Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Название проекта»

по дисциплине «Проектный практикум»

Заказчик: Макаров Артем Васильевич. _____

Куратор: Макаров Артем Васильевич. _____

Студенты команды свага

Погирейчик Андрей Александрович. _____

Беликов Никита Васильевич. _____

Егоров Евгений Андреевич. _____

Якшибаев Данил Димович. _____

Уварова Ольга Александровна. _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях стремительного развития индустрии разработки программного обеспечения, где проекты становятся всё более сложными, а требования к срокам и качеству возрастают, эффективное управление задачами является критически важным фактором успеха [1]. Разрозненные инструменты, такие как электронные таблицы или устаревшие системы, уже не способны обеспечить необходимую прозрачность, гибкость и скорость взаимодействия в командах, включающих аналитиков, разработчиков, тестировщиков, дизайнеров и руководителей проектов. Для решения этих проблем был инициирован проект по созданию веб-приложения, предназначенного для централизованного управления задачами команды разработки. Система направлена на упрощение процессов планирования, отслеживания и выполнения задач, предоставляя пользователям интуитивно понятный интерфейс, инструменты визуализации и возможности анализа эффективности работы.

Разрабатываемое приложение ориентировано на повышение производительности команд за счёт автоматизации управления задачами, обеспечения прозрачности процессов и контроля сроков выполнения. Оно позволяет создавать, редактировать, удалять и назначать задачи, поддерживает работу по спринтам, а также предоставляет функционал фильтрации, поиска и организации задач с использованием тегов и вложенности. Система предусматривает сбор метрик, таких как количество завершённых задач, время их выполнения и динамика изменения статусов, что даёт руководителям возможность принимать обоснованные решения на основе данных. Кроме того, приложение включает инструменты визуализации, такие как диаграмма Ганта, Burndown chart и Cumulative Flow Diagram, которые обеспечивают наглядное представление прогресса и помогают выявлять узкие места в рабочих процессах [2].

Применение системы рассчитано на команды разработки программного обеспечения, включающие специалистов различного профиля: аналитиков, планирующих и распределяющих задачи; разработчиков, выполняющих задания; тестировщиков, контроли качества; дизайнеров, работающих над пользовательским интерфейсом; а также руководителей проектов, отслеживающих общий прогресс. Программа применима как для небольших стартапов, так и для крупных организаций с распределёнными командами, где требуется управление большим количеством задач. Использование микросервисной архитектуры, контейнеризации и современного технологического стека обеспечивает гибкость, масштабируемость и адаптивность системы к различным условиям эксплуатации.

По завершении проекта ожидается создание полнофункционального веб-приложения, соответствующего заявленным требованиям и обеспечивающего высокую производительность, включая обработку до 1000 запросов в секунду с временем отклика не более 200 мс для 95% запросов и поддержку до 1 000 000 задач. Система предоставит RESTful API для интеграции с внешними платформами, функционал нотификаций и журнал аудита изменений, а также возможность формирования аналитических отчётов. В ходе разработки был сформирован план задач, прикрепленный в таблице А приложения А, который уточнялся на каждом этапе итераций, что позволило гибко реагировать на изменения требований. Ожидается, что внедрение системы сократит время выполнения задач на 15–20%, повысит прозрачность процессов [3] и создаст прочную основу для дальнейшего развития, включая расширение функционала и интеграцию с другими инструментами управления проектами.

1 Аналитика

1.1 Анализ и сопоставление аналогов разрабатываемого продукта

В данном разделе рассмотрим пять популярных инструментов управления проектами: Trello, Yandex Tracker, Jira, GitHub Projects и Strive.app — в качестве аналогов разрабатываемой системы. Каждый из этих инструментов имеет свои особенности, преимущества и недостатки, которые были проанализированы и учитывались при разработке.

1.1.1 Trello

Trello — это известный инструмент для управления проектами, построенный на концепции визуальных Канбан-досок. Он ориентирован на простоту и удобство использования.

Преимущества:

– Простота: интуитивно понятный интерфейс позволяет быстро создавать задачи (карточки) и перемещать их по доскам.

– Визуализация: Канбан-доски дают ясное представление о статусе задач [4].

– Интеграции: поддерживает подключение к популярным сервисам, таким как Slack, Google Drive и GitHub.

Недостатки:

– Ограниченная функциональность: отсутствуют продвинутые возможности, такие как управление зависимостями задач или аналитика.

