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
«Разработка платформы для управления задачами команды разработки»

по дисциплине «Проектный практикум»

Заказчик: Фамилия И.О.	Макаров Артем Васильевич
Куратор: Фамилия И.О. ученая степень, ученое звание, должность	Макаров Артем Васильевич
Студенты команды «QWERTF»	
Фамилия И.О.	Алимбетов Илья Вадимович
Фамилия И.О.	Маяковский Владислав Александрович
Фамилия И.О.	Коляда Егор Денисович
Фамилия И.О.	Щавровская Полина Константиновна
Фамилия И.О.	Утробин Владислав Александрович

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях динамично развивающегося банковского сектора и роста сложности внутренних процессов «Альфа-Банк» возникает необходимость в эффективном управлении задачами. Существующие инструменты зачастую не обеспечивают достаточной гибкости, прозрачности и аналитики, что приводит к снижению продуктивности команд и усложнению контроля за выполнением задач.

Конкретные аспекты проблемы:

1) отсутствие единой системы управления задачами – работа ведется в разрозненных инструментах (почта, мессенджеры, таблицы), что приводит к потере актуальной информации, дублированию задач и снижению прозрачности процессов;

2) недостаточная визуализация рабочих процессов – команды не могут оперативно оценить нагрузку, выявить узкие места или проанализировать эффективность выполнения задач из-за отсутствия наглядных метрик (Cumulative Flow Diagram, Throughput);

3) ручное управление приоритетами и статусами – отсутствие автоматизированного контроля за дедлайнами, приоритетами и распределением задач между сотрудниками увеличивает время на согласование и повышает риск срывов сроков;

4) ограниченные возможности для аналитики – руководство не получает данных для принятия решений (например, среднее время выполнения задач, загруженность сотрудников, динамика завершения проектов).

Решение:

Наш проект предлагает внедрение веб-ориентированной системы управления задачами на основе методологии Канбан, которая обеспечит:

1) автоматизацию создания, назначения и контроля задач с гибкой настройкой workflow;

2) визуализацию процессов через канбан-доски;

3) аналитику производительности (Cumulative Flow Diagram, Throughput) для оптимизации рабочих процессов;

4) командное взаимодействие между пользователями.

Цель:

Создать удобный, безопасный и масштабируемый трекер задач, который повысит эффективность управления проектами в «Альфа-Банк» за счет автоматизации, прозрачности процессов и аналитики.

Задачи:

1) построить er-диаграмму (приложение А);

2) разработать интуитивный интерфейс с поддержкой канбан-досок, drag-and-drop и настраиваемыми колонками;

3) реализовать метрики производительности (Throughput, Cumulative Flow Diagram) для анализа эффективности команд;

4) внедрить гибкую систему фильтрации (по тегам, исполнителям, приоритетам).

Таким образом, реализация проекта позволит «Альфа-Банк» существенно повысить эффективность управления проектами, улучшить контроль за исполнением задач и оптимизировать рабочие процессы подразделений.

Актуальность проекта обусловлена необходимостью внедрения современных инструментов управления в условиях роста сложности банковских операций и увеличения количества параллельно выполняемых проектов. Современные бизнес-процессы требуют четкой координации между отделами, прозрачности выполнения задач и возможности оперативного анализа производительности. Система управления задачами, разрабатываемая в рамках проекта, позволит сократить время на согласование рабочих вопросов и повысить дисциплину исполнения, что будет способствовать ускорению реализации инициатив банка и повышению качества внутренних процессов.

Разрабатываемая система будет применяться для управления всеми типами задач – от повседневных операционных вопросов до стратегических проектов развития. Она станет единой платформой для планирования, контроля и анализа работы сотрудников и подразделений. Система будет интегрирована с корпоративными сервисами банка и займет центральное место в инструментарии управления бизнес-процессами организации.

По завершении проекта ожидается создание полнофункциональной системы управления задачами, которая обеспечит:

1) автоматизированный контроль всего жизненного цикла задач – от создания до завершения;

2) возможность гибкой настройки рабочих процессов под специфику различных подразделений;

3) прозрачность выполнения задач для всех участников процесса;

4) упрощение процессов планирования и формирования отчетности.

Система станет ключевым инструментом повышения операционной эффективности банка, позволит сократить непроизводительные затраты времени сотрудников и обеспечит руководство достоверными данными для принятия управленческих решений.

1 Требования заказчика и пользователей

Для успешной разработки веб-ориентированной системы управления задачами на основе методологии Канбан был проведен детальный анализ требований как со стороны заказчика, так и со стороны конечных пользователей системы. Что позволило четко определить функционал и параметры, которые система должна удовлетворять.

Требования заказчика:

1) автоматизация управления задачами – поддержка канбан-досок с гибкой настройкой колонок и статусов;

2) визуализация рабочих процессов – отображение метрик (Throughput, Cumulative Flow Diagram) для анализа эффективности команд;

3) безопасность данных – шифрование конфиденциальной информации;

4) возможность добавлять задачи нескольких типов;

5) интуитивно понятный интерфейс для управления для управления задачами.

Требования пользователей:

1) простой и интуитивный интерфейс – drag-and-drop перемещение задач, цветовая маркировка приоритетов;

2) удобное создание и отслеживание задач – фильтрация по исполнителям, тегам, срокам;

3) командное взаимодействие – комментирование задач, назначение исполнителей;

4) аналитика в реальном времени – дашборды с метриками (CFD, Throughput);

5) доступ по ссылке – быстрый общий доступ к доскам без сложных настроек.

2 Календарный план

На основе проанализированных требований и пожеланий заказчика был сформирован детальный план разработки системы управления задачами.

Анализ требований и проектирование:

1) проведение серии встреч с заказчиком для уточнения ключевых требований и бизнес-метрик;

2) исследование аналогов и выявление лучших практик для внедрения в систему;

3) разработка ER-диаграммы и проектирование архитектуры БД;

4) создание макетов основных интерфейсов (канбан-доски, фильтрация, отчеты).

Разработка:

1) реализация базового функционала;

2) разработка интерфейсов.

Тестирование:

1) проводить тестирование на каждом этапе разработки (модульное, интеграционное, пользовательское);

2) учет обратной связи от заказчика и доработка системы.

Первый и второй этап плана проекта (приложение В).

3 Анализ

Целевая аудитория:

1) сотрудники банка «Альфа-Банк» (разработчики, аналитики, менеджеры проектов) – основные пользователи системы, которые ежедневно работают с задачами;

2) руководители подразделений – для контроля рабочих процессов и аналитики эффективности команд.

Многие сотрудники сталкиваются с проблемами:

1) отсутствие единого инструмента для управления задачами;

2) сложность отслеживания статусов и приоритетов задач;

3) необходимость ручного сбора аналитики по выполнению задач.

Чем существующие решения не устраивают целевую аудиторию?

1) разрозненность инструментов – задачи хранятся в разных системах (почта, мессенджеры, таблицы);

2) отсутствие визуализации процессов – сложно оценить загрузку команды и статус выполнения задач;

3) ручной сбор метрик – нет автоматизированной аналитики;

4) сложные интерфейсы – существующие системы требуют длительного обучения.

Внедрение системы управления задачами на базе методологии Канбан позволит:

1) упростить процесс постановки и контроля задач;

2) повысить прозрачность рабочих процессов;

3) улучшить взаимодействие между командами;

4) обеспечить руководство аналитикой для принятия решений.

Потребности целевой аудитории:

1) удобное создание и управление задачами (разные типы: баги, эпики, задачи);

2) наглядное отображение статусов через канбан-доски;

3) гибкая настройка workflow под процессы конкретных команд;

4) автоматический расчет метрик (Cumulative Flow Diagram, Throughput);

5) фильтрация задач по исполнителям, тегам, приоритетам;

6) уведомления об изменениях статусов и дедлайнах;

7) комментирование задач для обсуждения деталей;

8) безопасный доступ с разными уровнями прав (RBAC);

9) простой и интуитивный интерфейс с drag-and-drop;

10) цветовая маркировка приоритетов и типов задач;

11) доступ по ссылке для быстрого обмена досками.

Эти функции полностью закрывают потребности сотрудников и руководителей «Альфа-Банк» в эффективном инструменте управления задачами.

3.1 Анализ конкурентов

Характеристики сравнения сайтов конкурентов (приложение С).

Наша система управления задачами на основе методологии Канбан не только соответствует функциональности существующих корпоративных решений, но и преодолевает их ключевые ограничения. В отличие от традиционных систем, где визуализация рабочих процессов часто ограничена стандартными шаблонами, мы предлагаем полностью настраиваемые канбан-доски с интеллектуальной аналитикой производительности, что позволяет командам гибко адаптировать систему под свои уникальные рабочие процессы.

4 Обзор архитектуры программного продукта

Основные компоненты архитектуры

Frontend (Клиентская часть):

1) технологии: React;

2) функциональность клиентской части обеспечивает взаимодействие пользователя с системой, предоставляя интерфейсы для работы с канбан-досками, создания и редактирования задач, визуализации рабочих процессов, фильтрации и поиска задач, просмотра аналитических отчетов и управления настройками проектов. Система позволяет удобно перемещать задачи между статусами, назначать исполнителей, отслеживать прогресс выполнения и получать уведомления об изменениях, обеспечивая полный цикл управления задачами в рамках методологии Канбан.

