**Министерство просвещения РК**

**ГККП «Алматинский экономический колледж»**



**ДИПЛОМНАЯ РАБОТА**

Специальность 1305000 – Информационные системы

Квалификация 1305023 – Разработчик программного обеспечения

**Тема**: Платформа для управления проектами и координации команд

Выполнил: Мирзахмет Султан.Ф,

Амувакиров Адиль.А

Группа: РПО 3-1

Руководитель: Абишев Н.Е

**Алматы, 2023**

**Министерство просвещения РК**

**ГККП «Алматинский экономический колледж»**

**ДОПУЩЕН К ЗАЩИТЕ**

Директор колледжа, д.т.н., профессор

 \_\_\_\_\_\_\_\_Ж.Н.Касымбек

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024г.

**ДИПЛОМНАЯ РАБОТА**

**Тема:** Платформа для управления проектами и координации команд

Специальность 1305000 - Информационные системы

Квалификация 1305023 - Разработчик программного обеспечения

Выполнили Мирзахмет.С.Ф,

Амувакиров.А.А

Руководитель Абишев.Н.Е

Нормаконтролер Курмангалиева А. Б.

**«Допущен к защите»**

Председатель ПЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись Ф.И.О., должность

**Министерство просвещения РК**

**ГККП «Алматинский экономический колледж»**

**ОТЗЫВ**

**руководителя о дипломной работе**

Фамилия, имя, отчество студента: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема дипломной работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка положительных и отрицательных качеств работы:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Общая оценка работы:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

С отзывом ознакомлен: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Министерство просвещения РК**

**ГККП «Алматинский экономический колледж»**

**Р Е Ц Е Н З И Я**

**на дипломную работу**

Дипломник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО)

Специальность: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Объем дипломной работы:

количество страниц\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

количество иллюстраций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Краткое описание дипломной работы и принятых решений

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Положительные стороны работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отрицательные стороны работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Предлагаемая оценка дипломной работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Место работы:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Занимаемая должность: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_»\_\_\_\_\_\_\_\_ 20\_\_г.

ОЗНАКОМЛЕНЫ:

|  |  |
| --- | --- |
| Директор АЭК,  д.т.н., профессор  \_\_\_\_\_\_\_\_\_Ж.Н.Касымбек  «\_\_\_»\_\_\_\_\_\_\_\_\_ 20\_\_г. | Председатель ПЦК «Базовой подготовки и компьютерных технологии»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_20\_\_г. |

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 9](#_Toc199865254)

[**1 ОСНОВНАЯ ЧАСТЬ** 11](#_Toc199865255)

[1.2 Аналоги платформы 16](#_Toc199865256)

[**2 СПЕЦИАЛЬНАЯ ЧАСТЬ** 22](#_Toc199865257)

[2.1 Постановка задачи. Выбор инструментов и их обоснование 22](#_Toc199865258)

[**2.2 Проектирование программного продукта** 30](#_Toc199865259)

[1. Анализ требований 30](#_Toc199865260)

[Архитектурное проектирование 32](#_Toc199865261)

[Проектирование базы данных 33](#_Toc199865262)

[Проектирование пользовательского интерфейса (UI/UX) 34](#_Toc199865263)

[Проектирование функциональных компонентов 36](#_Toc199865264)

[Проектирование безопасности 37](#_Toc199865265)

[Проектирование тестирования 38](#_Toc199865266)

[Проектирование масштабируемости и производительности 40](#_Toc199865267)

[Документация 41](#_Toc199865268)

[2.3 Разработка и тестирование приложения. Deploy приложения 42](#_Toc199865269)

[**1.** **ЭКОНОМИЧЕСКАЯ ЧАСТЬ** 45](#_Toc199865270)

[**1.1** **Технико** - **экономическое обоснование** 45](#_Toc199865271)

[**1.2** **Расчет затрат на разработку** 47](#_Toc199865272)

[**1.3** **Расчет затрат на заработную плату** 47](#_Toc199865273)

[**1.4** **Расчет затрат на социальные отчисления** 48](#_Toc199865274)

[**1.5** **Расчет затрат на электроэнергию** 49](#_Toc199865275)

[**1.6** **Расчет накладных расходов** 51](#_Toc199865276)

[**1.7** **Расчет на общие расходы** 52](#_Toc199865277)

[ЗАКЛЮЧЕНИЕ 52](#_Toc199865278)

# **ВВЕДЕНИЕ**

В условиях стремительного развития информационных технологий и повсеместной цифровизации различных сфер деятельности существенно увеличивается потребность в эффективных инструментах, предназначенных для управления задачами и отслеживания этапов их выполнения. Организации, проектные команды и отдельные специалисты сталкиваются с необходимостью точного контроля над ходом реализации поставленных задач на протяжении всего их жизненного цикла, что неизбежно требует применения автоматизированных систем, позволяющих повысить прозрачность процессов и продуктивность труда.

Целью данной дипломной работы является проектирование и разработка программного инструмента, обеспечивающего отслеживание поэтапного прогресса задач. Подобная программа позволит пользователям наглядно визуализировать этапы выполнения работ, анализировать текущее состояние и оперативно принимать решения для улучшения рабочих процессов.

В процессе исследования и разработки будут рассмотрены существующие методики управления задачами, изучены современные технологические возможности для создания программных продуктов, а также детально проанализированы потребности пользователей в эффективных средствах контроля выполнения задач. Итогом работы станет практическое программное решение, предлагающее набор инструментов для мониторинга задач и повышения их выполнения.

Актуальность темы объясняется высокими требованиями к качеству планирования и мониторинга задач в условиях необходимости выполнения множества задач одновременно и ограниченности доступных ресурсов. Предлагаемое программное решение предназначено как для командного взаимодействия, так и для индивидуального применения в различных сферах профессиональной деятельности.

Кроме того, внедрение автоматизированного инструмента для управления задачами позволит существенно снизить риски, связанные с человеческим фактором, такие как ошибки планирования, несвоевременная постановка задач и неточности в отслеживании их выполнения. Благодаря автоматизации процессов мониторинга задач сотрудники смогут сосредоточиться непосредственно на выполнении своих прямых обязанностей, что повысит их эффективность и уменьшит временные затраты на рутинные операции.

Разработанное решение также способствует улучшению коммуникации и координации между участниками проекта. Поскольку все этапы и статусы выполнения задач будут прозрачно отображены в системе, каждый сотрудник сможет видеть актуальное состояние задач и своевременно корректировать свои действия в зависимости от текущих условий и прогресса работы других участников.

Дополнительно, программное обеспечение будет включать функции аналитики и отчётности, что позволит руководителям и сотрудникам отслеживать тенденции в выполнении задач, выявлять узкие места и оперативно принимать необходимые меры по оптимизации рабочих процессов. В долгосрочной перспективе это обеспечивает повышение общей результативности организации и снижает издержки на реализацию проектов.

Современные методы и технологии, используемые в рамках данной работы, обеспечивают возможность масштабирования и адаптации инструмента под конкретные нужды различных компаний и команд. Возможность настройки и интеграции системы с уже существующими решениями (например, почтовыми клиентами, календарями и корпоративными сервисами) делает предложенный инструмент универсальным и удобным для широкой аудитории пользователей.

Особое внимание при разработке программного решения уделяется вопросам удобства использования и интуитивной понятности интерфейса. Продукт будет ориентирован на пользователей с разным уровнем технической подготовки, что минимизирует барьеры при внедрении системы в организационную структуру и ускорит адаптацию сотрудников к новым инструментам.

Таким образом, разработка предлагаемой системы является актуальной задачей не только с точки зрения повышения эффективности управления задачами, но и с позиции общего улучшения организационной культуры и рабочего климата внутри команды или компании.

# **1 ОСНОВНАЯ ЧАСТЬ**

* 1. **Анализ предметной области**

Разрабатываемая программа предназначена для оптимизации управления и повышения эффективности выполнения задач в таких сферах, как проектное управление, разработка программного обеспечения, образовательные и обучающие процессы и другие области деятельности. Основной задачей предлагаемого инструмента является обеспечение пользователям возможности визуально контролировать и анализировать ход выполнения задач в многозадачной или многокомпонентной среде.

1. **Общее описание предметной области**  
   В современных условиях задачи зачастую состоят из множества этапов и подзадач, выполнение которых требует четкой последовательности и эффективного контроля. В таких обстоятельствах важно не только планировать работу, но и постоянно отслеживать прогресс, выявлять возникающие проблемы на ранних стадиях, своевременно корректировать планы и распределение ресурсов.  
   Использование специализированных программных решений позволяет систематизировать задачи, четко определять ключевые этапы, назначать ответственных сотрудников и задавать временные рамки для выполнения каждого этапа. Это особенно актуально в условиях жестких временных ограничений и ограниченности ресурсов, когда успех задачи зависит от согласованности действий всех членов команды.

Дополнительно, современные подходы к управлению задачами требуют наличия механизмов обратной связи между участниками проекта и руководством, что позволяет оперативно реагировать на изменения условий выполнения работ и своевременно принимать корректирующие меры. Сложные проекты обычно включают множество взаимосвязанных задач, которые должны быть скоординированы и отслежены, что делает необходимым применение специализированных инструментов.

1. **Важность отслеживания прогресса задачи**  
   Эффективный мониторинг прогресса задач помогает не только строго соблюдать запланированные сроки выполнения, но и улучшает качество планирования, предотвращает риски, связанные с отклонениями от графика, а также повышает прозрачность всего процесса для всех участников проекта. Это особенно актуально в следующих сферах:  
   • Проектное управление, где задачи являются частью комплексных проектов с множеством участников и взаимосвязанных этапов.  
   • Разработка программного обеспечения, где важна последовательность и контроль каждой стадии процесса (анализ, проектирование, кодирование, тестирование и внедрение).  
   • Образовательные и учебные процессы, где задачи требуют регулярной оценки и проверки выполнения промежуточных шагов.

Отслеживание прогресса задач также имеет критическое значение для своевременного выявления проблемных зон, предотвращения перегрузки сотрудников и перераспределения ресурсов с целью достижения максимально эффективного результата. Регулярный мониторинг позволяет повысить мотивацию участников проекта за счет наглядного представления их вклада и достижения промежуточных результатов.

1. **Ключевые компоненты системы**  
   Ключевыми компонентами разрабатываемого инструмента являются:  
   • Задачи и подзадачи: Формирование и структурирование задач с выделением ключевых этапов и определением их зависимостей.  
   • Мониторинг прогресса: Визуальное представление текущего состояния задач, позволяющее оперативно обновлять данные и процент выполнения.  
   • Система уведомлений: Автоматизированные напоминания и уведомления, предотвращающие пропуск важных сроков и этапов.  
   • Графические интерфейсы: Использование визуализации в виде диаграмм, графиков и других инструментов, обеспечивающих наглядность мониторинга.  
   • Многоуровневое планирование: Поддержка планирования на разных уровнях детализации, обеспечивающая мониторинг глобальных и отдельных аспектов выполнения задач.

Дополнительно, система должна включать модуль аналитики и отчетности, который позволит выявлять тенденции и оценивать эффективность выполнения задач за определенные периоды. Также важен компонент управления правами доступа, который обеспечит надежную защиту данных и разграничение прав пользователей.

1. **Проблемы и вызовы**  
   Несмотря на наличие различных систем мониторинга, существует ряд вызовов, требующих внимания при разработке и использовании подобных решений:  
   • Масштабируемость системы: необходимость эффективного управления большим количеством задач и их подэтапами с использованием производительных алгоритмов и структур данных.  
   • Интеграция с внешними системами: обеспечение совместимости разрабатываемого решения с существующими инструментами управления проектами, коммуникациями и другими сервисами.  
   • Точность и актуальность данных: оперативное и точное обновление информации о текущем состоянии задач с отражением реального положения дел.  
   • Удобство интерфейса и его адаптивность: создание интуитивно понятного и легко настраиваемого пользовательского интерфейса.

Дополнительно, необходимо учитывать вопросы безопасности и конфиденциальности данных, особенно при работе с чувствительной информацией. Также важно предусмотреть высокую производительность и устойчивость системы к нагрузкам, чтобы обеспечить бесперебойную работу в условиях интенсивного использования.

