

# С

# Global ERP

Трансформация корпоративной платформы  
для эпохи **автономных AI-агентов**

## Аудитория

Стейкхолдеры

Технические архитекторы

Руководители разработки

## Ключевые технологии

MCP Protocol

Agent Runtime

GenUI

ML-Platform

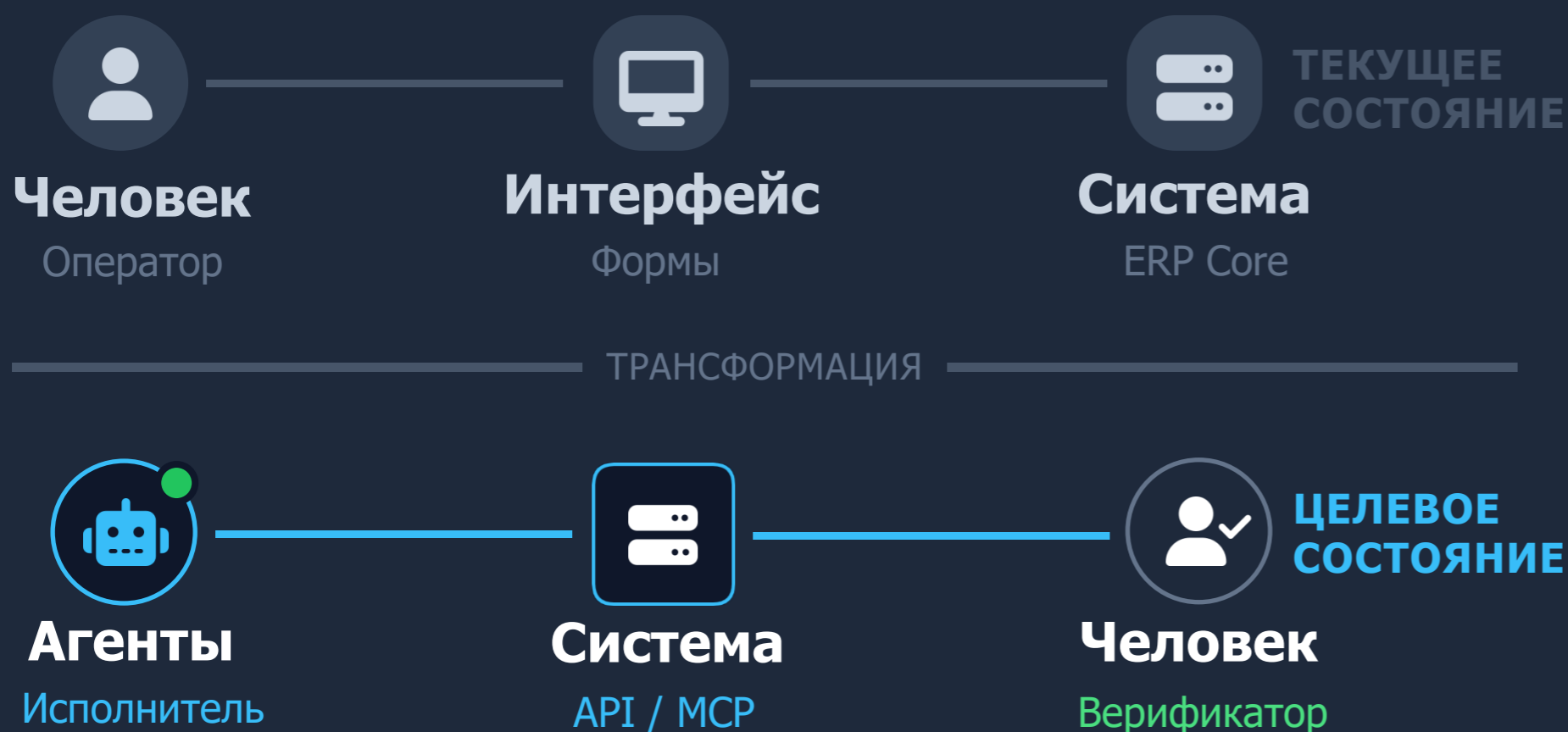
• РАЗДЕЛ 01 • КОНТЕКСТ И ВЫЗОВ

# Зачем: Проблемы текущей модели и ценность трансформации

Анализ ограничений существующих формоцентричных интерфейсов (UI 2.0) и обоснование необходимости перехода к интент-ориентированной архитектуре на базе автономных AI-агентов.

# Сдвиг парадигмы: от форм к агентам

## Модель Взаимодействия



## Проблемы пользователей

Проблема	Источник
☰ Меню — сотни пунктов, линейному пользователю нужны 5 интерфейсов	Внедренец
📍 Навигация утомляет при частом открытии одной выборки	Разработчик
↔ Переключение между приложениями — долго, потеря контекста	Разработчик
🔊 Перегруженность интерфейсов: «белый шум», сложно ориентироваться	Внедренец
🖱️ Глубокая вложенность — слишком много кликов (10-15 на типичную операцию)	Внедренец
📄 Нет единого UX-гайдлайна — разные экраны ведут себя по-разному	Внедренец

## Рыночный Контекст

AI-Copilots в ERP

**SAP Joule, Oracle AI, MS Copilot**

Новый Стандарт

**Model Context Protocol (MCP)**

Тренд 2026

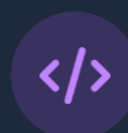
**Автономные агенты выполняют задачи, а не просто отвечают на вопросы**

# Бизнес-ценность Agent Ready



## Для бизнеса

- ✓ Автоматизация рутины **без программирования** (документооборот, сверки)
- ✓ Сокращение времени обучения: агент знает как работать с системой
- ✓ Новый класс интеграций: внешние агенты (контрагенты) через MCP



## Для разработки

- ✓ Агентская разработка: автономное выполнение тикетов с контекстом
- ✓ Семантическое ODM авто-обогащает AI-контекст без ручной документации
- ✓ Прикладные агенты создаются **аналитиками** без Java/Scala



## Для платформы

- ✓ Позиционирование Global3 как **Agent Ready ERP** — конкурентное преимущество
- ✓ MCP-совместимость открывает экосистему внешних AI-инструментов
- ✓ Фундамент для GenUI — следующего поколения интерфейсов

## ВЫИГРЫШИ ПО РОЛЯМ

### HR-Рекрутер



Задача: Сбор утренней сводки

**Авто-подготовка (~20 мин экономии ежедневно)**

### Кладовщик



Задача: Инвентаризация и сверка

**Агент формирует акты, человек только проверяет**

### Снабженец



Задача: Мониторинг остатков

**Автозаявки при снижении порога без ручного контроля**

### Разработчик



Задача: Выполнение тикетов

**Агент пишет код, создает MR, итерирует по ревью**

# 3 парадигмы UI

ПРОШЛОЕ  
ДАнные В ЦЕНТРЕ



## UI 1.0

### Витрина Базы Данных

- Справочники и Транзакции как прямое отражение таблиц БД
- Пользователь должен знать модель данных системы
- Сотни пунктов меню, сложная навигация

#### Ключевая проблема

"Чтобы получить отчет, нужно знать где лежат данные"

НАСТОЯЩЕЕ  
ЗАДАЧИ В ЦЕНТРЕ



## UI 2.0

### Сценарии Работы

- Фокус на цели пользователя, скрытие сложности БД
- Современный UX, дашборды, виджеты

#### Фундамент (2025)

- ✓ Фейслифтинг: чистый стиль
- ✓ Рабочий центр (Ctrl+K)

БУДУЩЕЕ  
НАМЕРЕНИЯ В ЦЕНТРЕ



## UI 3.0

### Генеративный ИИ

- ✓ Агент строит UI под запрос пользователя в рантайме
- ✓ ERP как сервис (No-UI): управление через намерения
- ✓ Generative UI: интерфейс создается "на лету"

#### Три составляющих

1. UI 2.0 Стабильность
2. GenUI Динамика
3. AI Intent Управление

# UI 3.0: Интент-ориентированный интерфейс

## СЕЙЧАС (UI 2.0)

Ручной процесс

### Задача:

#### Найти просроченные заявки

1. Открыть меню
2. Найти раздел "Заявки"
3. Открыть выборку
4. **Настроить фильтр**
5. Выбрать поле "Дата"
6. Задать условие "> 01.03"
7. Выбрать поле "Статус"
8. Задать "Просрочена"
9. Нажать "Применить"
10. Просмотреть результат

**10-15**

кликов мыши

## БУДУЩЕЕ (UI 3.0)

Процесс AI-агента

### Задача:

#### Найти просроченные заявки

 Пользователь

"Покажи просроченные заявки за март"

 Агент анализирует...

 **Агент**

Нашел 12 заявок. Фильтр применен.

**~1**

запрос (NL Query)

## КЛЮЧЕВЫЕ СВОЙСТВА

- **Промпт — дефолт**  
Стартовая точка, а не shortcut
- **Доступность**  
Вне рабочего места (мессенджеры)
- **Эффективность**  
Сокращение цепочки действий

## ПЕРВЫЕ ШАГИ UX 3.0

- 1 Промпт-фильтр данных
- 2 GenUI карточки
- 3 Генеративные виджеты
- 4 AI-ассистент аналитики

• РАЗДЕЛ 02 • КОНЦЕПЦИЯ AGENT READY

# Что: Концепция Agent Ready ERP

# 02

---

Раскрытие сути интент-ориентированной платформы через 6 определяющих свойств автономных агентов, архитектуру генеративных интерфейсов (GenUI) и пайплайн NL-фильтрации.

# 6 определяющих свойств



01

## Автономность

Работают с иерархией задач и верификацией

Message Bus

Supervisor



02

## Безопасность

Доступ через существующие механизмы прав и ролей

UserPrincipal

Role-Based



03

## Контекст

Семантическое понимание данных и процессов

M-schema

RAG Index

BPM



04

## Права

Гранулярный контроль доступа к данным и операциям

AdminMeta

Row-Level

Per-Op



05

## Human-in-the-Loop

Обязательное одобрение критических действий

Hooks

Approvals

Audit



06

## Агентская разработка

Автономное выполнение тикетов и автообогащение

GitLab

Auto-Docs

**Ключевой принцип****Агент = Системный пользователь****(Своя ESession, роли, профили — не обходит права, а подчиняется им)**

# GenUI: Генеративные интерфейсы

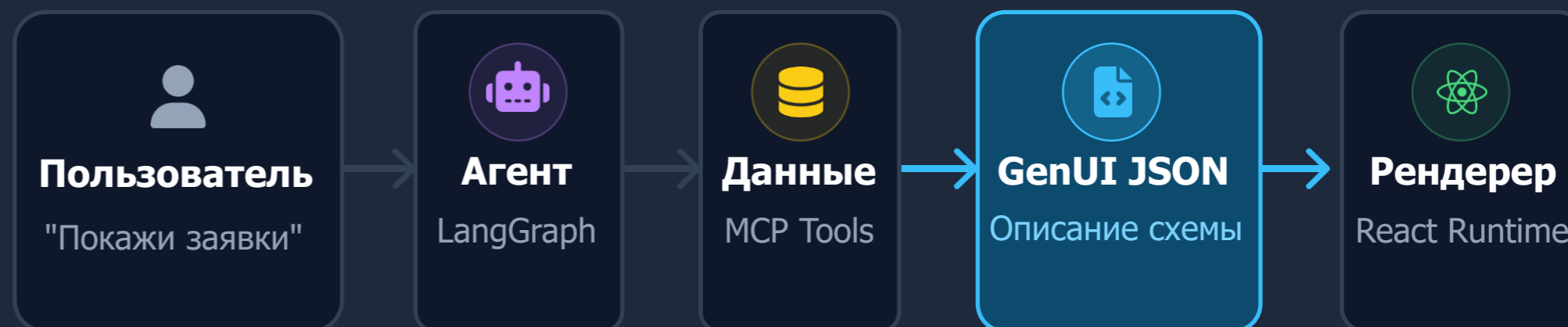
## Определение

Интерфейс, динамически генерируемый AI в рантайме под контекст задачи.