– Масштабируемость: при большом количестве задач доски становятся громоздкими и менее удобными.

1.1.2 Yandex Tracker

Yandex Tracker — российский инструмент управления проектами от компании Яндекс, ориентированный на команды, использующие Agile-методологии.

Преимущества:

– Agile-поддержка: инструменты для спринтов, планов задач и Канбан-досок.

– Интеграция с экосистемой Яндекс: удобная связь с сервисами Яндекс (почта, облако).

– Кастомизация: настраиваемые рабочие процессы и поля задач.

Недостатки:

– Региональная специфика: основной фокус на российский рынок, что может ограничивать глобальное использование (например, из-за языковых или технических барьеров).

– Сложность для новичков: требует знакомства с Agile для эффективного использования.

1.1.3 Jira

Jira — один из лидеров среди инструментов управления проектами, особенно популярный в разработке ПО. Он поддерживает Agile, Scrum и Канбан.

Преимущества:

– Гибкость: широкие возможности настройки рабочих процессов и типов задач.

– Agile-функции: поддержка спринтов, отчетов (Burndown Chart, Velocity Chart) и планирования задач.

– Интеграции: совместимость с инструментами разработки.

Недостатки:

– Сложность: интерфейс может быть перегруженным и требует времени на освоение.

– Производительность: при больших объемах данных возможны задержки.

1.1.4 GitHub Projects

GitHub Projects — инструмент управления задачами, встроенный в платформу GitHub, предназначенный в первую очередь для разработчиков.

Преимущества:

– Интеграция с GitHub: прямая связь с репозиториями, коммитами и иными запросами.

– Простота: базовые функции Канбан-досок и списков задач.

– Удобство для разработчиков: упрощает отслеживание задач, связанных с кодом.

Недостатки:

– Ограниченная функциональность: нет поддержки спринтов или глубокой аналитики.

– Узкая специализация: не подходит для задач, не связанных с разработкой.

GitHub Projects удобен для команд, работающих с кодом на GitHub, но для более широких нужд его возможностей недостаточно.

1.1.5 Strive.app

Strive.app — относительно новый инструмент, акцентирующий внимание на производительности и удобстве совместной работы.

Преимущества

– Фокус на продуктивности: инструменты для управления временем и приоритетами.

– Интуитивный интерфейс: современный и удобный дизайн.

– Документация: возможность прикреплять файлы для документирования пространств и проектов.

Недостатки

– Малая известность: ограниченная поддержка сообщества и интеграций.

– Функциональность: может не покрывать потребности сложных проектов.

На основе анализа аналогов система была спроектирована с учетом их преимуществ и недостатков. Было решено создать решение, которое:

– Обеспечивает гибкость настройки рабочих процессов, как в Jira, но с более простым и интуитивным интерфейсом, вдохновленным Trello.

– Включает продвинутые инструменты аналитики, превосходящие возможности Trello, но остающиеся доступными для команд любого размера.

Исходя из анализа конкурентов были составлены требования, на основе которых создавался продукт.

1.2 Анализ требований заказчика и пользователей к программному продукту

1.2.1 Обзор проблемы и предлагаемого решения

Текущая ситуация показывает, что используемое программное обеспечение не удовлетворяет потребности заказчика в области трекинга задач и аналитики. Основные проблемы текущего программного обеспечения включают:

– Ограниченный функционал: недостаточные возможности для управления задачами и анализа данных.

– Отсутствие интеграций: нет механизмов взаимодействия с внешними системами.

– Невозможность доработки: отсутствует гибкость для добавления новых функций.

– Ограниченная доступность: некоторые решения заблокированы на территории Российской Федерации, что создаёт барьеры для использования.

Для решения этих проблем заказчик инициировал разработку нового веб-приложения — трекера задач с расширенным функционалом. Этот инструмент должен не только покрывать текущие потребности в управлении задачами и аналитике, но и быть гибким для оптимизации под будущие требования. Целевая аудитория — команды, которые организуют свою работу через доски задач, такие как Канбан или Скрам-доски.

1.2.2 Ключевые требования заказчика и пользователей

На основе описания проблемы, решения и целевой аудитории были выделены следующие требования к новому продукту. Они разделены на функциональные и нефункциональные категории.