Backend (Серверная часть):

1) технологии: Python, Django;

2) основные компоненты: REST API для взаимодействия с клиентской частью, сервис аутентификации (JWT, OAuth2), сервис управления задачами (workflow);

3) база данных: PostgreSQL.

Связи между компонентами:

1) frontend ↔ backend: через REST API с JWT-аутентификацией;

2) backend ↔ БД: ORM Django для работы с PostgreSQL;

Обоснование выбора архитектурного решения:

1) эффективность разработки и скорость развертывания: использование React для frontend и Django (Python) для backend позволяет значительно ускорить процесс разработки благодаря готовым решениям, большому количеству библиотек и активному сообществу. React обеспечивает создание отзывчивого и удобного пользовательского интерфейса, а Django предоставляет мощный ORM для работы с базой данных и упрощает разработку REST API;

2) масштабируемость и гибкость: разделение на frontend и backend с взаимодействием через REST API обеспечивает гибкость масштабирования каждого компонента отдельно. При увеличении числа пользователей можно масштабировать backend-серверы, а при увеличении сложности интерфейса – ресурсы frontend;

3) надежность и безопасность: использование JWT или OAuth2 для аутентификации обеспечивает надежную защиту API и пользовательских данных. PostgreSQL – надежная и масштабируемая база данных, гарантирующая сохранность и целостность данных;

4) поддержка методологии Kanban: выбранная архитектура идеально подходит для реализации функциональности канбан-системы. Frontend предоставляет удобные интерфейсы для работы с задачами, а backend обеспечивает логику управления задачами, workflow и нотификациями, поддерживая полный цикл управления задачами в рамках методологии Kanban;

5) простота интеграции: REST API позволяет легко интегрировать систему с другими сервисами и приложениями, например, с системами уведомлений, почтовыми сервисами и другими инструментами для повышения эффективности работы.

5 Прототипирование

Макеты для клиентов (см. рисунок 1 – см. рисунок 15).