*Не AI-Assisted Design для разработчиков, а runtime-генерация для конечного пользователя.*

Outcome-Oriented Intent-Based

## Архитектурный поток



## Типы GenUI-компонентов

## Типы компонентов

Тип	React компонент	Описание и назначение
table	GenUITable	Таблица данных с сортировкой, фильтрацией и пагинацией (AgGrid)
card	GenUICard	Карточка отдельного документа или объекта с полями и статусами
form	GenUIForm	Форма ввода или редактирования данных с валидацией
chart	GenUIChart	Визуализация данных: bar, line, pie, donut (Recharts/ECharts)
summary	GenUISummary	Сводка KPI, ключевые метрики, счетчики (Big Number style)
action_list	GenUIActionList	Список доступных действий или рекомендуемых шагов (кнопки/ссылки)
composite	GenUIComposite	Контейнер для комбинации нескольких компонентов в одном view

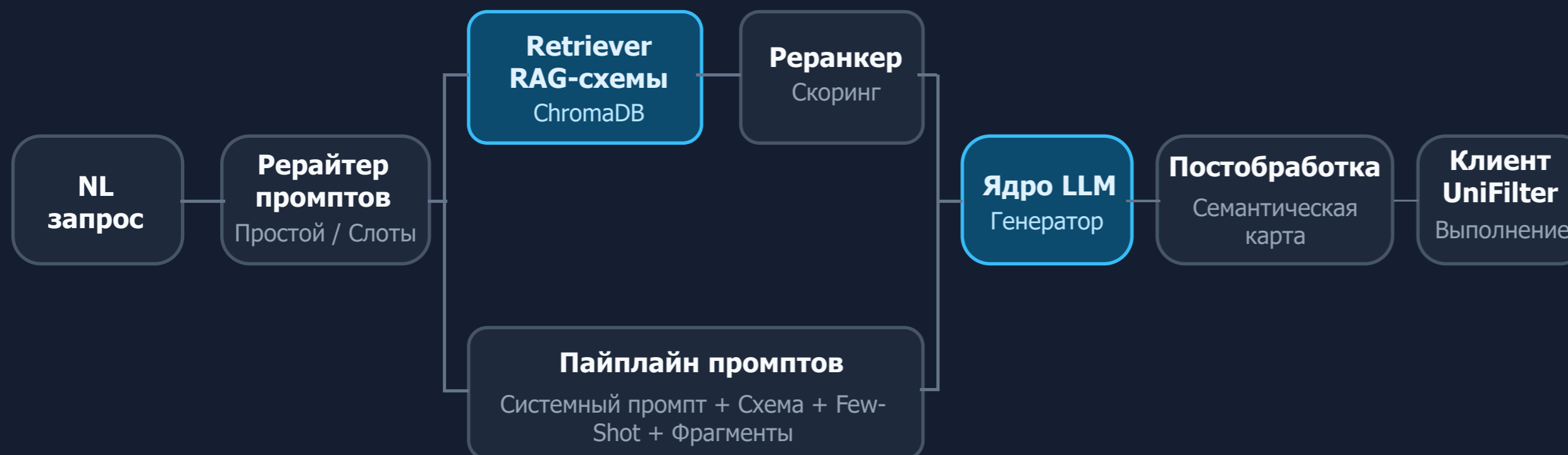
## Ограничения и Защита

- Фиксированный набор**  
 Агент **не генерирует** произвольный React/HTML код. Только JSON-структуру из разрешенного списка компонентов.
- Безопасная стилизация**  
 Внешний вид контролируется платформой через CSS-переменные. Агент не может сломать верстку.
- Данные через MCP**  
 GenUI не обращается к БД напрямую. Данные поставляются только через авторизованные MCP Data Tools.
- Fallback механизм**  
 Если JSON невалиден, рендерер автоматически переключается на текстовый/markdown режим отображения.

# Пайплайн NL-фильтрации

**ОНЛАЙН**

Онлайн-пайплайн: NL запрос → Построение SQL-фильтра



**МАСШТАБ ЗАДАЧИ**  
(Схема БД "Rcr\_CandidateVacancy")

<b>195</b> Таблиц	<b>2.1K</b> Полей
<b>453</b> Связей	<b>357KB</b> Размер

**Результаты прототипа**

**Btk\_Class** **100% (7/7)**

**Rcr\_CandidateVacancy**  
Уже работает **55% (16/29)**

Достижимо **83% (24/29)**

+5 fix retriever, +3 fix data. Оставшиеся 5 кейсов - промпт-тюнинг LLM.

**LLM И АРХИТЕКТУРА**

Прототип: gpt-5-mini, gpt-oss-120b (строят JOIN'ы и 3-уровневые связи)

Продакшен: **qwen3.5-35b** (self-hosted, без облачных провайдеров)

**ТЕХНОЛОГИИ**

- Модульный CLI-бенчмарк
- SWEEP (матрица экспериментов)
- Multi-view embeddings
- Slot-aware retrieval

**ОФЛАЙН**

Офлайн-пайплайн: Обогащение схемы и артефакты



# Визуализация: фильтрация промптом

## Интеллектуальный поиск и фильтрация через естественный язык

**AI-режим - альтернативный способ задания условий фильтрации**

AI может интерпретировать ваш запрос неточно. Пожалуйста, проверьте предложенные фильтры перед применением.

опыт до 3 лет и зп до 100000 или опыт от 6 лет и зарплата до 200000

Фильтры: ( Опыт работы ≤ 3 X Желаемая з/п ≤ 100 000 X ) ИЛИ ( Опыт работы ≥ 6 X Желаемая з/п ≤ 200 000 X )

ФИО	Вакансия	Отдел	Желаемая з/п	Валюта	Опыт раб
Иванов Владимир Александрович	UX/UI дизайнер	Системный отдел	80 000	Рубль	
Григорьев Виктор Павлович	главный бухгалтер	Бухгалтерия	90 000	Рубль	
Петров Аркадий Петрович	главный бухгалтер	Бухгалтерия	80 000	Рубль	
Меньшов Дмитрий Евгеньевич	главный бухгалтер	Бухгалтерия	100 000	Рубль	
Голубева Александра Павловна	UX/UI дизайнер	Системный отдел	120 000	Рубль	
Лаврова Ирина Александровна	Системный аналитик	Отдел внедрения систем			
Пунько Михаил Валерьевич	Системный аналитик	Отдел внедрения систем			

**Открывает условия фильтрации в модальном окне универсального фильтра**

**Промпт-фильтр данных**  
AI-режим позволяет вводить запросы на естественном языке и автоматически преобразует их в структуру условий универсального фильтра, упрощая и ускоряя работу с фильтрацией.

**Атрибуты фильтрации**

Наименование	Вид сравнения	Значение
Группа Отбор И		
Группа Или		
Группа И		
Опыт работы	Меньше или равно	3
Желаемая з/п	Меньше или равно	100000
Группа И		
Опыт работы	Больше или равно	6
Желаемая з/п	Меньше или равно	200000

• РАЗДЕЛ 03 • ГЛОБАЛЬНАЯ АРХИТЕКТУРА

# Как устроено: Техническая архитектура системы

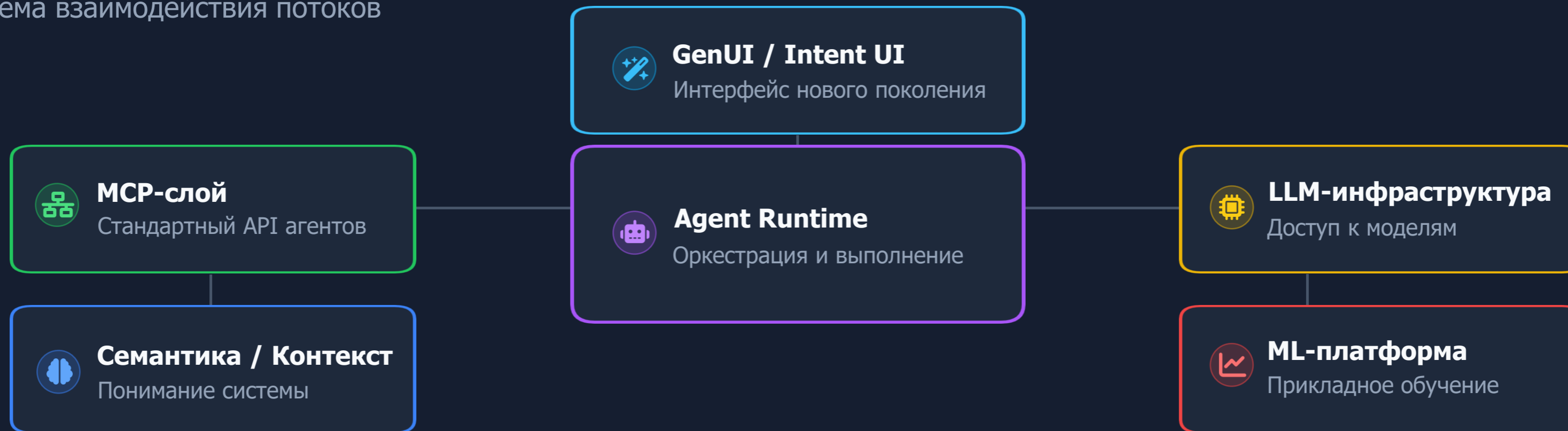
# 03

---

Детальный обзор многоуровневой архитектуры платформы: от слоя интеграции Model Context Protocol (MCP) и гибридного Agent Runtime до семантического контекста и LLM-инфраструктуры.







