

Корпоративный агентный портал

Функциональная концепция и техническая архитектура

Параметр	Значение
Версия / дата	0.1 / 02 октября 2026 года
Статус	Проект для обсуждения и последующего согласования
Объект	Портал, подключение моделей, агентные сценарии и эксплуатация
Граница	Развёртывание, обучение и GPU-инфраструктура моделей вне проекта
Документ	Описание целевой системы

Принцип: пользователь собирает помощника из разрешённых возможностей; платформа обеспечивает доступ, бюджет, выполнение и проверяемый результат.

Сроки, ресурсы и целевые показатели являются проектными оценками. Функциональность не объявляется уже реализованной. Выбор агентного runtime подтверждается сравнительными испытаниями.

Содержание

1. Назначение и основные решения	3
2. Границы проекта и этапы поставки	4
3. Пользователи и рабочий интерфейс	6
4. Каталог функциональных требований	8
5. Технологический стек и альтернативы	10
6. Логическая архитектура и компоненты	12
7. Данные и основные контракты	14
8. Подключение внутренних и внешних моделей	16
9. Идентификация, LDAP и локальные группы	17
10. Пространства, роли и политика доступа	18
11. Лимиты, бюджетные счета и учёт	19
12. Аналитика расхода, использования и совместного доступа	21
13. Выбор агентного runtime и опыт Anthropic	23
14. Экономика токенов и сравнительные испытания	25
15. Помощники и конструктор сценариев	27
16. Жизненный цикл задания и восстановление	28
17. Веб-поиск и контролируемый fetch	30
18. Работа с документами и файлами	31
19. RAG и базы знаний	32
20. Код и изолированные рабочие среды	33
21. MCP, инструменты и плагины	34
22. SMTP и уведомления	35
23. Единая администрация и оркестраторы	36
24. Сетевая архитектура и матрица доступов	38
25. Хранилища, секреты и сроки хранения	41
26. Развёртывание и отказоустойчивость без обязательного Kubernetes	42
27. Сайзинг серверов приложения	43
28. Offline-установка, обновление и восстановление	46
29. Наблюдаемость, аудит и безопасность	47
30. Проверки и критерии приёмки	48
31. Порядок реализации и результаты этапов	50
32. Риски, предположения и открытые решения	52
33. Источники и терминология	53
34. Дизайн портала и панели управления	55

1. Назначение и основные решения

Проект создаёт внутренний корпоративный портал, через который сотрудники работают с ИИ, собирают помощников, запускают агентные сценарии, обрабатывают документы и выполняют задания с кодом. Пользователь получает единое рабочее пространство. Администратор управляет подключениями, правами, ресурсами, лимитами и эксплуатацией.

Портал поддерживает модели в корпоративной сети, модели на других доступных серверах, внешних провайдеров и прямые API-подключения. Для приложения модель является внешним сервисом с проверенным контрактом. Проект не разворачивает модельные серверы, не выбирает GPU и не обучает модели.

Предлагаемый стек: Go для корпоративного API; React и TypeScript для интерфейса; PostgreSQL для транзакционных данных; S3-совместимое хранилище для файлов; Python workers для готовых агентных runtime; Temporal для фоновых заданий. Базовое развёртывание — несколько VM с Docker Compose и Ansible. Nomad или Kubernetes рассматриваются отдельно при необходимости автоматического размещения контейнеров.

Решение о конкретном harness принимается после сравнительных испытаний. LangGraph рассматривается как кандидат для управляемых сценариев, Deep Agents — для открытых сложных задач. Для обычного чата применяется прямой вызов модельного API. Экономичность измеряется стоимостью принятой задачи, а не наличием определённой библиотеки.

Что получает организация

- Контролируемый доступ сотрудников к разрешённым модельным подключениям и инструментам.
- Помощники и сценарии, которые можно создавать, версионировать и публиковать в рабочих пространствах.
- Личные и общие бюджеты, панель аналитики и прозрачный учёт всех модельных вызовов.
- Разделение доступа к документам, выполнения кода и исходящих интернет-соединений.
- Воспроизводимая установка, наблюдаемость, восстановление заданий и резервное копирование.

Документ является проектом целевой системы. Обязательные требования пользователя отделяются от предлагаемых решений; сроки, численные ограничения и ресурсы подлежат проверке на пилоте.

2. Границы проекта и этапы поставки

Подтверждённые требования

Закрытая инфраструктура с разрешением только необходимых соединений; отдельный интернет-доступ сервиса веб-поиска; отсутствие прямого интернет-доступа у `harness workers` и песочниц; гибкое подключение внутренних и внешних моделей; интеграция с LDAP; локальные пользователи и группы; лимиты пользователей и групп; SMTP по разрешению; документы, код, RAG, MCP и расширения; автоматизированная установка; распределённая эксплуатация; масштаб от 100 до 3000 пользователей.

Внешние модели требуют отдельного разрешённого сетевого пути. Ограничение интернет-доступа `workers` не отменяет разрешённую передачу данных модельному шлюзу. Политика определяет, какие данные конкретное подключение вправе получить.

Входит в проект	Отдельная задача или внешняя зависимость
Каталог модельных подключений, адаптеры API, секреты и проверка совместимости	Развёртывание и настройка inference-серверов
Учёт потребления, импорт цен, аналитика и лимиты	Обучение, fine-tuning и подбор конкретной модели
Интеграция готового <code>harness</code> и сред выполнения	Разработка собственного универсального агентного цикла
Получение embeddings через зарегистрированный API для RAG	GPU-инфраструктура и размещение embedding-моделей
Подготовка и установка приложения в закрытой сети	Корпоративная PKI, физическая сеть и гипервизоры
Управление подключениями к LDAP, SMTP, Git, MCP и S3	Эксплуатация исходных корпоративных систем

Обозначения релизов

R1 — корпоративный пилот: аутентификация, модельные подключения, чат, базовый конструктор помощника, общий бюджет группы, базовая аналитика, документы, один управляемый сценарий, отдельный поисковый сервис, аудит и offline-установка.

R2 — функциональная версия: RAG с ACL, визуальные сценарии, согласования, код в песочнице, MCP, SMTP, реестр расширений, управление расписаниями и глобальными настройками.

R3 — распределённая эксплуатация: HA, масштабирование workers, проверка нагрузки, аварийное восстановление, upgrade/rollback, эксплуатационные регламенты и передача поддержки.

Новый провайдер или формат документа становится поддерживаемым после контрактных тестов. Формулировка «различные модели» не означает автоматическую совместимость со всеми существующими API.

