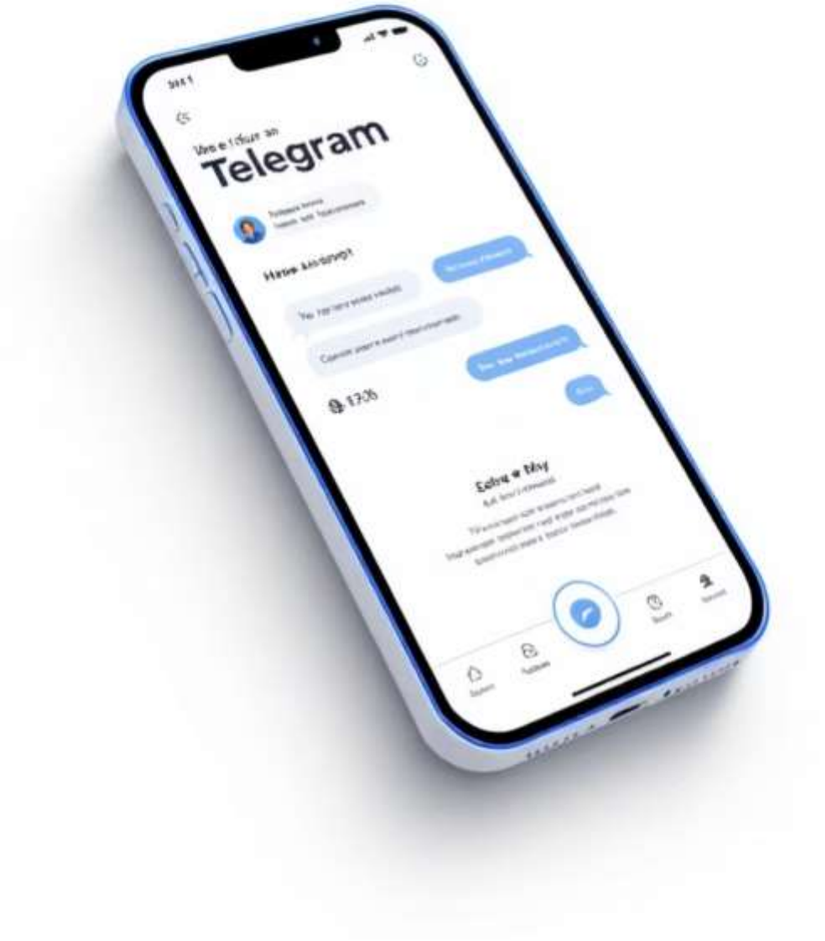
Journey Agent

Персональный ИИ-помощник для планирования выходных



⚠ ПРОБЛЕМА

Сложность планирования досуга

Современные пользователи тратят часы на поиск и организацию отдыха. Информация разбросана по десяткам источников, а процесс планирования требует учёта множества факторов.

Информационный хаос

События разбросаны по Telegram-каналам, сайтам, агрегаторам

Проблема планирования

Учёт времени, местоположения, погоды, транспорта

Когнитивная нагрузка

Ручное составление маршрута и проверка совместимости

РЕШЕНИЕ

Journey Agent: Интеллектуальное планирование с Multi-Agent системой

Персональный помощник на базе LLM автоматизирует весь цикл планирования выходных — от сбора данных до готового маршрута с учётом всех ваших предпочтений и ограничений.

Полный конвейер

От сбора данных до готового маршрута в едином workflow

Интеллектуальный RAG

Семантический поиск с самопроверкой релевантности результатов

Инструменты агентов

Интеграция с погодой, картами, веб-поиском для точности планов

Безопасность

Многоуровневая модерация на входе и выходе системы

ДАННЫЕ

Источники событий и единая модель данных

1. KudaGo API

Крупнейший агрегатор событий в России с охватом Москвы и Санкт-Петербурга. Более 1000 событий в базе: концерты, выставки, театры, лекции, мастер-классы.

- Структурированные данные с адресами, датами, ценами
- Категоризация по типам событий
- Прямые ссылки на источники

Статус: ✓ Реализовано и загружено

2. Telegram каналы

Персонализированный парсинг событий из Telegram-каналов через официальный API. Пользователи добавляют свои любимые открытые каналы через бота.