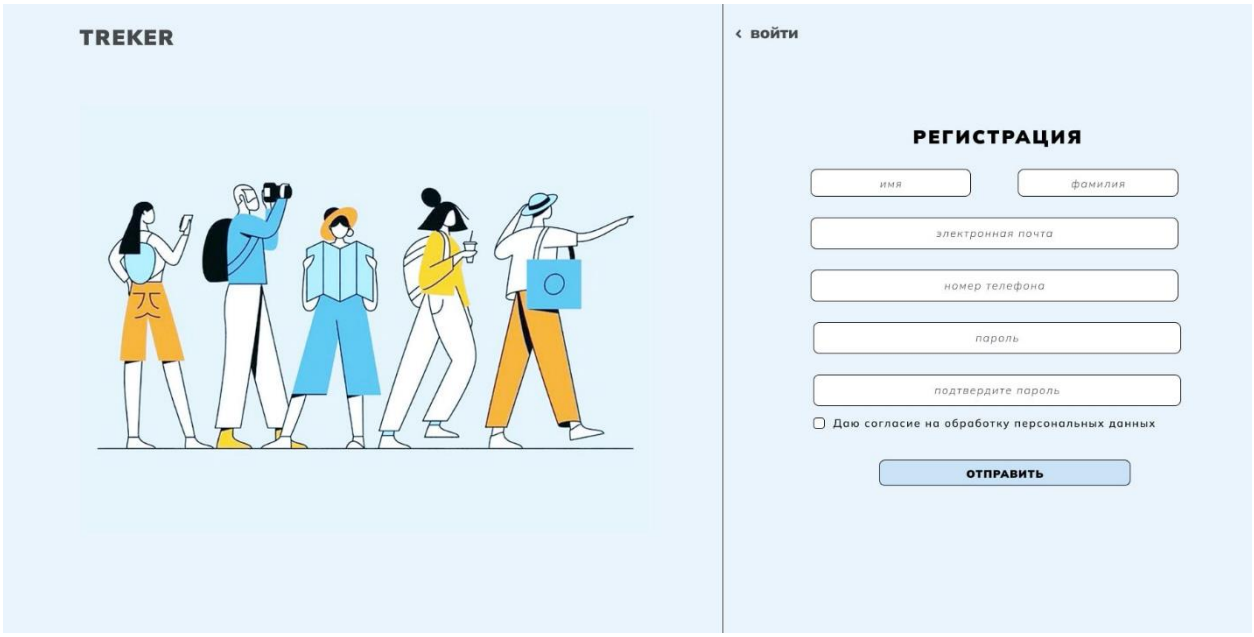


Рисунок 1 – Регистрация

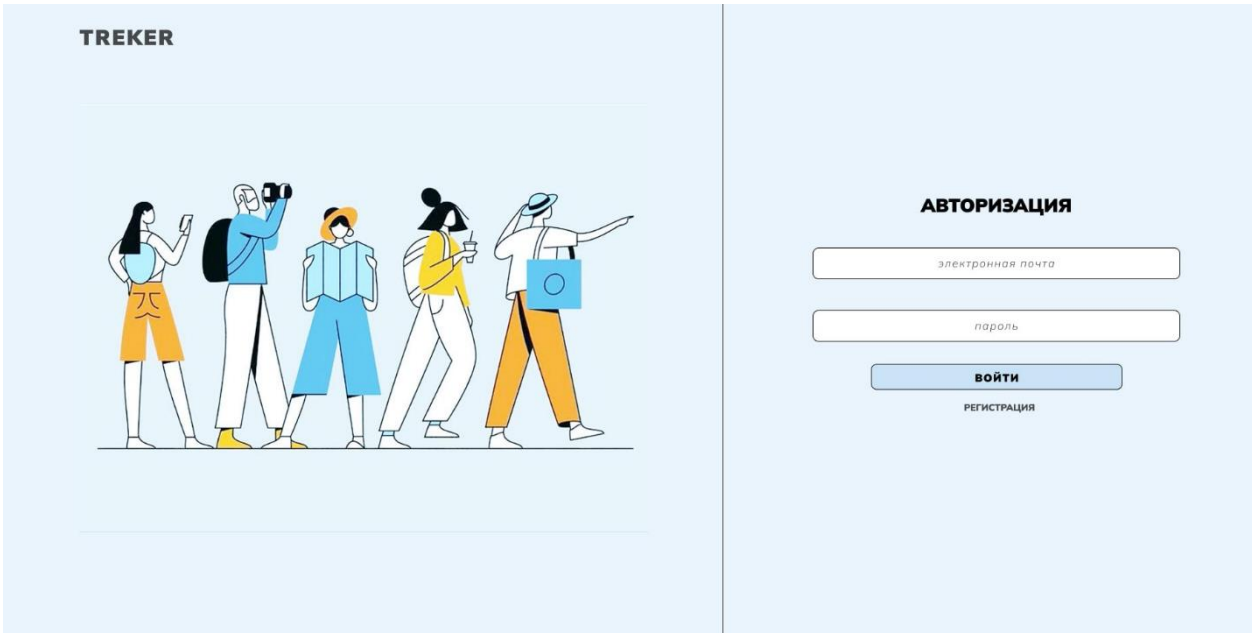


Рисунок 2 – Авторизация

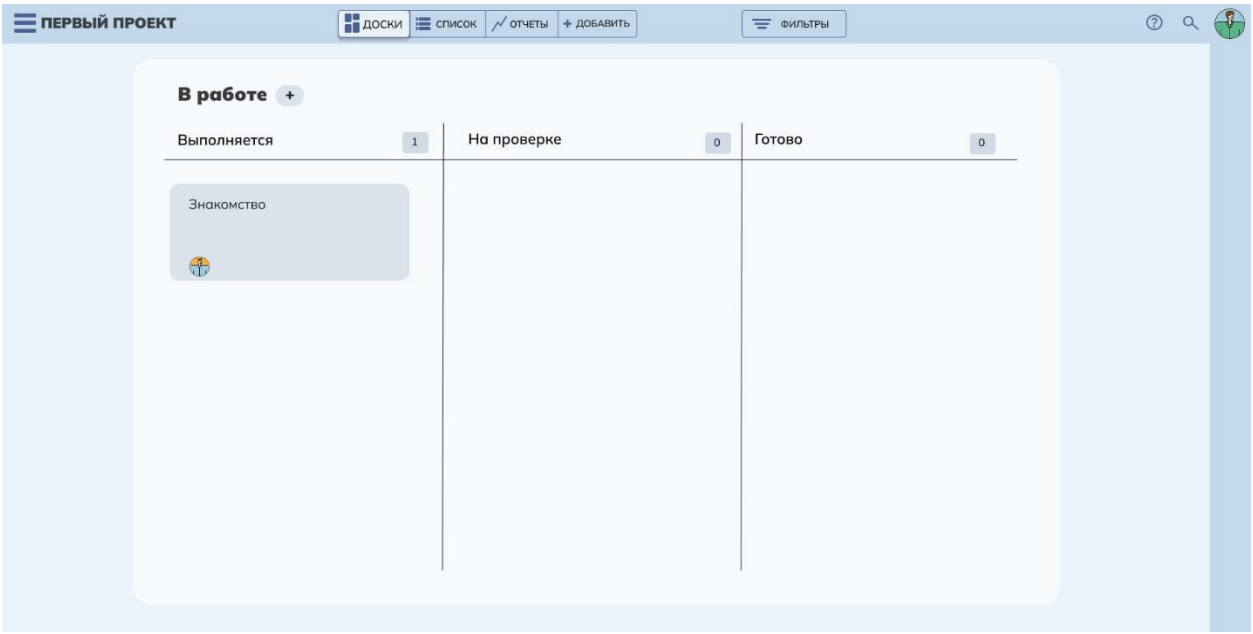


Рисунок 3 – Начальная страница

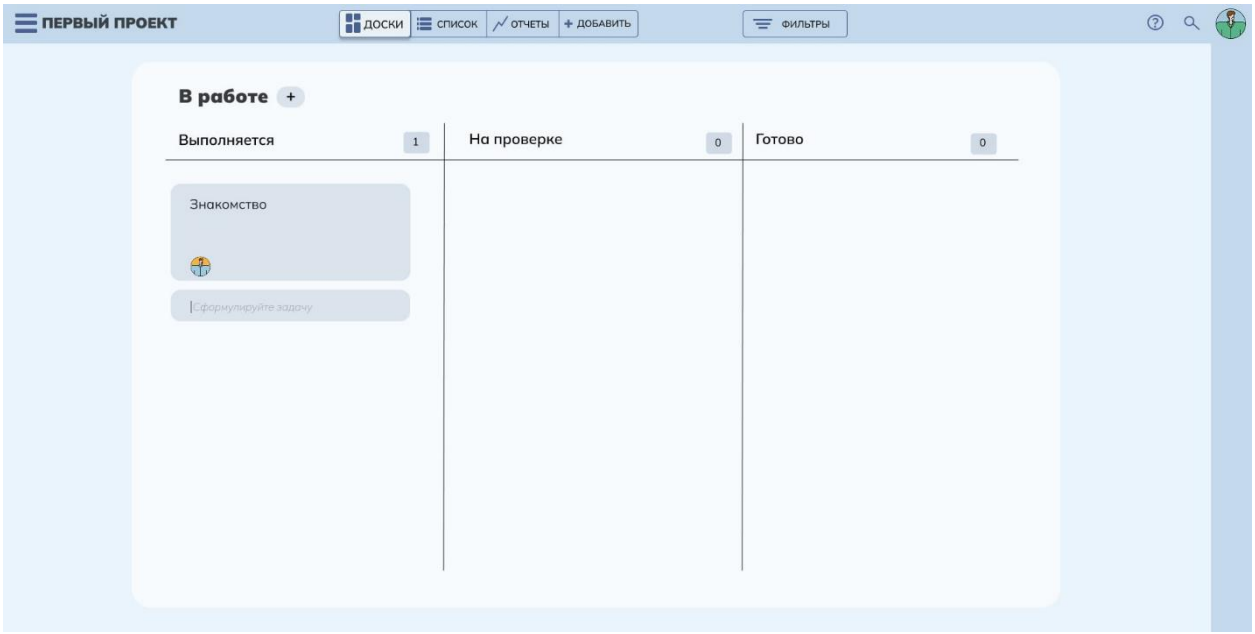


Рисунок 4 – Добавление задачи

