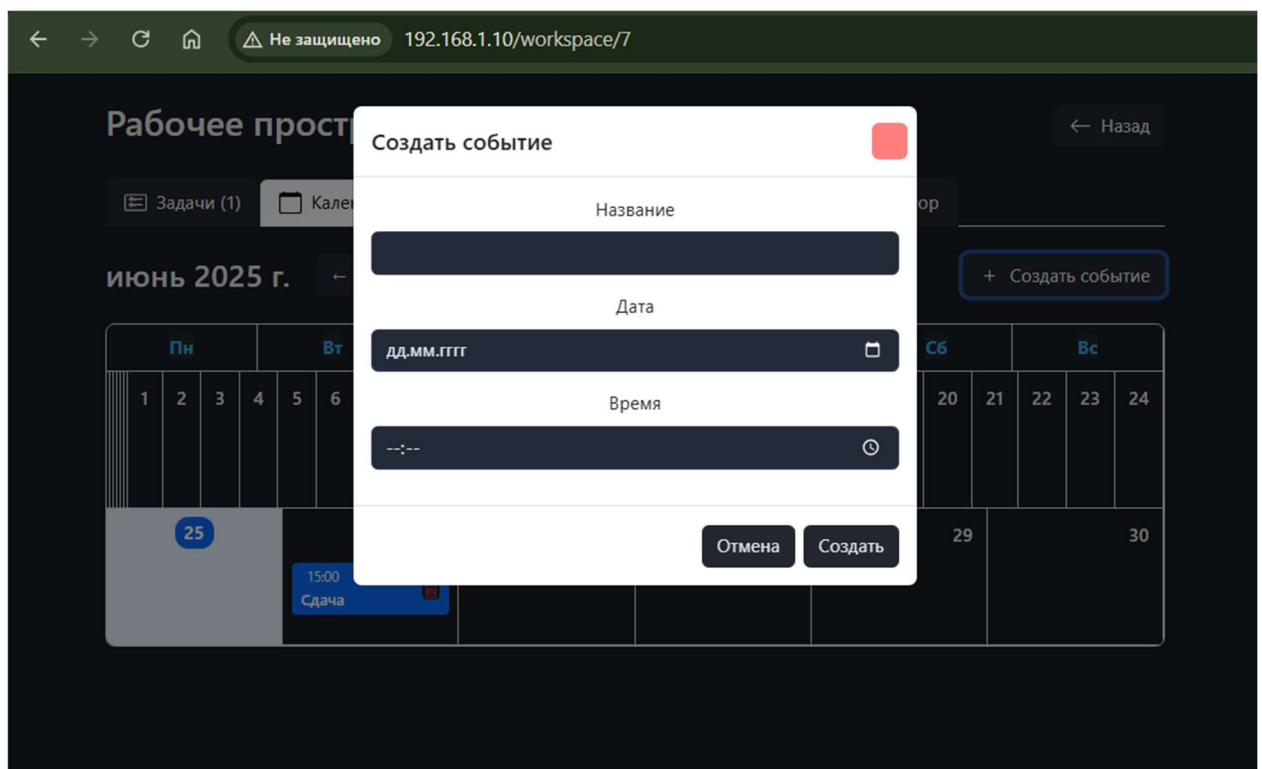


Рисунок Г.6 – Тестирование добавления события

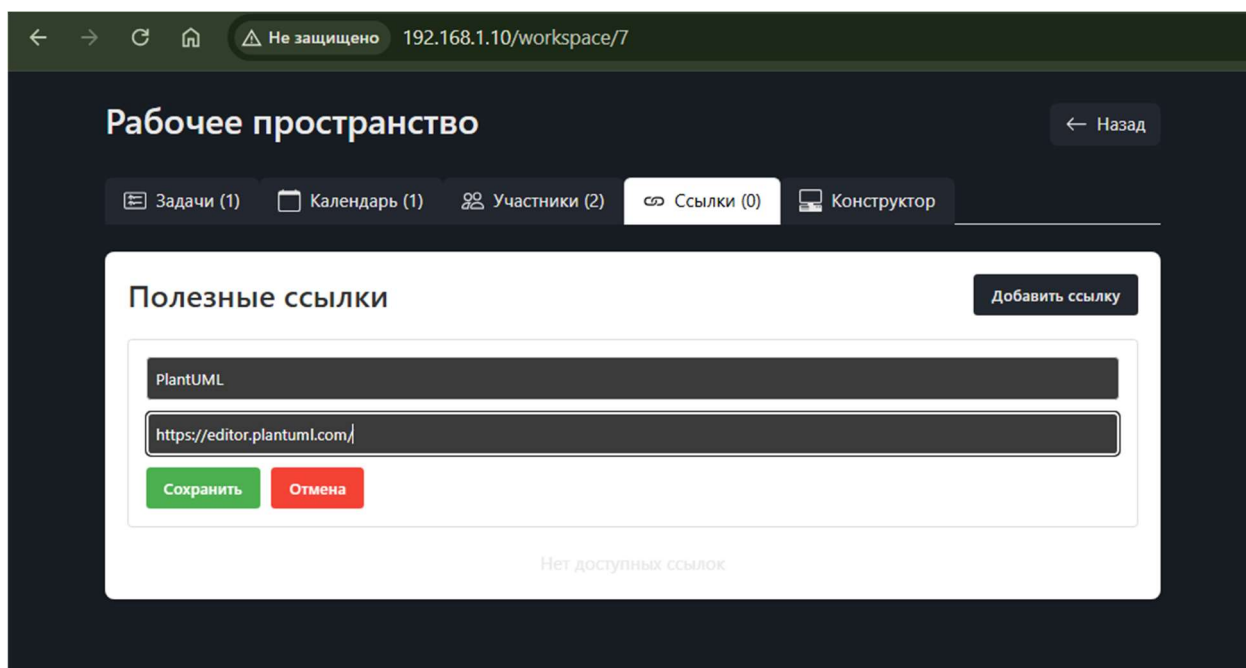