Функциональные требования

– Управление задачами:
    - Создание, редактирование, удаление задач с указанием заголовка, описания, приоритета и исполнителя.
    - Отслеживание статусов задач (например, «Новая», «В работе», «Завершена»).
    - Фильтрация и поиск задач по различным параметрам (статус, исполнитель, приоритет).
    - Поддержка спринтов, вложенных задач (подзадач) и тегов для категоризации.
– Аналитика и отчётность:
    - Сбор метрик: количество завершённых задач, время нахождения задач в каждом статусе, общее время выполнения задач.
    - Генерация отчётов и визуализация данных (например, через диаграммы).
– Визуализация процессов:
    - Поддержка визуальных инструментов, таких как Канбан-доски, диаграммы Ганта или Burndown chart, для отслеживания прогресса.
– Управление пользователями:
    - Регистрация и аутентификация пользователей.
    - Разграничение ролей (например, администратор, участник команды) с разными уровнями доступа.
– Интеграции:

- Возможность подключения к внешним системам (например, системы контроля версий, мессенджеры, CI/CD-инструменты) через API или готовые модули.

– Гибкость и расширяемость:

- Архитектура, позволяющая добавлять новые функции и модули в будущем без значительных изменений в коде.
- Нефункциональные требования

– Доступность в РФ:

- Система должна быть полностью работоспособна на территории Российской Федерации, избегая использования технологий или сервисов, которые заблокированы в стране.

– Производительность:

- Быстрая обработка операций с задачами (например, создание, обновление, удаление).
- Способность выдерживать значительную нагрузку (например, сотни пользователей и тысячи задач).

– Безопасность:

- Шифрование данных пользователей и задач.
- Безопасная аутентификация и контроль доступа.

– Удобство использования:

- Интуитивно понятный интерфейс, адаптированный для командной работы через доски задач.
- Поддержка работы на разных устройствах (десктоп, мобильные устройства).

1.3 Учёт специфики целевой аудитории

Целевая аудитория — команды, работающие с досками задач, — требует:

– Простого и наглядного управления задачами через визуальные инструменты (например, Канбан-доски).

– Быстрого доступа к аналитике для оценки эффективности работы.

– Гибкости в настройке процессов и интеграции с инструментами, которые уже используются в команде.

2 Обзор архитектуры программного продукта

Архитектура системы построена на принципах микросервисного подхода, что обеспечивает гибкость, масштабируемость и высокую отказоустойчивость. Такой выбор был обусловлен необходимостью создания системы, способной адаптироваться к изменениям требований и поддерживать независимую разработку и развертывание компонентов.

2.1 Основные компоненты

2.1.1 Сервис управления задачами

– Функционал:

- Создание, редактирование, удаление задач.
- Управление статусами задач и их категоризацией.
- Фильтрация и поиск задач по различным параметрам (приоритет, исполнитель, дедлайн).

– Роль: центральный элемент системы, обеспечивающий взаимодействие пользователей с задачами.

2.1.2 Сервис аналитики и метрик

– Функционал:

- Сбор данных о выполнении задач (время выполнения, количество завершенных задач).
- Генерация отчетов и дашбордов (динамика статусов, производительность команды).
- Анализ зависимостей и узких мест в рабочих процессах.

– Роль: предоставляет командам инструменты для оценки эффективности и планирования.

2.1.3 Сервис пользователей

– Функционал:

- Управление учетными записями, ролями и правами доступа.
- Аутентификация (OAuth, JWT) и авторизация пользователей.

– Роль: обеспечивает безопасность и контроль доступа к системе.

2.1.4 Шина сообщений

– Функционал:

- Асинхронная передача событий между сервисами, используется для сбора данных аналитики и отправки писем.
- Буферизация сообщений для обработки пиковых нагрузок.

– Роль: снижает связанность между микросервисами и повышает отказоустойчивость.

2.2 Взаимодействие компонентов

Компоненты взаимодействуют через:

– RESTful API: для синхронных операций, таких как запрос списка задач или обновление профиля пользователя.

– Kafka: для асинхронных событий. Например, при создании задачи сервис управления задачами отправляет событие в Kafka, которое затем обрабатывается сервисом аналитики для обновления метрик.

– Пример процесса:

– Пользователь создает задачу через интерфейс.

– Сервис управления задачами сохраняет задачу в базе данных и отправляет событие в Kafka.

– Сервис аналитики получает событие и обновляет статистику (например, увеличивает счетчик задач в определенном статусе).

– Сервис пользователей проверяет права доступа при необходимости.