- Автоматическая синхронизация каждые 6 часов
- Event Miner Agent извлекает события через LLM

Статус: ✓ Реализовано с автосинхронизацией

Unified Event Model

```
{
 "title": "Название события",
 "description": "Описание",
 "location": "Адрес/место",
 "date": "2026-01-20",
 "time": "19:00",
 "tags": ["концерт", "музыка"],
 "url": "https://...",
 "source": "kudago / telegram",
 "owner": "user_123 / null"
}
```

Все события нормализуются в единую схему и хранятся в векторной БД Weaviate с Contextionary для семантического поиска. Персонализация через фильтрацию по owner-полю.

АРХИТЕКТУРА

Multi-Agent система и микросервисная архитектура

Два основных pipeline

Data Ingestion

Офлайн сбор данных из KudaGo и Telegram → Event Miner Agent → Vector DB

Online Serving

Telegram Bot → Guardrails → Self-RAG → Planning Graph → Response

Ключевые компоненты системы

01

Event Miner Agent

Извлечение структурированных событий через ИИ агента. Парсер данных с агрегаторов событий

02

Self-RAG Agent

Интеллектуальный поиск с проверкой памяти, извлечением города, семантическим поиском в Weaviate и оценкой релевантности. До 3 итераций переформулировки.

03

Planning Graph

Planner Agent создаёт план с инструментами (погода, карты, веб-поиск). Critic Agent валидирует по логистике, бюджету, времени. 1-2 цикла улучшения.

04

Guardrails

Двухуровневая модерация (вход/выход) через GPT-4o-mini + heuristics fallback. Уровни: allow, soft sanitization, block.

Технический стек

Backend

- Python 3.8+
- FastAPI
- Pydantic

AI/ML

- LangChain + LangGraph
- OpenAI GPT-4o
- Weaviate

Telegram

- aiogram (бот)
- Telethon (парсинг)