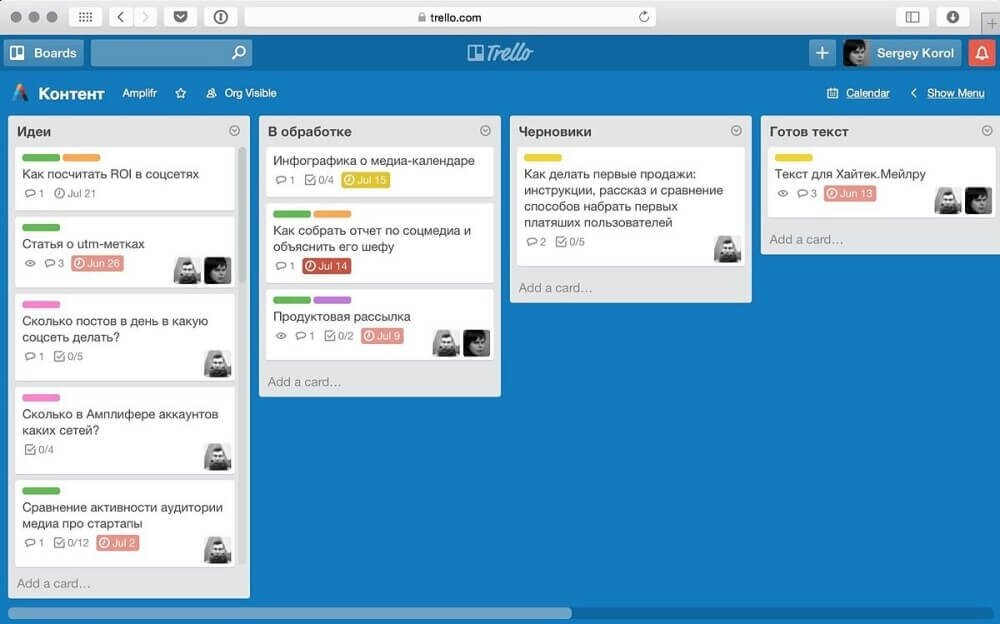
1. **Текущие решения и их недостатки**  
   В настоящее время на рынке представлены различные системы, такие как Trello, Asana, Jira, однако они не всегда удовлетворяют потребности пользователей в полной мере из-за ограничений в гибкости настройки, сложности интерфейса, а также недостаточных возможностей детализированной визуализации этапов и текущего состояния выполнения задач.

Кроме того, существующие решения могут иметь проблемы с интеграцией в текущую инфраструктуру компаний, ограниченные возможности кастомизации под специфические требования и недостаточную поддержку мультиплатформенности и мобильных устройств.

1. **Потребности рынка**  
   При разработке эффективного инструмента мониторинга задач необходимо учитывать такие ключевые требования пользователей, как:  
   • Простой и персонализируемый интерфейс.  
   • Доступность использования для сотрудников с разным уровнем технических навыков.  
   • Возможность интеграции с популярными инструментами и сервисами (календарь, почта и др.).  
   • Поддержка многозадачности и гибкое планирование.  
   • Возможности аналитики и формирования отчетов для выявления проблемных мест и повышения общей эффективности выполнения задач.

Также востребованы функции совместного доступа и коллективной работы, возможность онлайн-коммуникации и быстрый обмен документами между участниками проектов.

### 1.2 Аналоги платформы



**Рис 1.2.1 - Trello**

**1. Trello**

Описание: Trello — это популярный инструмент для управления проектами, основанный на принципе канбан-доски. В Trello можно создавать задачи (карточки), организовывать их в списки и перемещать между различными этапами выполнения.

Основные функции:

Возможность создания карточек задач с подзадачами.

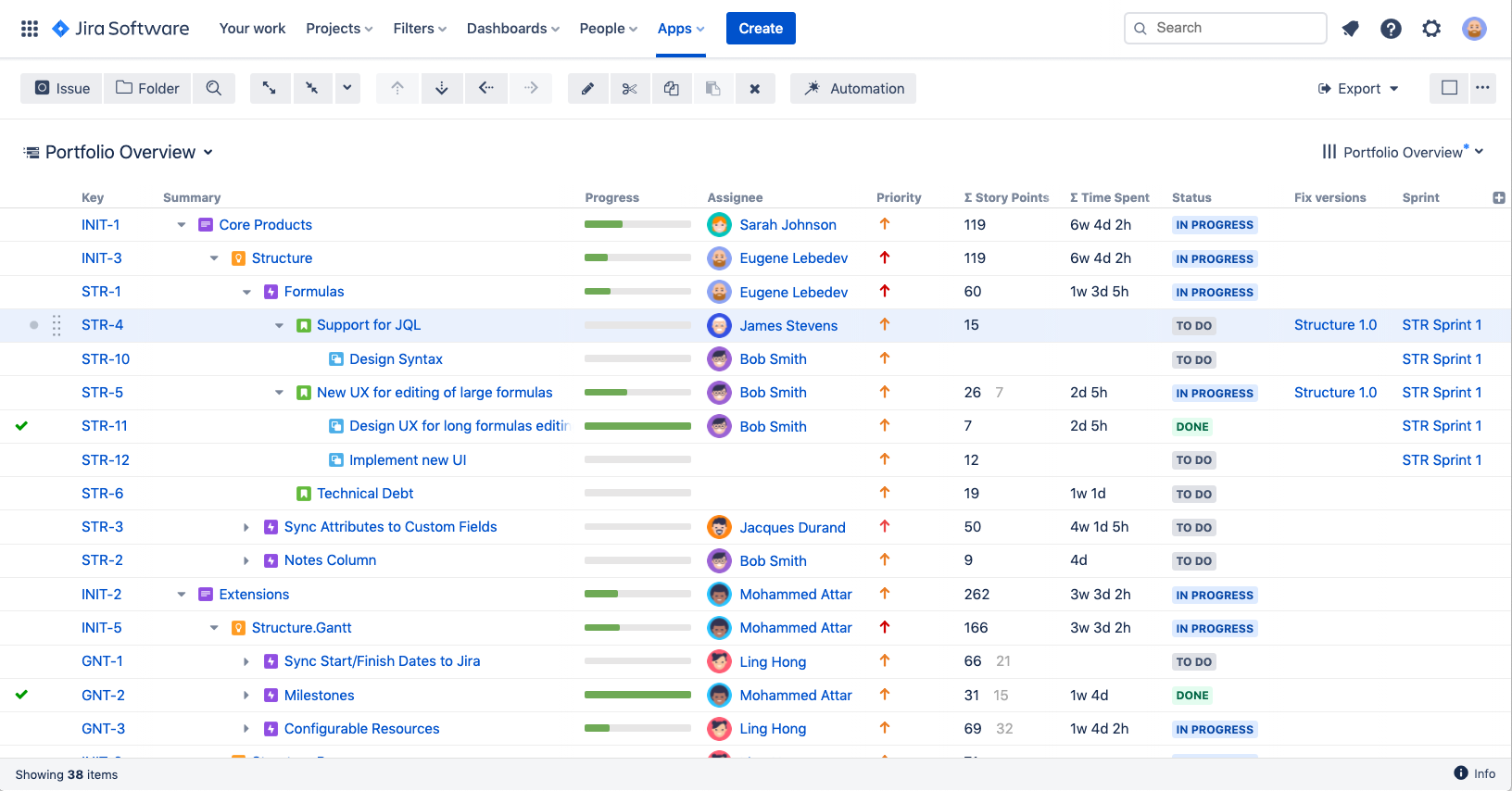
Установка сроков, напоминаний и комментариев.

Визуализация прогресса через доски, списки и карточки.

Интеграция с различными сервисами (Slack, Google Drive, и др.).

Совместная работа над проектами.

Недостатки: Ограниченная детализация задач, нет глубоких аналитических возможностей, а также могут возникнуть трудности в управлении большими проектами с множеством участников.



**Рис 1.2.3-Jira**

**2.Jira**

Описание: Jira — это система управления проектами, ориентированная на командную работу, особенно в разработке программного обеспечения. Она широко используется для отслеживания задач и багов, а также для планирования работы в команде.

Основные функции:

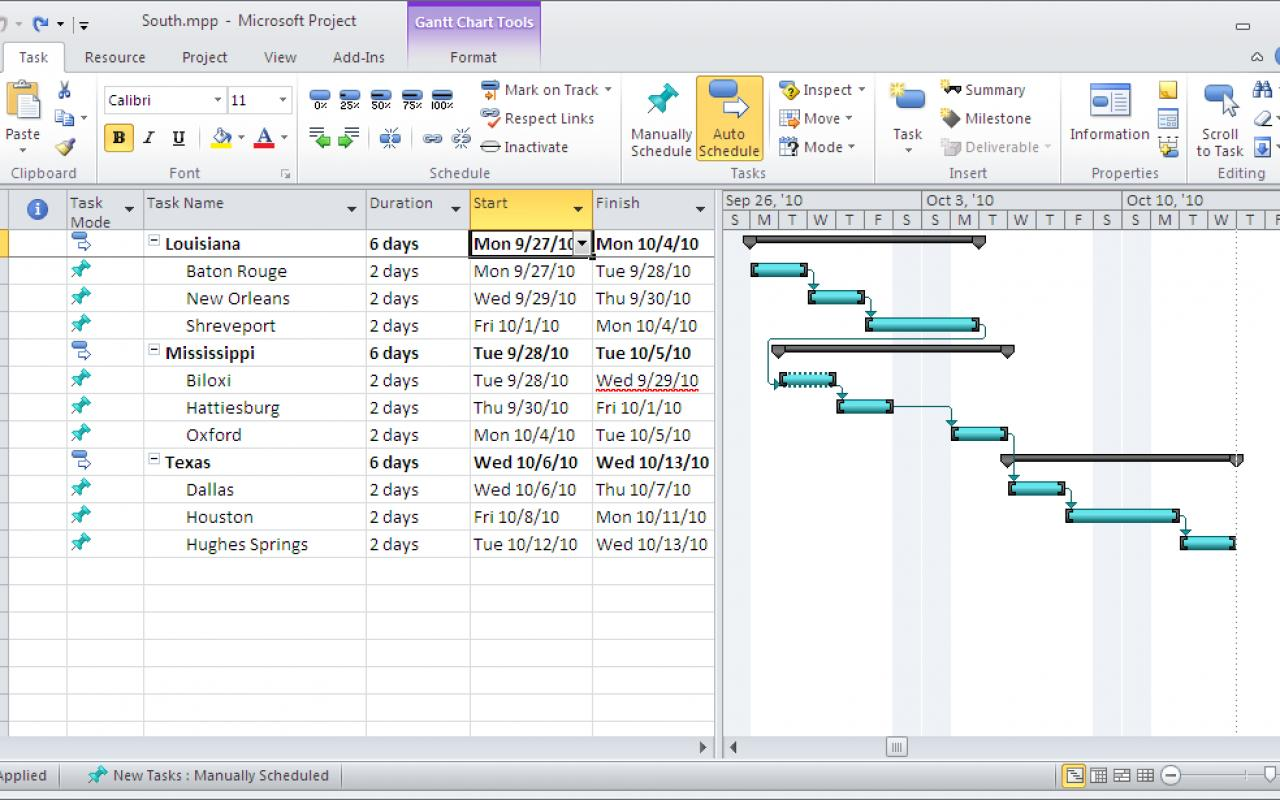
Поддержка работы с задачами, эпиками и историями.

Возможность визуализации и отслеживания прогресса через диаграммы и отчеты.

Гибкость в настройке рабочих процессов и автоматизаций.

Интеграция с другими инструментами разработки (например, Bitbucket).

Недостатки: Для использования Jira требуется опыт в настройке и управлении системой, а также высокая стоимость для небольших команд.



**Рис 1.2.5 - Microsoft Project**

**3.Microsoft Project**

Описание: Microsoft Project — это мощное средство для планирования и отслеживания проектов, которое широко используется в корпоративной среде для управления сложными проектами с множеством этапов.

Основные функции:

Поддержка диаграмм Ганта для отслеживания прогресса.

Многоуровневая детализация задач и подзадач.

Установка зависимостей между задачами.