# Направления трансформации

Схема взаимодействия потоков

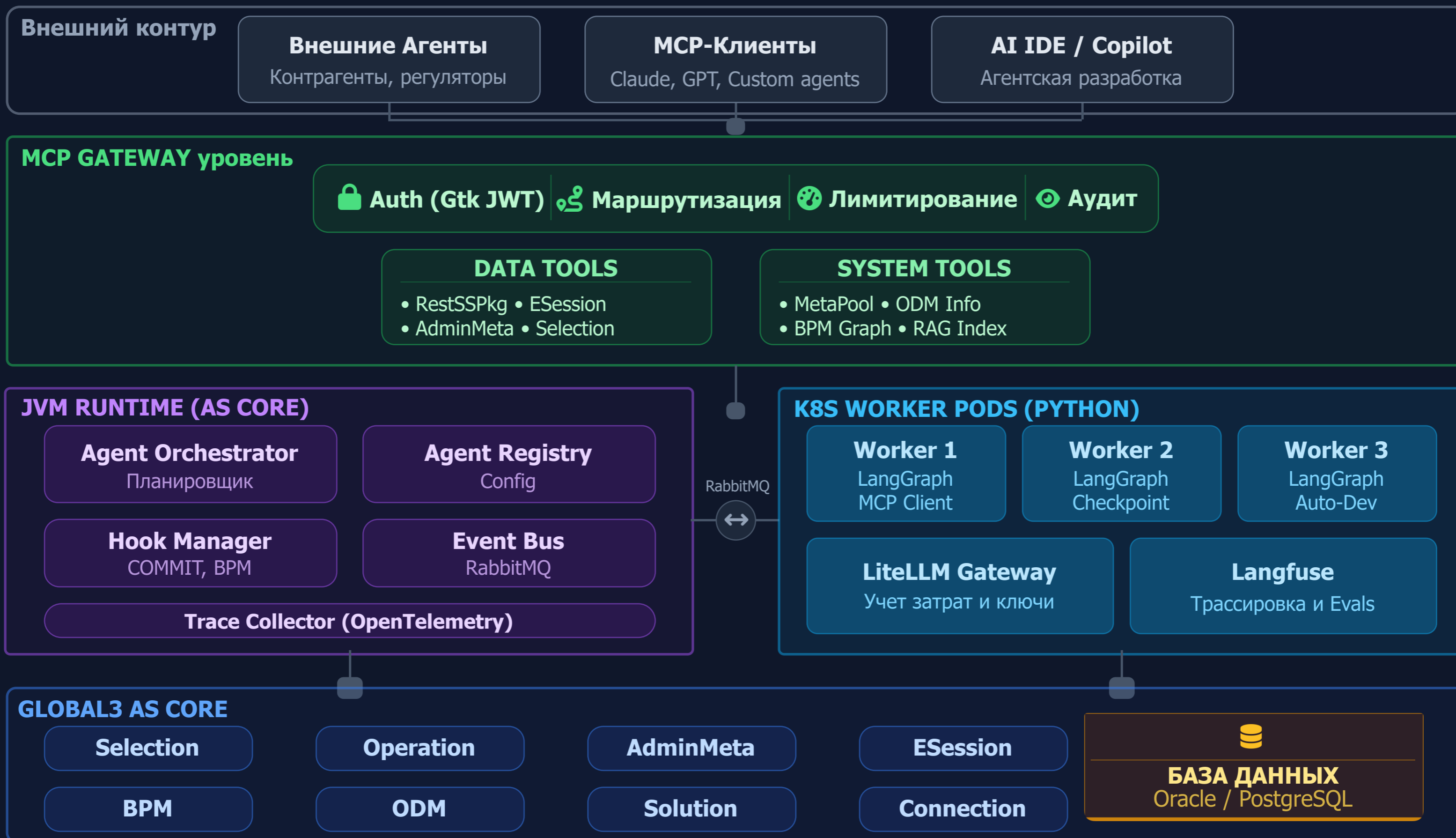


## Детальное описание направлений

6 основных направлений

Направление	Что даёт	Ключевые компоненты
 <b>MCP-слой</b>	Стандартный, безопасный API для подключения любых агентов (внутренних и внешних) к данным системы.	MCP GatewayData ToolsSystem ToolsGtk JWT
 <b>Agent Runtime</b>	Среда выполнения и управления жизненным циклом агентов. Гибридная модель (JVM + K8s).	Orchestrator (JVM)K8s WorkersEvent BusScheduler
 <b>Семантика / Контекст</b>	Позволяет агентам «понимать» структуру данных и бизнес-логику системы без хардкода.	AVM DocsODM M-schemaRAG IndexJEXL Skills
 <b>LLM-инфраструктура</b>	Единый шлюз доступа к языковым моделям. Абстрагирует от конкретного провайдера (OpenAI, local).	LiteLLMvLLMLangfuseCost Tracking
 <b>GenUI / Intent UI</b>	Интерфейс нового поколения, генерируемый под задачу пользователя. Снижает когнитивную нагрузку.	GenUI RendererNL-FilterChat InterfaceReact Components
 <b>ML-платформа</b>	Инструменты для классического машинного обучения (прогнозирование, классификация, аномалии).	Scikit-learnPyTorchSpaCyLangChain

# Общая архитектура (Agent Ready)



## КЛЮЧЕВЫЕ ПРИНЦИПЫ:

- Переиспользование абстракций
- Агент = Системный пользователь
- Гибридный Runtime
- Безопасность по умолчанию
- Только Open Source

# MCP-слой: Gateway & Tools

## MCP REST Конечные точки (SSE)

/app/sys/mcp/

**GET** /tools/list Перечень доступных инструментов

**POST** /tools/call Вызов инструмента (JSON-RPC)

**GET** /resources/list Список ресурсов (файлы, доки)

## Поток аутентификации и обнаружения



**Gtk JWT Token**  
Bearer Header



**UserPrincipal**  
isAgent=true



**AdminMeta**  
Проверка прав



**AVM / ODM**  
Определения



**Infinispan**  
Кэш инструментов

## Data Tools (Прикладной контур)

Package: RestSS/ESPkg

Имя инструмента	Сессия	Описание
<a href="#">selection_list</a>	<b>SS</b>	Получение списка доступных выборок
<a href="#">selection_open</a>	<b>ES</b>	Открытие выборки, получение структуры атрибутов
<a href="#">selection_query</a>	<b>ES</b>	Чтение данных с фильтрацией, сортировкой, пагинацией
<a href="#">selection_nl_filter</a>	<b>ES</b>	NL-фильтрация (LLM → SQL макрос)
<a href="#">selection_insert</a>	<b>ES</b>	Вставка новой записи в выборку
<a href="#">selection_edit</a>	<b>ES</b>	Редактирование полей существующей записи
<a href="#">selection_delete</a>	<b>ES</b>	Удаление записи (если разрешено правами)
<a href="#">selection_commit</a>	<b>ES</b>	Фиксация изменений транзакции
<a href="#">operation_execute</a>	<b>ES</b>	Выполнение серверной операции (вкл. BPM)

## System Tools (Системный контур)

Только чтение

Имя инструмента	Описание
<a href="#">meta_selection_info</a>	Метаданные выборки: атрибуты, типы
<a href="#">odm_class_info</a>	Полное ODM-описание класса
<a href="#">bpm_process_info</a>	Описание BPM: состояния, переходы
<a href="#">odm_search</a>	RAG-поиск по семантическому индексу
<a href="#">sys_get_constants</a>	Получение системных констант
<a href="#">user_profile_info</a>	Инфо о пользователе и правах

### Обнаружение инструментов

Выборки с <documentation> автоматически становятся инструментами

## Documentation Tools (Платформенный контур)

Отдельный MCP-сервер для поиска по документации платформы: спецификации API, ODM, code-style, ADR. Развёртывается как Docker-контейнер.

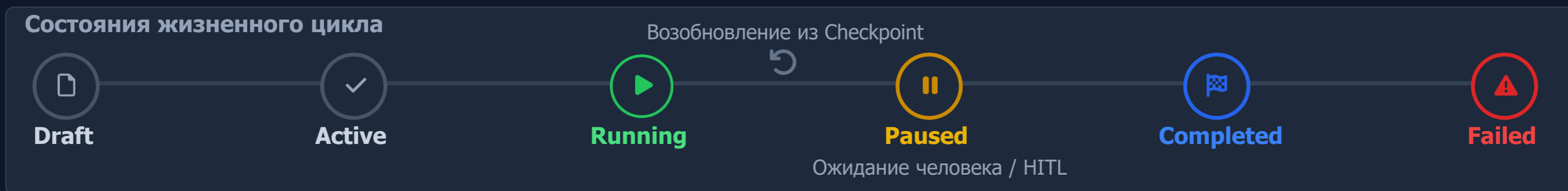
Имя инструмента	Описание
<a href="#">rag_retrieve_tool</a>	Поиск чанков по документации (без LLM). Возвращает фрагменты с source_url и section_path.
<a href="#">rag_answer_tool</a>	Полный RAG-пайплайн: поиск + LLM-генерация ответа. Для пользовательских вопросов по платформе.

**Скиллы:** /global-docs, /global-docs-setup | **IDE:** Claude Code, Cursor, Windsurf, Cline

# Среда выполнения агента



Типы агентов		
ТИП	СОЗДАТЕЛЬ	ВОЗМОЖНОСТИ И ОГРАНИЧЕНИЯ
<b>System</b>	Платформа	MonitorAgent, DataQuality. Запуск при старте. Полный доступ (System Tools).
<b>Application</b>	Аналитик (No-Code)	Бизнес-задачи. Конфиг в BTK_AgentConfig. MaxDepth=3. Доступ через AdminMeta.
<b>User</b>	Пользователь	Личные помощники. MaxDepth=1. Строгие квоты. Только свои данные.



# Семантика и контекст

## СТРУКТУРА СКИЛЛА АГЕНТА

Атрибуты

Операции

**ВЫБОРКА**  
(Selection)

Фильтры

Связи

Семантическое Описание

"Скилл = Выборка + Семантика"

## JEXL SKILLS No-Code

- ✓ Динамические скиллы от аналитиков и пользователей
- ✓ Автоматическая регистрация как MCP tool
- ✓ Безопасная песочница (sandbox) для выполнения

## AVM: интерфейс

```
<selectionname="Rcr_Candidate">
  <documentation>
    База кандидатов с контактами...
  </documentation>
  <attributename="Salary">
    <documentation>Ожидаемая ЗП</...>
  </attribute>
  ...
</selection>
```

Application View Markup

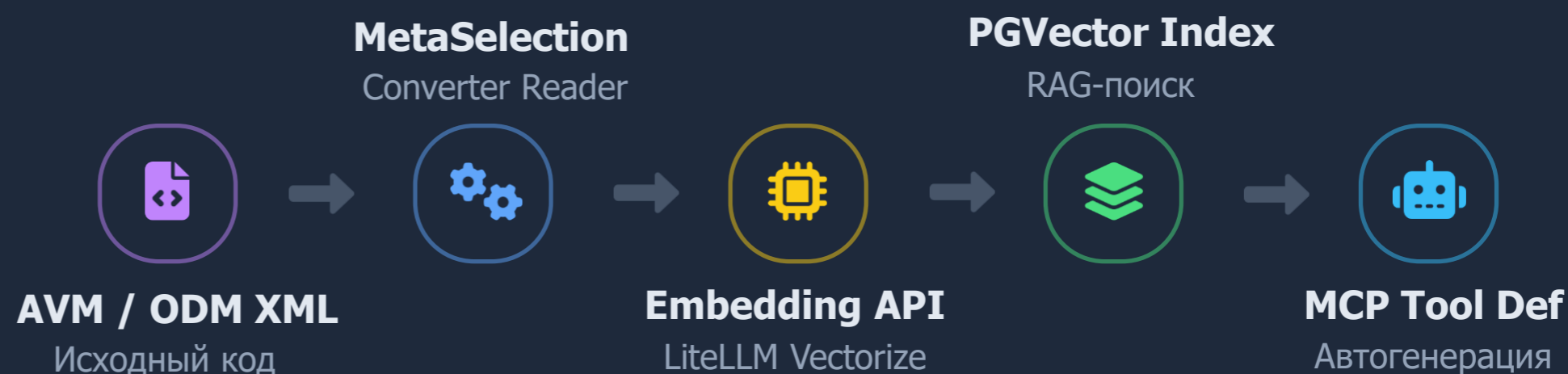
## ODM: данные (m-schema)

```
<m-schemaclass="Rcr_Candidate">
  <domain>HR, Recruitment</domain>
  <purpose>Хранение профилей
  соиск.</purpose>
  <related>
    <linkto="Rcr_Vacancy" type="master"/>
  </related>
</m-schema>
```

Object Domain Model

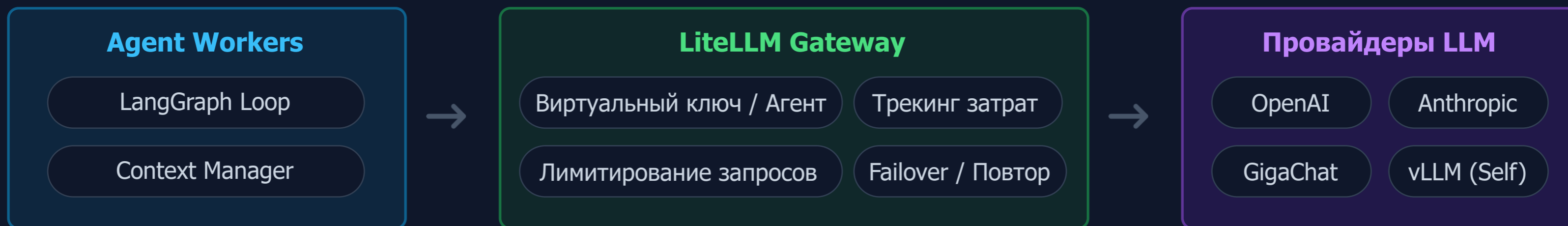
## ПАЙПЛАЙН: ОТ ОПИСАНИЯ К RAG-ПОИСКУ

Автоматизированный пайплайн



Кеширование определений в **Infinispan**

# LLM-инфраструктура



### Langfuse Observability

- Трассировка — Полный цикл запроса
- Eval-датасеты — Регрессионное тестирование
- Дашборд затрат — Биллинг по агентам

### Компоненты стека

Компонент	Лицензия	Назначение
LiteLLM	MIT	Gateway к 100+ провайдерам. Унифицированный API (OpenAI format).
LangGraph	MIT	Фреймворк агентов. Граф состояний (BPM-like), persistence.
vLLM	Apache 2.0	Self-hosted inference (Llama, Qwen). Опционален (для on-premise).
Langfuse	MIT	Observability: трассировка, датасеты, аналитика затрат.