3. Пользователи и рабочий интерфейс

Основной интерфейс — рабочее пространство со списком помощников, сценариев, заданий и документов. Пользователь видит выбранное модельное подключение, статус задания, использованные источники, результаты и свой доступный бюджет. Интерфейс показывает рабочие события и результаты инструментов; скрытые рассуждения модели не являются обязательным объектом отображения или хранения.

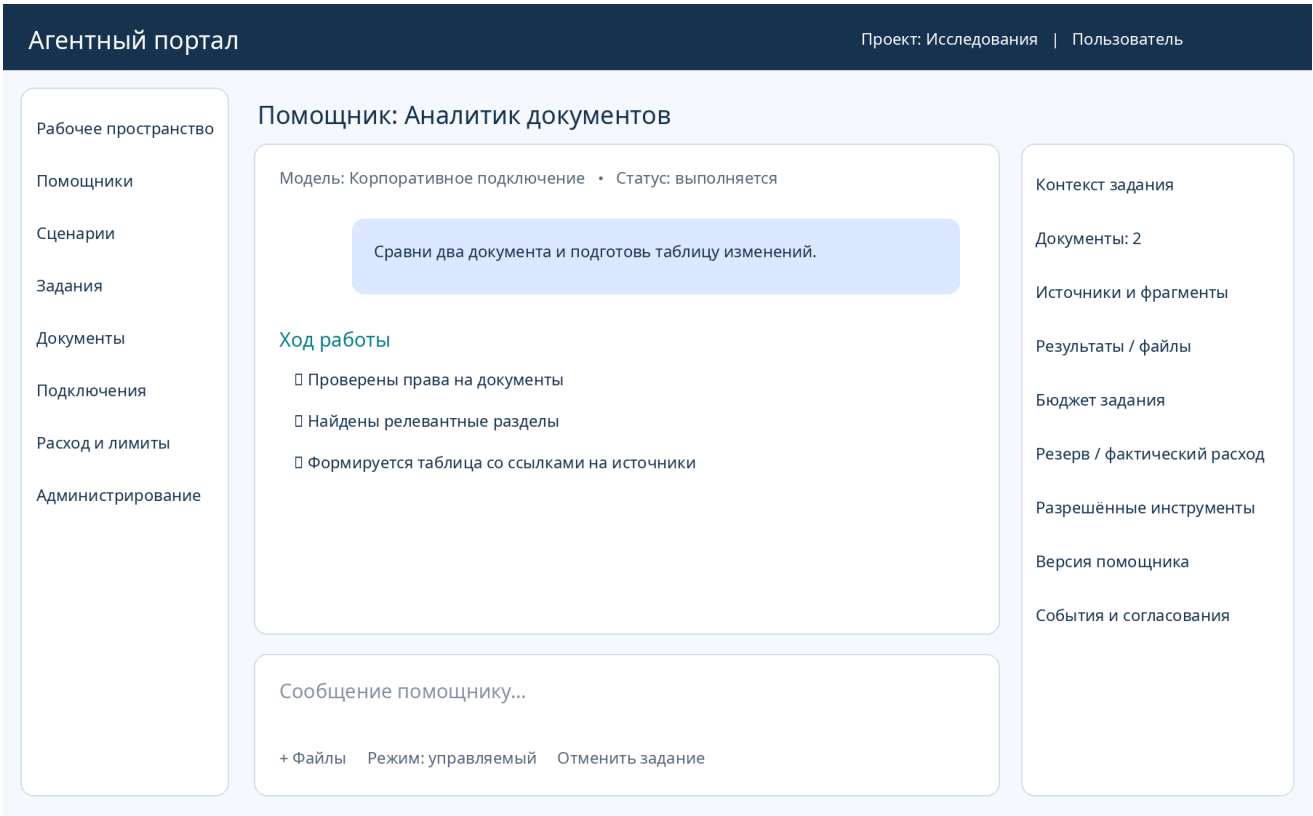


Рисунок 1. Предлагаемая структура рабочего пространства; это концептуальная схема, а не готовый интерфейс.

Роль	Основная работа	Доступные действия
Сотрудник	Чат, документы и выполнение задач	Выбор разрешённого помощника, просмотр своих заданий и расхода
Автор помощников	Настройка инструментов и инструкций	Черновик, тестовый запуск, версия, публикация в разрешённую область
Руководитель пространства	Управление командой и общим бюджетом	Локальные группы, приглашения, отчёты, лимиты в пределах делегированных прав
Администратор платформы	Подключения и глобальные политики	Каталог API, инструментов, шаблонов, правила доступа и конфигурация
Оператор	Эксплуатация	Состояние сервисов, очереди, workers, обновления и восстановление

Роль	Основная работа	Доступные действия
Аудитор	Проверка событий	Чтение разрешённого аудита без автоматического доступа к содержимому документов

Основные экраны

Чат и рабочее пространство; библиотека помощников; редактор помощника; редактор сценария; список заданий; карточка задания с событиями и артефактами; документы и базы знаний; модельные подключения; аналитика расхода, использования и совместного доступа; согласования; глобальное администрирование; состояние инфраструктуры; журнал аудита.

Предусматриваются поиск, фильтры, доступность с клавиатуры и адаптация под ноутбук. Для сложного редактирования сценариев приоритетен desktop-интерфейс. Ошибка сообщает понятную причину: отсутствуют права, исчерпан бюджет, не поддерживается возможность API, недоступен инструмент или задача остановлена.

4. Каталог функциональных требований

Требования имеют стабильные идентификаторы. В плане реализации и приёмке используются те же обозначения. Релиз в таблице — план включения в поставку, а не отметка уже выполненной работы.

ID	Функция	Требуемое поведение	Релиз
F01	Вход и LDAP	OIDC, LDAP-группы, локальные учётные записи, блокировка доступа	R1
F02	Пространства	Личная и командная область, роли и ACL	R1
F03	Подключения моделей	Внутренние, внешние и прямые API; секреты и health-check	R1
F04	Каталог возможностей	Проверенная поддержка tools, JSON, streaming и контекста	R1
F05	Чат	Потоковый ответ, история, файлы, восстановление просмотра	R1
F06	Конструктор помощника	Инструкция, модель, документы, инструменты, профиль выполнения	R1
F07	Версии и публикация	Черновик, неизменяемая версия, область видимости, архивирование	R1
F08	Пользовательские лимиты	Расход, параллельность, длительность и размер данных	R1
F09	Общие бюджеты	Групповой счёт, личный предел внутри него, резервирование	R1
F10	Панель аналитики	Расход, использование моделей/tools/документов, совместный доступ и бюджеты	R1 → R2
F11	Задания	Очередь, статус, отмена, события, артефакты и причины ошибки	R1
F12	Готовый runtime	Интеграция выбранной библиотеки без собственного agent loop	R1
F13	Сценарии	Управляемые шаги, условия, ограниченные циклы и версии	R1 → R2
F14	Согласования	Подтверждение человеком перед разрешённым внешним действием	R2
F15	Веб-поиск	Отдельный search/fetch сервис и контролируемая передача запросов	R1