Расширенная отчетность и прогнозирование.

Недостатки: Это платное решение, которое может быть сложным для новичков. Требует серьезных навыков для использования.

**1.3 Современные методы разработки web-приложения**

Для создания современного и интерактивного веб-приложения используются различные языки программирования и технологии. Наиболее популярными на сегодняшний день являются: Python, JavaScript, HTML и CSS. Каждый из этих языков решает определённый набор задач и применяется на своём уровне веб-разработки.

HTML (Hyper Text Markup Language) — это язык разметки гипертекста, который используется для создания структуры веб-страниц. С его помощью можно определять текст, изображения, таблицы, формы и другие элементы интерфейса. HTML-страницы объединяются в сайт под общим доменным именем. Однако такие страницы сами по себе статичны и не взаимодействуют с пользователем динамически.

CSS (Cascading Style Sheets) — каскадные таблицы стилей, определяющие внешний вид элементов HTML. Они позволяют создавать адаптивный, кроссбраузерный дизайн, задавая оформление заголовков, таблиц, форм и других компонентов страницы. CSS используется для построения как простых, так и сложных пользовательских интерфейсов. Используя современные методики CSS, такие как Flexbox и Grid, разработчики могут эффективно создавать гибкие и отзывчивые дизайны.

JavaScript — язык программирования, применяемый для реализации клиентской логики на стороне пользователя. Он позволяет изменять содержимое страниц без перезагрузки, обрабатывать действия пользователя, валидировать формы и создавать динамические элементы, такие как выпадающие меню и всплывающие окна. В рамках проекта используется «чистый» JavaScript (Vanilla JS), без тяжёлых библиотек, что обеспечивает высокую производительность и простоту сопровождения. Также широко используются современные JavaScript-фреймворки и библиотеки, такие как React, Angular и Vue.js, которые предлагают разработчикам удобные инструменты для создания сложных и масштабируемых пользовательских интерфейсов.

Python — это мощный высокоуровневый язык программирования, поддерживающий различные парадигмы (объектно-ориентированную, функциональную, процедурную). Он активно применяется на стороне сервера для обработки запросов, работы с базой данных, управления пользователями и бизнес-логикой. Благодаря своей читаемости и большому количеству библиотек Python широко применяется в веб-разработке.

Фреймворк — это каркас для разработки приложений, содержащий базовые модули и шаблоны архитектуры. Он позволяет ускорить процесс разработки, повысить надёжность и удобство сопровождения кода. В данном проекте применялись следующие фреймворки и технологии:

• Django — популярный Python-фреймворк, предоставляющий готовые решения для маршрутизации, работы с базами данных, аутентификации, административной панели и защиты от уязвимостей. Он позволяет сконцентрироваться на логике приложения, минимизируя количество шаблонного кода.

• Django REST Framework — надстройка над Django, обеспечивающая удобное построение REST API. Это необходимо для взаимодействия между клиентской частью (фронтендом) и серверной частью (бэкендом), а также интеграции с мобильными приложениями.

• Django Channels — расширение Django для поддержки протокола WebSocket. Оно позволяет реализовать обмен сообщениями в реальном времени — например, чат между консультантом и клиентом. Для промежуточного хранения и передачи сообщений используется Redis.

• PostgreSQL — современная реляционная система управления базами данных, используемая для хранения информации о пользователях, заявках и консультациях. Она обеспечивает надёжность, масштабируемость и поддержку транзакций.

Кроме того, важную роль играет использование инструментов для контроля версий (например, Git), что позволяет эффективно управлять изменениями в коде и совместно работать над проектом нескольким разработчикам одновременно.

Для автоматизации тестирования и обеспечения качества кода применяются различные средства и методики, такие как юнит-тесты, интеграционное тестирование и Continuous Integration (CI), позволяющие оперативно выявлять и устранять ошибки.

Использование технологий контейнеризации, таких как Docker, значительно упрощает развертывание приложения на различных средах и обеспечивает высокую стабильность и масштабируемость системы.

Таким образом, выбранный стек технологий (Python + Django + JavaScript + PostgreSQL) обеспечивает высокую скорость разработки, надёжность, безопасность и простоту поддержки веб-приложения, соответствующего задачам по подбору и проведению консультаций.

# **2 СПЕЦИАЛЬНАЯ ЧАСТЬ**

### 2.1 Постановка задачи. Выбор инструментов и их обоснование

Необходимо разработать программный продукт для управления проектами и координации команд.

Для разработки приложения будут использованы следующие инструменты и технологии:  
• HTML, CSS, JavaScript (Front-end);  
• Python 3;  
• Django Framework (Back-end);  
• PostgreSQL database;  
• Visual Studio Code;  
• Git (система контроля версий);  
• Docker (контейнеризация);  
• Redis (для организации кэша и брокера сообщений);  
• Django REST Framework (для реализации API);  
• Django Channels (для поддержки WebSocket);  
• Nginx (для развертывания и обеспечения веб-сервера);  
• Gunicorn (для запуска и масштабирования Django-приложения).

HTML, CSS и JavaScript – это основные технологии, применяемые для создания клиентской части веб-приложений.

HTML (HyperText Markup Language) используется для построения структуры веб-страниц, что позволяет размещать тексты, изображения, видео, формы, таблицы и другие элементы, необходимые для работы пользователей с информацией и взаимодействия с системой.

CSS (Cascading Style Sheets) обеспечивает визуальное оформление элементов страницы, включая шрифты, цвета, размеры, отступы, а также позиционирование элементов. CSS позволяет легко создавать адаптивный дизайн, который будет корректно отображаться на всех типах устройств: от мобильных телефонов до настольных компьютеров.

JavaScript применяется для добавления интерактивности страницам. С помощью JavaScript реализуются функции обработки действий пользователей, динамическое обновление контента без перезагрузки страницы (например, через Ajax и Fetch API), анимации, валидация форм и создание сложных динамических интерфейсов.

Эти технологии позволяют создавать веб-приложения с:  
• гибкой структурой и адаптивностью интерфейса;  
• стилизованным и пользовательским дизайном;  
• возможностью асинхронного обмена данными с сервером;  
• интуитивно понятными и отзывчивыми интерфейсами, которые обеспечивают комфортное использование системы.

Django — мощный и широко используемый веб-фреймворк на языке Python, который позволяет быстро и эффективно создавать сложные и масштабируемые веб-приложения. Django следует архитектурному паттерну MTV (Модель–шаблон–представление), что облегчает разделение функциональности и улучшает управляемость кодовой базы.

Архитектура Django включает:

• Модели (ORM) — отвечают за доступ, обработку, хранение и валидацию данных. Django ORM предоставляет удобный интерфейс взаимодействия с базой данных без необходимости написания SQL-кода вручную, что значительно упрощает разработку и поддержку приложения.

• Шаблоны (Templates) — описывают, как данные будут отображаться пользователю. Шаблонизатор Django позволяет удобно вставлять данные и выполнять минимальную логику непосредственно на страницах, обеспечивая безопасность и удобство визуализации данных.

• Представления (Views) — обрабатывают входящие запросы, получают данные от моделей и передают их в шаблоны. Представления также реализуют всю логику обработки HTTP-запросов и бизнес-процессы приложения.

Дополнительные компоненты, используемые в Django:

• URL-маршрутизаторы (URL Dispatcher) — обеспечивают перенаправление запросов от браузера в соответствующие представления, реализуя понятную и организованную маршрутизацию.

• Middleware (промежуточное программное обеспечение) — управляет запросами и ответами на различных этапах их обработки, обеспечивая аутентификацию, логирование, обработку ошибок и другие функции.

• Django REST Framework — позволяет легко и быстро разрабатывать REST API для взаимодействия с внешними клиентами и фронтенд-приложениями, а также поддерживает авторизацию и сериализацию данных.

• Django Channels — предоставляет возможность интеграции WebSocket, что позволяет реализовать обмен данными в режиме реального времени (например, для чатов, уведомлений и систем онлайн-обновлений статуса задач).

• PostgreSQL — надежная, мощная и гибкая реляционная система управления базами данных, обеспечивающая высокую производительность, поддержку транзакций, полнотекстовый поиск, обработку сложных запросов и масштабируемость.

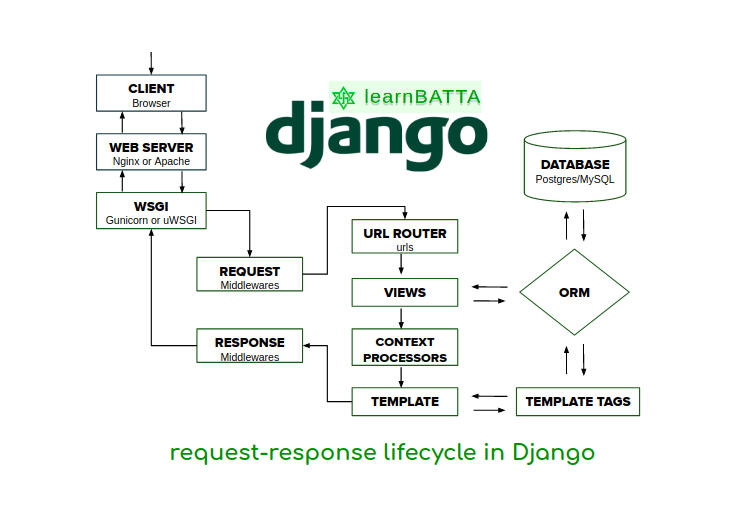
• Redis — используется для организации эффективного кэширования данных и обеспечения передачи сообщений в режиме реального времени, значительно повышая производительность системы.

• Docker — технология контейнеризации, которая облегчает развертывание приложения в разных средах, обеспечивает легкость масштабирования, переносимость и стабильность работы приложения.

• Git — система контроля версий, обеспечивающая совместную работу команды разработчиков над проектом, хранение истории изменений и возможность быстрого возврата к предыдущим версиям.

• Nginx и Gunicorn — используются для эффективного развертывания и запуска веб-приложения в продакшн-средах, обеспечивая стабильность и быструю обработку запросов.

Использование этих технологий и инструментов позволяет создать современный, надежный, безопасный и легко сопровождаемый программный продукт, отвечающий всем потребностям управления проектами и координации команд.



**Рис 2.1.1 – Принцип работы фреймворка Django**

Фреймворк Django включает в себя множество специализированных компонентов и инструментов, обеспечивающих быструю и эффективную разработку полноценных веб-приложений. Среди встроенных возможностей можно выделить:

• Встроенный веб-сервер, предназначенный исключительно для целей разработки и отладки. Он позволяет запускать проект без необходимости устанавливать сторонние серверы (например, Apache или Nginx), что значительно упрощает тестирование.

• Система сигналов (Signals) — механизм обратного вызова, который позволяет определённым компонентам приложения реагировать на конкретные события. С помощью сигналов можно, например, автоматически отправлять уведомление при регистрации нового пользователя или обновлении данных в базе.

• Модуль интернационализации (i18n) — инструмент, позволяющий адаптировать веб-приложение под разные языки и локали. Это особенно важно для создания проектов с широкой аудиторией, охватывающей пользователей из разных стран.

• Встроенная система аутентификации и авторизации — мощный инструмент управления доступом, который позволяет проверять личность пользователей, назначать им права и группы, защищать страницы от несанкционированного доступа и подключать сторонние решения (например, OAuth, двухфакторную авторизацию и др.).

• Middleware (промежуточные слои обработки запросов) — цепочка фильтров, которые обрабатывают каждый запрос и ответ. С их помощью можно реализовать кэширование, сжатие данных, переадресацию, ограничение доступа по IP и другие операции, происходящие «за кулисами».

• Административная панель (Django Admin) — одно из главных преимуществ Django. Это автоматически генерируемый интерфейс для управления содержимым сайта: добавлением, редактированием и удалением объектов. Его можно настроить и кастомизировать под нужды проекта.

• Интегрированная система тестирования — Django включает средства для написания юнит-тестов на Python, которые позволяют проверять корректность работы кода, обнаруживать и устранять ошибки ещё до публикации проекта.

• Механизм комментариев — возможность подключения системы комментариев, которая может быть встроена как часть модели, связанной с любым объектом (статьей, товаром, публикацией и т.д.).