## Измерения бенчмарков

**✓ Качество**  
% Успеха  
Langfuse Evals / LLM-as-judge

**💰 Стоимость**  
\$/Задача  
LiteLLM Cost Tracking

**⌚ Скорость**  
Задержка (p95)  
Langfuse Traces

**🛡️ Безопасность**  
% Guardrails  
AdminMeta Logs / Scorer

**🔧 Autotune агентов**  
Запланировано на **Этап 3**  
Эволюционная оптимизация промптов и инструментов через Langfuse API

• РАЗДЕЛ 04 • БЕЗОПАСНОСТЬ И ML

# Инфраструктура: Безопасность и AI- платформа

---

# 04

Обзор архитектуры безопасности, концепции «Агент как пользователь», моделей контроля Human-in-the-Loop и интеграции существующих классических ML-сервисов в качестве инструментов для агентов.

# Безопасность и контроль

## Агент как пользователь



### BTK\_Agent

UserPrincipal(isAgent=true)

ROLE\_HR\_VIEW


ROLE\_API\_EXEC


## Изолированная ESession

Собственная сессия, не блокирует реальных пользователей. Выполняется с квотами на ресурсы.

## Уровень безопасности AdminMeta

 Доступ к выборке  
`isAccessDenied()`

 Запрет операций  
`getDisabledOperationList()`

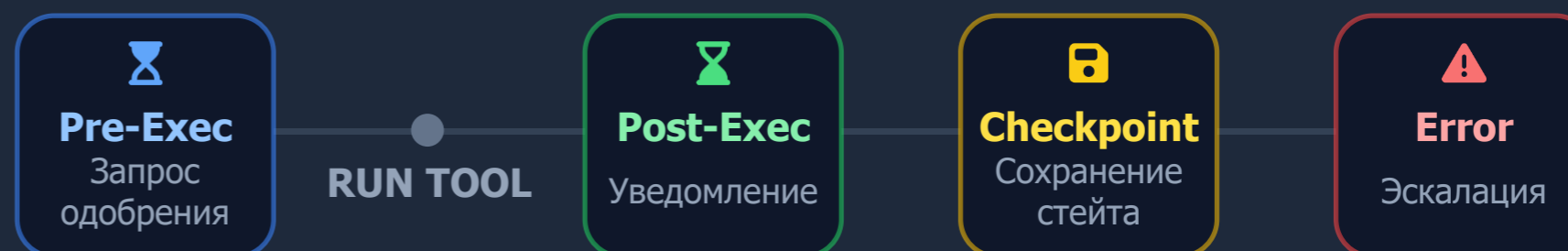
 Только чтение полей  
`getReadOnlyAttributeList()`

 Row-Level Security  
&DefUniFltMacros (SQL WHERE)

## Модели контроля

Модель	Механизм	Сценарий использования
<b>ВРМ-задача</b>	Vpm_Task	Формальное согласование. Агент ждет завершения задачи.
<b>Event Bus</b>	WebSocket Push	Быстрое подтверждение в чате. Inline-кнопки (Одобрить).
<b>Делегация</b>	Impersonation	Агент работает от имени пользователя под его правами.

## Хуки событий



## Аудит действий

OpenTelemetry

Виджет AgentTrace

span-a1b22026-03-10 09:15:22 agent.id=rcr\_bot MCP:selection\_open OK

span-a1b32026-03-10 09:15:23 agent.id=rcr\_bot MCP:selection\_nl\_filter OK

span-a1b42026-03-10 09:15:25 agent.id=rcr\_bot MCP:selection\_insert WAIT\_APPROVAL

# ML-платформа Global ERP

 Платформа по созданию, обучению и доставке сервисов искусственного интеллекта внутри контура Global ERP

## Жизненный цикл ML



### 1. Анализ

Выбор парадигмы,  
постановка задачи



### 2. Обработка данных

Выявление, сбор и  
очистка данных



### 3. Создание модели

Выбор модели,  
параметризация



### 4. Обучение

Тренировка модели на  
данных



### 5. Доставка и работа

API, мониторинг, запуск  
в кластере

## Технологический стек по парадигмам

### Классическое ML

Scikit-learn

Классификация,  
кластеризация, регрессия

### Глубокое обучение

PyTorch

CUDA

Нейросети для сложных  
задач

### NLP

SpaCy

Thinc

Обработка текста, NER

### Генеративный AI

LangChain

Llama / HF


RAG, чат-боты, генерация


### Векторная БД

PGVector

Эмбединги,  
семантический поиск

## Реализованные сервисы

 RAG для документации

 Предиктив значений полей



### Связь с Agent Ready

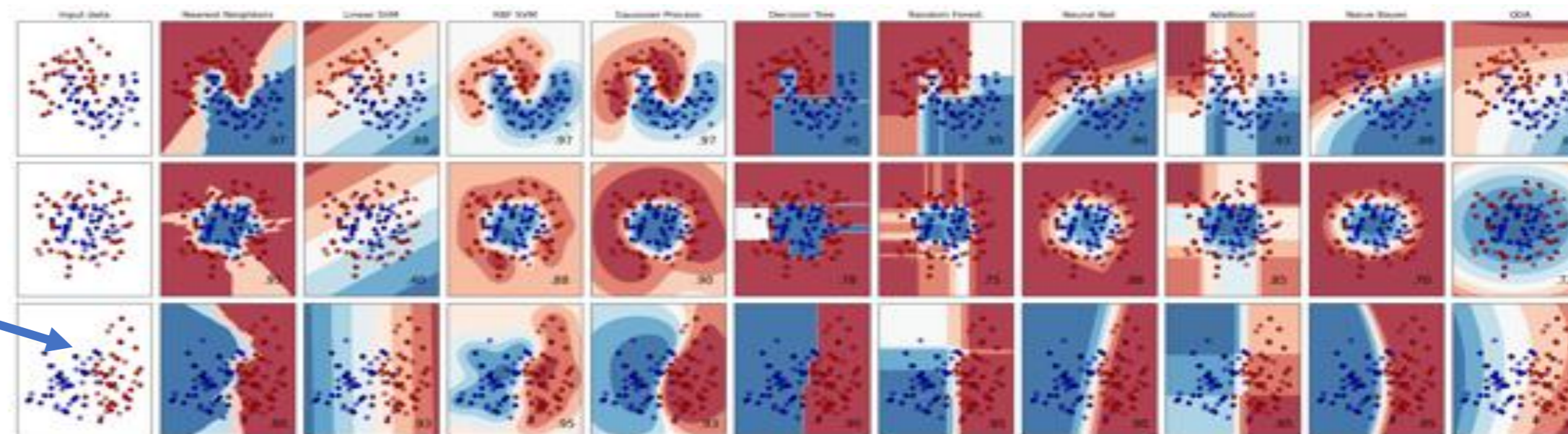
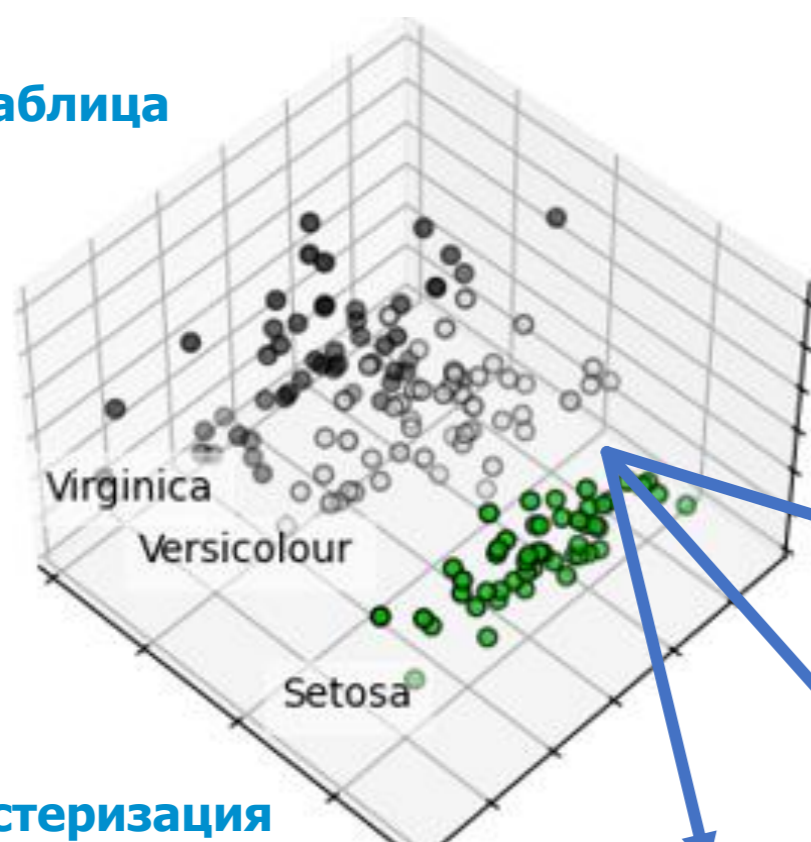
Единый Python/LLM стек. ML-модели становятся **MCP Tools**.

