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: Макаров А.В. _____

Куратор: Макаров А.В.
руководитель направления _____

Студенты команды «Chill Team»

Солдатов Т.Ю. _____

Кедров М.А. _____

Куликов Я.А. _____

Гилев А.Д. _____

Балабух А.А. _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью настоящего проекта является разработка и внедрение веб-приложения для управления задачами и проектами в IT-направлениях Альфа-Банка. Система призвана упростить планирование, исполнение и контроль задач внутри команд, а также предоставить управленцам и аналитикам актуальные данные о прогрессе, эффективности и нагрузке в рамках спринтов и долгосрочных проектов.

Задачи проекта включают:

1. создание единого пространства для командной работы на основе Kanban-подхода и матрицы Эйзенхауэра,
2. интеграцию инструментов анализа производительности и эффективности команд,
3. внедрение механизмов сбора метрик,
4. обеспечение удобного интерфейса для разных ролей: разработчиков, тестировщиков, тимлидов, менеджеров и аналитиков,
5. обеспечение совместимости с экосистемой внутренних продуктов Альфа-Банка,
6. развертывание минимально жизнеспособного продукта (MVP) до 31 мая 2025 года.

Альфа-Банк является технологически ориентированной организацией с развитой цифровой инфраструктурой. Однако текущие инструменты управления задачами зачастую фрагментированы, не предоставляют необходимого уровня аналитики и не учитывают специфику внутренних процессов. Внедрение собственного решения обеспечивает: гибкость настройки под внутренние процессы, сбор и анализ данных для принятия управленческих решений, контроль эффективности на всех уровнях разработки ПО.

Это особенно важно и актуально в условиях растущей цифровизации, необходимости ускорения Time-to-Market и повышения прозрачности процессов.

Область применения системы ориентирована на команды, работающие по гибким методологиям (Agile, Scrum, Kanban), и будет адаптирована под внутренние стандарты разработки в банке.

Ожидаемым результатом является создание MVP веб-приложения с полным набором функций, соответствующих требованиям заказчика.

1 Основная часть

1.1 Информацию о работе каждого участника в отдельности

Гилев Андрей выполнял роль продуктового и системного аналитика. Он отвечал за сбор, анализ и формализацию требований от всех заинтересованных сторон, включая разработчиков, менеджеров и аналитиков. На основании этих требований Андрей формировал подробные пользовательские сценарии и создавал документацию, отражающую бизнес-логику системы. В ходе проекта он также активно участвовал в планировании спринтов, помогал команде с приоритизацией задач и контролировал соответствие реализуемых функций ожиданиям пользователей.

Солдатов Тимофей исполнял обязанности тимлида и технического координатора проекта. Он организовывал работу команды, выстраивал техническую архитектуру решения и принимал ключевые инженерные решения. Тимофей отвечал за распределение задач между участниками, проведение код-ревью, соблюдение сроков и качества итогового продукта. Также он выступал техническим ментором, помогая коллегам решать сложные задачи и принимать архитектурные решения.

Кедров Максим занимался разработкой серверной части приложения. Он реализовал основную бизнес-логику на стороне бэкенда, разработал API для взаимодействия с клиентской частью и обеспечил надёжную работу с базой данных. Кроме того, Максим внедрил механизмы обработки ошибок, реализовал безопасную авторизацию пользователей через токены (например, JWT) и настроил сбор и обработку ключевых аналитических метрик. Он также активно участвовал в тестировании серверной части, что способствовало стабильной работе системы.

Балабух Александр выступал в роли дизайнера и UX-специалиста. Он

разработал макеты интерфейсов, включая Kanban-доски, матрицу Эйзенхауэра и панели аналитики. Александр обеспечил удобную и понятную навигацию в системе, активно взаимодействовал с разработчиками для корректного внедрения визуальных решений.