• Система защиты — Django изначально разработан с учетом безопасности. Он обеспечивает защиту от наиболее распространённых атак, включая:  
o XSS (внедрение вредоносного кода в веб-страницу),  
o CSRF (межсайтовая подделка запросов),  
o SQL-инъекции,  
o перебор паролей и другие угрозы.

Дополнительные возможности Django:

• ORM (Object-Relational Mapping) — мощный механизм, упрощающий взаимодействие с базой данных, позволяющий избегать написания прямых SQL-запросов и работать с данными в объектно-ориентированном стиле.

• Встроенная система миграций — автоматизированный механизм изменения структуры базы данных, позволяющий легко вносить изменения в схемы данных без потери существующих данных.

• Кэширование данных — поддержка различных механизмов кэширования, таких как Memcached и Redis, которые значительно ускоряют работу приложения за счёт минимизации обращений к базе данных.

• Поддержка REST API — с помощью Django REST Framework можно легко создать полноценное REST API, которое обеспечивает простое взаимодействие с внешними приложениями и сервисами.

• Поддержка WebSocket — использование Django Channels для реализации асинхронного взаимодействия между сервером и клиентами в режиме реального времени.

Работа с базой данных SQLite

По умолчанию Django использует встроенную базу данных SQLite — лёгкую СУБД на языке C, не требующую отдельного серверного окружения. Она идеально подходит для разработки, тестирования и небольших проектов. SQLite хранит все данные в одном файле, обеспечивая высокую производительность и простоту резервного копирования.

База данных — это логически структурированное хранилище информации. СУБД, такие как SQLite, позволяют добавлять, изменять, удалять и читать данные, а также управлять доступом к ним.

SQLite удобна для следующих сценариев:

• локальные и мобильные приложения;  
• MVP и прототипы;  
• личные и учебные проекты;  
• небольшие веб-сайты с ограниченной нагрузкой.

Её преимущества:

• Простота и минимальная настройка — база данных создаётся автоматически;  
• Кроссплатформенность — работает на всех ОС;  
• Высокая производительность при работе с файлами и чтении данных;  
• Минимальные требования к ресурсам — не требует сервера и администрирования;  
• Открытый исходный код — можно использовать и модифицировать без ограничений.

SQLite взаимодействует с приложением напрямую — без отдельного клиента или сервера. Это делает её удобной для встраиваемых решений, браузеров (Chrome, Firefox), мобильных приложений, мессенджеров и даже устройств IoT (например, дронов, автомобилей и т.д.).

Дополнительные особенности SQLite:

• Поддержка транзакций — обеспечивает целостность и надёжность данных;  
• Простота резервного копирования — достаточно сохранить один файл;  
• Минимальные задержки при доступе к данным — отлично подходит для приложений с высокой частотой запросов;  
• Широкое использование — поддерживается огромным количеством платформ и языков программирования;  
• Возможность интеграции с другими базами данных — данные можно легко экспортировать и импортировать в другие СУБД.

Таким образом, использование Django и SQLite обеспечивает простоту и эффективность разработки веб-приложений, позволяя быстро создавать и поддерживать проекты любой сложности.

**2.2 Проектирование программного продукта**

Проектирование веб-приложения для координации команды включает определение архитектуры, интерфейсов и взаимодействия компонентов. Основная цель — создать удобную систему для постановки задач, деления их на этапы и отслеживания прогресса.

### 1. Анализ требований

• Функциональные требования:  
o создание задач;  
o деление задач на этапы;  
o отображение статуса выполнения этапов;  
o возможность добавления комментариев и заметок;  
o уведомления пользователей о важных событиях;  
o фильтрация и поиск задач по различным критериям;  
o регистрация и авторизация пользователей;  
o управление профилями пользователей и их ролями.

• Нефункциональные требования:  
o удобный и интуитивно понятный интерфейс (HTML/CSS/JS);  
o безопасность данных и конфиденциальность (Django-средства защиты);  
o масштабируемость (возможность добавления новых функций без значительных изменений структуры);  
o высокая производительность и быстрая реакция системы;  
o адаптивность интерфейса под различные устройства (десктоп, планшет, мобильный телефон);  
o резервное копирование и восстановление данных;  
o мониторинг и логирование активности пользователей.

Сбор и документирование требований

Требования собирались на основе анализа аналогов, опросов потенциальных пользователей и изучения специфики командной работы. Был составлен подробный документ, фиксирующий структуру задач, этапов, механизмы отслеживания прогресса, а также требования к безопасности, производительности и масштабируемости.

Приоритизация требований

Для эффективного проектирования была проведена приоритизация требований. В первую очередь реализуются ключевые функции: создание задач, деление на этапы и отслеживание прогресса. Второстепенные требования, такие как улучшение интерфейса, расширенные функции поиска и фильтрации задач, а также дополнительные роли пользователей, запланированы на последующие этапы разработки.



### Архитектурное проектирование

Архитектурное проектирование определяет структуру системы для управления координацией команды.

• Архитектурный стиль: используется клиент-серверная модель. Django обрабатывает серверную часть, фронтенд реализован с помощью HTML, CSS и JavaScript.

• Компоненты системы:  
o Фронтенд: отображение задач, этапов и прогресса, комментариев, уведомлений;  
o Бэкенд (Django): обработка логики создания задач, хранения этапов и статуса, обработки пользовательских запросов;  
o База данных: хранение информации о пользователях, задачах, этапах и комментариях (SQLite на этапе разработки);  
o API: взаимодействие между фронтендом и бэкендом через HTTP-запросы и RESTful API.

• Взаимодействие компонентов: фронтенд отправляет запросы к бэкенду через стандартные Django views или REST API. Обновление интерфейса реализовано с использованием JavaScript (AJAX-запросы).

• Выбор технологий:  
o Язык программирования: Python (Django);  
o Шаблоны: HTML с Django Template Language;  
o Стилизация: CSS;  
o Клиентская логика: JavaScript;  
o База данных: SQLite.

• Дополнительные технологии:  
o Использование Docker для удобного развертывания и масштабирования;  
o Интеграция Redis для реализации кэширования и обработки сообщений в реальном времени;  
o Поддержка WebSocket для реализации уведомлений и чатов.



### Проектирование базы данных

Для системы координации команды была выбрана реляционная база данных, так как данные имеют чёткую структуру и связи.

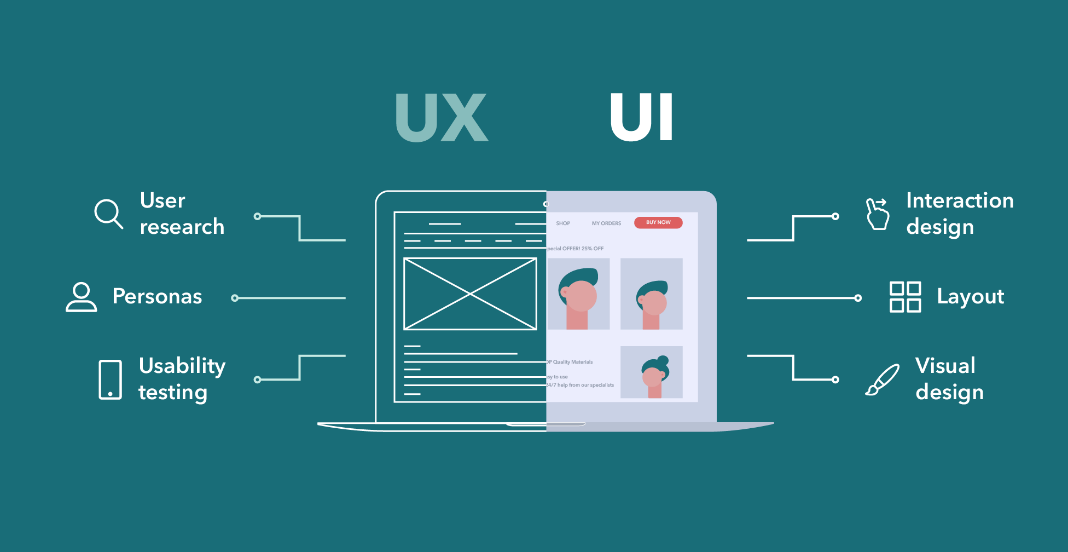
• Тип базы данных: используется реляционная СУБД SQLite (на этапе разработки), встроенная в Django.

• Схема базы данных включает основные таблицы:  
o Пользователь — информация о зарегистрированных участниках;  
o Задача — описание задач, автор, дата создания;  
o Этап — привязка к задаче, описание, статус выполнения;  
o Комментарии — привязанные к задачам и этапам комментарии пользователей;  
o Уведомления — уведомления пользователей о важных событиях.

• Связи:  
o Один пользователь может создавать много задач;  
o Каждая задача может включать несколько этапов;  
o Каждый этап и задача могут содержать множество комментариев.

• Производительность: структура базы оптимизирована для небольших команд. При необходимости возможен переход на PostgreSQL или другие производительные решения.

• Безопасность данных: реализована защита от SQL-инъекций средствами Django ORM, используется встроенная авторизация и проверка прав доступа, шифрование конфиденциальных данных.



### Проектирование пользовательского интерфейса (UI/UX)

Интерфейс проекта направлен на простое и наглядное управление задачами и этапами.

• Прототипы: разработаны макеты основных страниц — список задач, детальная страница задачи с этапами, комментариями и индикатором прогресса, профиль пользователя и панель администратора.

• UX-решения: интерфейс ориентирован на минимальные действия пользователя — быстрое добавление задач, редактирование этапов, переключение статусов, добавление комментариев без перезагрузки страницы.

• UI-дизайн: используется светлая цветовая палитра, читаемые шрифты и простые иконки для обозначения состояния задач, адаптивность под мобильные устройства, использование интерактивных элементов и анимаций для повышения удобства использования.

• Дополнительные UX/UI возможности:  
o возможность персонализации интерфейса под предпочтения пользователя;  
o использование интерактивных подсказок и всплывающих уведомлений для улучшения опыта взаимодействия;  
o интуитивная навигация и минималистичный дизайн для быстрого освоения системы новыми пользователями.

* **Адаптивность**: интерфейс реализован с использованием адаптивной верстки (CSS), что обеспечивает корректное отображение на ПК и мобильных устройствах.

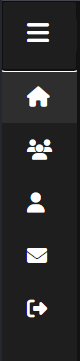
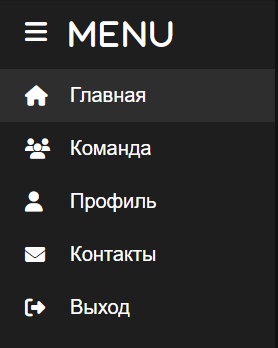
 

Рис 2.2.4 - Проектирование пользовательского интерфейса (UI/UX)

### Проектирование функциональных компонентов

Функциональные компоненты определяют, как реализованы основные действия системы координации команды и каким образом пользователь взаимодействует с системой.

• Диаграмма потока данных (DFD): данные от пользователя (создание задачи, добавление этапов, изменение статуса) проходят через веб-интерфейс, обрабатываются на серверной стороне фреймворком Django, сохраняются в базе данных (SQLite/PostgreSQL) и возвращаются обратно во фронтенд, где отображаются с использованием HTML/CSS и JavaScript. Приложение поддерживает двухстороннюю связь между интерфейсом и сервером, что обеспечивает динамическое обновление информации.

• Логика приложения: на серверной стороне реализованы представления (views), которые обрабатывают входящие запросы. Каждое представление отвечает за отдельную функциональность, такую как создание задач, добавление этапов, обновление прогресса, удаление элементов, проверка прав пользователя. JavaScript на стороне клиента позволяет реализовать динамические элементы интерфейса — например, интерактивное переключение статуса этапа, отображение уведомлений, автозаполнение форм.

• Интеграция REST API: хотя на текущем этапе интеграции с внешними сервисами не предусмотрены, архитектура проекта построена таким образом, чтобы в будущем легко подключить REST API. Это позволит интегрировать приложение с корпоративными системами, мессенджерами, календарями, облачными сервисами и мобильными приложениями.