Опыт Self-Hosted

Переисп. инфры

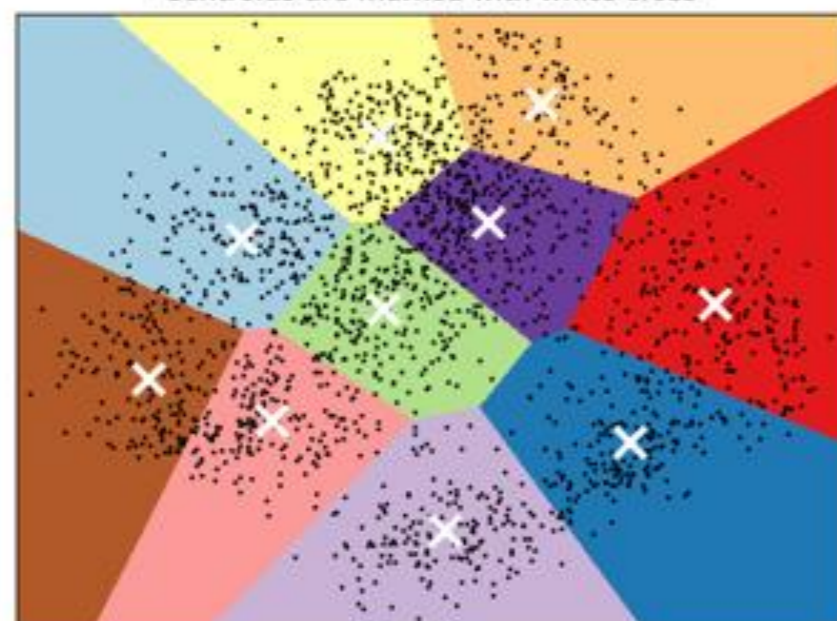
# Skikit: Классическое обучение

Таблица

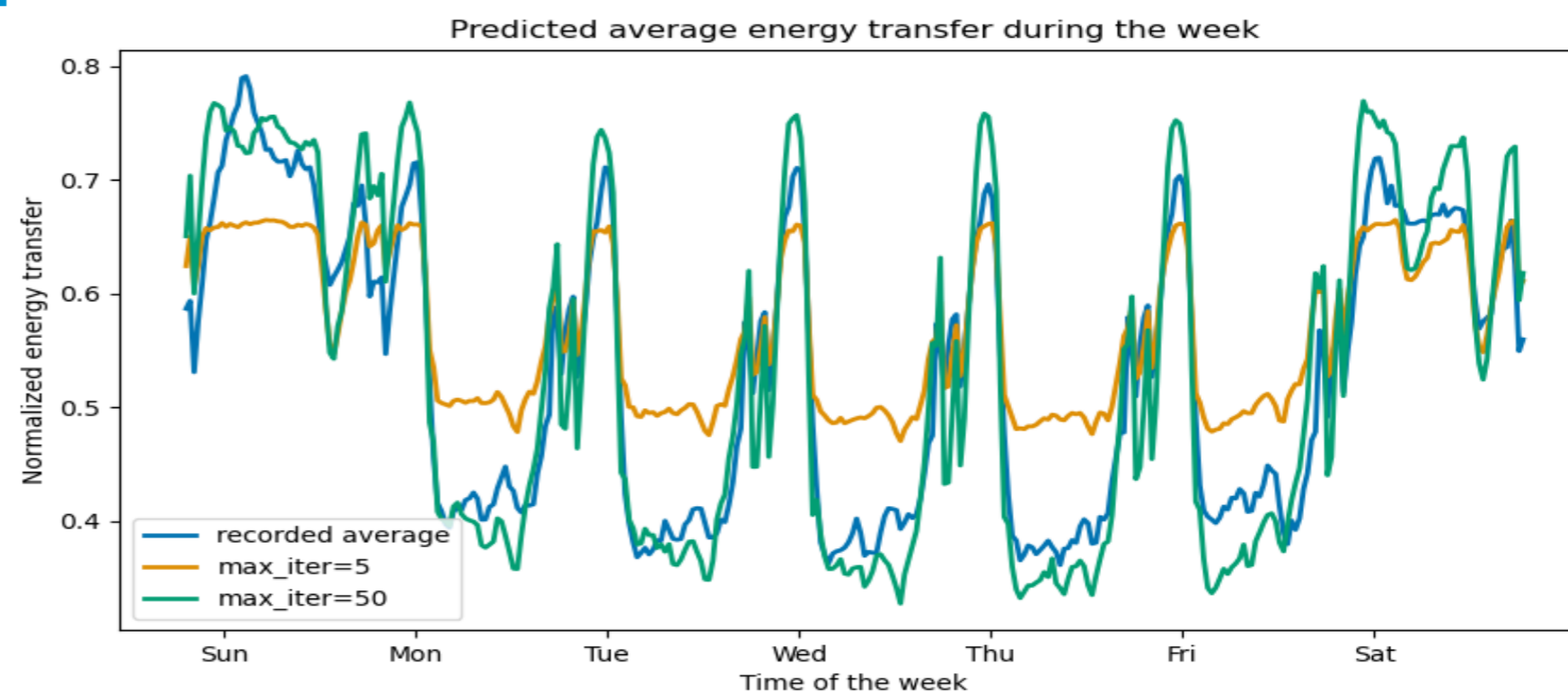


Кластеризация

K-means clustering on the digits dataset (PCA-reduced data)  
Centroids are marked with white cross



Регрессия



# PyTorch: Глубокое обучение

## Гибкая структура входных данных

### Обработка языка

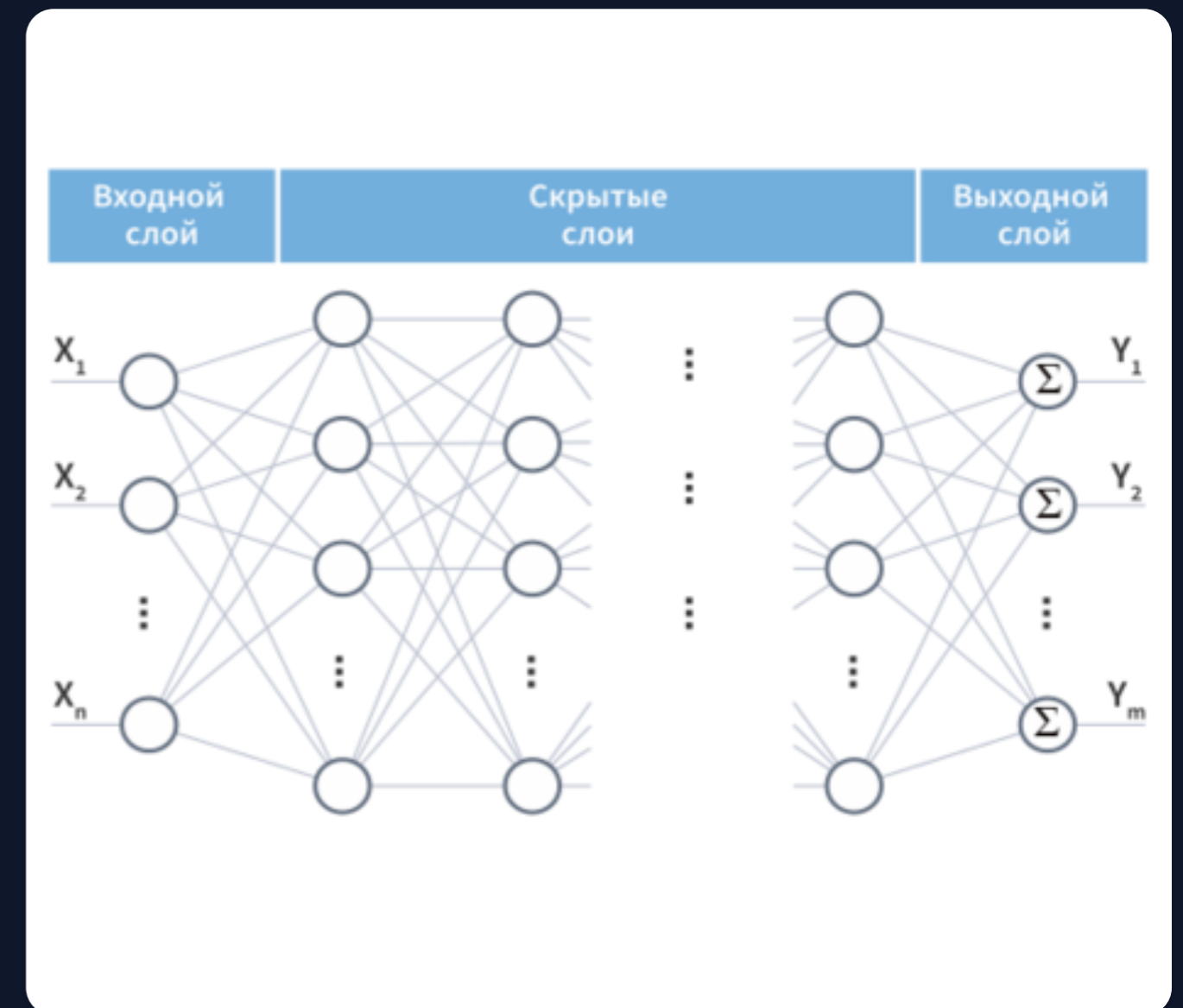
- Ответы на вопросы
- Перевод
- Подведение итогов

### Компьютерное зрение

- Классификация изображений
- Определение объекта
- Сегментация

### Аудио

- Распознавание речи
- Классификация

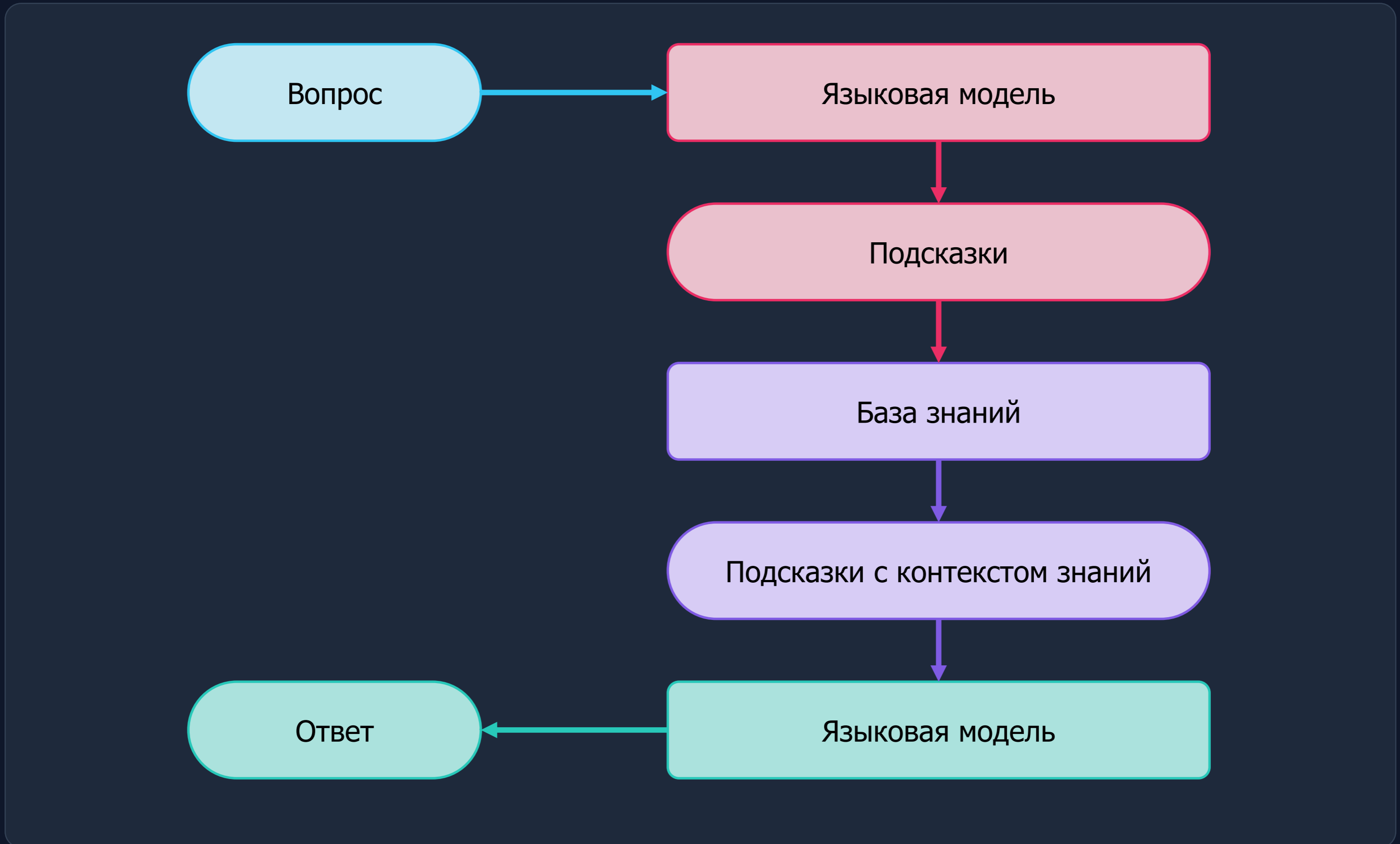


# SpaCy: Обучение по тексту



- Распознавание сущности
- Кластеризация

# Langchain: Генеративный ИИ



# Разработка умного чат-бота для документации с использованием RAG-архитектуры

```

agent.py > RAGAgent > _load_models
1 import yaml
2 import os
3 import asyncio
4 import json
5 from typing import Dict, Any, List, Optional, Iterable
6 from dataclasses import dataclass
7 from load_models import load_emb_model, load_llm_model
8 from langchain_postgres.vectorstores import PGVector
9 from langchain_core.runnables import RunnablePassthrough
10 from langchain_core.prompts import ChatPromptTemplate
11 from langchain_core.output_parsers import StrOutputParser
12 from langchain_core.documents import Document
13
14 SYSTEM_PROMPT_TEMPLATE = (
15     "Вы - полезный AI-ассистент, который помогает пользователям [ ] информацией из предоставленной документации. "
16     "Используйте только информацию из контекста ниже для ответа на вопросы.\n\n"
17     "Контекст:\n{context}\n\n"
18     "Важно:\n"
19     "- Отвечайте кратко и по существу\n"
20     "- Если ответа нет в контексте, скажите [ ] этом\n"
21 )
    
```

Problems Output Debug Console Terminal Ports

LLM-модель загружена.  
Агент готов к работе.  
Введите ваш вопрос (`q` для завершения):  
> Что такое балансировщик нагрузки?

