Техническое описание архитектуры продукта

Технической описание архитектуры продукта

Версия: 1.0 **Дата:** 3 мая 2026

Источник: Ожерельев В.А. файл Техническое_описание_архитектуры.md

1. Обзор продукта

1.1 Назначение системы

Мультиагентная AI-система **Grace CRM Assistant** предназначена для автоматизации процессов продаж и повышения эффективности работы отдела продаж компании.

Система реализует цифровых AI-ассистентов (агентов), которые анализируют данные CRM, контролируют качество данных, формируют рекомендации менеджерам и обеспечивают управленческий контроль.

Система интегрируется с Grace CRM (Laravel BAP) через API и webhooks и функционирует как надстройка (advisory layer) над CRM, не нарушая принцип *CRM = source of truth*

1.2 Цели внедрения

Основные бизнес-цели системы:

- Повышение конверсии продаж

- Снижение потерь лидов
- Повышение качества данных в CRM
- Автоматизация контроля работы менеджеров
- Ускорение реакции на клиентские события
- Формирование единого корпоративного знания (knowledge layer)

1.3 Функциональные возможности

Система обеспечивает:

1.3.1 AI-ассистирование продаж

- Анализ текущего состояния сделки
- Генерация рекомендаций (следующее действие, срок, обоснование)
- Контекстный анализ на основе CRM + базы знаний

1.3.2 Контроль качества данных

- Проверка полноты карточек сделок
- Выявление пропущенных активностей
- Формирование задач менеджерам

1.3.3 Уведомления и эскалации

- Уведомления через Telegram / CRM
- Эскалация на РОП при отсутствии реакции
- SLA-контроль

1.3.4 Корпоративный поисковый контур

- Поиск по CRM-данным (OpenSearch)
- Поиск по базе знаний (RAGflow)
- Гибридный поиск (semantic + keyword)

1.3.5 Управление знаниями

- Индексация документов
- Формирование wiki-страниц
- Поиск с цитированием
- Анализ актуальности знаний

1.4 Состав системы

Система включает два функциональных контура: контур продаж и контур базы знаний.

Контур 1 -- Sales AI (Grace CRM Assistant)

Агент	Наименование	Назначение
Sync Agent	**Агент синхронизации**	Получает данные из Grace CRM через API/webhooks, обновляет PostgreSQL и инициирует индексацию в OpenSearch
Quality Agent	**Агент контроля качества данных**	Проверяет полноту и корректность данных CRM, выявляет пробелы, формирует задачи и сигналы
Recommendation Agent	**Агент рекомендаций**	Анализирует данные CRM и базы знаний, формирует конкретные рекомендации менеджерам (действие, срок, обоснование)
Notification Agent	**Агент уведомлений**	Доставляет сообщения пользователям (корпоративный мессенджер / CRM), отслеживает статус реакции
Orchestrator	**Оркестратор (LLM-слой)**	Управляет агентами, приоритизацией задач, SLA, маршрутизацией и эскалациями

Контур 2 -- Knowledge AI (база знаний)

Агент	Наименование	Назначение
Ingest Agent	**Агент загрузки знаний**	Принимает документы, структурирует их, создает/обновляет сущности базы знаний и инициирует индексацию
Query Agent	**Агент поиска знаний**	Выполняет поиск (семантический + полнотекстовый) и формирует ответы с ссылками на источники
Lint Agent	**Агент аудита знаний**	Анализирует базу знаний, выявляет противоречия, устаревшие данные и логические разрывы

1.5 Ключевые принципы

- Grace CRM является основным источником операционных данных (source of truth)
- AI-система выполняет advisory-функцию (рекомендации, контроль)
- Все изменения в CRM происходят только через API
- Архитектура построена на слабой связанности (loosely coupled services)
- Используется событийная модель (webhooks + polling)
- Система разворачивается on-premise

Дополнительно:

- Система предоставляет **собственные API-интерфейсы**, позволяющие в будущем:
  - интеграцию с внешними системами (ERP, 1C, BI и др.)
  - доступ к данным OpenSearch (статистике) и базе знаний
  - получение AI-сигналов и рекомендаций
- Архитектура допускает расширение через внешний API-layer (integration gateway)