• Реализация обратной связи: система предусматривает отправку уведомлений пользователям о важных событиях (например, завершение этапа, назначение задачи), что повышает уровень координации внутри команды.

### Проектирование безопасности

Для защиты пользовательских данных и обеспечения стабильной и безопасной работы системы реализованы следующие меры безопасности:

• Аутентификация и авторизация: используется встроенная система аутентификации Django, включающая регистрацию, вход, смену и восстановление пароля. Каждому пользователю присваиваются права доступа, позволяющие разграничить действия между участниками проекта (например, администратор, исполнитель, наблюдатель). Только автор задачи может её редактировать или удалять.

• Шифрование данных: передача данных через сеть осуществляется по защищённому протоколу HTTPS (при развертывании на сервере), пароли пользователей сохраняются в базе в зашифрованном виде с использованием безопасных алгоритмов хэширования (PBKDF2, bcrypt).

• Защита от атак: o SQL-инъекции предотвращаются использованием Django ORM, которое автоматически экранирует пользовательские данные; o CSRF (межсайтовая подделка запросов) блокируется встроенным механизмом Django — каждый запрос содержит уникальный токен; o XSS (межсайтовый скриптинг) исключается автоматическим экранированием пользовательского ввода в шаблонах; o Защита от перебора паролей реализуется с помощью ограничения количества попыток входа (можно подключить сторонние библиотеки, например, django-axes).

• Дополнительные меры: o Защита административной панели с помощью двухфакторной аутентификации; o Ограничение доступа к API на основе IP-адресов или токенов; o Регулярное обновление зависимостей и проверка уязвимостей в сторонних библиотеках.

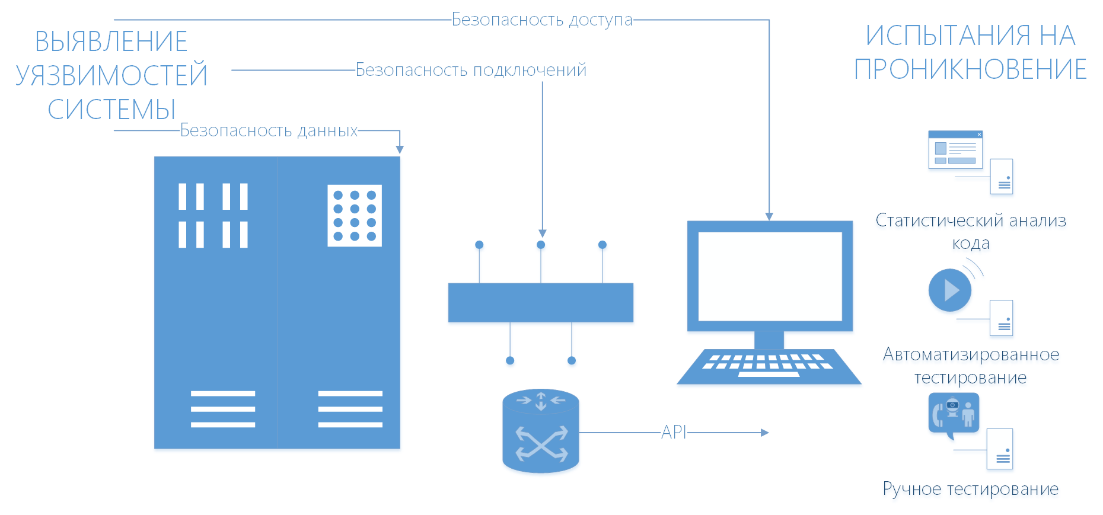


Рис 2.2.6 - Проектирование безопасности

### Проектирование тестирования

Тестирование системы играет ключевую роль в обеспечении её надёжности, устойчивости и функциональной полноты.

• Типы тестов: o Юнит-тесты для моделей (проверка создания, обновления и удаления объектов) и представлений (обработка запросов); o Интеграционные тесты для оценки взаимодействия компонентов: от представлений до базы данных; o Функциональные тесты для проверки пользовательских сценариев; o UI-тесты с использованием Selenium, обеспечивающие эмуляцию действий пользователя в браузере.

• Стратегия тестирования: o Покрытие тестами основных операций: регистрация, авторизация, создание задач, добавление и редактирование этапов, проверка прав доступа; o Обработка некорректного ввода и несанкционированных попыток доступа к защищённым ресурсам; o Проверка отклика системы под нагрузкой (при необходимости — стресс-тесты с использованием Apache JMeter).

• Инструменты тестирования: o Django встроенный модуль unittest; o Pytest для более гибкого написания тестов; o Selenium и Playwright для автоматизированного тестирования интерфейса; o GitHub Actions или GitLab CI/CD для автоматического запуска тестов при коммите и развертывании.



Рис 2.2.7 – Проектирование тестирования

### Проектирование масштабируемости и производительности

Система создаётся с учетом возможного увеличения числа пользователей, расширения функциональности и роста объёмов данных.

• Масштабирование: o Горизонтальное масштабирование путём разделения логики на отдельные микросервисы (например, чат, уведомления, задачи); o Вертикальное масштабирование — переход на более производительное серверное окружение; o Переход с SQLite на PostgreSQL или другую промышленную СУБД с поддержкой кластеризации.

• Производительность: o Оптимизация SQL-запросов через индексы, выборочные выборки, пагинацию данных; o Использование кэширования на уровне шаблонов, представлений и данных (Redis); o Минимизация количества запросов к серверу за счёт асинхронного взаимодействия через AJAX и WebSocket.

• Мониторинг и логирование: o Логирование всех действий пользователя (аутентификация, работа с задачами); o Использование Sentry для отслеживания ошибок и исключений в реальном времени; o Внедрение Prometheus и Grafana для мониторинга метрик производительности и состояния системы.

• Резервное копирование: o Регулярное создание бэкапов базы данных и конфигурационных файлов; o Хранение резервных копий на удалённых хранилищах или облачных платформах.

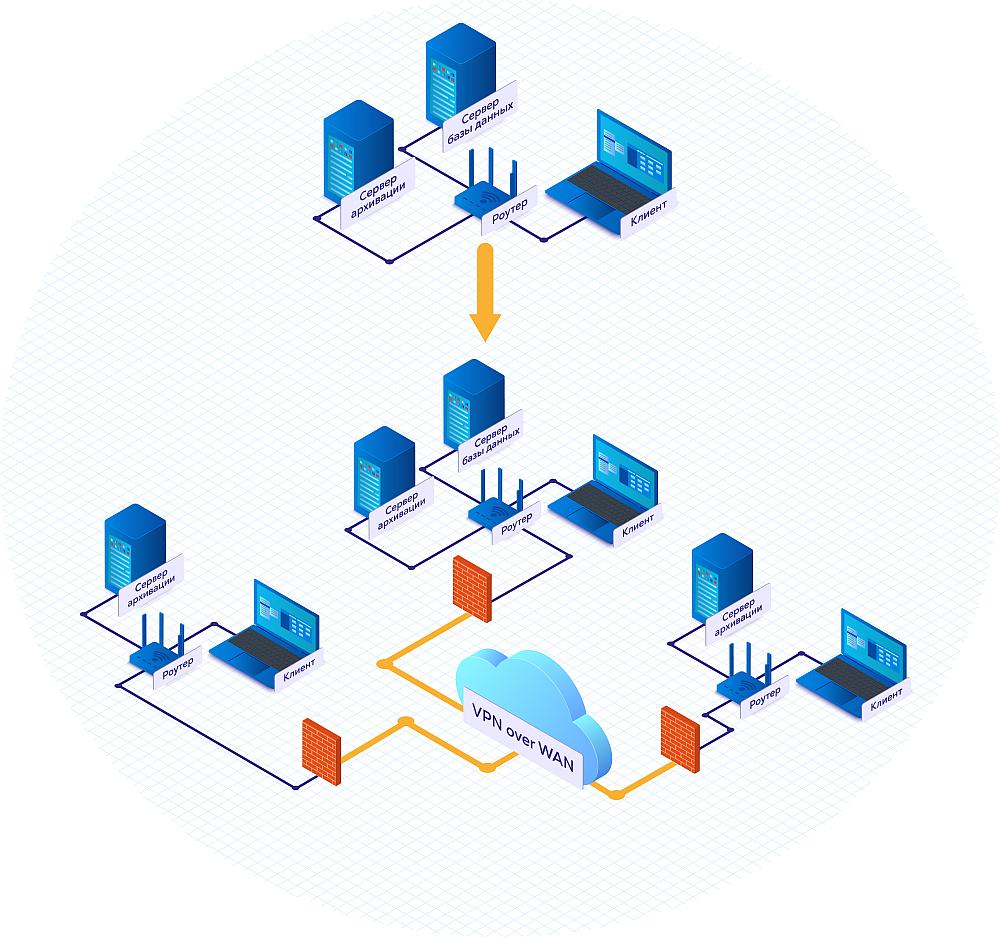


Рис 2.2.8 - Проектирование масштабируемости и производительности

### Документация

Документация необходима как для разработчиков, так и для конечных пользователей системы.

• Техническая документация: o Подробное описание архитектуры системы, включая диаграммы компонентов и их взаимодействия; o Описание моделей и их связей в базе данных; o API-документация с примерами запросов и ответов (в случае подключения Django REST Framework); o Инструкции по развёртыванию системы на разных средах (локально, staging, production); o Описание структуры каталогов и зависимостей проекта.

• Пользовательская документация: o Иллюстрированное руководство по работе с системой: регистрация, авторизация, создание и управление задачами и этапами, просмотр отчётов; o Часто задаваемые вопросы (FAQ); o Справка по настройкам уведомлений и персонализации интерфейса; o Руководство администратора системы: управление пользователями, настройка прав доступа, просмотр логов активности.

• Поддержка и сопровождение: o Каналы обратной связи: форма обращения, e-mail поддержки; o Журнал изменений (changelog) с указанием новых функций и исправлений; o План развития проекта (roadmap) с описанием ожидаемых обновлений и доработок.

### 2.3 Разработка и тестирование приложения. Deploy приложения

Весь фронтенд проекта был разработан с использованием чистого CSS и JavaScript, без применения фреймворков вроде Bootstrap 5. Главная страница имеет следующее представление (рисунок 2.3.2 – Категории в главной странице).



Рисунок 2.3.1 – Главная страница

Категории также отображены в главной странице приложения (показано в рисунке 2.3.2 – Категории в главной странице):

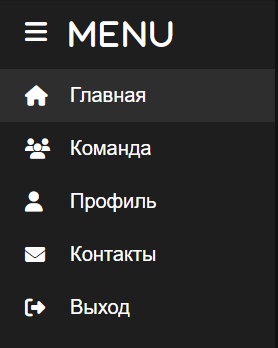


Рисунок 2.3.2 – Меню в главной странице

Меню выполнено с помощью ссылок <a> которые переносят пользователя на другую html страницу:

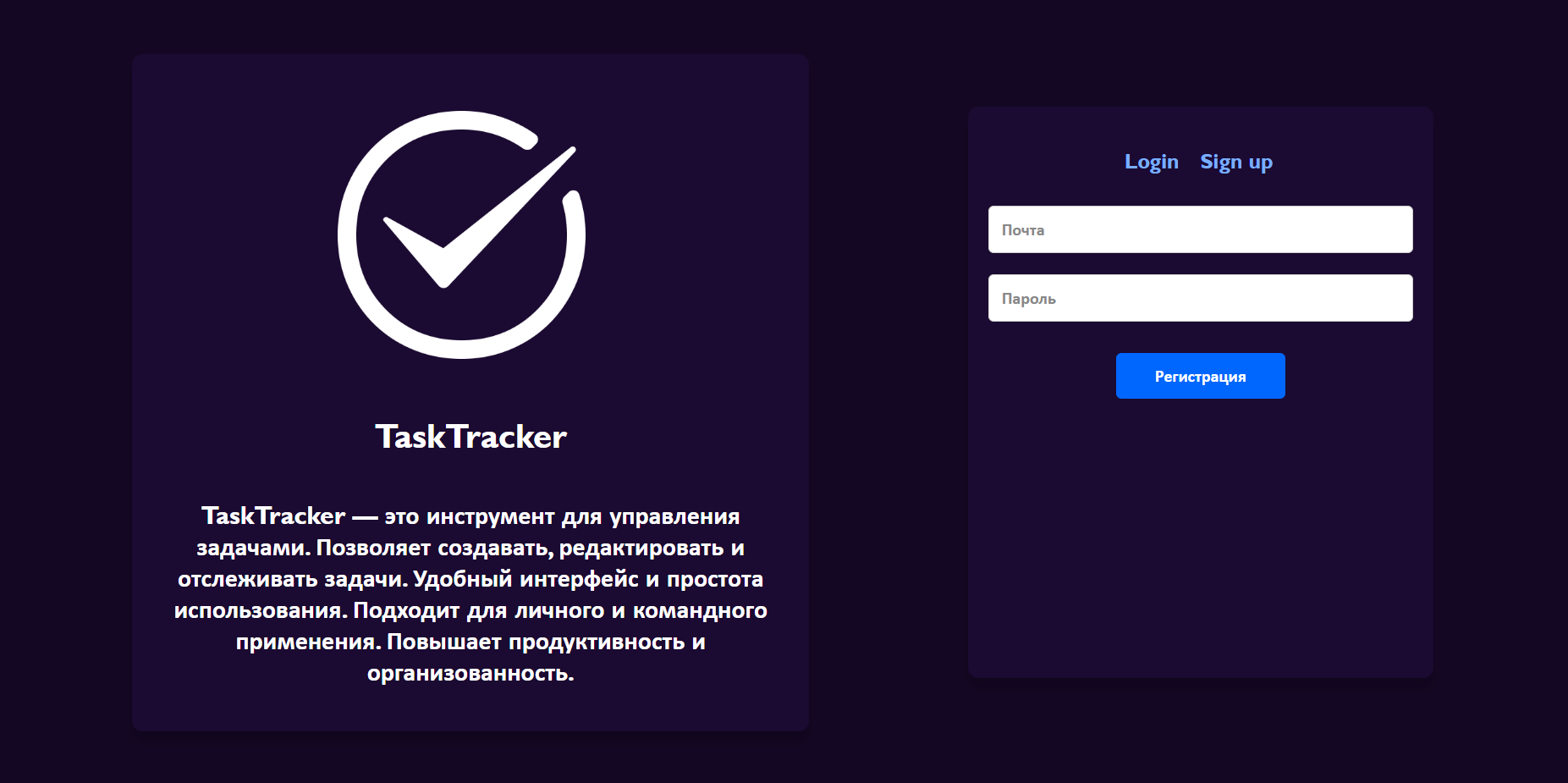


Рисунок 2.3.4 – Страница авторизации

Авторизация выполняется с помощью POST запроса, где основная форма это AuthenticationForm и поля для входа username & password:

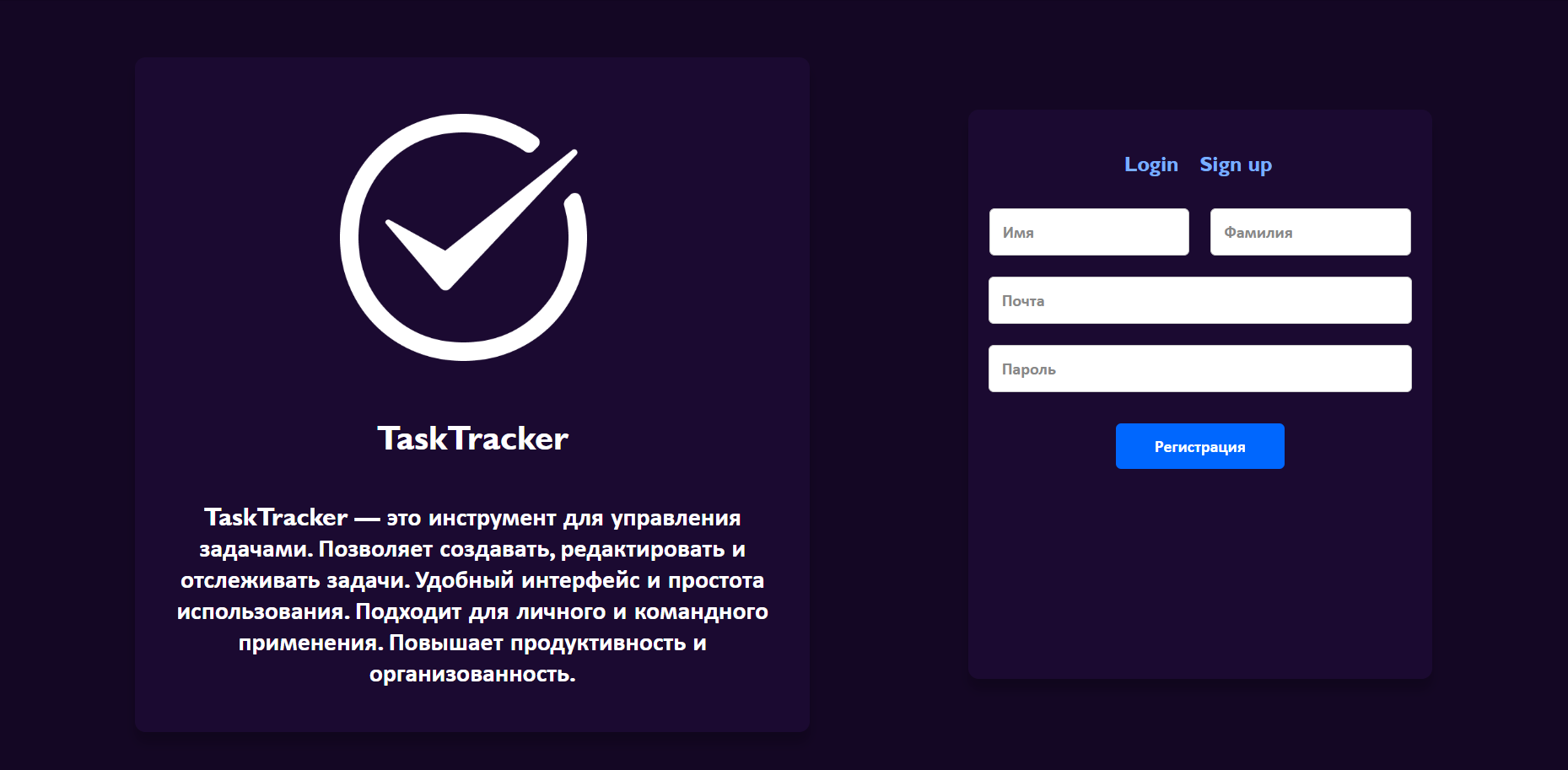


Рисунок 2.3.6 – Страница регистрации пользователя

Регистрация выполняется с помощью POST запроса, где основная форма это NewUserForm. При регистрации данные пользователя сохраняются в базу Sqlite3, после регистрации происходит процесс авторизации:

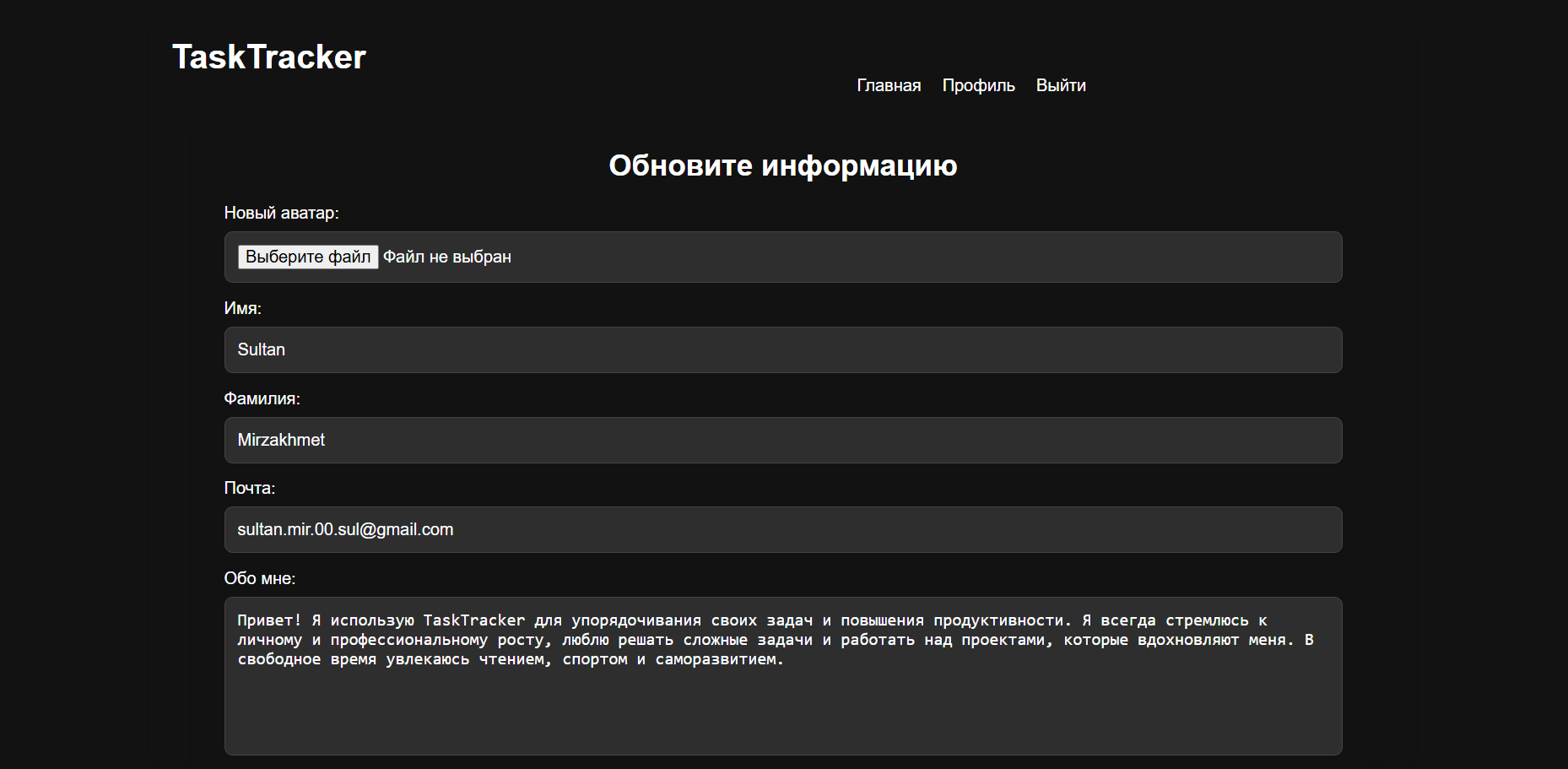


Рисунок 2.3.8 – Изменение профиля

1. **ЭКОНОМИЧЕСКАЯ ЧАСТЬ**
   1. **Технико** - **экономическое обоснование**

Технико-экономическое обоснование для разработки системы управления и координации командной работы базируется на ряде актуальных факторов.

Прежде всего, в условиях стремительного роста числа удалённых команд и проектов наблюдается потребность в эффективных цифровых инструментах для управления задачами и взаимодействия между участниками команды. Разработка подобной системы позволит обеспечить участников проекта удобным и централизованным решением для постановки задач, контроля за их выполнением, отслеживания прогресса и своевременного обмена информацией, что в конечном итоге повышает производительность и снижает риски организационных сбоев.

Во-вторых, внедрение такой системы позволяет улучшить прозрачность рабочих процессов и ускорить принятие управленческих решений. Руководитель получает возможность в реальном времени наблюдать за этапами выполнения задач, анализировать загруженность сотрудников и перераспределять ресурсы при необходимости. Это особенно важно для средних и крупных команд, работающих над комплексными проектами с множеством параллельных задач.

Разработка системы реализуется с использованием современных веб-технологий: Python и Django на серверной стороне обеспечивают надёжную логику и обработку данных, а HTML, CSS и JavaScript используются для создания интерактивного и удобного пользовательского интерфейса. Такой технологический стек позволяет быстро разрабатывать масштабируемые и устойчивые решения.

Кроме того, эксплуатация данной системы является экономически обоснованной, поскольку она может быть адаптирована под различные команды и отрасли. В перспективе возможно внедрение платного доступа к расширенному функционалу (например, аналитике, интеграции с внешними сервисами, автоматизированной отчётности), что открывает возможности для монетизации проекта.