Ответ:  
AI: Балансировщик нагрузки распределяет пользователей между серверами приложений в кластере.

Рекомендуемые разделы для изучения:  
- Обзор > Архитектура системы > Балансировщик нагрузки – doc-gsf-guide-main

> q  
Завершение работы.  
PS C:\Users\t.boIdovskaya\Documents\prototypeRAG> [ ]

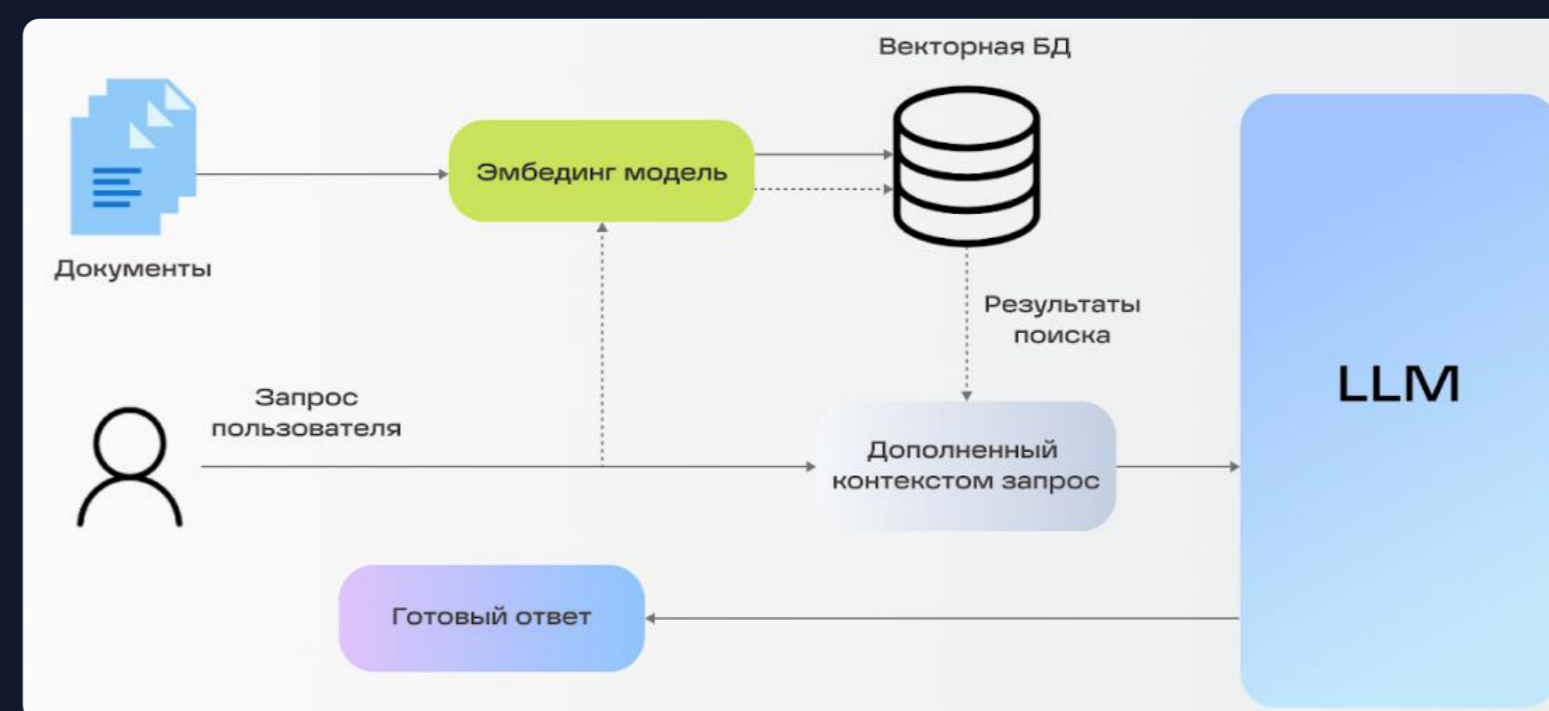
## RAG-система для чат-бота по документации:

### AI-модели:

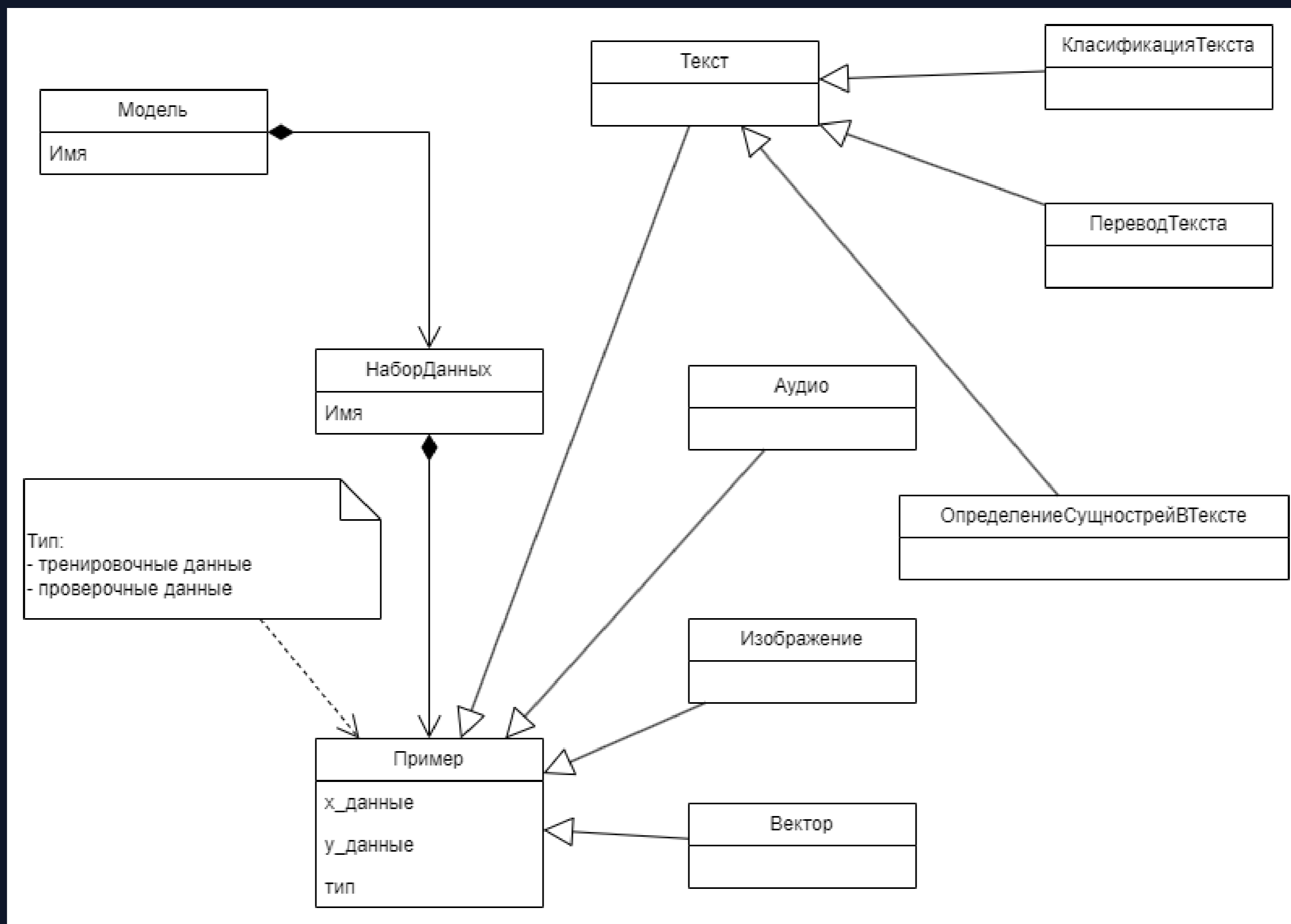
- LLM: локальная Llama-3.1-8B-Instruct с Hugging Face
- Embedding: ru-en-RoSBERTa