Куликов Яков занимался разработкой фронтенд-части приложения. Он реализовал клиентский интерфейс, обеспечил взаимодействие с серверным API и внедрил визуализацию задач, фильтрацию, отображение метрик и прочие элементы. Яков отвечал за пользовательский опыт, логику Kanban-досок и реализацию матрицы Эйзенхауэра. Также он работал над адаптивной верстkой, индикаторами загрузки и обработкой ошибок на клиенте.

1.2. Разбор требований заказчика и пользователей к программному

1.2.1 Авторизация

Функциональные требования: Пользователь должен иметь возможность войти в систему, используя логин и пароль.

Нефункциональные требования: Система обеспечивает безопасность данных пользователя.

1.2.2 Создание проекта

Функциональные требования: Пользователь имеет возможность создать новый проект, указав: название проекта, его описание, сроки реализации проекта, добавить людей, зарегистрированных в системе и присвоить им роли в проекте. Система позволяет пользователю установить сроки выполнения.

1.2.3 Создание, назначение и распределения задач

Функциональные требования: Пользователь, создавший проект, может создать, редактировать и удалять задачу в этом проекте, ввести ее название и написание, а также назначить ответственного за эту задачу, ставить сроки выполнения задачи. Пользователь, создавший проект, может предоставить права другим пользователям в проекте. Система должна уведомлять других участников о созданной для них задачи. Каждый пользователь проекта видит только те задачи на доске, которые доступны для его роли. При создании проекта на доске задач существуют столбцы ‘По умолчанию’: В очереди, В разработке, Тестируется, Выполнено

Нефункциональные требования: Задача и уведомления о ее создании приходят пользователю в течение минуты.

1.2.4 Доска задач: todo, in process, done

Функциональные требования: Пользователь, создавший задачу, должен иметь возможность распределить задачу на матрице. Пользователь, создавший проект, должен иметь возможность разделять задачи на категории: to do, in process, done. Пользователи, не создавшие проект, но находящиеся в проекте должны. Получать изменения их задач на доске.

Нефункциональные требование: Система обновляет задачи в реальном времени.

1.2.5 Определение приоритетности задачи: Матрица Эйзенхауэра

Функциональные требования: Пользователь, создавший задачи, может распределить их приоритетность по матрице Эйзенхауэра в рамках одной матрицы. Для других пользователей должна быть возможность отображения результата в виде матрицы Эйзенхауэра по тем задачам, которые относятся к этому пользователю.

Нефункциональные требования: Система при изменении задач корректно отображает новые результаты для всех участников.

1.2.6 Отслеживание выполнения задач

Функциональные требования: Пользователь должен иметь возможность обновлять статус выполнения задачи. Исходя из обновляемых результатов подзадач система должна отображать выполненный прогресс пользователя по задаче. Система должна отслеживать сроки выполнения пользователем задач.

Нефункциональные требования: Обновление статуса и прогресса по задачам должно происходить в реальном времени.

1.2.7 Получение метрик по проекту и эффективности команды

Функциональные требования: При запросе система дает отчет о выполненных задачах и эффективности работы команды.

Нефункциональные требования: Сохранение отчета возможно в форматах CSV и PDF.

1.2.8 Закрытие задач и проекта

Функциональные требования: У пользователя при завершении всех подзадач в задачи задача автоматически переходит в раздел выполненных.

Нефункциональные требования: Завершенная задача пользователя пропадает из матрицы Эйзенхауэра.

1.2.9 Фильтры Kanban доски

При активации фильтра «Исполнитель» система отображает выпадающий список, содержащий перечень всех исполнителей, зарегистрированных в базе данных (поле User_name из таблицы USER).

Пользователь имеет возможность выбрать одного или нескольких исполнителей для фильтрации задач. После применения фильтра в интерфейсе отображаются исключительно те задачи (из таблицы TASK), которые назначены выбранным исполнителям. В случае если ни один исполнитель не был выбран, фильтрация не осуществляется, и система продолжает отображать полный перечень задач без изменений.

