Техническое задание

на разработку мини-CRM системы

Project-Planner

Версия 4.0
2025

Таблица изменений

Версия	**Дата**	**Автор**	**Описание изменений**
1.0	09.10.2025	Мельников Ю.Д.	Начало написания документа
1.5	20.10.2025	Гориченко М.С.	Внедрение анализа предметной области
2.0	1.11.2025	Мельников Ю.Д.	Изменение названия проекта
2.5	5.11.2025	Гориченко М.С.	Интеграция технологического стека
3.0	6.11.2025	Мельников Ю.Д.	Добавление требований к интерфейсу
3.5	13.11.2025	Гориченко М.С.	Интеграция схемы архитектуры системы
4.0	22.11.2025	Мельников Ю. Д.	Добавлено дополнение: цветовая схема и макеты интерфейса

Содержание

1. Основная информация о проекте

Название проекта: Project-Planner — мини-CRM система для организации задач и мероприятий с интеграцией Telegram-бота.

Планируемые внешние интеграции: Telegram API, Yandex Cloud.

Заказчики и исполнители: Мельников Юрий и Гориченко Максим

1.1. Плановые сроки выполнения работ

Этап разработки	Начало	Окончание	Продолжительность
Планирование и анализ требований	15.09.2025	28.09.2025	2 недели
Определение требований	29.09.2025	05.10.2025	1 неделя
Проектирование	06.10.2025	26.10.2025	3 недели
Разработка	27.10.2025	22.12.2025	8 недель
Тестирование	23.12.2025	19.01.2026	4 недели
Развертывание и стабилизация	20.01.2026	31.01.2026	2 недели
Итого			**20 недель**

1.2. Оценка затрат на разработку

Предполагаемые расходы на реализацию проекта включают:

- Оплата труда фронтенд-разработчика — 160 000 руб.;
- Оплата труда бэкенд-разработчика — 120 000 руб.;
- Оплата хостинга — 300 руб./мес (3 600 руб./год);
- Аренда домена — 1 300 руб./год;
- Программное обеспечение и инструменты разработки — 0 руб. (используются бесплатные версии).

Итого: около 284 900 руб.

1.3. Назначение и цели создания системы

Цель: создать лёгкий веб-инструмент для управления задачами и уведомлениями с Telegram-ботом для малых команд, студенческих объединений и стартапов.

Задачи:

- Проанализировать существующие решения;
- Разработать архитектуру и спроектировать базу данных;
- Реализовать CRUD-операции для задач и пользователей;
- Создать Telegram-бота для уведомлений и быстрого доступа;
- Реализовать минимальную аутентификацию и безопасность;
- Подготовить MVP с возможностью масштабирования.

Сервис предназначен для малых команд, студенческих объединений и стартапов, которым требуется лёгкий инструмент для управления задачами, отслеживания прогресса и уведомлений через Telegram. В отличие от крупных корпоративных CRM-систем, Project-Planner прост в установке, не требует сложной регистрации и подходит для локального развертывания.

2. Анализ предметной области

2.1. Целевая аудитория

Целевой аудиторией являются:

- Небольшие команды и стартапы;
- Студенческие клубы и объединения;
- Инициативные группы и волонтёрские объединения;
- Организаторы мероприятий;
- Учебные проекты.

2.2. Проблема и текущее состояние

Современные CRM-системы ориентированы на крупные компании и содержат избыточный функционал, требующий времени на освоение. Малые команды и волонтёрские объединения нуждаются в простом и лёгком решении для организации задач.

2.3. Анализ аналогов

Система	**Сильные стороны**	**Слабые стороны**
Trello	Удобная доска задач, система меток и уведомлений	Требует регистрации, не интегрируется с Telegram
Notion	Гибкость, база данных, шаблоны	Избыточен для малых команд, сложен в настройке
Bitrix24	CRM-модули, отчётность, интеграции	Перегруженный интерфейс, высокая стоимость
Asana	Отличная визуализация проектов, мобильные приложения	Ограниченный бесплатный тариф, нет Telegram-уведомлений

2.4. Существующие проблемы

- Слишком сложные интерфейсы для небольших команд;
- Требуют постоянного интернет-доступа и регистрации;
- Отсутствует нативная интеграция с Telegram;
- Нет простого решения для локального или облачного развёртывания.