– Обоснование выбора архитектуры

– Микросервисная архитектура была выбрана по следующим причинам:

– Масштабируемость: каждый сервис может масштабироваться независимо. Например, при высокой нагрузке на аналитику можно увеличить ресурсы только для этого сервиса.

– Гибкость: добавление новых функций (например, сервиса уведомлений) не требует переработки всей системы.

– Отказоустойчивость: сбой одного сервиса (например, аналитики) не влияет на работу других компонентов.

Для хранения данных используется PostgreSQL, обеспечивающая надежность и поддержку сложных запросов. Контейнеризация с помощью Docker (контейнерная платформа Docker) упрощает развертывание и обеспечивает единообразие сред разработки и продакшена. CI/CD-процессы настроены с использованием Gtihub Actions, что ускоряет доставку обновлений.

3 Описание методологии разработки

Разработка велась по Agile-методологии с использованием двухнедельных спринтов. Еженедельные созвоны с заказчиком позволяли поддерживать постоянное взаимодействие, получать обратную связь и корректировать направление работы.

3.1 Ключевые аспекты методологии

3.1.1 Двухнедельные спринты

– Каждый спринт начинался с планирования, где команда определяла приоритетные задачи.

– По окончании спринта проводилась демонстрация результатов и ретроспектива для анализа успехов и проблем.

– Такой ритм обеспечивал поэтапную реализацию функционала и регулярные контрольные точки.

3.1.2 Еженедельные встречи с заказчиком

– Созвоны проводились для обсуждения прогресса, уточнения требований и согласования приоритетов.

– Обратная связь позволяла оперативно вносить изменения, минимизируя риск отклонения от ожиданий заказчика.

3.1.3 Еженедельные командные созвоны

– Созвоны проводились для планирования задач, синхронизации между членами команды.

3.2 Преимущества подхода

– Гибкость: Agile позволял адаптироваться к изменениям требований, что особенно важно в проектах с высокой неопределенностью.

– Прозрачность: регулярные демонстрации и встречи обеспечивали видимость прогресса для всех участников.

– Качество: итеративная разработка с постоянным тестированием позволяла выявлять и устранять проблемы на ранних этапах.

Пример из практики:

В одном из спринтов была обнаружена несостыковка представлений заказчика о функционале реализации проектов: в понимании команды проект – это отдельная доска, а в понимании заказчика – пространство с несколькими, никак не связанными досками. В итоге за неделю была переделана реализацию, в том числе схему базы данных, переписали требования и дополнили дизайн.

Нагрузочное Тестирование включало: оценка производительности при высоких нагрузках. Использовалась python библиотека locust

3.3 Информация о планировании деятельности

Планирование велось с учетом навыков членов команды, приоритетов проекта и жёстких временных рамок. Задачи распределялись следующим образом:

– Frontend-разработчик: реализация интерфейса приложения.

– Backend-разработчики: создание микросервисов, базы данных и подключение Kafka. Разработчики сами распределяли задачи между собой.

– Тимлид-аналитик: организация работы команды, общение с заказчиком, общий анализ, создание артефактов, выгрузка приложения на сервер.

– Дизайнер: разработка макетов, обеспечение удобного для пользователя UI/UX интерфейса.

Для управления задачами использовался GitHub Project, что обеспечивало:

– Отслеживание статуса задач в реальном времени.

– Прозрачность процессов для команды и заказчика.

– Удобное планирование спринтов.

– Процесс планирования

– Распределение: задачи назначались с учетом текущей загрузки и навыков разработчиков.

4 Информация о работе участников

4.1 Погирейчик Андрей

В ходе проекта осуществлялось аналитическое сопровождение, управленческая координация команды и настройка процесса выгрузки приложения. На начальном этапе выполнялась разработка и согласование технического задания с представителем заказчика, а также организация работы коллектива. В последующем создавалась структура базы данных, распределялись задачи между участниками и велась отчётность по ключевым параметрам проекта при постоянном взаимодействии с заказчиком. При приближении контрольных точек готовились аналитические материалы и записывались видеопрезентации для демонстрации результатов. В заключительной фазе осуществлялась контейнеризация приложения с помощью Docker, развертывался собственный сервер с подключением домена и настраивался непрерывный деплой через CI/CD.

