УТВЕРЖДАЮ  
Руководитель отдела информационных технологий  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Ф.И.О./  
«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ г.

**ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ ЗАДАЧАМИ**

Версия: \_\_\_\_\_\_\_\_  
Дата утверждения: \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ г.  
Разработчик: Отдел информационных технологий  
Классификация: Внутренний документ, ограниченный доступ

**1. Введение**

**1.1. Назначение документа**

Настоящий документ определяет архитектуру, компонентный состав, протоколы взаимодействия, форматы данных и функциональные требования к программной системе управления задачами (далее — СУЗ).

Документ предназначен для разработчиков, системных архитекторов, DevOps-инженеров и администраторов информационной безопасности.

**1.2. Область применения**

СУЗ предназначена для организации внутреннего процесса управления задачами, событиями и документацией в распределенной команде. Система поддерживает интеграцию с Telegram, веб-интерфейс, генерацию аналитических отчетов с использованием LLM и экспорт данных в форматах CSV/XLSX.

**1.3. Используемые стандарты и соглашения**

1.3.1. Все даты и временные метки представлены в формате ISO 8601 (UTC).

1.3.2. Имена параметров, полей, файлов и переменных — в стиле snake\_case.

1.3.3. Формат ответов API соответствует JSON:API (адаптированная реализация).

1.3.4. Все HTTP-запросы к серверу защищены токеном JWT.

1.3.5. Кодирование текста — UTF-8.

**2. Обзор архитектуры**

Система реализована в виде монолитного приложения на базе фреймворка Flask, с компонентной модульной структурой и внешними зависимостями через REST/Webhook. Архитектура включает:

* Telegram Bot — внешний клиент для взаимодействия через платформу Telegram.
* Kanban Web Interface — SPA-приложение для визуального управления задачами.
* Flask Server — центральный сервер, включающий модули:
  + - API Gateway — маршрутизация и валидация запросов.
    - Auth Module — аутентификация и авторизация.
    - Tasks Module — управление задачами.
    - Users Module — управление пользователями.
    - Export Module — формирование отчётов.
    - LLM Report Module — генерация аналитики через OpenAI API.
    - Config Manager — загрузка и валидация конфигурации.
* Data Storage — файловая система с CSV-хранилищем и Redis-кэшем.
* Внешние зависимости: Telegram Platform, OpenAI API.

Примечание: Архитектурная схема представлена в Приложении А.

**3. Конфигурация системы**

**3.1. Файл конфигурации**

Конфигурация системы определяется в файле config.yaml. Все чувствительные параметры (токены, ключи) задаются через переменные окружения и не хранятся в открытом виде.

**3.2. Основные секции**

**3.2.1. Безопасность (security.\*)**

| **Параметр** | **Тип** | **Значение по умолчанию** | **Описание** |
| --- | --- | --- | --- |
| enabled | boolean | true | Включение режима безопасности |
| validation\_method | string | "telegram\_username" | Метод идентификации: "telegram\_username" / "telegram\_user\_id" |
| session\_timeout\_hours | integer | 24 | Время жизни сессии |
| admin\_only\_endpoints | array | ["/api/users", "/api/system/\*"] | Эндпоинты, доступные только админам |
| rate\_limiting.requests\_per\_minute | integer | 100 | Ограничение запросов/мин |
| rate\_limiting.llm\_requests\_per\_day | integer | 50 | Ограничение LLM-запросов/день |

**3.2.2. Логирование (logging.\*)**

| **Параметр** | **Описание** |
| --- | --- |
| level | Уровень логирования (DEBUG, INFO, WARNING, ERROR) |
| file\_path | Путь к лог-файлу |
| retention\_days | Срок хранения логов |

**3.2.3. Производительность (performance.\*)**

* cache\_enabled: флаг включения Redis-кэша.
* cache\_ttl\_seconds: время жизни кэша.
* csv\_read\_batch\_size: размер пакетного чтения CSV.

**3.2.4. Интеграции**

* Telegram: bot\_token, webhook\_url.
* LLM: api\_key, model, timeout\_seconds, cache\_minutes.
* Сервер: host, port, ssl\_enabled, cors\_origins.

**4. Аутентификация и авторизация**

**4.1. Механизм**

Аутентификация реализована через JWT-токены, выдаваемые по результату проверки telegram\_username в файле users.csv.  
 Сессии кэшируются в Redis с TTL = session\_timeout\_hours.

**4.2. Эндпоинт аутентификации**

POST /api/telegram/auth

* Вход: telegram\_username (обязательный), full\_name (опционально).
* Выход: JWT-токен, данные пользователя, права доступа.
* Поведение при security.enabled = false: автоматическая регистрация нового пользователя.
* Поведение при security.enabled = true: только аутентификация существующих пользователей.

**4.3. Ролевая модель**

| **Роль** | **Права** |
| --- | --- |
| admin | Полный доступ: пользователи, экспорт, системные настройки |
| manager | Создание/редактирование задач, экспорт, LLM-анализ |
| member | Работа только со своими задачами (назначенные/созданные) |
| viewer | Только чтение |

**5. Структуры данных**

Все данные хранятся в CSV-файлах с фиксированным форматом. Используются следующие файлы:

**5.1. Структура данных users.csv**

* Обязательные поля:
  + telegram\_username
  + full\_name
  + role
  + is\_active
  + registered\_at
* Опциональные:
  + last\_login, email
  + department

**5.2. Структура данных tasks.csv**

* Обязательные поля:
  + task\_id, title
  + status
  + creator
  + created\_at
  + updated\_at
  + priority
* Опциональные:
* assignee
* due\_date
* completed\_at
* tags

Поле tags содержит JSON-строки

**5.3. Структура данных events.csv**

* Содержит календарные события.
* Поле participants — JSON-строка с массивом telegram\_username.