ID	Функция	Требуемое поведение	Релиз
F16	Документы	Загрузка, разбор, результаты, источники и удаление	R1
F17	RAG	Индексация, поиск по ACL, цитаты и синхронизация удаления	R2
F18	Код	Отдельная песочница, diff, тесты, артефакты и разрешённый Git	R2
F19	MCP	Реестр серверов, права инструментов и выполнение через шлюз	R2
F20	Плагины	Версионизируемые расширения в изолированных сервисах	R2
F21	SMTP	Разрешённая отправка, адресаты, вложения и журнал доставки	R2
F22	Расписания	Периодические задания, сервисная идентичность и бюджет	R2
F23	Глобальная панель	Политики, подключения, лимиты, шаблоны и версии настроек	R1 → R2
F24	Аудит	Автор действия, объект, результат, версия политики и корреляция	R1
F25	Наблюдаемость	Состояние сервисов, метрики, очереди и уведомления	R1 → R3
F26	Offline-поставка	Комплект зависимостей, проверка установки, upgrade/rollback	R1 → R3
F27	Распределённое выполнение	Пулы workers, изоляция, перенос задания и масштабирование	R3
F28	Восстановление	Backup/restore, проверка отказов и эксплуатационные инструкции	R3
F29	Экономическая оценка	Все вызовы, стоимость принятой задачи, сравнение runtime	R1 → R2
F30	Инфраструктурное управление	Просмотр узлов и пулов; ограниченные операции через оркестратор	R3

5. Технологический стек и альтернативы

Предлагаемый стек выбирается по доступности библиотек, понятной эксплуатации в закрытом контуре и возможности заменять компоненты через контракт. В документе не закрепляются плавающие версии: конкретные версии фиксируются после проверки совместимости и включаются в манифест поставки.

В оформлении используются знакомые паттерны современных агентных рабочих пространств: боковая навигация, спокойная типографика, центральная область задачи, поток событий, артефакты и правый контекст. Фирменные логотипы и точные копии интерфейсов других продуктов не требуются. Отдельный интерактивный макет показывает проектный вариант.

Компонент	Основной кандидат	Обоснование и ограничение
Корпоративный API	Go	Транзакции, политики, администрирование и поток событий; агентный цикл не переписывается на Go
Веб-приложение	React + TypeScript + Vite	Статическая поставка без обязательного SSR; редактор сценариев — React Flow после пилота
Runtime workers	Python	Готовые агентные библиотеки, MCP и обработка документов
Хранилище состояния	PostgreSQL	Раздельные БД/роли для портала, IdP, Temporal и checkpoints
Начальный RAG	pgvector + полнотекстовый поиск	Отдельная векторная БД добавляется при подтверждённой необходимости
Файловое хранилище	Корпоративное S3	Версии, артефакты и крупные результаты вне транзакционной БД
Аутентификация	Существующий IdP / Keycloak	OIDC; LDAP и локальные пользователи обслуживаются IdP [S05]
Модельный шлюз	LiteLLM Proxy, кандидат	Проверить редакцию, покрытие API и лицензию нужных функций [S04]
Задания	Self-hosted Temporal	Долгие процессы, таймеры и восстановление; требует эксплуатации сервиса [S03]
Развёртывание	Compose + Ansible	Статические роли на нескольких VM; воспроизводимая установка и обновление [S06, S21]
Входной трафик	Nginx / корпоративный LB	TLS и маршрутизация; не выполняет функции планировщика заданий
Телеметрия	OpenTelemetry + корпоративный стек	Только разрешённые внутренние получатели; без обязательного SaaS

Если команда значительно сильнее в Python, API на FastAPI является допустимой альтернативой Go. Это уменьшает число языков и стоимость первых итераций. Число 3000 зарегистрированных пользователей само по себе не обосновывает переписывание backend.

Для минимизации собственной разработки всего портала можно отдельно оценить Dify. Его готовый конструктор способен сократить часть работ, но корпоративные права, изолированные пространства и лицензионные условия нуждаются в проверке. Текущий проект ориентируется на собственный портал с готовыми runtime, а не на глубокий fork чужого продукта [S15].

Kubernetes не является обязательной зависимостью. В базовом варианте Compose управляет контейнерами каждого сервера, Ansible поддерживает согласованную конфигурацию серверов, а Temporal распределяет задания между доступными workers. Автоматического переноса контейнера с потерянной VM этот вариант не обещает: доступность обеспечивается заранее запущенными репликами и запасом ёмкости.