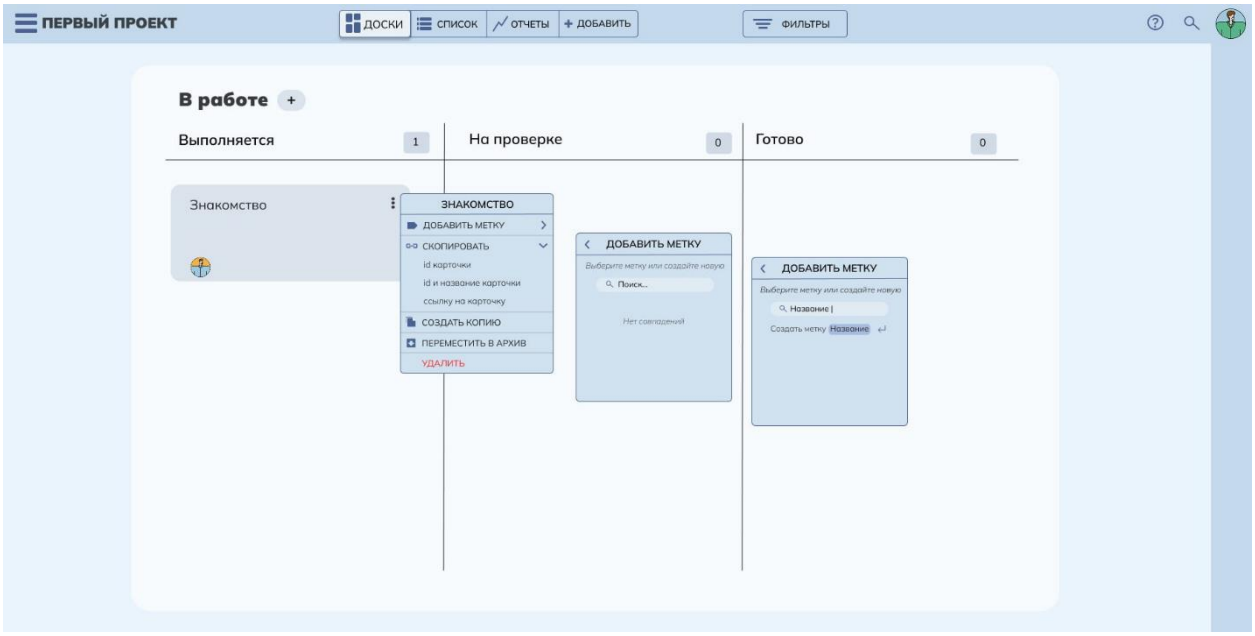


Рисунок 5 – Свойства карточки

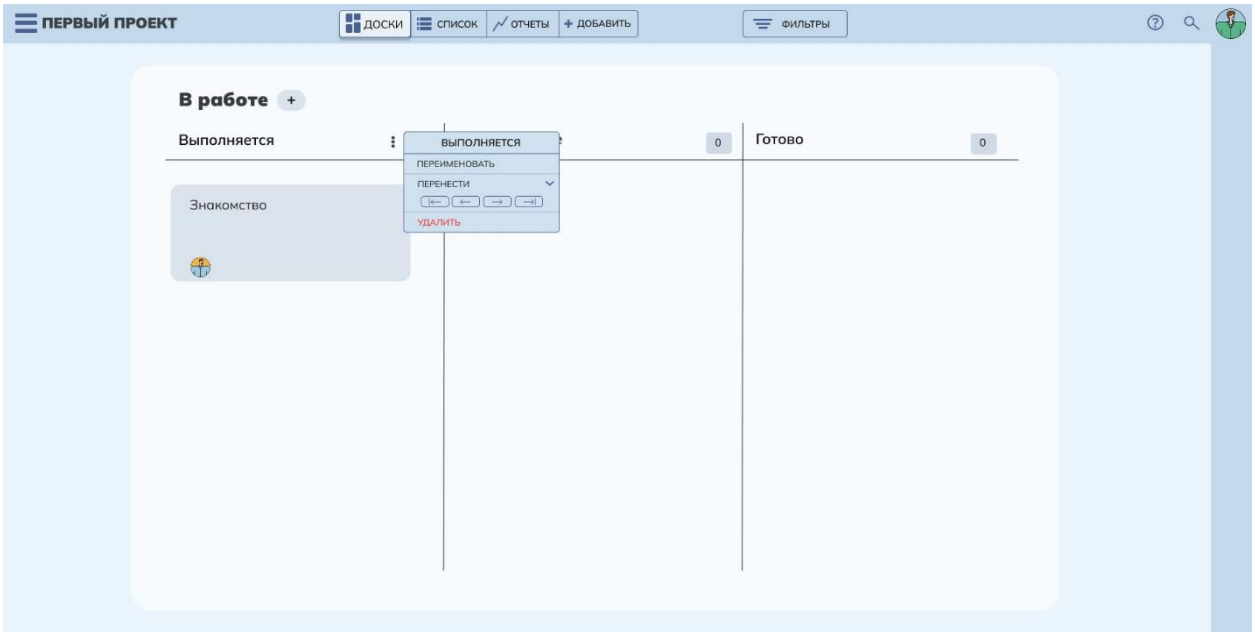


Рисунок 6 – Свойства колонки

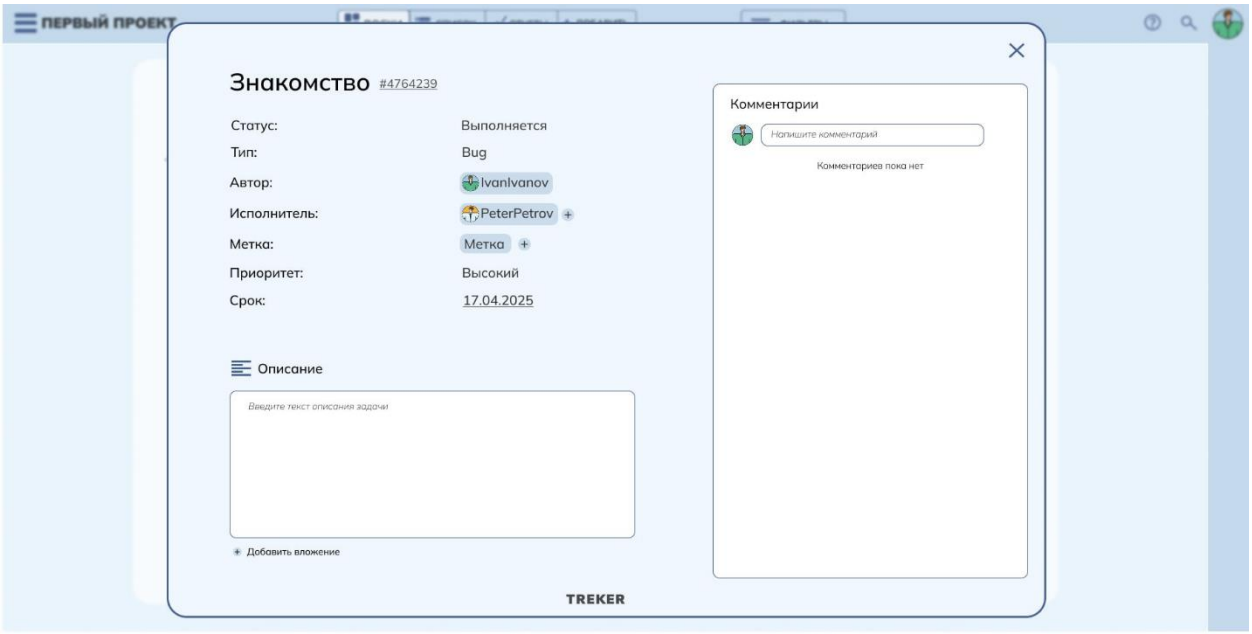


Рисунок 7 – Описание задачи

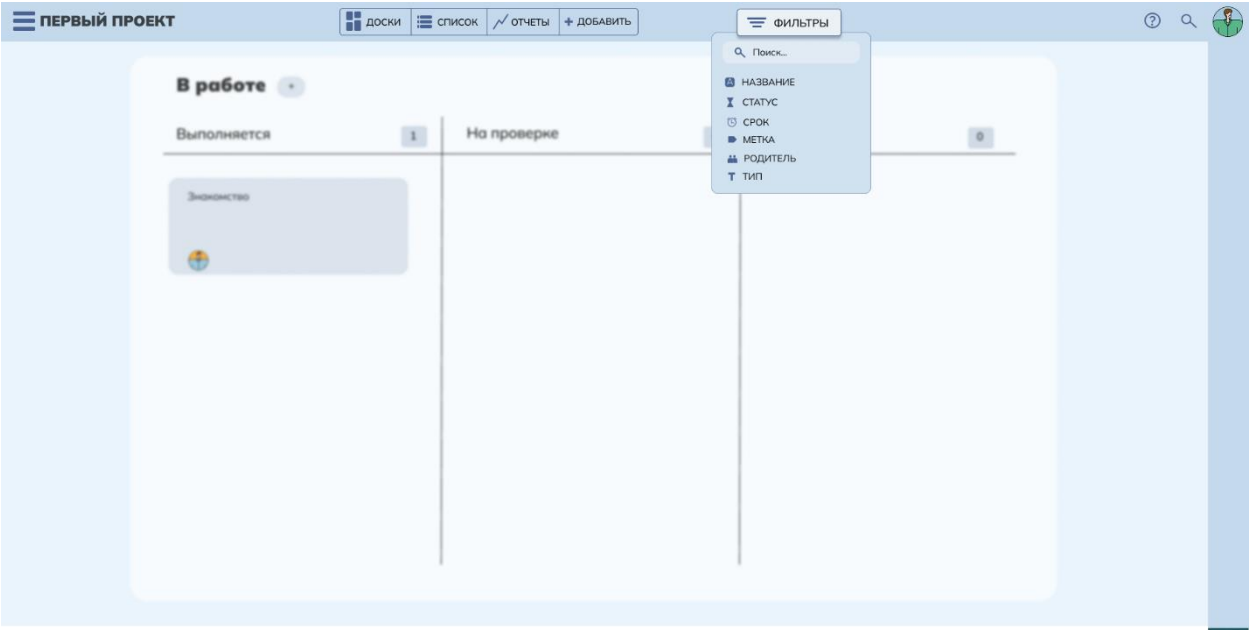


Рисунок 8 – Выбор фильтров

