МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Программные средства для осознанного планирования жизни

Студент _____ Дюжева Анна Николаевна
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И.Ю.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ 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, 2024

Федеральное государственное бюджетное образовательное учреждение высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«25» декабря 2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

БАКАЛАВРА

1. Студент Дюжева Анна Николаевна гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы Программные средства для осознанного планирования жизни

Тема утверждена приказом ПГУ №_____от "_____"_____20_____

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

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

1) Назначение: программные средства для осознанного планирования жизни с возможностью поиска и визуализации табличных данных.

2) Среда выполнения: Windows 10.

Языковые средства разработки:,JavaScript, C#, SQL, UML.

Инструментальные средства: Visual Studio Code, JetBrains Rider, Swagger, npm, Docker, PostgreSQL, Draw.io, Visual Paradigm 17.0.

Технологии разработки ООП, микросервисная архитектура.

3) Основные функции приложения:

а) регистрация и авторизация;

б) настройка системы в соответствии с предпочтениями пользования;

в) хранение и систематизация информации о деятельности пользователя;

г) отслеживание времени, затраченного на выполнение задач;

д) анализ продуктивности деятельности пользователя;

е) визуализация аналитических данных для лучшего понимания результатов.

5. Перечень подлежащих разработке вопросов:

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

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 154 листа, 24 рисунка, 11 таблиц, 20 использованных источников и 8 приложений.

ОСОЗНАННОЕ ПЛАНИРОВАНИЕ ЖИЗНИ, WEB-ПРИЛОЖЕНИЕ, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, REST, UML, POSTGRESQL, C#, JS, БАЗА ДАННЫХ, REACTJS.

Объект исследования – программные средства для осознанного планирования жизни.

Цель выпускной квалификационной работы – разработать программные средства для осознанного планирования жизни с возможностью анализа эффективности деятельности человека.

Технологии разработки – разработка клиентской части приложения проводилась на языке программирования JavaScript с использованием библиотеки ReactJS в среде разработки Visual Studio Code. Разработка серверной части приложения производилась на языке программирования C# в среде разработки JetBrains Rider. Разработка базы данных проводилась в СУБД PostgreSQL.

Среда исполнения – Windows 10, web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть программных средств для осознанного планирования жизни. Все требования к разработке приложения соблюдены.

Содержание

Введение 7

1 Анализ программных средств для осознанного планирования жизни 8

1.1 Исследование предметной области 8

1.2 Обзор и анализ программных средств для осознанного планирования жизни 8

1.2.1 Web-приложение «Todoist» 8

1.2.2 Web-приложение «TickTick» 9

1.2.3 Web-приложение «Google Calendar» 10

1.2.4 Web-приложение «Trello» 11

1.2.5 Сравнение программных средств для осознанного планирования жизни 12

1.3 Анализ функциональных требований 14

1.4 Анализ нефункциональных требований 17

2 Проектирование программных средств для осознанного планирования жизни 19

2.1 Архитектура приложения 19

2.2 Проектирование структур данных 20

2.2.1 Концептуальная модель базы данных 20

2.2.2 Логические модели баз данных 22

2.3 Проектирование пользовательского интерфейса 22

2.4 Проектирование классов программных средств для осознанного планирования жизни 23

3 Реализация программных средств для осознанного планирования жизни 25

3.1 Реализация базы данных 25

3.2 Реализация серверной части приложения 25
3.3 Реализация клиентской части приложения 27
3.4 Реализация интерфейса приложения 29
4 Контроль качества программного обеспечения 39
4.1 Обоснование выбора режима тестирования и метрик кода 39
4.2 Тестирование программных средств для осознанного планирования жизни 40
4.2.1 Функциональное тестирование 40
4.2.2 Модульное тестирование 42
4.3 Оценка качества программного кода 44
4.3.1 Метрика Холстеда 44
4.3.2 Метрика Чепина 46
4.4 Планирование разработки и оценка бюджета 47
Заключение 49
Список использованных источников 51
Приложение А (справочное) Глоссарий 53
Приложение Б (рекомендуемое) Атрибуты сущностей базы данных 56
Приложение В (рекомендуемое) Графическая часть 65
Приложение Г (рекомендуемое) Листинг серверной части приложения 73
Приложение Д (рекомендуемое) Листинг клиентской части приложения 92
Приложение Е (рекомендуемое) Результаты функционального тестирования 139
Приложение Ж (рекомендуемое) Листинг модульных тестов 147
Приложение И (рекомендуемое) Результаты модульного тестирования 152

Введение

Грамотное планирование жизни является одной из главных составляющих успешного выполнения любой деятельности, оно закладывает прочные основы для достижения поставленных целей вне зависимости от сферы деятельности [1].

В настоящее время люди стараются правильно распоряжаться таким ценным ресурсом как время, а также пытаются повысить свою продуктивность. Это возможно лишь в условиях постоянного контроля своей деятельности.

Разработка программных средств для осознанного планирования жизни является актуальной темой на сегодняшний день, так как такие средства помогают автоматизировать сбор и анализ данных о ежедневной занятости человека и позволяют наглядно увидеть недочеты распределения времени по направлениям сфер жизни.

Целью выпускной квалификационной работы является разработка программных средств для осознанного планирования жизни, которые позволят человеку анализировать эффективность своей деятельности, путем сравнения затраченного времени по направлениям жизни с запланированным, а также контролировать достижение различных целей той или иной деятельности.

Исходя из цели данной работы, были поставлены следующие задачи:

– проанализировать предметную область и аналоги похожего программного обеспечения, поставить задачу на разработку;

– спроектировать программные средства, а также структуры данных разрабатываемого приложения;

– разработать модули программных средств;

– проконтролировать качество программного обеспечения.

Успешное решение поставленных задач позволит создать эффективный инструмент для осознанного планирования жизни.

1 Анализ программных средств для осознанного планирования жизни

1.1 Исследование предметной области

Программные средства для осознанного планирования – это инструменты, которые дают возможность пользователю фиксировать и достигать цели своей деятельности, проводить анализ эффективности использования временных ресурсов, а также структурировать информацию о всех задачах, запланированных пользователем в одном месте [2].

Деятельность человека многообразна, поэтому удобнее всего рассматривать ее в контексте разных сфер жизни и направлений. Сфера жизни – это область деятельности, а направление – конкретный вид деятельности в рамках сферы. Например, в сферу личностного роста могут входить такие направления, как обучение в университете, прохождение курсов для повышения квалификации и многое другое. В рамках данной работы будет использоваться именно такое структурное разделение деятельности человека.

Программные средства для осознанного планирования жизни разрабатываются для различных категорий пользователей, таких как студенты, работающие или же просто для тех, кто хочет максимизировать свою продуктивность [3].

Обязательным условием использования таких приложений является регистрация пользователя в системе для сохранения и систематизации информации о его деятельности.

1.2 Обзор и анализ программных средств для осознанного планирования жизни

1.2.1 Web-приложение «Todoist»

Приложение «Todoist» является типичным примером приложения для планирования. Интерфейс приложения приведен на рисунке 1.

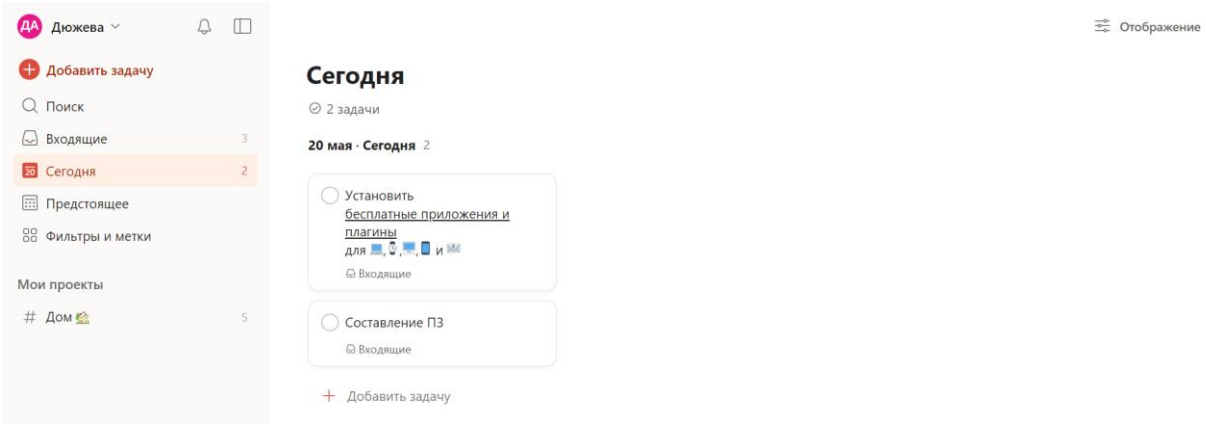

Рисунок 1 – Интерфейс приложения «Todoist»

Среди достоинств данного приложения можно выделить то, что оно предоставляет пользователю простой и интуитивно понятный интерфейс, возможность создавать проекты и подзадачи, прикреплять файлы, а также устанавливать сроки и напоминания для задач.

Главным недостатком, с точки зрения пользователя, можно считать невозможность полного отслеживания времени выполнения задач, получения отчетов о продуктивности пользователя, а также разделения всех задач по сферам жизни.

1.2.2 Web-приложение «TickTick»

Приложение «TickTick» – это популярный инструмент для планирования задач, предлагающий широкий набор функций для управления своим временем. Интерфейс приложения представлен на рисунке 2.

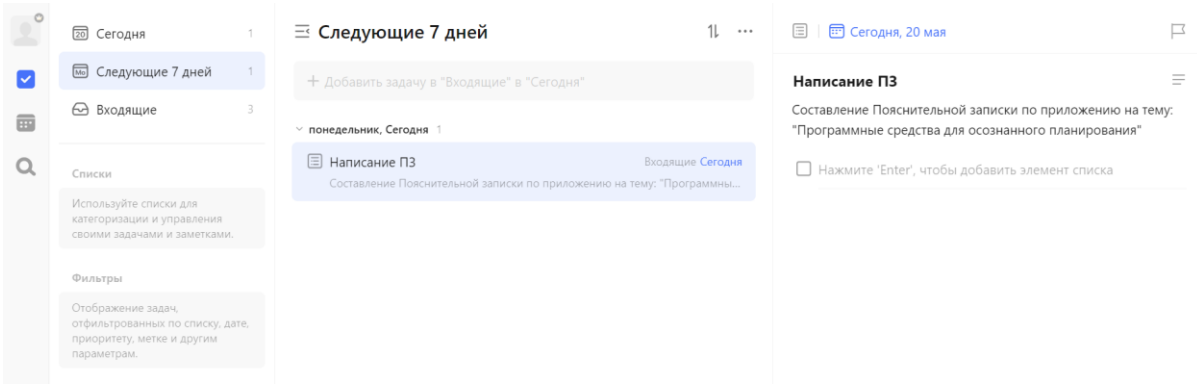

Рисунок 2 – Интерфейс приложения «TickTick»

Среди достоинств этого приложения можно выделить то, что оно предоставляет пользователю возможность разделения задач по спискам, которые пользователь сам создает. «TickTick» предоставляет возможность устанавливать таймеры для задач и отслеживать время, затраченное на их выполнение, а также создавать подзадачи и цели. Приложение предлагает пользователю представление задач в виде календаря и анализ затраченного времени по ним за определенный период.

Недостатком этого приложения является отсутствие возможности планирования времени, которое пользователь хотел бы потратить на то или иное направление жизни и добавить его в анализ для сравнения с фактически затраченным. Также «TickTick» не предоставляет гибкой системы для классификации задач по сферам жизни, проектам и направлениям.

1.2.3 Web-приложение «Google Calendar»

Классическим инструментом для планирования является приложение «Google Calendar», интерфейс которого представлен на рисунке 3.

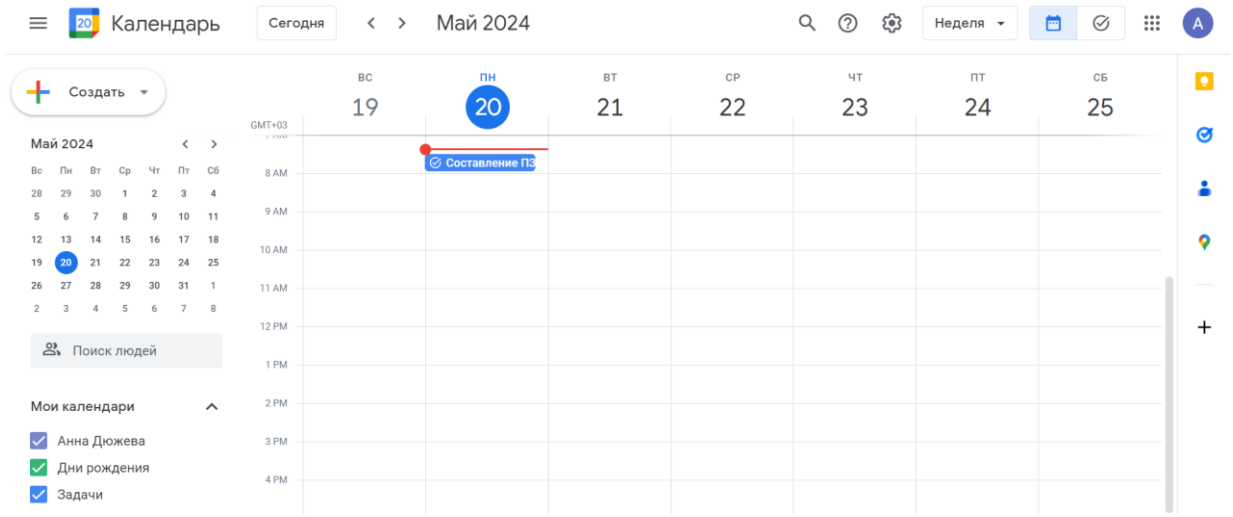

Рисунок 3 – Интерфейс приложения «Google Calendar»

Главным достоинством приложения является интеграция с экосистемой Google, разнообразие уведомлений и напоминаний, возможность добавления событий, синхронизация между устройствами, удобный и наглядный интерфейс.

Основными недостатками является то, что нет возможности отслеживать время выполнения задач, соотносить их с определенным направлением, проектом или же целью, а также нет возможности проводить анализ продуктивности пользователя.

1.2.4 Web-приложение «Trello»

«Trello» – это бесплатное приложение, которое использует визуальный подход к управлению проектами и задачами, предоставляет пользователю удобный интерфейс. Данное приложение популярно среди команд разработчиков, а также среди частных лиц, которые ищут простой и эффективный способ организовать свои проекты и задачи. Интерфейс приложения «Trello» представлен на рисунке 4.

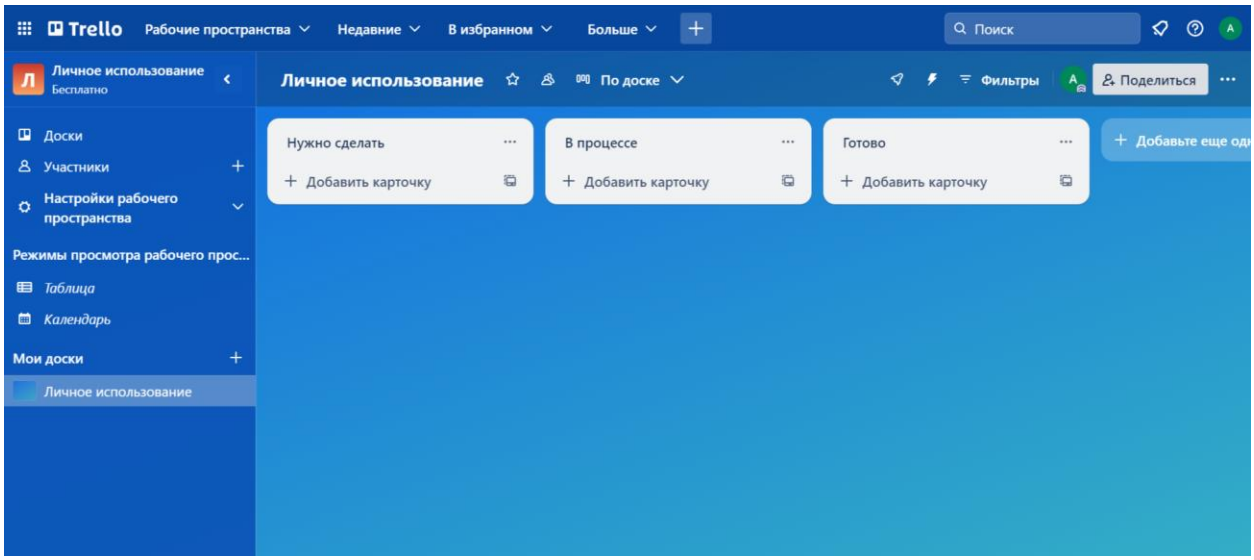

Рисунок 4 – Интерфейс приложения «Trello»

«Trello» позволяет создавать доски с этапами реализации задач, делиться ими с другими пользователями и настраивать визуальную составляющую приложения.

К недостаткам можно отнести отсутствие встроенного функционала для отслеживания времени по задачам, а также то, что данное приложение предназначено прежде всего для управления проектами в команде и может быть неудобным в личном использовании.

1.2.5 Сравнение программных средств для осознанного планирования жизни

Проведем сравнительный анализ рассмотренных выше программных средств, таких как «Todoist», «TickTick», «Google Calendar», «Trello», по ключевым функциям.

Для оценки приложений будет использоваться трехбалльная шкала соответствия. В таблице 1 представлены характеристики оценки.

Таблица 1 – Шкала баллов

Балл	Характеристика оценки
0	функция (качество) полностью отсутствует
1	существует частичная реализация функции (качества)
2	функция (качество) полностью реализована

Результаты сравнения программных средств для осознанного планирования жизни представлены в таблице 2.

Таблица 2 – Сравнение программных средств аналогичного назначения

Критерий \ Приложение	«Todoist»	«TickTick»	«Google Calendar»	«Trello»
1. Возможность добавление задач	2	2	2	2
2. Возможность распределять задачи по сферам жизни, направлениям, проектам	1	1	0	0
3. Возможность настраивать цели	1	2	0	0
4. Возможность отслеживать время выполнения задач	1	2	0	0
5. Возможность установления сроков выполнения	2	2	2	1
6. Возможность визуального планирования (календарь)	2	2	2	2
7. Возможность добавления запланированного времени по направлениям	0	0	0	0

Продолжение таблицы 2

Критерий \ Приложение	«Todoist»	«TickTick»	«Google Calendar»	«Trello»
8. Возможность анализа продуктивности пользователя	1	2	0	0
ИТОГО	10	13	6	5

Для наглядности, полученные результаты отображены на диаграмме, приведенной на рисунке 5.

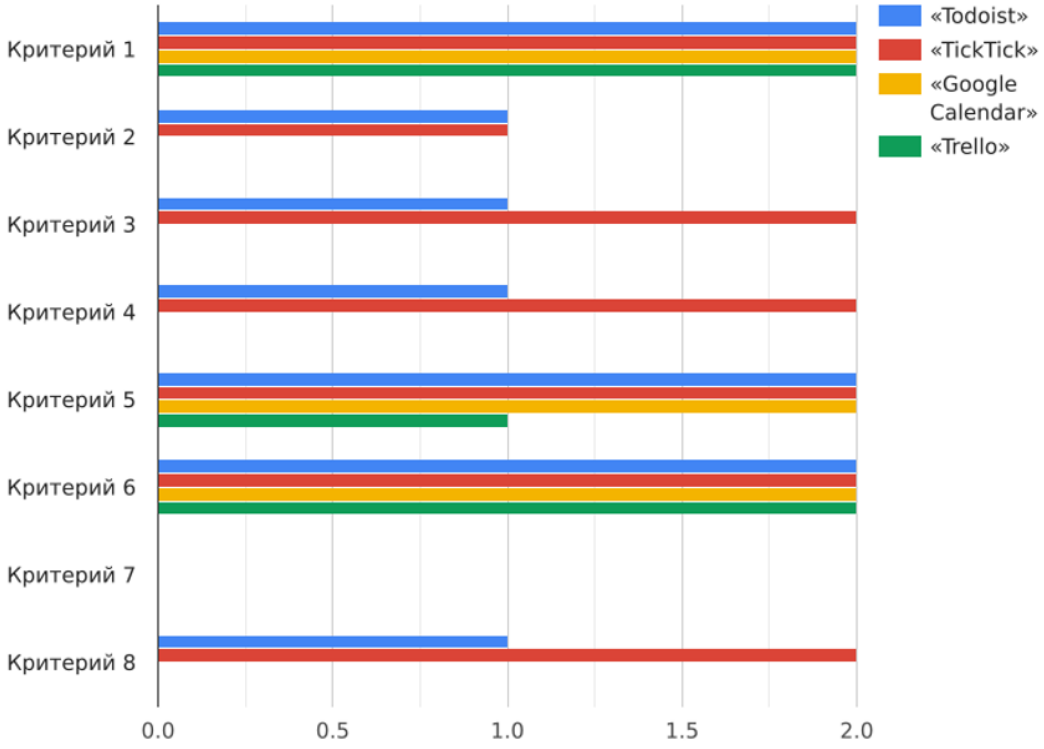

Рисунок 5 – Результаты сравнения приложений-аналогов

Подводя итог обзора программных средств для осознанного планирования жизни можно сделать вывод, что ни одно из рассмотренных решений не соответствует объявленным критериям в полном объеме. Среди всех продуктов-аналогов «TickTick» выделяется большим количеством реализованных функций, однако данное приложение не обеспечивает пользователям возможность добавления запланированного времени, а также не обеспечивает глубокий функционал классификации задач по сферам жизни, направлениям и проектам.

В результате данного обзора было принято решение разработать собственное программное средство. В новое программное средство для

осознанного планирования жизни будут включены функции планирования времени, распределения задач по сферам жизни, направлениям и проектам, отслеживания прогресса и анализа продуктивности, что делает его более полноценным инструментом для планирования и управления задачами.

1.3 Анализ функциональных требований

В процессе анализа задачи на разработку программных средств для осознанного планирования жизни были выделены следующие функциональные требования:

- регистрация и авторизация пользователя;
- редактирование данных о пользователе;
- настройка системы: создание сфер жизни, направлений, целей и проектов;
- удаление отдельных настроек системы (удаление сфер жизни, направлений, сфер жизни и целей);
- добавление задачи;
- удаление задачи;
- изменение задачи;
- отслеживания время выполнения задач;
- просмотр календарей с запланированным и фактическим затраченным временем по направлениям;
- добавление плана на определенный промежуток времени по направлениям;
- изменение плана на определенный промежуток времени по направлениям;
- удаление плана за определенный промежуток времени;
- формирование отчета с анализом продуктивности деятельности человека за определенный промежуток времени;
- просмотр отчета о деятельности человека за определенный период.

В соответствии с выявленными требованиями была разработана диаграмма вариантов использования UML, представленная на рисунке 6.

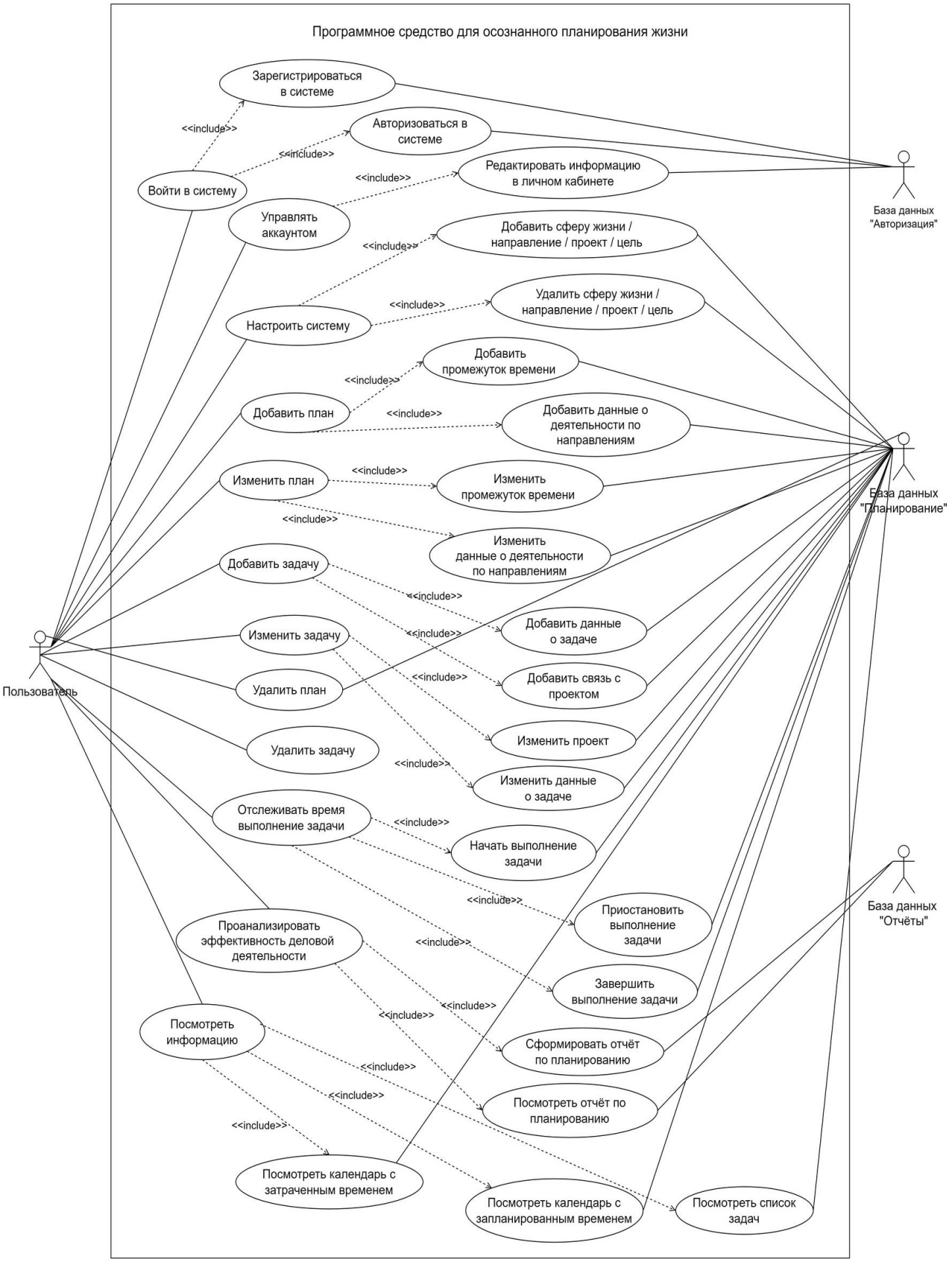

Рисунок 6 – Диаграмма вариантов использования

Ниже приведены спецификации наиболее важных бизнес-процессов.

Сценарий варианта использования «Войти в систему» представлен на рисунке 7.

Прецедент: Войти в систему
ID: 1
Краткое описание: Пользователь регистрируется или авторизуется в приложении
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение; 2. Пользователь не был зарегистрирован/авторизован.
Основной поток: 1. Пользователь переходит на страницу входа в приложение 2. Пользователь вводит данные для входа в существующий аккаунт (логин и пароль) 3. Пользователь получает доступ к функционалу приложения Альтернативные потоки: 2.а Пользователь вводит данные для регистрации (логин, пароль, электронную почту, дату рождения)
Постусловия: 1. Пользователь вошел в приложение 2. Пользователю доступен функционал приложения

Рисунок 7 – Сценарий варианта использования «Войти в систему»

Сценарий варианта использования «Настроить систему» изображен на рисунке 8.

Прецедент: Настроить систему
ID: 2
Краткое описание: Пользователь настраивает сферу жизни, направление, проект, цель.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь прошел авторизацию 2. Пользователь перешёл на вкладку «Настройка планирования»
Основной поток: 1. Пользователь добавляет сферу жизни, направление, проект, цель Альтернативные потоки: 1.а Пользователь редактирует сферу жизни, направление, проект, цель 1.б Пользователь удаляет сферу жизни, направление, цель
Постусловие: 1. Система настроена по предпочтениям пользователя

Рисунок 8 – Сценарий варианта использования «Настроить систему»

Сценарий варианта использования «Добавить задачу» представлен на рисунке 9.

Прецедент: Добавить задачу
ID: 3
Краткое описание: Пользователь добавляет задачу в систему
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь прошел авторизацию 2. Пользователь настроил систему под свои предпочтения 3. Пользователь перешел на вкладку с функционалом добавления задач
Основной поток: 1. Добавить данные о задаче 2. Привязать задачу к конкретному проекту
Постусловие: 1. Задача создана 2. Задача добавлена в систему

Рисунок 9 – Сценарий варианта использования «Добавить задачу»

В результате анализа функциональных требований были выделены варианты использования программы. Для некоторых вариантов использования, которые являются основными для работы разрабатываемого программного средства, были представлены их спецификации, позволяющие отобразить последовательность действий пользователя и системы.

1.4 Анализ нефункциональных требований

В программном средстве для осознанного планирования жизни предусматривается разработка трех баз данных: база данных «Авторизация», база данных «Планирование» и база данных «Отчеты». Каждая база данных будет хранить информацию, относящуюся к соответствующему микросервису. Разработка баз данных будет проводиться с использованием PosgreSQL на языке SQL, который является стандартом. Базы данных будут храниться на удаленном сервере.

Клиентская часть приложения будет написана на языке JS с использованием библиотеке ReactJS в интегрированной среде разработки

Microsoft Visual Code. Для сборки модулей клиентского приложения в единый файл будет использовать сборщик Webpack.

Серверная часть приложения будет реализована с использованием микросервисной архитектуры. В разрабатываемом приложении будет выделено три микросервиса, таких как «Авторизация», «Планирование», «Отчеты», каждый микросервис обращается к своей базе данных, взаимодействие между ними происходит через REST API. Для развертывания приложения в продуктивной среде будет использоваться Docker, каждый микросервис содержится в своем Docker контейнере. Серверная часть приложения написана на языке программирования C#.

Использование PostgreSQL как базы данных обеспечивает высокую производительность, масштабируемость и надежность. ReactJS, в сочетании с Webpack, позволяет разработать эффективное и масштабируемое клиентское приложение. Микросервисная архитектура позволяет разделять функциональность приложения на независимые модули, что упрощает разработку, тестирование и обслуживание. Взаимодействие серверной и клиентской части приложения осуществляется через REST API [4].

Построение диаграмм осуществляется с помощью языка UML 2.0 в среде draw.io. Это бесплатный онлайн-сервис, позволяющий создавать диаграммы, схемы и прочие представления данных [5].

Разрабатываемое web-приложение должно быть оптимизировано для работы в современных браузерах, таких как Google Chrome (версии 80 и выше), Mozilla Firefox (версии 75 и выше), Apple Safari (версии 13 и выше), Microsoft Edge (версии 80 и выше).

2 Проектирование программных средств для осознанного планирования жизни

2.1 Архитектура приложения

Архитектура программных средств для осознанного планирования жизни представляет собой микросервисную архитектуру в сочетании с шаблоном проектирования MVC. Контроллер и модель реализованы внутри каждого микросервиса, они отвечают за хранение и обработку данных, а также обрабатывают действия пользователя, соединяют модели и представления. Клиентское приложение обеспечивает отображение данных и служит основным интерфейсом для получения команд от пользователей, которые впоследствии направляются на сервер посредством REST API [6].

На рисунке 10 изображена диаграмма развертывания сервиса, которая отображает архитектуру программных средств для осознанного планирования жизни.

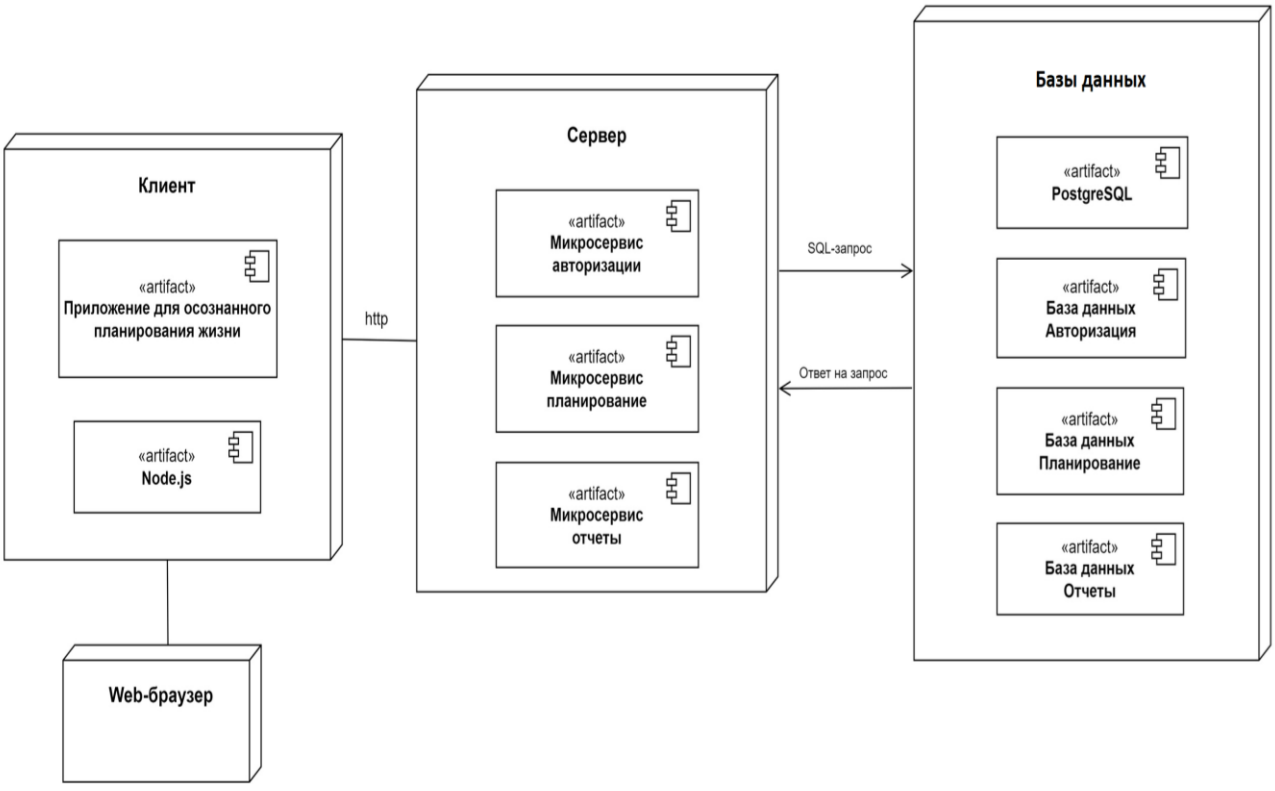

Рисунок 10 – Архитектура программных средств для осознанного планирования жизни

Серверная часть приложения представлена в виде трех независимых микросервисов, каждый из которых имеет свой функционал, отвечающий за определенную часть целостного приложения, каждый сервис взаимодействует с отдельной базой данных. Микросервисы могут обращаться друг к другу через API для совместной обработки данных. Таким образом, серверная часть выступает промежуточным слоем между запрашиваемыми ресурсами и базами данных [7].