6. Логическая архитектура и компоненты

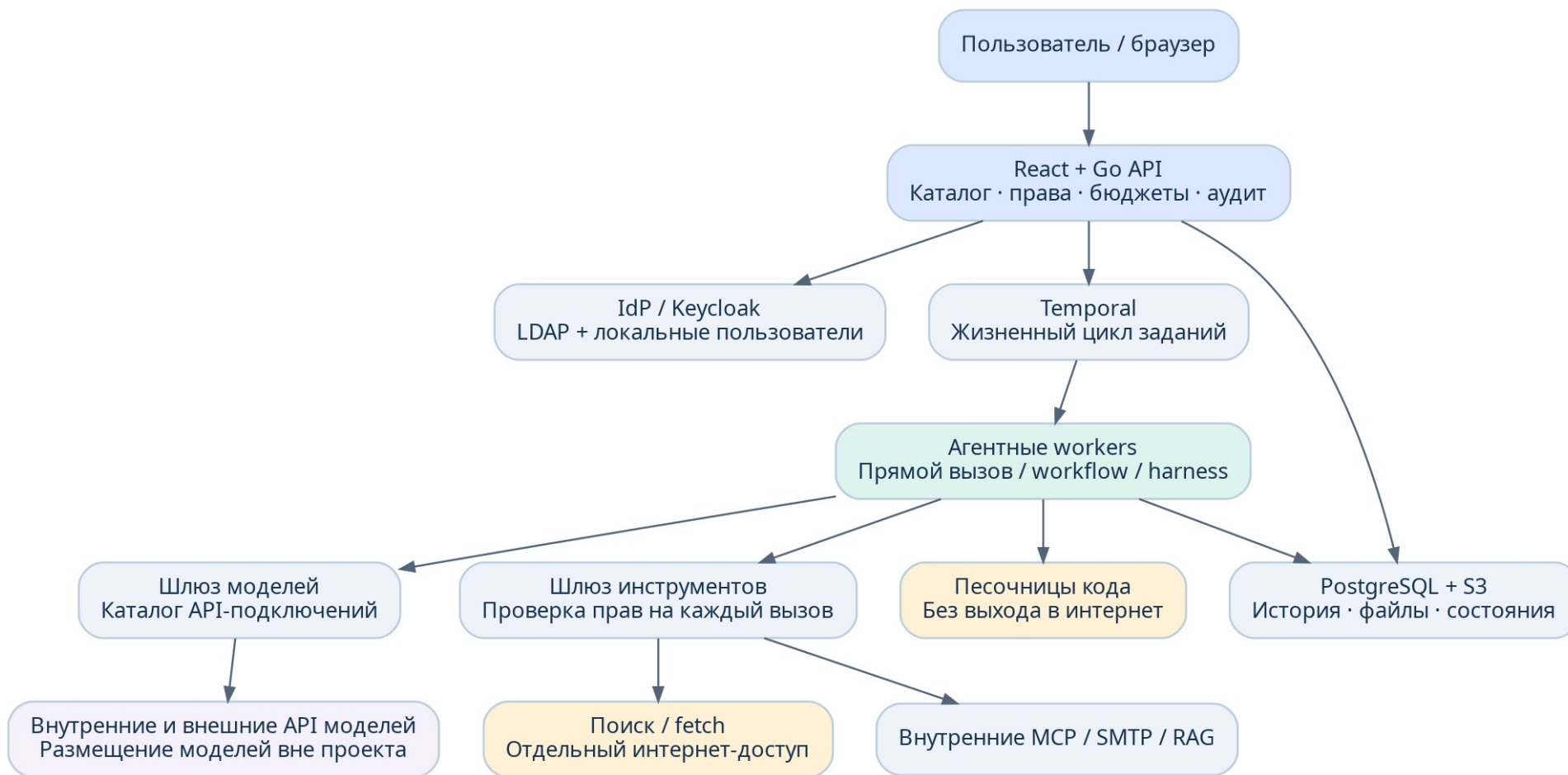


Рисунок 2. Состав портала и связи компонентов. API моделей находится за границей инфраструктуры приложения.

Корпоративный API отвечает за пользователей, пространства, помощников, бюджеты и аудит. На старте это модульное Go-приложение, а не набор мелких микросервисов. Компоненты выделяются в отдельные процессы, когда требуется независимое масштабирование, сетевое ограничение или иной жизненный цикл.

Область	Ответственность	Что не должна делать
Control plane портала	Политики, конфигурация, каталоги, права и диспетчеризация	Исполнять пользовательский shell в процессе API
Оркестратор заданий	Состояния, таймеры, ожидание событий и доставка работы	Самостоятельно определять доступ пользователя к документам
Runtime workers	Вызовы выбранного harness и обработка шагов	Обходить модельный или инструментальный шлюз
Модельный шлюз	API-адаптеры, маршрутизация, usage и admission по бюджету	Незаметно менять внешнее/внутреннее направление данных
Шлюз инструментов	ACL, параметры, согласования, корреляция и секреты	Разрешать произвольный URL или выполнение от имени создателя помощника
Песочницы	Изолированное выполнение кода и файловые операции	Иметь доступ к ключам платформы или всему кластеру
Search/fetch	Поиск и извлечение внешнего содержимого	Служить универсальным HTTP-proxy для workers

Большие результаты передаются ссылками на контролируемые артефакты. События интерфейса используют SSE; выполнение задания не зависит от длительности браузерного соединения. Сервисные соединения аутентифицируются и проходят проверку TLS.

7. Данные и основные контракты

Модель данных должна отражать независимость пользователя, группы доступа, пространства и бюджетного счёта. Членство в нескольких группах не создаёт автоматически несколько источников финансирования одного задания.

Сущность	Основные поля и связи	Правило
User / Group / Membership	IdP subject, тип группы, источник, версия членства	Не использовать изменяемое имя LDAP как единственный идентификатор
Workspace / RoleBinding	Пространство, участник, роль, область действия	У каждого объекта есть владелец и ACL
ModelConnection / ModelProfile	Endpoint, API-тип, secret_ref, capabilities, политика данных	Профиль модели отделён от фактического endpoint
Assistant / AssistantVersion	Инструкция, инструменты, знания, runtime, лимиты	Запущенное задание закрепляет неизменяемую версию
Scenario / ScenarioVersion	Схема шагов, входные/выходные поля, ограничения	Изменение черновика не меняет активное задание
Run / Step / Event	Инициатор, профиль, бюджет, состояние, sequence, trace_id	Состояния сохраняются независимо от UI
BudgetAccount / Reservation / LedgerEntry	Период, валюта/единицы, резерв, расход, ключ операции	Проводки идемпотентны и проверяемы
PriceVersion / Estimate	Прайс, дата действия, источник, входные допущения	Прогноз воспроизводим после изменения тарифа
Document / Chunk / Artifact	Источник, версия, ACL, классификация, ссылка S3	Доступ проверяется при каждом чтении и передаче модели
Tool / MCPConnection / PluginVersion	Schema, endpoint, permissions, версия, риск действия	Пользователь выбирает только опубликованные возможности
Approval / DeliveryAttempt	Решение, согласующий, адресаты, результат отправки	Решение привязано к параметрам конкретного действия
AuditEvent / ConfigRevision	Автор, объект, версия политики, before/after	Секреты и лишнее содержимое не попадают в журнал

Ключевые API: каталог подключений; версии помощников; создание и отмена run; поток событий; расчёт estimate; просмотр budget account; регистрация инструментов; публикация config revision. Для изменяющих запросов поддерживается ключ идемпотентности. Пагинация использует стабильный cursor.

Начальный контракт runtime состоит из операций start, resume, cancel, get_state и stream_events. Он нормализует события и артефакты, но не пытается сделать checkpoints разных библиотек

взаимозаменяемыми. Перенос работающего задания на другой harness в первой версии не поддерживается.

8. Подключение внутренних и внешних моделей

Модели подключаются как готовые API. Поддерживаются OpenAI-compatible Chat Completions, Responses при подтверждённой совместимости, Anthropic Messages через соответствующий адаптер и провайдерские API через проверенные интеграции. Название «OpenAI-compatible» не является сертификатом поддержки всех функций.

Жизненный цикл подключения

1. Администратор задаёт endpoint, API-тип, доверенный CA, проху-маршрут и ссылку на секрет.
2. Выполняются безопасные проверки связи, аутентификации и ограничений сетевого назначения.
3. Контрактные тесты проверяют streaming, usage, отмену, tools и structured output, если они нужны.
4. Администратор утверждает доступные профили, группы, классы данных и прайс.
5. Подключение публикуется; состояние и ошибки видны оператору. При отключении новые вызовы прекращаются.