Фильтр «Дедлайн» при активации инициирует появление всплывающего окна с интерфейсом календаря, позволяющего пользователю задать временной диапазон. Пользователь может указать начальную и конечную даты периода, после подтверждения которого система отфильтрует задачи. В результате отображаются только те записи (включая задачи из таблицы TASK и дефекты из таблицы DEFECT), у которых значение дедлайна (поля Deadline и End_date соответственно) попадает в указанный диапазон, включая граничные даты. Если временной интервал не задан, фильтр остается неактивным, и данные выводятся в исходном виде.

При нажатии на фильтр «Приоритет» открывается выпадающий список с доступными вариантами: «Высокий», «Средний» и «Низкий». Пользователь может выбрать один или несколько уровней приоритета для фильтрации задач. После применения фильтра в интерфейсе отображаются только те задачи, которые соответствуют выбранным уровням приоритета. Если ни один вариант не выбран, фильтр не применяется, и задачи отображаются в стандартном порядке. Дополнительно предусмотрена возможность сортировки задач по приоритету - в порядке убывания (от высокого к низкому) или возрастания (от низкого к высокому), что позволяет гибко настраивать отображение списка задач в соответствии с текущими потребностями пользователя.

1.2.10 Отображение задач, согласно выбранной команде

При нажатии на поле «Выбрать проект» появляется всплывающий список со всеми командами, в которых находится пользователь. Project_id и Title из PROJECT. При нажатии на команду, загружаются те задачи пользователя, которые у него есть в этой команде Task_id, Title, Deadline, Status_id из TASK

1.3 Описание методологии разработки

В рамках проекта использовалась гибкая методология разработки Agile. Команда работала по итерационной модели, разделенной на спринты продолжительностью две недели. Такой подход позволил регулярно получать обратную связь, вносить коррективы на основе пользовательских сценариев и постепенно наращивать функциональность системы, сохраняя высокую скорость разработки.

Каждый спринт начинался с планирования задач, на котором определялись приоритетные функции и распределялись роли. По окончании каждого спринта команда проводила демонстрацию реализованного функционала, обсуждала результаты и вносила изменения в бэклог. Использование Agile-практик позволило эффективно взаимодействовать между аналитиками, разработчиками, дизайнером и тимлидом.

1.4 Этапы разработки

Проект велся поэтапно в соответствии с планом спринтов и ключевых задач. На первом этапе, с 13 марта по 19 марта 2025 года, команда сосредоточилась на разработке требований и составлении технического задания. Продуктовый аналитик сформулировал основные пользовательские сценарии, были определены ключевые функции продукта и составлена документация, отражающая бизнес-логику и цели проекта.

С 19 марта по 2 апреля 2025 года проходил этап дизайна и настройки инфраструктуры проекта. Были разработаны макеты пользовательского интерфейса, определена структура веб-приложения, а также выполнена базовая настройка сервера. Параллельно проводилась автоматизация рабочих процессов, включая настройку репозиториев, CI/CD и окружений для разработки и тестирования.

С 2 апреля по 2 мая 2025 года велась активная разработка клиентской и серверной частей системы. Фронтенд-разработчик реализовал интерфейсы Kanban-доски и матрицы Эйзенхауэра, подключение к API, фильтрацию и обработку состояния. Бэкенд-разработчик создавал REST API, настраивал базу данных и реализовывал бизнес-логику. Этап включал также интеграцию аналитики и реализацию пользовательской авторизации.

В начале мая, до 4 мая, завершались задачи по доработке бэкенда и верстке интерфейсов. Это включало стилизацию компонентов, устранение UI-багов и финальную настройку серверных методов.

18 мая был проведён этап интеграции клиентской и серверной частей, обеспечивает связку фронтенда с бэкэндом, корректную работу API и передачу данных между слоями.

Финальный этап — отладка, тестирование и размещение проекта на хостинге — завершился к 31 мая 2025 года.

