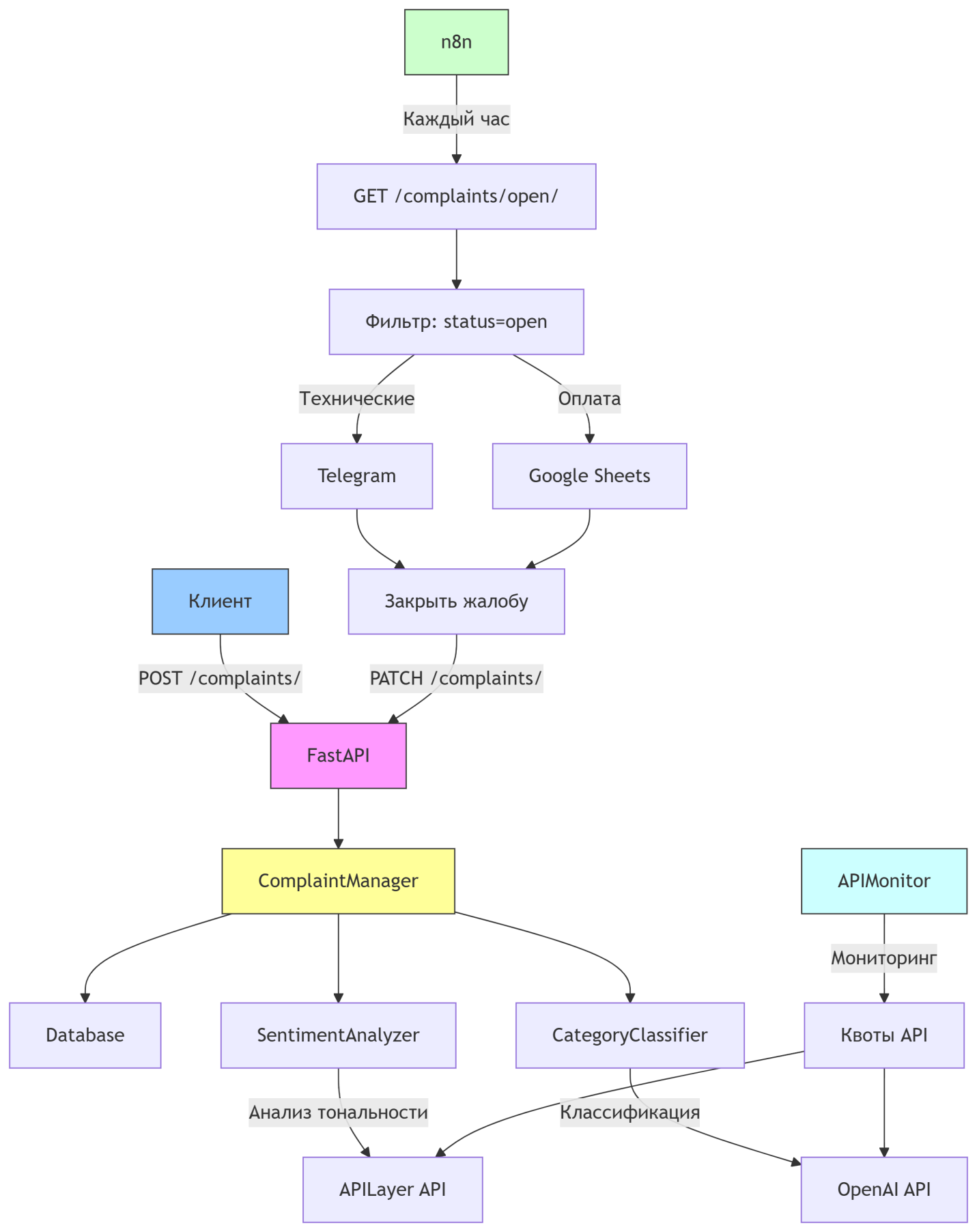
Блок-схема архитектуры системы обработки жалоб



Пояснение компонентов:

1. **Клиент**  
   Отправляет POST-запросы с жалобами в формате JSON
2. **FastAPI** (Розовый)  
   Основное приложение, обрабатывает запросы:
   * POST /complaints/ - создание жалобы
   * GET /complaints/open/ - получение открытых жалоб
   * GET /quota - проверка квот API
3. **ComplaintManager** (Желтый)  
   Управляющий класс, координирует процессы:

class ComplaintManager:

async def create\_complaint(self, text):

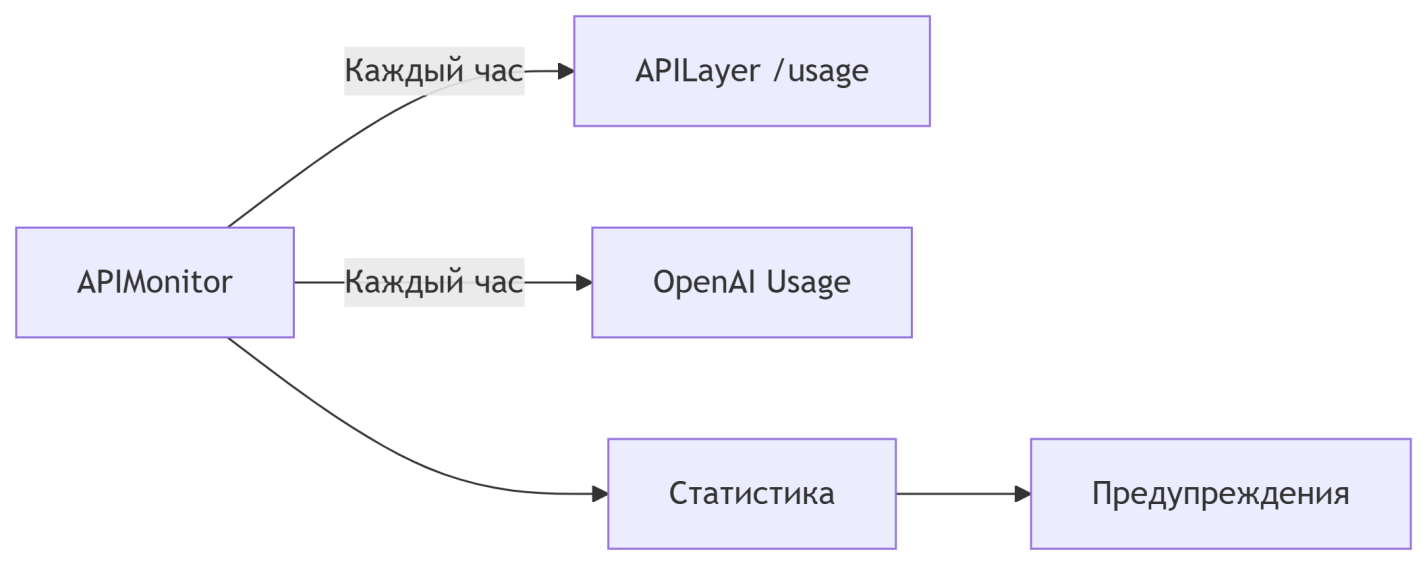
# 1. Сохранить в БД

# 2. Анализ тональности

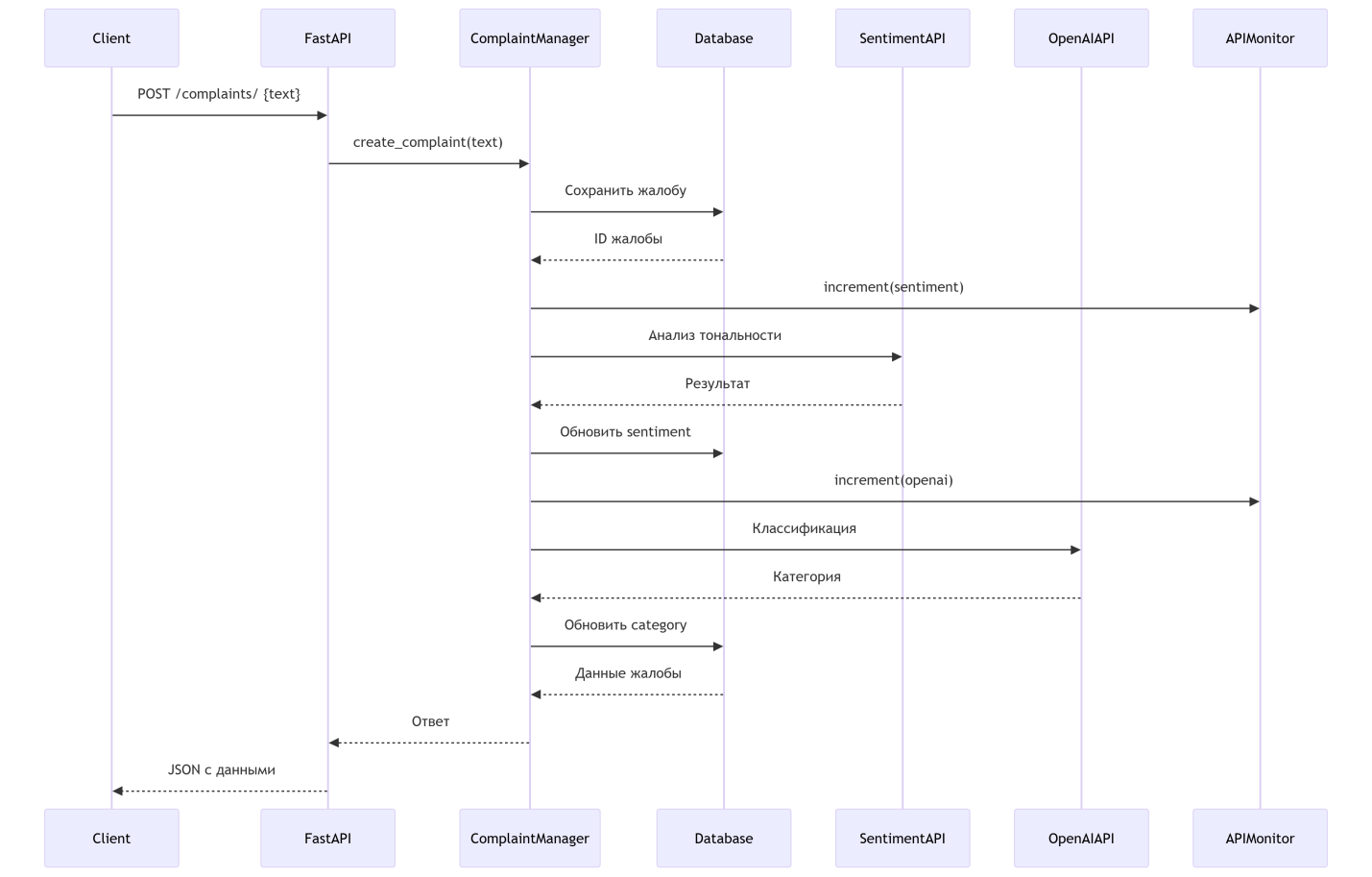
# 3. Классификация категории

# 4. Обновить мониторинг

1. **Интеграции**:
   * SentimentAnalyzer: Отправляет текст в APILayer
   * CategoryClassifier: Использует OpenAI API
   * Database: SQLite хранилище
2. **APIMonitor** (Голубой)  
   Отслеживает использование API:



**n8n Workflow** (Зеленый)  
Автоматизация обработки



### Ключевые особенности архитектуры:

1. **Централизованное управление** через ComplaintManager
2. **Асинхронная обработка**:
   * Анализ тональности выполняется синхронно с ответом
   * Классификация категории - фоновая задача
3. **Мониторинг квот**:
   * Автоматическая проверка каждый час
   * Статусы: OK/WARNING/LIMIT EXCEEDED
4. **Автоматизация** через n8n:
   * Фильтрация по статусу и времени
   * Разные действия для разных категорий
5. **Отказоустойчивость**:
   * При ошибках внешних API используются значения по умолчанию
   * Все операции логируются

Система обеспечивает полный цикл обработки жалоб - от создания до автоматического закрытия, с интеграцией внешних сервисов и мониторингом ресурсов.