Настройка	Требуемое поведение
Размещение	Внутренняя сеть, корпоративный проху-провайдер либо прямой внешний API
Секрет	Только secret_ref; UI отображает маску и дату изменения
Возможности	Проверенный набор tools, streaming, JSON, multimodal и usage
Параметры	Ограниченный whitelist настроек, совместимых с API и политикой
Контекст	Предел входа и выхода; резерв перед отправкой; понятная ошибка переполнения
Fallback	Только явно разрешённые профили с теми же ограничениями данных
Стоимость	Версионизируемый тариф или внутренние единицы; отдельно cache и иные категории
Наблюдаемость	Латентность, ошибки, очередь, usage completeness и состояние соединения

Пользователь выбирает опубликованный профиль, а не произвольный URL. Пользовательские подключения допустимы только как отдельная управляемая функция с проверкой endpoint и полномочий; в R1 они отключены.

Fallback не пересылает закрытые данные внешнему провайдеру при отказе внутреннего API без соответствующей политики. Ротация ключа не раскрывает его worker или песочнице. Набор поддерживаемых провайдеров версионизируется вместе с приложением.

Вызов embeddings для RAG также является зарегистрированным подключением. Выбор и эксплуатация серверов embeddings остаются внешней зависимостью проекта.

9. Идентификация, LDAP и локальные группы

Приложение аутентифицирует пользователя через OIDC. Корпоративный IdP предпочтителен, если он уже существует и передаёт необходимые группы. Иначе поставляется Keycloak с read-only LDAP-федерацией, локальными пользователями и отдельными аварийными административными учётными записями [S05].

Требуемое поведение

- Настройка нескольких разрешённых LDAP endpoints и доверенного CA; секрет bind-account хранится отдельно.
- Сопоставление групп по стабильным идентификаторам; отображаемые названия можно менять без потери политик.
- Локальные группы создаются в портале и могут включать LDAP-пользователей. Они не записываются обратно в LDAP.
- Явные роли администратора, оператора, автора, пользователя и аудитора; MFA через возможности IdP для привилегированных операций.
- Контроль отключения пользователя и отзыва группы; повторная проверка доступа перед новым инструментом и продолжением задания.
- Раздельное управление сервисными идентичностями и обычными пользователями.

Членство из JWT может устареть. Для критичных действий используется актуальная версия политики и проверка блокировки. Предлагаемый предел распространения обычного изменения группы — не более пяти минут после синхронизации; немедленная административная блокировка хранится в портале и действует на границах новых операций. LDAP-синхронизация и время жизни токенов должны подтверждать этот предел на испытании.

Если LDAP недоступен, нельзя создавать новые привилегированные разрешения на основании отсутствующих данных. Поведение существующих сессий задаётся политикой: ограниченное чтение по непросроченному кэшу либо полный запрет. Аварийный локальный доступ документируется и журналируется.

Регистрация пользователей по внешнему email отключена по умолчанию. Совпадение email не является достаточным основанием для слияния LDAP и локальной учётной записи.

10. Пространства, роли и политика доступа

Права задаются сочетанием роли, области действия и ограничений объекта. Проверка выполняется сервером, включая запросы к файлам, SSE, истории, экспортам и API инструментов. Скрытие кнопки в UI не заменяет авторизацию.

Действие	Пользователь	Автор / владелец пространства	Администратор
Использовать помощника	По ACL	По ACL	По ACL; админство не даёт автоматического чтения данных
Создать черновик	По разрешению	Да, в своей области	Да
Опубликовать командную версию	Нет либо по делегированию	В своей области	Глобальные шаблоны
Добавить endpoint модели/МСР	Нет	Запрос администратору	После проверки назначения
Изменить общий бюджет	Нет	В пределах делегированного счёта	В пределах административных полномочий
Прочитать чужой документ	Нет	Только по ACL документа	Только по отдельному разрешению
Смотреть аудит	Свои события	События пространства	По роли аудитора/администратора

Эффективные права задания — пересечение прав инициатора, опубликованной версии помощника, пространства и глобальной политики. Для расписания инициатором является явно зарегистрированная сервисная идентичность. Её права не наследуются молча от автора сценария.

Каждый документ имеет класс данных: например, публичный, внутренний или ограниченный. Конкретная классификация согласуется с организацией. Связь «класс данных → разрешённые подключения и инструменты» хранится в политике. Дочерние задания и сводки сохраняют ограничения исходных данных.

Явный запрет имеет приоритет над разрешением. Исключения требуют области, срока действия и ответственного владельца. Изменение групп не должно открывать доступ к ранее закешированным чужим результатам. Cache key включает пространство и область доступа; данные разных пользователей не объединяются в общий кэш без проверки эквивалентности ACL.

11. Лимиты, бюджетные счета и учёт

Группа доступа и бюджетный счёт — разные сущности. Несколько LDAP-групп могут использовать один проектный счёт. Один пользователь может работать с несколькими счетами, но при запуске выбирается ровно один разрешённый источник оплаты; индивидуальные ограничения при этом сохраняются.

Измерение	Пример ограничения	Место контроля
Денежный / условный бюджет	Месяц, день, проект, одно задание	Транзакционный ledger и модельный шлюз
Токены	Input/output, общий предел задания	Каждый модельный вызов и все дочерние задания
Параллельность	Генерации, агентные runs, песочницы	Admission и пул workers
Длительность / шаги	Таймаут, число итераций, глубина подагентов	Оркестратор и runtime-профиль
Данные	Размер файла, число документов, объём индекса	Загрузка и pipeline
Инструменты	Число web-запросов, писем, дорогих операций	Шлюз инструментов
CPU / RAM / диск	Профиль песочницы и срок жизни	Исполнитель песочницы и host limits

Порядок списания

- Проверить права, период бюджета и источник оплаты. Отдельно проверить личный предел и общий счёт.
- До модельного вызова атомарно зарезервировать верхнюю оценку расхода по входу и разрешённому максимуму выхода.
- Записать reservation ID, версию прайса, run ID и уникальный ключ вызова. Параллельные запросы видят уже занятый резерв.
- После ответа получить нормализованный usage, сверить фактический расход, освободить остаток и записать проводку.
- Если ответ потерян или usage отсутствует, пометить расход как неподтверждённый. До сверки удерживать консервативный резерв по заданному регламенту.

Повторные попытки, суммаризация, валидационные исправления и подагенты учитываются в том же бюджетном дереве. Удаление истории или создание нового ключа не обнуляет расход. Отмена задания не означает бесплатный уже выполненный модельный запрос.