Рисунок Г.7 – Тестирование добавления ссылки

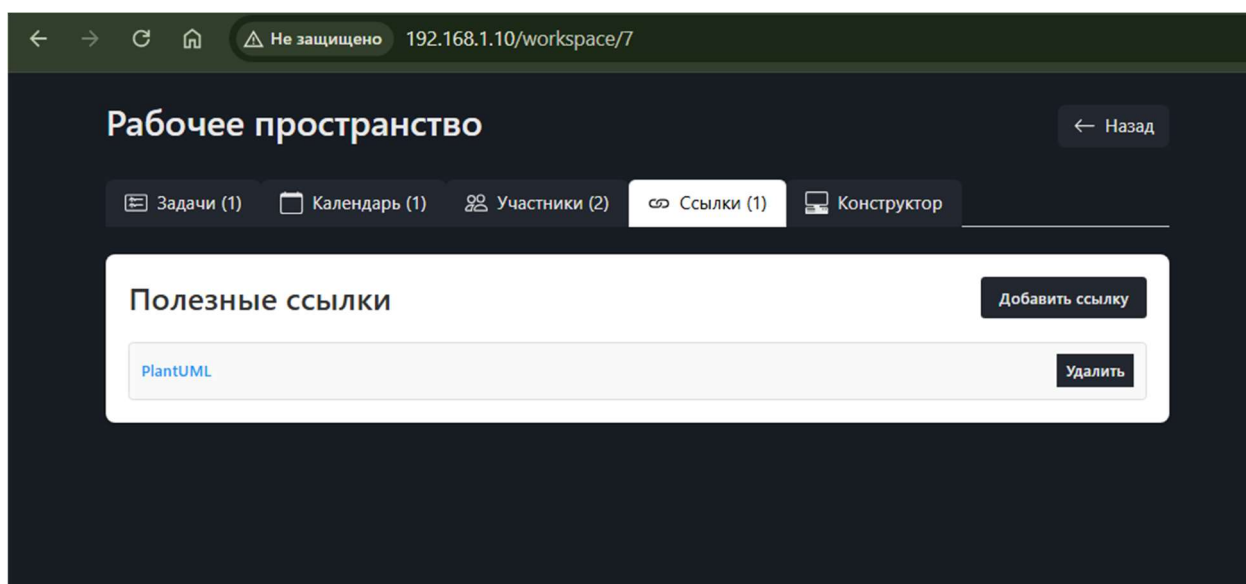