Таким образом, разработка системы управления и координации командной работы отвечает современным требованиям цифровизации рабочих процессов, использует эффективные технологические инструменты и обладает значительным потенциалом как в плане повышения эффективности командной деятельности, так и в коммерческом аспекте.

* 1. **Расчет затрат на разработку**

Расчет затрат на разработку сайта для управления персональными финансами производится по формуле (1):

**Собщ = С осн.+Снакл. = (Сз. п. + Сс.о. + Сс.н. +Соосмс + Сэл. + Саморт. + ОПВ раб) + Снак.** (1)

**Где:**

**Собщ** - итоговые расходы

**Сз.п.** - заработная плата

**Сс.о** - социальные отчисления

**Сс.н.** - социальный налог

**Соосмс** - мед.страхование работодателя за работника

**С.аморт** - амортизация основных средств

**Снакл.** - накладные расходы

ОПВ раб – пенсионные взносы работодателя за работнтка

**Т** - период выполненной работы

* 1. **Расчет затрат на заработную плату**

Расчет затрат на заработную плату для программиста проекта сайта для управления персональными финансами производится по формуле (2):

**Сз.п = Ссрз/п.** (2)

**Где:**

**Сз.п** - заработная плата

**Сср.з/п** - средняя заработная плата

**Решение:**

Сз.п. = 260 000 тенге

**Пояснение:**

Средняя заработная плата программиста составляет 490 000 тенге. Однако, поскольку в проекте участвуют два разработчика, эту сумму следует распределить между ними. Указанная заработная плата считается оправданной с учётом объёма и сложности задач, поставленных в рамках разработки системы управления и координации командной деятельности.

Программист отвечает за реализацию функционала, включая создание личных кабинетов, инструментов для постановки задач, отслеживания их статуса, системы комментариев и уведомлений. Также необходимо разработать логическую часть взаимодействия клиентской и серверной стороны, обеспечив стабильную работу всех модулей системы.

Важным аспектом является реализация системы разграничения прав доступа и обеспечения безопасности, так как участники команды работают с конфиденциальной информацией. Программист должен реализовать защиту данных, провести тестирование интерфейсов и бизнес-логики, а также устранить выявленные ошибки до внедрения системы.

Кроме того, в его обязанности входит сопровождение проекта после запуска: внесение улучшений, обновлений и предоставление технической поддержки пользователям. Всё это требует высокой квалификации и значительных временных затрат, что и обосновывает указанную заработную плату.

* 1. **Расчет затрат на социальные отчисления**

Расчет затрат на социальные отчисления для программиста проекта сайта для управления персональными финансами производится по формулам (3), (4), (5):

*Сс.о = (Сз.п – 10%) 5%*\* (3)

**Решение:**

Сс.о = (260 000 – 10%) \* 5% = **11 700 тенге**

**Свосмс = Сз.п × 2%** (5)

**Решение:**

Свосмс = 260 000 \* 2% = **5 200 тенге**

**Соосмс = 260 000 \* 3% = 7 800 тенге**

**Сс.н = (Сз.п – 10% Сз.п – 2% Сз.п) \* 11% – Сс.о** (4)  
Сс.н = (260 000 – 10% – Свосмс) \* 11% – Сс.о  
Сс.н = (260 000 – 26 000 – 5 200) \* 11% – 11 700 = **13 178 тенге**

* 1. **Расчет затрат на электроэнергию**

Расчет затрат на электроэнергию потраченную в период разработки производится по формуле (6):

**Сэл = tdwq** (6)

**Где:**

**t –** время работы компьютера в день, час

**d–** количество рабочих дней в месяце

**w–** мощность компьютера, кВТ

**q–** стоимость единицы электроэнергии, тг/кВт

**Сэл** – расход на электроэнергию

**Решение:**

Сэл = 8 \* 22 \* 2.1 \* 38 = **14 025.6 тенге**

**Пояснение:**

Количество рабочих дней в апреле составляет 22 дня, мощность компьютера программиста составляет 2квт, так как данный компьютер оснащен хорошей сборкой для удобной работы в программах, которые нужны для разработки.

Стоимость единицы электроэнергии в данный период времени с учетом НДС составляет 36,6 тенге

Время работы компьютера в день составляет 7 часов, не считая времени, когда компьютер находится в спящем режиме.

**Расчет амортизационных отчислений основных средств**

Расчет затрат на амортизацию основных средств в период разработки производится по формуле (7):

**Саморт = Рк \* Na : K** (7)

**Где:**

**Рк –** стоимость компьютера, тг

**Na –** норма амортизации, %

**K –** количество месяцев в году,

Для расчета нормы амортизации воспользуемся следующей формулой (8):

**Na =1/n\*100% (**8)

**Решение:**

**Na=1/n\*100%=1/6\*100%=16,67%**  
**Саморт = 530 000 \* 16,67% : 12 = 7 370,8 тенге**

**Пояснение:**

Стоимость компьютеров Lenovo IdeaPad Gaming 3 16IAH7 и ASUS TUF Gaming F15, используемых разработчиками, составляет 1 000 000 тенге. Эти устройства оснащены качественными комплектующими, что обеспечивает стабильную и бесперебойную работу современных программ для разработки программного обеспечения.

Норма амортизации составляет 20%.

* 1. **Расчет накладных расходов**

Расчет затрат на накладные расходы в период разработки производится по формуле (9):

**Снакл. = Сосн. \* 25% = (Сз.п + Сс.о + Сс.н + Соосмс + Саморт + ОПВ раб. + Сэл) \* 25%**  
**= (260000 + 11700 + 13178 + 7800 + 7370,8 + 6450 + 14025,6) \* 25%**  
**= 318524,4 \* 25% = 79631,1 тенге** (9)

**Где:**

**Снак –** накладные расходы

**Сз/п** – заработная плата

**Сс.о** – социальные отчисления

**Сс.н** – социальный налог

**Соосмс** – отчисления на медицинское страхование работника от работодателя

С восмс медицинское страхование работника

**Саморт** – амортизация основных средств

**Сэл** – расход на электроэнергию

**ОПВ раб (пенсионные отчисления работодателя за работника) составляют 2,5% от Сз.п:= 260 000 \* 2,5% = 6 500 тенге**

**Решение:**

Снакл. = Сосн \* 25% = 318524,4 \* 25% = 79 631,1 тенге

* 1. **Расчет на общие расходы**

Расчет общих затрат за месяц на разработку сайта для управления персональными финансами производится по формуле (1):

**Решение:**

Собщ = 318524,4 + 79 631,1 = 398 155,5 тенге

Таким образом, затраты на разработку сайта двумя программистами в течение одного месяца составят 796 311 тенге.

## **БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ**

### Характеристика помещения

Рассмотрим помещение, в котором будет применяться программное обеспечение (рисунок ниже (будет после уточнения)). Помещение имеет размеры: длина (L) = 5 метров, ширина (B) = 6 метров, высота окна (H) = 1,5 метров. Имеет 1 окно с южной стороны, 15 посадочных, рабочих мест и 4 светильника ЛБ2\*40. Имеется огнетушитель, план эвакуации и тревожная кнопка. В аудитории есть 10 ПК.

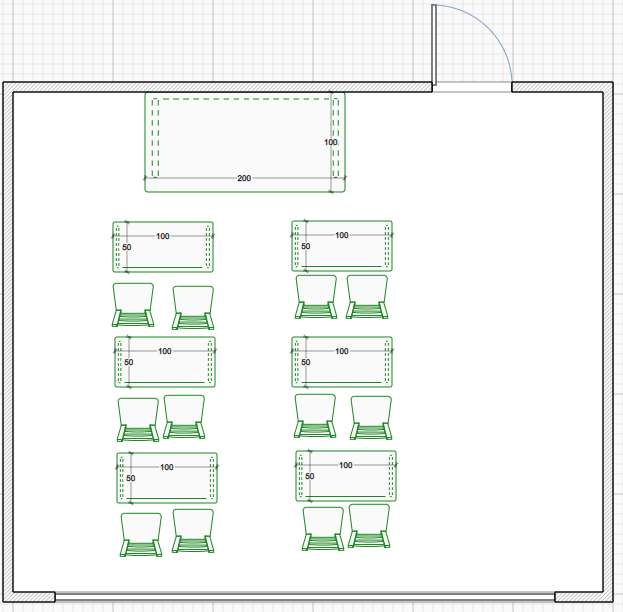


Рисунок 4.1.1 – План аудитории.

### Анализ аудитории

Узнав все параметры и характеристики нашей аудитории, мы можем сделать выводы, что естественное освещение в кабинете представляет собой немало важное значение, по этой причине размещение компьютеров выполнено в нужном порядке, потому что свет с окон не падает непосредственно на сидячего. Таким образом, не подвергая студента к переутомлению глаза при работе.

Меры безопасности полностью соблюдены. Правильное расположение техники и электрокабелей в кабинете предусмотрено, а также строгое соблюдение мер пожарной безопасности (огнетушитель, план эвакуации и тревожная кнопка).

Аудитория расположена в здании Алматинского университета энергетики и связи, на 3 этаже. Помещение отдаленно проезжей части дороги, таким образом различные источники шума, не способны оказывать влияние на процесс работы студентов.

Для предотвращения воздействия статического электричества, а также возникновения электромагнитного воздействия на человека рекомендуется регулярная вентиляция в помещении. К сожалению, в аудитории имеется только естественный обдув при открывании окон.

При выборе оборудования для системы вентиляции необходимо рассчитать следующие параметры:

* производительность по воздуху;
* мощность калорифера;
* рабочее давление, создаваемое вентилятором;
* скорость потока воздуха и площадь сечения воздуховодов;
* допустимый уровень шума.

### Производительность по воздуху

Проектирование системы вентиляции начинается с расчета требуемой производительности по воздуху или «прокачки», измеряемой в кубометрах в час. Для этого необходим поэтажный план помещений с экспликацией, в которой указаны наименования (назначения) каждого помещения и его площадь.

Расчет начинается с определения требуемой кратности воздухообмена, которая показывает сколько раз в течение одного часа происходит полная смена воздуха в помещении. Например, для помещения площадью 50 квадратных метров с высотой потолков 3 метра (объем 150 кубометров) двукратный воздухообмен соответствует 300 кубометров в час. Требуемая кратность воздухообмена зависит от назначения помещения, количества находящихся в нем людей, мощности тепловыделяющего оборудования и определяется СНиП (Строительными Нормами и Правилами). Так, для большинства жилых помещений достаточно однократного воздухообмена, для офисных помещений

требуется 2-3 кратный воздухообмен. Для определения требуемой производительности необходимо рассчитать два значения воздухообмена: по кратности по количеству людей, после чего выбрать большее из этих двух значений.

Расчет воздухообмена по кратности: L = n \* S \* H, где:

L – требуемая производительность приточной вентиляции м3/ч;

n – нормируемая кратность воздухообмена для жилых помещений; n = 1, для офисов;

n = 2,5;

S – площадь помещения;

м2; H – высота помещения, м2.

Расчет воздухообмена по количеству людей: L = N \* Lнорм, где, L – требуемая производительность приточной вентиляции, м3/ч; N – количество людей;

Lнорм – норма расхода воздуха на одного человека: 2.

В состоянии покоя – 20 м3/ч, работа в офисе – 40 м3/ч, при физической нагрузке – 60 м3/ч.

Рассчитав необходимый воздухообмен, выбираем вентилятор или приточную установку соответствующей производительности. При этом необходимо учитывать, что из-за сопротивления воздухопроводной сети происходит падение производительности вентилятора. Зависимость производительности от полного давления можно найти по вентиляционным характеристикам, которые приводятся в технических характеристиках оборудования.

Мощность калорифера: Калорифер используется в приточной системе вентиляции для подогрева наружного воздуха в холодное время года. Мощность калорифера рассчитывается исходя из производительности системы вентиляции, требуемой температурой воздуха на выходе системы и минимальной температурой наружного воздуха. Два последних параметра определяются СНиП. Температура воздуха, поступающего в жилое помещение, должна быть не ниже +18°С. Минимальная температура наружного воздуха зависит от климатической зоны и для Алматы равна -26°С (рассчитывается как средняя температура самой холодной пятидневки самого холодного месяца в 13 часов).