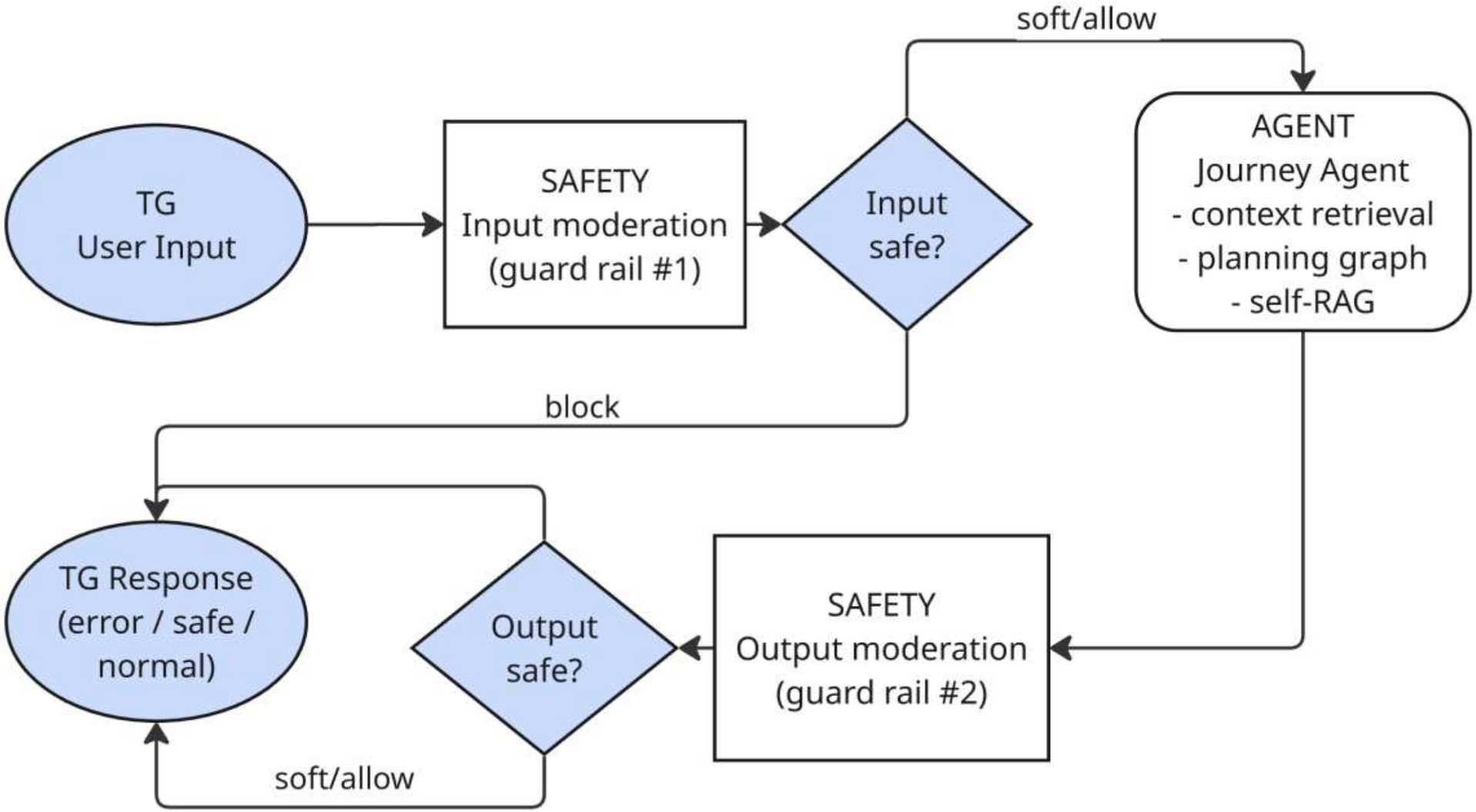
Infrastructure

- Docker Compose
- Uvicorn

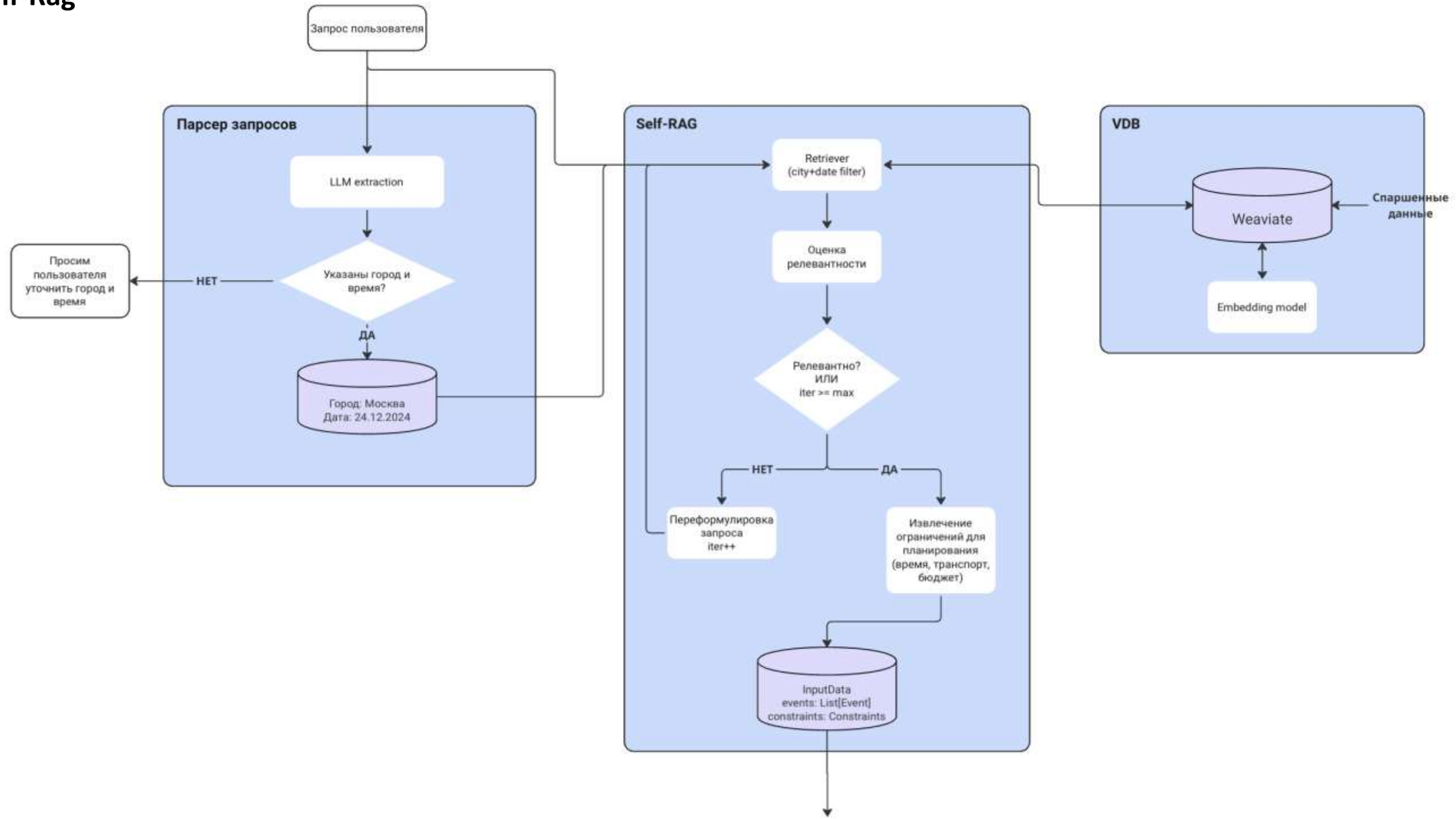
Микросервисная архитектура

5 Docker-сервисов: **weaviate** (векторная БД), **contextionary** (векторизация), **api** (REST API), **bot** (Telegram интерфейс), **sync-worker** (фоновая синхронизация каналов каждые 6 часов).