**5.4. Структура данных docs.csv**

* Поля:
  + doc\_id
  + title
  + content (или file\_path)
  + creator
  + access\_level (public/team/private)
  + version

**Примечание.** Все даты хранятся в формате YYYY-MM-DD HH:MM:SS.

**6. API-интерфейсы**

**6.1. Управление задачами**

| **Метод** | **Эндпоинт** | **Описание** |
| --- | --- | --- |
| GET | /api/tasks | Получение фильтрованного списка задач (пагинация, фильтры по статусу, назначенному, приоритету, тегам) |
| POST | /api/tasks | Создание новой задачи |
| PUT | /api/tasks/{task\_id} | Обновление задачи (только автор или назначенный) |

**6.2. Экспорт данных**

| **Метод** | **Эндпоинт** | **Формат** | **Особенности** |
| --- | --- | --- | --- |
| GET | /api/export/tasks.csv | CSV | Поддержка фильтрации и выбора колонок |
| GET | /api/export/tasks.xlsx | XLSX | Многолистовой отчёт: Tasks, Users, Events, Summary |

**6.3. LLM-анализ**

POST /api/llm/analyze/tasks

* Поддержка периодов:
  + - last\_week
    - last\_month
    - custom
* Метрики:
  + - productivity
    - bottlenecks
    - team\_performance
* Ответ в формате JSON с рекомендациями и прогнозами.
* Кэширование результата на 60 мин.

**6.4. Управление пользователями**

Только для admin:

* POST /api/users — создание нового пользователя.
* Валидация формата telegram\_username, уникальности, корректности роли.

**7. Реальное время и события**

**7.1. WebSocket-соединение**

* Адрес: wss://<host>/ws
* Авторизация: через JWT-токен в параметрах подключения.
* Поддерживаемые события:
  + TASK\_CREATED, TASK\_UPDATED, TASK\_DELETED
  + EVENT\_UPDATED
  + USER\_ONLINE / USER\_OFFLINE

**7.2. Рассылка**

Все клиенты, подписанные на каналы, получают обновления. Используется publish-subscribe паттерн.

**8. Мониторинг и надежность**

**8.1. KPI системы**

| **Метрика** | **Целевое значение** | **Метод измерения** |
| --- | --- | --- |
| Время отклика API | < 500 мс | Prometheus |
| Доступность | > 99.5% | Health-check (HTTP 200 /health) |
| Процент завершённых задач | > 80% | Анализ tasks.csv |
| Среднее время выполнения задачи | < 3 дня | Расчёт по completed\_at – created\_at |

**8.2. Логирование**

Формат лога:

**<timestamp> <level> [<module>] <сообщение>**

Пример:

**2024-01-15 10:30:15 INFO [auth] Пользователь @developer\_alex успешно аутентифицирован**

**9. Безопасность**

**9.1. Требования**

* Все данные в движении защищены TLS 1.2+.
* Все секреты передаются через переменные окружения или Vault.
* Внешние CORS-источники строго ограничены.
* Rate limiting применяется на уровне API Gateway.

**9.2. Соответствие**

Система соответствует внутренним требованиям по безопасной разработке.