Рисунок Г.8 – Тестирование списка ссылок

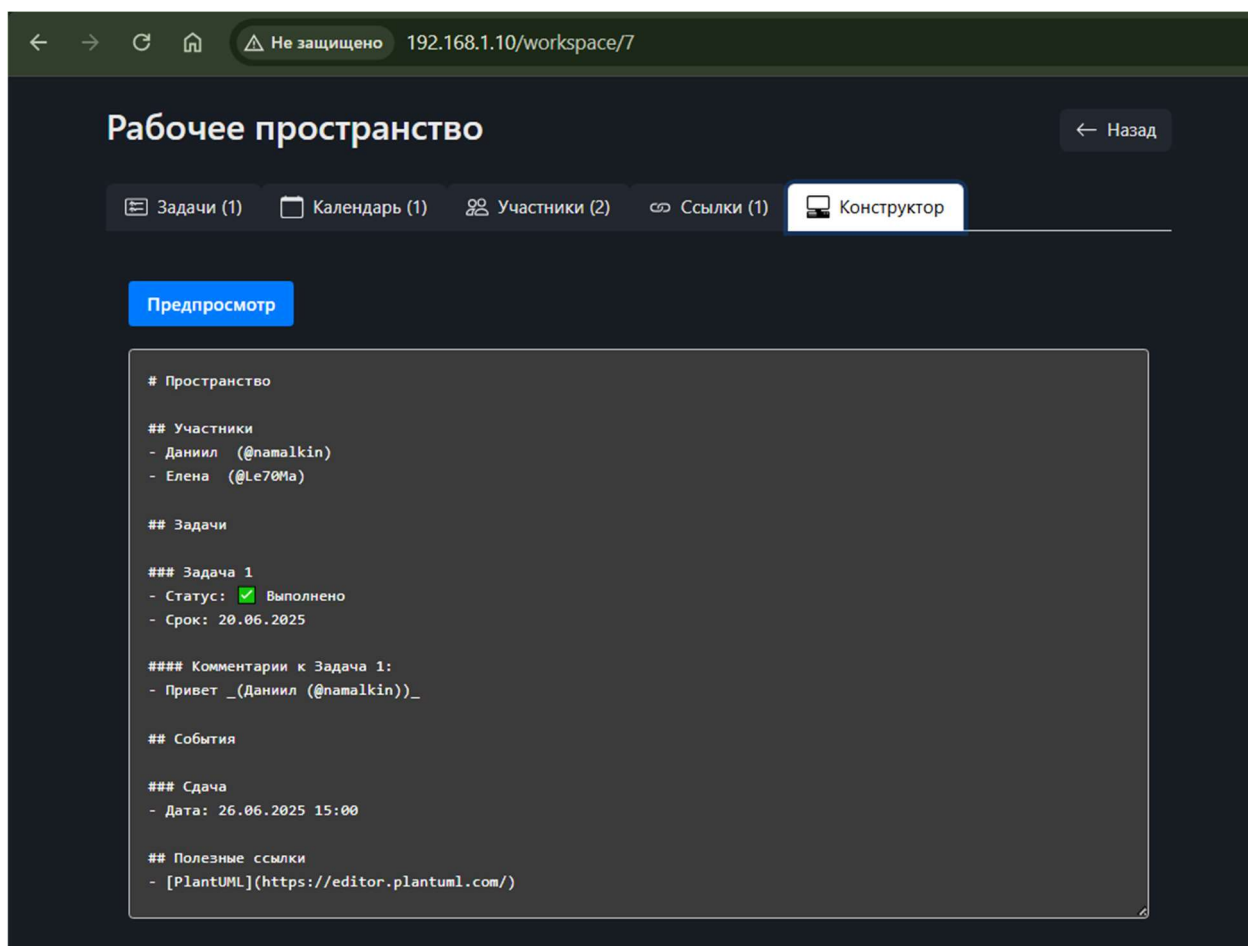


Рисунок Г.9 – Тестирование редактирования