1.5 Видеофиксация прогресса

В качестве промежуточных отчётов команда подготовила два видеоролика для контрольной точки 1 [1] и контрольной точки 2 [2], отражающих процесс разработки, демонстрацию реализованных функций. Эти видеозаписи заменяют классические текстовые протоколы и выполняют функцию интерактивной отчетности, особенно удобной в условиях командной работы в Agile.

1.6 Тестирования на промежуточных этапах, разбор выявленных ошибок

В ходе тестирования были выявлены и успешно устранены ключевые проблемы, связанные с согласованностью работы фронтенд-разработчиков, полнотой API-документации, валидацией данных и структурой базы данных. Команда оперативно реагировала на недочеты:

1. стандартизация кода – различие в подходах разработчиков было нивелировано согласованием единого стиля реализации, что улучшило поддерживаемость кода,
2. уточнение API – пропущенные в Swagger методы были документированы, что исключило ошибки интеграции между фронтендом и бэкендом,
3. оптимизация UX – ограничение длины названий задач до 15 символов повысило удобство восприятия интерфейса,
4. доработка архитектуры – корректировка ER-диаграммы с добавлением сущности «Доска» устранила пробелы в логике данных.

Несмотря на тщательную подготовку аналитической документации, часть нюансов (например, ограничения полей или дополнительные сущности) была уточнена лишь на этапе тестирования. Это подчеркивает важность: пилотного тестирования критичных компонентов на ранних стадиях, регулярного ревью требований с привлечением разработчиков и тестировщиков, гибкости процессов для оперативного внесения изменений.

Итог: Проект успешно завершен благодаря оперативному устранению недочетов и слаженной работе команды. Полученный опыт свидетельствует о необходимости более детального проектирования с учетом edge-кейсов, что будет учтено в будущих итерациях. Для совершенствования процессов рекомендуется: внедрить чек-листы валидации требований для аналитиков, проводить кросс-ревью между разработчиками на этапе проектирования API,

добавить предварительное тестирование UX на реалистичных данных.

Команда продемонстрировала высокую эффективность в решении проблем, что создает прочный фундамент для дальнейшего развития продукта.

1.7 Анализ и сопоставление аналогов разрабатываемого продукта

В настоящее время на рынке представлено множество решений для управления проектами и задачами, каждое из которых обладает уникальными характеристиками. Лидером в данной области является система Jira [3], представляющая собой полноценную Agile-платформу с поддержкой методологий Scrum и Kanban. Данное решение отличается мощными возможностями трекинга времени и аналитики, а также обширной экосистемой интеграций, превышающей 3000 подключений. Однако сложность интерфейса и необходимость продолжительного обучения персонала существенно ограничивают область его эффективного применения.

Альтернативным решением выступает Yougile [4], ориентированный на кастомизацию рабочих процессов. Платформа предлагает гибкую систему управления задачами через стикеры, поддерживает локальное развертывание и включает встроенные средства коммуникации. Вместе с тем, система демонстрирует ограниченные возможности в части аналитики и интеграции с корпоративными сервисами, что снижает её привлекательность для крупных организаций.

Среди специализированных решений следует отметить Яндекс.Трекер [5], обеспечивающий автоматизацию процессов через API и поддержку гибридных методов управления. Несмотря на развитую функциональность, продукт имеет ограниченную экосистему, ориентированную преимущественно на внутренние сервисы компании. Аналогичным образом система Weeek [6], предлагающая инструменты тайм-менеджмента и интеграцию с дизайн-приложениями, в большей степени подходит для индивидуального использования, нежели для комплексного управления проектами в корпоративной среде.

Особого внимания заслуживают нишевые продукты, такие как Shtab [7], предназначенный для контроля удалённых сотрудников, и Kaiten [8],

ориентированный на продуктовые команды. Первый обеспечивает мониторинг рабочего времени с функцией скриншотов, но обладает избыточным уровнем контроля, второй предлагает специализированные инструменты для работы с гипотезами, но имеет ограничения по масштабированию. Решение Strive [9], делающее акцент на прозрачности процессов, демонстрирует эффективность в части отображения активности сотрудников, однако уступает конкурентам по глубине аналитических возможностей.