Ноль означает запрет расхода; отсутствие ограничения обозначается явно, а не числом 0. Период сброса, часовой пояс, перенос остатков и правила увеличения бюджета версионизируются. Рекомендуемые уведомления — при 70%, 90% и исчерпании бюджета; пороги настраиваются.

При недоступности ledger новые платные операции отклоняются. Для точного hard limit обязательно проверяется способность подключённого API ограничивать выход и предоставлять

usage. Без этой способности показывается ограничение точности учёта; платформа не обещает недоказуемую точность до токена.

12. Аналитика расхода, использования и совместного доступа

Под «калькулятором» в требованиях понимается прежде всего аналитическая панель: фактические расходы, использование моделей, tools и документов, публикация помощников и совместный доступ. Прогноз бюджета — дополнительный режим, а не основное назначение. Аналитика работает по событиям платформы и ledger; не требует модельных вызовов для расчётов.

Разделы панели

Раздел	Показатели	Разрезы
Расход	Подтверждённый расход, резерв, неизвестный usage, бюджет и остаток	Период, пользователь, группа, пространство, счёт, подключение
Использование	Активные пользователи, runs, calls, токены, ошибки и время	Помощник, сценарий, runtime-профиль, модельное подключение
Инструменты	Вызовы, уникальные пользователи, ошибки, длительность, повторное использование	Tool, MCP/server, команда и тип операции
Документы	Загрузка, индексация, retrieval, чтение, экспорт и downloads	База знаний, источник, владелец, пространство и класс данных
Совместный доступ	Публикации, share grants/revokes, доступные аудитории и фактическое использование	Помощник, tool, документ, группа, роль и срок разрешения
Результативность	Принятые задачи, стоимость принятого результата, оценки и незавершённые задания	Класс задачи и версия помощника

Совместный доступ и использование не смешиваются: объект может быть доступен 100 сотрудникам, но использован только пятью. «Популярный инструмент» означает фактические вызовы/пользователей за выбранный период, а не число выданных разрешений.

Расчёты

Стоимость вызова = сумма непересекающихся категорий billable tokens × тариф категории / 1 000 000. Адаптер различает uncached input, cache-read, cache-write и иные категории конкретного API. Если provider usage содержит общий input и вложенный cached count, их нельзя посчитать дважды. Прайс имеет валюту, источник, дату и версию.

Стоимость задания включает все calls, retries, подзадачи и платные инструменты. Стоимость принятой задачи = расход всех испытаний / количество принятых результатов; при нуле принятых результатов показатель не вычисляется. Число токенов само по себе не считается качеством работы.

Прогноз месяца строится по наблюдаемому расходу и выбранным допущениям: активные пользователи × задания в день × рабочие дни × средняя стоимость задания. Показывается диапазон, а не обещание точной будущей суммы. Инфраструктурный TCO портала учитывается отдельно. Разные валюты объединяются только при явно заданном курсе и дате.

События и доступ

События: run.started/completed, model.call/usage, tool.called, document.uploaded/read/retrieved/exported, share.granted/revoked, assistant.published и approval.resolved. Они имеют уникальный ID, timestamp, workspace, principal, объект и correlation ID. Повтор доставки не увеличивает агрегаты. Последнее успешное обновление и полнота данных видны пользователю.

Аналитика содержит фильтры времени, группы и пространства, сравнение периодов, drill-down до разрешённого run, CSV/XLSX export и предупреждения о бюджете. Пользователь видит свои данные; руководитель — разрешённую командную область; аудитор — события по роли. Названия недоступных документов и содержание запросов не раскрываются через агрегаты или экспорт. Персональный мониторинг и сроки хранения согласуются с организацией.

Источником финансового факта является ledger. Событийная аналитика использует инкрементальные агрегаты и проверяемый backfill; произвольные запросы dashboard не блокируют транзакции учёта. Предлагаемый лаг обновления — до пяти минут; остаток бюджета для нового вызова проверяется синхронно.

13. Выбор агентного runtime и опыт Anthropic

Ни один рассмотренный harness не объявляется самым экономичным до испытаний на задачах организации. Рассмотрение Deep Agents отражает функциональное соответствие, а не доказанный выигрыш по токенам. Предлагается разделение: прямые вызовы для чата; управляемые workflows для предсказуемых операций; готовый harness для задач с открытым планом.

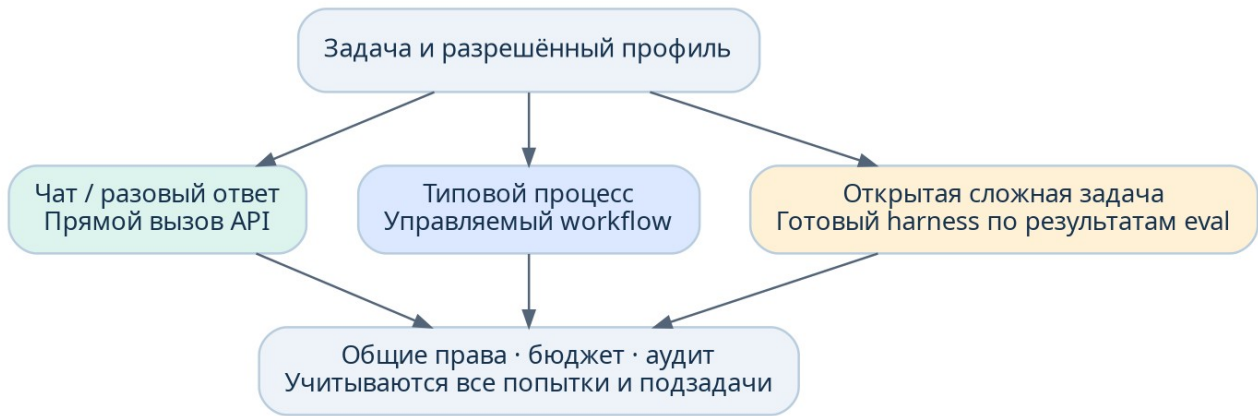


Рисунок 3. Режим выполнения выбирается по задаче; бюджеты и права общие для всех режимов.

Кандидат	Что проверяем	Предлагаемая роль
Прямой модельный API	Качество ответа при одном/нескольких заранее определённых вызовах	Базовая линия для чата и простых преобразований
LangGraph / готовый LangChain agent	Контроль графа, инструментов, событий и checkpoints	Управляемые сценарии; без необязательных сложных возможностей [S01–S02]
Pydantic AI	Типизированные tools/output, интеграции провайдеров, восстановление	Альтернативный кандидат для тех же коротких сценариев [S16]
Deep Agents	Контекст, файловые tools, подагенты и открытые задачи	Сложные задания, если результат оправдывает расход [S01]
OpenHands SDK	Изменения кода, тесты и состояние удалённой среды	Специализированный coding-кандидат [S10]
Claude Agent SDK	Готовый harness Claude Code, tools и permissions	Кандидат отдельного Claude-профиля; не предполагать поддержку произвольных моделей [S18]

На чём остановилась Anthropic

Публично Anthropic описывает reuse собственного harness Claude Code в Claude Agent SDK: работа с файловой системой и инструментами, сбор контекста, действие и проверка результата.