конструктора

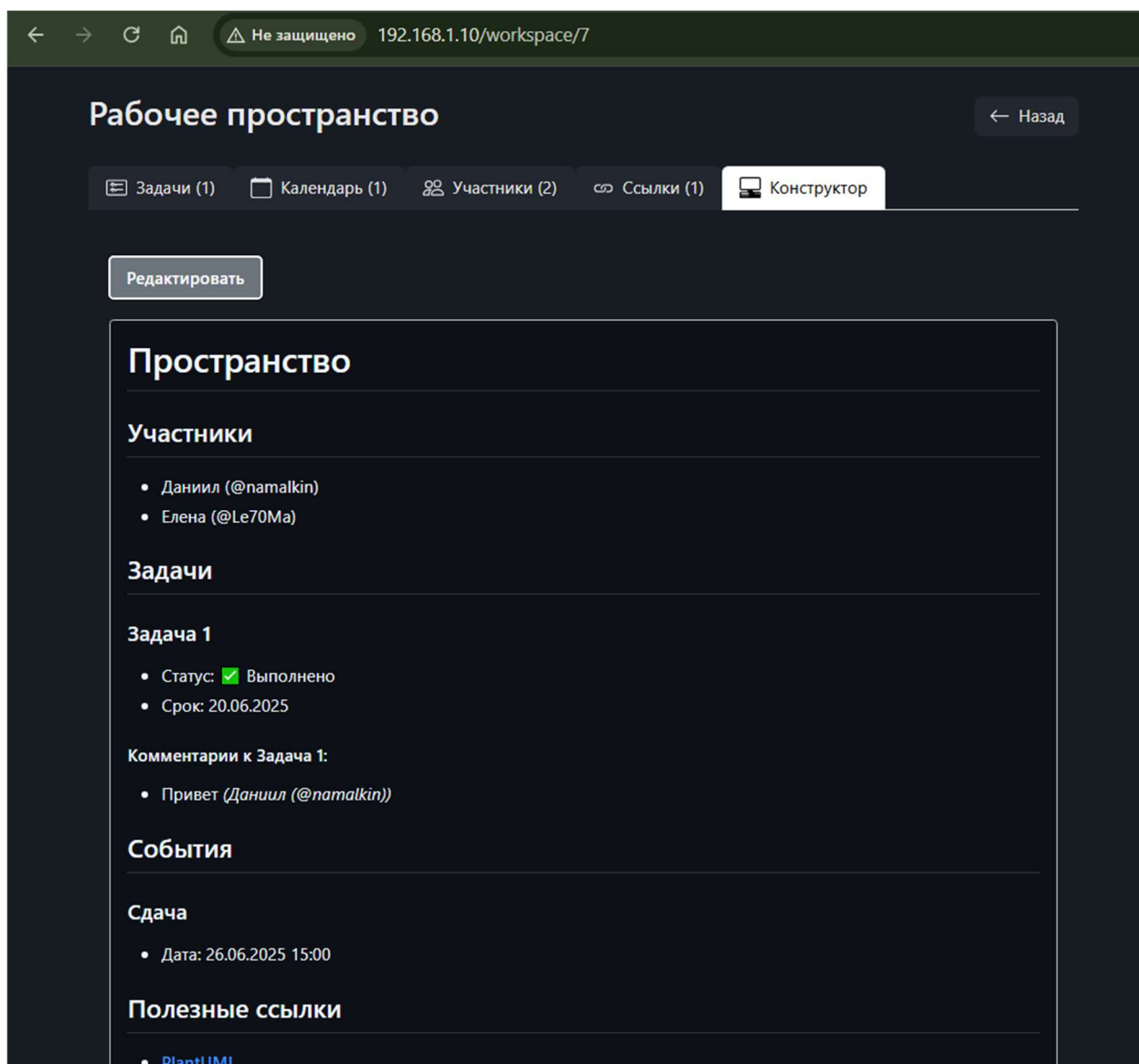


Рисунок Г.10 – Тестирование предпросмотра конструктора