Наш продукт, созданный специально для экосистемы Альфа Банка, объединяет сильные стороны Jira (мощная аналитика и отчетность) [3], Yougile (гибкость и кастомизация) [4] и Weeek (удобство интерфейса) [6], дополняя их глубокой интеграцией с банковскими сервисами, автоматизацией рутинных процессов и встроенными инструментами контроля эффективности. В отличие от сложных и универсальных решений вроде Jira [3], мы предлагаем интуитивно понятный, безопасный и адаптированный под нужды банка инструмент, который повышает прозрачность работы, ускоряет процессы и упрощает управление командами — без лишнего обучения и с мгновенным результатом.

Для более детального анализа конкурентов обратитесь к таблице 1.

Таблица 1 – «Таблица анализа конкурентов»

	Критерии оценивания	Название конкурентов				
		Strive	Jira	Яндекс. Трекер	Weeek	Shtab
Критерий 1	Тип отображения	Канбан, таймлайн, список	Канбан, скрам, список, roadmap	Канбан, скрам	Доски, календарь	Канбан, матрицы
Критерий 2	Интеграция	-	Экосистема Atlassian	API для импорта	Google, Figma	-
Критерий 3	Автоматизация	-	Гибкие workflow, Jira Automate	Правила, API	-	Авто скрин
Критерий 4	Командные функции	Онлайн--активность	Комментарии, поручения	Очереди задач	Чат	Учет зарплат

ЗАКЛЮЧЕНИЕ

В рамках разработки программного продукта для управления задачами внутри экосистемы Альфа-Банка была реализована MVP-версия, соответствующая основным требованиям заказчика и базовым ожиданиям пользователей. Продукт обеспечивает ключевые функции: постановку и отслеживание задач, разграничение ролей, базовую визуализацию прогресса с помощью диаграмм. Пользователи отметили понятность интерфейса и логичность пользовательских сценариев, что подтверждает обоснованность принятых архитектурных и дизайнерских решений.

Анализ выполнения требований показал, что продукт успешно покрывает основной функционал управления задачами. Однако часть функций реализована в минимальном виде и требует дальнейшей проработки. Визуализация прогресса ограничена диаграммой контроля и накопительной диаграммой, а возможность генерации отчетов (например, в формате PDF).

По итогам тестирования продукт демонстрирует стабильную работу в основных сценариях. Выявленные баги были оперативно устранены. Архитектура приложения, основанная на разделении логики, данных и интерфейса, обеспечивает высокую тестируемость, масштабируемость и удобство сопровождения.

Для дальнейшего развития и повышения ценности продукта рекомендуется сосредоточиться на следующих направлениях:

Доработка функциональности: улучшение фильтрации задач, расширение ролей, настройка статусов и гибкость взаимодействия с задачами.

Расширение системы метрик: добавление новых визуализаций (burndown-чарт, распределение времени, загрузка исполнителей), а также аналитики по эффективности команды.

Формирование отчетности: внедрение функции генерации отчетов в

формате PDF с возможностью кастомизации данных.

Открытие исходного кода: публикация исходного кода проекта на открытой платформе (например, GitHub) с документацией и API, что позволит сообществу разрабатывать собственные плагины, расширять функциональность и способствовать развитию экосистемы.

Развитие панели фильтров, визуальных подсказок, группировок и тегов задач для лучшего пользовательского опыта.