**Приложение А. Архитектурная схема**

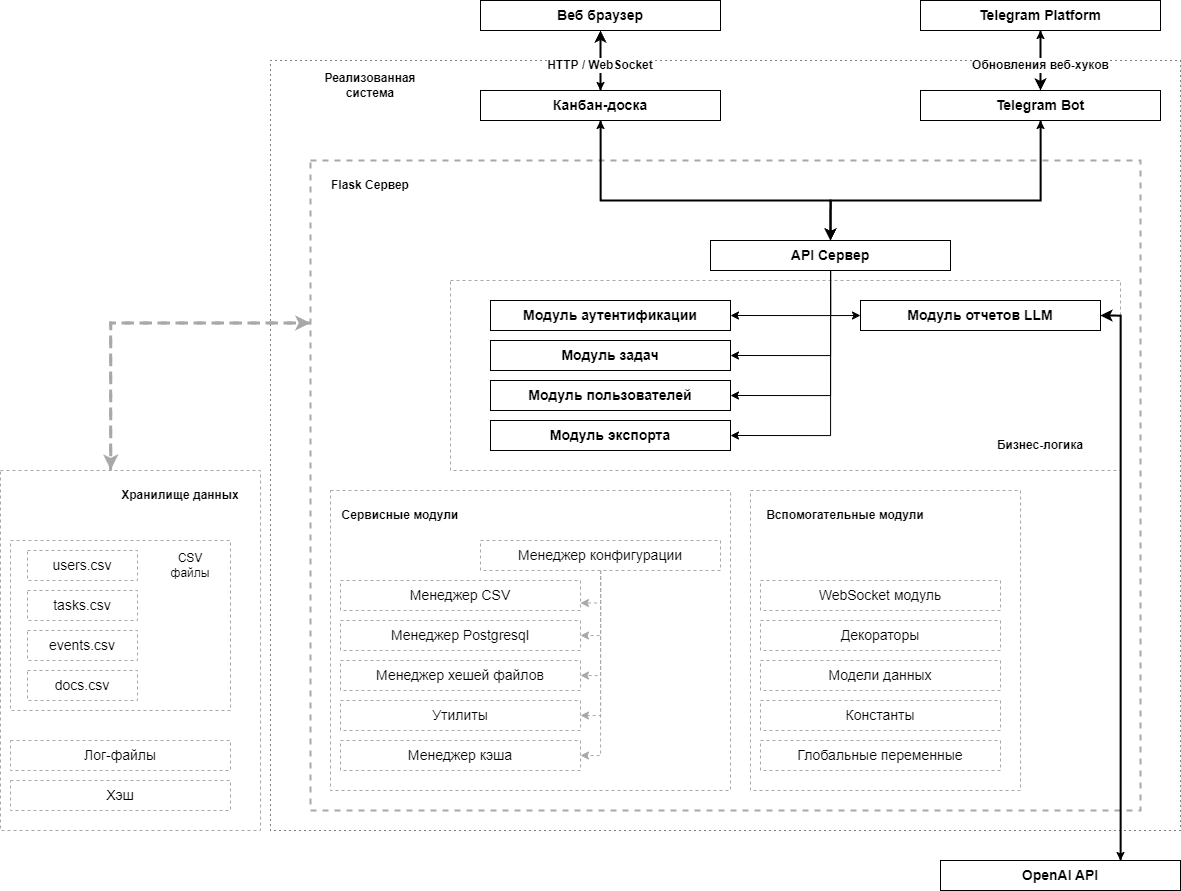


Рис. А.1 – Общая архитектура системы

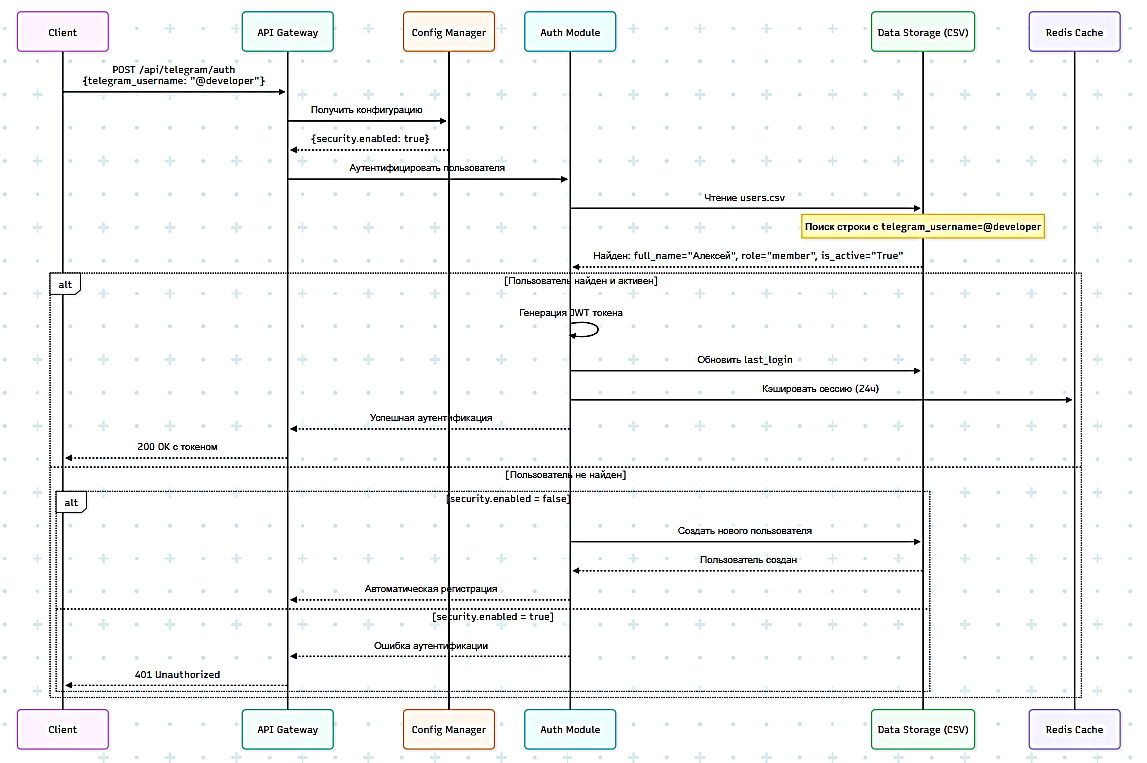
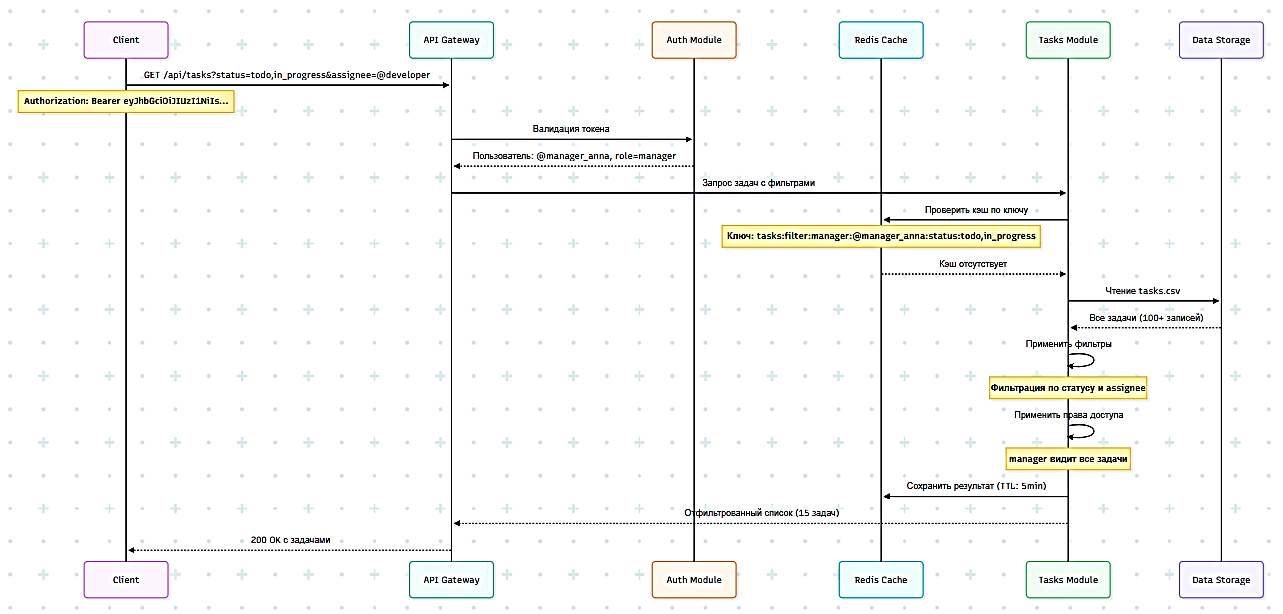