После получение данных микросервисом от БД, он их анализирует, создает ответ в виде JSON и отправляет его назад клиенту для визуализации. Этот процесс подчеркивает адаптивность и возможность масштабирования системы, обеспечивая автономную и эффективную работу каждого из элементов.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Концептуальная модель – это модель, отображающая данные и связи предметной области с учетом их семантического содержания, независимо от представления в компьютере [8].

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

Таблица 3 – Сущности БД

Название сущности	Идентификатор сущности	Назначение сущности
Пользователь	DbUser	Содержит сведения о пользователе
Задача	DbTask	Содержит сведения о задачах пользователя
Проект	DbProject	Содержит сведения о проектах пользователя
Затраченное время	DbRealExecutionTime	Содержит информацию о затраченном времени на выполнение задачи

Продолжение таблицы 3

Название сущности	Идентификатор сущности	Назначение сущности
Сфера жизни	DbLifeSphere	Содержит сведения о сферах пользователя
Цель	DbGoal	Содержит сведения о целях пользователя
Направление	DbActionDirection	Содержит сведения о направлениях пользователя
Планируемое время	DbPlanningTime	Содержит информацию о запланированном времени на определенное направление
Информация о файле	DbFileInfo	Содержит сведения о файле
Данные файла	DbFileData	Содержит контент (содержимое) файла
Ссылка на файл	DbFileLink	Содержит ссылку на файл
Настройки формирования файла	DbReportSettings	Содержит сведения о настройках формирования файлов

Графическое представление концептуальной модели представлено на рисунке 11.

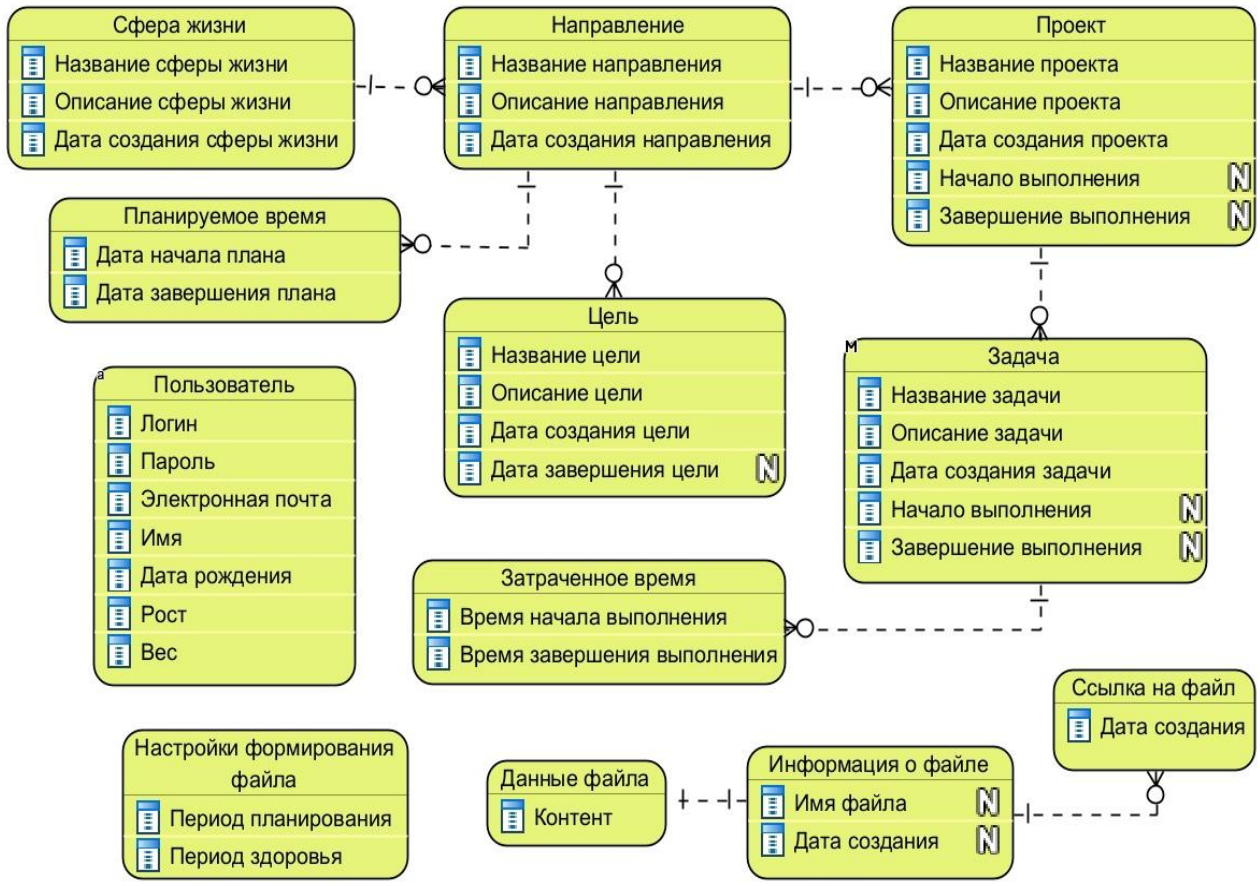

Рисунок 11 – Концептуальная модель базы данных предметной области «Осознанное планирование жизни»

Данное представление построено в среде Visual Paradigm 17.0 в нотации IDEF1X.

Наборы атрибутов для каждой сущности предметной области и их характеристики приведены в приложении Б.

2.2.2 Логические модели баз данных

В разрабатываемых программных средствах для осознанного планирования жизни была выделена разработка трех баз данных: «Авторизация», «Планирование», «Отчеты».

Разделение было произведено по следующим причинам:

– разделяя данные по функциональным областям, удобнее управлять их хранением и обработкой;

– использование нескольких баз данных позволяет сбалансировать нагрузку;

– разделение данных по нескольким БД позволяет эффективнее организовывать резервное копирование и восстановление данных [9].

Логические модели проектируемых баз данных представлены на рисунках В.1, В.2 и В.3 в приложении В.

2.3 Проектирование пользовательского интерфейса

Интерфейс web-приложения должен отвечать всем функциональным требованиям, а также должен быть удобным в использовании.

Проектируемый интерфейс включает в себя страницы входа, регистрации и главной страницы с навигационным меню по приложению. На главной странице должны располагаться вкладки с различным содержимым, таким как список имеющихся в системе задач, настройки системы, календари с запланированным и фактическим затраченным временем, а также графики с анализом деятельности человека.

Кроме вкладок и страниц в приложении должны быть реализованы модальные окна добавления таких объектов, как «Задача», «Сфера жизни»,

«Направление», «Проект», «Цель», «Запланированное время», модальные окна детального просмотра объектов системы, созданных пользователем.

Каждая составляющая интерфейса будет реализована с использованием компонентов библиотеки ReactJS, контролирующих свое состояние и функциональность. Применение архитектуры, основанной на компонентах, упрощает расширение приложения и использование кода повторно [10].

2.4 Проектирование классов программных средств для осознанного планирования жизни

В ходе проектирования серверной части приложения, для каждого из микросервисов были разработаны диаграммы классов согласно шаблону проектирования MVC.

Диаграммы классов для микросервисов «Планирование», «Авторизация» и «Отчеты» представлены на рисунках В.4, В.5 и В.6 в приложении В.

Краткое описание функционального назначения наиболее значимых классов микросервиса «Планирование» приведено в таблице 4.

Таблица 4 – Компоненты диаграммы классов микросервиса «Планирование»

Компонент	Назначение
Program	Точка входа в программу
HttpContextExtensions	Класс, предоставляющий методы расширения для `HttpContext`, извлечение токена авторизации из запроса.
Controller	Класс, реализующий обработку пользовательских запросов, взаимодействие между моделями и представлениями.
ActionDirectionService	Класс, предоставляющий бизнес-логику для управления данными, связанными с направлениями жизни.
ActionDirectionRepository	Класс, содержащий механизмы для обработки данных, связанных с направлениями жизни.
ActionDirection	Модель, использующаяся для отображения данных о направлении деятельности.
LifeSphereService	Класс, предоставляющий бизнес-логику для управления данными, связанными со сферами жизни.
LifeSphereRepository	Класс, содержащий механизмы для обработки данных, связанных со сферами жизни.

Продолжение таблицы 4

Компонент	Назначение
LifeSphere	Модель, использующаяся для отображения данных о сфере жизни.
ProjectService	Класс, предоставляющий бизнес-логику для управления данными, связанными с данными о проектах.
ProjectRepository	Класс, содержащий механизмы для обработки данных, связанных с проектами.
ProjectDirection	Модель, использующаяся для отображения данных о проекте.
TaskService	Класс, предоставляющий бизнес-логику для управления данными, связанными с задачами.
TaskRepository	Класс, содержащий механизмы для обработки данных, связанных с задачами.
Task	Модель, использующаяся для отображения данных о задаче.
RealExecutionTimeService	Класс, предоставляющий бизнес-логику для управления данными, связанными с затраченным временем.
RealExecutionTimeRepository	Класс, содержащий механизмы для обработки данных, связанных с затраченным временем.
RealExecutionTime	Модель, использующаяся для отображения данных о затраченном времени.
DataContext	Класс, содержащий методы для выполнения различных операций с базой данных.
DapperContext	Класс, использующий библиотеку Dapper для доступа к базе данных, обеспечивающий выполнение SQL-запросов и преобразование ответов БД в объекты приложения.

Компоненты диаграммы классов микросервисов «Отчеты» и «Авторизация» имеют идентичное функциональное назначение классов и придерживаются такой же логики обработки запроса от пользователя, как и в микросервисе «Планирование».

3 Реализация программных средств для осознанного планирования жизни

3.1 Реализация базы данных

Физическая модель базы данных – это самое детальное описание ее реализации, включая типы данных, индексы, физическое размещение файлов.

На основе концептуальной модели базы данных и логических были построены физические модели базы данных для баз, использующихся в разрабатываемом программном средстве [11].

Физическая модели баз данных «Планирование», «Авторизация и Отчеты» были построены в среде Visual Paradigm 17.0 в нотации IDEF1X, они представлены на рисунках В.7, В.8 и В.9 в приложении В.

База данных, соответствующая разработанной физической модели, была развернута с помощью механизма миграций.

Механизм миграций необходим для версионирования и синхронизации структуры БД.

3.2 Реализация серверной части приложения

Серверная часть программных средств для осознанного планирования жизни разрабатывалась в среде программирования JetBrains Rider, на языке программирования C#. Код серверной части приложения приведен в приложении Г.

Серверная часть состоит из трех микросервисов. Каждый микросервис содержит модели, отражающие структуру баз данных, а также сериализаторы для преобразования данных моделей в удобный формат для обмена между сервером и клиентом. Кроме того, реализованы специализированные классы для миграции данных, что способствует адаптации и поддержке структуры данных в соответствии с изменяющимися требованиями. Эти классы важны для автоматизации процессов обновления схем данных, что помогает

минимизировать возможные риски потери информации при внедрении нововведений и облегчает развертывание обновлений в операционной среде.

Разработанные сервисы предоставляют REST API для взаимодействия. Реализованные методы описаны в таблице 5.

Таблица 5 – Методы API программных средств для осознанного планирования жизни

URL запроса	Http метод	Описание
/User/all	GET	Получение информации о всех пользователях
/Auth/auth	POST	Авторизация пользователя
/Auth/reg	POST	Регистрация пользователя
/life-sphere/by-user/{userId}	GET	Получение всей информации о планировании определенного пользователя
/life-sphere	POST	Создание сферы жизни
/life-sphere/{Id}	DELETE	Удаление сферы жизни
/action-direction	POST	Создание направления сферы жизни
/action-direction/{Id}	DELETE	Удаление направления сферы жизни
/goal	POST	Создание цели
/goal/{Id}	DELETE	Удаление цели
/project	POST	Создание проекта
/project/{Id}	DELETE	Удаление проекта
/task/add	POST	Создание задачи
/task/{Id}	DELETE	Удаление задачи
/planning-time	POST	Создание запланированного времени
/planning-time/{Id}	DELETE	Удаление запланированного времени
/real-execution-time	POST	Создание затраченного времение
/real-execution-time/{Id}	DELETE	Удаление затраченного времени
/report/planning/{userId}	GET	Получение отчета по планированию по определенному пользователю
/files/{id}	GET	Получение файла

Разработанные микросервисы обернуты в docker контейнеры, это сделано с целью дальнейшего разворачивания приложение на других серверах с платформами, поддерживаемыми docker [12].

Конфигурация взаимодействия между контейнерами описана в файле docker-compose.yml. Процесс развертывания начинается с развертывания баз данных, после чего происходит запуск микросервисов.

3.3 Реализация клиентской части приложения

Клиентская часть программных средств для осознанного планирования жизни разрабатывалась в среде программирования Visual Studio Code, на языке программирования JavaScript с использованием библиотеки ReactJS. Код клиентской части приложения приведен в приложении Д.

Сборка клиентской части приложения осуществляется посредством модульного сборщика Webpack, который облегчает управление зависимостями, модулями и компонентами в проекте.

На рисунке 12 представлена диаграмма компонентов клиентской части разрабатываемого приложения.

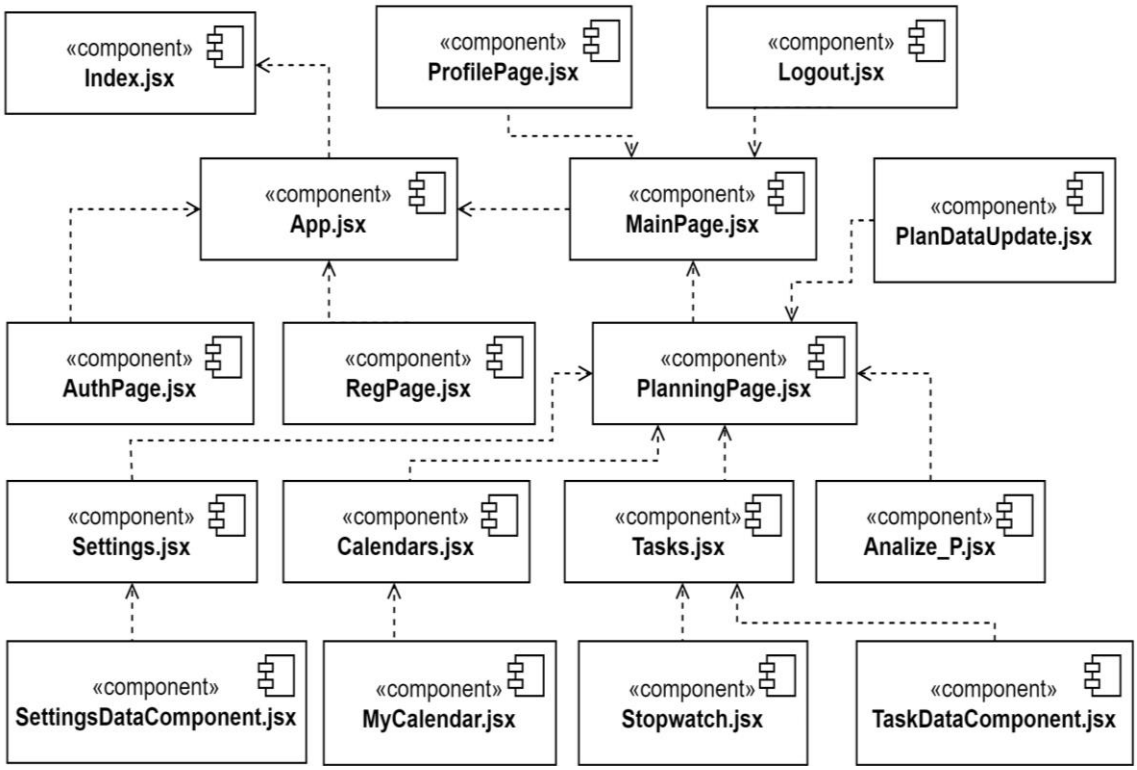

Рисунок 12 – Диаграмма компонентов клиентской части приложения

Описание компонентов клиентской части приложения представлено в таблице 6.

Таблица 6 – Компоненты клиентской части приложения

Компонент	Назначение
Index.jsx	Входная точка для React приложения
App.jsx	Корневой компонент, содержащий в себе маршрутизацию.
AuthPage.jsx	Компонент, реализующий функционал страницы авторизации.
RegPage.jsx	Компонент, реализующий функционал страницы регистрации.
MainPage.jsx	Компонент, реализующий функционал главной страницы, включающий в себя маршрутизацию по компонентам доступным после авторизации пользователем.
ProfilePage.jsx	Компонент, реализующий функционал страницы личного компонента с выводом информации о пользователе.
Logout.jsx	Компонент, реализующий функционал выхода из системы.
PlanningPage.jsx	Компонент, реализующий функционал страницы с разделами планирования и маршрутизацией по ним, так же в данном компоненте вызывается компонент PlanDataUpdate для получения информации о деятельности пользователя.
PlanDataUpdate.jsx	Компонент, использующийся для получения всех данных о пользователе по планированию при авторизации.
Settings.jsx	Компонент, реализующий функционал настройки системы.
Settings.jsx	Компонент, реализующий функционал настройки системы.
SettingsDataComponent.jsx	Компонент, реализующий отображение информации о настройках системы.
Tasks.jsx	Компонент, реализующий функционал страницы задач.
TasksDataComponent.jsx	Компонент, реализующий отображение информации о задачах.
Calendars.jsx	Компонент, реализующий функционал отображения календарей с запланированным и затраченным временем.
MyCalendar.jsx	Компонент предоставляющий методы для вычисления высоты и параметров календаря.
Stopwatch.jsx	Компонент, отвечающий за отслеживания времени по определенной задачи.
Analize_P.jsx	Компонент, реализующий анализ продуктивности деятельности пользователя, включает в себя использование библиотек для графического отображения данных.

Для передачи данных между сервером и клиентом используется библиотека axios. Выбор данной библиотеки обусловлен удобством настройки запросов и простотой перехвата исключений.

3.4 Реализация интерфейса приложения

В соответствии с требованиями к пользовательскому интерфейсу, предоставленными в пункте 2.3, были реализованы следующие страницы web-приложения для осознанного планирования жизни:

– страница авторизации;

– страница регистрации;

– страница личного кабинета пользователя;

– страница с функционалом планирования.

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

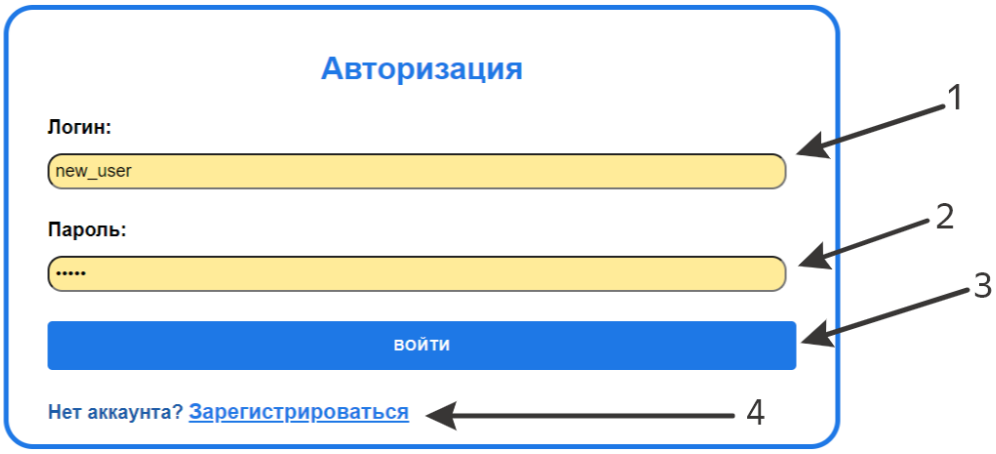

Рисунок 13 – Страница авторизации

Основные элементы страницы авторизации:

1 – поле ввода логина пользователя;

2 – поле ввода пароля пользователя;

3 – кнопка «Войти»;

4 – кнопка перехода на страницу регистрации.

При нажатии на кнопку «Войти» система проверяет правильность введенных данных пользователем и проверяет наличие зарегистрированного пользователя в системе. Если обработка данных произведена успешна, то пользователь переходит на страницу с функционалом планирования.

Страница регистрации, на которую пользователь может перейти со страницы авторизации, представлена на рисунке 14.

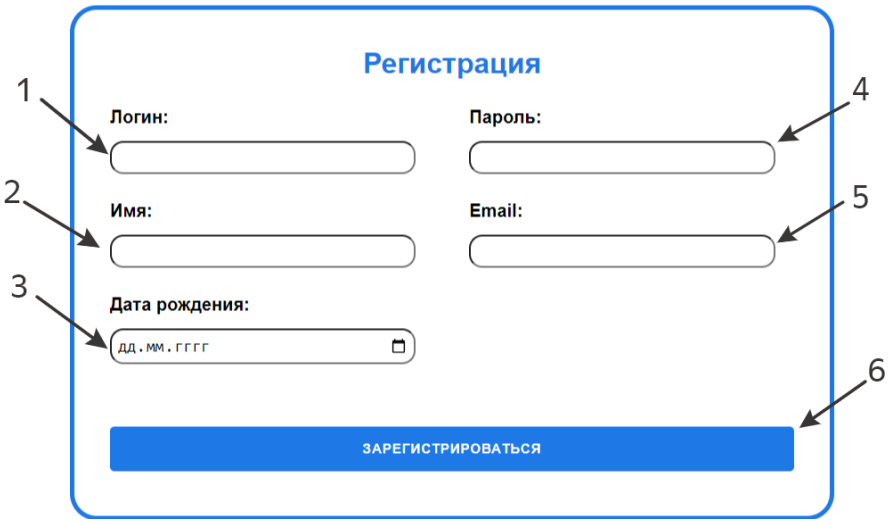

Рисунок 14 – Страница регистрации

Основные элементы страницы регистрации:

1 – поле ввода логина пользователя;

2 – поле ввода имени пользователя;

3 – поле ввода даты рождения пользователя;

4 – поле ввода пароля пользователя;

5 – поле ввода электронной почты пользователя;

6 – кнопка «Зарегистрироваться».

При нажатии на кнопку «Зарегистрироваться» система проверяет правильность введенных данных пользователем, а также проверяет наличие пользователя с таким логином в системе, если данного пользователя в системе еще нет и данные корректны, то пользователь заносится в систему и происходит переход на страницу авторизации.

Страница личного кабинета, на которую можно попасть после авторизации в приложении представлена на рисунке 15.

Рисунок 15 – Страница личного кабинета пользователя

Страница личного кабинета пользователя содержит информацию о пользователе, а также кнопку «Изменить данные», которая предназначена для изменения таких параметров как имя, дата рождения, пароль и электронная почта пользователя.

Страница с функционалом планирования содержит в себе несколько вкладок: вкладка с настройками системы, вкладка с задачами, вкладка с календарями и вкладка с анализом продуктивности.

Содержимое вкладки настройка системы представлено на рисунке 16.

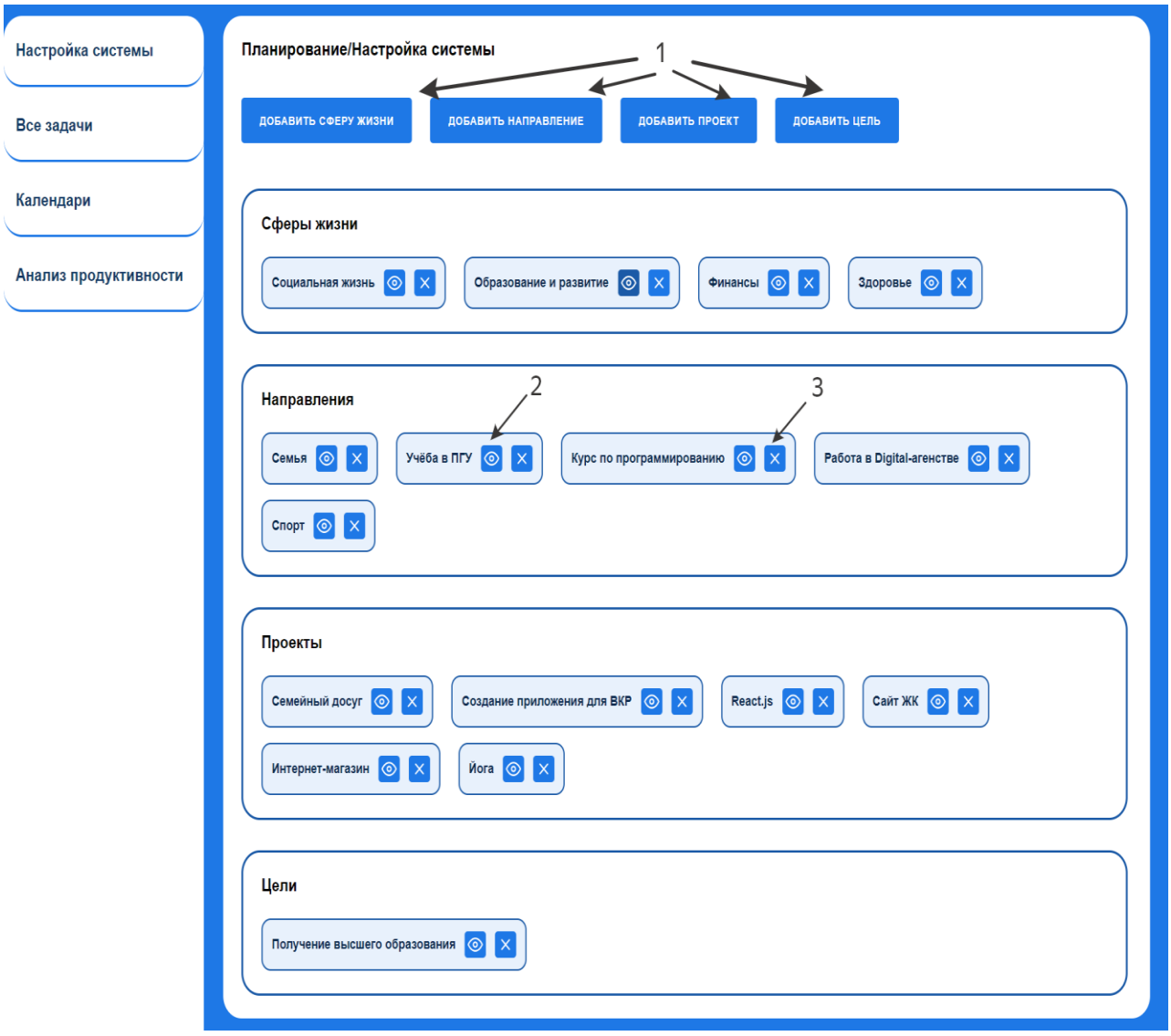

Рисунок 16 – Вкладка с настройками системы

Основные элементы вкладки с настройками системы:

1 – кнопки для добавления новых объектов системы;

2 – кнопка для просмотра детальной информации об объекте системы;

3 – кнопка для удаления выбранного объекта системы.

На данной страницы выводятся объекты, связанные только с настройками системы, также стоить заметить, что все эти объекты созданы пользователем.

При клике на кнопку для добавления нового объекта системы такого как, сфера жизни, направления, проекта или цели открывается соответствующее модальное окно с формой.

Модальное окно для добавления проекта представлено на рисунке 17, модальные окна с добавлением сферы жизни, направления и цели выглядят идентично по дизайну.

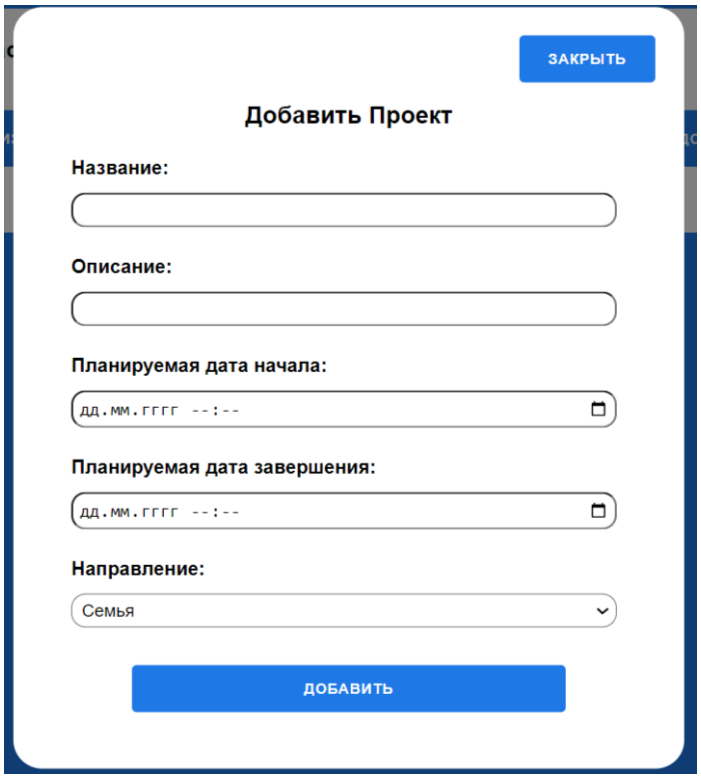

Рисунок 17 – Модальное окно для добавления проекта

При клике на кнопку для просмотра детальной информации об объекте системы такого как сфера жизни, направление, проект и цель открывается модальное окно с подробным описанием.

Модальное окно для детального просмотра информации о направлении представлено на рисунке 18. Модальные окна с детальным просмотром информации о сфере жизни, проекте или же цели выглядят идентично по

дизайну, меняется лишь содержимое, которое зависит от добавленной пользователем информации.

Рисунок 18 – Модальное окно детального просмотра направления

Содержимое вкладки с задачами представлено на рисунке 19.

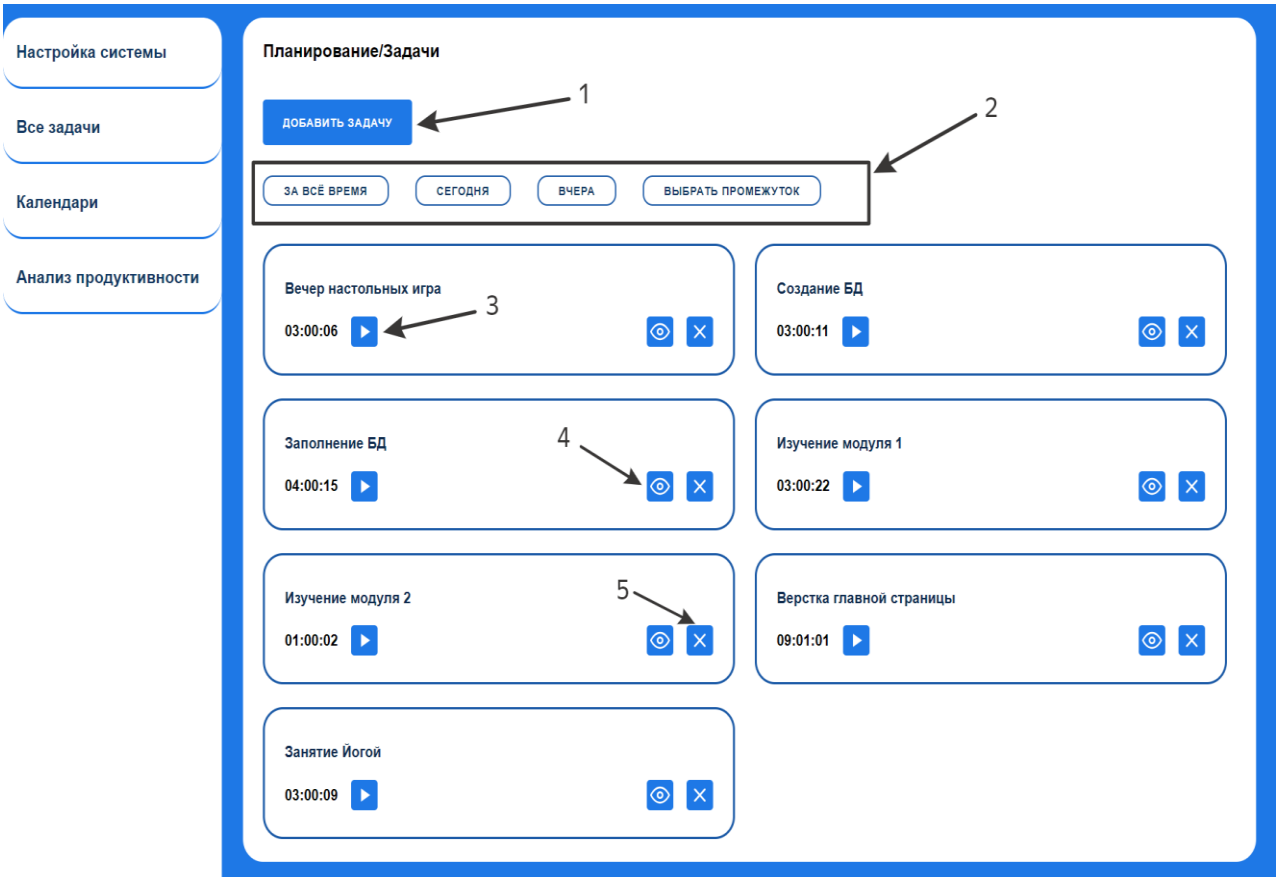

Рисунок 19 – Вкладка с задачами

Вкладка с задачами содержит такие элементы как:

1 – кнопки для добавления задачи;

2 – кнопки для фильтрации задач по дате добавления их в систему;

3 – кнопка для отслеживания времени выполнения задач;

4 – кнопка для детального просмотра содержимого задачи;

5 – кнопка для удаления задачи из системы.

При клике на кнопку для добавления новой задачи открывается модальное окно для добавления задачи, представленное на рисунке 20.