4.2 Егоров Евгений

В ходе проекта была разработана клиентская часть веб-приложения с опорой на макет, представленный в Figma. Все страницы были сверстаны в соответствии с дизайном и адаптированы под десктоп. Для реализации пользовательской логики использовались хуки и возможности React. Управление состоянием приложения реализовано с помощью Redux, что позволило централизованно обрабатывать данные, поступающие с сервера, и обновлять интерфейс в режиме реального времени. Для взаимодействия с серверной частью используется библиотека axios. в процессе разработки проводилось тестирование эндпоинтов API. Внедрена система маршрутизации, обеспечивающая навигацию между страницами. Реализован drag and drop-функционал с помощью для управления задачами внутри интерфейса. В целях повышения надёжности кода применялся язык TypeScript, обеспечивающий типобезопасность на всех уровнях проекта. При создании интерфейсных компонентов использовался подход atomic design.

4.3 Уварова Ольга

В рамках проекта был разработан дизайн, который отвечает потребностям целевой аудитории. Основное внимание уделялось созданию интуитивно понятного интерфейса, обеспечивающего удобство навигации и взаимодействия пользователей с сайтом. Общая стилистика сайта была сформирована на основе анализа конкурентных решений и предпочтений целевой аудитории. Разработка визуальных элементов, таких как кнопки, иконки и фоновые изображения, проводилась с целью создания единого стиля, который отражает концепцию проекта. Также была разработана логика обработки пользовательских взаимодействий, что позволяет эффективно реагировать на действия пользователей и улучшать их опыт. Вся работа по созданию дизайна и прототипов выполнялась в Figma, что обеспечило гибкость в процессе разработки и возможность быстрого внесения изменений на основе обратной связи от команды и куратора

4.4 Беликов Никита

В рамках проекта выполнялась разработка серверной части приложения. Совместно с тимлидом был выбран технологический стек и архитектурный подход. Для обеспечения удобства взаимодействия с клиентской частью подготовлена спецификация OpenAPI. Базы данных PostgreSQL для каждого микросервиса были созданы методом code-first с использованием Entity Framework и объединены посредством механизма внешних данных postgres_fdw. Реализована контейнеризация баз данных, а также настройка маршрутизации запросов через gateway с использованием YARP. Внедрена система обмена сообщениями на основе Apache Kafka. Добавлена поддержка авторизации и регистрации пользователей с использованием MD5-хеширования и соли. Реализована бизнес-логика микросервисов. Реализовано взаимодействие с базами данных. Разработаны REST-эндпоинты для микросервисов и подготовлена техническая документация по ним.

4.5 Якшибаев Данил

В рамках проекта в роли второго бекенд-разработчика выполнялась разработка серверной части приложения. Основное внимание уделялось проектированию и разработке схем баз данных, а также созданию и поддержке всей логики микросервисов — включая бизнес-логику и логику работы с данными. Для валидации моделей использовался Fluent Validation. Применялась микросервисная архитектура с базами данных PostgreSQL, созданными по методу code-first с помощью Entity Framework. Обеспечивалось взаимодействие с базами данных и реализация REST-эндпоинтов. Также велась работа по настройке маршрутизации запросов через gateway на основе YARP и контейнеризации баз данных.

ЗАКЛЮЧЕНИЕ

Разработанный программный продукт в полном объёме соответствует ключевым требованиям заказчика и конечных пользователей: механизмы создания, редактирования и отслеживания задач реализованы в соответствии с техническим заданием, а инструменты визуализации и аналитики обеспечивают необходимую прозрачность процессов. Архитектура проекта оказалась гибкой и масштабируемой, что позволило в ходе разработки оперативно вносить изменения в рабочие потоки и расширять функционал без серьёзных переделок существующего кода. Взаимодействие компонентов через RESTful API и шину сообщений Kafka подтвердило свою эффективность при нагрузке, соответствуя ожиданиям по производительности и надёжности.

Качество программного продукта в значительной степени определяется результатами проведённого тестирования. Нагрузочные тесты показали высокую стабильность бизнес-логики сервисов управления задачами и пользователей, ещё предстоит разработать модульные тесты, для точечной проверки работы приложения. Анализ причин указал на необходимость оптимизации настроек брокера.