Рис. А.2 – Архитектура API аутентификации POST /api/telegram/auth

Рис. А.3 – Архитектура управления задачами GET /api/tasks

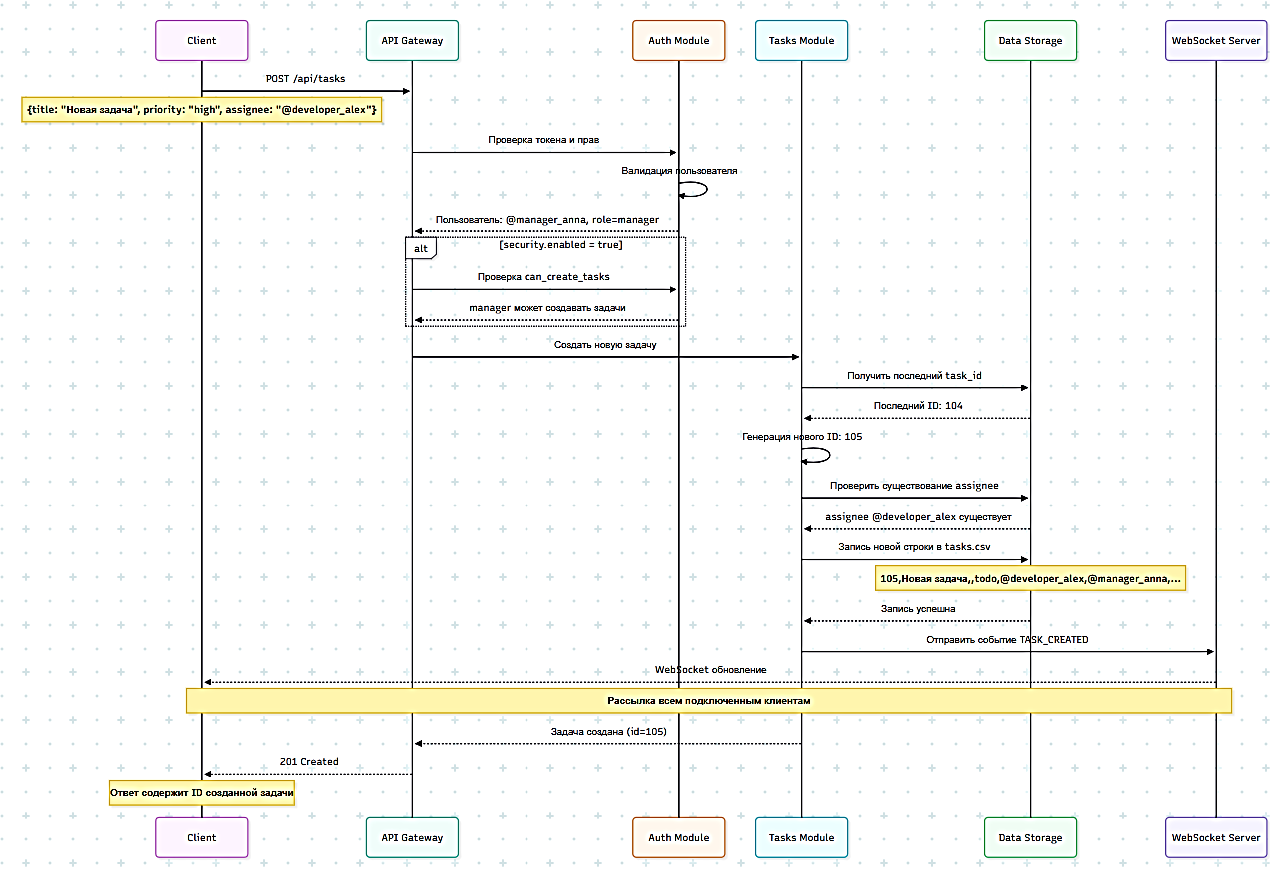


Рис. А.4 – Архитектура API управления задачами POST /api/tasks