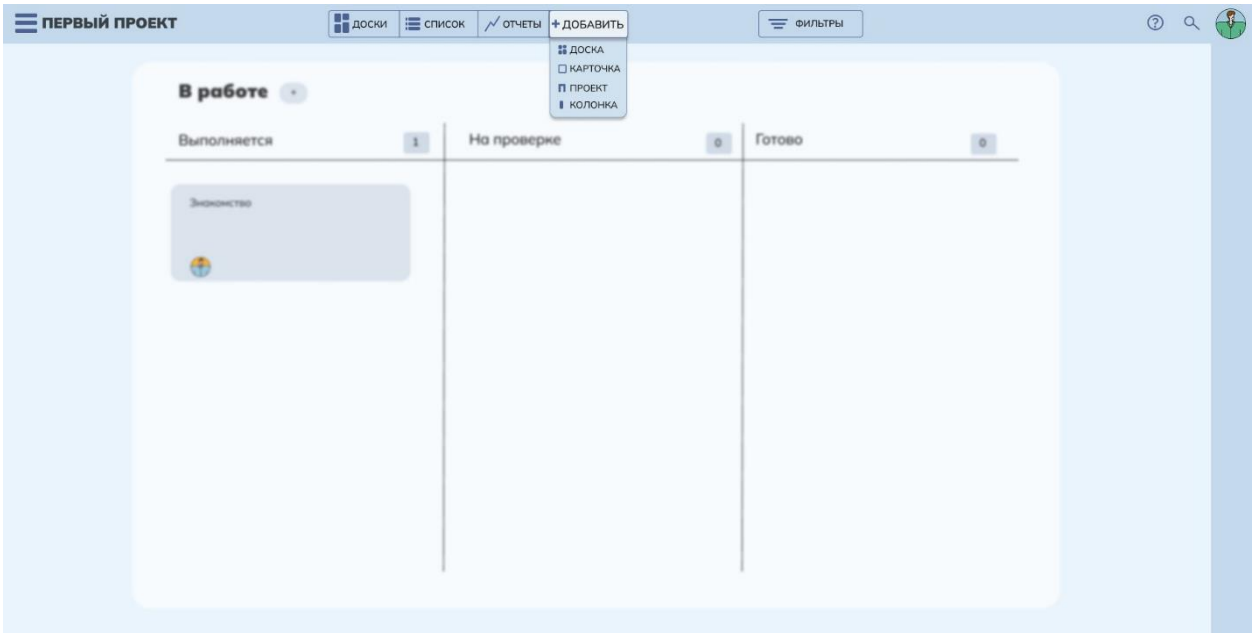


Рисунок 9 – Добавление элементов

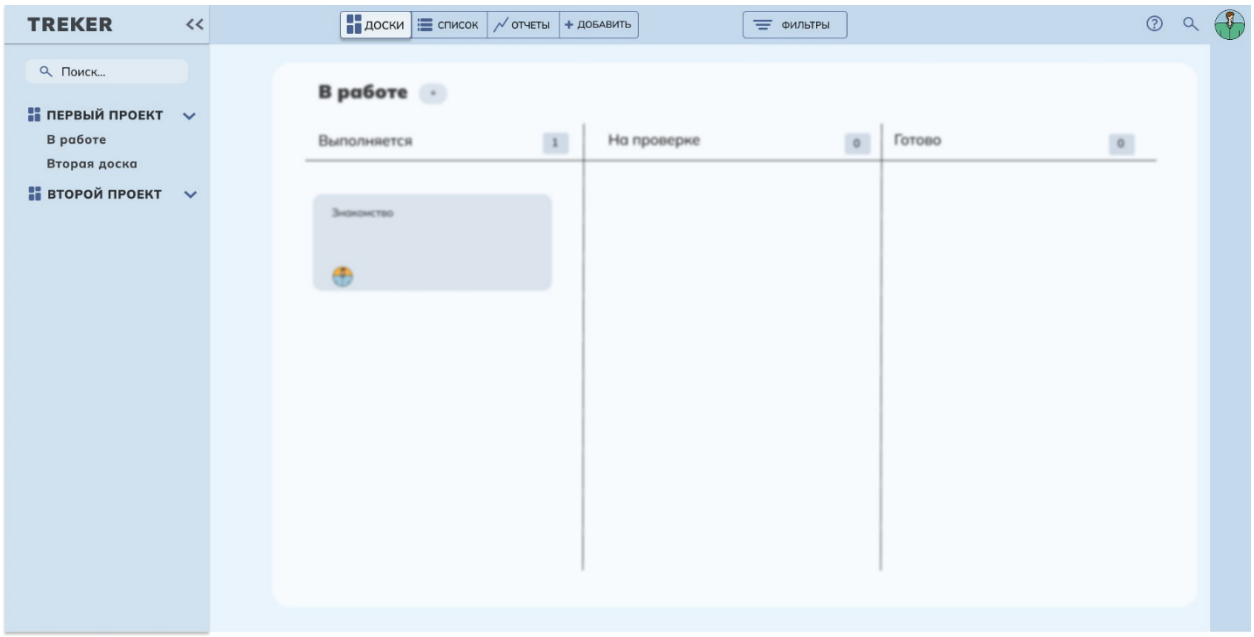


Рисунок 10 – Выбор проекта

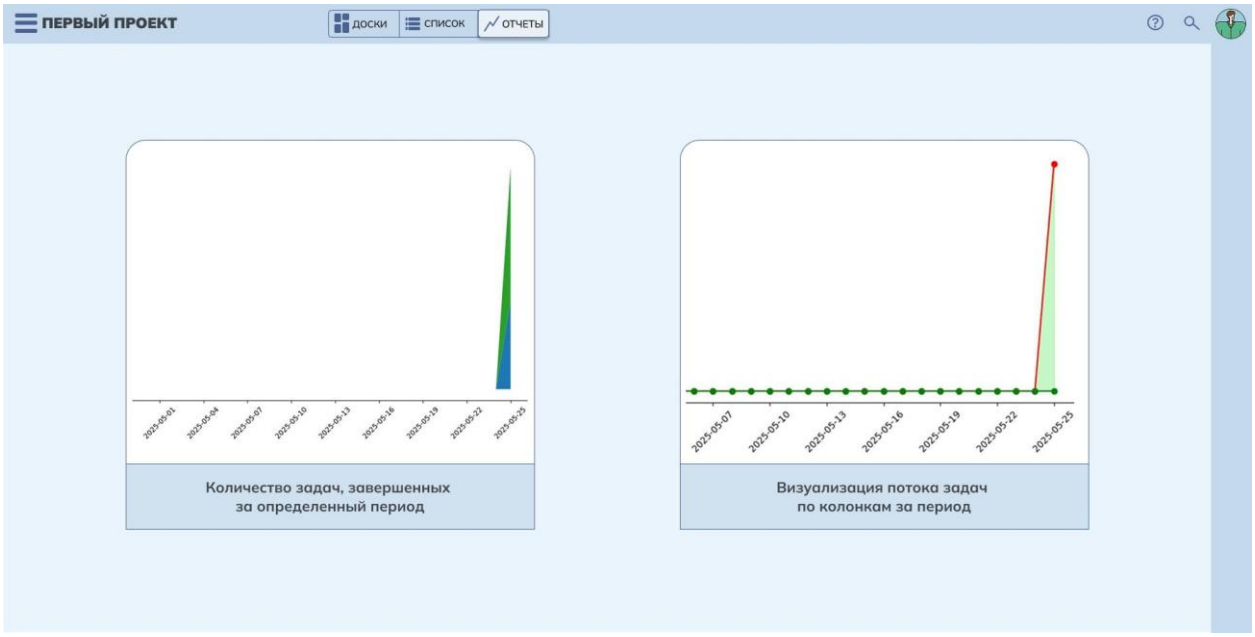


Рисунок 11 – Страница «Отчеты»

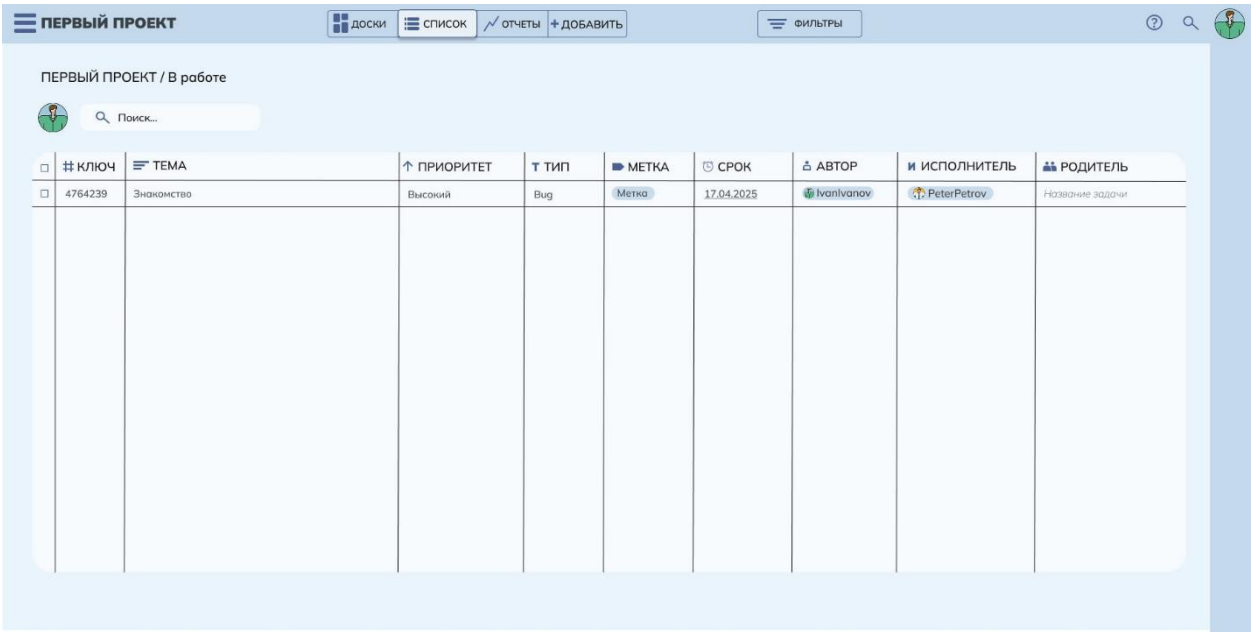


Рисунок 12 – Страница «Список»