**Фреймворк:** LangChain

**Векторная БД:** PostgreSQL с расширением PGVector

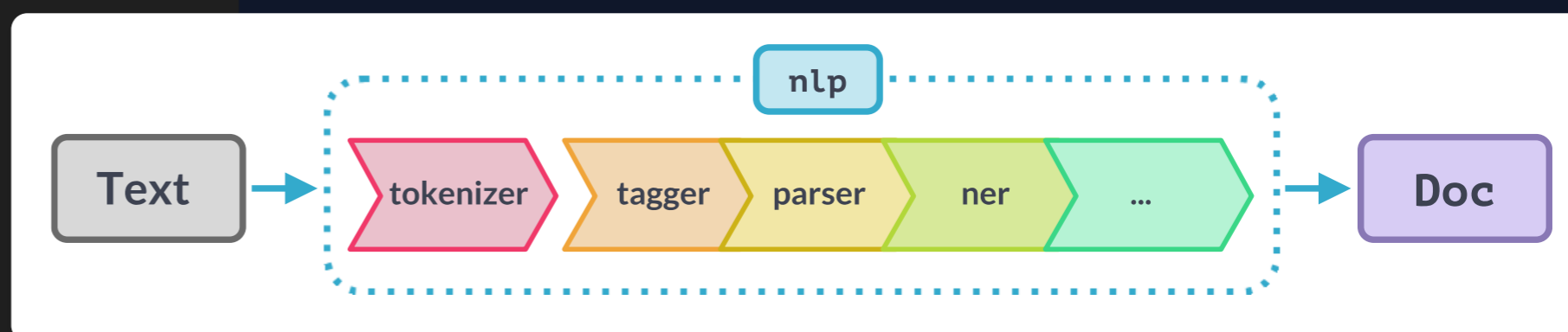


# Сбор данных



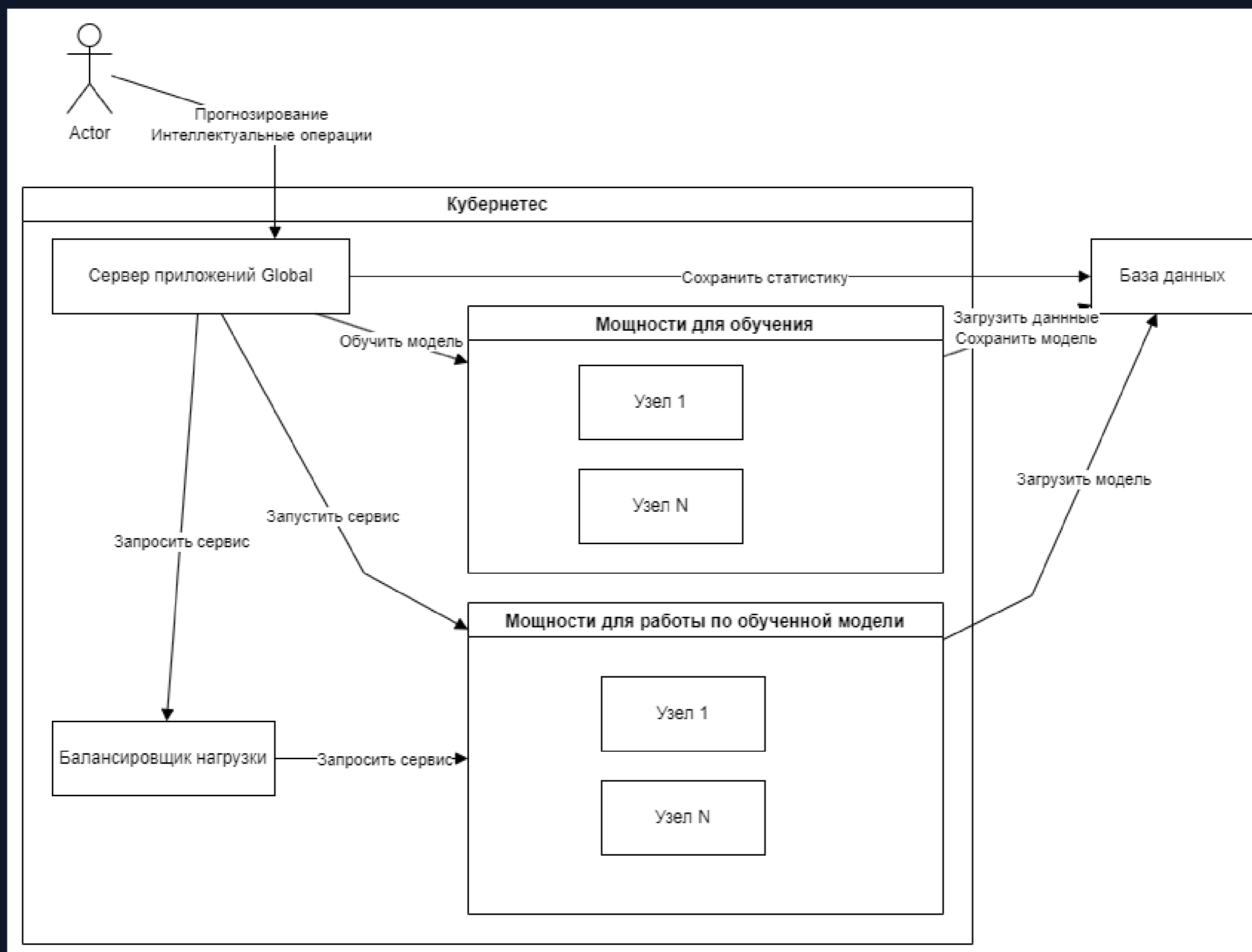
# Создание модели

```
[nlp]
lang = "ru"
pipeline = ["tok2vec", "textcat"]
batch_size = 1
[components]
[components.tok2vec]
factory = "tok2vec"
[components.tok2vec.model]
@architectures = "spacy.Tok2Vec.v2"
[components.tok2vec.model.embed]
@architectures = "spacy.CharacterEmbed.v2"
width = ${components.tok2vec.model.encode.width}
rows = 7000
nM = 64
nC = 8
include_static_vectors = true
[components.tok2vec.model.encode]
@architectures = "spacy.TorchBiLSTMEncoder.v1"
width = 256
depth = 2
dropout = 0.0
[components.textcat]
```



Данный сервис используется в системе для предсказания значений полей в объектах системы на основе поступающей на вход текстовой информации

# Обучение, доставка и работа с моделью



• РАЗДЕЛ 05 • ПРАКТИЧЕСКИЕ КЕЙСЫ

# Доказательства: Реальные кейсы применения

# 05

---

Демонстрация работы Agent Ready платформы на примере четырех бизнес-задач из модулей рекрутмента (RCR), складского учета (STK) и управления снабжением (CSC).

КЕЙС: РЕКРУТМЕНТ (RCR)

# Утренняя сводка рекрутера

## 🕒 Бизнес-задача

HR тратит **15-20 минут** каждое утро на ручной просмотр 4-5 разрозненных выборок (отклики, собеседования, оферы, вакансии).

## 🤖 Решение Agent Ready

Агент автоматически собирает данные, анализирует их, формирует **персонализированную сводку** и доставляет ее в чат при входе в систему. Время подготовки: 0 минут.

## Поток процесса



## Что демонстрирует



### JEXL Skills

- Аналитик создает логику сбора данных без написания кода агента.
- Использование Low-Code подхода.



### Agent Scheduler

- Реакция на системный триггер OnConnect.
- Предварительная проверка условий (роль пользователя, время).



### GenUI Composite

- Комбинирование 4 разных виджетов в одном сообщении.
- Динамическая верстка сложного интерфейса на лету.



### Персонализация

- Настройки отображаемых блоков берутся из профиля.
- Индивидуальные пороги в userPreferences.



### Безопасность

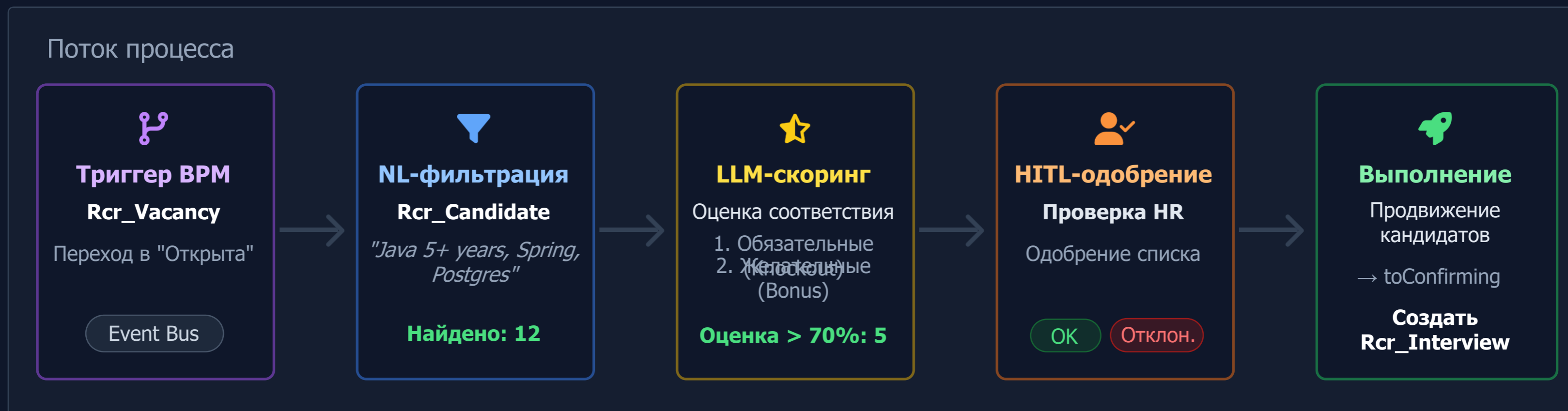
- AdminMeta Row-Level ограничения.
- Агент видит только данные конкретного HR через idRespEmpl.

КЕЙС: РЕКРУТМЕНТ (RCR)

# Скрининг кандидатов

**Бизнес-задача:** HR тратит ~2 часа на первичный скрининг резюме по одной вакансии.

**Решение:** Агент автоматически сопоставляет требования с профилями и продвигает подходящих.



## Выборки и операции

ВЫБОРКА	ОПЕРАЦИЯ	РОЛЬ В ПРОЦЕССЕ
Rcr_Vacancy	selection_query	Источник требований к кандидату (hard skills)
Rcr_Candidate	nl_filter	Семантический поиск по базе резюме (PDF/Doc)
Rcr_CandidateVacancy	op_execute	Продвижение по воронке (Formed → Confirming)
Rcr_Interview	selection_insert	Создание документа собеседования для одобренных

## Что демонстрирует

**Триггер Event Bus**  
Агент реагирует на бизнес-событие (смена статуса вакансии), а не расписание.

**NL-фильтрация**  
Сложный критерий поиска задается естественным языком, трансформируется в SQL.

**Подход HITL**  
Критическое решение (кого пригласить) остается за человеком. Агент готовит шорт-лист.

КЕЙС: СКЛАД (STK)

# Инвентаризация склада

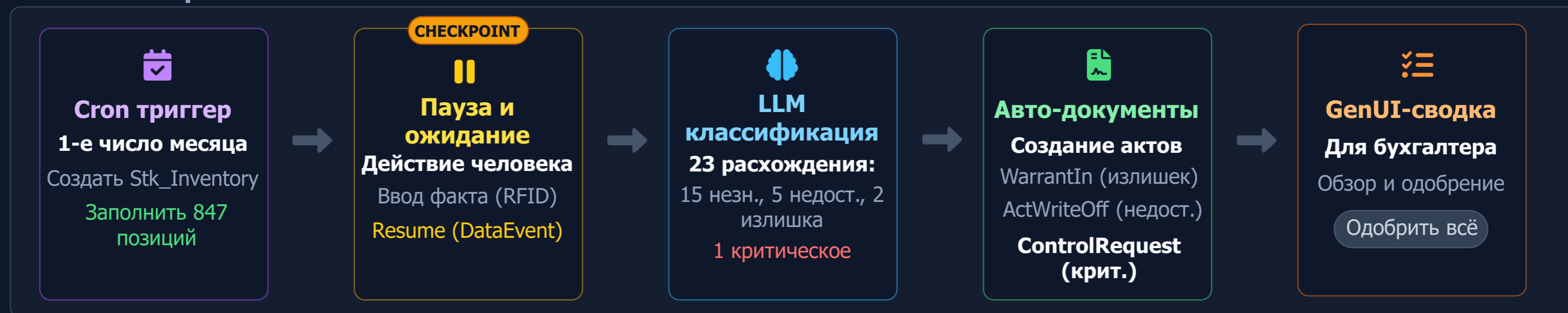


**Бизнес-задача:** Ручной подсчет и сверка 800+ позиций занимает 2-3 дня с простоями.



**Решение:** Агент готовит данные, ждет ввода факта, классифицирует расхождения и создает акты.

## Поток с контрольной точкой



## Выборки

Выборка	Действие	Назначение
Stk_Inventory/Det	create/fill	Базовый документ инвентаризации (план/факт)
Stk_WarehouseRemainder	query	Источник учетных остатков на дату
Stk_ActWriteOff	auto_create	Акт списания для позиций с недостачей
Stk_ControlRequest	alert	Запрос на контрольный пересчет (критическое)

## Что демонстрирует



### Checkpoint / Resume

Агент "засыпает" в ожидании данных от человека и "просыпается" по событию DataStore.



### LLM Классификация

Интеллектуальная классификация причин расхождений (ошибка ввода vs кража).



### Авто-документация

Агент сам создает подчиненные документы, человек только утверждает результат.

КЕЙС: СКЛАД + СНАБЖЕНИЕ (STK + CSC)

# Автопополнение запасов

**Бизнес-задача:** Контроль неснижаемых остатков и автоматическое формирование заявок на пополнение без участия человека.



**СОБЫТИЕ:**  
**data.changed.Stk\_WarrantOut**

Отгрузка со склада (включена пакетная дедупликация)



**StockMonitorAgent**

Оркестрация, деление бюджета, агрегация результатов

ResourceBudget 60% Использовано



**AnalyzerWorker**

Проверяет остатки vs минимумы

$Stk\_WarehouseRemainder < Stk\_GoodLimit$

Найдено: 3 товара ниже мин.

MCP: stk\_query



**ForecastWorker**

ML-прогноз потребления

История 6 мес. → Prophet/Arima

Рек.: +150 ед. (сезонность)

MCP: ml\_predict



**ProcurementWorker**

Поиск на других складах или закупка

Проверить другие склады?

Создать Csc\_GoodsDemand

MCP: csc\_insert



**Паттерн Supervisor**

Один агент управляет группой узкоспециализированных воркеров.



**Прогноз ML**

Интеграция с ML-сервисом для умного расчета количества.



**Кросс-модульная интеграция**

Агент склада создает документ в снабжении (stk → csc) через API.