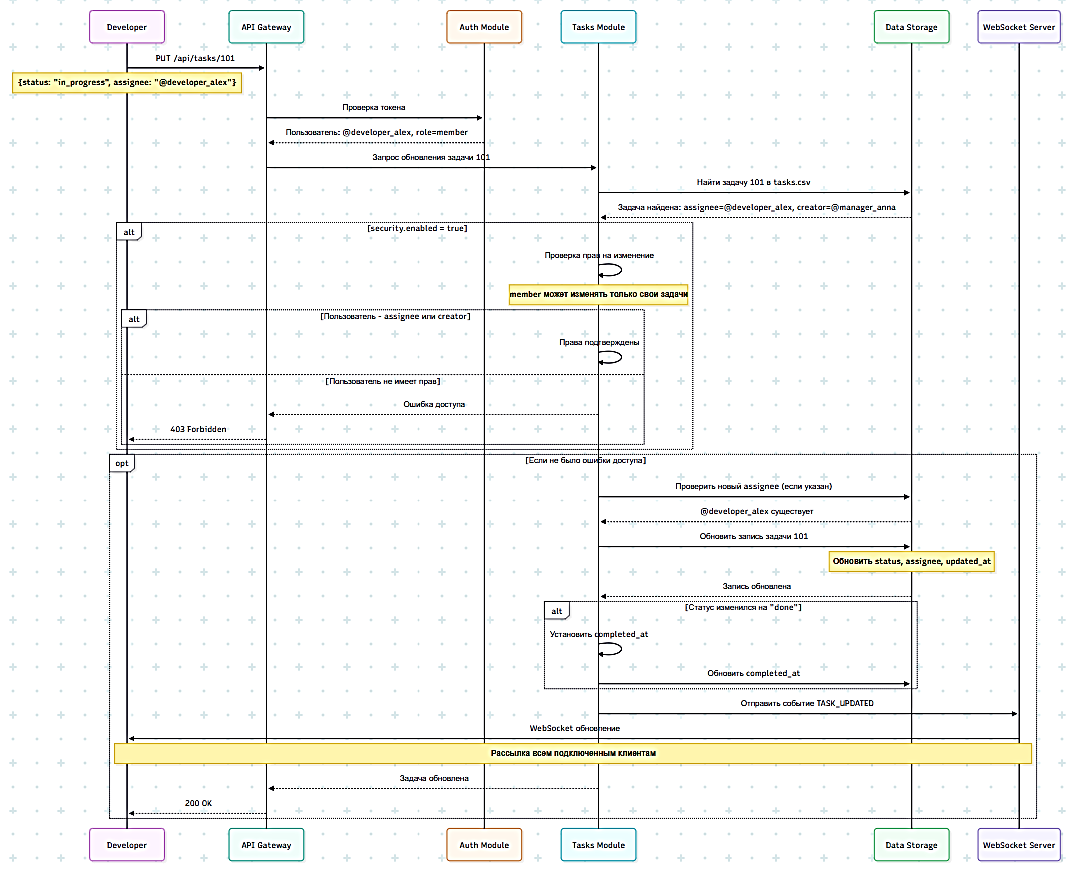


Рис. А.5 – Архитектура API управления задачами PUT /api/tasks/{task\_id}

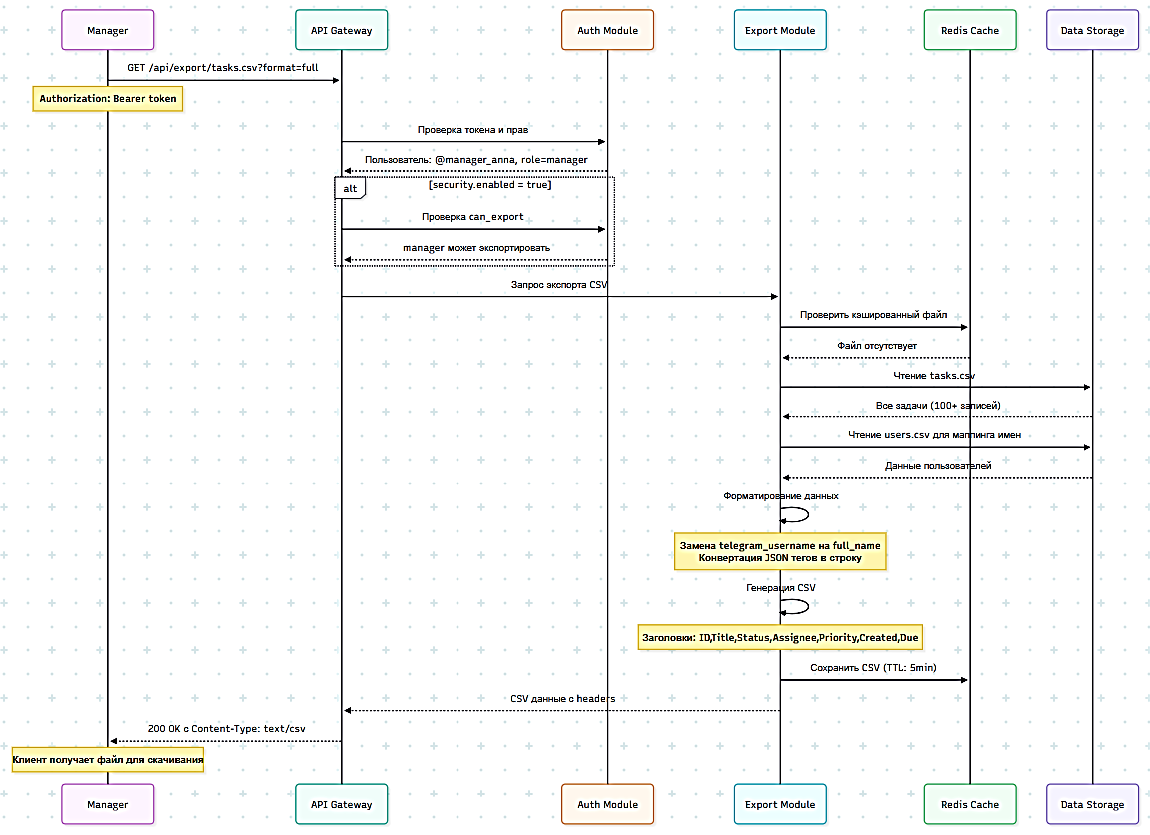