Для дальнейшего развития продукта целесообразно усилить возможности масштабирования за счёт автоматической балансировки нагрузки на микросервисы и внедрения механизмов горизонтального масштабирования брокера сообщений. Расширение системы отчётности путём интеграции с дополнительными источниками данных (например, CI/CD-пайплайнами или сторонними системами мониторинга) позволит получить более детализированную картину эффективности работы команды. Кроме того, внедрение адаптивных уведомлений и более гибкой системы прав доступа создаст основу для применения приложения в более крупных и распределённых проектах, где требования к безопасности и оперативности принятия решений особенно высоки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Оценка эффективности методологии Agile в управлении проектами разработки ПО: исследование успешных и неудачных примеров [Электронный ресурс] // CyberLeninka.ru. — 2024. — URL: https://cyberleninka.ru/article/n/otsenka-effektivnosti-metodologii-agile-v-upravlenii-proektami-razrabotki-po-issledovanie-uspeshnyh-i-neudachnyh-primerov (дата обращения: 20.05.2025).

2. Кузьмин М. А. Перспективы применения инструментов визуализации данных в поддержке принятия управленческих решений / М. А. Кузьмин. — Текст: непосредственный // Молодой учёный. — 2024. — № 28 (527). — С. 53–60.

3. Технология визуализации данных как инструмент совершенствования процесса поддержки принятия решений [Электронный ресурс] / И. И. Фамилия. — URL: https://cyberleninka.ru/article/n/tehnologiya-vizualizatsii-dannyh-kak-instrument-sovershenstvovaniya-protsessa-podderzhki-prinyatiya-resheniy (дата обращения: 20.05.2025).

4. Why a Simple To-Do List Tool Is Winning Over Legions of Fans [Электронный ресурс] // Wired. — 2014. — URL: https://www.wired.com/2014/08/why-a-simple-to-do-list-tool-is-winning-over-legions-of-fans/ (дата обращения: 20.05.2025).

ПРИЛОЖЕНИЕ А

План задач

Таблица 1 – План задач

задача	Ответственный	Длительность
Аналитика		
Расписать ТЗ и согласовать его с заказчиком	Погирейчик Андрей	3 недели
Создать user stories	Погирейчик Андрей	>1 неделя
Создать use case продукта	Погирейчик Андрей	>1 неделя
Спроектировать приложение	Погирейчик Андрей	>1 неделя
Найти домен и сервер	Погирейчик Андрей	
Бэкенд		
Спроектировать взаимодействие между сервисами	Беликов Никита	3 неделя
Создать основу веб приложения	Беликов Никита	2 неделя
Спроектировать бд	Якшибаев Данил	3 недели
Создать бд и настроить соединение	Беликов Никита	>1 неделя
Разработать сервис авторизации и управления аккаунтом	Беликов Никита Якшибаев Данил	2 недели
Разработать сервис управления доской	Беликов Никита Якшибаев Данил	5 недель
Разработать сервис уведомлений	Беликов Никита Якшибаев Данил	2 недели
Разработать сервис аналитики	Беликов Никита Якшибаев Данил	4 неделя
Добавить API для внешних взаимодействий	Беликов Никита Якшибаев Данил	2 недели
Написание тестов	Беликов Никита Якшибаев Данил	3 недели
Настроить docker для деплоя	Беликов Никита Якшибаев Данил	1 неделя
Настроить ci/cd	Погирейчик Андрей	1 неделя

Продолжение таблицы 1

Фронтенд		
Создать главную страницу	Егоров Евгений	2 недели
Создать регистрацию и авторизацию	Егоров Евгений	1 неделя
Создать ЛК	Егоров Евгений	2 недели
Создать главную страницу	Егоров Евгений	3 недели
Создать страницу управления проектами	Егоров Евгений	2 неделя
Создать меню	Егоров Евгений	2 недели
Добавить отображение аналитики	Егоров Евгений	1 неделя
Создать доску задач	Егоров Евгений	1 неделя
Добавить фильты на доску	Егоров Евгений	2 недели
Создать различные виды отображений	Егоров Евгений	2 недели
Создать карточку задачи	Егоров Евгений	>1 неделя
Дизайн		
Проработать стилистику сайта	Уварова Ольга	1 неделя
Макет главной страницы	Уварова Ольга	2 недели
Макет регистрации и авторизации	Уварова Ольга	2 недели
Макет ЛК	Уварова Ольга	2 недели
главной страницы	Уварова Ольга	1 неделя
Макет меню	Уварова Ольга	1 неделя
Макет таблицы задач с разными видами отображения	Уварова Ольга	2 недели
Макет вывода аналитики	Уварова Ольга	1 неделя
Макет карточки задач	Уварова Ольга	1 неделя