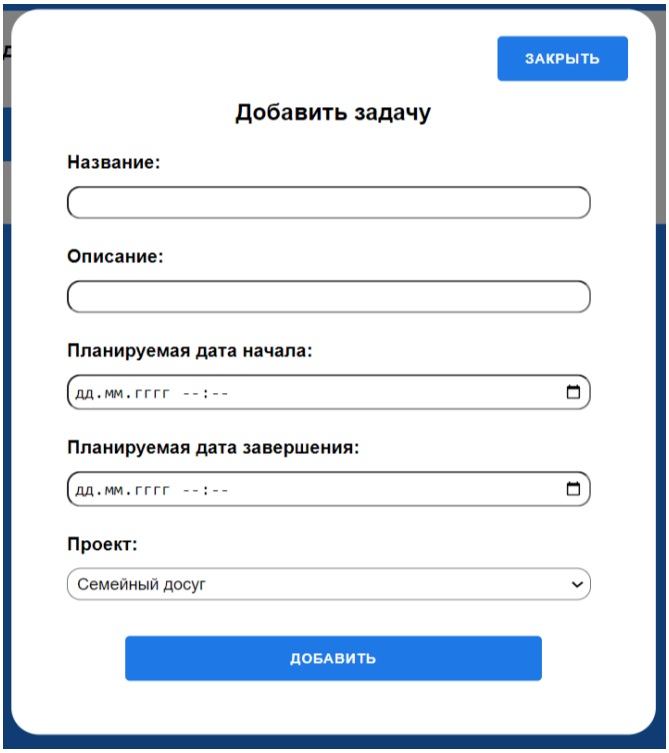

Рисунок 20 – Модальное окно для добавления задачи

Основные элементы вкладки с календарями:

1 – кнопки для навигации по календарю;

2 – календарь;

3 – блок, показывающий продолжительность направления;

4 – кнопка для перехода к другому виду календаря;

5 – кнопка для добавления запланированного времени.

Основные элементы вкладки с календарями одинаковые как для календаря с фактически затраченным временем, так и для календаря с запланированным временем.

Высота блока, показывающего продолжительность направления зависит от количества затраченных или же запланированных часов, данные элементы

разноцветные, так как они наглядно демонстрируют, что это разные направления, при наведении на элемент появляется название направления.

Содержимое вкладки с календарем с запланированным временем представлена на рисунке 21.

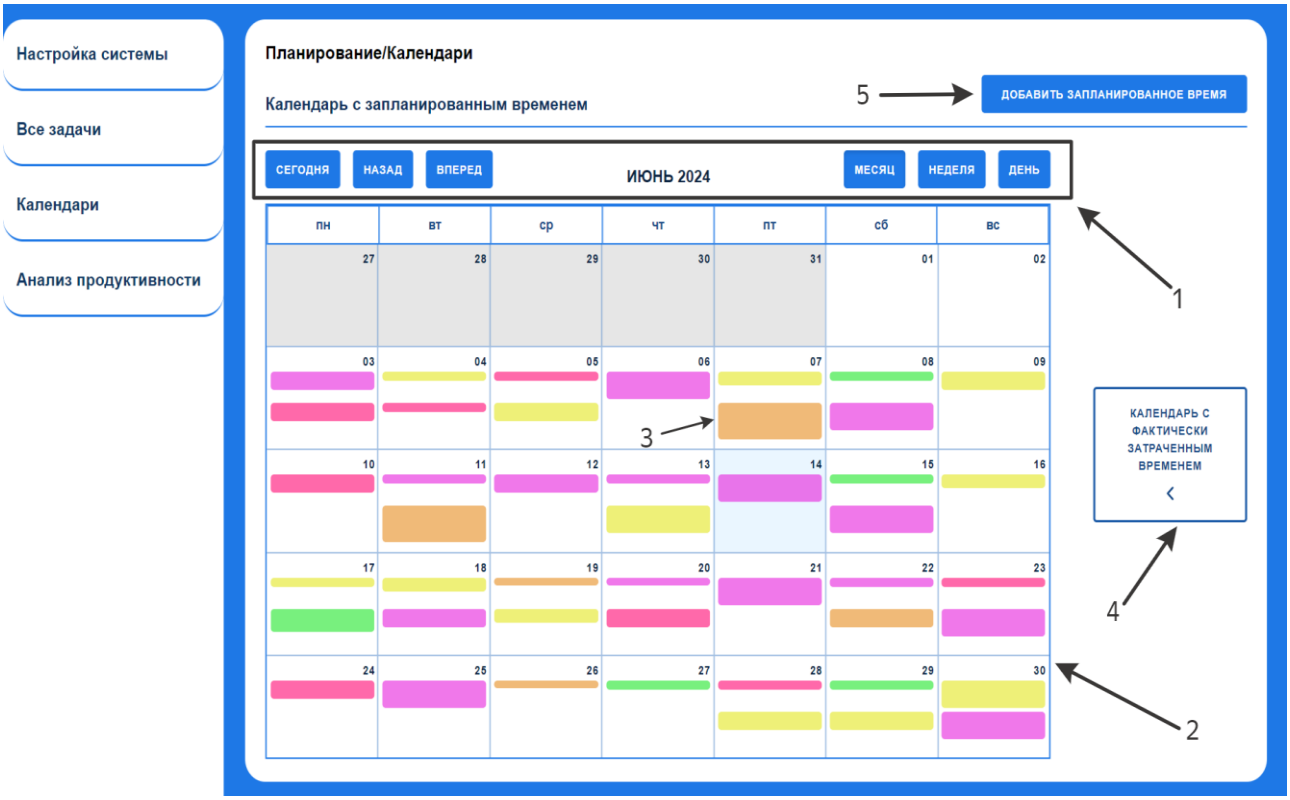

Рисунок 21 – Вкладка с календарем с запланированным временем

При клике на кнопку 5 открывается модальное окно с добавлением запланированного времени, изображенное на рисунке 22.

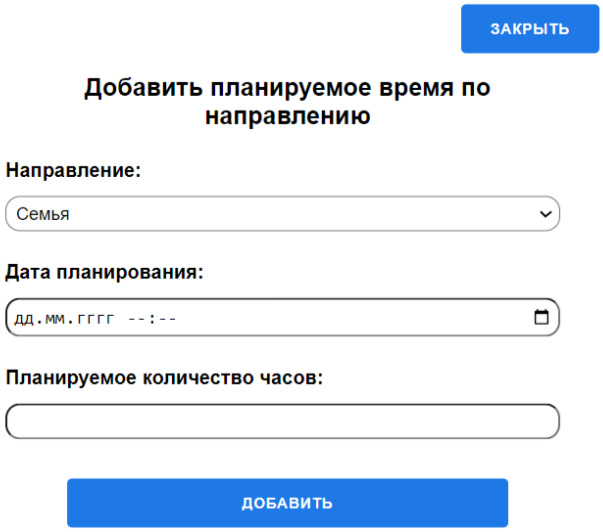

Рисунок 22 – Модальное окно с добавлением планируемого времени

Содержимое вкладки с календарем с фактически затраченным временем представлена на рисунке 23.

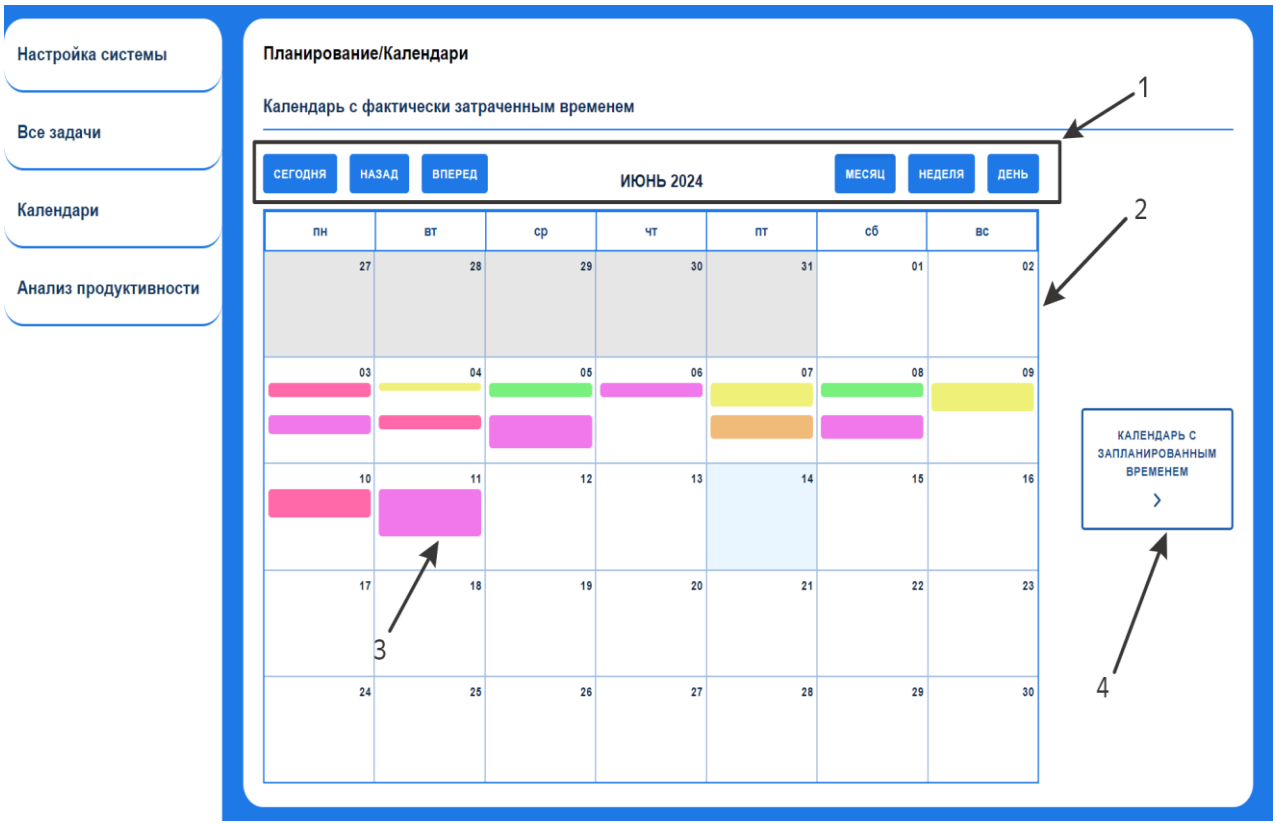

Рисунок 23 – Вкладка с календарем с затраченным временем

Вкладка с анализом продуктивности содержит несколько графиков, которые наглядно демонстрируют деятельность человека за последнюю неделю.

На первом графике изображено сравнение фактически затраченного времени по всем направлениям за последнюю неделю с запланированным.

В таблицу, представленную в отчете, выводится информация о всех задачах, выполненных пользователем за последнюю неделю.

На втором графике изображены все направления и фактически затраченное время по ним на каждый день в часах, на третьем графике хранится идентичная информация, но только берутся значения запланированного времени.

На рисунке 24 изображена вкладка с анализом продуктивности пользователя.

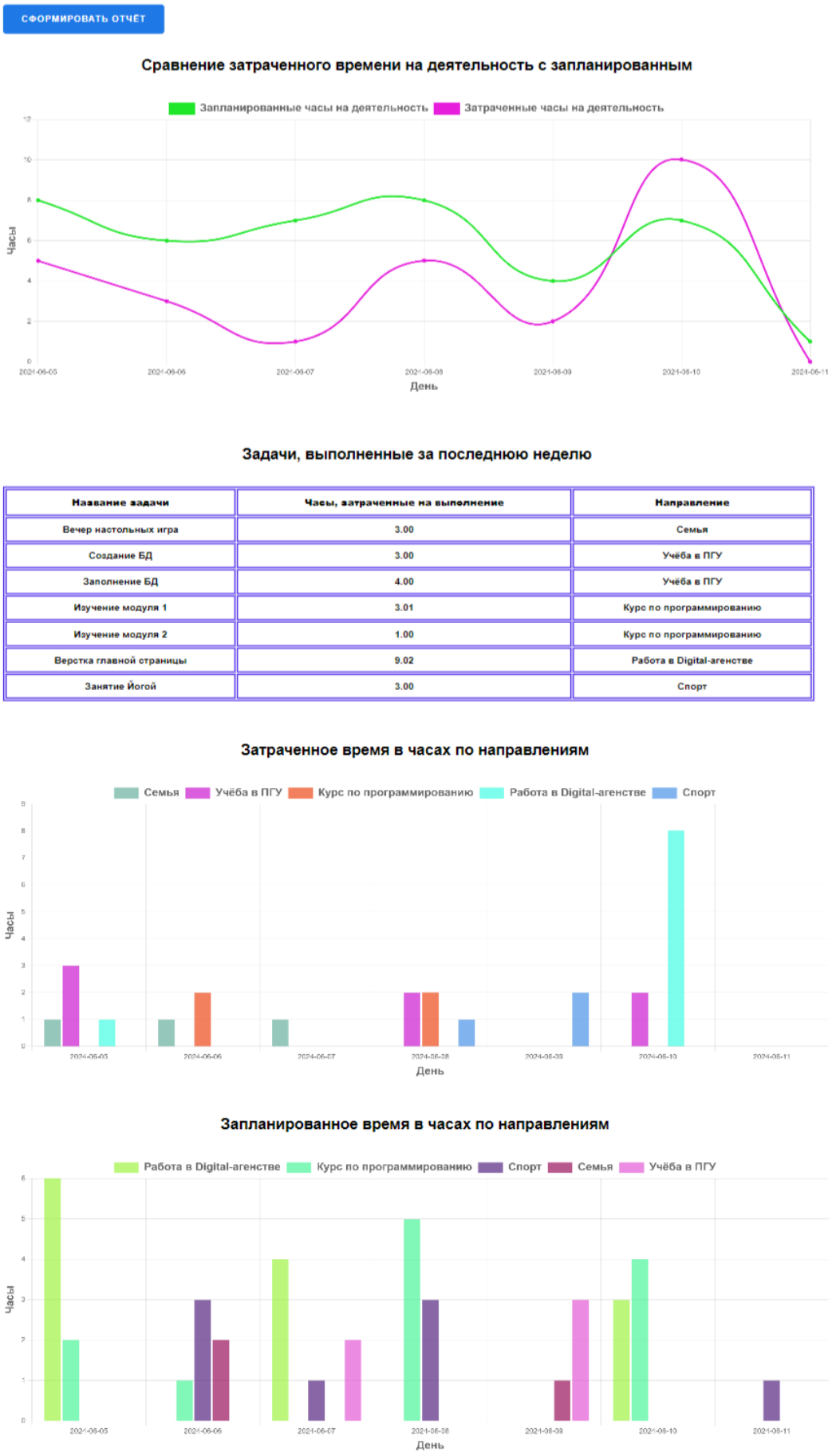

Рисунок 24 – Вкладка с анализом продуктивности

Разработка графического интерфейса производилась с использованием каскадной таблицы стилей CSS3 и гипертекстовой разметки HTML5. HTML5 служит для формирования структуры веб-страниц. CSS3, в свою очередь, применяется для настройки дизайна, включая параметры шрифтов, цветовую гамму и многое другое.

Для корректного отображения календарей использовалась библиотека react-big-calendar. Графическое представление аналитических данных достигнуто с помощью библиотеки chart.js. Эта библиотека предназначена для построения разнообразных типов диаграмм, что позволяет наглядно демонстрировать различные аналитические показатели. Благодаря ее гибкости и адаптации под разные устройства, она эффективно интегрируется в общий интерфейс, облегчая представление и анализ комплексных данных [13].

4 Контроль качества программного обеспечения

4.1 Обоснование выбора режима тестирования и метрик кода

Тестирование приложения различными методами необходимо для обеспечения качества программного обеспечения, улучшения его надежности, усиления защиты от потенциальных угроз, а также для повышения удобства использования приложения пользователем [14].

Наиболее популярны следующие виды тестирования:

– модульное тестирование;
– тестирование производительности;
– функциональное тестирование;
– интеграционное тестирование.

Модульное тестирование направлено на изолированную проверку отдельных компонентов и модулей системы. Автоматизация модульных тестов предоставляет возможность проводить тестирование с высокой скоростью с использованием серверов непрерывной интеграции.

Тестирование производительности представляет собой методику, направленную на оценку надежности, скорости и отзывчивости программного обеспечения.

Функциональное тестирование – это методика тестирования программного обеспечения, направленная на проверку соответствия поведения разработанного продукта заданным функциональным требованиям [15].

Интеграционное тестирование проверяет корректность взаимодействия модулей и сервисов программного обеспечения.

В качестве метода ручного тестирования программного обеспечения было выбрано функциональное тестирование. Данный выбор обусловлен тем, что функциональное тестирование позволит всесторонне оценить соответствие системы заявленным требованиям и проверить ее работоспособность, моделируя реальные условия использования.

Кроме того, в рамках автоматического тестирования был выбран метод модульного тестирования. Этот подход эффективен, поскольку он позволяет проверить работу отдельных сервисов и компонентов приложения при минимальных временных затратах на создание и отладку продукта.

Метрики программного кода служат ключевым инструментом для анализа и оценки качественных характеристик программного обеспечения.

Они предоставляют возможность детально оценить технические характеристики продукта, прогнозировать трудозатраты на разработку, количественно анализировать качество инженерных решений и оценивать сложность и надежность разрабатываемых систем, что способствует повышению общей стабильности и производительности итогового программного продукта [16].

Для оценки качества разработанного программного кода web-приложения для осознанного планирования жизни были выбраны наиболее популярные метрики кода, такие как метрика Холстеда и метрика Чепина.

Метрика Холстеда – это метрика, основанная на анализе операндов, операторов и количества строк программного кода. С ее помощью можно оценить сложность программы, определить объем работ и возможность расширения функционала.

Метрика Чепина фокусируется на оценке качества внутренней логики модулей программы. Она оценивает количество и типы переменных, которые используются в модуле [17].

4.2 Тестирование программных средств для осознанного планирования жизни

4.2.1 Функциональное тестирование

После завершения разработки программного обеспечения было проведено функциональное тестирование. Результаты тестирования приведены в приложении Е. План тестирования представлен в таблице 7.

Таблица 7 – План функционального тестирования

Назначение теста	Описание теста	Результаты теста
Проверка отображения настроек системы	Пользователь переходит на вкладку «Настройка системы», на которой отображены все добавленные настройки	Все объекты настройки системы корректно отображены (рисунок Е.1)
Проверка отображения задач в системе	Пользователь переходит на вкладку «Задачи», на которой отображены все добавленные задачи	Все задачи пользователя корректно отображены (рисунок Е.2)
Проверка отображения затраченного времени в календаре	Пользователь переходит на вкладку с календарем, на котором отображена продолжительность его деятельности	В календаре с затраченным временем корректно отображена продолжительность выполнения направлений (рисунок Е.3)
Проверка отображения запланированного времени в календаре	Пользователь переходит на вкладку с календарем, на котором отображена продолжительность его запланированной деятельности	В календаре с запланированным временем корректно отображена продолжительность запланированного времени по направлениям по датам в часах (рисунок Е.4)
Проверка добавления новой сферы жизни	Пользователь добавляет новую сферу жизни	Система добавляет сферу жизни в базу и выводит сообщение «Данные успешно добавлены» (рисунок Е.5)
Проверка добавления нового направления	Пользователь добавляет новое направление	Система добавляет направление в базу и выводит сообщение «Данные успешно добавлены» (рисунок Е.6)
Проверка добавления нового проекта	Пользователь добавляет новый проект	Система добавляет проект в базу и выводит сообщение «Данные успешно добавлены» (рисунок Е.7)
Проверка добавления задачи	Пользователь добавляет новую задачу	Система добавляет задачу в базу и выводит сообщение «Данные успешно добавлены» (рисунок Е.8)
Проверка просмотра информации о задаче	Пользователь просматривает подробную информацию о задаче	Система отображает подробную информацию о задаче (рисунок Е.9)
Проверка добавления запланированного времени	Пользователь добавляет запланированное время по направлению	Система добавляет новую запись о запланированного времени в систему и выводит сообщение «Данные успешно добавлены» (рисунок Е.10)
Проверка фиксирования время выполнения задачи	Пользователь запускает время по задаче	Система фиксирует время выполнения задачи (рисунок Е.11)

Продолжение таблицы 7

Назначение теста	Описание теста	Результаты теста
Проверка удаления задачи	Пользователь удаляет задачу	Система удаляет задачу из базы и она перестает отображаться на вкладке с задачами (рисунок Е.12)
Проверка удаления проекта	Пользователь удаляет проект	Система удаляет проект из базы и он перестает отображаться на вкладке с настройками системы (рисунок Е.13)
Проверка удаления направления	Пользователь удаляет направление	Система удаляет направление из базы и оно перестает отображаться на вкладке с настройками системы (рисунок Е.14)
Проверка удаления сферы жизни	Пользователь удаляет сферу жизни	Система удаляет сферу жизни из базы и она перестает отображаться на вкладке с настройками системы (рисунок Е.15)
Проверка формирования отчета о продуктивности деятельности пользователя	Пользователь нажимает на кнопку «Сформировать отчет»	Система формирует отчет о деятельности пользователя за последнюю неделю и отображает его (рисунок Е.16)
Проверка авторизации пользователя с неверным паролем	Осуществляется проверка введенных пользователем данных в поле пароль	Приложение произвело проверку введенных данных пользователем и вывело сообщение «Неверный пароль» (рисунок Е.17)
Проверка регистрации пользователя, с некорректной электронной почтой	Осуществляется проверка введенных пользователем данных в поле электронная почта	Приложение произвело проверку введенных данных пользователем и вернуло ошибку «Электронная почта введена некорректно» (рисунок Е.18)
Проверка регистрации пользователя с логином, который был уже занят в системе	Система просматривает список зарегистрированных пользователей	Приложение произвело поиск по списку зарегистрированных пользователей и вывело сообщение «Пользователь с таким логином уже зарегистрирован» (рисунок Е.19)

Программа успешно прошла все функциональные тесты, показав правильность работы программных средств и полное соответствие функциональным требованиям.

4.2.2 Модульное тестирование

Для обеспечения корректности работы разработанного приложения и проверки ожидаемого поведения программы проведено модульное

тестирование. Тестирование компонентов добавления и отображения задач в системе, разработанных на языке JavaScript осуществляется с помощью фреймворка Jest и библиотеки Enzyme [18].

Полный код модульных тестов приведен в приложении Ж. Результаты модульного тестирования приведены в приложении И. План модульного тестирования представлен в таблице 8.

Таблица 8 – План модульного тестирования

Тестируемый компонент	Тест	Ожидаемый результат	Актуальный результат
Tasks	Должен создаться	Пройден	Пройден (рисунок И.1)
	Должен отобразиться заголовок задачи	Пройден	Пройден (рисунок И.1)
	Должно отобразиться описание задачи	Пройден	Пройден (рисунок И.1)
	Должна отобразиться дата начала задачи	Пройден	Пройден (рисунок И.1)
	Должна отобразиться дата окончания задачи	Пройден	Пройден (рисунок И.1)
	Должно отобразиться название проекта	Пройден	Пройден (рисунок И.1)
TaskDataComponent	Должен создаться	Пройден	Пройден (рисунок И.2)
	Должны отобразиться кнопки фильтров	Пройден	Пройден (рисунок И.2)
	Должны отобразиться поля ввода диапазона дат при нажатии на кнопку "Выбрать промежуток"	Пройден	Пройден (рисунок И.2)
	Должно открыться модальное окно при нажатии на кнопку "show"	Пройден	Пройден (рисунок И.2)
	Должен создаться запрос через axios для удаления задачи при нажатии кнопки удаления	Пройден	Пройден (рисунок И.2)
	Должен обновиться список задач при удалении задачи	Пройден	Пройден (рисунок И.2)

Продолжение таблицы 8

Тестируемый компонент	Тест	Ожидаемый результат	Актуальный результат
PlanDataUpdate	Должен создаться запрос через axios для получения данных о планировании	Пройден	Пройден (рисунок И.3)
Modal	Должно создаться модальное окно	Пройден	Пройден (рисунок И.4)
	Должно отобразиться содержимого модального окна	Пройден	Пройден (рисунок И.4)
	Должен вызываться метод onClose при нажатии кнопки закрытия	Пройден	Пройден (рисунок И.4)

Программные средства для осознанного планирования жизни успешно прошли все этапы модульного тестирования.

4.3 Оценка качества программного кода

4.3.1 Метрика Холстеда

Для расчета метрик Холстеда рассматривается функция filterHistories компонента TaskDataComponent. Данная функция реализует функционал фильтрации задач по датам. Код данной функции представлен в листинге Д.14 приложения Д.

Для вычисления данной метрики требуется посчитать число вхождения уникальных операторов (N1) и операндов (N2), а также общее количество операторов (n1) и операндов (n2).

Операторы функции filterHistories и число их вхождений приведены в таблице 9.

Таблица 9 – Операторы функции filterHistories

№	Операторы	Число вхождений
1	=>	4
2	<=	1
3	>=	1
4	===	2

Продолжение таблицы 9

№	Операторы	Число вхождений
5	-	1
6	.	13
7	=	3
8	&&	1
9	default	1
10	filter	3
11	return	5
12	case	4
13	switch	1
14	getDate	1
15	setDate	1
16	setHours	5
17	const	3
18	new	5
19	n1 = 18	N1 = 42

Операнды функции filterHistories и число их вхождений приведены в таблице 10.

Таблица 10 – Операнды функции filterHistories

№	Операнды	Число вхождений
1	filterHistories	1
2	today	4
3	Date	3
4	yesterday	3
5	filter	4
6	'today'	1
7	taskHistories	3
8	history	6
9	'yesterday'	1
10	'range'	1
11	historyDate	2
12	dateRange	2
13	startDate	1
14	endDate	1
15	'all'	1
16	n2 = 15	N2 = 34

Данные, полученные в ходе обработки программного кода функции filterHistories компонента TaskDataComponent были использованы для расчета характеритик программы по формулам из метрики Холстеда. Результаты вычислений представлены в таблице 11.

Таблица 11 – Результаты вычислений по метрике Холстеда

Значение	Результат
Длина реализации программы	76
Словарь программы	33
Длина программы	133.7
Объем программы	383.04
Потенциальный объем программы	8
Теоретический уровень программы	0.002
Сложность программы	20.4
Работа программирования	191 520

Все полученные показатели программы находятся в пределах нормы и удовлетворяют требованиям по метрикам программного кода: длина реализации меньше длины программы, следовательно, несовершенства в программе отсутствуют; теоретический уровень программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема; возможности языка программирования использованы на достаточном уровне.

4.3.2 Метрика Чепина

Расчет метрики Чепина будет производиться над компонентом, отвечающим за анализ продуктивности пользователя, исходный код данного модуля представлен в листинге Д.5 приложения Д.

Для расчета метрики Чепина берется все множество переменных, составляющих список ввода-вывода, а затем разбивается на 4 функциональные группы:

– вводимые переменные для расчетов и для обеспечения вывода (P);

– модифицируемые внутри программы переменные (M);

– переменные, участвующие в управлении работой программного модуля (C);

– не используемые в программе переменные (T).

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе. Метрика Чепина:

$$Q = a1 \times P + a2 \times M + a3 \times C + a4 \times T,$$

где a1, a2, a3, a4 – весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный 3, имеет функциональная группа C, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: a1=1, a2=2, a4=0.5. Весовой коэффициент группы T не равен 0, поскольку «паразитные» переменные не увеличивают сложность потока данных программы, но иногда затрудняют ее понимание.

В модуле, отвечающем за анализ продуктивности пользователя было выделено 12 вводимых переменных, 8 модифицируемых внутри, 1 управляющая переменная и не оказалось ни одной «паразитной» переменной. Таким образом, формула подсчета метрики Чепина для данного модуля:

$$Q = 12 + 2 \times 8 + 3 \times 1 + 0.5 \times 0 = 31.$$

Данный результат вычисления метрики Чепина указывает на то, что рассматриваемый компонент имеет относительно небольшой размер и невысокую сложность исходного кода.

4.4 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета являются критически важными этапами для успешной реализации программного продукта [19].

Для управления проектом была выбрана программа Microsoft Office Project 2007, в которой реализован функционал для планирования, отслеживания и контроля над ходом выполнения различных проектов.

После занесения данных об этапах создания приложения в программу

было выполнено планирование разработки программных средств, а также была произведена оценка бюджета. Для визуального отображения проведения работ используется диаграмма Ганта, отображающая вехи разработки программного обеспечения [20].

Сгенерированная диаграмма Ганта, представлена на рисунке В.10 в приложении В.

При планировании разработки было выделено 44 задачи. За дату начала работ было взято 8 апреля 2024. Согласно полученной диаграмме окончание работ по проекту запланировано на 13 июня.

Для оценки занятости трудовых ресурсов программой был сгенерирован отчет, представленный на рисунке В.11 в приложении В.

По данным отчета об использовании трудовых ресурсах можно сделать вывод, что общая занятость трудовых ресурсов составляет 572 часа с 08.04.2024 по 13.06.2024 с учетом пятидневной рабочей недели.

Для оценки бюджета программой был сгенерирован отчет по общим затратам по проекту. Отчет представлен на рисунке В.12 в приложении В.

По отчету об общих затратах проекта можно сделать вывод, что для разработки программных средств для осознанного планирования жизни потребуется 155700 рублей.

Заключение

В рамках выпускной квалификационной работы спроектированы и разработаны программные средства для осознанного планирования жизни.

Клиентская часть была разработана с использованием библиотеки React на языке программирования JavaScript, серверная часть была разработана как три микросервиса, разработанных на языке программирования C#, взаимодействие между микросервисами и с клиентской частью приложения было реализовано посредством Rest API.

На стадии анализа требований проведен анализ предметной области, сравнительный анализ приложений аналогичного назначения, а также выявлены функциональные и нефункциональные требования для разработки нового программного обеспечения. изучение структурного разделения деятельности человека. Кроме того, на данном этапе изучено структурное разделение деятельности человека.

На стадии проектирования раработана архитектура программного обеспечения, выделены классы разрабатываемого приложения, а также выявлены требования к интерфейсу и разработаны концептуальная и логические модели базы данных.

Разработка программных средств для осознанного планирования жизни, включала в себя реализацию базы данных в PostgreSQL, трех микросервисов в среде разработки JetBrains с использованием ASP.NET Core и одного клиентского приложения в среде разработки Visual Studio Code с использованием Node, сборщика Webpack и библиотеки React.

Также был проведен контроль качества разработанного web-приложения путем функционального и модульного тестирования. Выполнен подсчет метрик Холстеда и Чепина. Модульное тестирование производилось с использованием фреймворка Jest, использующегося для тестирования JavaScript кода. В результате тестирования выявлена корректность работы программы и ее

соответствие функциональным требованиям, подсчет метрик показал достаточный уровень разработанного приложения.

Произведено планирование разработки, а также функционально-стоимостной анализ разработки программных средств. Затраты на проект составили 155700 рублей. Начало работ было определено как 8 апреля, а окончание работ по проекту запланировано на 13 июня текущего года. Затраты трудовых ресурсов составили 572 часа.

Таким образом, в результате выполнения выпускной квалификационной работы разработаны программные средства для осознанного планирования жизни, позволяющие структурировать задачи пользователя по сферам жизни, направлениям и проектам, назначать цели, которые требуется достичь пользователю, отслеживать время выполнения задач, добавлять план деятельности, а также анализировать эффективность пользователя.

Программное обеспечение отвечает требованиям качества, надежности и производительности, соответствует современным тенденциям в области программной инженерии. Оно может быть использовано для планирования учебного процесса студентами, планирования деятельности работающих людей.

Список использованных источников

1. Калинин, С.И. Тайм-менеджмент. Практикум по управлению временем / С.И. Калинин. – СПб. : Речь, 2006. – 373 с.

2. Дюжева, А.Н. Осознанное планирование жизни с использованием специализированных программных средств / А.Н. Дюжева, Чапаев И.А., И.Ю. Балашова // Новые информационные технологии и системы: материалы XX международной науч.-техн. конф. – Пенза: Изд-во Пензенского государственного университета, 2023. – С. 218 – 221.

3. Дюжева, А.Н. Повышение продуктивности деятельности студентов вузов с использованием программных средств для осознанного планирования жизни / А.Н. Дюжева, Чапаев И.А., И.Ю. Балашова // Иновационное образовательное пространство педагогического вуза. Проблемы и перспективы: материалы всероссийской науч.-пр. конф. – Саранск: Изд-во Мордовского государсвенного университета, 2023. – С. 179 – 183.

4. Ньюман, С. Строим микросервисы: проектирование масштабируемых систем / С. Ньюман. – М.: ДМК Пресс, 2016. – 384 с.

5. Draw.io // Skillfactory Media. [Электронный ресурс]. URL: https://blog.skillfactory.ru/draw-io/ (дата обращения: 13.06.2024).

6. REST API // Skillfactory Media. [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/rest-api/ (дата обращения: 13.06.2024).

7. Ричардсон, К. Микросервисные шаблоны: Проектирование и создание микросервисов / К. Ричардсон, М. Фаулер. – СПб.: Питер, 2021. – 528 с.

8. Рудь В. Концептуальное логическое и физическое моделирование данных // Информационный портал компании «Марк Аврелий» [Электронный ресурс]. URL: https://marcus-aurelius.ru/articles/Data_modeling.html (дата обращения 03.06.2024).

9. Фримен, А. Pro React 16: разрабатывайте современные приложения / А. Фримен, Э. Гартшорн. – СПб.: БХВ, 2019. – 450 с.

10. Рамакришнан, Р. Базы данных от простого к сложному / Р. Рамакришнан, Й. Герк. – М.: Вильямс, 2019. – 672 с.

11. Васильева А. Нотации баз данных // Архив «О Visio по-русски» за 2011 – 2018 [Электронный ресурс]. URL: https://rus-visio-blog-archive.github.io/2012/12/17/487.htm (дата обращения 03.06.2024).

12. Бэнкс, А. React и Redux: Создание приложений нового поколения на JavaScript / А. Бэнкс, Е. Порселло. – М.: Питер, 2020. – 320 с.

13. ТОП-10 платформ для разработки программного обеспечения // otzyvmarketing. [Электронный ресурс]. URL: https://otzyvmarketing.ru/articles/platformy-dlya-razrabotki-programmnogo-obespecheniya/ (дата обращения: 13.06.2024).

14. Крауч, Р. Тестирование ПО. Основы, методы и термины / Р. Крауч, В.Д. Влиет. – СПб.: Питер, 2003. – 256 с.

15. Куликов, С.С. Тестирование программного обеспечения. Базовый курс / С.С. Куликов – Минск: EPAM Systems, 2021. – 300 с.

16. Программный код и его метрики // Тематический блог «Хабр». [Электронный ресурс]. URL: https://habr.com/ru/companies/intel/articles/106082/ (дата обращения 13.06.2024).

17. Метрики как средства управления качеством // OSP [Электронный ресурс] URL: https://www.osp.ru/os/2009/08/10748698 (Дата обращения: 26.05.2024).

18. Тестирование React приложения // Jest. [Электронный ресурс]. URL: https://jestjs.io/ru/docs/tutorial-react (дата обращения: 13.06.2024).

19. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. –102 с.

20. Создаем диаграмму Ганта // UNISENDER [Электронный ресурс] URL: https://www.unisender.com/ru/blog/chto-takoe-diagramma-ganta-i-kak-ee-sdelat/ (Дата обращения: 13.06.2024).