Рис. А.6 – Архитектура API экспорта данных GET /api/export/tasks.csv

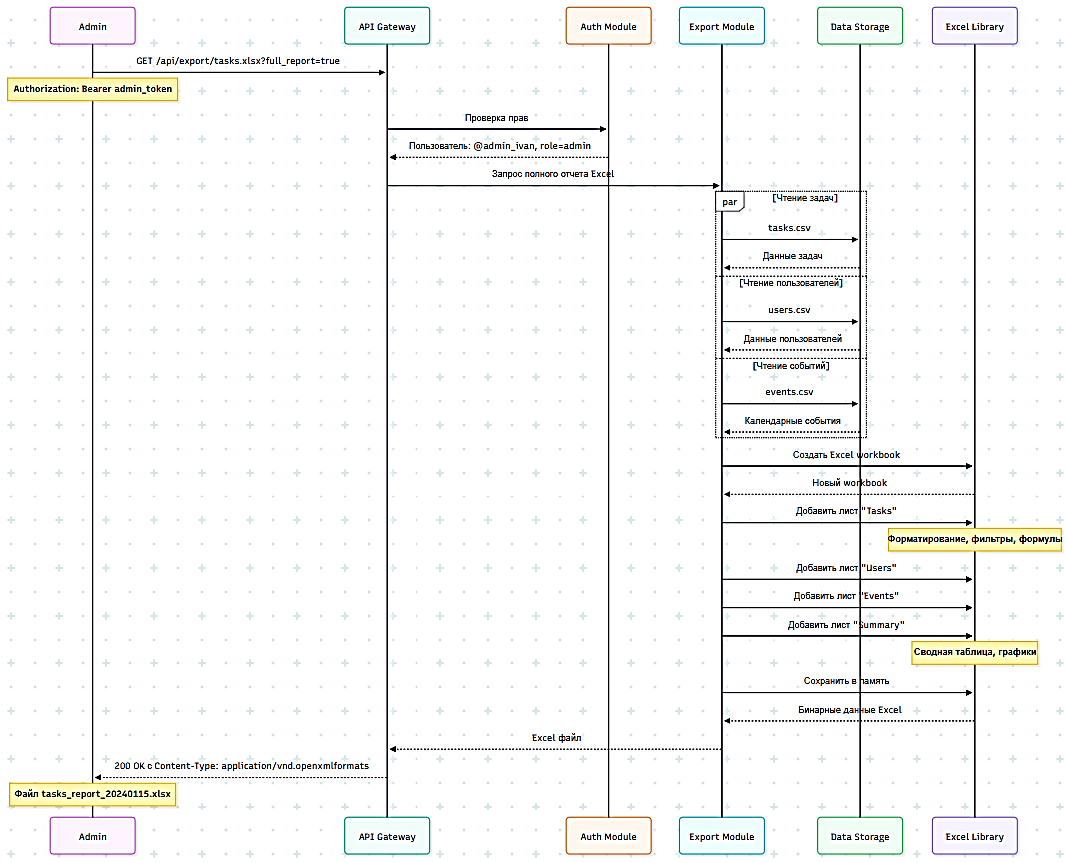


Рис. А.7 – Архитектура API экспорта данных GET /api/export/tasks.xlsx

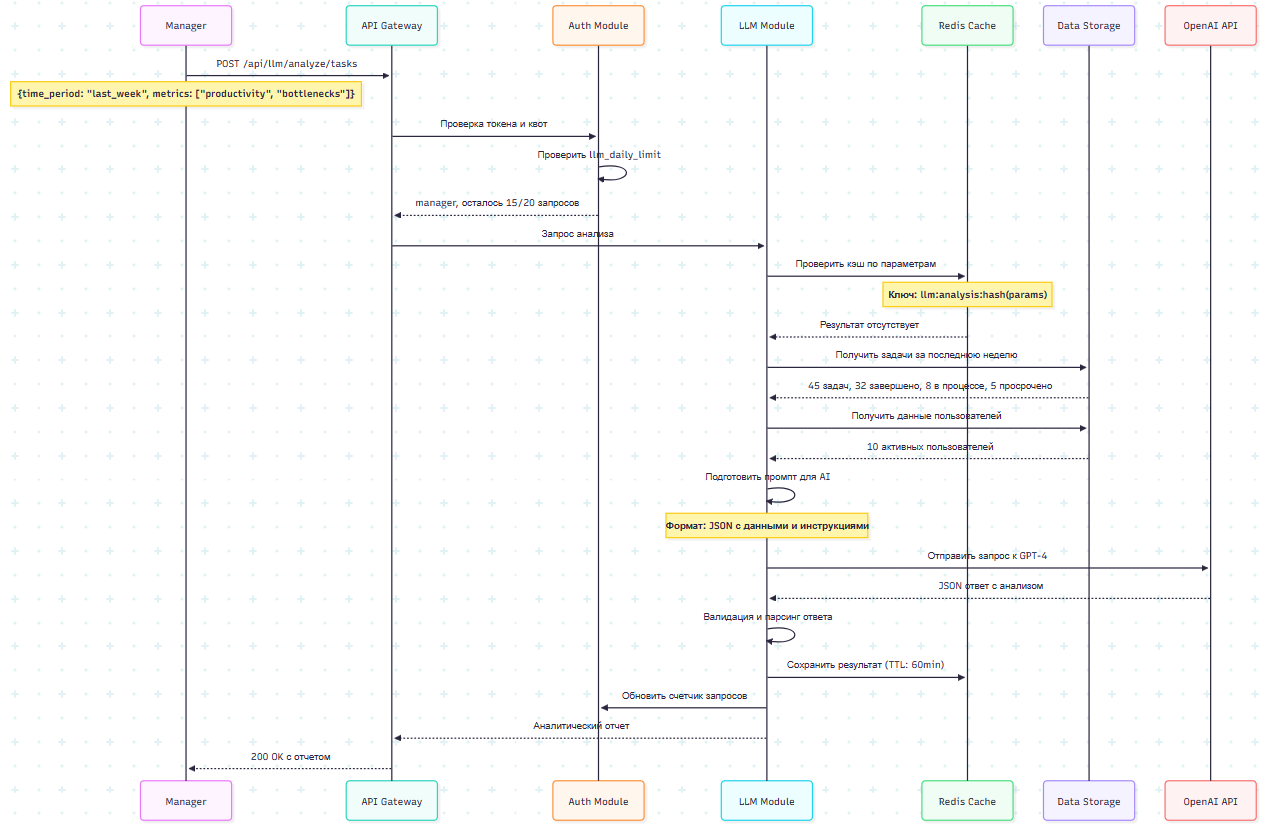