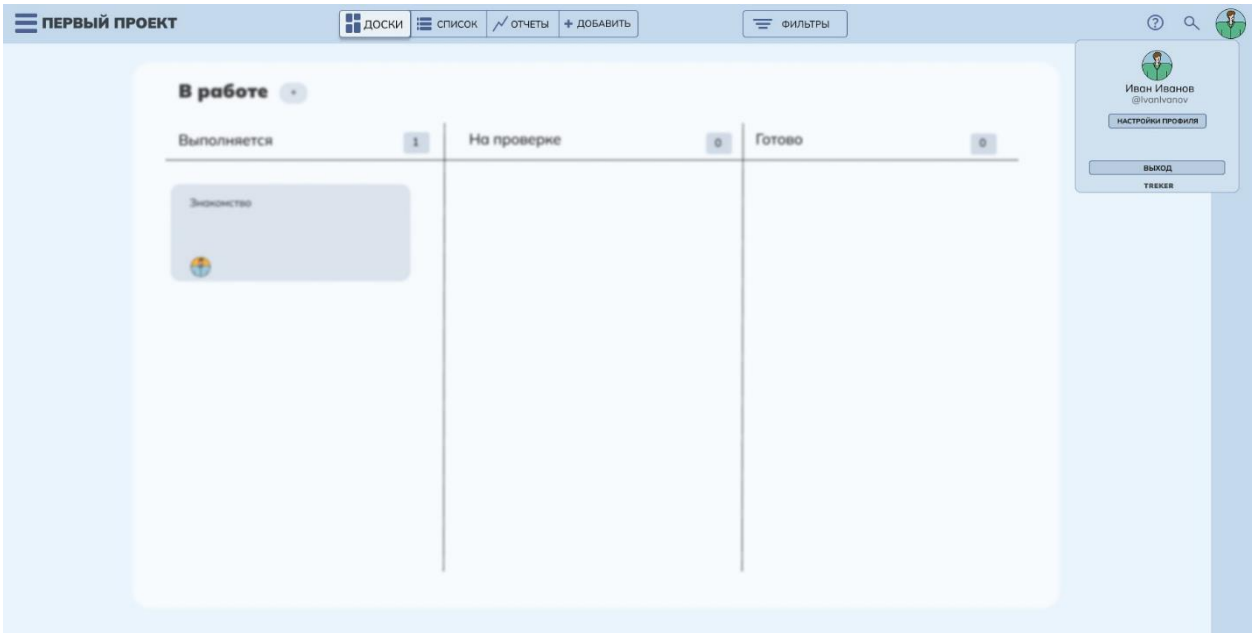


Рисунок 13 – Отображение профиля

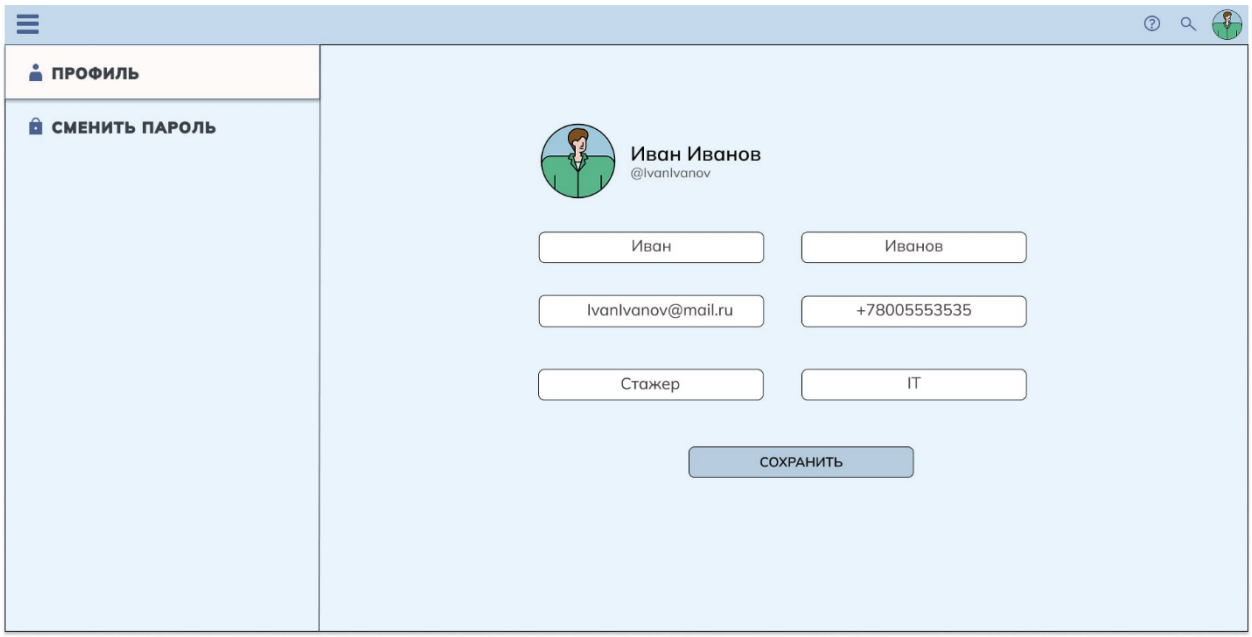


Рисунок 14 – Редактирование профиля

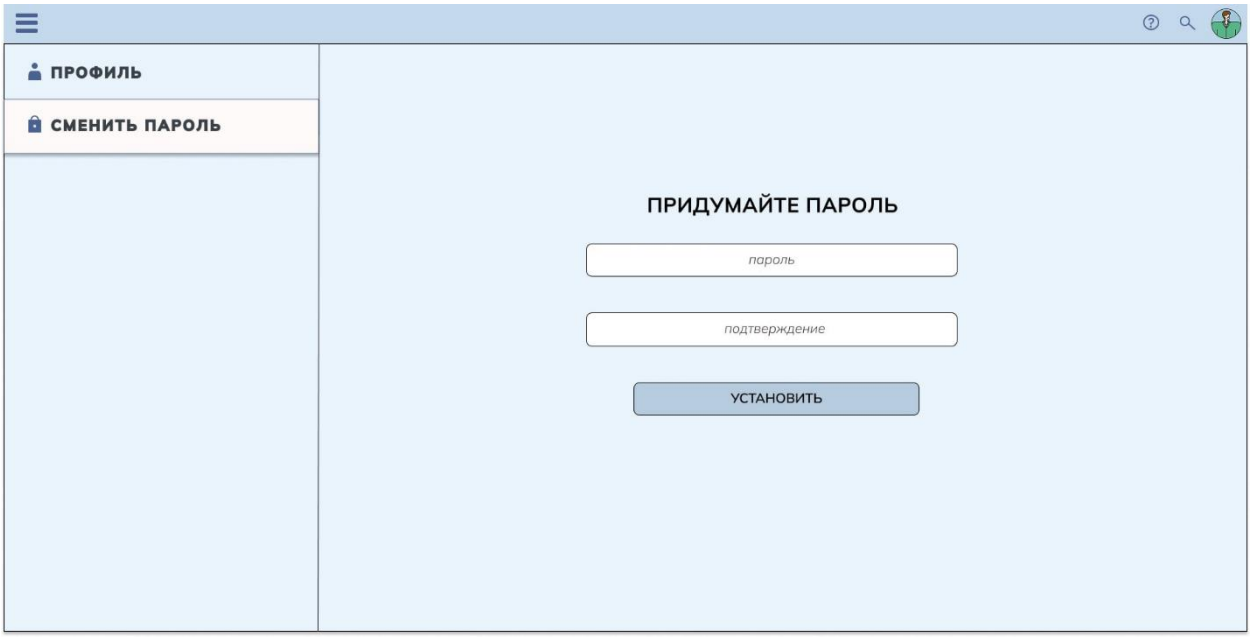


Рисунок 15 – Смена пароля

6 Методология разработки

Для проекта «Разработка платформы для управления задачами команды разработки» выбрана методология Agile, а именно – Kanban, благодаря своей гибкости, прозрачности и фокусу на непрерывном потоке задач без жестких временных рамок спринтов. Этот подход позволяет эффективно адаптироваться к динамично меняющимся требованиям заказчика, оперативно вносить изменения и минимизировать простои за счет визуализации рабочего процесса.

Итеративный процесс:

1) задачи поступают в работу по мере готовности, а команда фокусируется на их непрерывном выполнении. Разработка и тестирование функциональности трекера задач происходят по мере приоритизации, что позволяет быстрее реагировать на изменения и оперативно вносить корректировки без привязки к жестким временным рамкам;

2) по мере завершения ключевых задач будет проводиться демонстрация разработанного функционала заинтересованным лицам (заказчику) для получения обратной связи и внесения корректировок в дальнейшую разработку.

Командная работа:

1) команда проекта будет состоять из:

– разработчиков (frontend и backend);

– тестировщиков;

– бизнес-аналитика;

– владельца продукта (заказчик).

2) ежедневные стендапы (Daily Kanban) будут проводиться для обсуждения прогресса, выявления препятствий и координации действий команды.

Прозрачность и взаимодействие:

1) взаимодействие с заинтересованными сторонами организуется в непрерывном режиме без привязки к фиксированным итерациям. Демонстрации готового функционала проводятся по мере завершения значимых задач, что позволяет оперативно получать обратную связь от владельца продукта;

2) ведение бэклога продукта, содержащего все требования к трекеру задач, с приоритезацией задач владельцем продукта. Использование Kanban-доски (например, Jira, Kaiten, GitHub) для визуализации статуса задач и отслеживания прогресса.

Планирование:

1) сформирован бэклог продукта, включающий в себя основные функциональные требования к трекеру задач, такие как:

– интуитивный интерфейс с канбан-досками и drag-and-drop;

– настраиваемые колонки и карточки задач;

– реализация метрик производительности (Throughput, Cumulative Flow Diagram);

– гибкая система фильтрации и отображение отчетов.

2) провели первоначальное планирование потока работ, выделив наиболее приоритетные задачи.

Разработка:

1) процесс разработки включает в себя:

– проектирование архитектуры системы;

– разработку frontend с использованием React;

– разработку backend с использованием Python и Django;

– интеграцию модулей и разработку API.

2) использование системы контроля версий Git для управления кодом, обеспечения совместной работы разработчиков и отслеживания изменений.

Разбор выявленных ошибок:

1) Трудности с внедрением новых технологий (JWT, Djoser, Swagger). В процессе разработки системы управления задачами возникли значительные сложности при работе с незнакомыми технологиями, которые пришлось осваивать параллельно с реализацией функционала. Основные проблемы сконцентрировались вокруг трёх ключевых компонентов: системы аутентификации на JWT-токенах, библиотеки Djoser для работы с пользователями и инструмента Swagger для документирования API. Эти проблемы продемонстрировали, что использование незнакомых технологий без предварительного глубокого изучения приводит к значительным временным затратам. Решение приходило только после тщательного анализа документации, изучения исходного кода библиотек и многочисленных экспериментов;

2) Проблема несогласованности в командной разработке. В ходе реализации проекта мы столкнулись с серьезными сложностями, вызванными различиями в подходах к разработке у членов команды. Основные проблемы проявились в трех ключевых аспектах: стиль написания кода, работа с миграциями базы данных и настройка окружения. Проблема обнаружилась, когда код одного разработчика перестал работать у других членов команды. Особенно критичной оказалась ситуация с миграциями - при слиянии веток система перестала применять изменения к БД. Для решения проблем несогласованности мы внедрили единые стандарты кодирования. Были установлены строгие правила работы с миграциями: запрет на самостоятельное создание нумерованных миграций, обязательная синхронизация перед созданием новых веток. Эти меры позволили значительно снизить количество конфликтов и проблем с совместимостью в процессе разработки;

3) Проблемы интеграции React-фронтенда с Django-бэкендом. Основные сложности возникли при разделении API и шаблонов — React приложение, изначально разрабатываемое как независимый проект, конфликтовало с Django шаблонами при передаче контекста и маршрутизации. Настройка CORS потребовала тщательной отладки: несмотря на подключение django-cors-headers, запросы блокировались из-за неправильного порядка middleware и различий в конфигурациях сред разработки. Особые проблемы доставила JWT-аутентификация — токены не сохранялись, refresh-механизм работал нестабильно. Эти проблемы удалось решить только после комплексного пересмотра архитектуры интеграции;

4) Сложности проектирования ER-диаграммы и нормализации БД. При проектировании структуры базы данных мы столкнулись с рядом концептуальных сложностей. Основные проблемы возникли при попытке отобразить реальные бизнес-процессы в реляционную модель. Первоначальная схема содержала несколько фундаментальных недостатков: нарушение принципов атомарности данных, избыточное дублирование информации и неочевидные взаимосвязи между сущностями. Особую сложность представляла нормализация данных — изначальная структура содержала повторяющиеся значения и составные атрибуты, что приводило к аномалиям при операциях обновления. Бизнес-правила требовали сложных ограничений на отношения между сущностями, которые было невозможно реализовать в первоначальной схеме. Эти проблемы проявились особенно остро при выполнении составных запросов и параллельном доступе к данным. Решение потребовало нескольких итераций перепроектирования, введения дополнительных таблиц-связок.