Приложение А

(справочное)

Глоссарий

БД (База данных) – это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

REST (Representational State Transfer) – это архитектурная концепция, определяющая набор ограничений для построения масштабируемых распределенных систем. Основные принципами REST является ресурсно-ориентированная архитектура, использование стандартных HTTP-методов для манипуляции ресурсами, передача данных без сохранения состояния на сервере, унифицированный интерфейс доступа.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

PostgreSQL – это открытая и мощная реляционная система управления данными, основанная на объектно-ориентированной парадигме, обеспечивающая высокую производительность, надежность и масштабируемость.

Фреймворк – программная платформа, задающая архитектурные принципы и общие правила разработки приложений в определенной предметной области.

Web-браузер – прикладное программное обеспечение для поиска и просмотра web-страниц, а также управления web-приложениями;

Web-приложение –это клиент-серверное программное решение, доступ к которому осуществляется через сетевой браузер посредством передачи данных по протоколу HTTP, размещается на удаленном сервере

JSON – JavaScript Object Notation, легковесный текстовый формат обмена данными, полностью независимый от языка реализации.

.NET – это платформа для разработчиков, состоящая из инструментов, языков программирования и библиотек для создания множества различных типов приложений. ASP.NET Core расширяет платформу разработчика .NET инструментами и библиотеками, специально предназначенными для создания веб-приложений.

MVC – это архитектурный паттерн, предписывающий разделение программного кода на три взаимосвязанных компонента: модель данных, представление для визуализации информации и контроллер для обработки пользовательских запросов.

Приложение Б

(рекомендуемое)

Атрибуты сущностей базы данных

Таблица Б.1 – Атрибуты отношения «DbTask»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор задачи	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f
ProjectID	Идентификатор проекта, к которому привязана задача	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
Created Data	Дата создания задачи	timestamp	2		нет		01.01.2023
Title	Название задачи	varchar	255	а-Я a-Z 0-9	нет		Сверстать страницу
Description	Содержание задачи	text	не ограничена	а-Я a-Z 0-9	нет		Нужно сверстать страницу с изображениями автомобилей
StartExecutionDate	Дата начала выполнения задачи	timestamp	2	01.01.1950-01.01.2023	да		01.01.2023
DeadlineDate	Дата завершения выполнения задачи	timestamp	2	01.01.1950-01.01.2023	да		01.01.2023

Таблица Б.2 – Атрибуты отношения «DbProject»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор проекта	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f
ActionDirectionID	Идентификатор направления, к которому привязан проект	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
Created Date	Дата создания проекта	timestamp	2	01.01.1950-01.01.2023	нет		01.01.2023

Продолжение таблицы Б.2

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Title	Название проекта	varchar	255	а-Я a-Z 0-9	нет		Создание сайта
Description	Содержание проекта	text	не ограничена	а-Я a-Z 0-9	нет		Нужно создать сайт
StartExecutionDate	Дата начала выполнения проекта	timestamp	2	01.01.1950-01.01.2023	да		01.01.2023
DeadlineDate	Дата завершения выполнения проекта	timestamp	2	01.01.1950-01.01.2023	да		01.01.2023

Таблица Б.3 – Атрибуты отношения «DbRealExecutionTime»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор затраченного времени	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f
TaskID	Идентификатор задачи, к которому привязано затраченное время	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
StartDate	Начало выполнения какой-либо задачи	timestamp	2	01.01.1950-01.01.2023	нет		01.01.2023
EndDate	Завершение выполнения какой-либо задачи	timestamp	2	01.01.1950-01.01.2023	нет		01.01.2023

Таблица Б.4 – Атрибуты отношения «DbLifeSphere»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор сферы жизни	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f

Продолжение таблицы Б.4

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
UserID	Идентификатор пользователя, к которому привязана сфера жизни	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
CreatedDate	Дата создания сферы жизни	timestamp	2		нет		01.01.2023
Title	Название сферы жизни	varchar	255	01.01.1950-01.01.2023	нет		Финансы
Description	Описание сферы жизни	text	не ограничена	а-Я а-Z 0-9	нет		К данной сфере жизни принадлежит все, что может приносить доход

Таблица Б.5 – Атрибуты отношения «DbGoal»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор цели	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f
ActionDirectionID	Идентификатор направления, к которому привязана цель	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
CreatedDate	Дата начала существования цели	timestamp	2	01.01.1950-01.01.2023	нет		01.01.2023
DeadlineDate	Дата осуществления поставленной цели	timestamp	2	01.01.1950-01.01.2023	да		01.01.2023
Title	Название цели	varchar	255	01.01.1950-01.01.2023	нет		Накопить на телефон

Таблица Б.6 – Атрибуты отношения «DbProject_DbGoal»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ProjectID	Идентификатор проекта	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
GoalID	Идентификатор цели	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f

Таблица Б.7 – Атрибуты отношения «DbActionDirection»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор направления	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f
Created Data	Дата создания направления жизни	timestamp	2		нет		01.01.2023
Title	Название сферы жизни	varchar	255	01.01.1950-01.01.2023	нет		Университет
Description	Описание направления	text	не ограничена	а-Я a-Z 0-9	нет		Направлено на учебу

Таблица Б.8 – Атрибуты отношения «DbActionDirectionFromLifeSphere»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
LifeSphereID	Идентификатор сферы жизни	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
ActionDirectionID	Идентификатор направления	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f

Продолжение таблицы Б.8

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ParentActionDirectionID	Идентификатор родительского направления	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f

Таблица Б.9 – Атрибуты отношения «DbPlanningTime»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор запланированного времени	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	2fddff21-d063-490c-b5c1-cbdf1644492f
ActionDirectionID	Идентификатор направления, к которому привязано запланированное время	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	2fddff21-d063-490c-b5c1-cbdf1644492f
EndDate	Дата окончания действия плана	timestamp	2		нет		01.01.2023

Таблица Б.10 – Атрибуты отношения «DbUser»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор пользователя	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	7fddff71-d063-490c-b5c1-cbdf1644497f
Login	Логин пользователя	varchar	255	01.01.1950-01.01.2023	нет	FK	7fddff71-d063-490c-b5c1-cbdf1644497f
PasswordHash	Пароль пользователя	varchar	255	01.01.1950-01.01.2023	нет		Sofia45

Продолжение таблицы Б.10

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Email	Электронная почта пользователя	varchar	255	01.01.1950-01.01.2023	нет		6548233Shh
Name	Имя пользователя	varchar	255	01.01.1950-01.01.2023	нет		sofi@mail.ru
Birthday	Дата дня рождения пользователя	timestamp	2	01.01.1950-01.01.2023	нет		12.02.2003
Height	Рост пользователя	Int4	4	1-9999999	нет		165
Weight	Вес пользователя	Int4	4	1-9999999	нет		55

Таблица Б.11 – Атрибуты отношения «DbFileInfo»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор информации о файле	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	7fddff71-d063-490c-b5c1-cbdf1644497f
UserID	Идентификатор пользователя, к которому привязана информация о файле	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	да	FK	7fddff71-d063-490c-b5c1-cbdf1644497f
FileName	Название файла	varchar	255	а-Я a-Z 0-9	да		File345_palnning
CreatedDate	Дата создания файла	timestamp	2	01.01.1950-01.01.2023	да		01.01.7073

Таблица Б.12 – Атрибуты отношения «DbFileData»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор данных файла	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	7fddff71-d063-490c-b5c1-cbdf1644497f
FileInfoID	Идентификатор информации файла, к которому привязаны данные файла	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	7fddff71-d063-490c-b5c1-cbdf1644497f
Content	Содержимое файла	Byte a	не ограничена		нет		байты

Таблица Б.13 – Атрибуты отношения «DBFileLink»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор ссылки на файл	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	7fddff71-d063-490c-b5c1-cbdf1644497f
FileInfoID	Идентификатор информации файла, к которому привязана ссылка	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	FK	7fddff71-d063-490c-b5c1-cbdf1644497f
CreatedDate	Дата создания ссылки	timestamp	2		нет		01.01.7073

Таблица Б.14 – Атрибуты отношения «DbReportPeriodSettings»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID	Уникальный идентификатор настройки формирования файла	uuid	7	00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz	нет	PK	7fddff71-d063-490c-b5c1-cbdf1644497f
PlanningPeriod	Период, в течение которого собирается информация	varchar	255	a-z	нет		every week

Приложение В

(рекомендуемое)

Графическая часть

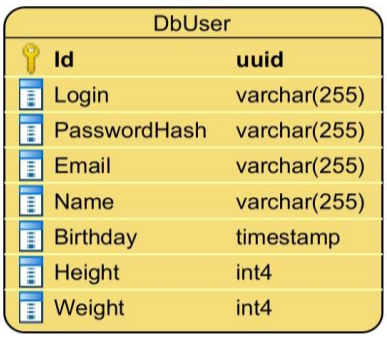

Рисунок В.1 – Логическая модель базы данных «Авторизация»

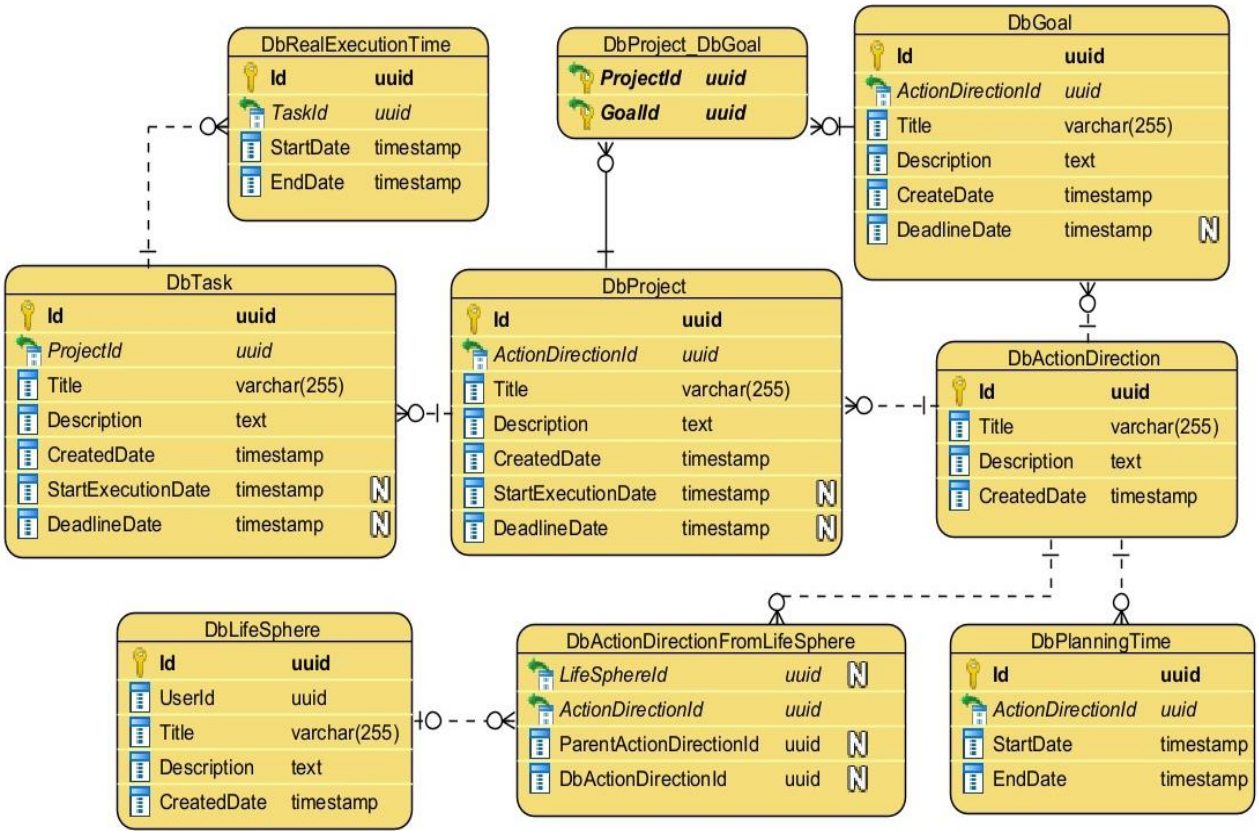

Рисунок В.2 – Логическая модель базы данных «Планирование»

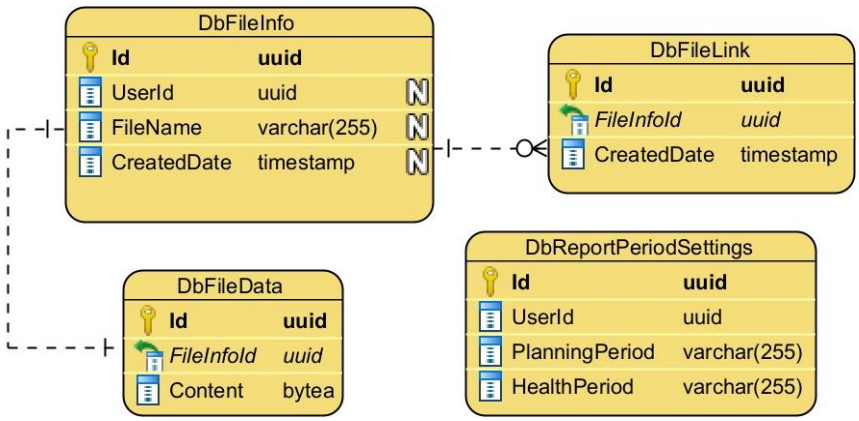

Рисунок В.3 – Логическая модель базы данных «Отчеты»

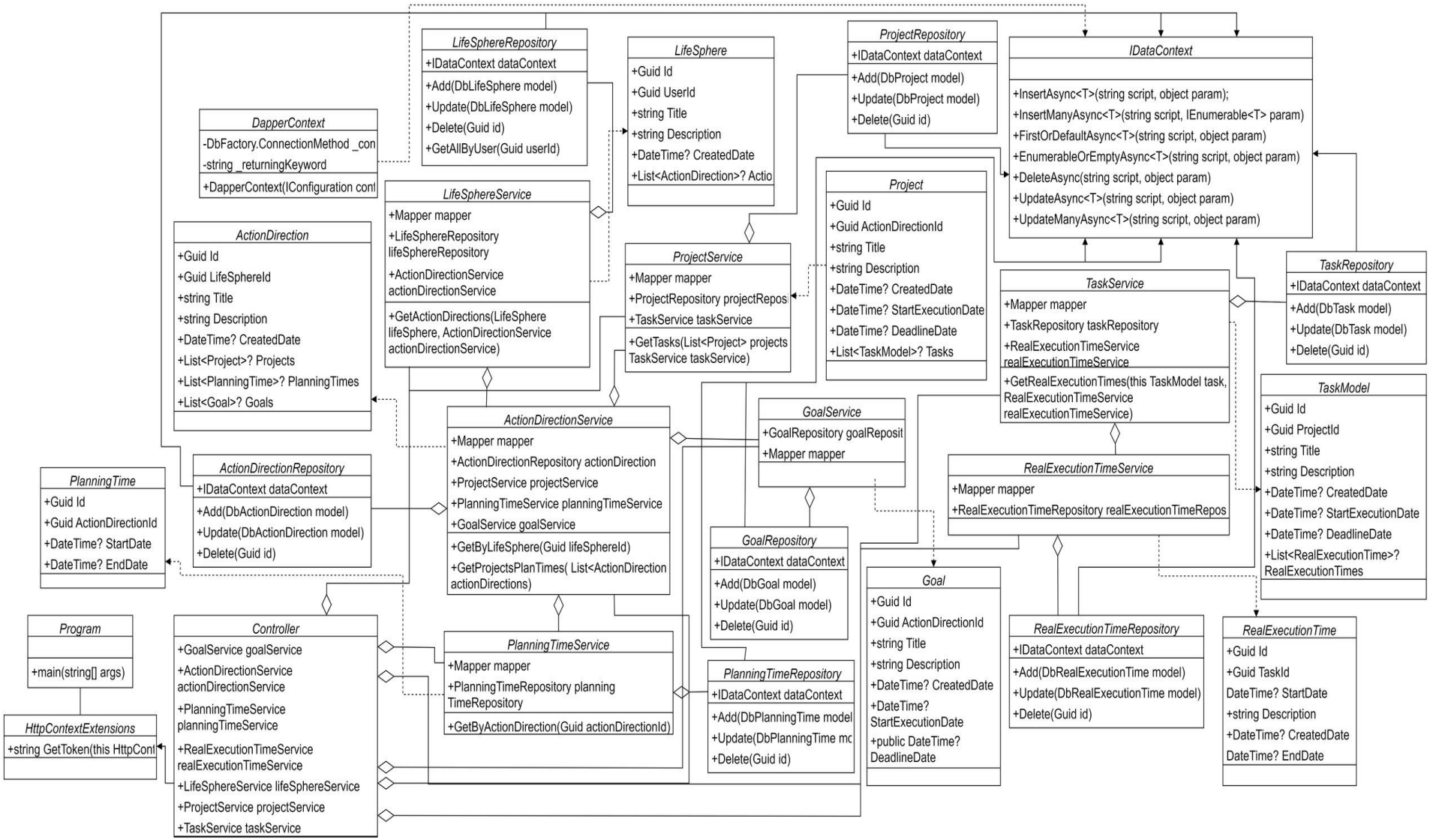

Рисунок В.4 – Диаграмма классов микросервиса «Планирование»

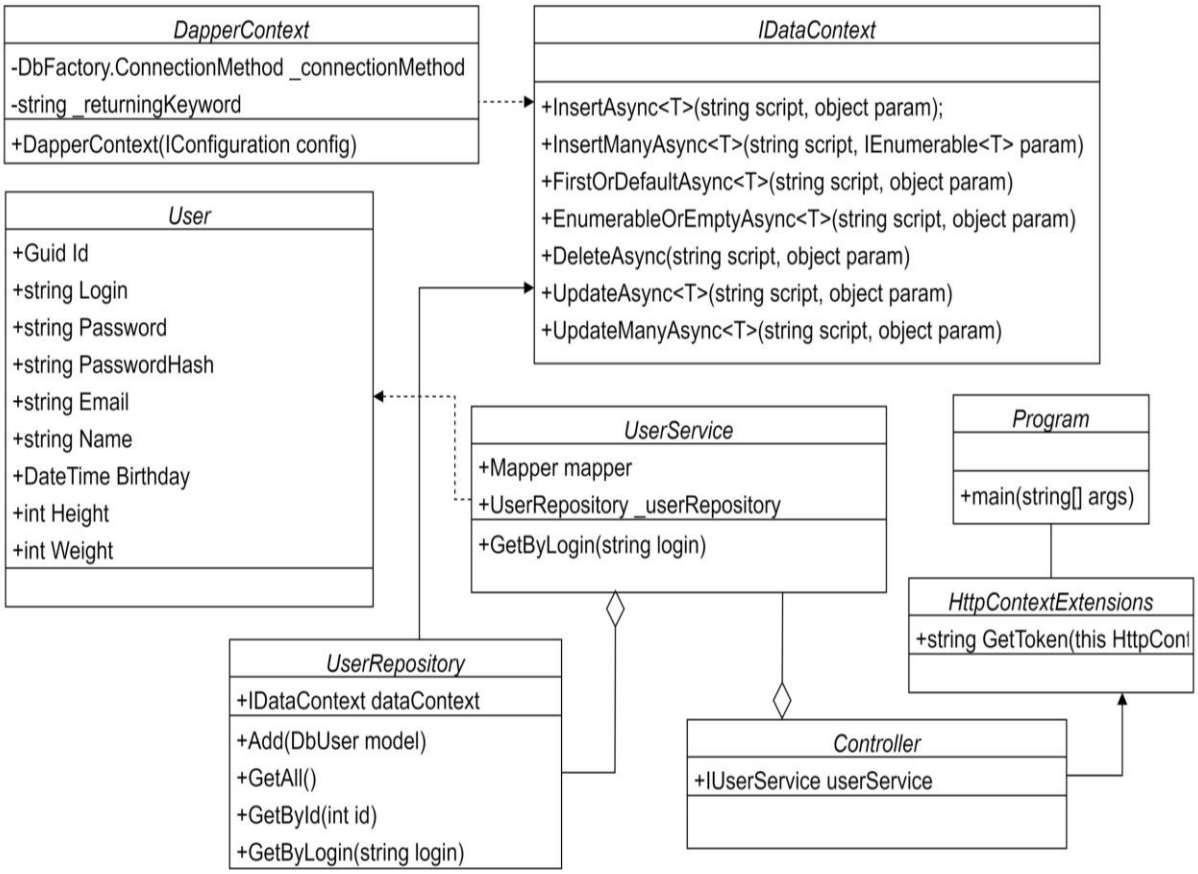

Рисунок В.5 – Диаграмма классов микросервиса «Авторизация»

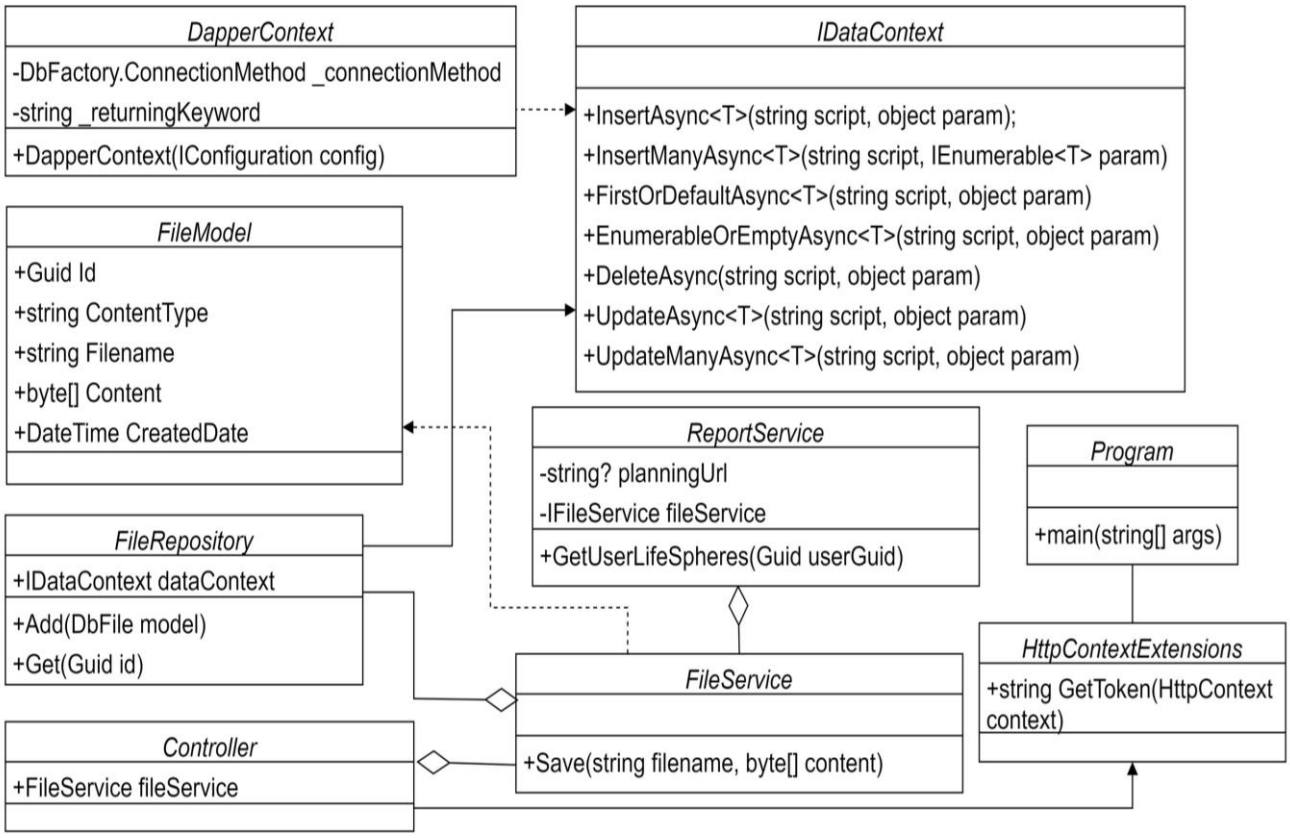

Рисунок В.6 – Диаграмма классов микросервиса «Отчеты»

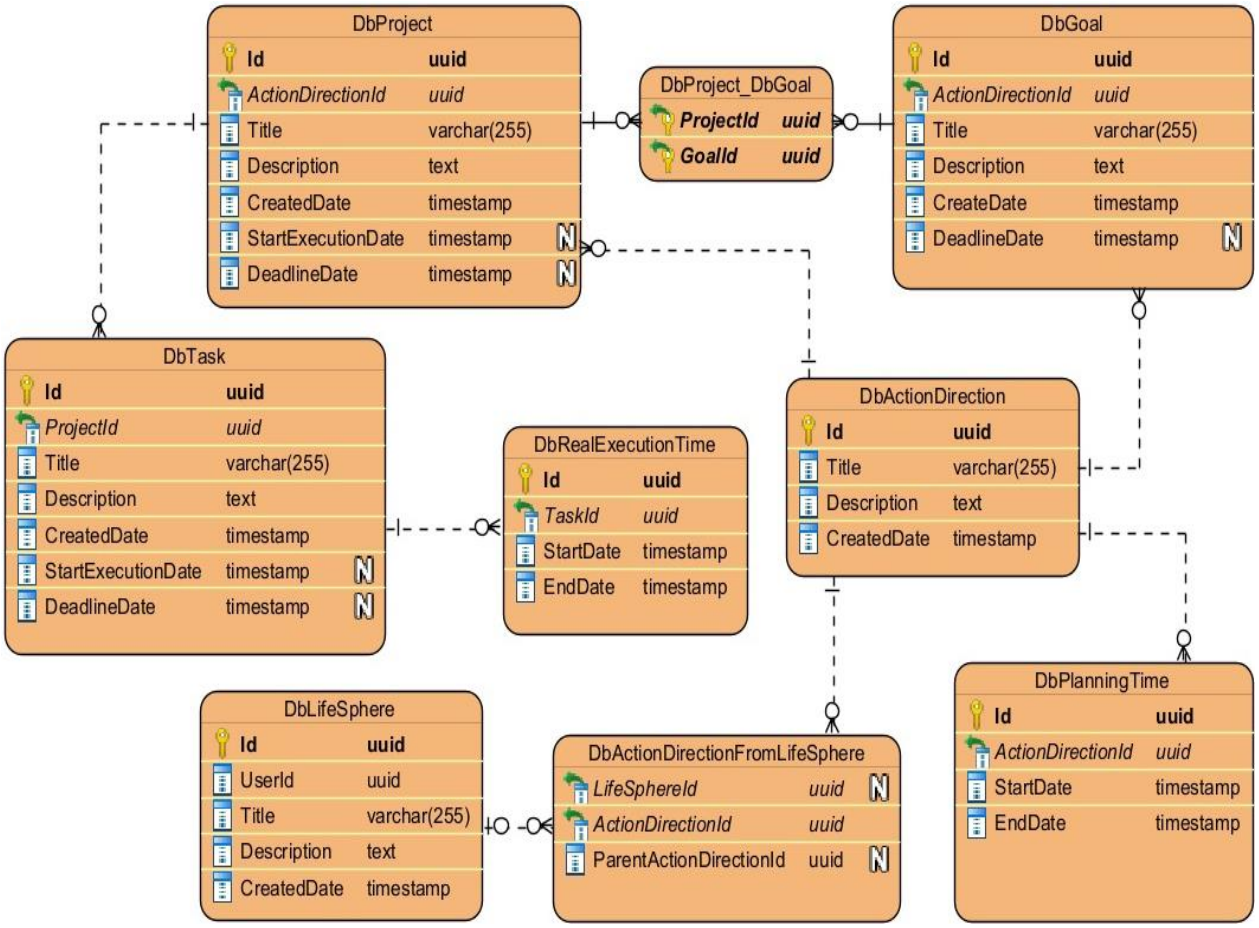

Рисунок В.7 – Физическая модель базы данных «Планирование»

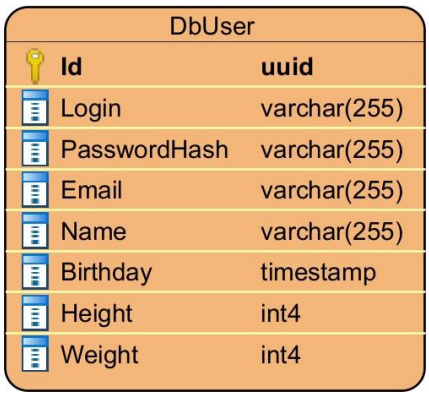

Рисунок В.8 – Физическая модель базы данных «Авторизация»

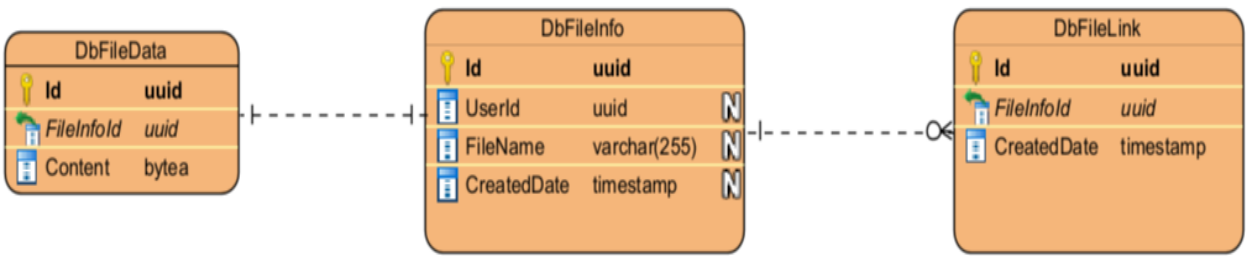

Рисунок В.9 – Физическая модель базы данных «Авторизация»

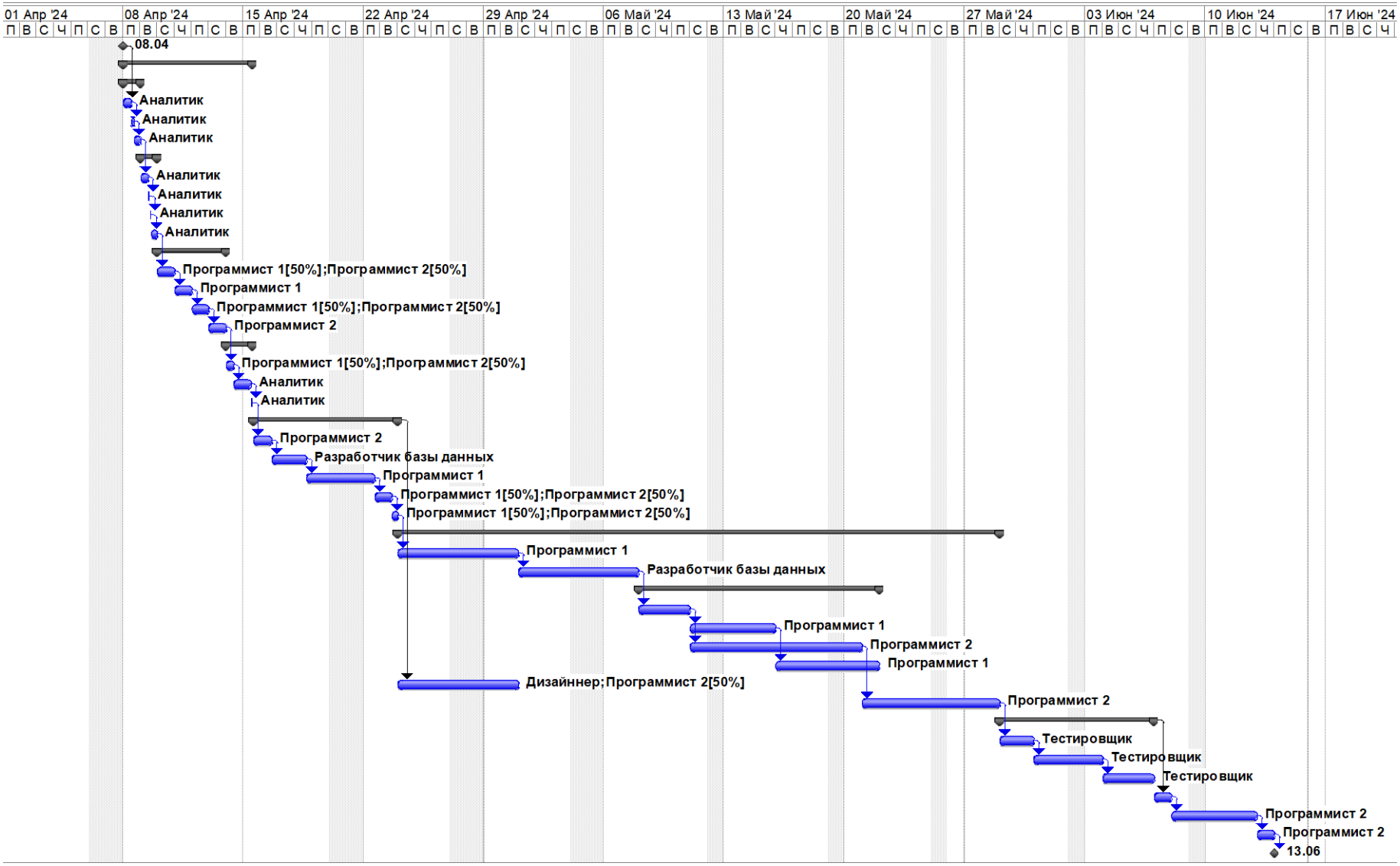

Рисунок В.10 – Диаграмма Ганта задач с распределением ресурсов

Использование ресурсов от Вс 16.06.24
Проект1

	Итого
Аналитик	26 ч
Анализ предметной области	2,63 ч
Анализ свойств объектов управления и информационных объектов предметной области	2,72 ч
Анализ процессов управления и ведения данных в предметной области	2,63 ч
Цели и задачи сравнительного анализа	2 ч
Выбор приложений-аналогов для анализа	2 ч
Обзор приложений-аналогов	2 ч
Результаты сравнительного анализа	2 ч
Планирование процесса разработки приложения	8 ч
Оценка временных и финансовых затрат на разработку	2 ч
Программист 1	150 ч
Цели разработки и назначение приложения	4 ч
Перечень функций, реализуемых в приложении	8 ч
Требования к организации серверного и клиентской части приложения	4 ч
Выбор нотации и программных средств планирования и оценки процесса разработки	1 ч
Обощенная архитектура программных средств для осознанного планирования жизни	16 ч
Требования к аппаратно-программному обеспечению клиентской части приложения и серверной приложения	4 ч
Обоснование выбора технологий и средств разработки приложения	1 ч
Разработка архитектуры приложения	40 ч
Разработка программного обеспечения микросервиса "Авторизация"	32 ч
Разработка программного обеспечения микросервиса "Отчеты"	40 ч
Программист 2	236 ч
Цели разработки и назначение приложения	4 ч
Требования к организации серверного и клиентской части приложения	4 ч
Постановка требований к интерфейсу программных средств	8 ч
Выбор нотации и программных средств планирования и оценки процесса разработки	1 ч
Моделирование функциональных возможностей приложения	10 ч
Требования к аппаратно-программному обеспечению клиентской части приложения и серверной приложения	4 ч
Обоснование выбора технологий и средств разработки приложения	1 ч
Разработка API методов	24 ч
Разработка программного обеспечения микросервиса "Планирование"	64 ч
Разработка пользовательского интерфейса	20 ч
Разработка клиентской части приложения	56 ч
Оформление и сбор документации	32 ч
Ввод в эксплуатацию	8 ч
Дизайннер	40 ч
Разработка пользовательского интерфейса	40 ч
Разработчик базы данных	56 ч
Разработка единой концептуальной модели данных	16 ч
Разработка физической модели базы данных	40 ч
Тестировщик	64 ч
Планирование процесса контроля качества программного обеспечения	16 ч
Тестирование программных средств	24 ч
Оценка качества программного кода	24 ч
Интернет	
Анализ предметной области и постановка задачи на разработку серверного приложения	
Проектирование программных средств	
Разработка программных средств для осознонного планирования жизни	
Контроль качества программного обеспечения	
Компьютеры	
Итого	572 ч