Рис. А.8 – Архитектура API LLM анализа POST /api/llm/analyze/tasks

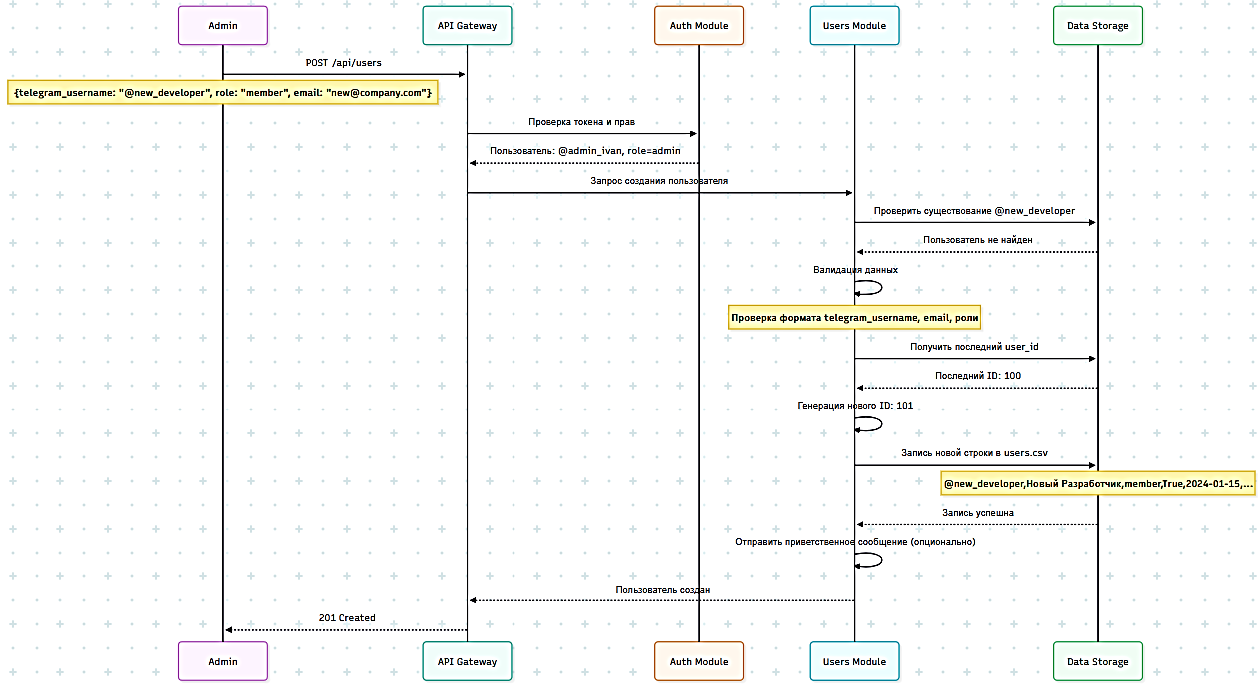


Рис. А.9 – Архитектура управление пользователями POST /api/users