7 Распределение задач между разработчиками

Frontend-разработка:

Перед началом разработки наша команда провела анализ возможных архитектурных подходов для Frontend-приложения. Были рассмотрены следующие варианты:

1) классическая архитектура;

2) модульная архитектура;

3) atomic Design;

4) FSD (Feature-Sliced Design).

Классическая архитектура, предполагающая простое разделение на переиспользуемые компоненты и страницы, была отвергнута из-за потенциальных сложностей с масштабированием проекта. Архитектура FSD показалась нам избыточно сложной для текущих задач. Atomic Design, хотя и выглядел привлекательно, требовал больше времени на освоение командой.

Мы остановились на модульной архитектуре, которая обеспечивала хороший баланс между структурой и простотой освоения. Наша реализация включает 4 слоя:

1) UI-слой: базовые компоненты интерфейса (кнопки, карточки задач) без привязки к бизнес-логике;

2) Components: составные компоненты, такие как фильтры задач (по тегам, исполнителям, приоритетам), которые включают простую бизнес-логику;

3) Modules:

– модуль канбан-доски с реализацией drag-and-drop;

– модуль аналитики с графиками CFD и Throughput;

– модуль управления участниками проекта

Каждый модуль содержит свой API для взаимодействия с backend и изолированную логику.

4) Pages:

– страница проекта (комбинация модулей доски, участников и аналитики);

– страница личного кабинета;

– страница бокового меню;

– страницы «Отчеты» и «Список».

После утверждения дизайн-макетов мы начали с разработки UI-кита, используя Ant Design как базовую библиотеку, дополнительно создавая кастомные компоненты для специфичных элементов канбан-доски.

На этапе, когда backend еще не был готов, для разработки и тестирования мы использовали mock-сервер, что позволило параллельно вести работу над:

– страницей проекта с канбан-доской;

– модальными окнами создания/редактирования задач;

– системой фильтрации задач (по тегам, исполнителям, приоритетам).

Особое внимание было уделено реализации drag-and-drop функционала для перемещения задач между колонками. Мы столкнулись с проблемой - при перемещении задачи в другую колонку статус не всегда обновлялся. Оказалось, что проблема была в неправильной последовательности вызовов API - сначала происходило визуальное перемещение, затем запрос к серверу. Мы изменили логику: сначала отправка запроса, и только после успешного ответа - визуальное обновление.

При разработке системы комментариев к задачам мы добавили возможность прикрепления файлов.

После готовности backend API мы провели интеграцию, заменив mock-данные на реальные запросы к серверу. Для удобства работы с API создали единый API-клиент с обработкой ошибок и повторными запросами при неудаче.

В процессе нагрузочного тестирования мы выявили и оптимизировали «узкие» места в рендеринге канбан-доски при большом количестве задач.

Решением стало виртуализированное отображение задач и мемоизация компонентов.

Backend-разработка:

Разработка backend-части системы управления задачами для «Альфа-Банк» началась с проектирования микросервисной архитектуры. Основной упор был сделан на создание надежного и масштабируемого решения с использованием Django и PostgreSQL. Первым был реализован сервис задач, включающий полный CRUD-функционал с поддержкой фильтрации по тегам, исполнителям и приоритетам. Особое внимание уделили реализации канбан-досок с drag-and-drop интерфейсом.

Для обеспечения безопасности внедрили JWT-аутентификацию с access/refresh токенами и трехуровневой системой RBAC (администратор, участник, наблюдатель). Access-токены генерируются с использованием HMAC SHA256 и имеют срок жизни 2 часа, refresh-токены хранятся в httpOnly cookies и действительны 30 дней. Интеграция с корпоративной Active Directory позволила упростить процесс онбординга сотрудников банка.

Для фронтенд-разработчиков подготовили подробную документацию в Swagger и Postman-коллекцию с примерами всех API-запросов. Дополнительно реализовали Webhook-API для интеграции с внешними системами, позволяющее подписываться на события в реальном времени.

ЗАКЛЮЧЕНИЕ

Разработанная платформа для управления задачами команды разработки в значительной степени соответствует сформулированным требованиям заказчика и его пользователей, направленным на оптимизацию рабочих процессов и повышение эффективности управления проектами. Итоговый программный продукт обеспечивает автоматизацию процессов, визуализацию рабочих потоков, а также углубленную аналитику, что отвечает современным потребностям управления в динамичной среде банковской деятельности.

Оценка качества, основанная на результатах тестирования, подтверждает стабильность и работоспособность системы. Реализована поддержка канбан-досок, функций drag-and-drop, гибкой настройки и других заявленных функциональностей.

Для дальнейшего развития продукта предлагается рассмотреть следующие направления:

1) улучшение интеграции: расширение и упрощение интеграции с другими корпоративными системами для обеспечения плавного и автоматизированного обмена данными;

2) продвинутая аналитика: добавление функций машинного обучения для автоматической приоритезации задач, прогнозирования сроков выполнения и выявления проблемных зон в рабочих процессах;

3) мобильные приложения: разработка мобильных приложений для iOS и Android позволит пользователям эффективно управлять задачами в любом месте и в любое время;

4) автоматизация рутинных задач: внедрение средств автоматизации для упрощения рутинных операций, таких как создание задач, назначение исполнителей и обновление статусов.

Таким образом, разработанная платформа обладает большим потенциалом для значительного повышения эффективности управления проектами. Успешное внедрение продукта в существующую инфраструктуру

и дальнейшее развитие функциональности обеспечат банку конкурентное преимущество и улучшат качество внутренних бизнес-процессов. Наша команда уверена, что поставленные цели достигнуты, а предложенные решения соответствуют требованиям заказчика.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Разработка серверной части веб-ресурса / В. В. Никулин, А. А. Олейников, А. А. Сорокин, А. В. Олейникова. — Санкт-Петербург : Лань, 2023. — 132 с. — ISBN 978-5-507-47868-2. — Текст : электронный // Лань: электронно-библиотечная система. — URL : https://e.lanbook.com/book/356102

2. Марков, В. С. Современный JavaScript. Подходы и фреймворки / В. С. Марков. – Москва : Альпина Паблишер, 2019. – 450 с.

3. Петров, К. В. JavaScript. Следующий шаг / К. В. Петров. – Москва: Книга по требованию, 2021. – 320 с.

4. Головатый, А. Django. Подробное руководство (2-е издание) / А. Головатый, Дж. Каплан-Мосс. – Москва, 2010. – 552 с.

5. Форсье, Д. Django. Разработка WEB-приложений на Python / Д. Форсье, П. Биссекс, У. Чан. – Москва, 2009. – 453 с.

6. Дронов, В. Django : практика создания Web-сайтов на Python / В. Дронов. – Санкт-Петергбург : «БВХ-Петербург», 2021. – 40 с.

7. Латкин, И. Я. TypeScript в действии : учебное пособие / И. Я. Латкин. – Новосибирск : НГТУ, 2022. – 280 с.

8. Овсянников, А. И. TypeScript: Углубленное руководство / А. И. Овсянников. – Екатеринбург : Урал. гос. ун-т, 2021. – 400 с.

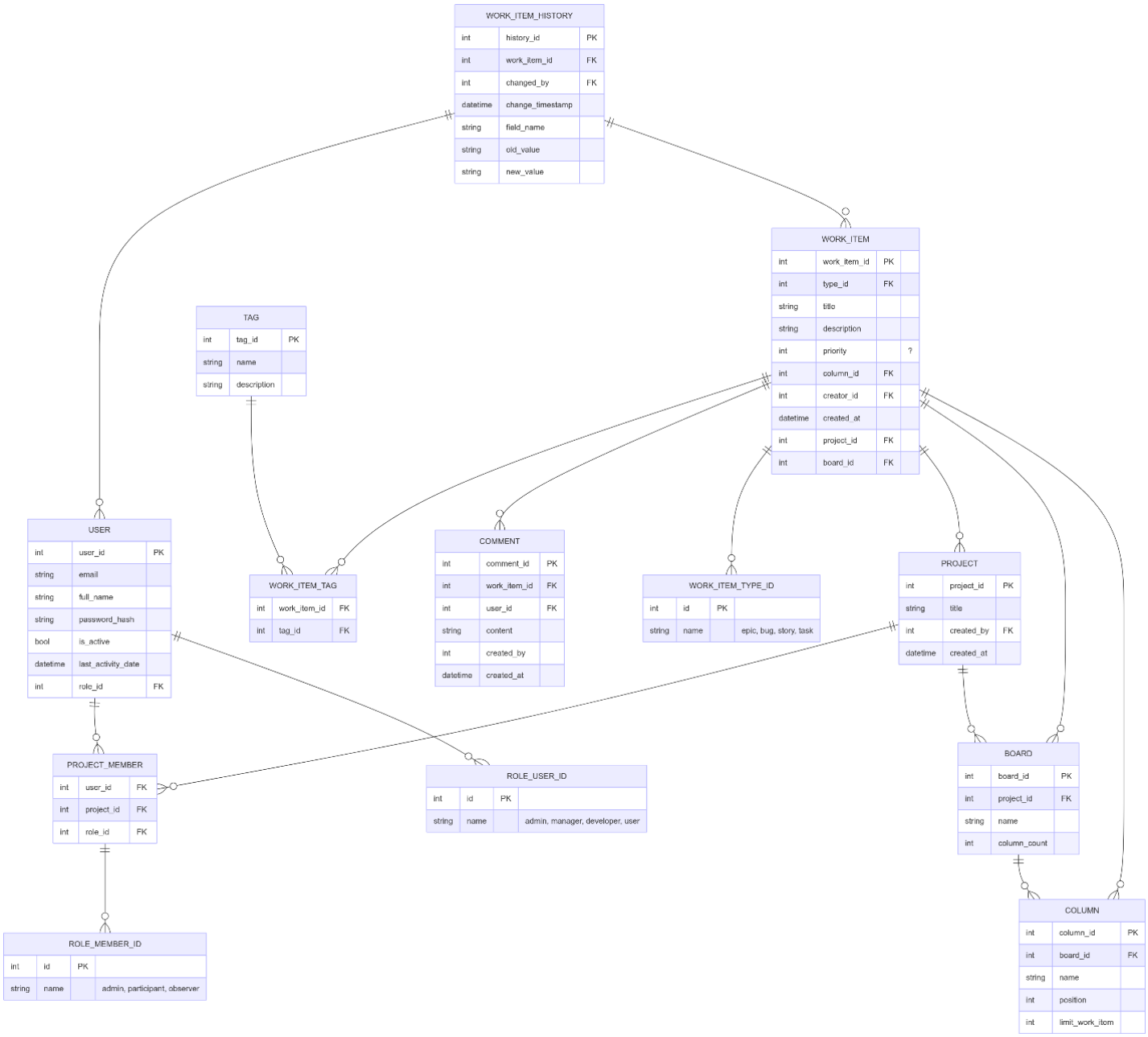
9. Кириченко, А. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков / А. Кириченко. – Санкт-Петербург : Наука и техника, 2021. – 432 с.