Рисунок В.11 – Отчет об использовании ресурсов

Отчет о бюджете от Вс 16.06.24
Проект1

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты
33	Разработка программного обеспечен	0,00 ₽	Пропорциональное	19 200,00 ₽
36	Разработка клиентской части прилож	0,00 ₽	Пропорциональное	16 800,00 ₽
35	Разработка пользовательского интер	0,00 ₽	Пропорциональное	14 000,00 ₽
28	Разработка архитектуры приложения	0,00 ₽	Пропорциональное	12 800,00 ₽
34	Разработка программного обеспечен	0,00 ₽	Пропорциональное	12 800,00 ₽
29	Разработка физической модели базы	0,00 ₽	Пропорциональное	12 000,00 ₽
32	Разработка программного обеспечен	0,00 ₽	Пропорциональное	10 240,00 ₽
42	Оформление и сбор документации	0,00 ₽	Пропорциональное	9 600,00 ₽
24	Обощенная архитектура программны	0,00 ₽	Пропорциональное	5 120,00 ₽
23	Разработка единой концептуальной м	0,00 ₽	Пропорциональное	4 800,00 ₽
39	Тестирование программных средств	0,00 ₽	Пропорциональное	4 800,00 ₽
40	Оценка качества программного кода	0,00 ₽	Пропорциональное	4 800,00 ₽
38	Планирование процесса контроля ка	0,00 ₽	Пропорциональное	3 200,00 ₽
22	Моделирование функциональных во	0,00 ₽	Пропорциональное	3 000,00 ₽
14	Перечень функций, реализуемых в п	0,00 ₽	Пропорциональное	2 560,00 ₽
13	Цели разработки и назначение прил	0,00 ₽	Пропорциональное	2 480,00 ₽
15	Требования к организации серверно	0,00 ₽	Пропорциональное	2 480,00 ₽
25	Требования к аппаратно-программно	0,00 ₽	Пропорциональное	2 480,00 ₽
16	Постановка требований к интерфей	0,00 ₽	Пропорциональное	2 400,00 ₽
43	Ввод в эксплуатацию	0,00 ₽	Пропорциональное	2 400,00 ₽
19	Планирование процесса разработки	0,00 ₽	Пропорциональное	2 000,00 ₽
5	Анализ свойств объектов управлени	0,00 ₽	Пропорциональное	680,00 ₽
4	Анализ предметной области	0,00 ₽	Пропорциональное	660,00 ₽
6	Анализ процессов управления и вед	0,00 ₽	Пропорциональное	660,00 ₽
18	Выбор нотации и программных сред	0,00 ₽	Пропорциональное	620,00 ₽
26	Обоснование выбора технологий и с	0,00 ₽	Пропорциональное	620,00 ₽
8	Цели и задачи сравнительного анал	0,00 ₽	Пропорциональное	500,00 ₽
9	Выбор приложений-аналогов для ан	0,00 ₽	Пропорциональное	500,00 ₽
10	Обзор приложений-аналогов	0,00 ₽	Пропорциональное	500,00 ₽
11	Результаты сравнительного анализа	0,00 ₽	Пропорциональное	500,00 ₽
20	Оценка временных и финансовых за	0,00 ₽	Пропорциональное	500,00 ₽
1	Начало проекта	0,00 ₽	Пропорциональное	0,00 ₽
31	Разработка API методов	0,00 ₽	Пропорциональное	0,00 ₽
41	Сдача проекта	0,00 ₽	Пропорциональное	0,00 ₽
44	Конец проекта	0,00 ₽	Пропорциональное	0,00 ₽
		0,00 ₽		**155 700,00 ₽**

Рисунок В.12 – Отчет об общих затратах проекта

Приложение Г

(рекомендуемое)

Листинг серверной части приложения

Листинг Г.1 – Файл ActionDirectionService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class ActionDirectionService(
    IMapper mapper,
    IActionDirectionRepository actionDirectionRepository,
    IProjectService projectService,
    IPlanningTimeService planningTimeService,
    IGoalService goalService) : IActionDirectionService
{
    public async Task<ActionDirection> Add(ActionDirection model)
    {
        return (await
actionDirectionRepository.Add(model.Map<DbActionDirection>(mapper))).Map<
ActionDirection>(mapper);
    }

    public async Task<ActionDirection> Update(ActionDirection model)
    {
        return (await
actionDirectionRepository.Update(model.Map<DbActionDirection>(mapper))).M
ap<ActionDirection>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await actionDirectionRepository.Delete(id);
    }

    public async Task<List<ActionDirection>> GetByIds(IEnumerable<Guid>
ids)
    {
        var res = (await
actionDirectionRepository.GetByIds(ids)).MapToList<ActionDirection>(mappe
r);

        await res.GetProjectsPlanningTimesGoals(projectService,
planningTimeService, goalService);

        return res;
    }

    public async Task<ActionDirection> GetById(Guid id)
    {
        var res = (await actionDirectionRepository.GetByIds(new
List<Guid> {id})).Map<ActionDirection>(mapper);

        await res.GetProjectsPlanningTimesGoals(projectService,
planningTimeService, goalService);
```

```
        return res;
    }

    public async Task<List<ActionDirection>> GetByLifeSphere(Guid
lifeSphereId)
    {
        var res = (await actionDirectionRepository.GetByLifeSpheres(new
List<Guid> {lifeSphereId})).MapToList<ActionDirection>(mapper);
        await res.GetProjectsPlanningTimesGoals(projectService,
planningTimeService, goalService);

        return res;
    }

    public async Task<List<ActionDirection>> GetByLifeSpheres(List<Guid>
lifeSphereIds)
    {
        var res = (await
actionDirectionRepository.GetByLifeSpheres(lifeSphereIds)).MapToList<Acti
onDirection>(mapper);
        await res.GetProjectsPlanningTimesGoals(projectService,
planningTimeService, goalService);

        return res;
    }
}

public static class ActionDirectionExtensions
{
    public static async Task<List<ActionDirection>>
GetProjectsPlanningTimesGoals(this List<ActionDirection>
actionDirections,
        IProjectService projectService,
        IPlanningTimeService planningTimeService,
        IGoalService goalService)
    {
        var actionDirectionIds = actionDirections.Select(p =>
p.Id).ToList();

        var projectsTask =
projectService.GetByActionDirections(actionDirectionIds);
        var planningTimesTask =
planningTimeService.GetByActionDirections(actionDirectionIds);
        var goalsTask =
goalService.GetGetByActionDirections(actionDirectionIds);

        var projects = await projectsTask;
        var planningTimes = await planningTimesTask;
        var goals = await goalsTask;

        var projectsGroupByActionDirection = projects
            .GroupBy(t => t.ActionDirectionId)
            .ToDictionary(t => t.Key, t => t.ToList());
        var planningTimesGroupByActionDirection = planningTimes
            .GroupBy(t => t.ActionDirectionId)
            .ToDictionary(t => t.Key, t => t.ToList());
        var goalsGroupByActionDirection = goals
            .GroupBy(t => t.ActionDirectionId)
```

```
            .ToDictionary(t => t.Key, t => t.ToList());

        actionDirections.ForEach(p =>
        {
            p.Projects =
projectsGroupByActionDirection.GetValueOrDefault(p.Id)?.ToList() ?? new
List<Project>();
            p.Goals =
goalsGroupByActionDirection.GetValueOrDefault(p.Id)?.ToList() ?? new
List<Goal>();
            p.PlanningTimes =
planningTimesGroupByActionDirection.GetValueOrDefault(p.Id)?.ToList() ??
new List<PlanningTime>();
        });

        return actionDirections;
    }

    public static async Task<ActionDirection>
GetProjectsPlanningTimesGoals(this ActionDirection actionDirection,
        IProjectService projectService,
        IPlanningTimeService planningTimeService,
        IGoalService goalService)
    {
        var projectsTask =
projectService.GetByActionDirection(actionDirection.Id);
        var planningTimesTask =
planningTimeService.GetByActionDirection(actionDirection.Id);
        var goalsTask =
goalService.GetGetByActionDirection(actionDirection.Id);

        var projects = await projectsTask;
        var planningTimes = await planningTimesTask;
        var goals = await goalsTask;

        actionDirection.Projects = projects;
        actionDirection.PlanningTimes = planningTimes.ToList() ?? new
List<PlanningTime>();
        actionDirection.Goals = goals;

        return actionDirection;
    }
}
```

Листинг Г.2 – Файл GoalService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class GoalService(IMapper mapper, IGoalRepository goalRepository)
: IGoalService
{
```

```
    public async Task<Goal> Add(Goal model)
    {
        return (await
goalRepository.Add(model.Map<DbGoal>(mapper))).Map<Goal>(mapper);
    }

    public async Task<Goal> Update(Goal model)
    {
        return (await
goalRepository.Update(model.Map<DbGoal>(mapper))).Map<Goal>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await goalRepository.Delete(id);
    }

    public async Task<List<Goal>> GetByIds(IEnumerable<Guid> ids)
    {
        return (await
goalRepository.GetByIds(ids)).MapToList<Goal>(mapper);
    }

    public async Task<Goal> GetById(Guid id)
    {
        return (await goalRepository.GetByIds(new List<Guid>
{id})).Map<Goal>(mapper);
    }

    public async Task<List<Goal>> GetGetByActionDirection(Guid
actionDirectionId)
    {
        return (await goalRepository.GetGetByActionDirections(new
List<Guid> {actionDirectionId})).MapToList<Goal>(mapper);
    }

    public async Task<List<Goal>> GetGetByActionDirections(List<Guid>
actionDirectionIds)
    {
        return (await
goalRepository.GetGetByActionDirections(actionDirectionIds)).MapToList<Go
al>(mapper);
    }
}
```

Листинг Г.3 – Файл LifeSphereService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;
namespace Services;
public class LifeSphereService(IMapper mapper, ILifeSphereRepository
lifeSphereRepository, IActionDirectionService actionDirectionService) :
ILifeSphereService
{
```

```
    public async Task<LifeSphere> Add(LifeSphere model)
    {
        return (await
lifeSphereRepository.Add(model.Map<DbLifeSphere>(mapper))).Map<LifeSphere
>(mapper);
    }

    public async Task<LifeSphere> Update(LifeSphere model)
    {
        return (await
lifeSphereRepository.Update(model.Map<DbLifeSphere>(mapper))).Map<LifeSph
ere>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await lifeSphereRepository.Delete(id);
    }

    public async Task<List<LifeSphere>> GetByIds(IEnumerable<Guid> ids)
    {
        var res = (await
lifeSphereRepository.GetByIds(ids)).MapToList<LifeSphere>(mapper);
        await res.GetActionDirections(actionDirectionService);

        return res;
    }

    public async Task<LifeSphere> GetById(Guid id)
    {
        var res = (await lifeSphereRepository.GetByIds(new List<Guid>
{id})).Map<LifeSphere>(mapper);
        await res.GetActionDirections(actionDirectionService);

        return res;
    }

    public async Task<List<LifeSphere>> GetAllByUser(Guid id)
    {
        var res = (await
lifeSphereRepository.GetAllByUser(id)).MapToList<LifeSphere>(mapper);
        await res.GetActionDirections(actionDirectionService);

        return res;
    }
}

public static class LifeSphereExtensions
{
    public static async Task<List<LifeSphere>> GetActionDirections(this
List<LifeSphere> lifeSpheres, IActionDirectionService
actionDirectionService)
    {
        var lifeSphereIds = lifeSpheres.Select(p => p.Id).ToList();
        var actionDirections = await
actionDirectionService.GetByLifeSpheres(lifeSphereIds);
```

```
        var actionDirectionsGroupByLifeSphere =
actionDirections.GroupBy(t => t.LifeSphereId).ToDictionary(t => t.Key, t
=> t.ToList());
        lifeSpheres.ForEach(p => p.ActionDirections =
actionDirectionsGroupByLifeSphere.GetValueOrDefault(p.Id)?.ToList() ??
new List<ActionDirection>());

        return lifeSpheres;
    }

    public static async Task<LifeSphere> GetActionDirections(this
LifeSphere lifeSphere, IActionDirectionService actionDirectionService)
    {
        var actionDirections = await
actionDirectionService.GetByLifeSphere(lifeSphere.Id);
        lifeSphere.ActionDirections = actionDirections;

        return lifeSphere;
    }
}
```

Листинг Г.4 – Файл PlanningTimeService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class PlanningTimeService(IMapper mapper, IPlanningTimeRepository
planningTimeRepository) : IPlanningTimeService
{
    public async Task<PlanningTime> Add(PlanningTime model)
    {
        return (await
planningTimeRepository.Add(model.Map<DbPlanningTime>(mapper))).Map<Planni
ngTime>(mapper);
    }

    public async Task<PlanningTime> Update(PlanningTime model)
    {
        return (await
planningTimeRepository.Update(model.Map<DbPlanningTime>(mapper))).Map<Pla
nningTime>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await planningTimeRepository.Delete(id);
    }

    public async Task<List<PlanningTime>> GetByIds(IEnumerable<Guid> ids)
    {
        return (await
planningTimeRepository.GetByIds(ids)).MapToList<PlanningTime>(mapper);
```

```
    }

    public async Task<PlanningTime> GetById(Guid id)
    {
        return (await planningTimeRepository.GetByIds(new List<Guid>
{id})).Map<PlanningTime>(mapper);
    }

    public async Task<List<PlanningTime>>
GetByActionDirections(List<Guid> actionDirectionIds)
    {
        return (await
planningTimeRepository.GetByActionDirections(actionDirectionIds)).MapToLi
st<PlanningTime>(mapper);
    }

    public async Task<List<PlanningTime>> GetByActionDirection(Guid
actionDirectionId)
    {
        return await GetByActionDirections(new List<Guid>
{actionDirectionId});
    }
}
```

Листинг Г.5 – Файл RealExecutionTimeService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class RealExecutionTimeService(IMapper mapper,
IRealExecutionTimeRepository realExecutionTimeRepository) :
IRealExecutionTimeService
{
    public async Task<RealExecutionTime> Add(RealExecutionTime model)
    {
        return (await
realExecutionTimeRepository.Add(model.Map<DbRealExecutionTime>(mapper))).
Map<RealExecutionTime>(mapper);
    }

    public async Task<RealExecutionTime> Update(RealExecutionTime model)
    {
        return (await
realExecutionTimeRepository.Update(model.Map<DbRealExecutionTime>(mapper)
)).Map<RealExecutionTime>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await realExecutionTimeRepository.Delete(id);
    }
```

```
    public async Task<List<RealExecutionTime>> GetByIds(IEnumerable<Guid>
ids)
    {
        return (await
realExecutionTimeRepository.GetByIds(ids)).MapToList<RealExecutionTime>(m
apper);
    }

    public async Task<RealExecutionTime> GetById(Guid id)
    {
        return (await realExecutionTimeRepository.GetByIds(new List<Guid>
{id})).Map<RealExecutionTime>(mapper);
    }

    public async Task<List<RealExecutionTime>> GetByTasks(List<Guid>
taskIds)
    {
        return (await
realExecutionTimeRepository.GetByTasks(taskIds)).MapToList<RealExecutionT
ime>(mapper);
    }

    public async Task<List<RealExecutionTime>> GetByTask(Guid taskId)
    {
        return (await realExecutionTimeRepository.GetByTasks(new
List<Guid> {taskId})).MapToList<RealExecutionTime>(mapper);
    }
}
```

Листинг Г.6 – Файл TaskService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class TaskService(ITaskRepository taskRepository,
IRealExecutionTimeService realExecutionTimeService, IMapper mapper) :
ITaskService
{
    public async Task<TaskModel> Add(TaskModel model)
    {
        return (await
taskRepository.Add(model.Map<DbTask>(mapper))).Map<TaskModel>(mapper);
    }

    public async Task<TaskModel> Update(TaskModel model)
    {
        return (await
taskRepository.Update(model.Map<DbTask>(mapper))).Map<TaskModel>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await taskRepository.Delete(id);
    }
```

```
    public async Task<List<TaskModel>> GetByIds(IEnumerable<Guid> ids)
    {
        return (await
taskRepository.GetByIds(ids)).MapToList<TaskModel>(mapper);
    }

    public async Task<TaskModel> GetById(Guid id)
    {
        return (await taskRepository.GetByIds(new List<Guid>
{id})).Map<TaskModel>(mapper);
    }

    public async Task<List<TaskModel>> GetByProjects(List<Guid> ids)
    {
        var tasks = (await
taskRepository.GetByProjects(ids)).MapToList<TaskModel>(mapper);
        await tasks.GetRealExecutionTimes(realExecutionTimeService);

        return tasks;
    }

    public async Task<List<TaskModel>> GetByProject(Guid id)
    {
        return await GetByProjects(new List<Guid> {id});
    }
}

public static class TaskModelExtensions
{
    public static async Task<List<TaskModel>> GetRealExecutionTimes(this
List<TaskModel> tasks, IRealExecutionTimeService realExecutionTimeService)
    {
        var taskIds = tasks.Select(p => p.Id).ToList();
        var realExecutionTimes = await
realExecutionTimeService.GetByTasks(taskIds);
        var realExecutionTimesGroupByTask = realExecutionTimes.GroupBy(t =>
t.TaskId).ToDictionary(t => t.Key, t => t.ToList());
        tasks.ForEach(p => p.RealExecutionTimes =
realExecutionTimesGroupByTask.GetValueOrDefault(p.Id)?.ToList() ?? new
List<RealExecutionTime>());

        return tasks;
    }

    public static async Task<TaskModel> GetRealExecutionTimes(this TaskModel
task, IRealExecutionTimeService realExecutionTimeService)
    {
        var realExecutionTimes = await
realExecutionTimeService.GetByTask(task.Id);
        task.RealExecutionTimes = realExecutionTimes;

        return task;
    }
}
```

Листинг Г.7 – Файл UserService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;
```

```
namespace Services;

public class UserService : IUserService
{
    private readonly IMapper _mapper;
    private readonly IUserRepository _userRepository;

    public UserService(IUserRepository userRepository, IMapper mapper)
    {
        _userRepository = userRepository;
        _mapper = mapper;
    }

    public async Task<User?> GetByLogin(string login)
    {
        return (await
_userRepository.GetByLogin(login)).Map<User>(_mapper);
    }

    public async Task<List<User>> GetAll()
    {
        return (await _userRepository.GetAll()).MapToList<User>(_mapper);
    }

    public async Task<User> Add(User user)
    {
        user.Id = Guid.NewGuid();
        user.PasswordHash = user.Password;

        return (await
_userRepository.Add(user.Map<DbUser>(_mapper))).Map<User>(_mapper);
    }
}
```

Листинг Г.8 – Файл DapperContext.cs

```
using System.Data;
using Dapper;
using DataAccess.DataContexts.Interfaces;
using Microsoft.Extensions.Configuration;

namespace DataAccess.DataContexts;

public class DapperContext : IDataContext
{
    private readonly DbFactory.ConnectionMethod _connectionMethod;
    private readonly string _returningKeyword;

    public DapperContext(IConfiguration config)
    {
        var dbSettings = DbFactory.GetDbSettings(config);
        _connectionMethod = dbSettings.Item1;
        _returningKeyword = dbSettings.Item2;
    }

    public async Task<T> InsertAsync<T>(string script, object param)
    {
```

```
        script = PrepareInsertQuery(script);

        await using var db = _connectionMethod();

        return await db.QueryFirstOrDefaultAsync<T>(script, param);
    }

    public async Task<IEnumerable<int>> InsertManyAsync<T>(string script,
IEnumerable<T> objects)
    {
        script = PrepareInsertQuery(script);

        await using var db = _connectionMethod();
        using IDbTransaction transaction = await
db.BeginTransactionAsync();

        var insertedObjectsIds = new List<int>();

        foreach (var item in objects)
        {
            var id = await db.ExecuteScalarAsync<int>(script, item,
transaction);
            insertedObjectsIds.Add(id);
        }

        transaction.Commit();

        return insertedObjectsIds;
    }

    public async Task<T?> FirstOrDefaultAsync<T>(string script, object
param)
    {
        await using var db = _connectionMethod();

        return await db.QueryFirstOrDefaultAsync<T>(script, param);
    }

    public async Task<IEnumerable<T>> EnumerableOrEmptyAsync<T>(string
script, object param)
    {
        await using var db = _connectionMethod();

        return await db.QueryAsync<T>(script, param);
    }

    public async Task<bool> DeleteAsync(string script, object param)
    {
        await using var db = _connectionMethod();

        return await db.ExecuteAsync(script, param) > 0;
    }

    public async Task<long> DeleteManyAsync(string script, object param)
    {
        await using var db = _connectionMethod();

        return await db.ExecuteAsync(script, param);
```

```
    }

    public async Task<T> UpdateAsync<T>(string script, object param)
    {
        await using var db = _connectionMethod();

        return await db.QueryFirstOrDefaultAsync<T>(script, param);
    }

    public async Task<IEnumerable<T>> UpdateManyAsync<T>(string script,
object param)
    {
        await using var db = _connectionMethod();

        return await db.QueryAsync<T>(script, param);
    }

    private string PrepareInsertQuery(string query)
    {
        return string.Format(query, _returningKeyword);
    }
}
```

Листинг Г.9 – Файл TaskController.cs

```
using Api.Controllers.Interfaces;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;
using Services.Models;

namespace Api.Controllers;

[ApiController]
[Route("task")]
public class TaskController(IMapper mapper, ITaskService taskService) :
Controller, ICrudController<TaskModel, Guid>
{
    private readonly IMapper _mapper = mapper;

    [HttpPost("add")]
    public async Task<ActionResult<TaskModel>> Add(TaskModel task)
    {
        return Ok(await taskService.Add(task));
    }

    [HttpGet("by-id/{id}")]
    public async Task<ActionResult<TaskModel>> GetById(Guid id)
    {
        return Ok(await taskService.GetById(id));
    }

    [HttpPut]
    public async Task<ActionResult<TaskModel>> Update(TaskModel model)
    {
        return Ok(await taskService.Update(model));
    }
```

```
    [HttpDelete("{id}")]
    public async Task<ActionResult<TaskModel>> Delete(Guid id)
    {
        return Ok(await taskService.Delete(id));
    }
}
```

Листинг Г.10 – Файл WebAppConfigurationExtensions.cs

```
using Api.Middlewares;
using DataAccess;
using DataAccess.DataContexts;
using DataAccess.DataContexts.Interfaces;
using DataAccess.Repositories;
using DataAccess.Repositories.Interfaces;
using FluentMigrator.Runner;
using Microsoft.OpenApi.Models;
using Services;
using Services.Interfaces;

namespace Api.Extensions;

public static class MigrationExtensions
{
    public static void AddMigrations(this IHost app)
    {
        using var scope = app.Services.CreateScope();
        var migrator =
scope.ServiceProvider.GetService<IMigrationRunner>();
        migrator?.MigrateUp();
    }

    public static void ConfigureMigrator(this IMigrationRunnerBuilder
migrationRunnerBuilder, ConfigurationManager config)
    {
        DbFactory.GetMigrationSetuper(config,
migrationRunnerBuilder)(migrationRunnerBuilder);
    }
}

public static class AuthAppExtensions
{
    public static void AddAuthenticationWithOptions(this
WebApplicationBuilder builder)
    {
        builder.Services.AddAuthorization();
    }
}

public static class SwaggerAppExtensions
{
    public static void AddSwaggerOptions(this WebApplicationBuilder
builder)
    {
        builder.Services.AddSwaggerGen(options =>
        {
            options.AddSecurityDefinition("Bearer",
                new OpenApiSecurityScheme
```

```
                {
                    Description = @"Введите JWT токен авторизации.",
                    Name = "Authorization",
                    In = ParameterLocation.Header,
                    Type = SecuritySchemeType.ApiKey,
                    BearerFormat = "JWT",
                    Scheme = "Bearer"
                });

            options.AddSecurityRequirement(new OpenApiSecurityRequirement
            {
                {new OpenApiSecurityScheme {Reference = new
OpenApiReference {Type = ReferenceType.SecurityScheme, Id = "Bearer"}},
new List<string>()}
            });
        });
    }
}

public static class WebAppConfigurationExtensions
{
    public static void ConfigureApp(this WebApplication app)
    {
        app.AddMigrations();
        app.UseAuthentication();
        app.UseAuthorization();
        app.UseMiddleware<GlobalExceptionMiddleware>();

        app.MapControllers();

        if (app.Environment.IsDevelopment() ||
app.Environment.IsEnvironment("Local"))
        {
            app.UseSwagger();
            app.UseSwaggerUI();
            app.UseDeveloperExceptionPage();
        }
    }

    public static void Configure(this WebApplicationBuilder builder)
    {
        Console.WriteLine(builder.Configuration["PGConnection"]);

        builder.Services.AddFluentMigratorCore()
            .ConfigureRunner(c =>
c.ConfigureMigrator(builder.Configuration))
            .AddLogging();

        builder.Services.AddControllers().AddNewtonsoftJson();

        builder.AddAuthenticationWithOptions();
        builder.ConfigureService();
        builder.AddSwaggerOptions();
        builder.Services.AddSwaggerGen();
        builder.Services.AddAutoMapper(typeof(AppMappingProfile));
    }
```

```
    private static void ConfigureService(this WebApplicationBuilder
builder)
    {
        builder.Services.AddTransientGroup();
        builder.Services.AddScopedGroup();
        builder.Services.AddSingletonGroup();
    }

    private static void AddTransientGroup(this IServiceCollection
services)
    {
        services.AddTransient<ITaskService, TaskService>();
        services.AddTransient<IRealExecutionTimeService,
RealExecutionTimeService>();
        services.AddTransient<IProjectService, ProjectService>();
        services.AddTransient<IActionDirectionService,
ActionDirectionService>();
        services.AddTransient<IPlanningTimeService,
PlanningTimeService>();
        services.AddTransient<IGoalService, GoalService>();
        services.AddTransient<ILifeSphereService, LifeSphereService>();
    }

    private static void AddScopedGroup(this IServiceCollection services)
    {
        services.AddScoped<ITaskRepository, TaskRepository>();
        services.AddScoped<IRealExecutionTimeRepository,
RealExecutionTimeRepository>();
        services.AddScoped<IProjectRepository, ProjectRepository>();
        services.AddScoped<IActionDirectionRepository,
ActionDirectionRepository>();
        services.AddScoped<IPlanningTimeRepository,
PlanningTimeRepository>();
        services.AddScoped<IGoalRepository, GoalRepository>();
        services.AddScoped<ILifeSphereRepository,
LifeSphereRepository>();
    }

    private static void AddSingletonGroup(this IServiceCollection
services)
    {
        services.AddSingleton<IDataContext, DapperContext>();
    }
}
```

Листинг Г.11 – Файл ActionDirectionRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.ActionDirections;

namespace DataAccess.Repositories;

public class ActionDirectionRepository(IDataContext dataContext) :
IActionDirectionRepository
{
    public async Task<DbActionDirection> Add(DbActionDirection model)
```

```
    {
        model.Id = Guid.NewGuid();

        return await
dataContext.InsertAsync<DbActionDirection>(ActionDirections.Insert,
model);
    }

    public async Task<DbActionDirection> Update(DbActionDirection model)
    {
        return await
dataContext.UpdateAsync<DbActionDirection>(ActionDirections.Update,
model);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await dataContext.DeleteAsync(ActionDirections.Delete, new
{id});
    }

    public async Task<IEnumerable<DbActionDirection>>
GetByIds(IEnumerable<Guid> ids)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbActionDirection>(ActionDirections.Ge
tByIds, new {ids});
    }

    public async Task<IEnumerable<DbActionDirection>>
GetByLifeSpheres(List<Guid> lifeSphereIds)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbActionDirection>(ActionDirections.Ge
tAllByLifeSpheres, new {lifeSphereIds});
    }
}
```

Листинг Г.12 – Файл UserRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.User;

namespace DataAccess.Repositories;

public class UserRepository : IUserRepository
{
    private readonly IDataContext _dataContext;

    public UserRepository(IDataContext dataContext)
    {
        _dataContext = dataContext;
    }

    public async Task<IEnumerable<DbUser>> GetAll()
    {
```

```
        return await
_dataContext.EnumerableOrEmptyAsync<DbUser>(Users.GetAll, new { });
    }

    public async Task<DbUser> Add(DbUser model)
    {
        return await _dataContext.InsertAsync<DbUser>(Users.Insert,
model);
    }

    public Task<IEnumerable<DbUser>> GetByIds(IEnumerable<int> ids)
    {
        throw new NotImplementedException();
    }

    public Task<DbUser> GetById(int id)
    {
        throw new NotImplementedException();
    }

    public async Task<DbUser?> GetByLogin(string login)
    {
        return await
_dataContext.FirstOrDefaultAsync<DbUser>(Users.GetByLogin, new {login =
login});
    }
}
```

Листинг Г.13 – Файл GoalController.cs

```
using Api.Controllers.Interfaces;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;
using Services.Models;

namespace Api.Controllers;

[ApiController]
[Route("goal")]
public class GoalController(IMapper mapper, IGoalService goalService) :
Controller, ICrudController<Goal, Guid>
{
    private readonly IMapper _mapper = mapper;

    [HttpPost]
    public async Task<ActionResult<Goal>> Add(Goal goal)
    {
        return Ok(await goalService.Add(goal));
    }

    [HttpGet("by-id/{id}")]
    public async Task<ActionResult<Goal>> GetById(Guid id)
    {
        return Ok(await goalService.GetById(id));
    }

    [HttpPut]
```

```
    public async Task<ActionResult<Goal>> Update(Goal model)
    {
        return Ok(await goalService.Update(model));
    }

    [HttpDelete("{id}")]
    public async Task<ActionResult<Goal>> Delete(Guid id)
    {
        return Ok(await goalService.Delete(id));
    }

    [HttpGet("by-action-direction/{actionDirectionId}")]
    public async Task<ActionResult<List<Goal>>> GetByActionDirection(Guid
actionDirectionId)
    {
        return Ok(await
goalService.GetGetByActionDirection(actionDirectionId));
    }
}
```

Приложение Д

(рекомендуемое)

Листинг клиентской части приложения

Листинг Д.1 – Файл AuthPage.jsx

```
import React, { useState } from 'react';
import axios from 'axios';
import { Link, useNavigate } from 'react-router-dom';

function AuthPage() {
    const [login, setLogin] = useState('');
    const [password, setPassword] = useState('');
    const [errorMessage, setErrorMessage] = useState('');
    const navigate = useNavigate();

    const handleSubmit = async (event) => {
        event.preventDefault();

        const withHash = true;
        const url = `/api1/auth/auth/?withHash=${withHash}`;

        try {
            const response = await axios.post(url, {
                Login: login,
                Password: password
            }, {
                headers: {
                    'Content-Type': 'application/json'
                }
            });

            if (response.status === 200) {
                console.log(response.data);
                localStorage.setItem('user',
JSON.stringify(response.data));
                navigate('/main');
            } else {
                setErrorMessage(response.data.message || 'Ошибка
авторизации');
            }
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };

    return (
        <div className="auth-page">
            <h1>Авторизация</h1>
            {errorMessage && <p className="error">{errorMessage}</p>}
            <form onSubmit={handleSubmit}>
                <div>
                    <label htmlFor="login">Логин:</label>
                    <input
                        type="text"
                        id="login"
                        value={login}
                        onChange={(e) => setLogin(e.target.value)}
                    />
                </div>
                <div>
```

```
                    <label htmlFor="password">Пароль:</label>
                    <input
                        type="password"
                        id="password"
                        value={password}
                        onChange={(e) => setPassword(e.target.value)}
                    />
                </div>
                <button type="submit">Войти</button>
            </form>
            <p>Нет аккаунта? <Link
to="/reg">Зарегистрироваться</Link></p>
        </div>
    );
}

export default AuthPage;
```

Листинг Д.2 – Файл RegPage.jsx

```
import React, { useState } from 'react';
import axios from 'axios';

function RegPage() {
    const [login, setLogin] = useState('');
    const [password, setPassword] = useState('');
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    const [birthday, setBirthday] = useState('');
    const [errorMessage, setErrorMessage] = useState('');
    const [height, setHeight] = useState(0);
    const [weight, setWeight] = useState(0);

    const handleSubmit = async (event) => {
        event.preventDefault();
        // Преобразовываем строковые значения в числа
        const numericHeight = Number(height);
        const numericWeight = Number(weight);

        // Проверка, что поля не пустые
        if (!login || !password || !name || !email || !birthday ||
!numericHeight || !numericWeight) {
            setErrorMessage("Все поля должны быть заполнены!");
            return;
        }

        const userData = {
            Login: login,
            Password: password,
            Name: name,
            Email: email,
            Birthday: birthday,
            Height: numericHeight,
            Weight: numericWeight
        };

        try {
```

```
            const withHash = true;
            const url = `/api1/auth/reg/?withHash=${withHash}`;
            console.log("URL being used:", url);

            axios.post(url, userData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessage('Вы успешно зарегистрированы!');
                navigate('/auth');
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    // Может содержать более конкретную информацию об
ошибке
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessage(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessage('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };

    return (
        <div className="reg-page">
            <h1>Регистрация</h1>
            {errorMessage && <p className="error">{errorMessage}</p>}
            <form onSubmit={handleSubmit}>
                <div>
                    <label htmlFor="login">Логин:</label>
                    <input
                        type="text"
                        id="login"
                        value={login}
                        onChange={(e) => setLogin(e.target.value)}
                    />
                </div>
                <div>
                    <label htmlFor="password">Пароль:</label>
                    <input
                        type="password"
                        id="password"
                        value={password}
                        onChange={(e) => setPassword(e.target.value)}
                    />
                </div>
                <div>
                    <label htmlFor="name">Имя:</label>
                    <input
                        type="text"
                        id="name"
                        value={name}
```