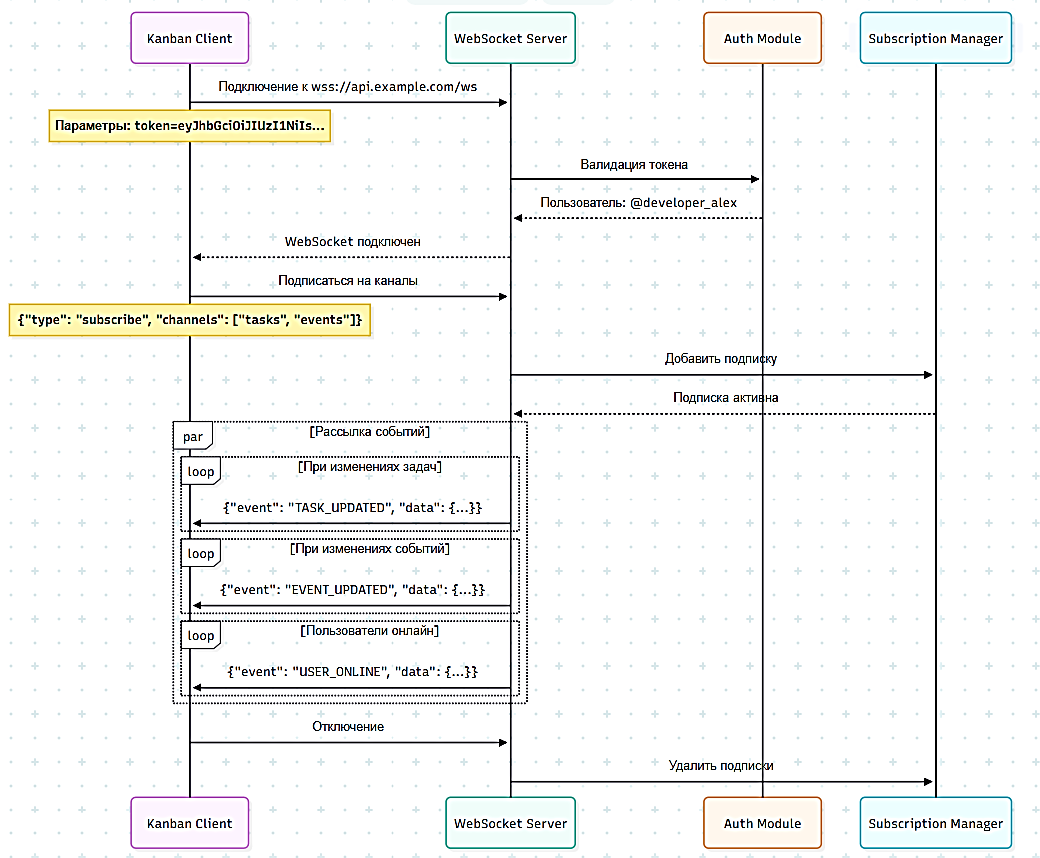


Рис. А.10 – Архитектура реальное обновление Kanban доски

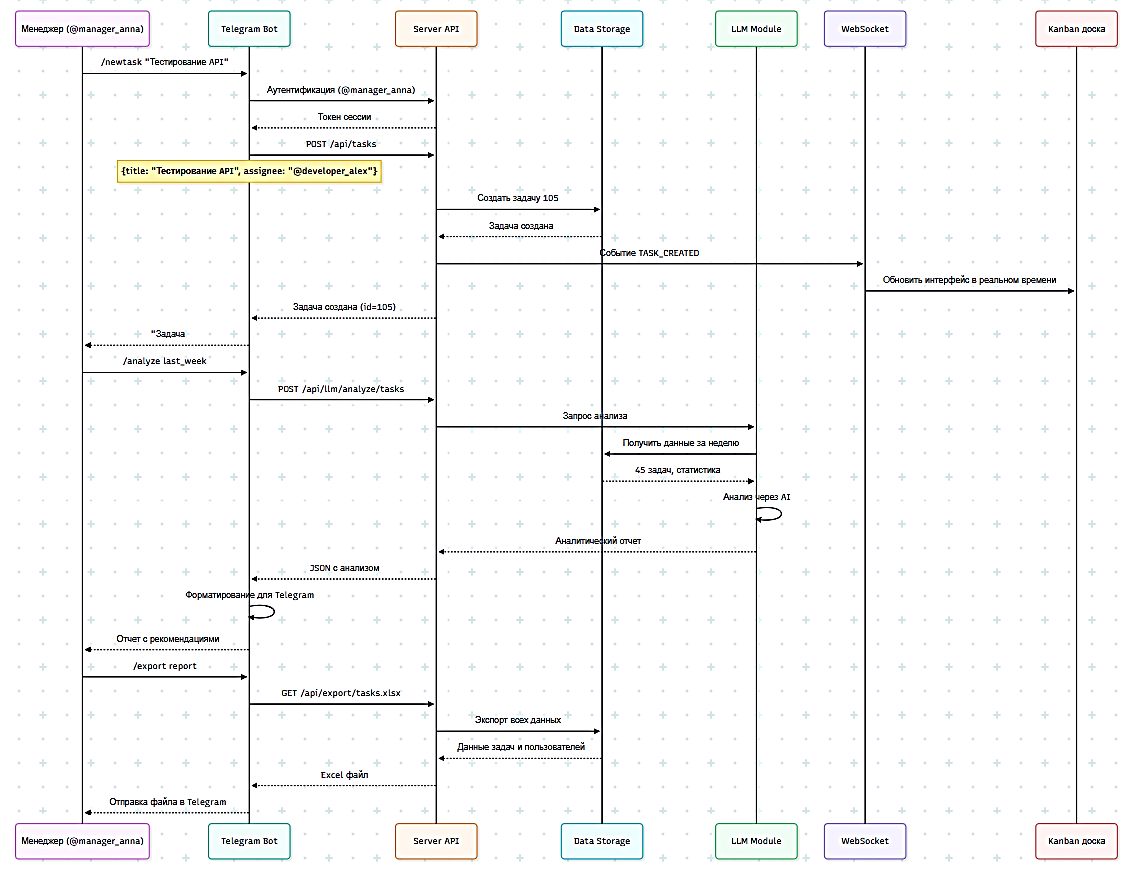


Рис. А.11 – Архитектура сценария «Менеджер создает задачу и получает AI анализ»