Требования к LLM-слою

- Оркестратор (LLM) должен поддерживать:

    - подключение различных провайдеров (OpenAI, GigaChat, YandexGPT и др.) или использование собственной (on-premise) LLM
    - возможность замены модели без изменения бизнес-логики
- Конфигурация должна включать:
    - выбор модели
    - параметры генерации (temperature, max tokens и др.)
    - routing запросов (multi-model strategy)

Коммуникационный слой

- Уведомления отправляются через:
    - **корпоративный мессенджер (например: Telegram / Mattermost)**
    - интерфейс CRM
- Конкретная платформа мессенджера выбирается на этапе внедрения
- Архитектура предусматривает абстракцию над каналом доставки (adapter pattern)

---

2. Архитектура решения

2.1 Общая архитектурная модель

Система построена по принципу **мультиагентной архитектуры с централизованной оркестрацией**:

- Оркестратор (LLM) управляет логикой
- Агенты выполняют специализированные функции
- Данные хранятся в нескольких специализированных слоях
- Интеграция осуществляется через API и webhooks

2.2 Основные компоненты

Система использует многослойную модель хранения данных, где каждый слой оптимизирован под свой класс задач.

Источники данных

- Grace CRM (основная система)
- Внешние документы (для базы знаний)

Слой данных

- PostgreSQL -- операционные данные
- OpenSearch -- аналитика
- RAGflow -- база знаний и семантический поиск

AI-слой

- Orchestrator (LLM)
- Набор агентов (Sales + Knowledge)

Интеграционный слой

- REST API (CRM ↔ AI)
- Webhooks (AI ↔ CRM)
- Telegram Bot API/Messenger API

2.3 Схема взаимодействия систем

Логическая схема взаимодействия:

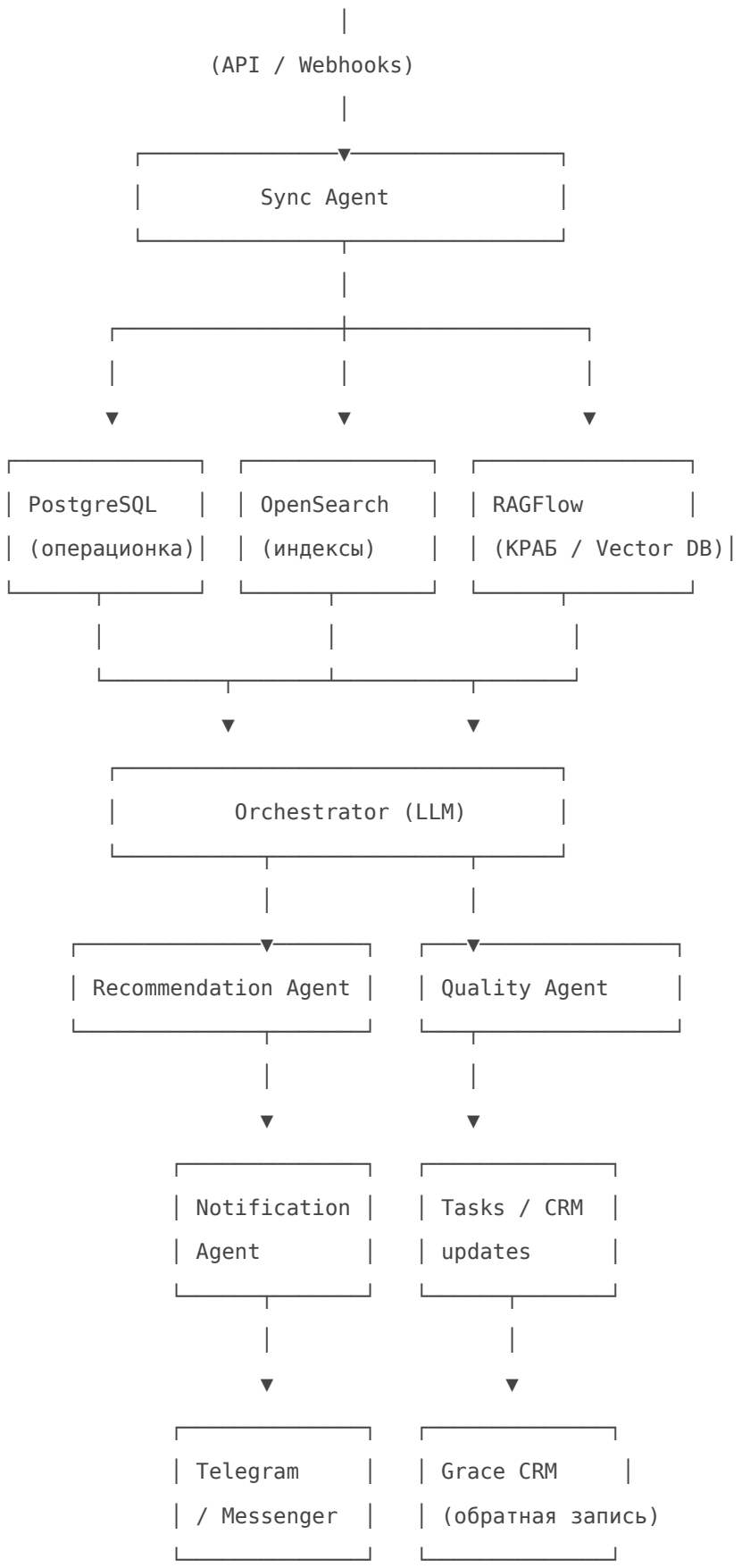
(API / Webhooks)
Sync Agent
PostgreSQL (операционка)
OpenSearch (индексы)
RAGFlow (КРАБ / Vector DB)
Orchestrator (LLM)
Recommendation Agent
Quality Agent
Notification Agent
Tasks / CRM updates
Telegram / Messenger
Grace CRM (обратная запись)