```
                        onChange={(e) => setName(e.target.value)}
                    />
                </div>
                <div>
                    <label htmlFor="email">Email:</label>
                    <input
                        type="email"
                        id="email"
                        value={email}
                        onChange={(e) => setEmail(e.target.value)}
                    />
                </div>
                <div>
                    <label htmlFor="birthday">Дата рождения:</label>
                    <input
                        type="date"
                        id="birthday"
                        value={birthday}
                        onChange={(e) => setBirthday(e.target.value)}
                    />
                </div>
                <div>
                    <label htmlFor="height">Рост (см):</label>
                    <input
                        type="number"
                        id="height"
                        value={height}
                        onChange={(e) => setHeight(e.target.value)}
                    />
                </div>
                <div>
                    <label htmlFor="weight">Вес (кг):</label>
                    <input
                        type="number"
                        id="weight"
                        value={weight}
                        onChange={(e) => setWeight(e.target.value)}
                    />
                </div>
                <button type="submit">Зарегистрироваться</button>
            </form>
        </div>
    );
}

export default RegPage;
```

Листинг Д.3 – Файл PlanningPage.jsx

```
import Calendars from './components/planning/Calendars';
import Settings from './components/planning/Settings';
import Tasks from './components/planning/Tasks';
import Analize_P from './components/planning/Analize_P';
import { Routes, Route, Link } from 'react-router-dom';
import React, { useEffect } from 'react';
import axios from 'axios';
import PlanDataUpdate from './components/planning/PlanDataUpdate';
```

```
function Planning() {

    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    PlanDataUpdate(user.id);

    return (
        <div className="content_col">
            <div className="leftnav">
                <Link to="setting">Настройка системы</Link>
                <Link to="tasks">Все задачи</Link>
                <Link to="calendars">Календари</Link>
                <Link to="analize_plan">Анализ продуктивности</Link>
            </div>
            <div className="content_razdel">
                <div className="content_razdel_style">
                    <Routes>
                        <Route path="setting" element={<Settings />} />
                        <Route path="tasks" element={<Tasks />} />
                        <Route path="calendars" element={<Calendars />}
/>
                        <Route path="analize_plan" element={<Analize_P
/>} />
                    </Routes>
                </div>
            </div>
        </div>
    );
}

export default Planning;
```

Листинг Д.4 – Файл PlanDataUpdate.jsx

```
import axios from 'axios';

function PlanDataUpdate(userId) {
    return axios.get(`/api2/life-sphere/by-user/${userId}`)
        .then(response => {
            localStorage.setItem('userPlanData',
JSON.stringify(response.data));
            return response.data; // Возвращаем данные для использования
промиса
        })
        .catch(error => {
            console.error('Error fetching health data:', error);
            throw error; // Пробрасываем ошибку для обработки в
вызывающем коде
        });
}

export default PlanDataUpdate;
```

Листинг Д.5 – Файл Analize_P.jsx

```
import React, { useState, useEffect } from "react";
```

```
import axios from "axios";
import { Line } from "react-chartjs-2";
import { subDays, format } from 'date-fns';
import { Bar } from "react-chartjs-2";
import { Doughnut } from 'react-chartjs-2';
import 'chart.js/auto';
import {
    Chart,
    LinearScale,
    CategoryScale,
    BarElement,
    LineElement,
    PointElement,
    LineController, Chart, ArcElement, DoughnutController, Tooltip
} from 'chart.js';

function getRandomColor() {
    const r = Math.floor(Math.random() * 256);
    const g = Math.floor(Math.random() * 256);
    const b = Math.floor(Math.random() * 256);
    const transparency = 0.7; // Вы можете выбрать или менять уровень
прозрачности

    return `rgba(${r}, ${g}, ${b}, ${transparency})`;
}

function Analize_P() {
    const [chartData, setChartData] = useState(null);
    const [chartData2, setChartData2] = useState(null);
    const [chartData3, setChartData3] = useState(null);
    const [dateLabels, setDateLabels] = useState([]);
    const [Dataaa, setDataaa] = useState([]);
    const [filterTableData, setFilterTableData] = useState([]);
    const userDataString = localStorage.getItem("user");
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;
    const userId = user ? user.id : null;

    const [lifeSphereHistories, setLifeSphereHistories] = useState([]);
    const [directionHistories, setDirectionHistories] = useState([]);
    const [tasksHistories, setTasksHistories] = useState([]);
    const [timesHistories, setTimesHistories] = useState([]);
    const [projectHistories, setProjectHistories] = useState([]);

    useEffect(() => {
        Chart.register(
            LinearScale,
            CategoryScale,
            BarElement,
            LineElement,
            PointElement,
            LineController, ArcElement, DoughnutController, Tooltip
        );
        fetchData(); // Загрузка данных при монтировании
    }, [userId]); // Перезагрузка данных при изменении userId

    const fetchData = async () => {
        if (!userId) return;
```

```
        try {
            const response = await axios.get(`/api2/life-sphere/by-
user/${userId}`);
            localStorage.setItem("userPlanData",
JSON.stringify(response.data));
            var userPlans = JSON.parse(localStorage.getItem('userPlans'))
|| [];
            const dataPlans = userPlans || [];
            const storedData = localStorage.getItem("userPlanData");
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userPlanData = parsedData;
                    const lifeSphere = userPlanData || [];
                    const actionDirection = lifeSphere.map(l =>
l.actionDirections).flat() || [];
                    const project = actionDirection.map(l =>
l.projects).flat() || [];
                    const task = project.map(l => l.tasks).flat() || [];
                    const times = task.map(l =>
l.realExecutionTimes).flat() || [];
                    setLifeSphereHistories(lifeSphere);
                    setDirectionHistories(actionDirection);
                    setProjectHistories(project);
                    setTasksHistories(task);
                    setTimesHistories(times);
                    processDataFact(actionDirection);
                    processDataFull(actionDirection, dataPlans);
                    processDirectionEfficiency(actionDirection,
dataPlans);
                    processDataForTable(actionDirection);
                }
            }
            processDataPlan(dataPlans);
        } catch (error) {
            console.error("Error fetching plan data:", error);
        }
    };

    const ActionTable = ({ data }) => {
        return (
            <table>
                <thead>
                    <tr>
                        <th>Название задачи</th>
                        <th>Часы, затраченные на выполнение</th>
                        <th>Направление</th>
                    </tr>
                </thead>
                <tbody>
                    {data.map((item, index) => (
                        <tr key={index}>
                            <td>{item.taskTitle}</td>
                            <td>{item.hoursSpent}</td>
                            <td>{item.directionTitle}</td>
                        </tr>
                    ))}
                </tbody>
```

```
            </table>
        );
    };

    const processDataFact = (actionDirections) => {
        const endDate = new Date();
        endDate.setDate(endDate.getDate() - 3);
        const labels = Array.from({ length: 7 }, (_, i) =>
format(subDays(endDate, 6 - i), "yyyy-MM-dd"));
        setDateLabels(labels);

        let dataByDirection = {};

        actionDirections.forEach(direction => {
            if (!dataByDirection[direction.title]) {
                dataByDirection[direction.title] = labels.reduce((obj,
label) => ({ ...obj, [label]: 0 }), {});
            }

            direction.projects.forEach(project => {
                project.tasks.forEach(task => {
                    task.realExecutionTimes.forEach(execution => {
                        const start = new Date(execution.startDate);
                        const end = new Date(execution.endDate);
                        const startDateFormatted = format(start, "yyyy-
MM-dd");

                        if (labels.includes(startDateFormatted)) {
                            const hours = (end - start) / (1000 * 60 *
60);

dataByDirection[direction.title][startDateFormatted] += hours;
                        }
                    });
                });
            });
        });

        const datasets = Object.keys(dataByDirection).map(direction => ({
            label: direction,
            data: labels.map(label =>
parseFloat(dataByDirection[direction][label].toFixed(2))),
            borderColor: getRandomColor(),
            backgroundColor: getRandomColor(),
            fill: false
        }));

        setChartData({
            labels,
            datasets
        });
    };

    const processDataForTable = (actionDirections) => {
        const tasksMap = {};

        actionDirections.forEach(direction => {
            direction.projects.forEach(project => {
```

```
                project.tasks.forEach(task => {
                    task.realExecutionTimes.forEach(executionTime => {
                        const start = new Date(executionTime.startDate);
                        const end = new Date(executionTime.endDate);
                        const hours = (end - start) / (1000 * 60 * 60);

                        const key = task.title + '-' + direction.title;

                        if (!tasksMap[key]) {
                            tasksMap[key] = {
                                taskTitle: task.title,
                                hoursSpent: 0, // начинаем с 0, будет
накапливаться
                                directionTitle: direction.title
                            };
                        }
                        tasksMap[key].hoursSpent += hours; // накапливаем
время
                    });
                });
            });
        });

        const tableData = Object.values(tasksMap);
        tableData.forEach(item => item.hoursSpent =
item.hoursSpent.toFixed(2));

        setFilterTableData(tableData);
    };


    const processDataPlan = (actionDirectionsPlan) => {
        const endDate = new Date();
        endDate.setDate(endDate.getDate() - 3);
        const labels = Array.from({ length: 7 }, (_, i) =>
format(subDays(endDate, 6 - i), "yyyy-MM-dd"));
        setDateLabels(labels);

        let dataByDirection = {};

        actionDirectionsPlan.forEach(direction => {
            if (!dataByDirection[direction.Title]) {
                dataByDirection[direction.Title] = labels.reduce((obj,
label) => ({
                    ...obj,
                    [label]: 0
                }), {});
            }

            const date = new Date(direction.DatePlan);
            const startDateFormatted = format(date, "yyyy-MM-dd");

            if (labels.includes(startDateFormatted)) {
                const parsedHours = Number(direction.HourAmmountPlan);
                if (!isNaN(parsedHours)) {  // Проверяем, успешно ли
произошло преобразование
                    dataByDirection[direction.Title][startDateFormatted]
+= parsedHours;
```

```
                } else {
                    console.error('HourAmmountPlan contains non-numeric
data:', direction.HourAmmountPlan);
                }
            }
        });

        const datasets = Object.keys(dataByDirection).map(direction => ({
            label: direction,
            data: labels.map(label =>
parseFloat(dataByDirection[direction][label].toFixed(2))),
            borderColor: getRandomColor(),
            backgroundColor: getRandomColor(),
            fill: false
        }));

        setChartData2({
            labels,
            datasets
        });
    };

    const processDataFull = (actionDirections, actionDirectionsPlan) => {
        const endDate = new Date();
        endDate.setDate(endDate.getDate() - 3);
        const startDate = subDays(endDate, 6);
        const labels = Array.from({ length: 7 }, (_, i) =>
format(subDays(endDate, 6 - i), "yyyy-MM-dd"));
        setDateLabels(labels);

        let dataByDate = {};

        actionDirections.forEach(direction => {
            direction.projects.forEach(project => {
                project.tasks.forEach(task => {
                    task.realExecutionTimes.forEach(execution => {
                        const start = new Date(execution.startDate);
                        const end = new Date(execution.endDate);
                        const startDateFormatted = format(start, "yyyy-
MM-dd");
                        if (labels.includes(startDateFormatted)) {
                            const hours = (end - start) / (1000 * 60 *
60);
                            dataByDate[startDateFormatted] =
(dataByDate[startDateFormatted] || 0) + hours;
                        }
                    });
                });
            });
        });

        const hoursDataFact = labels.map(label =>
parseFloat((dataByDate[label] || 0).toFixed(2)));

        let dataByDate2 = {};

        actionDirectionsPlan.forEach(direction => {
            const date = new Date(direction.DatePlan);
```

```
            const startDateFormatted = format(date, "yyyy-MM-dd");

            if (labels.includes(startDateFormatted)) {
                const parsedHours = Number(direction.HourAmmountPlan);
                if (!isNaN(parsedHours)) {  // Проверяем, успешно ли
произошло преобразование
                    dataByDate2[startDateFormatted] =
(dataByDate2[startDateFormatted] || 0) + parsedHours;
                } else {
                    console.error('HourAmmountPlan contains non-numeric
data:', direction.HourAmmountPlan);
                }
            }
        });

        const hoursDataPlan = labels.map(label =>
parseFloat((dataByDate2[label] || 0).toFixed(2)));

        setChartData3({
            labels,
            datasets: [
                {
                    label: 'Запланированные часы на деятельность',
                    data: hoursDataPlan,
                    borderColor: 'rgb(30, 230, 40)',
                    backgroundColor: 'rgb(30, 230, 40)',
                    fill: false,
                    lineTension: 0.5,
                },
                {
                    label: 'Затраченные часы на деятельность',
                    data: hoursDataFact,
                    borderColor: 'rgb(230, 30, 220)',
                    backgroundColor: 'rgb(230, 30, 220)',
                    fill: false,
                    lineTension: 0.5,
                }
            ]
        });

    };

    const processDirectionEfficiency = (actionDirections,
actionDirectionsPlan) => {
        const directionEfficiency = {};

        actionDirections.forEach(direction => {
            let totalFactHours = 0;
            let totalPlanHours = 0;

            direction.projects.forEach(project => {
                project.tasks.forEach(task => {
                    task.realExecutionTimes.forEach(execution => {
                        const start = new Date(execution.startDate);
                        const end = new Date(execution.endDate);
                        const hours = (end - start) / (1000 * 60 * 60);
                        totalFactHours += hours;
                    });
```

```
                });
            });

            const planDirection = actionDirectionsPlan.find(plan =>
plan.Title === direction.title);
            if (planDirection) {
                totalPlanHours = Number(planDirection.HourAmmountPlan);
            }

            if (totalPlanHours > 0) {
                directionEfficiency[direction.title] = totalPlanHours /
totalFactHours;
            } else {
                directionEfficiency[direction.title] = 0;
            }
        });

        const sortedDirections = Object.entries(directionEfficiency)
            .sort((a, b) => b[1] - a[1])
            .map(([title, efficiency]) => ({ title, efficiency }));

        setDataaa(sortedDirections);
    };

    const DirectionEfficiencyList = ({ efficiencyData }) => {
        return (
            <div>
                <h2>Анализ эффективности по направлениям</h2>
                <table>
                    <thead>
                        <tr>
                            <th>Направление</th>
                            <th>Соотношение затраченного времени к
запланированному</th>
                        </tr>
                    </thead>
                    <tbody>
                        {efficiencyData.map(({ title, efficiency },
index) => (
                            <tr key={index}>
                                <td>{title}</td>
                                <td>{efficiency.toFixed(2)}</td>
                            </tr>
                        ))}
                    </tbody>
                </table>
            </div>
        );
    };

    return (
        <div className="analize">
            <button onClick={fetchData}>Сформировать отчёт</button>
            <div className='grafik' style={{ height: "550px", width:
"100%" }}>
                <h2>Сравнение затраченного времени на деятельность с
запланированным</h2>
                {chartData3 && (
```

```
                    <Line
                        data={chartData3}
                        options={{
                            plugins: {
                                legend: {
                                    labels: {
                                        font: {
                                            size: 18,
                                            weight: 'bold'
                                        }
                                    }
                                }
                            },
                            scales: {
                                y: {
                                    type: 'linear',
                                    beginAtZero: true,
                                    title: {
                                        display: true,
                                        text: 'Часы',
                                        font: {
                                            size: 18,
                                            weight: 'bold'
                                        }
                                    }
                                },
                                x: {
                                    type: 'category',
                                    labels: dateLabels,
                                    title: {
                                        display: true,
                                        text: 'День',
                                        font: {
                                            size: 18,
                                            weight: 'bold'
                                        }
                                    }
                                }
                            },
                            responsive: true,
                            maintainAspectRatio: false
                        }}
                    />
                )}
            </div>
            <h2 className="table_title">Задачи, выполненные за последнюю
неделю</h2>
            <ActionTable data={filterTableData} />
            <div className='grafik' style={{ height: "550px", width:
"100%" }}>
                <h2>Затраченное время в часах по направлениям</h2>
                {chartData && (
                    <Bar
                        data={chartData}
                        options={{
                            plugins: {
                                legend: {
                                    labels: {
```

```
                                    font: {
                                        size: 18,
                                        weight: 'bold'
                                    }
                                }
                            }
                        },
                        scales: {
                            y: {
                                type: 'linear',
                                beginAtZero: true,
                                title: {
                                    display: true,
                                    text: 'Часы',
                                    font: {
                                        size: 18,
                                        weight: 'bold'
                                    }
                                }
                            },
                            x: {
                                type: 'category',
                                labels: dateLabels,
                                title: {
                                    display: true,
                                    text: 'День',
                                    font: {
                                        size: 18,
                                        weight: 'bold'
                                    }
                                }
                            }
                        },
                        responsive: true,
                        maintainAspectRatio: false
                    }}
                />
            )}
        </div>
        <div className='grafik' style={{ height: "550px", width:
"100%" }}>
            <h2>Запланированное время в часах по направлениям</h2>
            {chartData2 && (
                <Bar
                    data={chartData2}
                    options={{
                        plugins: {
                            legend: {
                                labels: {
                                    font: {
                                        size: 18,
                                        weight: 'bold'
                                    }
                                }
                            }
                        },
                        scales: {
                            y: {
```

```
                                    type: 'linear',
                                    beginAtZero: true,
                                    title: {
                                        display: true,
                                        text: 'Часы',
                                        font: {
                                            size: 18,
                                            weight: 'bold'
                                        }
                                    }
                                },
                                x: {
                                    type: 'category',
                                    labels: dateLabels,
                                    title: {
                                        display: true,
                                        text: 'День',
                                        font: {
                                            size: 18,
                                            weight: 'bold'
                                        }
                                    }
                                }
                            },
                            responsive: true,
                            maintainAspectRatio: false
                        }}
                    />
                )}
            </div>
            <DirectionEfficiencyList efficiencyData={Dataaa} />
        </div>
    );
}

export default Analize_P;
```

Листинг Д.6 – Файл Tasks.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Modal from '../Modal';
import planDataUpdate from './PlanDataUpdate';
import TaskDataComponent from './TaskDataComponent';

function Tasks() {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [isOpenTask, setIsOpenTask] = useState(false);

    //Задача
    const [titleTask, setTitleTask] = useState('');
    const [descriptionTask, setDescriptionTask] = useState('');
    const [startDateTask, setStartDateTask] = useState('');
    const [endDateTask, setEndDateTask] = useState('');
    const [projectTask, setProjectTask] = useState('');
```

```
    const [errorMessage, setErrorMessage] = useState('');

    const [tasksDataUpdated, setTasksDataUpdated] = useState(false);
    const [planData, setPlanData] =
useState(JSON.parse(localStorage.getItem('userPlanData')));

    const [tasksHistories, setTasksHistories] = useState([]);
    const [projectHistories, setProjectHistories] = useState([]);

    const closeModalTask = () => {
        setIsOpenTask(false);
    };

    useEffect(() => {
        if (isOpenTask) {
            setTasksDataUpdated(false); // Сбрасываем флаг при открытии
модального окна
        } else {
            setTasksDataUpdated(true); // Выставляем обновленное значение
при закрытии модального окна
        }
        const updateTasksData = () => {
            const storedData = localStorage.getItem('userPlanData');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userPlanData = parsedData;
                    const lifeSphere = userPlanData || [];
                    const actionDirection = lifeSphere.map(l =>
l.actionDirections).flat() || [];
                    const project = actionDirection.map(l =>
l.projects).flat() || [];
                    const task = project.map(l => l.tasks).flat() || [];
                    setTasksHistories(task);
                    setProjectHistories(project);
                } else {
                    console.log('Parsed data is not an array or is
empty');
                }
            } else {
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
            if (event.key === 'userPlanData') {
                updateSettingsData();
            }
        };

        updateTasksData();

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };
```

```
    }, [isOpenTask, tasksHistories, projectHistories]);

    const openModalTask = () => {
        setIsOpenTask(true);
    };

    const handleSubmitTask = async (event) => {
        event.preventDefault();

        if (!titleTask || !descriptionTask || !projectTask ||
!startDateTask || !endDateTask) {
            setErrorMessage("Все поля должны быть заполнены!");
            return;
        }

        const userTaskData = {
            UserId: user.id,
            CreatedDate: new Date(),
            Title: titleTask,
            Description: descriptionTask,
            ProjectId: projectTask,
            StartExecutionDate: startDateTask,
            DeadlineDate: endDateTask
        };

        try {
            const url = `/api2/task/add`;

            axios.post(url, userTaskData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessage('Данные успешно добавлены!');
                planDataUpdate(user.id);
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessage(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessage('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };

    return (
        <div>
            <h2>Планирование/Задачи</h2>
            <div className='row-btn'>
                <button onClick={openModalTask}>Добавить задачу</button>
            </div>
```

```
            {isOpenTask && (
                <Modal onClose={closeModalTask}>
                    <h2>Добавить задачу</h2>
                    {errorMessage && <p
className="error">{errorMessage}</p>}
                    <form onSubmit={handleSubmitTask}>
                        <div>
                            <label htmlFor="titleTask">Название:</label>
                            <input
                                type="text"
                                id="titleTask"
                                value={titleTask}
                                onChange={(e) =>
setTitleTask(e.target.value)}
                            />
                        </div>
                        <div>
                            <label
htmlFor="descriptionTask">Описание:</label>
                            <input
                                type="text"
                                id="descriptionTask"
                                value={descriptionTask}
                                onChange={(e) =>
setDescriptionTask(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="startDateTask">Планируемая
дата начала:</label>
                            <input
                                type="datetime-local"
                                id="startDateTask"
                                value={startDateTask}
                                onChange={(e) =>
setStartDateTask(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="endDateTask">Планируемая дата
завершения:</label>
                            <input
                                type="datetime-local"
                                id="endDateTask"
                                value={endDateTask}
                                onChange={(e) =>
setEndDateTask(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="projectTask">Проект:</label>
                            <select value={projectTask} onChange={e =>
setProjectTask(e.target.value)}>
                                {projectHistories.map(project => <option
key={project.id} value={project.id}>{project.title}</option>)}
                            </select>
                        </div>
                        <button type="submit">Добавить</button>
```

```
                    </form>
                </Modal>
            )}
            <div className='history'>
                {tasksDataUpdated && <TaskDataComponent />}
            </div>
        </div>
    );
};

export default Tasks;
```

Листинг Д.7 – Файл TaskDataComponent.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import planDataUpdate from './PlanDataUpdate';
import Modal from '../Modal';
import Stopwatch from './Stopwatch';

function formatDateTime(dateString) {
    const options = { year: 'numeric', month: 'numeric', day: 'numeric',
hour: '2-digit', minute: '2-digit', second: '2-digit' };
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function formatDate(dateString) {
    const options = { year: 'numeric', month: 'numeric', day: 'numeric'
};
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function formatTime(timeString) {
    if (typeof timeString === 'string') {
        return timeString.slice(0, 5);
    }
    return 'Время не задано';
}

function TaskDataComponent() {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [taskHistories, setTaskHistories] = useState([]);
    const [lifeSphereHistories, setLifeSphereHistories] = useState([]);
    const [directionHistories, setDirectionHistories] = useState([]);
    const [projectHistories, setProjectHistories] = useState([]);

    const [filter, setFilter] = useState([]);
    const [dateRange, setDateRange] = useState({ startDate: new Date(),
endDate: new Date() });
    const [filteredHistories, setFilteredHistories] = useState([]);

    const [objectName, setObjectName] = useState(null);
    const [objectTitle, setObjectTitle] = useState(null);
    const [objectDescription, setObjectDescription] = useState(null);
    const [objectDate, setObjectDate] = useState(null);
```

```
    const [objectStartDate, setObjectStartDate] = useState(null);
    const [objectEndDate, setObjectEndDate] = useState(null);
    const [objectProject, setObjectProject] = useState(null);
    const [objectDirection, setObjectDirection] = useState(null);
    const [objectLifeSphere, setObjectLifeSphere] = useState(null);

    const [isOpen, setisOpen] = useState(false);

    const closeModal = () => {
        setisOpen(false);
    };

    const openModal = () => {
        setisOpen(true);
    };

    const filterHistories = () => {
        const today = new Date();
        today.setHours(0, 0, 0, 0);
        const yesterday = new Date(today);
        yesterday.setDate(yesterday.getDate() - 1);

        switch (filter) {
            case 'today':
                return taskHistories.filter(history =>
                    new Date(history.createdDate).setHours(0, 0, 0, 0)
=== today.getTime());
            case 'yesterday':
                return taskHistories.filter(history =>
                    new Date(history.createdDate).setHours(0, 0, 0, 0)
=== yesterday.getTime());
            case 'range':
                return taskHistories.filter(history => {
                    const historyDate = new Date(history.createdDate);
                    return historyDate >= dateRange.startDate &&
historyDate <= dateRange.endDate;
                });
            case 'all':
            default:
                return taskHistories;
        }
    };

    useEffect(() => {
        const updateTaskData = () => {
            const storedData = localStorage.getItem('userPlanData');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userPlanData = parsedData;
                    const lifeSphere = userPlanData || [];
                    const actionDirection = lifeSphere.map(l =>
l.actionDirections).flat() || [];
                    const project = actionDirection.map(l =>
l.projects).flat() || [];
                    const task = project.map(l => l.tasks).flat() || [];
                    setLifeSphereHistories(lifeSphere);
                    setDirectionHistories(actionDirection);
```

```
                    setProjectHistories(project);
                    setTaskHistories(task);
                } else {
                    console.log('Parsed data is not an array or is
empty');
                }
            } else {
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
            if (event.key === 'userPlanData') {
                updateTaskData();
            }
        };

        updateTaskData();

        setFilteredHistories(filterHistories());

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };
    }, [filter, dateRange, taskHistories, lifeSphereHistories,
directionHistories, projectHistories]);

    const onShowTask = (taskHistoryId) => {
        const task = taskHistories.find(task => task.id ===
taskHistoryId);
        let name = 'Задача';
        setObjectName(name);
        setObjectTitle(task.title);
        setObjectDescription(task.description);
        setObjectDate(task.createdDate);
        setObjectStartDate(task.startExecutionDate);
        setObjectEndDate(task.deadlineDate);
        let target_project = task.projectId;
        const project = projectHistories.find(project => project.id ===
target_project);
        setObjectProject(project.title);
        let target_direction = project.actionDirectionId;
        const direction = directionHistories.find(direction =>
direction.id === target_direction);
        setObjectDirection(direction.title);
        let target_sphere = direction.lifeSphereId;
        const sphere = lifeSphereHistories.find(sphere => sphere.id ===
target_sphere);
        setObjectLifeSphere(sphere.title);
        openModal();
    };

    const handleDeleteTask = (taskHistoryId) => {
        const updatedHistories = taskHistories.filter(history =>
history.id !== taskHistoryId);
        setTaskHistories(updatedHistories);
```

```
        planDataUpdate(user.id);
    };

    const onDeleteTask = async (taskHistoryId) => {
        try {
            const response = await
axios.delete(`/api2/task/${taskHistoryId}`);
            if (response.data) {
                handleDeleteTask(taskHistoryId);
            } else {
                console.error('Failed to delete task history');
            }
        } catch (error) {
            console.error('An error occurred while trying to delete task
history:', error);
        }
    };

    function getCurrentTime(taskId) {
        let totalTimeInSeconds = 0;
        const task = taskHistories.find(task => task.id === taskId);
        if (!task) {
            return 0;
        }
        const times = task.realExecutionTimes || [];
        times.forEach(timeRecord => {
            const startDate = new Date(timeRecord.startDate);
            const endDate = new Date(timeRecord.endDate);
            const difference = endDate.getTime() - startDate.getTime();
            const differenceInSeconds = difference / 1000;
            totalTimeInSeconds += differenceInSeconds;
        });
        return totalTimeInSeconds;
    }


    return (
        <div>
            <button className="button_filter" onClick={() =>
setFilter('all')}>За всё время</button>
            <button className="button_filter" onClick={() =>
setFilter('today')}>Сегодня</button>
            <button className="button_filter" onClick={() =>
setFilter('yesterday')}>Вчера</button>
            <button className="button_filter" onClick={() =>
setFilter('range')}>
                Выбрать промежуток
            </button>
            {filter === 'range' && (
                <>
                    <input
                        type="date"
                        value={dateRange.startDate &&
!isNaN(dateRange.startDate.getTime()) ?
dateRange.startDate.toISOString().split('T')[0] : ''}
                        onChange={e => setDateRange({ ...dateRange,
startDate: new Date(e.target.value) })}
                    />
```

```
                    <input
                        type="date"
                        value={dateRange.endDate &&
!isNaN(dateRange.endDate.getTime()) ?
dateRange.endDate.toISOString().split('T')[0] : ''}
                        onChange={e => setDateRange({ ...dateRange,
endDate: new Date(e.target.value) })}
                    />
                </>
            )}
            <div className='info_block'>
                {filteredHistories.map((history, index) => (
                    <div className='info_box' key={index}>
                        <div className='info_box_content'>
                            <div className='setting_item'>
                                {history.title}
                            </div>
                        </div>
                        <div className='task_row_button'>
                            <Stopwatch
initialSeconds={getCurrentTime(history.id)} taskId={history.id} />
                            <div className='row_buttons_task'>
                                <button className='btn_show_planning'
onClick={() => onShowTask(history.id)}>
                                    <svg class="eye-icon"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round">
                                        <path d="M1,12s4-8,11-
8,11,8,11,8-4,8-11,8S1,12,1,12Z"></path>
                                        <circle cx="12" cy="12"
r="3"></circle>
                                    </svg>
                                </button>
                                <button className='btn_delete_planning'
onClick={() => onDeleteTask(history.id)}>
                                    <svg class="close-icon"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round">
                                        <line x1="18" y1="6" x2="6"
y2="18"></line>
                                        <line x1="6" y1="6" x2="18"
y2="18"></line>
                                    </svg>
                                </button>
                            </div>
                        </div>
                    </div>
                ))}
            </div>
            {isOpen && (
                <Modal onClose={closeModal}>
                    <h2>{objectName}</h2>
                    <div className='modal_content_planning'>
                        <div class="setting_item">Название:
<p>{objectTitle}</p></div>
```