3. Требования к системе

3.1. Функциональные требования

- Создание, редактирование и удаление проектов и задач;
- Настройка статусов задач (Kanban-доска);
- Назначение задач пользователям;
- Уведомления через Telegram-бот;
- Просмотр статистики по проектам;
- Фильтрация и поиск по задачам;
- Поддержка нескольких проектов и участников;
- Управление пользователями и ролями.

3.2. Требования к интерфейсу

- Аккуратный, минималистичный дизайн;
- Цветовая схема включает два цвета и их градиент;
- Возможность изменения темы интерфейса через соответствующую кнопку;
- Все элементы интерфейса реализованы с использованием фреймворка для упрощения разработки;
- Веб-приложение доступно только на ПК.

3.3. Требования к локализации

- Интерфейс сервиса должен поддерживать минимум русский язык;
- Автоматическое определение языка интерфейса в зависимости от местоположения пользователя:
  - Для СНГ — русский язык;
  - В остальных случаях — английский язык;
- Возможность ручного изменения языка интерфейса через соответствующую кнопку.

3.4. Нефункциональные требования

- Минималистичный и интуитивно понятный интерфейс;
- Локальное развертывание и простая установка;
- Безопасное хранение паролей (hashlib);
- Возможность интеграции с облаком (Yandex Cloud);
- Документация API.

3.5. Архитектура системы

Сервис представляет собой веб-приложение, доступное только на ПК. Архитектура системы основана на клиент-серверной микросервисной модели и разделена на три слоя: базовый, дополнительный и внешний.

3.5.1 Базовый слой

Базовый слой включает основные компоненты, обеспечивающие функциональность и взаимодействие системы:

- **Бэкенд:** реализуется на платформе Flask, выполняет бизнес-логику, обработку запросов и взаимодействие с хранилищем данных;
- **Фронтенд:** реализуется на HTML/CSS/JS с Bootstrap, обеспечивает интерактивный интерфейс пользователя и обмен данными с сервером через REST API;
- **База данных:** используется SQLite в качестве основной реляционной СУБД для хранения структурированных данных приложения.

3.5.2 Дополнительный слой

Дополнительный слой предназначен для повышения производительности, мониторинга и аналитики:

- **Кэширование:** реализуется с помощью In-memory cache, используется для ускорения обработки часто запрашиваемых данных и уменьшения нагрузки на основную базу данных;
- **Сбор метрик:** осуществляется при помощи Prometheus, который агрегирует показатели работы сервисов и инфраструктуры;
- **Визуализация метрик:** осуществляется через Grafana, предоставляющую наглядные дашборды и инструменты анализа производительности.

3.5.3 Внешний слой

Внешний слой содержит интеграции с внешними системами и сервисами:

- **Хранилище файлов:** используется локальное хранилище для загрузки и хранения медиафайлов;
- **Telegram API:** интеграция с Telegram для уведомлений и управления через бота;
- **Yandex Cloud:** возможность интеграции с облачными сервисами Yandex.

3.5.4 Схема архитектуры системы

3.5.5 Контейнеризация и развертывание

Каждый микросервис (за исключением внешних интеграций) развёртывается в отдельном Docker-контейнере, что обеспечивает:

- Изоляцию окружений;
- Гибкость масштабирования;
- Удобство развёртывания.