Схема взаимодействия систем (Mermaid)**

> *(диаграмма `tekhnicheskoy-opisanie-arkhitektury-produkta.mermaid` отсутствовала в исходнике Gramax — восстановить)*

2.4 Потоки данных (основной сценарий)

1. Синхронизация

- Sync Agent получает данные из CRM
- Обновляет PostgreSQL
- Обновляет индексы OpenSearch (webhook + polling)

2. Анализ

- Quality Agent проверяет данные
- Recommendation Agent формирует рекомендации
- Используются:
  - OpenSearch (история)
  - Vector DB (знания)
  - PostgreSQL (операционка)

3. Оркестрация

- Orchestrator:
  - управляет приоритетами
  - контролирует SLA
  - инициирует эскалации

4. Доставка

- Notification Agent:
  - отправляет сообщения (Telegram / CRM)
  - фиксирует статус

5. Обратная запись

- AI записывает:
  - рекомендации
  - задачи
  - активности в CRM

2.5 Архитектурные особенности

- Событийная модель:
  - primary: webhooks
  - secondary: polling (5–10 мин)
  - fallback: reindex
- Разделение хранения:
  - OLTP -> PostgreSQL
  - Search -> OpenSearch
  - Semantic -> Vector DB (RAGFlow)
- Масштабируемость:
  - агенты независимы
  - можно добавлять новые роли

---

3. Описание компонентов решения

--

3.1 PostgreSQL -- операционный слой (OLTP)

PostgreSQL используется как **основное операционное хранилище AI-системы**, обеспечивающее консистентность и управление внутренним состоянием.

Назначение:

- хранение служебных данных системы
- фиксация состояния агентов
- хранение очередей задач и событий
- управление lifecycle рекомендаций

Типы данных:

- статус обработки сущностей (sync state, offsets)
- задачи и сигналы (quality issues, SLA events)
- рекомендации (черновики, статусы, feedback)
- логи работы агентов
- настройки системы и пользователей

Почему PostgreSQL:

- транзакционность (ACID) -- критично для согласованности
- удобство работы с реляционными структурами
- поддержка сложных запросов и join-логики
- надежность для хранения состояния системы

Роль в архитектуре:

➡ System of record для AI-слоя (но не для бизнес-данных CRM)

3.2 OpenSearch -- поисковый и аналитический слой

OpenSearch используется как **унифицированный слой быстрого доступа к данным CRM и аналитики**.