Открытие исходного кода и поддержка системы расширений (плагинов) закладывают основу для создания активного сообщества вокруг проекта, что даст импульс к быстрому развитию продукта и его адаптации под различные сценарии использования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Chill Team. Видеоролик по прогрессу контрольной точки 1 [Электронный ресурс] / Chill Team. – VK Видео. – URL: https://vkvideo.ru/video-186355076_456239017 (дата обращения: 23.05.2025).
2. Chill Team. Видеоролик по прогрессу контрольной точки 2 [Электронный ресурс] / Chill Team. – VK Видео. – URL: https://vkvideo.ru/video-186355076_456239018 (дата обращения: 23.05.2025).
3. Atlassian. Jira Software [Электронный ресурс] / Atlassian. – URL: https://www.atlassian.com/software/jira (дата обращения: 23.05.2025).
4. Yougile. Онлайн-система управления проектами [Электронный ресурс] / Yougile. – URL: https://yougile.com/ru (дата обращения: 23.05.2025).
5. Яндекс.Трекер [Электронный ресурс] / Яндекс. – URL: https://tracker.yandex.ru (дата обращения: 23.05.2025).
6. Weeek. Платформа для управления задачами [Электронный ресурс] / Weeek. – URL: https://weeek.net (дата обращения: 23.05.2025).
7. Shtab. Контроль удалённых сотрудников [Электронный ресурс] / Shtab. – URL: https://shtab.app (дата обращения: 23.05.2025).
8. Kaiten. Визуальное управление задачами [Электронный ресурс] / Kaiten. – URL: https://kaiten.io (дата обращения: 23.05.2025).
9. Strive. Система управления командами [Электронный ресурс] / Strive. – URL: https://strive.app (дата обращения: 23.05.2025).
10. Chill Team. Техническое задание на разработку проекта [Электронный ресурс] : текстовый документ / Chill Team. – Google Docs. – URL:

https://docs.google.com/document/d/1slngqj2s9_ynhIyfWMDXnbRLnxKmEcDR0u6HZc2niVA/edit?usp=sharing (дата обращения: 23.05.2025).

11. Chill Team. Чек-лист по функционалу проекта [Электронный ресурс] : текстовый документ / Chill Team. – Google Docs. – URL: https://docs.google.com/document/d/1S9tvppiU3WQTgSepnmeGPZvPlGwhJn4I4Gl88m1_-NU/edit?usp=sharing (дата обращения: 23.05.2025).
12. Chill Team. Исходный код проекта [Электронный ресурс] / Chill Team. – GitHub. – URL: https://github.com/MaxKedroff/TaskTracker (дата обращения: 23.05.2025).
13. Chill Team. Дизайн интерфейса "Alfa Chill" [Электронный ресурс] / Chill Team. – Figma. – URL: https://www.figma.com/design/HsHxUM44UQU5UTM6FBobzC/Alfa-Chill?node-id=0-1&t=aAMRUurKKVNqsW2V-1 (дата обращения: 23.05.2025).
14. Chill Team. Документация API проекта [Электронный ресурс] / Chill Team. – Swagger. – URL: http://194.87.146.14/index.html (дата обращения: 23.05.2025).
15. Chill Team. Карта пользовательских историй (User Story Map) [Электронный ресурс] / Chill Team. – Figma. – URL: https://www.figma.com/board/kMFGnuhEUD2YtLfoxlUh80/USER-STORY-MAP---CHILL-TEAM?node-id=0-1&t=RdmSnKL7MFMAX00r-1 (дата обращения: 23.05.2025).

Приложение А

(обязательное)

Вспомогательные материалы

В приложение включены вспомогательные материалы, сопровождающие выполнение проектной работы по разработке информационной системы управления задачами в команде "Chill Team":

1. техническое задание (ТЗ) – документ с изложением требований, функций, целей и критериев приемки системы [10],
2. чек-лист по функционалу – документ, подтверждающая реализацию ключевых пользовательских сценариев и требований [11],
3. репозиторий исходного кода – публичный репозиторий на платформе GitHub содержит весь код проекта, организованный по backend- и frontend-модулям [12],
4. прототип пользовательского интерфейса – интерактивный дизайн интерфейса, разработанный в Figma, отражающий основные пользовательские сценарии [13],
5. документация API (Swagger) – автоматически сгенерированная документация REST API, описывающая структуру запросов, ответов и маршрутов [14],
6. user Story Map – карта пользовательских историй и функциональных блоков проекта, визуализированная в Figma [15].