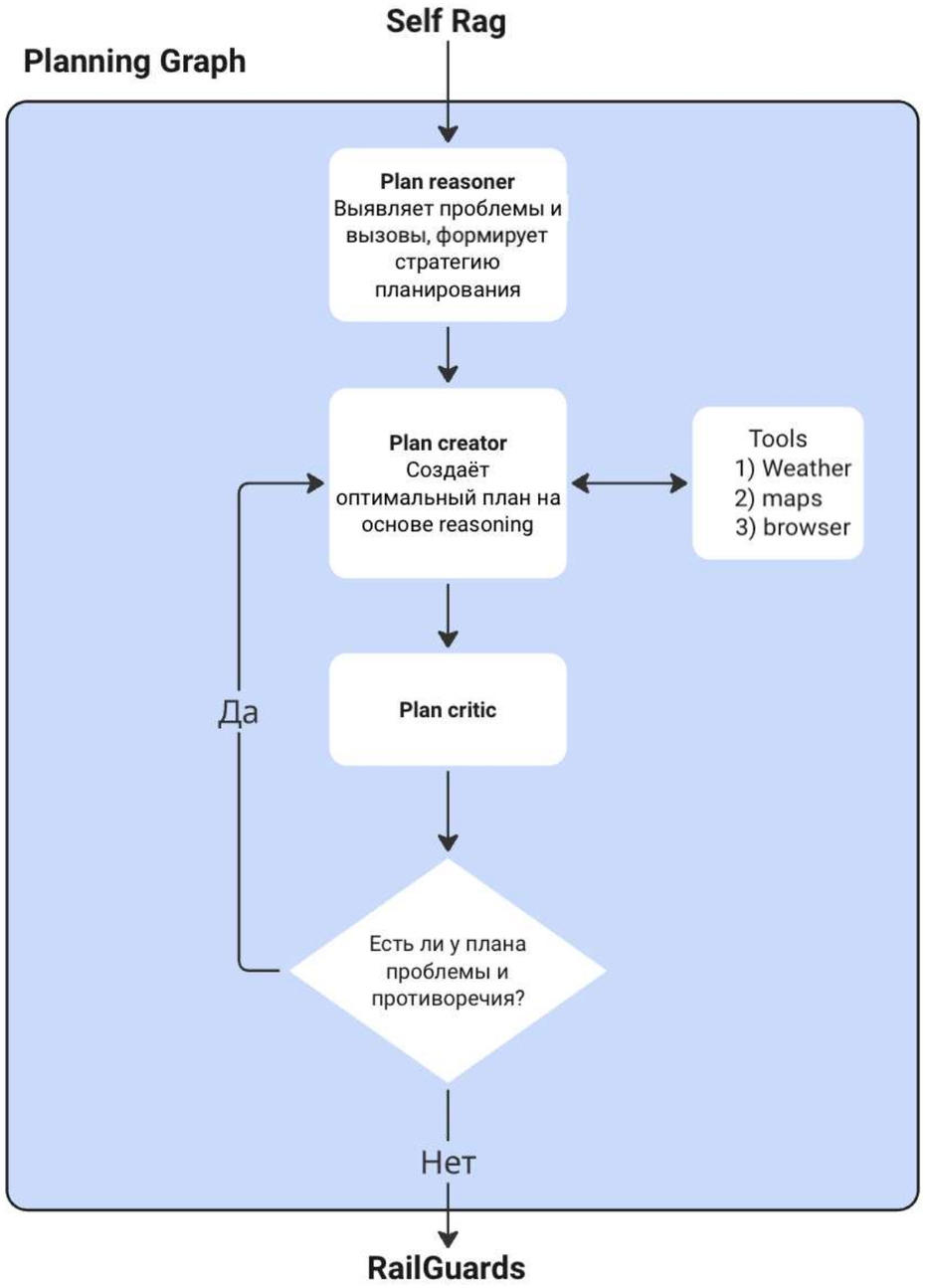
RailGuards



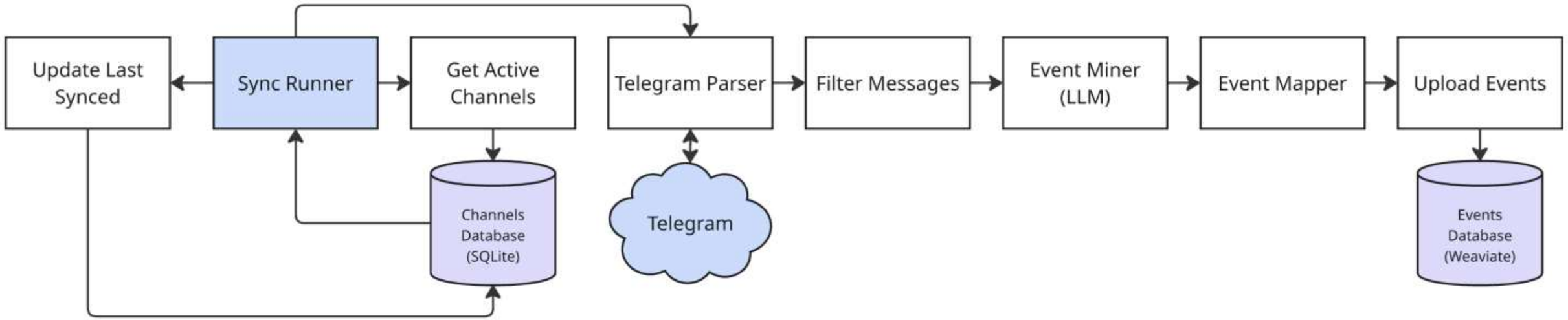
Self-Rag

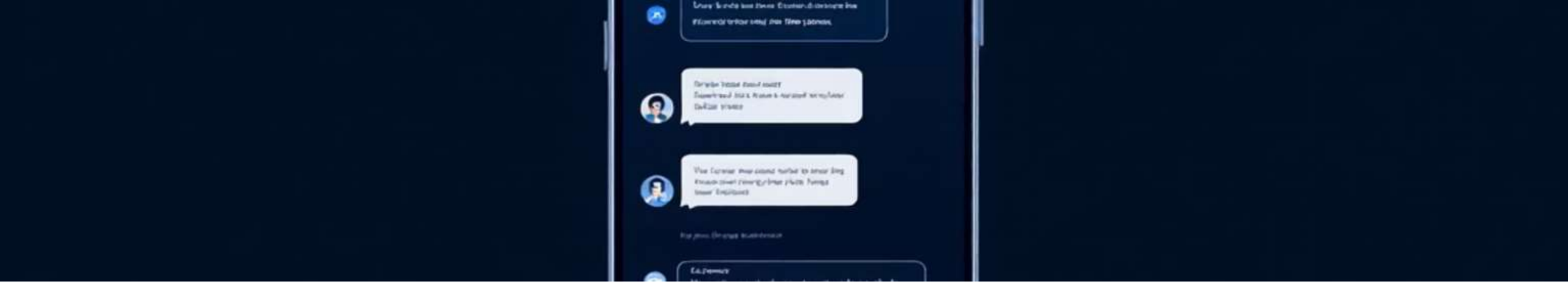


Planning Graph



Sync-worker





ДЕМОНСТРАЦИЯ

Демо системы

***также есть в github**

МЕТРИКИ

Оценка качества системы

1. Целостная оценка (LLM Judge) Три агента-оценщика анализируют финальные планы по релевантности, практичности и качеству изложения.

1 Общий балл
7.52 / 10

2 Релевантность
7.35 / 10

3 Практичность
7.68 / 10

4 Качество
7.55 / 10

2. Event Extractor (LLM Judge)

8.23

Полнота извлеченных данных

Из 10 баллов

9.53

Категоризация

Из 10 баллов

3. RAG Evaluation

0.686

MRR

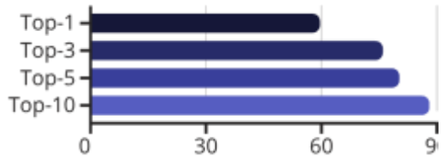
82%

NDCG@10

88%

Recall@10

Recall по позициям



4. Production метрики (LangSmith)

Стоимость

- P50: $0.0025 за запрос
- P99: $0.005 за запрос
- Avg: 15,846 токенов

Латентность

- P50: 31 секунда
- P99: 43 секунды
- LLM P50: 1.6 сек

Надёжность

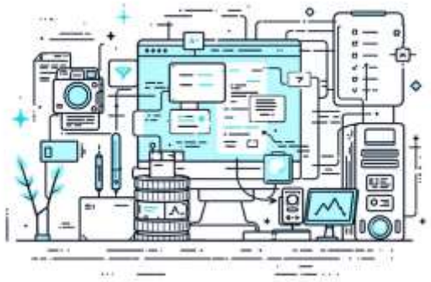
- 124 вызова LLM
- Стабильная работа

РЕЗУЛЬТАТЫ

Что получилось: ключевые достижения

Полноценный MVP

Рабочий Telegram бот с удобным интерфейсом, сквозной pipeline от данных до плана, Docker-контейнеризация.



Эффективный RAG

Recall@10: 88% — система находит релевантные события. Self-RAG с переформулировкой, персонализация через owner-фильтрацию, извлечение ограничений из естественного языка.

Multi-Agent система

LangGraph оркестровка с явными state transitions. Planner ⇄ Critic итеративное улучшение. Tool calling для погоды, карт, веб-поиска. Event Miner: 8.78/10.

Безопасность

Двухуровневая модерация (input + output), LLM-based + heuristics fallback, санитизация без блокировки.

Evaluation методология

Оффлайн метрики RAG (MRR, Recall@K, NDCG), LLM as a Judge для целостной оценки, тестовые датасеты.

Мониторинг

LangSmith интеграция для метрик латентности, стоимости, ошибок. Детальное логирование каждого шага.

Data Pipeline

Определены источники данных, настроены пайплайны обработки и оценки данных.

РАЗВИТИЕ

Будущее развитие и текущие ограничения

1

Tool calling оптимизация

Инструменты внедрены, но LLM использует их избыточно (100+ вызовов). Требуется настройка промптов и логики вызовов для снижения латентности и стоимости.

2

Расширение базы данных

Текущий объём ~1000 событий (только Москва и СПб). План: добавление других городов, парсинг дополнительных источников, краудсорсинг событий.

3

Персистентная память пользователя

Система не запоминает предпочтения между сессиями. Нужна БД предпочтений, история посещений, implicit feedback для персонализации.

4

Улучшение качества планов

Текущая оценка 5.75/10 за изложение. Требуется чёткое форматирование, временные слоты, учёт переездов, запас времени, альтернативные варианты.

5

Оптимизация маршрутов

LLM не всегда корректно понимает географию. Решение: активное использование Maps API, эмбеддинги координат, TSP solver для оптимизации.

6

Обратная связь пользователей

Нет механизма сбора feedback. План: кнопки 👍/👎, форма обратной связи, A/B тестирование промптов для измерения качества в проде.

КОМАНДА

Команда проекта

Сухов Андрей

Lead, Planning agent, Tg-bot, Metrics, Docker

Гапеева Анастасия

Railguards, Tools, Sync worker, Docker

Иоган Максим

Database, Self-Rag, Docker, Tg message parser

Бойкова Екатерина

Data, Metrics