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.
Куратор: Макаров А. В
ученая степень, ученое звание, должность

Студенты команды:
Евстигнеева Е.С
Долгих А.С
Горбунов И.С
Михайленко К.А
Шутов К.В

Екатеринбург, 2025

Оглавление

ВВЕДЕНИЕ

Цель и задачи проекта:

Цель проекта — разработать веб-приложение, предназначенное для управления задачами команды разработки, которое позволит улучшить координацию, повысить прозрачность процессов и оптимизировать рабочие процессы с помощью визуальных инструментов и автоматизации.

Задачи:

1) Разработать удобную Kanban-доску с возможностью отслеживания статуса задач;
2) Реализовать сбор и отображение метрик (диаграмма суммарного потока, отчеты по созданным и решённым вопросам);
3) Обеспечить удобный интерфейс для командного взаимодействия;
4) Настроить сбор данных для анализа эффективности команды;
5) Провести пилотирование системы в условиях реального использования.

Актуальность проекта:

В современных командах разработки актуальна проблема неэффективного управления задачами, низкой прозрачности процессов и слабой координации между участниками. Проект направлен на решение этих задач с помощью простого, понятного и адаптивного инструмента.

Область применения:
Приложение разрабатывается как внутренняя платформа для Альфа-Банка, предназначенная для использования внутри IT-команд, занимающихся разработкой, тестированием и сопровождением продуктов банка. Продукт предназначен для внутреннего корпоративного использования и будет интегрирован в существующую инфраструктуру.

Ожидаемые результаты и планируемые достижения по завершении проекта:

1) Создание полноценного веб-приложения для управления задачами команды разработки, основанного на методологии Kanban, с возможностью визуального контроля и обновления задач в реальном времени.
2) Повышение прозрачности процессов разработки: все участники команды смогут отслеживать текущее состояние задач, видеть приоритеты и прогресс, что упростит планирование и контроль выполнения.

3) Интеграция инструментов для оценки эффективности команды, включая диаграмму суммарного потока, отчёт по созданным и решённым задачам, а также диаграмму Ганта — для визуализации сроков и загрузки.
4) Автоматизация рутинных процессов (создание, назначение и перемещение задач), что приведёт к снижению времени на управление проектом и уменьшению количества ошибок.
5) Проведение пилотного внедрения и проверка жизнеспособности решения, включая анализ того, насколько команда способна самостоятельно разработать, поддерживать и масштабировать данное приложение.
6) Документированная архитектура и требования, которые можно использовать в будущем при развитии или передаче проекта.
7) Подтверждение гипотезы, что собственное решение может стать альтернативой готовым продуктам (таким как Jira или Trello), при этом быть адаптированным под конкретные внутренние процессы команды.

1. Основная часть

1.1. **Работа участников проекта:**

1.1.1. Работа тимлида – Михайленко Кирилл Андреевич:

Введение

В рамках проекта, инициированного компанией Альфа Банк, я, как тимлид, взял на себя ответственность за координацию команды, управление процессами и обеспечение выполнения всех задач в соответствии с ожиданиями заказчика. Основной целью было создание веб-сервиса, соответствующего заданному техническому стеку и бизнес-целям. В данном документе описан процесс моей работы, ключевые задачи и выводы, сделанные в ходе реализации проекта.

1. **Формирование команды и распределение ролей**

Первым шагом было формирование команды, где каждый участник получил роль, соответствующую его навыкам и опыту. Это позволило эффективно использовать сильные стороны каждого члена команды и обеспечить слаженную работу. Я организовал начальный созвон с заказчиком, чтобы обсудить концепцию проекта, его долгосрочные цели и возможные пути развития. Это помогло заложить основу для дальнейшей работы и обеспечить единое видение проекта.

2. **Планирование и организация**

Совместно с аналитиком мы разработали видение проекта, определив примерный функционал приложения и пользовательские сценарии. Был составлен календарный план с четкими дедлайнами, которые стали ориентиром на протяжении всего проекта. Регулярные созвоны с куратором и заказчиком позволяли держать всех участников в курсе текущего прогресса. После каждого обсуждения я подводил итоги и передавал команде четкие инструкции, что обеспечивало прозрачность и координацию.

3. **Работа с макетами и функционалом**

Совместно с дизайнером мы прорабатывали макеты интерфейса, уделяя внимание их функциональности и удобству для пользователей. После получения обратной связи от заказчика мы оперативно вносили правки, дорабатывали функционал и устраняли недочеты. Это требовало внимательного подхода и постоянного взаимодействия с командой.

4. **Техническая поддержка и обучение**

Как тимлид, я изучал необходимый стек технологий, чтобы поддерживать фронтенд- и бэкенд-разработчиков. Это включало углубление в детали работы с выбранными инструментами и предоставление рекомендаций по оптимизации. Также я занимался тестированием веб-сервиса, выявлением багов и проверкой соответствия календарного плана.

5. **Рефлексия и улучшение процессов**

После каждого этапа мы проводили внутренние обсуждения, анализируя успехи и выявляя слабые места. Это позволило нам осознать нехватку определенных знаний и принять решение об их углублении. Мы увеличили частоту внутренних совещаний, чтобы лучше анализировать ошибки и предотвращать их в будущем. Также мы договорились задавать заказчику более детализированные вопросы, чтобы минимизировать недопонимания.

6. Итоги и планы на будущее

За семестр команда успешно справилась с задачей создания веб-сервиса. Еженедельные созвоны помогли сплотить команду и достичь поставленных целей. В будущем мы планируем доработать проект, расширить его функционал и продолжать развивать наши навыки. Роль тимлида позволила мне не только координировать процессы, но и создать атмосферу сотрудничества, где каждый участник чувствовал свою значимость.

1.1.2. Работа фронтенд разработчика – Шутов Кирилл Вячеславович:

В данной проектной работе фронтенд-разработчик занимался разработкой пользовательского интерфейса и интеграции с бэкенд-частью приложения.

Стек технологий:
В проекте в качестве основного инструмента разработки используется JavaScript-библиотека React с дополнением в виде React Router. Проект реализуется на языке программирования TypeScript и языках разметки JSX и CSS. Также для автоматизации рабочих процессов внутри проекта используется Vite.

Этапы разработки:
1) Аналитика требований к проекту.
2) Проектирование архитектуры проекта.
3) Реализация интерфейса по прототипу.
4) Настройка роутинга.
5) Интеграция с бэкендом.
6) Тестирование.
7) Деплоймент.

Функции фронтенд-разработчика:
1) Адаптивная верстка UI.
2) Реализация интерактивности.
3) Настройка переключения страниц сайта.
4) Настройка взаимодействия с бэкендом.
5) Обеспечение производительности и оптимизации.
6) Тестирование и отладка интерфейса

В данном проекте у получилось реализовать все необходимые для MVP функции приложения: реализованя регистрация аккаунтов и логин пользователя, выход из аккаунта пользователя, переключение страниц приложения, создание и редактирование задач, получение подробной информации о задаче, создание проектов, создание досок в проекте, а также

создана страница сводки для остальных страниц сервиса, которые пока не интегрированы с бэкенд-частью приложения.
Основные трудности при разработке возникли при интеграции сервиса с бэкенд-частью, а именно в реализации функций, требующих авторизацию пользователя и запись файлов cookie. Данную проблему удалось решить совместно с бэкенд-разработчиком путем изменения настроек CORS и параметров cookie’s. Также трудности возникли при настройке роутинга и URN страниц, т. к. сама структура нашего приложения многоуровневая и простроить правильную иерархию маршрутов затруднительно. В связи с этим пользователь мог видеть в консоли большое количество ошибок при переключении между страницами приложения. Решить данную проблему удалось с помощью библиотеки React Router DOM и ее компонента Browser Router

1.1.3. Работа бэкенд разработчика – Горбунов Иван Сергеевич:

Задачи:

1. Проектирование и миграции БД:

Спроектировал схему базы данных (PostgreSQL) для хранения задач, проектов, пользователей и их взаимосвязей.
Применил миграции с помощью Entity Framework Core.

2. Разработка API:

Реализовал REST API на C# (ASP.NET Core) для работы с задачами, досками, проектами и пользователями.
Добавил аутентификацию и авторизацию.

3. Демон для сбора статистики:

Написал фоновый сервис для сбора метрик (время выполнения задач, активность пользователей и др.).

4. Телеметрия и мониторинг:

Настроил логирование через Serilog (вывод в консоль и файлы).
Интегрировал Prometheus для сбора метрик и Grafana для визуализации.

5. Хостинг:

Развернул сервисы в контейнерах (Docker).

Стек технологий:
Backend: C#, ASP.NET Core, Entity Framework Core
База данных: PostgreSQL
Мониторинг: Prometheus, Grafana
Логирование: Serilog
Развертывание: Docker

Трудности и их решение:

Проблемы с Prometheus:
Не удавалось поднять контейнер из-за SELinux и ограничений непривилегированного пользователя.
Решение: Отключил SELinux в тестовой среде и настроил правильные права для пользователя.

Результаты:

6. Готовое API для управления задачами и проектами.
7. Рабочий демон для сбора статистики с метриками в Grafana.
8. Настроенная телеметрия (логи в файл + консоль, алерты в Telegram).
9. Успешное развертывание в контейнерах.

Вывод:

Проект был успешно реализован, несмотря на сложности с инфраструктурой мониторинга. Получилось создать удобное API с системой сбора метрик и алертинга, что позволяет отслеживать состояние системы в реальном времени.

1.1.4. Работа дизайнера – Долгих Александра Сергеевна:

Основной задаче была разработка прототипов системы. Первым делом были продуманы макеты дизайна. Целью было сделать систему, которой будет удобно и эстетически приятно пользоваться, а также на которую будет комфортно перейти с Jira.

Основой цветовой гаммы стали зелёный и голубой цвета, они представляют из себя сочетание аналогией по цветовому кругу. Зелёный цвет придаёт свежести нашему продукту, тогда как голубой придаёт ему лёгкости и в то же время стимулирует работу мозга. Эти цвета значительно отличаются от цветов, используемых в Jira, что придаёт нашему проекту уникальности.

Были разработаны несколько разделов: «Доска», «Сводка», «Хронология», «Календарь». На каждом экране присутствует боковое и верхнее меню. Боковое меню включает в себя Название проекта с которым в данный момент работает человек, а также кнопки переключения между разделами. Верхнее меню содержит аватар профиля с возможностью выхода из профиля и перехода на страницу профиля, настройки, кнопку помощи, а также выпадающие меню «Проекты», «Мои задачи», «Команды», «Ещё» и кнопку «Создать». Рассмотрим подробнее каждый из разделов системы.

Раздел «Доска» содержит несколько колонок: «К выполненю», «В работе» и «Готово». В колонку «К выполнению» можно добавлять карточки с задачами, а затем перетаскивать их между колонками. Мини-версии карточек задач содержат название задачи, маркер о типе задачи, а также аватар профиля человека, кому назначена данная задача. Также на экране раздела «Доска» присутствует название самой доски, возможность поиска задачи по названию, фильтрация по типам задач и Список участников данной доски.

Раздел «Сводка» содержит несколько экранов. На первом есть сводка статусов задач с круговой диаграммой, распределение рабочей нагрузки

команды и список всех задач проекта. При переключении между отчётами можно увидеть диаграммы суммарного потока, созданные и решённые запросы.

Раздел «Хронология» содержит своего рода диаграмму Ганта, распределённую по кварталам. На ней прописаны задачи и их так же можно добавлять.