Таким образом, при включении калорифера на полную мощность он должен нагревать поток воздуха на 44°С. Поскольку сильные морозы в Алматы непродолжительны, в приточных системах можно устанавливать калориферы, имеющие мощность меньше расчетной. При этом приточная система должна иметь регулятор производительности для уменьшения скорости вентилятора в холодное время года.

При расчете мощности калорифера необходимо учитывать следующие ограничения: Возможность использования однофазного (220 В) или трехфазного

(380 В) напряжения питания. При мощности калорифера свыше 5 кВт необходимо 3-х фазное подключение, но в любом случае 3-х фазное питание предпочтительней, так как рабочий ток в этом случае меньше. Максимально допустимый ток потребления. Ток, потребляемый калорифером, можно найти по формуле: I = P / U, где

I – максимальный потребляемый ток, А;

Р – мощность калорифера, Вт; U – напряжение питание (220 В – для однофазного питания; 660 В (3 × 220В) – для трехфазного питания.)

В случае если допустимая нагрузка электрической сети меньше, чем требуемая, можно установить калорифер меньшей мощности. Температуру, на которую калорифер сможет нагреть приточный воздух, можно рассчитать по формуле: ΔT = 2,98 \* P / L, где

ΔT – разность температур воздуха на входе и выходе системы приточной вентиляции,°С;

Р — мощность калорифера, Вт;

L – производительность вентиляции, м3/ч.

Типичные значения расчетной мощности калорифера – от 1 до 5 кВт для квартир, от 5 до 50 кВт для офисов.

Если использовать электрический калорифер с расчетной мощностью не представляется возможным, следует установить калорифер, использующий в качестве источника тепла воду из системы центрального или автономного отопления (водяной калорифер).

Рабочее давление, скорость движения воздуха в воздуховодах, уровень шума.

После расчета производительности по воздуху и мощности калорифера приступают к проектированию воздухораспределительной сети, которая состоит из воздуховодов, фасонных изделий (переходников, разветвителей, поворотов) и распределителей воздуха (решеток или диффузоров). Расчет воздухораспределительной сети начинают с составления схемы воздуховодов. Далее по этой схеме рассчитывают три взаимосвязанных параметра — рабочее давление, создаваемое вентилятором, скорость потока воздуха и уровень шума. Воздухообмен в производственных помещениях необходим для очистки воздуха от вредностей: для удаления вредных веществ (выделяющихся вредных газов, паров и пыли); для удаления избыточного тепла.

В данной работе произведен расчет потребного воздухообмена (L м3/ч), для очистки воздуха от вредных газов и паров и для удаления избыточного тепла с помощью механической общеобменной вентиляции.

### Расчет воздухообмена

Потребный воздухообмен определяется по формуле 4.1:

*L*  *G*1000

*xв*  *xн*

, м3/ч (4.1)

где, L, м3/ч – потребный воздухообмен;

G, г/ч – количество вредных веществ, выделяющихся в воздух помещения; xв, мг/м3 – предельно допустимая концентрация вредности в воздухе рабочей зоны помещения, согласно ГОСТ 12.1.005-88 по для бутил ацетата равен

200 мг/м3;

xн, мг/м3 – максимально возможная концентрация той же вредности в воздухе населенных мест, согласно СН-3086-84 для бутил ацетата равен 0,1 мг/м3. Применяется также понятие кратности воздухообмена (n), которая показывает сколько раз в течение одного часа воздух полностью сменяется в помещении. Значение n  3 может быть достигнуто естественным

воздухообменом без устройства механической вентиляции.

Кратность воздухообмена определяется по формуле 4.4.2:

n = L/Vп, ч-1 (4.2)

где, Vп =А х В х Н = 30 х 20 х 5 = 3000 м3 – внутренний объем помещения, м3. Согласно СН 245-71, кратность воздухообмена n 10 недопустима.

Для расчета потребного воздухообмена необходимо определить количество вредных веществ, выделяющихся в воздух помещения.

Испарение растворителей и лаков обычно происходит при покраске различных изделий. Количество летучих растворителей, выделяющихся в воздухе помещений можно определить по следующей формуле 4.4.3:

*G*  *a* *A**m**n* , г/ч (4.3)

100

где, а, м2/ч - средняя производительность по покраске одного рабочего, составляющая при покраске пульверизатором а=50 м2/ч;

А = 180 г/м2 – расход лакокрасочных материалов;

m = 75 % – процент летучих растворителей, содержащихся в лакокрасочных материалах;

n = 4 – число рабочих, одновременно занятых на покраске. Подставив данные значения в выражение 3, получим:

G = 50 х 180 х 75 х 4 /100 = 27000, 0 г/ч.

Тогда величина потребного воздухообмена в соответствии с выражением 1 составит:

L = 1000 х 27000 / (200 – 0,1) = 135678 м3/ч,

А потребная кратность воздухообмена в соответствии с формулой 2 может быть определена как:

n = 135678 / 3000 = 45,2.

Так как согласно СН 245-71, кратность воздухообмена n 10 недопустима, можно сделать вывод, что производить такое количество покрасочных работ в данном помещении недопустимо.

# Расчет потребного воздухообмена для удаления избыточного тепла

Расчет потребного воздухообмена для удаления избыточного тепла производится по формуле 4.5.1:

*L*  *Qизб*

, м3/ч (4.4)

 *с* *t*

где, L, м3/ч – потребный воздухообмен; Qизб, ккал/ч – избыточное тепло;

в = 1.206 кг/м3 – удельная масса приточного воздуха; c в = 0,24 ккал/кг.град – теплоемкость воздуха.

t = t вых - t пр = 24 – 19 = 5, oC (4.5) где, t вых, oC – температура уделяемого воздуха;

t пр, oC – температура приточного воздуха.

Таким образом, для определения потребного воздухообмена необходимо определить количество избыточного тепла по формуле 4.5.3:

Qизб = Qоб + Qосв + Qл + Qр - Qотд , ккал/ч (4.6) где, Qоб, ккал/ч – тепло, выделяемое оборудованием;

Qосв, ккал/ч – тепло, выделяемое системой освещения;

Qл, ккал/ч – тепло, выделяемое людьми в помещении;

Qр, ккал/ч – тепло, вносимое за счет солнечной радиации; Qотд, ккал/ч – теплоотдача естественным путем.

Определяем количество тепла, выделяемого оборудованием

Qоб = 860 Роб  1 = 860 х 0,26 х 0,25 = 55, ккал/ч (4.7) где, Y1 = 1 – коэффициент перехода тепла в помещение, зависящий от вида

оборудования;

Роб= 0.26, кВт - мощность, потребляемая оборудованием;

Роб, = Рном Y2 Y3  Y4, кВт (4.8) где, Рном, – номинальная (установленная) мощность электрооборудования

помещения;

Y2 – коэффициент использования установленной мощности, учитывающий превышение номинальной мощности над фактически необходимой;

Y3 – коэффициент загрузки, т.е. отношение величины среднего потребления мощности (во времени) к максимально необходимой;

Y4 – коэффициент одновременности работы оборудования.

При ориентировочных расчетах произведение всех четырех коэффициентов можно принимать равным:

Y1 Y2  Y3 Y4 = 0,25 (4.9)

Определяем количество тепла, выделяемого системой освещения:

Qосв = 860  Росв    b  cos () = 860 х 1,68 х 0,47 х 0,75 = 509 ккал/ч (4.10) где,  = 0,47 – коэф.перевода электрической энергии в тепловую,

b – коэффициент одновременности работы (при работе всех светильников

b = 1);

сos() = 0,7 - 0,8 – коэффициент мощности;

Росв= 42 х 0,04 = 1,68, кВт – мощность осветительной установки.

Определяем количество тепла, выделяемого находящимися в помещении людьми.

Qл = N \* qл= 15 х 60 = 900, ккал/ч (4.11) где, N =15 – количество людей в помещении;

qл = 60, ккал/ч – тепловыделения одного человека.

Определяем количество тепла, вносимого за счет солнечной радиации:

Qр = m  S qост = 1,5 х 5 х 125 = 937,5 ккал/ч (4.12)

где, m = 8 + 1 – количество окон S = 3, м2 – площадь одного окна

qост=125 ккал/ч – солнечная радиация через остекленную поверхность. Определяем теплоотдачу, происходящую естественным путем

Если нет дополнительных условий, то можно считать ориентировочно, что Qотд = Qр для холодного и переходного периодов года (среднесуточная

температура наружного воздуха ниже +10 oC).

Для теплого периода года (среднесуточная температура воздуха выше +10

oC) принимаем Qотд = 0.

Таким образом в соответствии с выражением, полученным ранее, получаем:

Qизб = Qоб + Qосв + Qл + Qр - Qотд = 55+509+900+937,5-937,5 = 1,464 ккал/ч.

В соответствии с потребным воздухообменом может быть вычислен: L = 60264 / (1,206 х 0,24 х 5) = 41,642 м3/ч,

Значит кратность воздухообмена равна:

n = 41642 / 3000 = 13,88

### Выводы

Таким образом, для очистки воздуха от вредных паров следует применять систему вентиляции, которая обеспечивает требуемую подачу воздуха Q = 41642 (м3/ч). Для обеспечения требуемой подачи воздуха был выбрать кондиционер Ditreex 24 F12 (R410). Данный кондиционер обеспечивает подачу воздуха до 900 м3 /ч. Характеристики выбранного кондиционера описаны в таблице 5.1, а внешний вид представлен на рисунке 4.6.1.



Рисунок 4.6.1 – Кондиционер Ditreex 24 F12 (R410)

Таблица 1 – Характеристики кондиционера

|  |  |
| --- | --- |
| Параметр | Значение |
| Производительность по холоду (Вт) | 7100 |
| Потребляемая мощность в режиме охлаждения (Вт) | 2510 |
| Рекомендуемая площадь охлаждения/обогрева (м2) | 60 |
| Количество конденсата (l/h) | 2 |
| EER/C.O.P. в режиме охлаждения (Вт/Вт) | 2,81/3,21 |
| Производительность по теплу (Вт) | 7300 |
| Потребляемая мощность в режиме обогрева (Вт) | 2280 |
| Потребляемый ток в режиме обогрева (A) | 11,2 |
| EER/C.O.P. в режиме обогрева (Вт/Вт) | 2,8 |
| Расход воздуха внутренним блоком (м3/ч) | 900/1050/1150 |
| Уровень шума внутреннего блока (дБ (А)) | 41/45/48 |
| Длина внутреннего блока (мм) | 1045 |

В аудитории 329 корпуса B кафедры IT-инжиниринг мы улучшили климатические условия для оптимальной и продуктивной работы установив кондиционер Ditreex 24 F12 (R410).

### ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы были выполнены проектирование, разработка и тестирование программного инструмента для отслеживания этапов выполнения задач. В результате проведённой работы удалось создать программное решение, которое позволяет повысить эффективность управления задачами, облегчить контроль над процессами, минимизировать ошибки, связанные с человеческим фактором, и повысить продуктивность труда.

Разработанное приложение обеспечивает удобный и интуитивно понятный интерфейс, доступный пользователям с различным уровнем технической подготовки. Система предоставляет функциональные возможности для мониторинга задач, аналитики и отчётности, что делает её универсальным инструментом как для индивидуального использования, так и для командного взаимодействия. Кроме того, предусмотрена возможность интеграции с другими корпоративными сервисами и инструментами.

Внедрение данного программного продукта способствует улучшению организационной культуры и рабочего климата, повышению общей результативности компании и снижению издержек на реализацию проектов. Программное решение гибко масштабируется и может быть адаптировано под конкретные потребности различных команд и организаций.

Таким образом, цели, поставленные в начале дипломной работы, были успешно достигнуты. Итоговый продукт демонстрирует практическую значимость предложенного подхода к управлению задачами и обладает потенциалом для дальнейшего развития и совершенствования.

### ****ПРИЛОЖЕНИЯ:****

Ссылка на github: <https://github.com/ad1l7/django-project.git>

QR-code:

