АСТРА ИИ Цифровой офис (ASTRA AI Digital Office)

Техническое описание

Москва, 2026

Введение

АСТРА ИИ Цифровой офис (ASTRA AI Digital Office) (далее — комплекс, продукт) представляет собой корпоративный AI-ассистент, предназначенный для интеллектуальной обработки документов и автоматизации рутинных задач сотрудников организации. Продукт обеспечивает взаимодействие пользователей с корпоративной базой знаний посредством чата на основе технологии retrieval-генеративного дополнения (RAG), а также предоставляет инструменты генерации структурированных артефактов: суммаризаций, презентаций и дашбордов. Отдельный класс задач охватывает специализированные модули, в первую очередь автоматизированную проводку универсальных передаточных документов (УПД) в системе 1С.

Продукт ориентирован на применение в средних и крупных организациях, использующих большие объёмы неструктурированной документации: договорной, технической, бухгалтерской, отчётной. Целевая аудитория включает сотрудников, работающих с документами (юристы, бухгалтеры, инженеры, руководители подразделений), а также администраторов, управляющих доступом и конфигурацией системы.

Цикл обработки данных. Работа с системой строится по замкнутому циклу, включающему следующие этапы. На этапе загрузки пользователь помещает исходные документы в файловое хранилище через веб-интерфейс. На этапе индексации сервис препроцессинга извлекает текст из документов (включая распознавание сканированных страниц и расшифровку аудио- и видеоматериалов), выполняет сегментацию на фрагменты, вычисляет векторные представления и помещает их в векторную базу данных. На этапе взаимодействия пользователь формулирует запрос в чате; оркестратор агента резолвирует контекст беседы, выполняет поиск релевантных фрагментов и передаёт их языковой модели вместе с инструментами генерации. На этапе генерации артефактов система создаёт запрошенный результат (текстовый ответ, документ в формате Markdown, HTML-презентацию или HTML-дашборд), сохраняет его в хранилище и возвращает пользователю. Специализированные модули реализуют целевые процессы, такие как сверка и проводка УПД в 1С, с пошаговым контролем статусов.

Принципы проектирования. Архитектура комплекса построена на следующих принципах.

- **Контейнеризация.** Все компоненты поставляются и выполняются в виде Docker-контейнеров, что обеспечивает воспроизводимость окружения и изоляцию зависимостей.
- **Микросервисная декомпозиция.** Функциональность разделена на независимые сервисы с чётко определёнными границами ответственности, взаимодействующие через синхронный REST и асинхронный обмен сообщениями.
- **Масштабируемость.** Комплекс допускает горизонтальное масштабирование отдельных сервисов в кластере Kubernetes; векторная база данных и объектное хранилище поддерживают увеличение объёмов данных.
- **Развёртывание on-premise и в частном облаке.** Система не зависит от внешних облачных сервисов для хранения данных; все компоненты могут быть размещены в инфраструктуре заказчика. Внешние модели машинного обучения обслуживаются совместимыми с OpenAI API инференс-серверами внутри периметра.

- **Безопасность.** Применяется сквозное шифрование файлов в покое (AES-256-GCM), двухтокенная модель аутентификации с проверкой сеанса на уровне шлюза, ролевая модель управления доступом (RBAC) и изоляция данных пользователей на уровне базы данных и векторного индекса.
- **Открытость интеграций.** Интеграция с внешними системами (1С, модели обработки документов) реализована через выделенный сервис-оркестратор, централизующий все исходящие вызовы.

Типы обрабатываемых данных

Комплекс обрабатывает несколько категорий данных, каждая из которых характеризуется собственным набором задач и методов обработки. Категории охватывают текстовые документы, мультимедийные материалы, историю чатов, юридические и бухгалтерские документы, а также векторные представления, формируемые в процессе индексации.

Текстовые документы

Текстовые документы составляют основной объём обрабатываемых данных. Система поддерживает форматы PDF (как с текстовым слоем, так и сканированные), DOCX и TXT. Парсинг PDF с текстовым слоем выполняется средствами pdfminer.six с рекурсивным обходом текстовых элементов. Сканированные PDF классифицируются маршрутизатором на основе анализа первых страниц и обрабатываются с применением оптического распознавания символов. Документы DOCX разбираются с извлечением параграфов, таблиц и структуры заголовков. Результат извлечения кэшируется в объектном хранилище для повторного использования.

Тип задачи	Описание
Извлечение текста из PDF с текстовым слоем	Парсинг средствами pdfminer.six; рекурсивный обход символов и текстовых блоков с сохранением порядка чтения
Распознавание сканированных PDF (OCR)	Классификация страниц, рендеринг в PNG (300 DPI) и пакетное распознавание моделью GLM-OCR; поддержка гибридных документов с частичным текстовым слоем
Разбор документов DOCX	Извлечение параграфов и таблиц с построением иерархического пути секций на основе стилей заголовков
Чтение текстовых файлов	Прямое декодирование TXT и Markdown в UTF-8 с определением кодировки
Сегментация на фрагменты	Токенизация (tiktoken, модель cl100k_base) с размером фрагмента 512 токенов и перекрытием 64 токена; разбиение с учётом границ предложений

Аудио и видео

Мультимедийные материалы в форматах MP3, MP4 и WAV обрабатываются для получения текстовых транскрипций. Нормализация исходного потока выполняется средствами ffmpeg: конвертация в монофонический PCM WAV с частотой дискретизации 16 кГц и фильтром верхних частот. Распознавание речи осуществляется моделью Qwen3-ASR с поканальной обработкой фрагментов длительностью 60 секунд.

Тип задачи	Описание
Нормализация медиапотока	Конвертация mp3/mp4/wav в mono 16 кГц PCM s16le WAV средствами ffmpeg с фильтром highpass и ресемплингом
Распознавание речи (ASR)	Транскрибирование WAV-фрагментов моделью Qwen3-ASR-1.7B через OpenAI-совместимый API; параллельная обработка до 16 фрагментов одновременно
Сегментация аудио	Разбиение нормализованного потока на 60-секундные фрагменты для пакетной транскрипции
Извлечение текста транскрипции	Объединение распознанных фрагментов с сохранением временных меток (start_ms, end_ms) в качестве локаторов

Изображения

Изображения обрабатываются преимущественно в составе сканированных PDF-документов. Распознавание выполняется мультимодальной моделью GLM-OCR, принимающей изображения в формате base64 data URI. Страницы документа рендерятся в растровый формат с разрешением 300 DPI; обработка ведётся пакетами с автоматическим повтором при сбоях. Максимальный объём документа составляет 500 страниц.

Тип задачи	Описание
Рендеринг страниц	Преобразование страниц PDF в PNG-изображения разрешением 300 DPI средствами pdf2image (poppler)
Распознавание текста на изображениях	Пакетное распознавание моделью GLM-OCR (batch 5, 3 повторные попытки с экспоненциальной задержкой)
Обработка гибридных документов	Распознавание только страниц без текстового слоя; сохранение исходного текста для страниц с текстом
Слияние результатов	Объединение распознанного и исходного текста с восстановлением сквозной нумерации страниц

Чаты и сообщения

История переписки пользователей индексируется для обеспечения контекстного поиска по содержанию бесед. Каждое сообщение сохраняется в реляционной базе данных с привязкой к чату, моделью и временными метаданными. После завершения цикла обработки агентом история чата

индексируется в векторной базе с детерминированными идентификаторами фрагментов, что обеспечивает идемпотентность повторной индексации.

Тип задачи	Описание
Индексация истории чата	Сегментация текста сообщений, вычисление эмбеддингов и сохранение в векторную базу с привязкой к chat_id и message_id
Поиск по контексту беседы	Векторный поиск релевантных фрагментов истории с фильтрацией по идентификатору чата
Генерация заголовков чатов	Автоматическое формирование краткого заголовка (3–7 слов) на основе первого сообщения пользователя
Хранение вложений	Сохранение ссылок на артефакты (документы, презентации, дашборды) в составе сообщений в формате JSONB

Юридические и бухгалтерские документы

К данной категории относятся универсальные передаточные документы (УПД), акты выполненных работ (КС-3) и XML-документы, импортируемые из системы 1С. Обработка включает извлечение структурированных полей, сверку номенклатуры, выявление расхождений и формирование вариантов устранения проблем с пошаговым контролем статуса.

Тип задачи	Описание
Обработка УПД	Извлечение полей документа, выявление проблем и вариантов их устранения, отслеживание шагов и статусов проводки
Обработка XML-документов 1С	Парсинг XML с сохранением извлечённых полей в формате JSONB; связывание документов с УПД
Сверка номенклатуры	Запрос каталога номенклатуры из 1С и проверка соответствия позиций документа
Поиск XML без привязки к УПД	Выборка документов, не сопоставленных с универсальными передаточными документами
Анализ расхождений (DIT)	Запрос к внешней модели DIT для получения вариантов устранения выявленных проблем

Векторные представления

Векторные представления формируются на этапе индексации и используются при поиске релевантных фрагментов в процессе ответа на запросы пользователя. Модель эмбеддингов multilingual-

e5-large формирует векторы размерностью 1024 с использованием префиксных конвенций: префикс passage для индексируемых фрагментов и префикс query для поисковых запросов. Хранение векторов осуществляется в векторной базе данных Qdrant с метрикой косинусного сходства; каждому пользователю соответствует отдельная коллекция.

Тип задачи	Описание
Вычисление эмбеддингов документов	Пакетная обработка фрагментов (batch 64) моделью multilingual-e5-large с префиксом passage
Вычисление эмбеддингов запросов	Формирование вектора поискового запроса с префиксом query
Векторный поиск	Запрос к коллекции Qdrant с фильтрацией по file_id и chat_id; возврат топ-релевантных фрагментов
Индексация артефактов	Извлечение текста из сгенерированных HTML-артефактов и индексация для последующего retrieval
Управление коллекциями	Автоматическое создание коллекции пользователя при первом добавлении данных; идемпотентный upsert с детерминированными идентификаторами (UUIDv5)

Архитектура

Архитектура комплекса построена по микросервисному принципу и реализует разделение на три функциональных компонента: бэкенд бизнес-логики, компонент искусственного интеллекта (DS) и графический интерфейс (фронтенд). Каждый компонент представлен набором независимых сервисов, упакованных в Docker-контейнеры и разворачиваемых в кластере Kubernetes.

Контейнерная модель

Все сервисы поставляются в виде Docker-образов, собираемых из отдельных Dockerfile в репозиториях каждого сервиса. Для сервисов бэкенда базовым образом служит snakepacker/python:3.10; сервисы DS используют Python 3.12 (сервис препроцессинга — Python 3.11). Фронтенд собирается в двухстадийной сборке на базе node:20-alpine с standalone-выводом. Для части сервисов DS и бэкенда предусмотрена опция сборки защищённого образа (Dockerfile_security) с компиляцией в standalone-исполняемый файл без поставки исходного кода в runtime-слое.

Оркестрация контейнеров в продуктивном окружении осуществляется средствами Kubernetes через Helm-чарт baum-ai-platform. На каждый сервис создаётся отдельный ресурс Deployment с сервисом типа ClusterIP. Шлюзовой трафик принимается nginx-контейнером, экспонируемым через NodePort. Для каждого окружения (разработка, тестирование, продуктивные стенды) предусмотрен отдельный файл конфигурации значений Helm.

Компоненты комплекса

Бэкенд реализован на языке Python 3.10 с использованием веб-фреймворка FastAPI 0.135.1 и сервера Uvicorn 0.41.0. Валидация данных обеспечивается Pydantic 2.12.5. Бэкенд включает семь сервисов: сервис аутентификации (auth-service), справочный сервис (general-service), менеджер файлового хранилища (fs-manager-service), сервис управления чатами (secretary-service), сервис внешних интеграций (api-external-service), сервис интеграции с 1С (one-c-dit-service) и агрегатор OpenAPI-спецификаций. Каждый сервис наследует единый базовый класс, обеспечивающий создание FastAPI-приложения, регистрацию REST-ресурсов и обработчиков Kafka, инициализацию отслеживания ошибок и запуск миграций базы данных.

Компонент искусственного интеллекта (DS) реализует агентную платформу на основе протокола Model Context Protocol (MCP) и состоит из четырёх сервисов: сервера агента (agent-server), MCP-сервера (mcp-server), сервиса модулей (tool-api) и сервиса препроцессинга (file-processing). Сервисы используют FastAPI 0.116.1 и фреймворк FastMCP 3.2.3. Внешние модели машинного обучения (языковая модель MiniMax-M2.7, модель эмбеддингов multilingual-e5-large, модель OCR GLM-OCR, модель ASR Qwen3-ASR-1.7B) обслуживаются совместимыми с OpenAI API инференс-серверами и не входят в состав репозиториев комплекса.

Графический интерфейс реализован на платформе Next.js 15.5.18 (App Router) с библиотекой React 19.1.0 и компонентной системой MUI v7. Управление состоянием и кэшированием данных осуществляется через Redux Toolkit и RTK Query. Структура кодовой базы следует методологии Feature-

Sliced Design с разделением на слои: настройки приложения (app), представления (views), виджеты (widgets), функциональности (features), сущности (entities) и общие модули (shared). Направление зависимостей строго нисходящее — от верхних слоёв к нижним.

Взаимодействие сервисов

Синхронное взаимодействие между фронтендом и бэкендом осуществляется через REST API за nginx-шлюзом. Внутреннее взаимодействие между сервисами бэкенда реализовано через асинхронный обмен сообщениями Kafka по шаблону request/reply: сервис-инициатор публикует сообщение в топик, консьюмер обрабатывает его и возвращает ответ в индивидуальный топик ответа. Этот паттерн реализован через примитив emit_and_wait_for_answer с корреляцией по уникальному идентификатору запроса.

Все исходящие вызовы во внешние системы (инференс-серверы DS, 1С, модель DIT) централизованы в сервисе api-external-service, который является единственной точкой egress. Бизнес-сервисы не обращаются к внешним системам напрямую; вместо этого они эмитят Kafka-события, обрабатываемые сервисом интеграций. Контекст пользователя передаётся в заголовках сообщений в виде JWT без срока действия, что обеспечивает сквозную передачу идентичности.

Канал взаимодействия	Протокол	Назначение
Фронтенд — nginx-шлюз	HTTP (REST, cookies)	Доступ к API бэкенда с прозрачной аутентификацией
nginx-шлюз — сервисы бэкенда	HTTP (REST, JWT Bearer)	Прокси-передача запросов с внедрением краткосрочного JWT
Между сервисами бэкенда	Kafka (request/reply)	Асинхронные вызовы между микросервисами
Бэкенд — DS (agent-server)	HTTP (REST, API-key + JWT)	Вызов оркестратора агента и генерация заголовков чатов
Бэкенд — DS (file-processing)	HTTP (REST, API-key + JWT)	Запуск препроцессинга и векторного поиска
DS — векторная БД (Qdrant)	HTTP (REST)	Операции upsert и поиска векторов
Бэкенд — 1С	HTTP (REST, Basic Auth)	Сверка номенклатуры и обновление XML-документов
Бэкенд — модель DIT	HTTP (REST, API-key)	Получение вариантов устранения расхождений

Шлюз аутентификации

Nginx-шлюз реализует механизм единого входа на основе директивы auth_request. Для каждого входящего запроса шлюз определяет по карте политик, является ли маршрут публичным. Публичные маршруты (health-check, документация API, вход и регистрация) проходят без проверки. Для

защищённых маршрутов шлюз выполняет внутренний подзапрос к сервису аутентификации, который валидирует сеансовый токен из cookie, проверяет fingerprint клиента и выпускает краткосрочный JWT (срок действия 900 секунд). Полученный JWT внедряется в заголовок Authorization и проксируется в целевой сервис. Заголовок с JWT исключается из ответа, возвращаемого клиенту. Такой подход избавляет фронтенд от необходимости хранения токенов доступа: единственным источником сеанса служит httpOnly-cookie.