Раздел «Календарь» содержит соответственно календарь на месяц с задачами по дням, где можно переключаться между месяцами.

Всё содержимое разделов было согласованно с заказчиком и макеты были переданы в программную разработку.

1.1.5. **Работа аналитика** – Евстигнеева Екатерина Сергеевна:

В рамках проекта по созданию системы управления задачами я выполняла роль аналитика. Основной моей задачей было собрать, структурировать и документировать требования заказчика, провести конкурентный анализ, определить цели, задачи и границы проекта, а также сопровождать команду дизайнеров и разработчиков в процессе реализации.Работа аналитика в этом проекте охватила полный цикл: от первичной инициации проекта до подготовки документации и поддержки команды в вопросах, касающихся логики и бизнес-потребностей. Благодаря тесному взаимодействию с заказчиком, дизайнером и командой разработки удалось обеспечить прозрачность проекта и минимизировать риски недопонимания требований на этапе реализации.

Основные выполненные задачи

1. Сбор и анализ требований

На первоначальном этапе проекта я провела серию интервью с заказчиком для уточнения потребностей, болей текущего процесса и желаемого функционала системы. На основе этого были сформулированы бизнес-требования, которые позже легли в основу технического задания (ТЗ).

2. Подготовка бизнес-требований

Документ бизнес-требований включал: постановку задачи, проблемы заказчика, цели, ожидаемые метрики и визуализации, ключевые ограничения. Он был согласован с заказчиком и передан в работу дизайнеру и команде разработки.

3. Создание технического задания

На основе бизнес-требований я подготовила подробное ТЗ, включающее функциональные и нефункциональные требования, диаграммы переходов между экранами, описание ролей пользователей и условий работы системы.

4. Визуализация структуры в Miro

С целью наглядного представления архитектуры продукта и пользовательских сценариев я создала структуру сайта в Miro. Эта визуализация помогла команде быстрее согласовать сценарии взаимодействия и упростила работу дизайнеру.

5. Анализ конкурентов

Я проанализировала несколько конкурирующих решений: Jira, Trello, ClickUp, Asana. Каждому аналогу были присвоены оценки по простоте, гибкости, аналитике, интерфейсу. Сделан вывод: наш продукт должен быть интуитивным, как Trello, но с метриками, как у Jira.

6. Сопровождение дизайна

Я активно взаимодействовала с дизайнером, обсуждала UX/UI решения, составляла текстовые фреймы, проводила ревью макетов на соответствие требованиям и логике пользователей.

Возможные трудности и как я с ними справилась:

1. Непонимание технических ограничений

Иногда заказчик не учитывал сложность реализации. Решение: я объясняла компромиссы, предлагала упрощенные MVP-варианты.

2. Ошибки в дизайне ролей

В первом прототипе были перепутаны роли. Решение: я подготовила диаграмму ролей и провела сессию ревизии с дизайнером.

Заключение

Работа аналитика в этом проекте была ключевой для определения вектора развития продукта. Благодаря выстроенной структуре требований, визуализации, грамотному взаимодействию с дизайнером и заказчиком, удалось избежать значительных переработок и обеспечить слаженное движение команды. У нас есть согласованные бизнес- и технические требования, база для реализации MVP, и команда понимает архитектуру и сценарии.

2.1. Требования заказчика и пользователей к программному продукту

Функциональные требования:

1. Возможность создания, редактирования и удаления задач.
2. Отображение задач в формате Kanban-доски с колонками ("To Do", "In Progress", "Done").
3. Возможность привязки задач к участникам команды.
4. Автоматический расчет и отображение метрик:
   4.1. Диаграмма суммарного потока задач;
   4.2. Отчет по созданным и решенным задачам;
   4.3. Отображение диаграммы Ганта для планирования сроков задач.
5. Возможность фильтрации и сортировки задач по статусу, дате, исполнителю.
6. Комментарии и обсуждение задач внутри карточки.
7. Возможность прикрепления файлов к задаче.
8. Авторизация пользователей и разграничение ролей (администратор, разработчик, менеджер).

Нефункциональные требования:

1. Удобный, интуитивно понятный пользовательский интерфейс.
2. Высокая производительность при большом количестве задач.
3. Сохранность и защита пользовательских данных.
4. Адаптивность интерфейса под разные устройства (десктоп/мобильные).

Бизнес-требования:

1. Повысить прозрачность и предсказуемость процесса разработки.
2. Снизить издержки на коммуникацию в команде.
3. Обеспечить гибкость управления задачами без бюрократизации.
4. Провести пилотное внедрение и оценить эффективность самописного решения.

3.1. Обзор архитектуры программного продукта:

Программный продукт представляет собой веб-приложение для управления задачами, построенное по принципу клиент-серверной архитектуры. Клиентская часть реализована с использованием библиотеки React и написана на языке TypeScript с JSX и CSS. Серверная часть построена на платформе ASP.NET Core
и написана на C# с использованием Entity Framework Core для взаимодействия с базой данных PostgreSQL. Хостинг осуществляется с помощью Docker-контейнеров.
Основные компоненты архитектуры:
- Клиентская часть (React, TypeScript): UI, роутинг, обработка данных;
- Серверная часть (ASP.NET Core): REST API, логика работы с задачами, проектами, пользователями;
- База данных (PostgreSQL): хранение сущностей и их связей;
- Фоновый демон: сбор и агрегация статистики;
- Мониторинг и логирование: Prometheus, Grafana, Serilog.

Обоснование архитектурного решения

Выбор React и TypeScript позволяет обеспечить модульную, масштабируемую и типобезопасную разработку клиентского интерфейса.
Использование ASP.NET Core на серверной стороне обеспечивает высокую производительность, безопасность и широкие возможности для построения API.
Docker-контейнеризация облегчает развертывание и масштабирование приложения.

Описание методологии разработки

В проекте использовалась методология Kanban. Она позволяет непрерывно поставлять функциональность, фокусироваться на потоке задач и улучшать процессы
на основе метрик. Задачи фиксировались в доске, проводились спринты, сессии ретроспектив и регулярные стендапы.

Этапы разработки

1) Аналитика требований к проекту
2) Проектирование архитектуры проекта
3) Реализация интерфейса по прототипу
4) Настройка роутинга
5) Интеграция с бэкендом
6) Тестирование
7) Деплоймент

4.1. Разработка и инфраструктура

1. Проектирование и миграции БД
Спроектирована схема БД на PostgreSQL для хранения задач, проектов, пользователей и связей между ними.
Миграции выполнены через Entity Framework Core.
2. Разработка API
REST API реализован на C# (ASP.NET Core). Включает CRUD-операции для задач, проектов, пользователей. Реализована авторизация и аутентификация.
3. Демон для сбора статистики
Фоновый сервис собирает метрики: длительность задач, активность пользователей и пр.
4. Телеметрия и мониторинг
- Serilog — логирование в консоль и файлы;
- Prometheus — сбор технических метрик;
- Grafana — визуализация данных мониторинга.
5. Хостинг
Все сервисы развернуты в Docker-контейнерах.
6. Используемый стек технологий
Frontend: React, TypeScript, Vite, CSS
Backend: C#, ASP.NET Core, Entity Framework Core
База данных: PostgreSQL
Мониторинг: Prometheus, Grafana
Логирование: Serilog
Развертывание: Docker

5.1. **Анализ и сопоставление аналогов разрабатываемого продукта:**

В рамках проекта по созданию веб-приложения для управления задачами была проведена аналитическая работа по изучению существующих решений на рынке. Цель анализа — выявить сильные и слабые стороны аналогичных систем, а также сформировать обоснование для проектных решений.