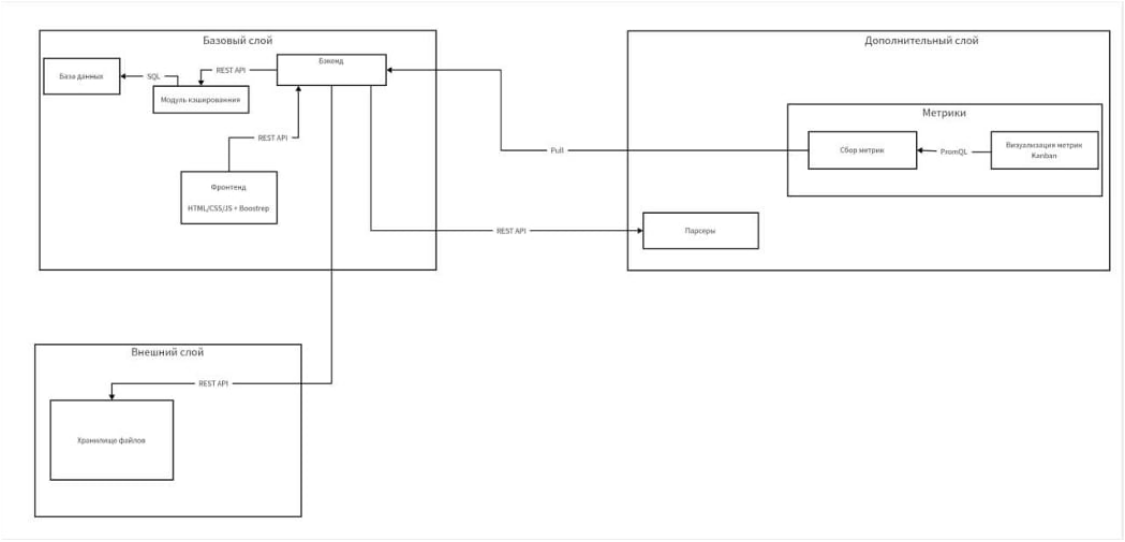

Рис. 1: Схема архитектуры системы Project-Planner

3.6. Технологический стек

- **Фронтенд:** HTML/CSS/JS + Bootstrap
- **Бэкенд:** Python (Flask)
- **База данных:** SQLite
- **Кэширование:** In-memory cache
- **Мониторинг:** Prometheus, Grafana
- **Хранение файлов:** Локальное хранилище
- **Инфраструктура:** Git, GitHub, Docker, Virtual Environment
- **Тестирование:** pytest, Postman
- **Аутентификация:** Flask Session, hashlib
- **Инструменты проектирования:** Draw.io, Figma
- **Документация:** Markdown, LaTeX

3.7. Требования к численности и квалификации персонала

- Заказчик системы единолично имеет полный доступ ко всем возможностям системы и обладает необходимой квалификацией для этого.
- В качестве дополнительного персонала могут выступать модераторы базы данных и модераторы проектов:
  - **Модератор базы данных** должен обладать компетентностью по работе с CRUD баз данных и понимать принципы работы с медиа.
  - **Модератор проектов** должен обладать компетентностью в рамках выбранного им проекта.
- Требований к режиму работы персонала нет.

3.8. Требования к безопасности

- Использование HTTPS для защиты соединения;
- Хеширование паролей пользователей с использованием hashlib;
- Контроль доступа к API;
- Минимизация пользовательских данных в базе;
- Защита от основных веб-уязвимостей.

3.9. Хранимые сущности

Основные сущности системы:

- **Пользователь:** Имя, логин, роль, Telegram ID, статус активности
- **Задача:** Название, описание, срок, статус, ответственный
- **Статус:** Название, цвет, приоритет
- **Проект:** Название, описание, дата создания, участники
- **Уведомление:** Текст, дата, получатель, статус прочтения
- **Telegram-пользователь:** Telegram ID, ссылка, связанные задачи и уведомления

3.10. Лицензия

Проект распространяется под лицензией **MIT License**.

Дополнение:

Цветовая схема

Рис. 2: Цветовая схема интерфейса Project-Planner

Макет интерфейса

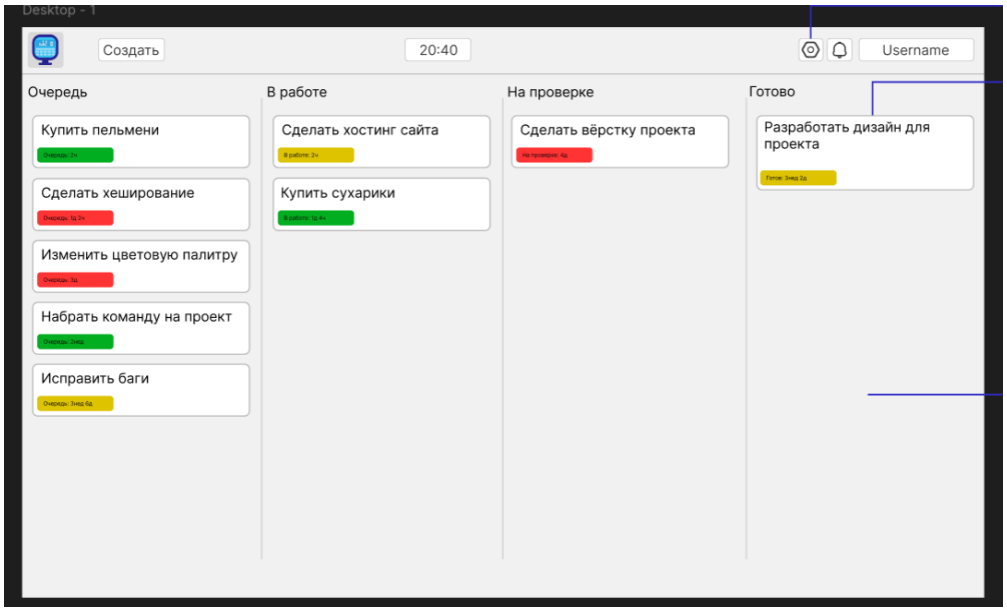

Рис. 3: Главный экран макета интерфейса

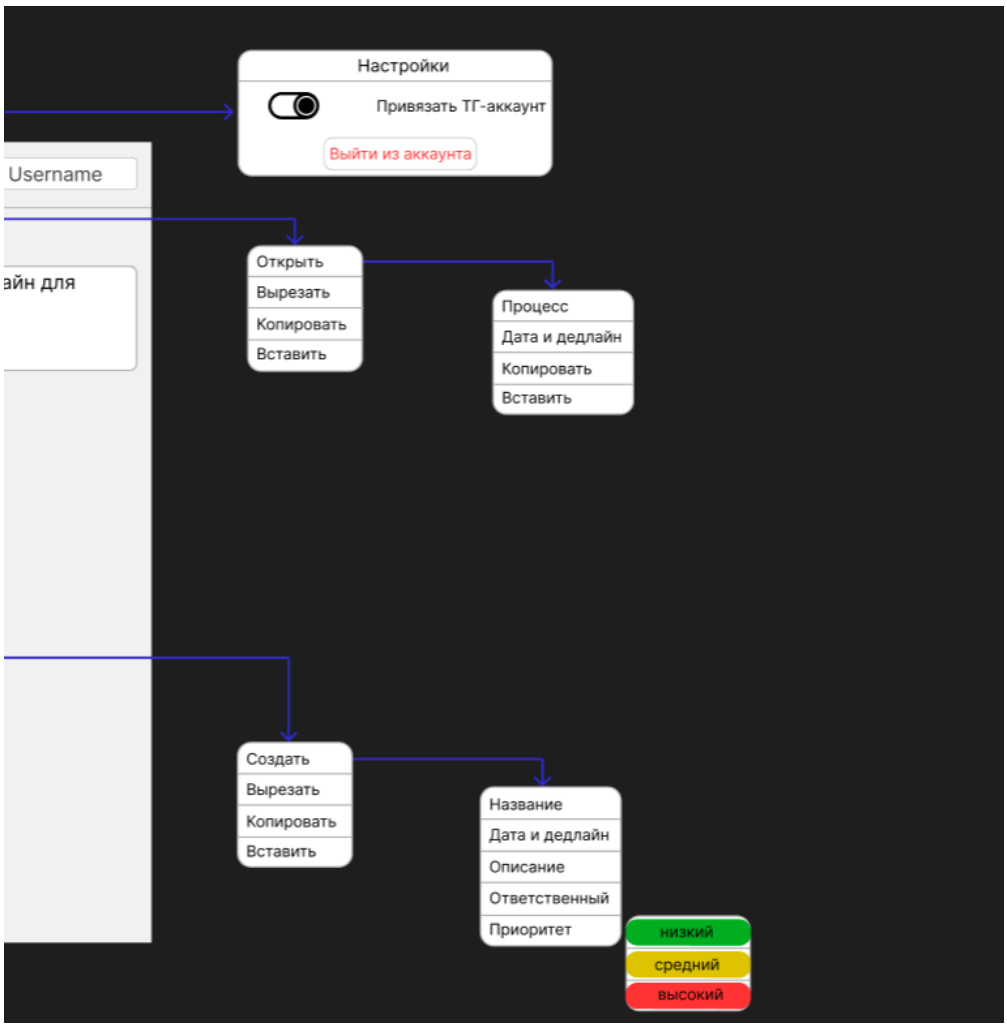

Рис. 4: Диалоговые окна