Конфигурации развёртывания

Комплекс поддерживает две основные конфигурации развёртывания, различающиеся по топологии размещения компонентов и требованиям к инфраструктуре. Выбор конфигурации определяется масштабом эксплуатации и требованиями к отказоустойчивости.

Одноузловая конфигурация

Одноузловая конфигурация предназначена для развёртывания на отдельной виртуальной машине средствами Docker Compose. Все сервисы бэкенда, DS-компонента, фронтенд и инфраструктурные компоненты (PostgreSQL, Redis, MinIO, Kafka, Qdrant) выполняются в контейнерах на одном узле. Компоненты объединены общей сетью; шлюз nginx принимает трафик на едином порту.

Данная конфигурация подходит для пилотного внедрения, демонстрационных стендов и небольших рабочих групп. Ресурсные требования к узлу определяются совокупным потреблением памяти и процессора всех контейнеров; ориентировочные значения приведены в таблице ниже. Дисковое пространство выделяется под данные PostgreSQL, объектное хранилище MinIO, журналы Kafka и векторную базу Qdrant. Инференс моделей машинного обучения может осуществляться на отдельном узле с GPU или на CPU-сервере в зависимости от требований к latency.

Спецификации оборудования (одноузловая конфигурация)

Компонент	Минимум	Рекомендуется
CPU	8 ядер	16 ядер
ОЗУ	16 ГБ	32 ГБ
Диск	100 ГБ SSD	500 ГБ SSD
GPU	опционально (для локального vLLM)	1× GPU 24 ГБ VRAM

Кластерная конфигурация

Кластерная конфигурация реализуется в кластере Kubernetes через Helm-чарт baum-ai-platform. Каждый сервис разворачивается как независимый ресурс Deployment с собственным сервисом ClusterIP. Шлюз nginx экспонируется через NodePort. Векторная база Qdrant и инфраструктурные компоненты (Kafka, PostgreSQL, Redis, MinIO) разворачиваются как отдельные ресурсы кластера. Журналы сервисов сохраняются на томах NFS.

Кластерная конфигурация обеспечивает горизонтальное масштабирование отдельных сервисов путём увеличения числа реплик, изоляцию сбоев на уровне подов и независимое обновление компонентов. Рекомендуемая топология включает отдельные узлы для сервисов приложения, инфраструктурных компонентов и (опционально) узлы с GPU для инференса моделей. На GPU-узлах выполняются серверы vLLM, обслуживающие языковую модель, модель эмбеддингов, модель OCR и модель ASR.

Спецификации оборудования (кластерная конфигурация, минимум 3 узла)

Узел	Роль	CPU	ОЗУ	Диск	GPU
Узел 1	Сервисы приложения (be + ds + fe)	16 ядер	32 ГБ	200 ГБ SSD	—
Узел 2	Инфраструктура (PG, Redis, Kafka, S3, Qdrant)	8 ядер	16 ГБ	500 ГБ SSD (NFS)	—
Узел 3	Инференс LLM/OCR/ASR (vLLM)	8 ядер	16 ГБ	100 ГБ SSD	1× GPU 40–80 ГБ VRAM

Параметр	Одноузловая конфигурация	Кластерная конфигурация
Средство оркестрации	Docker Compose	Kubernetes (Helm)
Топология	Все сервисы на одной ВМ	Отдельный Deployment на сервис
Сетевой шлюз	nginx (порт 8090)	nginx (NodePort)
Масштабирование	Вертикальное	Горизонтальное (реплики)
Хранение журналов	Локальные тома	NFS storage class
Брокер сообщений	Kafka (Zookeeper)	Kafka (KRaft, Bitnami)
Назначение	Пилот, демо, малые группы	Продуктивная эксплуатация
GPU	Опционально, внешний узел	Опционально, выделенные GPU-узлы

> *Инференс моделей машинного обучения (LLM, эмбеддинги, OCR, ASR) во всех конфигурациях осуществляется внешними серверами vLLM, совместимыми с OpenAI API. Комплекс не включает код обучения или fine-tuning моделей; все модели рассматриваются как внешние компоненты инфраструктуры.*
>
> *GPU требуется только при локальном развёртывании серверов инференса (vLLM); возможно использование внешних API LLM (совместимых с OpenAI API) без GPU на узлах комплекса. В кластерной конфигурации для хранения журналов сервисов и общего (shared) хранилища используется NFS.*

Основные свойства

Комплекс обладает рядом свойств, определяющих его эксплуатационные и функциональные характеристики. Свойства сгруппированы по ключевым аспектам: эффективность, масштабируемость, надёжность и безопасность.