```
                        <div
class="setting_item">Описание:<p>{objectDescription}</p></div>
                        <div
class="setting_item">Проект:<p>{objectProject}</p></div>
                        <div
class="setting_item">Направление:<p>{objectDirection}</p></div>
                        <div class="setting_item">Сфера
жизни:<p>{objectLifeSphere}</p></div>
                        <div class="setting_item">Дата создания:
<p>{formatDate(objectDate)}</p></div>
                        <div class="setting_item">Дата начала:
<p>{formatDate(objectStartDate)}</p></div>
                        <div class="setting_item">Дата завершения:
<p>{formatDate(objectEndDate)}</p></div>
                    </div>
                </Modal>
            )}
        </div>
    );
}

export default TaskDataComponent;
```

Листинг Д.8 – Файл Settings.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Modal from '../Modal';
import planDataUpdate from './PlanDataUpdate';
import SettingsDataComponent from './SettingsDataComponent';

function Settings() {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [isOpenLifeSphere, setIsOpenLifeSphere] = useState(false);
    const [isOpenDirection, setisOpenDirection] = useState(false);
    const [isOpenProject, setisOpenProject] = useState(false);
    const [isOpenGoal, setisOpenGoal] = useState(false);

    //Сфера жизни
    const [titleLifeSphere, setTitleLifeSphere] = useState('');
    const [descriptionLifeSphere, setDescriptionLifeSphere] =
useState('');
    //Направление
    const [titleDirection, setTitleDirection] = useState('');
    const [descriptionDirection, setDescriptionDirection] = useState('');
    const [lifeSphereDirection, setLifeSphereDirection] = useState('');
    //Проект
    const [titleProject, setTitleProject] = useState('');
    const [descriptionProject, setDescriptionProject] = useState('');
    const [startDateProject, setStartDateProject] = useState('');
    const [endDateProject, setEndDateProject] = useState('');
    const [directionProject, setDirectionProject] = useState('');
    //Цель
    const [titleGoal, setTitleGoal] = useState('');
    const [descriptionGoal, setDescriptionGoal] = useState('');
```

```
    const [startDateGoal, setStartDateGoal] = useState('');
    const [endDateGoal, setEndDateGoal] = useState('');
    const [directionGoal, setDirectionGoal] = useState('');

    const [errorMessage, setErrorMessage] = useState('');
    const [errorMessageTwo, setErrorMessageTwo] = useState('');
    const [errorMessageThree, setErrorMessageThree] = useState('');
    const [errorMessageFour, setErrorMessageFour] = useState('');

    const [settingsDataUpdated, setSettingsDataUpdated] =
useState(false);
    const [planData, setPlanData] =
useState(JSON.parse(localStorage.getItem('userPlanData')));

    const [lifeSphereHistories, setLifeSphereHistories] = useState([]);
    const [directionHistories, setDirectionHistories] = useState([]);

    const closeModalLifeSphere = () => {
        setIsOpenLifeSphere(false);
    };

    const closeModalDirection = () => {
        setisOpenDirection(false);
    };

    const closeModalProject = () => {
        setisOpenProject(false);
    };

    const closeModalGoal = () => {
        setisOpenGoal(false);
    };

    useEffect(() => {
        if (isOpenLifeSphere || isOpenDirection || isOpenProject ||
isOpenGoal) {
            setSettingsDataUpdated(false); // Сбрасываем флаг при
открытии модального окна
        } else {
            setSettingsDataUpdated(true); // Выставляем обновленное
значение при закрытии модального окна
        }
        const updateSettingsData = () => {
            const storedData = localStorage.getItem('userPlanData');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userPlanData = parsedData;
                    const lifeSphere = userPlanData || [];
                    const actionDirection = lifeSphere.map(l =>
l.actionDirections).flat() || [];
                    setLifeSphereHistories(lifeSphere);
                    setDirectionHistories(actionDirection);
                } else {
                    console.log('Parsed data is not an array or is
empty');
                }
            } else {
```

```
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
            if (event.key === 'userPlanData') {
                updateSettingsData();
            }
        };

        updateSettingsData();

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };

    }, [isOpenLifeSphere, isOpenDirection, isOpenProject, isOpenGoal,
lifeSphereHistories, directionHistories]);

    const openModalLifeSphere = () => {
        setIsOpenLifeSphere(true);
    };

    const openModalDirection = () => {
        setisOpenDirection(true);
    };

    const openModalProject = () => {
        setisOpenProject(true);
    };

    const openModalGoal = () => {
        setisOpenGoal(true);
    };

    const handleSubmitLifeShere = async (event) => {
        event.preventDefault();

        if (!titleLifeSphere || !descriptionLifeSphere) {
            setErrorMessage("Все поля должны быть заполнены!");
            return;
        }

        const userLifeSphereData = {
            UserId: user.id,
            CreatedDate: new Date(),
            Title: titleLifeSphere,
            Description: descriptionLifeSphere
        };

        try {
            const url = `/api2/life-sphere`;

            axios.post(url, userLifeSphereData, {
                headers: {
                    'Content-Type': 'application/json'
```

```
                }
            }).then(response => {
                setErrorMessage('Данные успешно добавлены!');
                planDataUpdate(user.id);
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessage(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessage('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };

    const handleSubmitDirection = async (event) => {
        event.preventDefault();

        if (!titleDirection || !descriptionDirection ||
!lifeSphereDirection) {
            setErrorMessageTwo("Все поля должны быть заполнены!");
            return;
        }

        const userDirectionData = {
            UserId: user.id,
            CreatedDate: new Date(),
            Title: titleDirection,
            Description: descriptionDirection,
            LifeSphereId: lifeSphereDirection
        };

        try {
            const url = `/api2//action-direction`;

            axios.post(url, userDirectionData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessageTwo('Данные успешно добавлены!');
                planDataUpdate(user.id);
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessageTwo(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessageTwo('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
```

```
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };

    const handleSubmitProject = async (event) => {
        event.preventDefault();

        if (!titleProject || !descriptionProject || !directionProject ||
!startDateProject || !endDateProject) {
            setErrorMessageThree("Все поля должны быть заполнены!");
            return;
        }

        const userProjectData = {
            UserId: user.id,
            CreatedDate: new Date(),
            Title: titleProject,
            Description: descriptionProject,
            ActionDirectionId: directionProject,
            StartExecutionDate: startDateProject,
            DeadlineDate: endDateProject
        };

        try {
            const url = `/api2/project`;

            axios.post(url, userProjectData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessageThree('Данные успешно добавлены!');
                planDataUpdate(user.id);
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessageThree(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessageThree('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };

    const handleSubmitGoal = async (event) => {
        event.preventDefault();

        if (!titleGoal || !descriptionGoal || !directionGoal ||
!startDateGoal || !endDateGoal) {
            setErrorMessageFour("Все поля должны быть заполнены!");
            return;
        }
```

```
        const userGoalData = {
            UserId: user.id,
            CreatedDate: new Date(),
            Title: titleGoal,
            Description: descriptionGoal,
            ActionDirectionId: directionGoal,
            StartExecutionDate: startDateGoal,
            DeadlineDate: endDateGoal
        };

        try {
            const url = `/api2/goal`;

            axios.post(url, userGoalData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessageFour('Данные успешно добавлены!');
                planDataUpdate(user.id);
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessageFour(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessageFour('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };


    return (
        <div>
            <h2>Планирование/Настройка системы</h2>
            <div className='row-btn'>
                <button onClick={openModalLifeSphere}>Добавить сферу
жизни</button>
                <button onClick={openModalDirection}>Добавить
направление</button>
                <button onClick={openModalProject}>Добавить
проект</button>
                <button onClick={openModalGoal}>Добавить цель</button>
            </div>
            {isOpenLifeSphere && (
                <Modal onClose={closeModalLifeSphere}>
                    <h2>Добавить сферу жизни</h2>
                    {errorMessage && <p
className="error">{errorMessage}</p>}
                    <form onSubmit={handleSubmitLifeShere}>
                        <div>
```

```
                            <label
htmlFor="titleLifeSphere">Название:</label>
                            <input
                                type="text"
                                id="titleLifeSphere"
                                value={titleLifeSphere}
                                onChange={(e) =>
setTitleLifeSphere(e.target.value)}
                            />
                        </div>
                        <div>
                            <label
htmlFor="descriptionLifeSphere">Описание:</label>
                            <input
                                type="text"
                                id="descriptionLifeSphere"
                                value={descriptionLifeSphere}
                                onChange={(e) =>
setDescriptionLifeSphere(e.target.value)}
                            />
                        </div>
                        <button type="submit">Добавить</button>
                    </form>
                </Modal>
            )}
            {isOpenDirection && (
                <Modal onClose={closeModalDirection}>
                    <h2>Добавить направление</h2>
                    {errorMessageTwo && <p
className="error">{errorMessageTwo}</p>}
                    <form onSubmit={handleSubmitDirection}>
                        <div>
                            <label
htmlFor="titleDirection">Название:</label>
                            <input
                                type="text"
                                id="titleDirection"
                                value={titleDirection}
                                onChange={(e) =>
setTitleDirection(e.target.value)}
                            />
                        </div>
                        <div>
                            <label
htmlFor="descriptionDirection">Описание:</label>
                            <input
                                type="text"
                                id="descriptionDirection"
                                value={descriptionDirection}
                                onChange={(e) =>
setDescriptionDirection(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="lifeSphereDirection">Сфера
жизни:</label>
                            <select value={lifeSphereDirection}
onChange={e => setLifeSphereDirection(e.target.value)}>
```

```
                                {lifeSphereHistories.map(sphere =>
<option key={sphere.id} value={sphere.id}>{sphere.title}</option>)}
                            </select>
                        </div>
                        <button type="submit">Добавить</button>
                    </form>
                </Modal>
            )}
            {isOpenProject && (
                <Modal onClose={closeModalProject}>
                    <h2>Добавить Проект</h2>
                    {errorMessageThree && <p
className="error">{errorMessageThree}</p>}
                    <form onSubmit={handleSubmitProject}>
                        <div>
                            <label
htmlFor="titleProject">Название:</label>
                            <input
                                type="text"
                                id="titleProject"
                                value={titleProject}
                                onChange={(e) =>
setTitleProject(e.target.value)}
                            />
                        </div>
                        <div>
                            <label
htmlFor="descriptionProject">Описание:</label>
                            <input
                                type="text"
                                id="descriptionProject"
                                value={descriptionProject}
                                onChange={(e) =>
setDescriptionProject(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="startDateProject">Планируемая
дата начала:</label>
                            <input
                                type="datetime-local"
                                id="startDateProject"
                                value={startDateProject}
                                onChange={(e) =>
setStartDateProject(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="endDateProject">Планируемая
дата завершения:</label>
                            <input
                                type="datetime-local"
                                id="endDateProject"
                                value={endDateProject}
                                onChange={(e) =>
setEndDateProject(e.target.value)}
                            />
                        </div>
```

```
                        <div>
                            <label
htmlFor="directionProject">Направление:</label>
                            <select value={directionProject} onChange={e
=> setDirectionProject(e.target.value)}>
                                {directionHistories.map(direction =>
<option key={direction.id}
value={direction.id}>{direction.title}</option>)}
                            </select>
                        </div>
                        <button type="submit">Добавить</button>
                    </form>
                </Modal>
            )}
            {isOpenGoal && (
                <Modal onClose={closeModalGoal}>
                    <h2>Добавить Цель</h2>
                    {errorMessageFour && <p
className="error">{errorMessageFour}</p>}
                    <form onSubmit={handleSubmitGoal}>
                        <div>
                            <label htmlFor="titleGoal">Название:</label>
                            <input
                                type="text"
                                id="titleGoal"
                                value={titleGoal}
                                onChange={(e) =>
setTitleGoal(e.target.value)}
                            />
                        </div>
                        <div>
                            <label
htmlFor="descriptionGoal">Описание:</label>
                            <input
                                type="text"
                                id="descriptionGoal"
                                value={descriptionGoal}
                                onChange={(e) =>
setDescriptionGoal(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="startDateGoal">Планируемая
дата начала:</label>
                            <input
                                type="datetime-local"
                                id="startDateGoal"
                                value={startDateGoal}
                                onChange={(e) =>
setStartDateGoal(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="endDateGoal">Планируемая дата
завершения:</label>
                            <input
                                type="datetime-local"
                                id="endDateGoal"
```

```
                                value={endDateGoal}
                                onChange={(e) =>
setEndDateGoal(e.target.value)}
                            />
                        </div>
                        <div>
                            <label
htmlFor="directionGoal">Направление:</label>
                            <select value={directionGoal} onChange={e =>
setDirectionGoal(e.target.value)}>
                                {directionHistories.map(direction =>
<option key={direction.id}
value={direction.id}>{direction.title}</option>)}
                            </select>
                        </div>
                        <button type="submit">Добавить</button>
                    </form>
                </Modal>
            )}
            <div className='history'>
                {settingsDataUpdated && <SettingsDataComponent />}
            </div>
        </div>
    );
};

export default Settings;
```

Листинг Д.9 – Файл SettingsDataComponent.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import planDataUpdate from './PlanDataUpdate';
import Modal from '../Modal';

function formatDate(dateString) {
    const options = { year: 'numeric', month: 'numeric', day: 'numeric'
};
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function SettingsDataComponent() {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [lifeSphereHistories, setLifeSphereHistor] = useState([]);
    const [directionHistories, setDirectionHistories] = useState([]);
    const [projectHistories, setProjectHistories] = useState([]);
    const [goalHistories, setGoalHistories] = useState([]);

    const [isOpenLifeSphere, setIsOpenLifeSphere] = useState(false);
    const [isOpenDirection, setisOpenDirection] = useState(false);
    const [isOpenProjectGoal, setisOpenProjectGoal] = useState(false);

    const [objectName, setObjectName] = useState(null);
    const [objectTitle, setObjectTitle] = useState(null);
    const [objectDescription, setObjectDescription] = useState(null);
```

```
    const [objectDate, setObjectDate] = useState(null);
    const [objectStartDate, setObjectStartDate] = useState(null);
    const [objectEndDate, setObjectEndDate] = useState(null);
    const [objectLifeSphere, setObjectLifeSphere] = useState(null);
    const [objectDirection, setObjectDirection] = useState(null);

    const closeModalLifeSphere = () => {
        setIsOpenLifeSphere(false);
    };

    const closeModalDirection = () => {
        setisOpenDirection(false);
    };

    const closeModalProjectGoal = () => {
        setisOpenProjectGoal(false);
    };

    const openModalLifeSphere = () => {
        setIsOpenLifeSphere(true);
    };

    const openModalDirection = () => {
        setisOpenDirection(true);
    };

    const openModalProjectGoal = () => {
        setisOpenProjectGoal(true);
    };

    useEffect(() => {
        const updateSettingsData = () => {
            const storedData = localStorage.getItem('userPlanData');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userPlanData = parsedData;
                    const lifeSphere = userPlanData || [];
                    const actionDirection = lifeSphere.map(l =>
l.actionDirections).flat() || [];
                    const project = actionDirection.map(l =>
l.projects).flat() || [];
                    const goal = actionDirection.map(l => l.goals).flat()
|| [];
                    setLifeSphereHistories(lifeSphere);
                    setDirectionHistories(actionDirection);
                    setProjectHistories(project);
                    setGoalHistories(goal);
                } else {
                    console.log('Parsed data is not an array or is
empty');
                }
            } else {
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
```

```
            if (event.key === 'userPlanData') {
                updateSettingsData();
            }
        };

        updateSettingsData();

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };

    }, [lifeSphereHistories, directionHistories, projectHistories,
goalHistories]);

    const handleDeleteLifeSphere = (lifeSphereHistoryId) => {
        const updatedHistories = lifeSphereHistories.filter(history =>
history.id !== lifeSphereHistoryId);
        setLifeSphereHistories(updatedHistories);
        planDataUpdate(user.id);
    };

    const onDeleteLifeSphere = async (lifeSphereHistoryId) => {
        try {
            const response = await axios.delete(`/api2/life-
sphere/${lifeSphereHistoryId}`);
            if (response.data) {
                handleDeleteLifeSphere(lifeSphereHistoryId);
            } else {
                console.error('Failed to delete life sphere history');
            }
        } catch (error) {
            console.error('An error occurred while trying to delete life
sphere history:', error);
        }
    };

    const handleDeleteDirection = (directionHistoryId) => {
        const updatedHistories = directionHistories.filter(history =>
history.id !== directionHistoryId);
        setDirectionHistories(updatedHistories);
        planDataUpdate(user.id);
    };

    const onDeleteDirection = async (directionHistoryId) => {
        try {
            const response = await axios.delete(`/api2/action-
direction/${directionHistoryId}`);
            if (response.data) {
                handleDeleteDirection(directionHistoryId);
            } else {
                console.error('Failed to delete direction history');
            }
        } catch (error) {
            console.error('An error occurred while trying to delete
direction history:', error);
        }
```

```
    };

    const handleDeleteProject = (projectHistoryId) => {
        const updatedHistories = projectHistories.filter(history =>
history.id !== projectHistoryId);
        setProjectHistories(updatedHistories);
        planDataUpdate(user.id);
    };

    const onDeleteProject = async (projectHistoryId) => {
        try {
            const response = await
axios.delete(`/api2/project/${projectHistoryId}`);
            if (response.data) {
                handleDeleteProject(projectHistoryId);
            } else {
                console.error('Failed to delete project history');
            }
        } catch (error) {
            console.error('An error occurred while trying to delete
project history:', error);
        }
    };

    const handleDeleteGoal = (goalHistoryId) => {
        const updatedHistories = goalHistories.filter(history =>
history.id !== goalHistoryId);
        setGoalHistories(updatedHistories);
        planDataUpdate(user.id);
    };

    const onDeleteGoal = async (goalHistoryId) => {
        try {
            const response = await
axios.delete(`/api2/goal/${goalHistoryId}`);
            if (response.data) {
                handleDeleteGoal(goalHistoryId);
            } else {
                console.error('Failed to delete goal history');
            }
        } catch (error) {
            console.error('An error occurred while trying to delete goal
history:', error);
        }
    };

    const onShowLifeSphere = (lifeSphereHistoryId) => {
        const item = lifeSphereHistories.find(item => item.id ===
lifeSphereHistoryId);
        let name = 'Сфера жизни';
        setObjectName(name);
        setObjectTitle(item.title);
        setObjectDescription(item.description);
        setObjectDate(item.createdDate);
        openModalLifeSphere();
    };

    const onShowDirection = (directionHistoryId) => {
```

```
        const item = directionHistories.find(item => item.id ===
directionHistoryId);
        let name = 'Направление';
        setObjectName(name);
        setObjectTitle(item.title);
        setObjectDescription(item.description);
        setObjectDate(item.createdDate);
        let target_id = item.lifeSphereId;
        const item2 = lifeSphereHistories.find(item => item.id ===
target_id);
        setObjectLifeSphere(item2.title);
        openModalDirection();
    };

    const onShowProject = (projectHistoryId) => {
        const item = projectHistories.find(item => item.id ===
projectHistoryId);
        let name = 'Проект';
        setObjectName(name);
        setObjectTitle(item.title);
        setObjectDescription(item.description);
        setObjectDate(item.createdDate);
        setObjectStartDate(item.startExecutionDate);
        setObjectEndDate(item.deadlineDate);
        let target_id = item.actionDirectionId;
        console.log(target_id);
        const item2 = directionHistories.find(item => item.id ===
target_id);
        setObjectDirection(item2.title);
        openModalProjectGoal();
    };

    const onShowGoal = (goalHistoryId) => {
        const item = goalHistories.find(item => item.id ===
goalHistoryId);
        let name = 'Цель';
        setObjectName(name);
        setObjectTitle(item.title);
        setObjectDescription(item.description);
        setObjectDate(item.createdDate);
        setObjectStartDate(item.startExecutionDate);
        setObjectEndDate(item.deadlineDate);
        let target_id = item.actionDirectionId;
        const item2 = directionHistories.find(item => item.id ===
target_id);
        setObjectDirection(item2.title);
        openModalProjectGoal();
    };

    return (
        <div className='planning'>
            <div className='info_box'>
                <h3>Сферы жизни</h3>
                <div className="planning_row">
                    {lifeSphereHistories.map((history, index) => (
                        <div className='info_box_planning' key={index}>
                            <div class="info_box_data" key={index}>
                                <p>{history.title}</p>
```

```
                                <button className='btn_show_planning'
onClick={() => onShowLifeSphere(history.id)}>
                                </button>
                                <button className='btn_delete_planning'
onClick={() => onDeleteLifeSphere(history.id)}>
                                </button>
                            </div>
                        </div>
                    ))}
                </div>
            </div>
            <div className='info_box'>
                <h3>Направления</h3>
                <div className="planning_row">
                    {directionHistories.map((history, index) => (
                        <div className='info_box_planning' key={index}>
                            <div class="info_box_data" key={index}>
                                <p>{history.title}</p>
                                <button className='btn_show_planning'
onClick={() => onShowDirection(history.id)}>
                                    <svg class="eye-icon"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round">
                                        <path d="M1,12s4-8,11-
8,11,8,11,8-4,8-11,8S1,12,1,12Z"></path>
                                        <circle cx="12" cy="12"
r="3"></circle>
                                    </svg>
                                </button>
                                <button className='btn_delete_planning'
onClick={() => onDeleteDirection(history.id)}>
                                </button>
                            </div>
                        </div>
                    ))}
                </div>
            </div>
            <div className='info_box'>
                <h3>Проекты</h3>
                <div className="planning_row">
                    {projectHistories.map((history, index) => (
                        <div className='info_box_planning' key={index}>
                            <div class="info_box_data" key={index}>
                                <p>{history.title}</p>
                                <button className='btn_show_planning'
onClick={() => onShowProject(history.id)}>
                                </button>
                                <button className='btn_delete_planning'
onClick={() => onDeleteProject(history.id)}>
                                </button>
                            </div>
                        </div>
                    ))}
                </div>
            </div>
            <div className='info_box'>
                <h3>Цели</h3>
```

```
                <div className="planning_row">
                    {goalHistories.map((history, index) => (
                        <div class="info_box_data" key={index}>
                            <p>{history.title}</p>
                            <button className='btn_show_planning'
onClick={() => onShowGoal(history.id)}>
                                <svg class="eye-icon"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round">
                                    <path d="M1,12s4-8,11-8,11,8,11,8-
4,8-11,8S1,12,1,12Z"></path>
                                    <circle cx="12" cy="12"
r="3"></circle>
                                </svg>
                            </button>
                            <button className='btn_delete_planning'
onClick={() => onDeleteGoal(history.id)}>
                                <svg class="close-icon"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round">
                                    <line x1="18" y1="6" x2="6"
y2="18"></line>
                                    <line x1="6" y1="6" x2="18"
y2="18"></line>
                                </svg>
                            </button>
                        </div>
                    ))}
                </div>
            </div>
            {isOpenLifeSphere && (
                <Modal onClose={closeModalLifeSphere}>
                    <h2>{objectName}</h2>
                    <div className='modal_content_planning'>
                        <div class="setting_item">Название:
<p>{objectTitle}</p></div>
                        <div
class="setting_item">Описание:<p>{objectDescription}</p></div>
                        <div class="setting_item">Дата создания:
<p>{formatDate(objectDate)}</p></div>
                    </div>
                </Modal>
            )}
            {isOpenDirection && (
                <Modal onClose={closeModalDirection}>
                    <h2>{objectName}</h2>
                    <div className='modal_content_planning'>
                        <div class="setting_item">Название:
<p>{objectTitle}</p></div>
                        <div
class="setting_item">Описание:<p>{objectDescription}</p></div>
                        <div class="setting_item">Сфера
жизни:<p>{objectLifeSphere}</p></div>
                        <div class="setting_item">Дата создания:
<p>{formatDate(objectDate)}</p></div>
                    </div>
```

```
                </Modal>
            )}
            {isOpenProjectGoal && (
                <Modal onClose={closeModalProjectGoal}>
                    <h2>{objectName}</h2>
                    <div className='modal_content_planning'>
                        <div class="setting_item">Название:
<p>{objectTitle}</p></div>
                        <div
class="setting_item">Описание:<p>{objectDescription}</p></div>
                        <div
class="setting_item">Направление:<p>{objectDirection}</p></div>
                        <div class="setting_item">Дата создания:
<p>{formatDate(objectDate)}</p></div>
                        <div class="setting_item">Дата начала:
<p>{formatDate(objectStartDate)}</p></div>
                        <div class="setting_item">Дата завершения:
<p>{formatDate(objectEndDate)}</p></div>
                    </div>
                </Modal>
            )}
        </div>
    );
}

export default SettingsDataComponent;
```

Листинг Д.12 – Файл Stopwatch.jsx

```
import React, { useState, useEffect } from 'react';
import planDataUpdate from './PlanDataUpdate';
import axios from 'axios';
import { formatISO } from 'date-fns';

function Stopwatch({ initialSeconds = 0, taskId = 0 }) {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [seconds, setSeconds] = useState(initialSeconds);
    const [task, setTask] = useState(taskId);
    const [isActive, setIsActive] = useState(false);
    const [startDateTask, setStartDateTask] = useState('');
    const [endDateTask, setEndDateTask] = useState('');

    useEffect(() => {
        let intervalId = null;

        if (isActive) {
            intervalId = setInterval(() => {
                setSeconds(prevSeconds => prevSeconds + 1);
            }, 1000);
        }

        return () => {
```

```
            if (intervalId) clearInterval(intervalId);
        };
    }, [isActive]);

    const handleStartTimeTask = async () => {
        const startDate = new Date();
        const formattedStartDate =
startDate.toISOString().replace(/\.\d{3}/, '');
        setStartDateTask(formattedStartDate);
    };

    const handleEndTimeTask = async () => {
        const endDate = new Date();
        const formattedEndDate = endDate.toISOString().replace(/\.\d{3}/,
'');

        const userTimeTaskData = {
            TaskId: task,
            StartDate: startDateTask,
            EndDate: formattedEndDate,
        };

        try {
            const url = "/api2/real-execution-time";

            const response = await axios.post(url, userTimeTaskData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            });

            if (response.status === 200) {
                planDataUpdate(user.id);
            } else {
                throw new Error("Некорректный ответ сервера");
            }
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            if (error.response && error.response.data) {
                console.error('Детали ошибки:', error.response.data);
            }
            console.log("Отправляемые данные:", userTimeTaskData);
        }
    };

    const handleStartStop = () => {
        if (!isActive) {
            setSeconds(initialSeconds);
            handleStartTimeTask();
        } else {
            handleEndTimeTask();
        }
        setIsActive(!isActive);
    };

    const formatTime = () => {
        const hours = Math.floor(seconds / 3600);
        const minutes = Math.floor((seconds % 3600) / 60);
```

```
        const secondsRemain = seconds % 60;
        return `${hours.toString().padStart(2,
'0')}:${minutes.toString().padStart(2,
'0')}:${secondsRemain.toString().padStart(2, '0')}`;
    };

    return (
        <div className='timer'>
            <h1>{formatTime()}</h1>
            <button onClick={handleStartStop}>{isActive ? <svg
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="currentColor" viewBox="0 0 24 24">
                <rect x="6" y="5" width="4" height="14" rx="2" />
                <rect x="14" y="5" width="4" height="14" rx="2" />
            </svg>
                : <svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" fill="currentColor" viewBox="0 0 24 24">
                    <path d="M8 5l11 7-11 7V5z" />
                </svg>
            }</button>
        </div>
    );
}

export default Stopwatch;
```

Листинг Д.13 – Файл Calendars.jsx

```
import React, { useState, useEffect } from 'react';
import MyCalendar from './MyCalendar';
import moment from 'moment';
import Modal from '../Modal';
import 'moment/locale/ru';
moment.locale('ru');

const Calendars = () => {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [isActive, setIsActive] = useState(false);

    const toggleClass = () => {
        setIsActive(!isActive);
    };

    //Планируемое время
    const [titleDirection, setTitleDirection] = useState('');
    const [datePlan, setDatePlan] = useState('');
    const [hourAmmountPlan, setHourAmmountPlan] = useState('');

    const [isOpen, setIsOpen] = useState(false);
    const [errorMessage, setErrorMessage] = useState('');
    const [directionHistories, setDirectionHistories] = useState([]);

    const closeModal = () => {
        setIsOpen(false);
    };
```

```
    const openModal = () => {
        setIsOpen(true);
    };

    useEffect(() => {
        const updateData = () => {
            const storedData = localStorage.getItem('userPlanData');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userPlanData = parsedData;
                    const lifeSphere = userPlanData || [];
                    const actionDirection = lifeSphere.map(l =>
l.actionDirections).flat() || [];
                    setDirectionHistories(actionDirection);
                } else {
                    console.log('Parsed data is not an array or is
empty');
                }
            } else {
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
            if (event.key === 'userPlanData') {
                updateData();
            }
        };

        updateData();

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };

    }, [directionHistories]);

    const formattedEventsFact = events.map(event => ({
        ...event,
        start: moment(event.start, "DD.MM.YYYY").toDate(),
        end: moment(event.start, "DD.MM.YYYY").add(event.duration,
'hours').toDate()
    }));

    const formattedEventsPlan = eventsPlan.map(event => ({
        ...event,
        start: moment(event.start, "DD.MM.YYYY").toDate(),
        end: moment(event.start, "DD.MM.YYYY").add(event.duration,
'hours').toDate()
    }));

    const handleSubmit = async (event) => {
        event.preventDefault();
```

```
        if (!titleDirection || !datePlan || !hourAmmountPlan) {
            setErrorMessage("Все поля должны быть заполнены!");
            return;
        }

        const userPlanData = {
            UserId: user.id,
            Title: titleDirection,
            DatePlan: datePlan,
            HourAmmountPlan: hourAmmountPlan,
        };

        var savedPlans = JSON.parse(localStorage.getItem('userPlans')) ||
[];
        savedPlans.push(userPlanData);
        localStorage.setItem('userPlans', JSON.stringify(savedPlans));

    };

    return (
        <div>
            <h2 className='calendars'>Планирование/Календари</h2>
            <div className={isActive ? 'calendar_row active' :
'calendar_row'}>
                <div className='first_calendar'>
                    <h2>Календарь с фактически затраченным временем</h2>
                    <div className='calendar'>
                        <div style={{ width: '80%', margin: '50px auto'
}}>
                            <MyCalendar events={formattedEventsFact} />
                        </div>
                        <button className="button_show_calendar"
onClick={toggleClass}>
                            <p>Календарь с запланированным временем</p>
                            <svg className='arrow_next'
xmlns="http://www.w3.org/2000/svg" width="30" height="30" viewBox="0 0 24
24" fill="none" stroke="rgb(21 71 132)" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round" transform="rotate(180, 12, 12)">
                                <path d="M9 18l6-6-6-6" />
                            </svg>
                        </button>
                    </div>
                </div>
                <div className='second_calendar'>
                    <h2>Календарь с запланированным временем
                        <button className='add_plan'
onClick={openModal}>Добавить запланированное время</button>
                    </h2>
                    <div className='calendar'>
                        <div style={{ width: '80%', margin: '50px auto'
}}>
                            <MyCalendar events={formattedEventsPlan} />
                        </div>
                        <button className="button_show_calendar"
onClick={toggleClass}>
                            <p>Календарь с фактически затраченным
временем</p>
```

```
                            <svg className='arrow_prev'
xmlns="http://www.w3.org/2000/svg" width="30" height="30" viewBox="0 0 24
24" fill="none" stroke="rgb(21 71 132)" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round" transform="rotate(180, 12, 12)">
                                <path d="M9 18l6-6-6-6" />
                            </svg>
                        </button>
                    </div>
                </div>
            </div>
            {isOpen && (
                <Modal onClose={closeModal}>
                    <h2>Добавить плвнируемое время по направлению</h2>
                    {errorMessage && <p
className="error">{errorMessage}</p>}
                    <form onSubmit={handleSubmit}>
                        <div>
                            <label
htmlFor="titleDirection">Направление:</label>
                            <select value={titleDirection} onChange={e =>
setTitleDirection(e.target.value)}>
                                {directionHistories.map(direction =>
<option key={direction.title}
value={direction.title}>{direction.title}</option>)}
                            </select>
                        </div>
                        <div>
                            <label htmlFor="datePlan">Дата
планирования:</label>
                            <input
                                type="datetime-local"
                                id="datePlan"
                                value={datePlan}
                                onChange={(e) =>
setDatePlan(e.target.value)}
                            />
                        </div>
                        <div>
                            <label htmlFor="hourAmmountPlan">Планируемое
количество часов:</label>
                            <input
                                type="number"
                                id="hourAmmountPlan"
                                value={hourAmmountPlan}
                                onChange={(e) =>
setHourAmmountPlan(e.target.value)}
                            />
                        </div>
                        <button type="submit">Добавить</button>
                    </form>
                </Modal>
            )}
        </div>
    );
};

export default Calendars;
```

Листинг Д.14 – Функция filterHistories

```
const filterHistories = () => {
     const today = new Date();
     today.setHours(0, 0, 0, 0);
     const yesterday = new Date(today);
     yesterday.setDate(yesterday.getDate() - 1);