Публикация не содержит протокола, который доказывал бы выбор этого harness по лучшей стоимости относительно LangGraph или других движков [S18].

В Managed Agents Anthropic отдельно описывает сохраняемый журнал сессии, harness и sandbox, а также проху к инструментам с секретами вне harness. Для нашего проекта это архитектурный ориентир: контекст, выполнение и секреты разделяются. Облачный продукт не считается автоматически подходящим для независимого размещения в закрытом контуре [S19].

В своём исследовательском продукте компания описала orchestrator-worker подход. По их наблюдениям агенты потребляли около 4× токенов чата, несколько агентов — около 15×. Это показатели конкретной системы, а не универсальные множители для финансовой аналитики [S20].

Предлагаемое решение

На этапе 0 сравнить прямой API, один управляемый agent/workflow и один сложный harness. Pydantic AI — допустимый альтернативный baseline; если он выигрывает по качеству, стоимости и сопровождению, выбрать его, не поддерживая два взаимозаменяемых runtime без необходимости. В R1 не устанавливать весь список кандидатов. Отдельный coding-runtime добавлять только при подтверждённой потребности.

14. Экономика токенов и сравнительные испытания

Токены — измеряемая часть расхода, но их количество не отражает пользу без проверки результата. Оптимизация должна сохранять качество, соблюдать права и учитывать все попытки, включая неудачные. API-цены и модельные SLA не фиксируются в этом проекте; они поступают из каталога подключений.

Методика выбора

Сформировать 40–60 репрезентативных задач: чат и преобразования; поиск по документам; исследование; действия через инструменты; изменения кода. При ограниченном пилоте допустим набор 20–30 задач, но выводы будут предварительными. Для каждого класса определить эталон или рубрику приёмки. Сравнить runtime на одном и том же модельном подключении, инструментах, данных и верхнем бюджете.

Выполнить не менее трёх повторов для недетерминированных задач в сбалансированном порядке. Зафиксировать версии prompt, инструмента, модели и runtime. Отдельно провести испытание на втором типе API для переносимости; результаты разных моделей не смешивать с эффектом harness.

Метрика	Способ измерения	Для чего используется
Task success rate	Результат по заранее утверждённым критериям	Основной показатель качества
Стоимость принятой задачи	Все затраты / число принятых результатов	Экономика с учётом неудач
Input/output/ cache и вызовы	Usage каждого call; отсутствие usage отмечается	Анализ причин расхода
Время выполнения	p50/p95 полного задания и время ожидания	Пользовательская полезность
Tool correctness	Валидные аргументы, допустимые операции и выполненные проверки	Надёжность интеграций
Повторные попытки	Причина и дополнительная стоимость	Поиск неэффективных циклов
Безопасность и восстановление	Отрицательные ACL-тесты и сбой worker	Обязательный критерий допуска
Стоимость сопровождения	Зависимости, offline-сборка и трудоёмкость интеграции	Выбор за пределами токенов

Для кода качество подтверждают тесты и review diff; для документов — верность источников и полнота ответа; для бизнес-действий — проверяемое состояние системы. LLM-as-judge применяется только как вспомогательная проверка с контролем человеком и программными тестами.

Правила экономичного выполнения

- Минимально достаточные tools и schemas для конкретного помощника; discovery вместо включения всего реестра в каждый prompt.
- Ограниченный объём результатов инструмента и RAG; полный материал остаётся доступным по контролируемой ссылке.
- Подагенты и суммаризация только в профилях, где они нужны; их вызовы тоже оплачиваются.
- Проверки формата и правил выполняются кодом, когда модель не требуется.
- Один владелец retry policy: не умножать повторы на уровнях gateway, SDK и Temporal.
- Маршрутизация и cache учитывают права, классификацию данных и реальную совместимость API.

Выбор фиксируется в ADR с результатами испытаний. При смене версии runtime выполняется тот же regression-набор. Заявление о проценте экономии допускается только после измерения; в проекте такие проценты не обещаются.

15. Помощники и конструктор сценариев

Помощник — версия инструкции и набора разрешённых возможностей. Автор задаёт назначение, модельный профиль, режим выполнения, knowledge sources, инструменты, ограничения и формат результата. Создание помощника не требует писать код.

Сценарий описывает управляемую последовательность шагов. В R1 поставляются шаблоны с формами настройки. В R2 появляется визуальный редактор с проверкой схемы до публикации. Обязательные шаги: входные данные, получение документа, разрешённый поиск, вызов модели, инструмент, условие, ограниченная параллельная ветка, согласование и формирование артефакта.

Правило	Реализация
Проверка схемы	Несуществующий tool/model, неподдерживаемое поле или цикл без предела блокирует публикацию
Вход и выход	Типизированные поля; JSON Schema для программных шагов
Циклы	Верхний предел итераций, времени, вызовов и расхода
Параллельность	Ограничение веток и общий родительский бюджет
Согласование	Решение привязано к версии действия и конкретным параметрам
Изменение версии	Новые runs используют новую версию; старые продолжают закреплённую
Экспорт	Декларативная схема без секретов; импорт проверяет зависимости
Пользовательский код	Только отдельный sandbox-шаг; не выполнение в API-процессе

Начальные шаблоны: ответ по документам; сравнение двух документов; исследование с источниками; структурирование входных данных; подготовка отчёта; создание черновика письма; отправка после согласования; анализ репозитория; исправление с запуском тестов.

Расписание имеет собственную сервисную идентичность, бюджет, часовой пояс и лимит параллельных запусков. При снятии разрешения плановый запуск прекращается. Управление расписанием использует возможности Temporal и серверные политики, а не браузерный таймер.

16. Жизненный цикл задания и восстановление



Рисунок 4. Задание сохраняет состояние независимо от браузера; повтор внешнего действия требует отдельной проверки.

Состояния: создано, ожидает допуска, в очереди, выполняется, ожидает пользователя, восстанавливается, завершено, отменяется, отменено и завершено с ошибкой. Переходы журналируются. У задания есть общий deadline; отдельные шаги имеют собственные timeout и retry policy.