Эффективность. Применение векторного поиска и агентной оркестрации позволяет сократить время получения релевантного ответа из больших объёмов документации. Кэширование результатов извлечения текста в объектном хранилище исключает повторную обработку ранее проиндексированных документов. Пакетная обработка изображений и аудиофрагментов с параллельным выполнением запросов к моделям распознавания обеспечивает высокую пропускную способность препроцессинга.

Масштабируемость. Микросервисная декомпозиция позволяет масштабировать независимо те сервисы, которые испытывают наибольшую нагрузку. Векторная база данных поддерживает увеличение объёмов индекса за счёт создания отдельных коллекций на пользователя. Объектное хранилище обеспечивает горизонтальное расширение дискового пространства. Кластерная конфигурация допускает увеличение числа реплик любого сервиса без изменения кода.

Надёжность. Каждый сервис экспонирует публичный health-check endpoint, используемый средствами оркестрации для контроля готовности и живучести. Миграции схемы базы данных выполняются автоматически при запуске сервиса. Идемпотентный upsert в векторную базу с детерминированными идентификаторами фрагментов гарантирует корректность повторной индексации. Отслеживание ошибок осуществляется через Sentry-совместимый бэкенд (Bugsink) с инициализацией при запуске каждого сервиса. Шлюз аутентификации использует кэш проверки токенов для снижения нагрузки на хранилище сеансов.

Безопасность. Потоковое шифрование файлов в покое осуществляется алгоритмом AES-256-GCM с мастер-ключом, хранимым в переменных окружения. Хэширование паролей выполняется алгоритмом argon2. Двухтокенная модель аутентификации разделяет долгоживущий сеансовый токен (httpOnly-cookie) и краткосрочный JWT доступа, что минимизирует риск компрометации учётных данных. Ролевая модель управления доступом (RBAC) с правами на уровне отдельных endpoints и фильтрацией данных по владельцу обеспечивает многопользовательскую изоляцию. Системные корневые директории файлового хранилища объявлены иммутабельными; удаление файлов реализовано через мягкое удаление (soft-delete) с сохранением истории версий.

Типовые сценарии применения

Комплекс поддерживает следующие типовые сценарии обработки данных и автоматизации задач.

- **RAG-чат по документам.** Пользователь загружает документы в источник знаний, формулирует вопрос в чате; система выполняет векторный поиск релевантных фрагментов и формирует ответ языковой модели с учётом извлечённого контекста.
- **Суммаризация документов.** На основе загруженного документа или набора вложений генерируется структурированный текстовый документ в формате Markdown, сохраняемый как артефакт в хранилище.

- **Генерация презентаций.** По содержанию документов формируется HTML-презентация с навигацией, поддержкой клавиатурного и сенсорного управления, режимом печати и ленивой загрузкой диаграмм (Chart.js).
- **Генерация дашбордов.** На основе данных документа создаётся интерактивный HTML-дашборд; процесс включает стадию планирования структуры с последующей генерацией итогового HTML.
- **Проводка УПД в 1С.** Универсальный передаточный документ обрабатывается с извлечением полей, сверкой номенклатуры, выявлением расхождений, выбором вариантов устранения проблем и пошаговым контролем статуса проводки.
- **Распределение КС-3.** Акты выполненных строительных работ обрабатываются через модуль оркестратора с задачами и статусами; предусмотрен режим ручной обработки.
- **Расшифровка совещаний и видеоматериалов.** Аудио- и видеофайлы нормализуются и транскрибируются; полученная текстовая расшифровка индексируется и становится доступной для чата и суммаризации.
- **Генерация заголовков чатов.** На основе первого сообщения пользователя автоматически формируется краткий заголовок беседы (3–7 слов), отображаемый в боковой панели навигации.
- **Многоинструментальная цепочка агента.** Оркестратор агента выполняет свободный цикл вызова инструментов (до 8 раундов), при необходимости последовательно обращаясь к нескольким модулям (например, суммаризация с последующей генерацией дашборда).
- **Вопрос-ответ по истории чата.** При отсутствии явно заданного инструмента QA-роутер классифицирует тип вопроса; для общих вопросов загружается полная история чата, для конкретных — выполняется retrieval по проиндексированным документам и истории переписки (коллекция Qdrant пользователя содержит и те, и другие).
- **Управление источниками знаний.** Пользователь создаёт источники cross (доступны во всех чатах) и tmp (привязанные к конкретному чату), управляет файлами через drag-and-drop интерфейс; файлы автоматически индексируются для RAG.
- **Управление шаблонами промптов.** Администратор конфигурирует многошаговые пайплайны промптов; каждый шаг шаблона соответствует одному промпту в цепочке, резолвящемуся в запросе к модели.
- **Администрирование доступа.** Управление участниками проектов, ролями и правами в рамках рабочего пространства (build project) с ролевой моделью user/admin.
- **Индексация сгенерированных артефактов.** Созданные презентации и дашборды индексируются в векторной базе (с извлечением текста из HTML), что делает их доступными для последующего retrieval.
- **Версионирование файлов.** Загрузка новых версий документов создаёт цепочку версий с сохранением возможности отката; предыдущие версии сохраняются в хранилище.
- **Мультиязычный интерфейс.** Графический интерфейс поддерживает локали ru и en с переключением через серверное действие и хранением выбора в cookie.