     switch (filter) {
       case 'today':
           return taskHistories.filter(history =>
               new Date(history.createdDate).setHours(0, 0, 0, 0) ===
today.getTime());
       case 'yesterday':
           return taskHistories.filter(history =>
               new Date(history.createdDate).setHours(0, 0, 0, 0) ===
yesterday.getTime());
       case 'range':
           return taskHistories.filter(history => {
               const historyDate = new Date(history.createdDate);
               return historyDate >= dateRange.startDate && historyDate
<= dateRange.endDate;
                });
       case 'all':
       default:
       return taskHistories;
     }
};
```

Приложение Е

(рекомендуемое)

Результаты функционального тестирования

Рисунок Е.1 – Отображение настроек системы

Рисунок Е.2 – Отображение задач

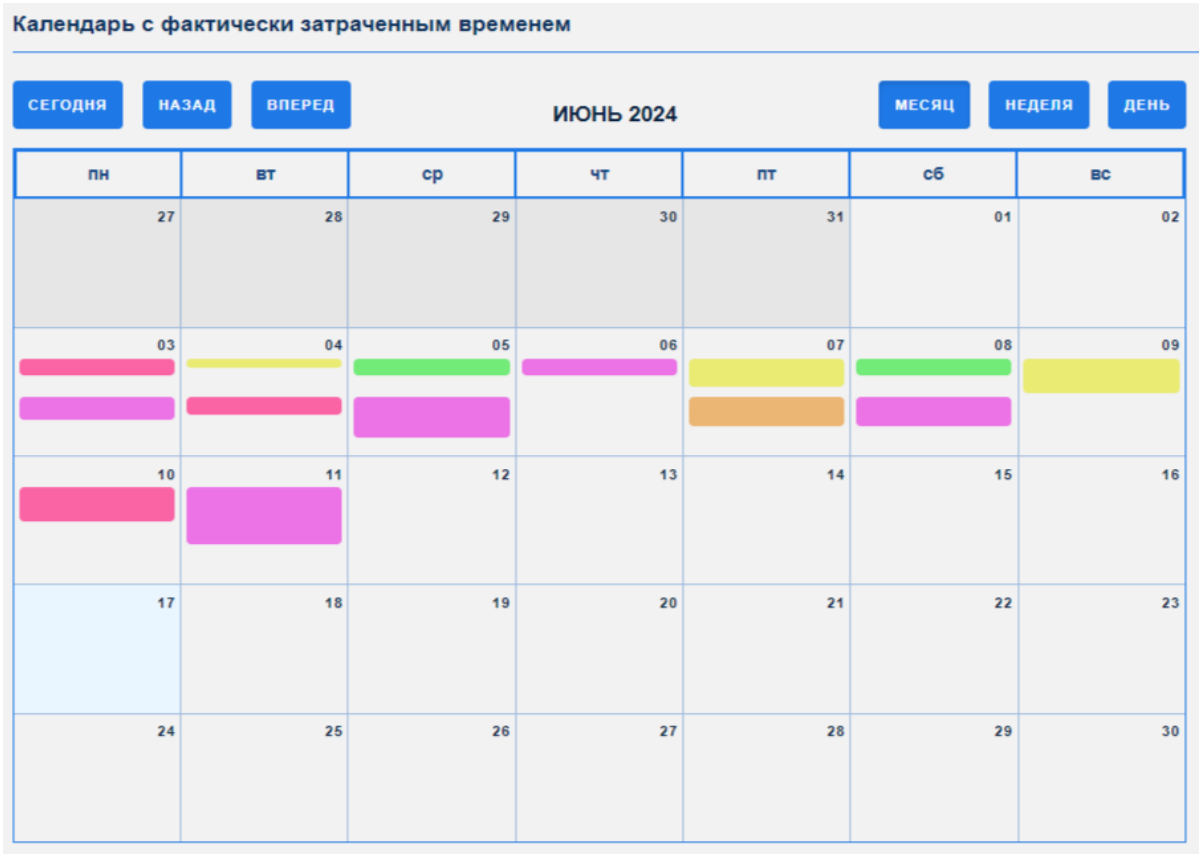

Рисунок Е.3 – Отображение затраченного времени в календаре

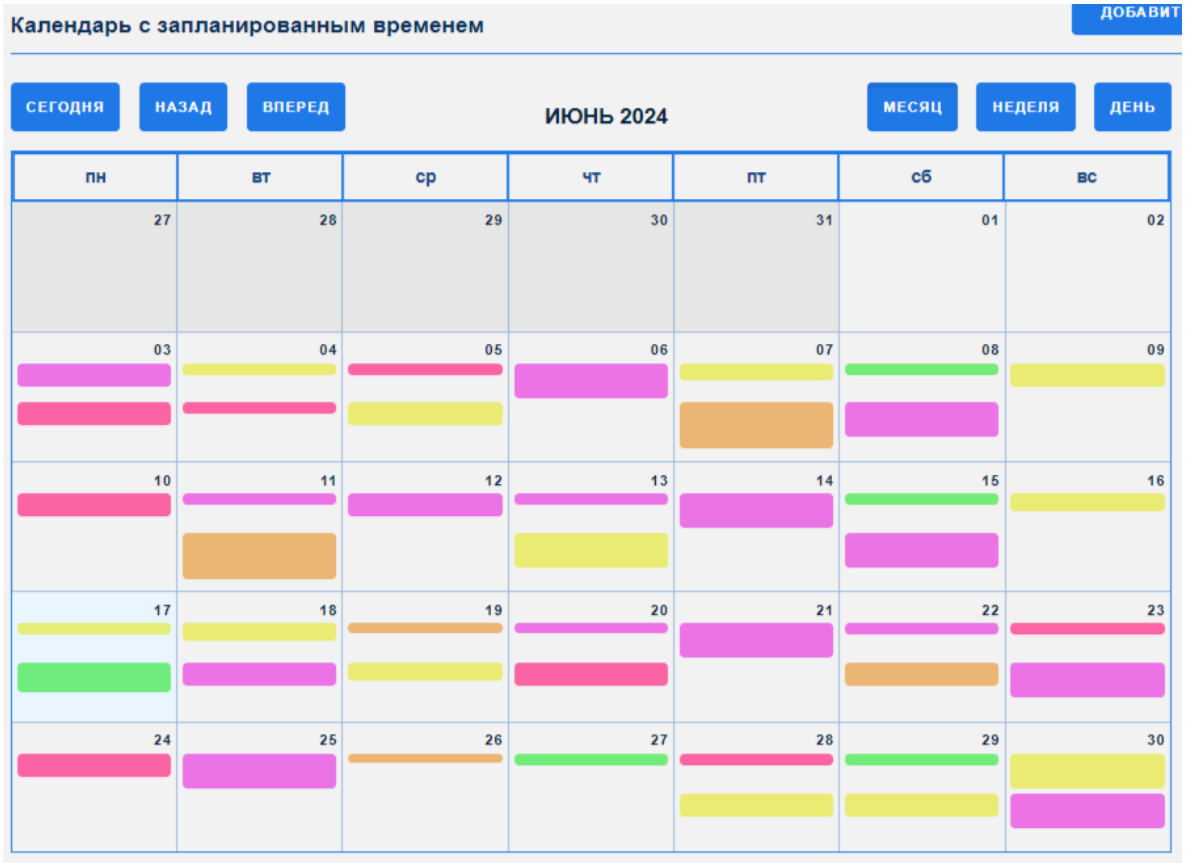

Рисунок Е.4 – Отображение запланированного времени в календаре

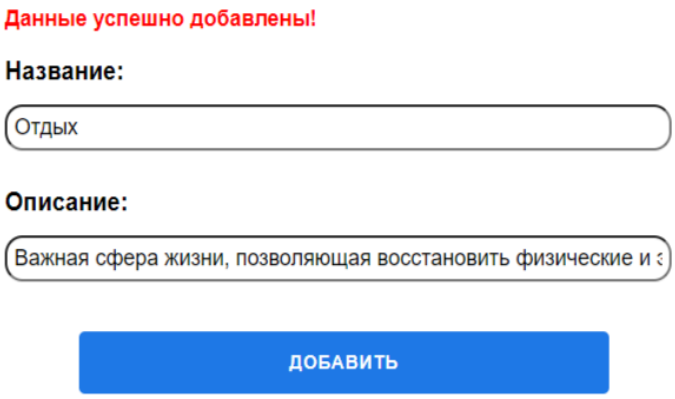

Рисунок Е.5 – Добавление сферы жизни

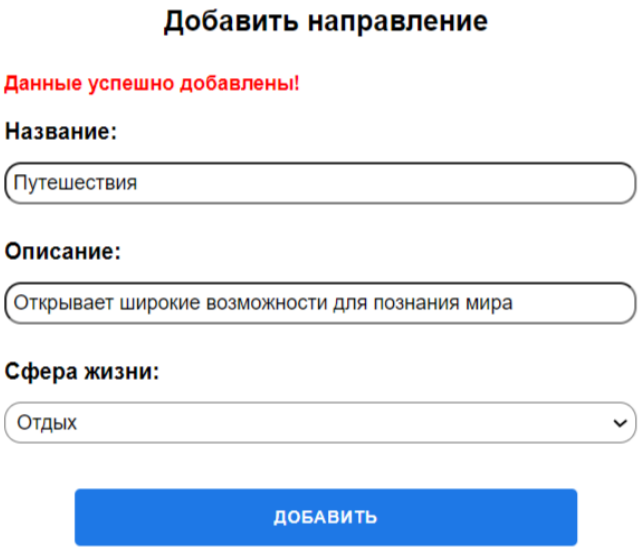

Рисунок Е.6 – Добавление направления

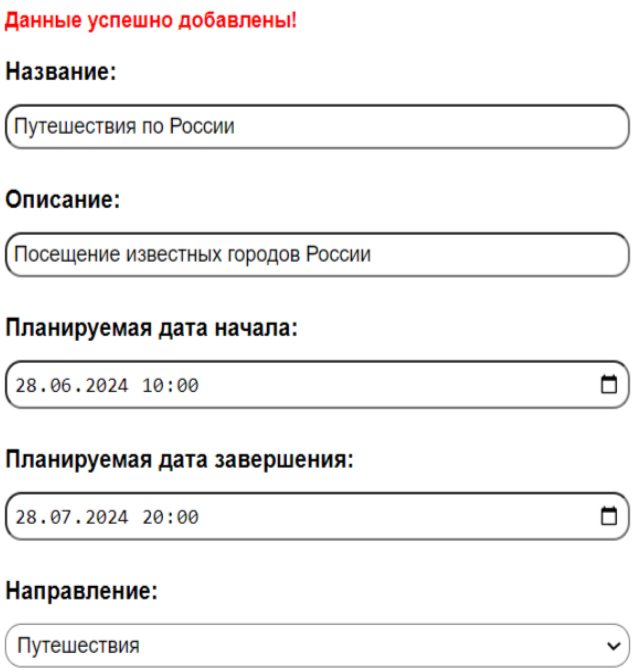

Рисунок Е.7 – Добавление проекта

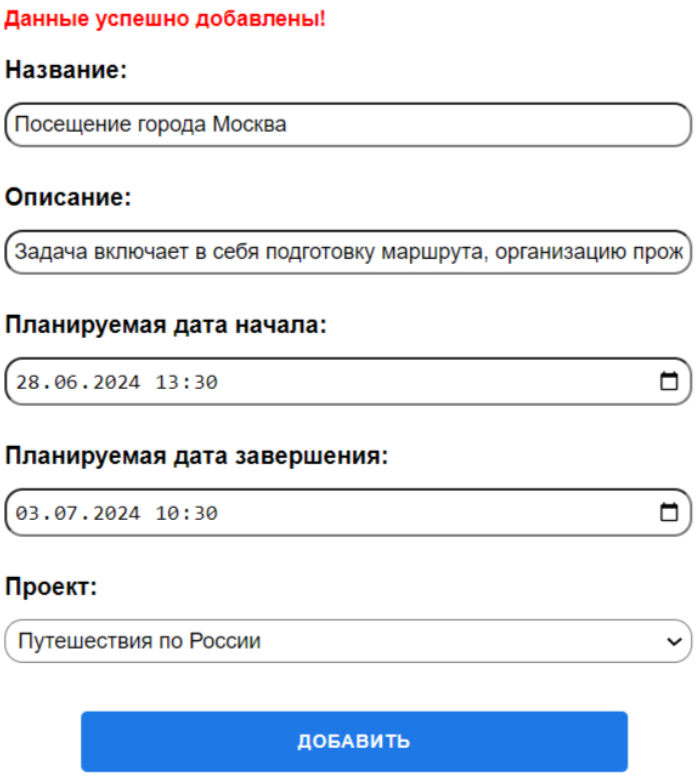

Рисунок Е.8 – Добавление задачи

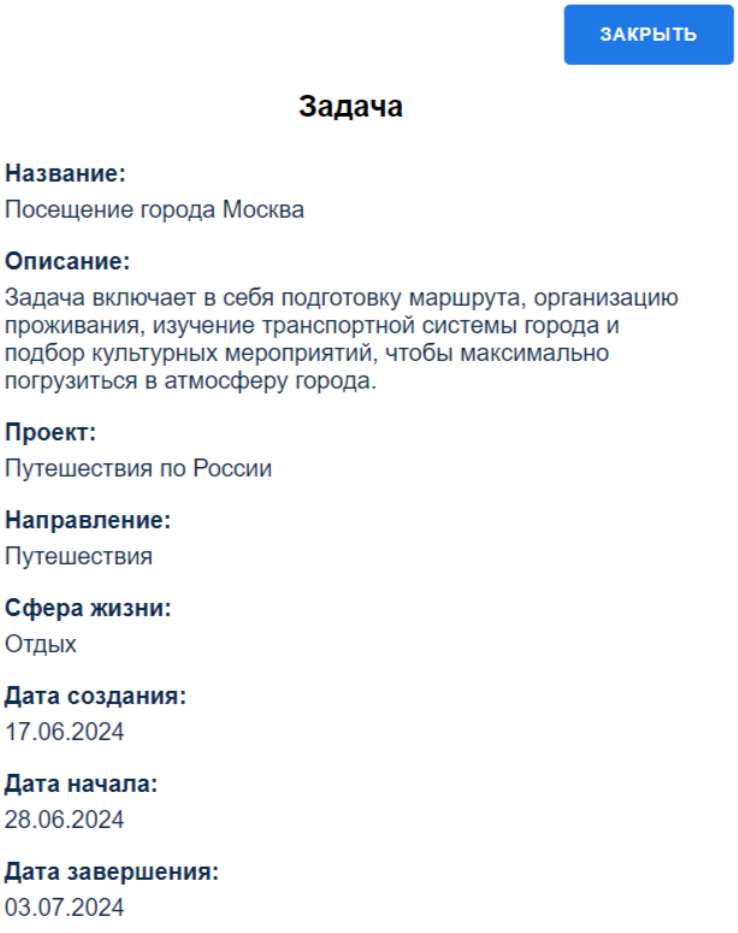

Рисунок Е.9 – Просмотр детальной информации о задаче

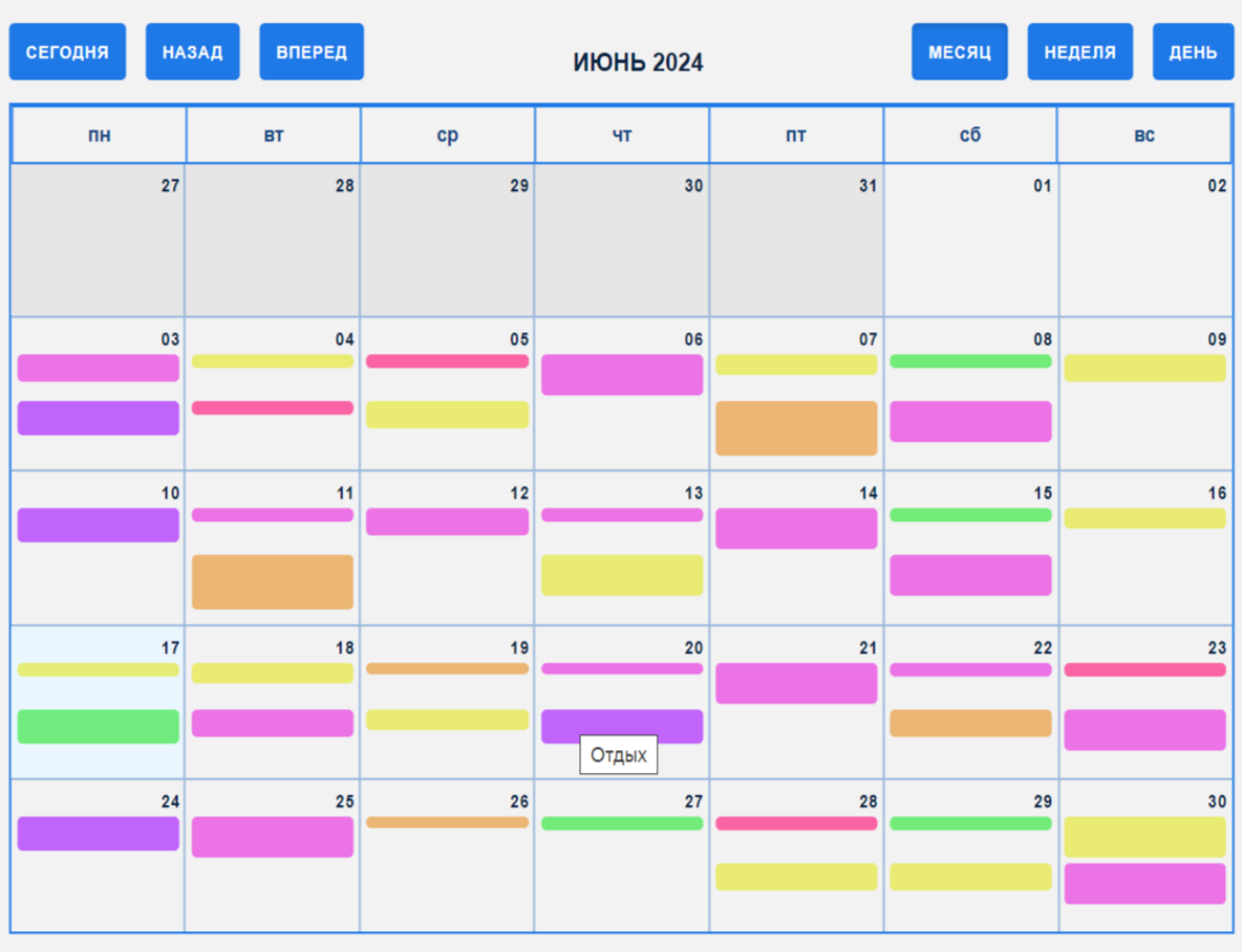

Рисунок Е.10 – Добавление запланированного времени

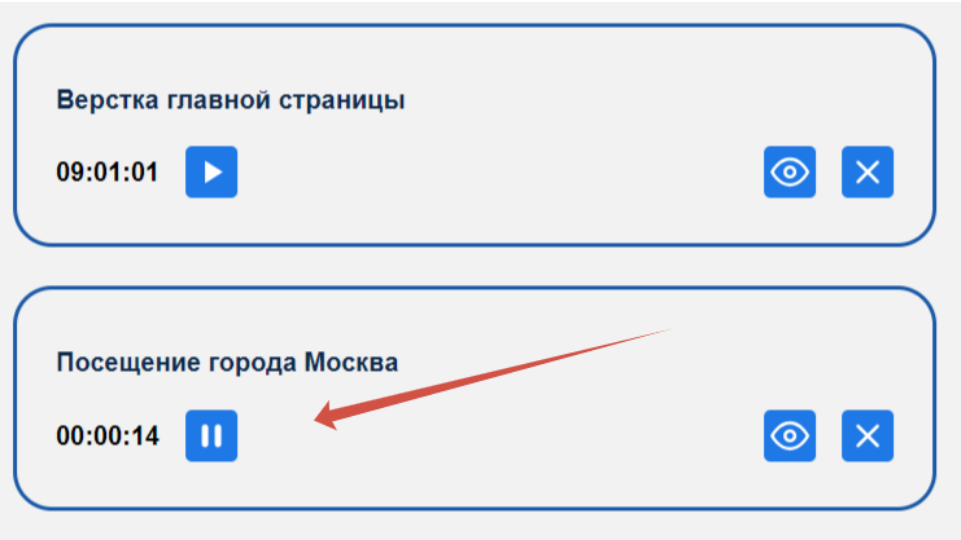

Рисунок Е.11 – Проверка отслеживания времени выполнения задачи

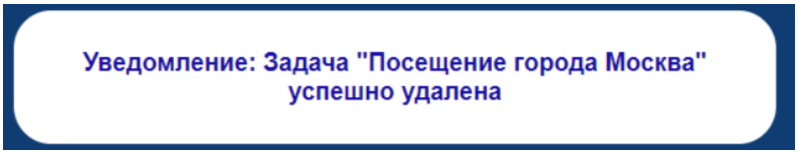

Рисунок Е.12 – Проверка удаления задачи

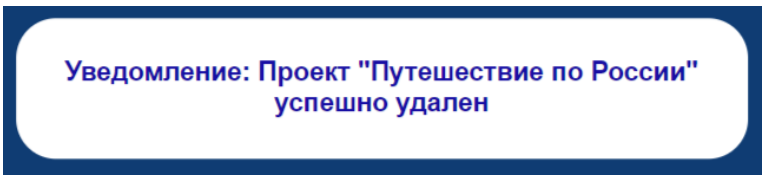

Рисунок Е.13 – Проверка удаления проекта

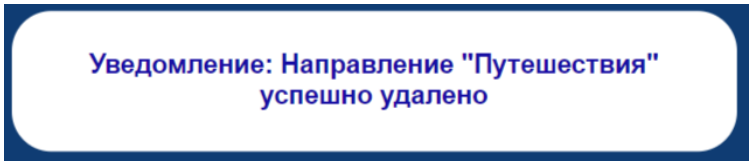

Рисунок Е.14 – Проверка удаления направления

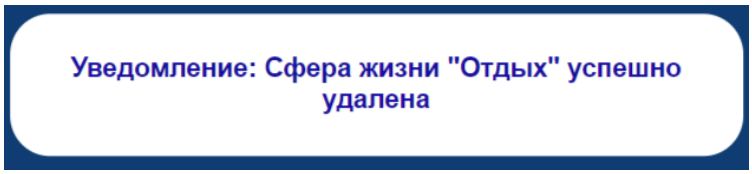

Рисунок Е.15 – Проверка удаления сферы жизни

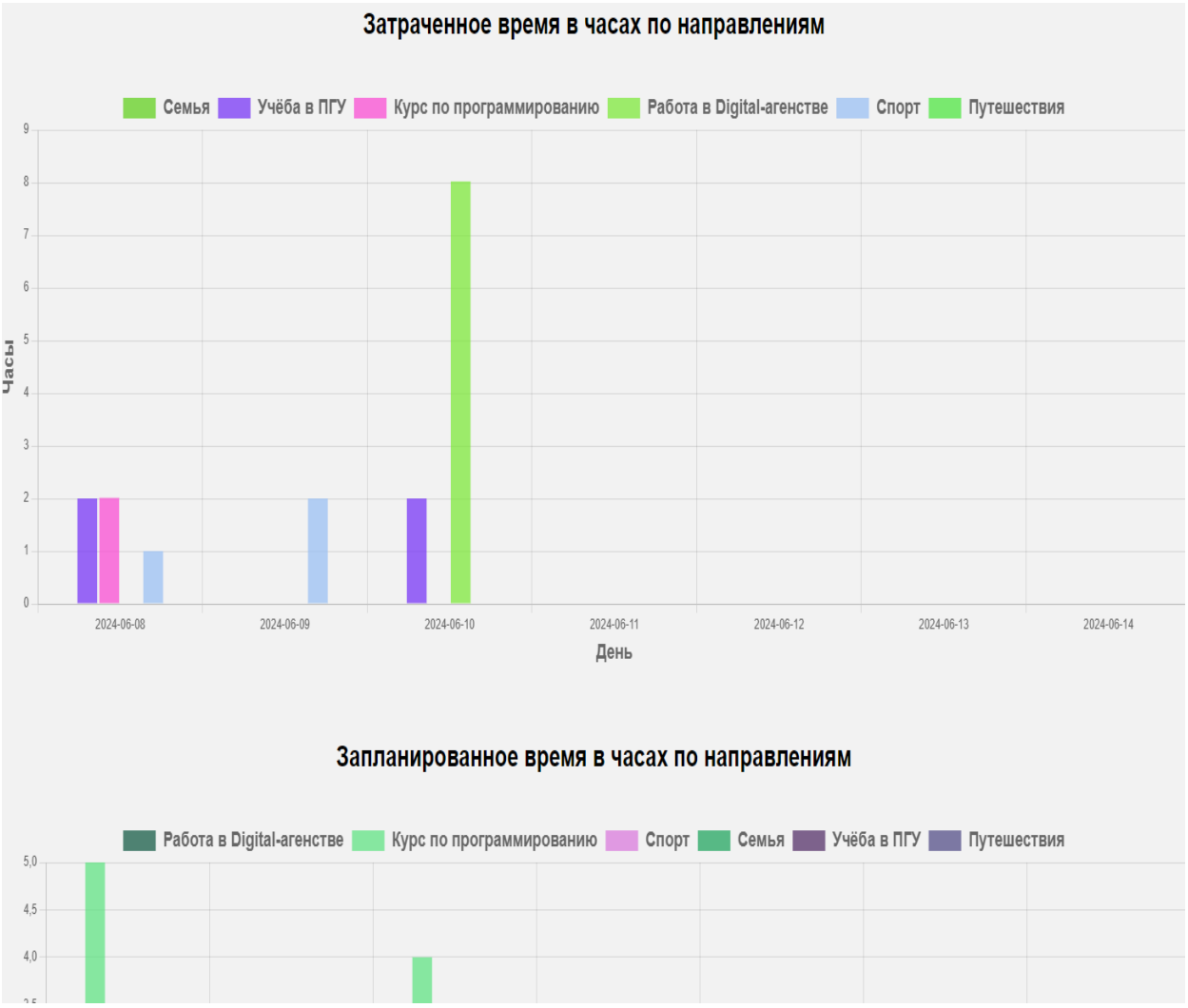

Рисунок Е.16 – Проверка отображения отчета

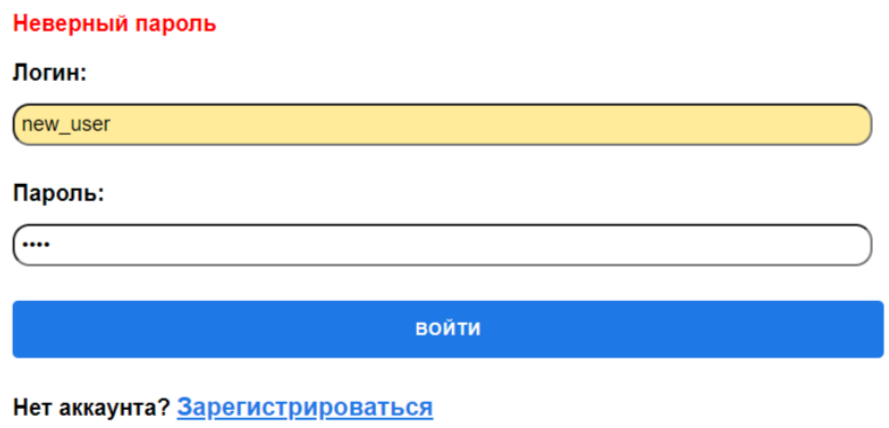

Рисунок Е.17 – Авторизация пользователя с неверным паролем

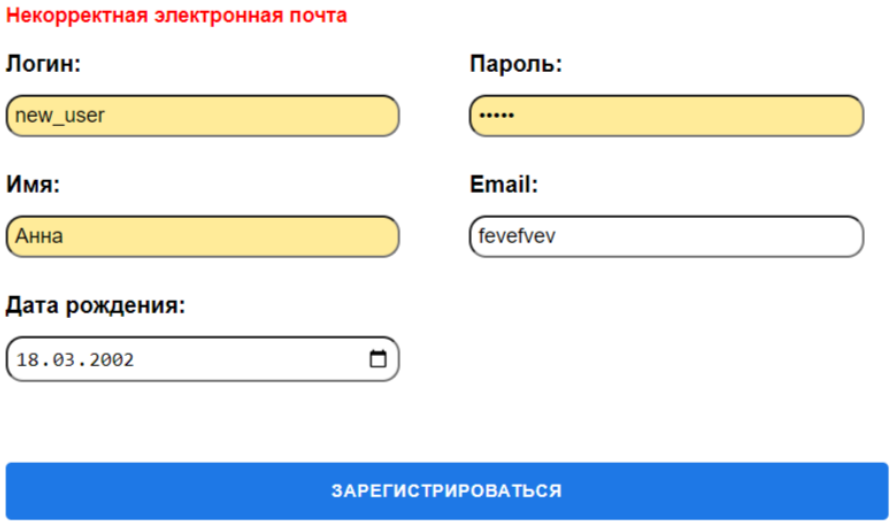

Рисунок Е.18 – Регистрация с некорректной почтой

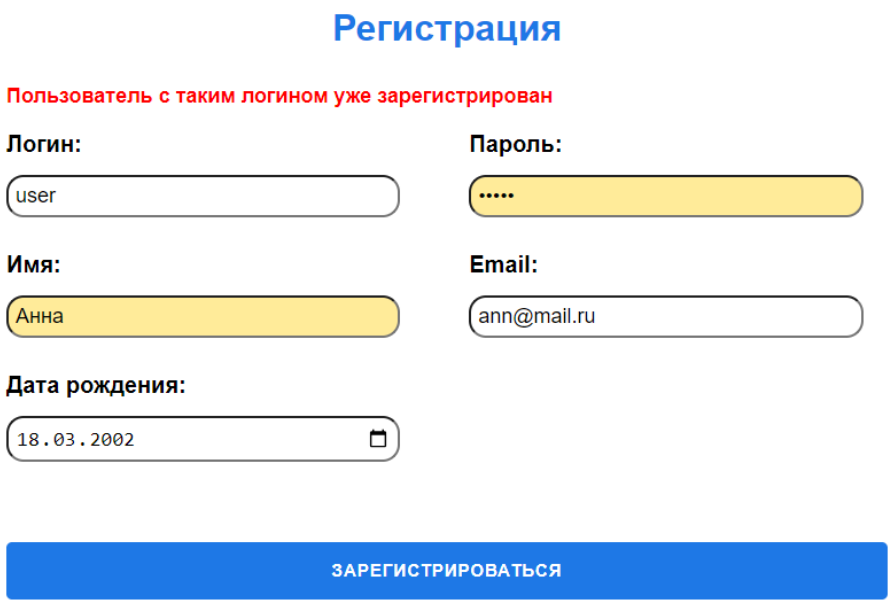

Рисунок Е.19 – Повторная регистрация пользователя

Приложение Ж

(рекомендуемое)

Листинг модульных тестов

Листинг Ж.1 – Файл Tasks.test.js

```
import React from 'react';
import { shallow } from 'enzyme';
import Task from '../components/planning/Tasks';

describe('Task', () => {
    let wrapper;

    beforeEach(() => {
        const mockTask = {
            id: 1,
            title: 'Тестовая задача',
            description: 'Это тестовое описание задачи',
            startExecutionDate: '2023-05-01T10:00:00',
            deadlineDate: '2023-05-05T18:00:00',
            projectName: 'Тестовый проект',
        };

        wrapper = shallow(<Task task={mockTask} />);
    });

    it('Должен рендериться компонент Task', () => {
        expect(wrapper.exists()).toBe(true);
    });

    it('Должен отображаться правильный заголовок задачи', () => {
        expect(wrapper.find('h3').text()).toEqual('Тестовая задача');
    });

    it('Должно отображаться правильное описание задачи', () => {
        expect(wrapper.find('.task-description').text()).toEqual('Это
тестовое описание задачи');
    });

    it('Должна отображаться правильная дата начала задачи', () => {
        expect(wrapper.find('.task-start-date').text()).toEqual('Дата
начала: 01.05.2023');
    });

    it('Должна отображаться правильная дата окончания задачи', () => {
        expect(wrapper.find('.task-end-date').text()).toEqual('Дата
окончания: 05.05.2023');
    });

    it('Должно отображаться правильное название проекта', () => {
        expect(wrapper.find('.task-project').text()).toEqual('Проект:
Тестовый проект');
    });
});
```

Листинг Ж.2 – Файл TaskDataComponent.test

```
import React from 'react';
import { shallow, mount } from 'enzyme';
import axios from 'axios';
import TaskDataComponent from
'../components/planning/TaskDataComponent';
```

```
jest.mock('axios');

describe('TaskDataComponent', () => {
    let wrapper;

    beforeEach(() => {
        wrapper = shallow(<TaskDataComponent />);
    });

    afterEach(() => {
        jest.clearAllMocks();
    });

    it('Должен рендериться TaskDataComponent', () => {
        expect(wrapper.exists()).toBe(true);
    });

    it('Должны отображаться кнопки фильтров', () => {

expect(wrapper.find('.button_filter').length).toBeGreaterThan(0);
    });

    it('Должны отображаться поля ввода диапазона дат при нажатии на
кнопку "Выбрать промежуток"', () => {
        wrapper.find('.button_filter').at(2).simulate('click');
        expect(wrapper.find('input[type="date"]').length).toBe(2);
    });

    it('Должно открываться модальное окно при нажатии на кнопку
"show"', () => {
        const mockTask = { id: 1, title: 'Тестовая задача' };
        wrapper.setState({ taskHistories: [mockTask] });
        wrapper.find('.btn_show_planning').at(0).simulate('click');
        expect(wrapper.find('Modal').exists()).toBe(true);
    });

    it('Должен создаваться запрос через axios для удаления задачи при
нажатии кнопки удаления', () => {
        const mockTask = { id: 1, title: 'Тестовая задача' };
        wrapper.setState({ taskHistories: [mockTask] });
        wrapper.find('.btn_delete_planning').at(0).simulate('click');

        expect(axios.delete).toHaveBeenCalledWith('/api2/task/1');
    });

    it('Должен обновляться список задач при удалении задачи', async ()
=> {
        const mockTask = { id: 1, title: 'Тестовая задача' };
        const mockResponse = { data: true };
        axios.delete.mockResolvedValueOnce(mockResponse);

        wrapper.setState({ taskHistories: [mockTask] });
        await wrapper.instance().onDeleteTask(1);

        expect(wrapper.state('taskHistories')).toEqual([]);
    });
});
```

Листинг Ж.3 – Файл PlanDataUpdate.test.js

```
import axios from 'axios';
import planDataUpdate from '../components/planning/PlanDataUpdate';

jest.mock('axios');

global.localStorage = {
    setItem: jest.fn(),
    getItem: jest.fn(),
    removeItem: jest.fn(),
    clear: jest.fn()
};

describe('planDataUpdate', () => {
    afterEach(() => {
        jest.clearAllMocks();
    });

    it('Должен создаваться запрос через axios для получения данных о
планировании пользователя', async () => {
        const mockUserId = 1;
        const mockResponse = { data: [{ id: 1, title: 'Тестовый план'
}] };
        axios.get.mockResolvedValueOnce(mockResponse);

        const result = await planDataUpdate(mockUserId);

        expect(axios.get).toHaveBeenCalledWith(`api2/life-sphere/by-
user/${mockUserId}`);
        expect(result).toEqual(mockResponse.data);

expect(localStorage.setItem).toHaveBeenCalledWith('userPlanData',
JSON.stringify(mockResponse.data));
    });

    it('Должна выводиться ошибка при сбое запроса axios', async () =>
{
        const mockUserId = 1;
        const mockError = new Error('Ошибка сети');
        axios.get.mockRejectedValueOnce(mockError);

        await
expect(planDataUpdate(mockUserId)).rejects.toThrow('Ошибка сети');
    });
});
```

Листинг Ж.4 – Файл Modal.test.js

```
import React from 'react';
import { mount } from 'enzyme';
import Modal from './Modal';

describe('Modal', () => {
    let wrapper;
    const mockOnClose = jest.fn();

    beforeEach(() => {
```

```
        wrapper = mount(
            <Modal onClose={mockOnClose}>
                <div>Содержимое модального окна</div>
            </Modal>
        );
    });

    afterEach(() => {
        mockOnClose.mockReset();
    });

    it('Должно рендериться модальное окно', () => {
        expect(wrapper.exists()).toBe(true);
    });

    it('Должно содержимое модального окна', () => {
        expect(wrapper.text()).toContain('Содержимое модального
окна');
    });

    it('должен вызывать onClose при нажатии кнопки закрытия', () => {
        wrapper.find('.modal_button').simulate('click');
        expect(mockOnClose).toHaveBeenCalled();
    });
});
```

Приложение И

(рекомендуемое)

Результаты модульного тестирования

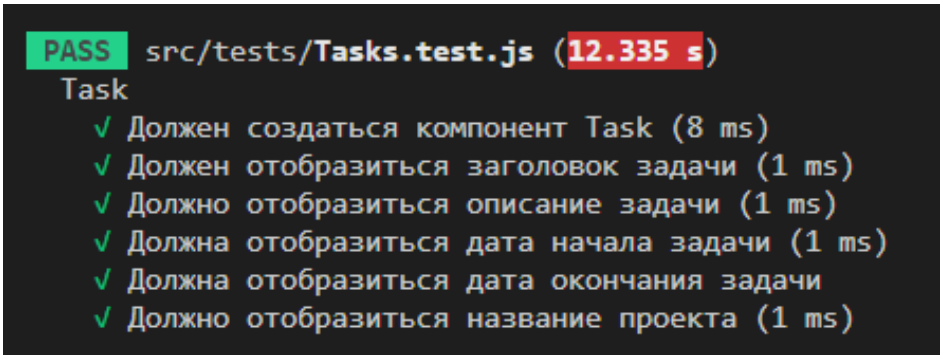

Рисунок И.1 – Модульное тестирование компонента Task

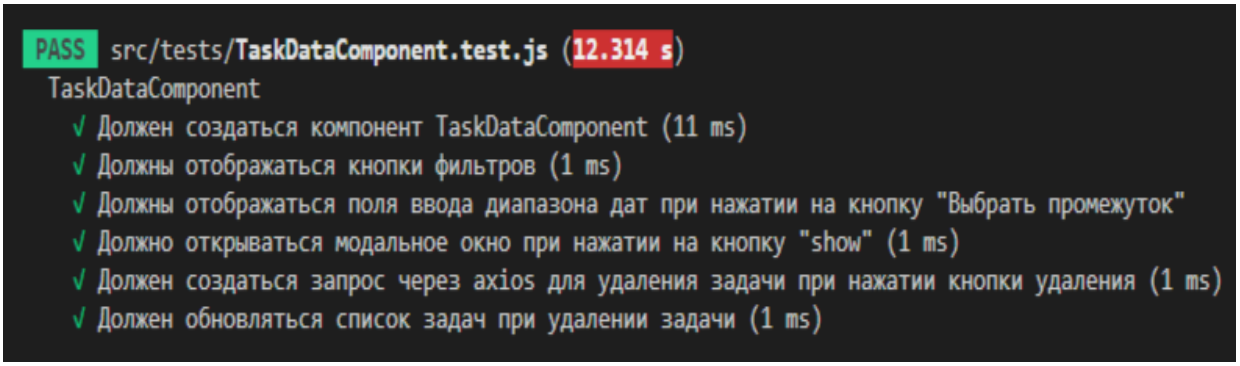

Рисунок И.2 – Модульное тестирование компонента TaskDataComponent

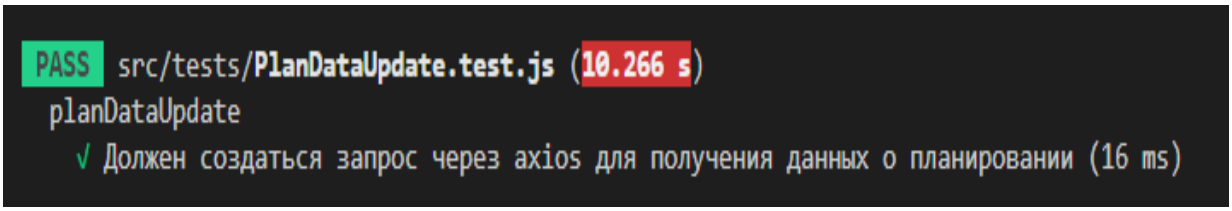

Рисунок И.3 – Модульное тестирование компонента PlanDataUpdate

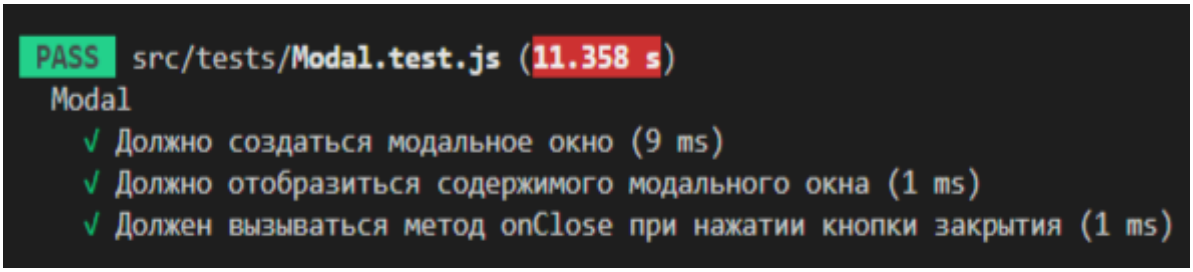

Рисунок И.4 – Модульное тестирование компонента Modal