**Дедупликация Event Bus**

Схлопывание множества событий отгрузки в одну задачу.

• РАЗДЕЛ 06 • ВНЕДРЕНИЕ И ВЫВОДЫ

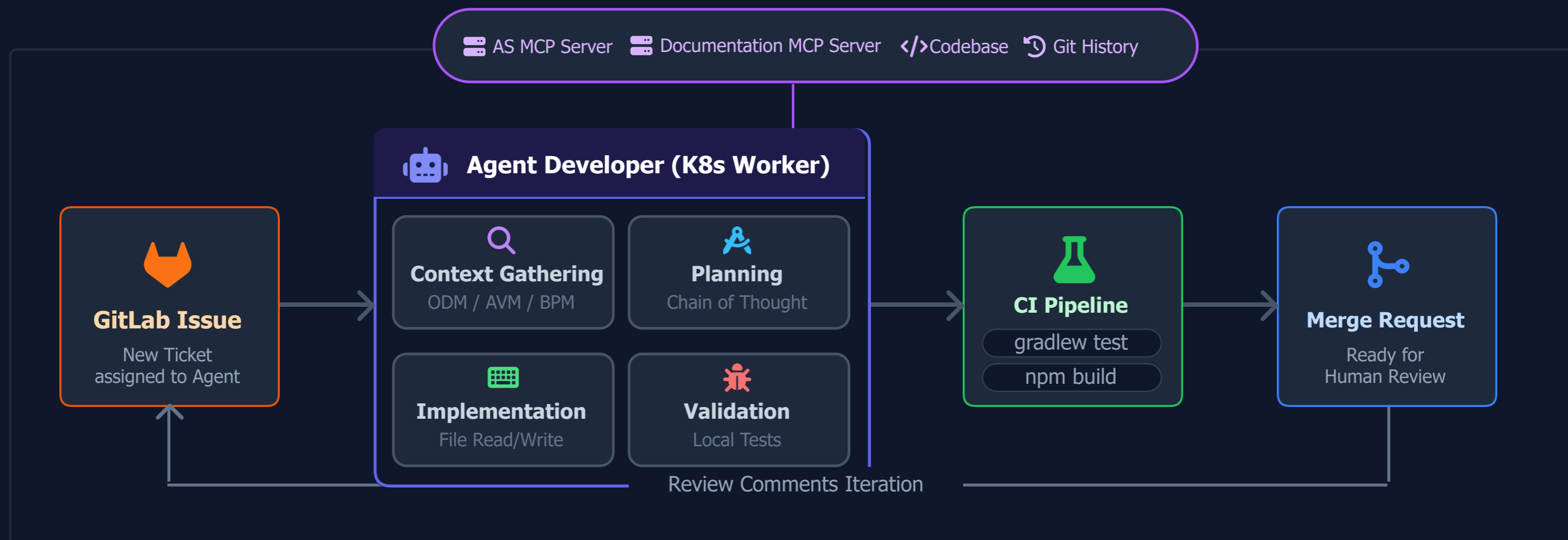
# План действий: Реализация и Roadmap

# 06

---

Детализированная дорожная карта трансформации из 4 этапов, стратегия инкрементального внедрения платформы и определение первых практических шагов для успешного старта проекта.

# Автономный Агент-разработчик



## Автообогащение Контекстом

Solves "Cold Start"

При изменении файлов ODM/AVM агент автоматически генерирует:



**M-schema** Domain, Purpose, Related entities metadata



**AVM documentation** Interface descriptions for Agent UI understanding



**regEmbeddings** dbData script to register semantic vectors



## Ограничения Безопасности



Нет прямого push в **master / release**. Только через MR.



Обязательное **Human Review** перед слиянием кода.



CI-метки **ci:skip-\*** игнорируются (агент обязан пройти тесты).



Production DB доступна только в режиме **Read-Only**.

# Как это выглядит: разработчик и AI-агент

Разработчик формулирует задачу, агент обращается к документации через MCP и генерирует код за сессию

## 1 Постановка задачи и поиск по документации

Запрос: «встроить Chart.js-визуализацию». Агент автоматически обращается к документации платформы через Tool и находит нужные разделы.

```
> Дай код с примером встраивания в представление прикладной выборки
Global ERP кастомную разметку с визуализацией данных выборки через
Chart.js.

• Skill(global-docs)
  L Successfully loaded skill

• globalerp-docs-rag - rag_retrieve_tool (MCP)(query: "кастомная
  разметка в представлении прикладной выборки custom main view", top_k: 12,
  score_threshold: 0.5)
  L {"result":{"request_id": "\49a46367-945d-4009-8bb9-4877961e486",
  "status": "\success", "chunks": [{"text": "Source: Выборка / Разметка
  выборки / Наследование разметки ... +267 lines (ctrl+o to expand)"}]}}
```

## 2 Архитектура решения + генерация кода

На основе извлеченного контекста агент предлагает архитектуру из 4 слоёв (Разметка, Backend, Выборка, Frontend) и переходит к генерации. Пишет XML-разметку (avm.xml) и backend-логику на Scala, которая собирает данные из мастер-выборки и передаёт их на визуализацию.

• Теперь у меня полная картина архитектуры. Вот комплексный пример встраивания Chart.js в представление прикладной выборки через механизм веб-расширений (extControl).

### Архитектура решения

Механизм состоит из 4 слоёв:

Слой	Файл	Назначение
Разметка	.avm.xml	Объявляет <extControl> во фрейме
Backend (Abi)	*Abi.scala	Подготавливает данные для фронтенда
Выборка (Avi)	*Avi.scala	Бизнес-логика, вызов backend при обновлении
Frontend	main.js	Рендерит Chart.js в DOM-элемент фрейма

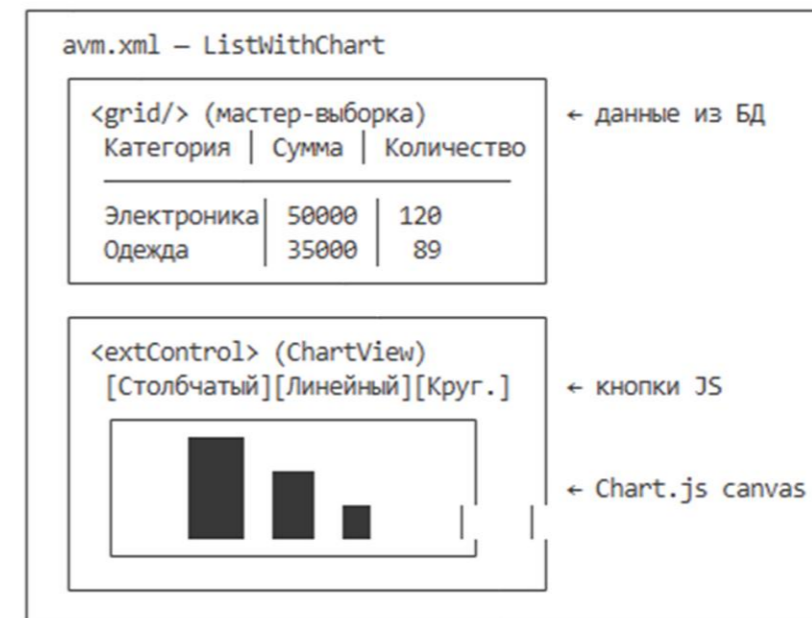
### 1. Разметка выборки – My\_SalesChart.avm.xml

```
<?xml version="1.0"?>
<view xmlns="http://www.global-system.ru/xsd/global3-view-1.0"
  name="My_SalesChart">
  <!-- Базовое отображение с данными -->
  <representation name="Default" caption="Продажи по категориям">
```

## 4 Готовый результат с объяснением

Полное решение сопровождается текстовым описанием потока данных и ключевых точек интеграции. Архитектура готова к проверке человеком.

### Как это работает



### Поток данных:

1. Мастер-выборка загружает строки из БД через onRefresh
2. Детальная выборка ChartView получает событие afterRefresh()
3. Avi вызывает backend.afterMasterRefresh() через selection.layout.view.asExtCor
4. Abi итерирует строки мастер-выборки, формирует JSON с labels и datasets
5. model.acceptChanges() отправляет данные на фронтенд
6. JS-контроль получает событие afterVisitModel() → вызывает updateChart()
7. Chart.js рендерит график в <canvas>

### Ключевые точки интеграции:

- extController.subscribeNodeChanges(...) – подписка на изменения модели (бэкенд)
- extController.postInputData(marker, data) – отправка данных от пользователя (фр)
- extController.postAsyncRequest(...) – произвольные запросы к бэкенду



**Ключевое преимущество:** агент работает с тем же контекстом платформы, что и человек-разработчик. Один MCP-сервис документации подключается к любому AI-инструменту — Claude, Codex, Cursor, OpenCode и другим.

# Roadmap: 4 этапа реализации

Параллельное выполнение 0a | 0b

## Phase 0a

### Инфраструктура

- MCP Gateway & Auth (Gtk JWT)
- Data Tools (Read-only)
- System Tools implementation
- LiteLLM + Langfuse setup
- Агентские пользователи

#### Result

Внешний доступ MCP-клиентов

## Phase 0b

### Семантика

- AVM Docs (50 выборок)
- M-schema v1 (50 классов)
- Автодискавери инструментов
- PGVector + RAG Индекс
- Первая RAG-индексация

#### Result

AI «понимает» бизнес-контекст

## Phase 1

### Первые агенты

- Data Tools CRUD операции
- NL-Фильтрация (LLM→SQL)
- Agent Runtime v1 (JVM/K8s)
- Worker v1 (LangGraph loop)
- Чат-интерфейс v1
- React-виджет трассировки
- Smoke-тесты платформы

#### Result

Чат-бот и запросы на естественном языке работают

## Phase 2

### Автономные

- Hook Manager & Event Bus
- Интеграция с BPM
- Human-in-the-Loop механизмы
- Прикладные агенты от аналитиков
- JEXL-скиллы и песочница
- Администрирование агентов
- Бенчмарки v1 & E2E-тесты

#### Result

Агенты выполняют задачи автономно с контролем человека

## Phase 3

### Полная платформа

- Мульти-агентная оркестрация
- GenUI Runtime Renderer
- A2A (Agent-to-Agent) протокол
- Пользовательские агенты
- Агентская разработка
- ML-сервис & Autotune / Replay

#### Result

Саморазвивающаяся платформа с генеративным UI

# Ключевые выводы

Текущее состояние

**Global ERP**

Form-centric monolith

ГЛУБОКАЯ  
ИНТЕГРАЦИЯ

Целевое состояние

**Agent Ready ERP**

Selection / Operation / AdminMeta

≠ Не просто надстройка "Copilot", как у SAP/Oracle

## 5 Архитектурных Принципов



### Переиспользование абстракций

Агенты используют те же механизмы: Selection, DataStore, AdminMeta. Не дублируем логику.



### Только Open Source

LiteLLM, LangGraph, Langfuse, vLLM. Полный контроль, никакого vendor lock-in.



### Провайдер-агностичность

Легкое переключение: OpenAI ↔ Anthropic ↔ GigaChat ↔ Self-hosted модели.



### Безопасность по умолчанию

Агент ≠ Суперпользователь. Соблюдает те же права и RLS, что и люди.



### Инкрементальная доставка

Каждый этап (0a, 0b, 1...) дает работающий бизнес-результат. Не "Big Bang" запуск.



НАЧАТЬ

ЭТАП 0a: ИНФРАСТРУКТУРА

### Развернуть MCP Gateway + LiteLLM

Обеспечить **безопасный доступ на чтение** данных Global3 для внешних AI-клиентов (Claude Desktop, ChatGPT, IDE).

🔧 MCP Gateway 🔑 Gtk JWT Auth 🌐 LiteLLM Proxy 🔍 Langfuse Tracing

Первый конкретный шаг

Открывает доступ к данным ERP для существующих AI-инструментов **без разработки полного Agent Runtime.**