- **Мониторинг брокера сообщений.** Kafka-UI предоставляет веб-интерфейс для инспекции топиков и сообщений брокера.

Функционал программных модулей

Функциональность комплекса реализована совокупностью программных модулей, сгруппированных по подсистемам. Каждая подсистема охватывает связанный набор ответственности и реализуется одним или несколькими микросервисами.

Подсистема хранения данных

Подсистема хранения данных обеспечивает персистентное хранение структурированных данных, сессий, файловых объектов и векторных представлений. В её состав входят PostgreSQL 16, Redis 7, объектное хранилище S3/MinIO и векторная база Qdrant.

PostgreSQL выступает единым реляционным хранилищем с тремя логическими схемами. Схема public содержит таблицы пользователей, рабочих проектов, ролей, прав, логического дерева файлов и версий файлов, каталога моделей и промптов. Схема secretary содержит таблицы источников знаний, чатов, сообщений, функциональностей, шаблонов и настроек шагов. Схема one_c_dit содержит таблицы XML-документов, УПД, проблем и вариантов, шагов и статусов. Миграции схемы выполняются средствами Alembic при запуске каждого сервиса; порядок применения гарантирует создание схемы public перед схемами сервисов. Доступ к данным реализован через SQLAlchemy 2.0 с асинхронным драйвером asyncpg; менеджер с областью видимости пользователя фильтрует записи по идентификатору пользователя и активному рабочему проекту из JWT.

Redis служит хранилищем сеансов аутентификации. Ключи сеансов содержат идентификатор пользователя, идентификатор сеанса и fingerprint клиента; срок действия сеанса составляет 14400 секунд с продлением при каждом обращении. Отдельный кэш проверки токенов (срок 10 секунд) снижает нагрузку на хранилище при обработке повторных запросов.

Объектное хранилище S3/MinIO хранит файловые blob-объекты в плоской структуре. Менеджер файлов оркестрирует согласованное состояние между логическим деревом в PostgreSQL и физическими объектами в S3. Потоковое шифрование AES-256-GCM применяется при записи с размером блока 65536 байт; метаданные шифрования (magic-префикс, версия формата, wrapped DEK, размер чанка) упакованы в бинарный заголовок inline в начале тела зашифрованного объекта (ciphertext). Корневые директории main_fs, source_fs и chat_result_fs создаются при запуске менеджера файлов и обеспечивают разделение исходных документов, источников знаний и результатов.

Векторная база Qdrant хранит эмбеддинги фрагментов документов и истории чатов. Каждому пользователю соответствует отдельная коллекция с метрикой косинусного сходства и размерностью 1024. Коллекции создаются автоматически при первом добавлении данных. Идентификаторы точек генерируются детерминированно (UUIDv5), что обеспечивает идемпотентность операций upsert. Payload точек содержит идентификаторы файла, фрагмента, чата и сообщения, локаторы (номер страницы, путь секции, временные метки) и метаданные конвейера обработки.

Подсистема аутентификации и управления доступом

Подсистема реализует двухтокенную модель аутентификации и ролевое управление доступом. Сервис аутентификации (auth-service) обрабатывает регистрацию, вход, выход, верификацию токенов и управление пользователями, ролями и правами.

При входе в систему пароль пользователя проверяется алгоритмом argon2. При успешной проверке создаётся сеанс: JWT без срока действия (тип session) сохраняется в Redis, а его значение устанавливается в httpOnly-cookie session_token. Для проверки fingerprint вычисляется SHA-256 (бесключевой хэш) от связки user agent, IP-адреса клиента и секретной соли. При каждом запросе к защищённому маршруту nginx-шлюз выполняет внутренний подзапрос верификации: сервис проверяет валидность сеанса и fingerprint, резолвит роли и права пользователя для активного рабочего проекта, выпускает краткосрочный JWT (HS256, срок 900 секунд) и возвращает его в заголовке X-Auth-Jwt. Шлюз перехватывает этот заголовок и внедряет JWT в заголовок Authorization при проксировании запроса в целевой сервис.

Авторизация на уровне сервисов реализована через декларацию прав: каждый обработчик указывает имя и описание требуемого права; при запуске сервиса права синхронизируются с базой данных. Маршруты, не требующие аутентификации (health-check, документация, вход и регистрация), объявляются как публичные. Роли user и admin определяют уровень привилегий; фильтрация данных по владельцу применяется в запросах к базе данных на основе идентификатора пользователя и рабочего проекта из JWT.

Подсистема файлового хранилища

Подсистема файлового хранилища (fs-manager-service) обеспечивает загрузку, хранение, версионирование, мягкое удаление и конвертацию файлов. Логическое дерево файлов хранится в PostgreSQL (таблицы files и file_versions) с поддержкой иерархии директорий, привязки к владельцу и рабочему проекту, атрибутов типа контента и размера. Физические blob-объекты хранятся в S3 в плоской структуре с путём, включающим идентификаторы файла и версии.