10. Кириченко, А. Справочник HTML. Кратко, быстро, под рукой / А. Кириченко. – Санкт-Петербург : Наука и техника, 2021. – 288 с.

ПРИЛОЖЕНИЕ А

(обязательное)

ER-диаграмма



ПРИЛОЖЕНИЕ В

(обязательное)

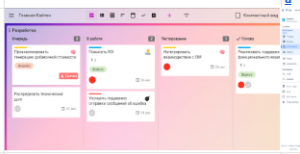
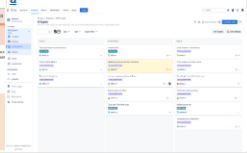
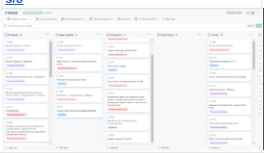
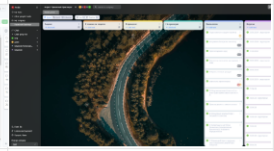
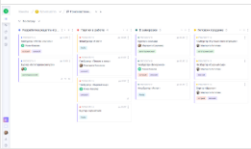
Календарный план

№	Работы	Ответственный	Длительность	Дата начала
Анализ (01.04.2025 - 07.04.2025)				
1.1	Идея для проекта	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	1 неделя	17.04.2025
	Встреча с заказчиком	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	30-50 минут	20.03.2025
1.2	Анализ аналог и выявление проблем	Алимбетов Илья В.	1 неделя	21.03.2025
1.3	Создание макета первой страницы	Щавровская Полина К.	1 неделя	21.03.2025
1.4	Создание er-диаграммы	Утробин Владислав А.	1 неделя	24.03.2025
	Встреча с заказчиком	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	30-50 минут	27.03.2025
1.5	Составить бизнес-требования	Алимбетов Илья В.	1 неделя	28.03.2025
1.6	Выбор опитмальных цветов и дизайн страницы "первое пространство"	Щавровская Полина К.	1 неделя	28.03.2025
1.7	Выбор основных метрик и ознакомиться с ними	Утробин Владислав А., Маяковский Владислав А.	1 неделя	31.03.2025
	Встреча с заказчиком	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	30-50 минут	03.04.2025
1.8	Подготовка к КТ1 (презентация, проверить открыты ли артефакты, запись видео)	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	3 дня	04.04.2025
Разработка (08.04.2025 - 28.04.2025)				
2.1	Сохранение пользователя при перезагрузке страницы через JWT-токен	Утробин Владислав А.	1 неделя	08.05.2025
	Встреча с заказчиком	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	30-50 минут	10.04.2025
2.2	Создание простейшей авторизации	Маяковский Владислав А.	1 неделя	11.04.2025
2.3	Составить ТЗ	Алимбетов Илья В.	1 неделя	11.04.2025
2.4	Верстка страниц авторизация и регистрация и их функционал	Коляда Егор Д.	1 неделя	12.04.2025
2.5	Дизайн раздела "Профиль"	Щавровская Полина К.	3 дня	13.04.2025
2.6	Настройка БД	Утробин Владислав А.	1 неделя	15.04.2025
2.7	Дизайн функционала кнопки "Фильтрация"	Щавровская Полина К.	6 дней	17.04.2025
	Встреча с заказчиком	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	30-50 минут	17.04.2025
2.8	Создание карточек проекта на сайте	Маяковский Владислав А.	1 неделя	19.04.2025
2.9	Реализовать валидацию формы в login	Коляда Егор Д.	1 неделя	21.04.2025
2.10	Доработка аналогов, в соответствии с комментариями	Алимбетов Илья В.	1 неделя	21.04.2025
2.11	Интеграция с API	Утробин Владислав А.	1 неделя	22.04.2025
2.12	Дизайн страницы "Добавить"	Щавровская Полина К.	2 дня	23.04.2025
2.13	Дизайн "Настройки профиля"	Щавровская Полина К.	3 дня	25.04.2025
	Встреча с заказчиком	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	30-50 минут	25.04.2025
2.14	Верстка админской панели	Маяковский Владислав А.	1 неделя	26.04.2025
2.15	Подготовка к КТ2 (презентация, проверить артефакты, подготовить ссылки)	Алимбетов Илья В., Щавровская Полина К., Коляда Егор Д., Утробин Владислав А., Маяковский Владислав А.	4 дня	26.04.2025

ПРИЛОЖЕНИЕ С

(обязательное)

Анализ аналогов

Характеристики сравнения	Сайты конкурентов				
Название	**Kaiten**	**Jira**	**GitHub**	**YouGile**	**Shtab**
Ссылка	https://kaiten.ru/	https://www.atlassian.com/software/jira	https://github.com/users/shchavr/projects/8	https://ru.yougile.com/	https://shtab.app/
Скриншот					
Краткое описание	Kaiten - это гибкая система управления задачами, построенная по принципам канбан. Подходит для команд, которые ценят простоту и наглядность работы.	Jira - мощный инструмент для управления проектами, особенно популярный в IT и среди крупных команд.	GitHub Projects - инструмент для управления задачами, интегрированный в экосистему GitHub. Подходит для разработчиков, работающих с Git.	YouGile - система управления проектами и коммуникации, предназначенная для эффективной работы больших команд.	Shtab - сервис для управления проектами с трекером времени.
Адаптив сайта	Есть	Есть	Есть, но мобильная версия упрощена	Есть	Есть, но не удобен для использования
WIP-лимиты	Предлагает наиболее гибкую систему WIP-лимитов среди рассматриваемых решений. Лимиты можно настраивать не только для отдельных колонок, но и для конкретных типов задач или исполнителей, что особенно ценно для команд со сложной структурой работ. Система автоматически подсвечивает превышение лимитов и может блокировать добавление новых задач в перегруженные статусы.	Требует для реализации WIP-лимитов установки дополнительных плагинов (например, Kanban for Jira).	Не имеет встроенной поддержки WIP-лимитов.	Предлагает продвинутую систему WIP-лимитов, превосходящую по гибкости многие аналоги. В отличие от Kaiten, где лимиты устанавливаются преимущественно на колонки, в YouGile можно настраивать ограничения с учетом множества параметров: по типам задач (например, не более 3 технических задач одновременно), по исполнителям (ограничение личной нагрузки), по приоритетам.	Поддерживает базовые WIP-лимиты на уровне колонок. Система лишь визуально выделяет перегруженные статусы, но не блокирует добавление новых задач.
Метрики и аналитика	Предоставляет базовый, но достаточный для большинства команд набор метрик: Cycle Time, Lead Time и Throughput. Визуализация выполнена в виде простых и понятных графиков, позволяющих быстро оценить эффективность работы.	Безусловный лидер в этой категории. Глубокая аналитика включает не только стандартные отчеты (Cumulative Flow, Velocity Chart), но и возможность создания сложных кастомных дашбордов.	Предлагает аналитику, ориентированную именно на разработчиков – частоту коммитов, связь между issues и pull requests, время code review. Для IT-команд этих данных обычно достаточно, но для общего управления проектами функционал слишком узкоспециализирован.	Система предоставляет не только стандартные метрики (Cycle Time, Lead Time), но и уникальные показатели вроде "коэффициента эффективности процессов", который анализирует соотношение времени продуктивной работы и простоев.	Предоставляет минимальную аналитику: время выполнения задач и общую загруженность команды.