Таблица: Сравнение аналогов

Название	Функциональность	Плюсы	Минусы
Jira	Полноценная система управления проектами с Kanban, Scrum, отчётами, расширениями.	Гибкость, широкая экосистема, мощная аналитика.	Сложный интерфейс, высокая стоимость, избыточность для небольших команд.
Trello	Kanban-доски, чек-листы, напоминания, Power-Ups.	Простота, интуитивный UI, быстрая настройка.	Ограниченный функционал без расширений, слабая аналитика.
YouTrack	Kanban, Scrum, отчеты, тайм-трекинг, интеграции.	Интерфейс проще, чем у Jira, мощная поддержка Agile.	Менее популярный, слабее поддержка сообщества.
ClickUp	Универсальное управление задачами, цели, документы, time tracking.	Широкий функционал, гибкость настроек.	Иногда перегружен, интерфейс требует привыкания.
Linear	Минимализм, высокая скорость, ориентированность на разработчиков.	Быстрота, лаконичность, хорошая UX/UI.	Меньше возможностей по кастомизации, слабее отчетность.

(таблица 1)

На основании анализа можно сделать следующие выводы:

- Jira и YouTrack подходят для крупных команд и комплексных проектов, но

требуют высокой степени погружения.
- Trello и ClickUp — удобны для небольших команд, однако их функционал ограничен или требует подключения расширений.
- Linear — современное и быстрое решение, но пока уступает по гибкости и возможностям отчётности.

Разрабатываемый продукт должен:
- Сочетать простоту интерфейса и гибкость настройки;
- Обеспечивать прозрачность процессов и контроль загрузки команды;
- Поддерживать Kanban, базовую аналитику и визуализацию метрик (например, диаграмму Ганта);
- Иметь потенциал для масштабирования.

2. Заключение

Командой был успешно реализован проект по созданию веб-приложения для управления задачами команды разработки. В ходе проекта нами была разработана функциональная и удобная система, которая отвечает требованиям по прозрачности процессов, визуализации задач и аналитике производительности.

Разработка велась с применением современных технологий: React и TypeScript на фронтенде, ASP.NET Core и PostgreSQL на бэкенде, с использованием Docker для развёртывания, а также инструментов мониторинга и логирования. Мы обеспечили настройку роутинга, интеграцию с серверной частью, внедрили диаграммы и метрики, включая диаграмму Ганта и отчёт по выполненным задачам.

Работа над проектом велась по этапам: от анализа требований и проектирования архитектуры до финального тестирования и деплоймента. На каждом этапе мы отслеживали прогресс и координировали задачи, что позволило эффективно справляться с возникающими трудностями и соблюдать сроки выполнения.

В результате, нами была создана система, которая соответствует заявленным бизнес-целям: она упрощает командную работу, повышает прозрачность процессов и помогает анализировать эффективность разработки. Приложение готово к использованию и обладает потенциалом для дальнейшего масштабирования.

3. Список использованных источников

1. Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс ; пер. с англ. — 3-е изд. — М. : Вильямс, 2012. — 576 с.
2. Кент, У. Нормализация баз данных / У. Кент ; пер. с англ. — М. : Вильямс, 2003. — 240 с.
3. Фаулер, М. Шаблоны корпоративных приложений / М. Фаулер ; пер. с англ. — М. : Питер, 2005. — 560 с.
4. Роб, П. Базы данных: проектирование, реализация и сопровождение / П. Роб, К. Корнелл ; пер. с англ. — М. : Вильямс, 2011. — 1120 с.
5. Бейли, Б. Юзабилити: искусство легкости / Б. Бейли ; пер. с англ. — М. : Питер, 2011. — 224 с.
6. Купер, А. Об интерфейсе: основы проектирования взаимодействия / А. Купер, Р. Райман, Д. Кронин ; пер. с англ. — М. : Вильямс, 2014. — 528 с.
7. Кресс, Г. Веб-аналитика: сбор данных, анализ поведения пользователей и оптимизация сайта / Г. Кресс, К. Гатри ; пер. с англ. — М. : Диалектика, 2010. — 352 с.