Версионирование реализовано через цепочку записей: каждая новая версия файла ссылается на родительскую и первую версию, что обеспечивает навигацию по истории изменений. Мягкое удаление устанавливает флаг deleted без физического удаления объекта. Системные корневые директории защищены от изменения и удаления. Подсистема поддерживает загрузку с заменой, скачивание с опциональной конвертацией форматов (включая Markdown в DOCX) и обслуживание Kafka-событий инициализации рабочих проектов.

Подсистема управления чатами и сообщениями

Подсистема управления чатами и сообщениями (secretary-service) реализует основную бизнес-логику взаимодействия пользователя с AI-ассистентом. Подсистема управляет чатами, источниками знаний, функциональностями, промптами и шаблонами шагов.

Чаты хранятся с текстовым индексом (GIN trgm) для быстрого поиска по названию. Сообщения содержат текст, вложения (JSONB), привязку к модели и промпту, признак источника (от пользователя или от чата) и временные метки; индексы обеспечивают эффективную выборку по чату с сортировкой по времени создания. Отправка сообщения инициирует Kafka-события в DS-компонент: генерацию заголовка чата, обработку сообщения и инициализацию RAG. Контекст пользователя передаётся в заголовках Kafka-сообщения.

Источники знаний различаются по области видимости через булевы флаги модели Sources: cross=True (доступны во всех чатах пользователя) и tmp=True (ассоциированы с конкретным чатом через join-таблицу ChatSourcesUsed). Функциональности (features) представляют типы AI-модулей, доступных пользователю; шаблоны промптов (templates и steps_template) определяют многошаговые пайплайны с разрешением промптов на этапе выполнения. При запуске сервиса выполняется сидирование справочных данных из JSON-шаблонов.

Подсистема искусственного интеллекта

Подсистема искусственного интеллекта (DS-компонент) реализует агентную оркестрацию, MCP-инструментную платформу, генерацию артефактов, препроцессинг и векторный поиск. Подсистема состоит из четырёх сервисов и векторной базы данных.

Сервер агента (agent-server) является единой точкой входа DS-компонента. Оркестратор выполняет свободный цикл вызова инструментов языковой моделью (до 8 раундов) без предварительного статического планирования. Схемы инструментов динамически обнаруживаются через MCP-протокол; оркестратор не хранит предопределённые описания инструментов. Аргументы, принадлежащие бэкенду (идентификаторы промптов, модели, пользователя и чата, тип вопроса), исключаются из схемы, видимой модели, и инжектируются перед вызовом инструмента. QA-роутер выполняет предварительный структурированный вызов классификации: определяется, является ли сообщение вопросом, и его тип (конкретный или общий). При конкретном вопросе выполняется retrieval по проиндексированным документам и истории переписки в коллекции Qdrant пользователя; при общем — загружается полная история чата. При явно заданном целевом инструменте (features_target_name) первый вызов форсируется в этот инструмент, после чего финальный вызов модели без инструментов формирует ответ пользователю. После каждого цикла история чата индексируется в векторной базе. Сервер агента также генерирует заголовки чатов.

MCP-сервер (mcp-server) проксирует вызовы инструментов между сервером агента и сервисом модулей через протокол Model Context Protocol. Сервер публикует четыре инструмента: суммаризацию (summarization), вопрос-ответ (question_answering), генерацию дашборда (dashboard) и генерацию презентации (presentation). Описания и схемы инструментов определяются на уровне MCP-сервера; исполнение делегируется в сервис модулей.

Сервис модулей (tool-api) реализует исполнение инструментов. Сервис считывает промпт-шаблоны из PostgreSQL, загружает документы из S3, вызывает языковую модель, сохраняет сгенерированные артефакты в S3 и индексирует их в Qdrant. Модули дашборда и презентации реализованы как двухстадийные: стадия планирования структуры и стадия генерации итогового артефакта. Промпты резолвятся на основе шаблонов шагов: каждый инструмент имеет основной промпт

и (для двухстадийных модулей) промпт планирования. Сервис также обеспечивает retrieval: эмбеддинг запроса с префиксом query, поиск по коллекции Qdrant с фильтрацией по файлам и чатам, возврат топ-8 релевантных фрагментов с метаданными.

Сервис препроцессинга (file-processing) реализует конвейер извлечения текста и индексации. Сервис обрабатывает PDF (текстовый и сканированный), DOCX, TXT, аудио и видео. Маршрутизатор PDF классифицирует документ по типу (текстовый, сканированный, гибридный) на основе анализа первых страниц. Извлечение текста выполняется соответствующим парсером; для сканированных страниц применяется OCR, для медиафайлов — нормализация ffmpeg и ASR. Результат извлечения кэшируется в S3. Сегментация выполняется токенизатором tiktoken (cl100k_base) с размером фрагмента 512 токенов и перекрытием 64 токена с учётом границ предложений. Эмбеддинги вычисляются моделью multilingual-e5-large (batch 64) и сохраняются в Qdrant. Сервис также обслуживает векторный поиск и удаление фрагментов файла.