Назначение:

- полнотекстовый поиск по CRM-данным
- агрегации и аналитика
- построение дашбордов
- быстрый доступ к денормализованным данным

Типы данных:

- сделки (projects)
- активности (activities)
- клиенты (clients)
- рекомендации (индекс для аналитики)

Особенности:

- данные поступают из CRM через Sync Agent
- хранятся в виде индексов (denormalized views)
- обновляются через:
  - webhooks (primary)
  - polling (fallback)
  - reindex (recovery)

Почему OpenSearch:

- высокая скорость поиска и агрегаций
- масштабируемость
- нативная поддержка аналитики (Dashboards)
- возможность построения BI без отдельного DWH

Роль в архитектуре:

➡ Search & Analytics Layer (read-heavy слой)

3.3 RAGFlow -- база знаний (Knowledge Layer)

Назначение:

- хранение корпоративных знаний
- семантический поиск
- генерация контекстных ответов

Дополнительно:

- поддержка RAG
- расширение графовой моделью (knowledge graph)

➡ Semantic Knowledge Layer

3.4 Итоговое разделение ответственности слоя данных

Слой	Система	Роль
Операционный	PostgreSQL	Состояние системы и агентов
Поисковый	OpenSearch	Быстрый доступ и аналитика CRM
Семантический	RAGFlow	Знания и контекст для AI

---

4. База знаний (компактное описание)

4.1 Какие документы входят в базу знаний

База знаний -- это:

структурированная модель опыта продаж компании + отраслевой контекст, а не просто хранилище документов.

База знаний должна содержать строго структурированные типы контента:

1. **Продуктовые материалы** Описания продуктов, УТП, ограничения, сценарии применения
2. **Кейсы продаж (ключевой блок)** Реальные сделки: контекст, действия менеджера, результат
3. **Скрипты и best practices** Шаблоны коммуникаций, стратегии ведения сделки
4. **Возражения и ответы** Типовые возражения клиентов и эффективные способы их обработки
5. **Отраслевые материалы** Боли, особенности и контекст различных отраслей
6. **Конкурентный анализ** Сравнение с конкурентами, аргументация
7. **Регламенты и процессы** Правила работы, этапы воронки, SLA
8. **Ошибки и анти-паттерны** Причины провалов сделок и нежелательные сценарии
9. **FAQ / быстрые ответы** Короткие стандартизированные формулировки

4.2 Что ищут менеджеры

Поисковые сценарии делятся на 4 типа:

Ситуационные

- что делать на текущем этапе сделки
- следующий шаг / действие

Контекстные

- как работать с конкретной отраслью или типом клиента

Поведенческие

- как обработать возражение
- как ускорить или закрыть сделку

Тактические

- примеры писем / сообщений
- формулировки и аргументы

4.3 Структура базы знаний

1. Иерархия

- Блок (например: Кейсы, Продукты)
- Подблок (например: Отрасль / Тип клиента)
- Документ (конкретная страница)

---

2. Метаданные (обязательные)

Каждый документ должен содержать:

- отрасль
- тип клиента
- стадия сделки
- продукт
- теги (возражения, сценарии и др.)

---

3. Граф связей (knowledge graph)

В рамках данной архитектуры база знаний дополнительно **структурируется с использованием графовой модели (граф БД / knowledge graph)** для этого используется логическая графовая модель внутри RAGFlow:

- связи между документами
- связи между сущностями (клиенты, отрасли, кейсы)
- контекстные зависимости

Пример связей между сущностями:

- кейсы ↔ продукты
- кейсы ↔ отрасли
- возражения ↔ ответы
- клиенты ↔ поведенческие паттерны

4.4 Как это использует AI

Recommendation Agent:

- ищет похожие кейсы
- извлекает успешные паттерны
- формирует действие

Quality Agent:

- проверяет наличие обязательных шагов

Orchestrator:

- учитывает контекст + знания

Revision #1
Created 2026-06-22 03:22:47 UTC by Claude Ops
Updated 2026-06-22 03:22:47 UTC by Claude Ops