Модуль DS-компонента	Функция	Ключевые технологии
agent-server	Оркестрация цикла вызова инструментов, QA-роутинг, генерация заголовков	OpenAI SDK, MCP-клиент, FastAPI
mcp-server	Публикация MCP-инструментов, проксирование вызовов	FastMCP 3.2.3, streamable-http
tool-api	Исполнение модулей, retrieval, индексация артефактов	OpenAI SDK, Qdrant, S3, PostgreSQL
file-processing	Препроцессинг, OCR, ASR, чанкинг, индексация	pdfminer, pdf2image, ffmpeg, GLM-OCR, Qwen3-ASR, e5-large

Подсистема интеграций

Подсистема интеграций (api-external-service) централизует все исходящие вызовы во внешние системы и является единственной точкой egress бэкенда. Подсистема не имеет публичных REST-эндпоинтов (кроме health-check); вся работа выполняется через Kafka-консьюмеры, обрабатывающие три группы запросов.

Запросы к DS-компоненту включают проверки доступности модели и RAG, отправку сообщения агенту (с таймаутом 3600 секунд), генерацию заголовка чата и инициализацию индексации. Запросы к 1С включают проверку доступности, сверку номенклатуры, получение каталога номенклатуры и обновление XML-документа. Запросы к модели DIT включают проверку доступности и получение вариантов устранения проблем. HTTP-клиенты реализованы через адаптер с поддержкой повторных попыток, таймаутов и SSL. Контекст пользователя для вызовов DS передаётся в заголовке X-User-Data-Jwt в виде JWT без срока действия; вызовы к 1С и DIT выполняются без передачи контекста пользователя.

Подсистема специализированных модулей

Подсистема специализированных модулей (one-c-dit-service) реализует обработку универсальных передаточных документов и XML-документов системы 1С. Подсистема хранит XML-документы с извлечёнными полями в формате JSONB, УПД со статусом (enum), проблемы и варианты их устранения (JSONB, связь 1:1 с УПД) и последовательность шагов с статусами (enum, упорядоченные по УПД).

Модуль УПД реализует полный цикл проводки: получение списка УПД, выбор XML-документа, получение данных сверки, получение шагов и статуса, выбор варианта устранения расхождения, обновление статуса шага и обновление XML-документа в 1С. Взаимодействие с 1С и моделью DIT осуществляется через подсистему интеграций. Подсистема использует собственную схему базы данных one_c_dit и обращается к S3 для хранения связанных файлов.

Подсистема графического интерфейса

Подсистема графического интерфейса (фронтенд) реализована на платформе Next.js 15.5.18 с использованием App Router и standalone-вывода сборки. Интерфейс построен на библиотеке компонентов MUI v7 и языке TypeScript в строгом режиме. Управление состоянием осуществляется через Redux Toolkit (RTK Query для кэширования данных сервера, hand-written slices для локального UI-состояния). Поддержка интернационализации обеспечивается библиотекой next-intl с локалями ru и en.

Структура кодовой базы следует методологии Feature-Sliced Design с шестью слоями: настройки приложения (store, providers), представления (полноэкранные компоненты, один на маршрут), виджеты (крупные независимые UI-блоки), функциональности (пользовательские сценарии), сущности (бизнес-объекты и RTK Query-сервисы) и общие модули. Направление зависимостей строго нисходящее; каждый слой экспонирует только именованные экспорты через публичный API. Глубокие импорты во внутренние модули других срезов запрещены и контролируются пользовательским ESLint-плагином.

Графический интерфейс включает два продуктовых раздела. Раздел Platform обеспечивает чат с AI-ассистентом, управление источниками знаний и шаблонами промптов. Раздел Modules содержит специализированные процессы: проводку УПД, распределение КС-3 и суммаризацию видео. Аутентификация на стороне клиента реализована через httpOnly-cookie session_token как единственный источник сеанса; токен доступа на клиенте не хранится. Защита маршрутов обеспечивается серверным middleware (проверка cookie на HTTPS) и клиентским AuthGuard (на основе запроса текущего пользователя). Чат реализован синхронно через REST API с optimistic-UI паттерном для ожидающих сообщений.

Подсистема логирования и мониторинга

Подсистема логирования и мониторинга обеспечивает отслеживание ошибок, контроль работоспособности сервисов и инспекцию асинхронных сообщений. Отслеживание ошибок реализовано через Sentry SDK 2.58.0 с self-hosted Sentry-совместимым бэкендом Bugsink. Инициализация отслеживания выполняется при запуске каждого сервиса; конфигурация задаётся через переменные окружения (DSN, окружение, релиз, имя сервера).

Контроль работоспособности реализован через публичные health-check endpoints. Каждый сервис экспонирует маршрут /api/health/, не требующий аутентификации. Средства оркестрации (Docker Compose и Kubernetes) используют эти endpoints для проверок готовности (readiness) и живучести (liveness) с начальной задержкой 20–30 секунд. Агрегатор OpenAPI-спецификаций дополнительно проверяет доступность каждого сервиса через отдельный endpoint на порту 8099. Шлюз nginx возвращает 200 ok на маршруте /health, защищённом через auth_request.

Инспекция асинхронных сообщений обеспечивается через Kafka-UI, предоставляющий веб-интерфейс для просмотра топиков, консьюмер-групп и сообщений брокера. Брокер сообщений Kafka в кластерной конфигурации работает в режиме KRaft (без Zookeeper) с использованием дистрибутива Bitnami. Журналы сервисов в кластерной конфигурации сохраняются на томах NFS с настраиваемым размером выделенного хранилища.