Temporal отвечает за бизнес-жизненный цикл задания, таймеры и доставку работы. Выбранный агентный runtime отвечает за внутреннее состояние и контекст. Если используется LangGraph, checkpoint хранится отдельно и связан с run ID. Недетерминированные модельные вызовы выполняются в Activities либо другом явно определённом исполнительном слое, не в детерминированном коде Workflow [S02–S03].

Состояние интерфейса восстанавливается по сохранённому event cursor. События доставляются как минимум один раз; UI удаляет дубли по sequence ID. Поток токенов группируется: не требуется транзакция БД на каждый токен. Подтверждённые результаты и финальный usage сохраняются отдельно.

Сбой и отмена

Потерянный worker выявляется по heartbeat/timeout. Новый worker получает право исполнения через действующую lease и проверку состояния; поздний результат старого исполнителя не должен запускать второй внешний эффект. Проверить это необходимо в fault-тестах.

Retry разрешается только для операций с понятной семантикой. Чтение можно повторить по политике; создание записи, Git push или отправка письма требуют idempotency/outbox и обработки неоднозначного результата [S25]. Для SMTP отдельно учитывается неоднозначное подтверждение: если неизвестно, принял ли relay письмо, нельзя бесконтрольно повторять отправку.

Отмена блокирует новые шаги и передаётся исполнителю. Уже отправленный внешний модельный запрос может завершиться и иметь стоимость; интерфейс показывает фактический результат отмены и сверку расхода. Остановленная песочница сохраняет разрешённые артефакты и освобождает ресурсы по политике.

17. Веб-поиск и контролируемый fetch

Веб-поиск выполняется отдельным сервисом с собственной исходящей политикой. Workers вызывают search/fetch как внутренние инструменты. Для первоначальной реализации рассматривается SearXNG либо согласованный search API; провайдер меняется через адаптер. Поиск и чтение найденной страницы разделены: первый возвращает найденные источники, второй загружает разрешённое содержимое [S08].

Операция	Разрешённый результат	Контроль
search(query)	Заголовок, URL, фрагмент и время поиска	Лимит запроса, классы данных, разрешённые engines
fetch(url)	Нормализованный текст и ссылка на snapshot	URL/IP-проверка, redirect-проверка, timeout и размер
fetch_document(url)	Проверенный файл, если функция разрешена	MIME, антивирус, архивные ограничения и ACL
browser_fetch	Извлечение страницы с JavaScript, отдельный профиль	Изолированный браузер, без корпоративных cookies и иных секретов

Обычный search/fetch не исполняет произвольный код страницы. Запросы к loopback, private/link-local адресам, metadata endpoints и внутренним административным службам блокируются; DNS и каждый redirect проверяются на этапе фактического соединения. Внутренний поиск реализуется отдельным connector с собственной allowlist.

Если сеть разрешает только ограниченный список сайтов, портал сообщает, что исследование ограничено этой областью. Для чтения произвольного найденного сайта требуется соответствующая политика именно fetch-сервиса; открывать доступ workers не нужно.

Поисковый запрос может содержать корпоративные данные. Для ограниченных классов данных внешний search запрещается, либо используется утверждённый запрос/процесс разрешения передачи. Простая фильтрация ключевых слов не считается доказательством отсутствия утечки. Источники веба считаются недоверенными данными, не инструкциями для изменения прав или запуска инструментов.

При недоступности поиска можно продолжить разрешённую работу по внутренним материалам, явно указав отсутствие внешней проверки. Нельзя выдавать старый кэш за свежий поиск; время snapshot показывается.

18. Работа с документами и файлами

Поддерживаемые форматы первой поставки: PDF, DOCX, TXT, Markdown, CSV и XLSX. OCR добавляется для согласованных языков и качества сканов. Разбор таблиц и изображений является отдельной проверяемой возможностью. Для локального parser рассматривается Docling и специализированные библиотеки по формату [S09].

Конвейер: загрузка → проверка типа/размера → карантин/антивирус → извлечение → представление текста и структуры → артефакты → при необходимости индекс. Пользователь видит статус и причину отказа. Большой файл обрабатывается фоновым заданием.

Настройка	Начальный проектный предел	Изменение
Файл интерактивной загрузки	25 MB	Администратор по ресурсу и формату
Файл фоновой обработки	100 MB	После проверки parser и timeouts
Число файлов в одном запросе	20	По политике пространства
Архивы	Отключены в R1	Позже с пределом распаковки, вложенности и MIME
OCR	Выбранные языки	Расширяется после тестового корпуса
Office-макросы	Не выполняются	Не включать как побочный эффект parsing

Пределы служат отправной точкой, а не окончательным SLA. Парсеры запускаются в ограниченной среде с deadline и memory limits. Повреждённый файл не должен блокировать worker pool. Формат определяется по содержимому и библиотеке, а не только расширению.

Результаты: краткое содержание, сравнение версий, извлечённые таблицы, структурированный JSON/CSV, отчёт DOCX/PDF и ссылки на источники. Для цитирования сохраняются page/section/sheet identifiers и версия документа.

Редактирование исходного DOCX с полным сохранением сложной вёрстки не обещается в R1. Первоначально формируются новые артефакты по шаблонам. Интерактивный офисный редактор — возможное отдельное расширение, не обязательная часть портала.

Удаление исходника инициирует удаление извлечённого текста, chunks, preview и неразрешённых кэшей. Порядок удаления backup определяется политикой хранения и не выдаётся за мгновенное физическое стирание всех копий.

19. RAG и базы знаний

База знаний принадлежит пространству и содержит версии источников. Векторизация вызывается через зарегистрированный embedding API; инфраструктура самого API находится вне проекта. Для начального объёма используется pgvector и полнотекстовый поиск. Переход на отдельный search/vector сервис подтверждается измерениями, а не числом зарегистрированных пользователей [S11].

Индексация

Документ получает версии, ACL, класс данных и статус обработки. Chunks сохраняют связь с исходником, его версией и координатами. Модель embeddings и размер вектора фиксируются в версии индекса. Смена embedding-профиля создаёт новый индекс либо управляемую переиндексацию; смешивать несовместимые векторы нельзя.

Получение контекста

Сначала определяется допустимая область данных, затем выполняется retrieval и повторная проверка актуального ACL перед передачей результата модели. Для approximate search фильтрация должна сохранять качество выборки; нужно проверять iterative scans, partitioning или отдельные индексы на реальных правах. Секретный chunk не передаётся модели с надеждой, что она его не процитирует.

Возможность	Критерий реализации
Semantic + lexical search	На эталонном корпусе измеряется релевантность обоих каналов
Цитирование	Ответ содержит существующие ссылки на доступные источники
Обновление	Изменённая версия не смешивается с актуальной без явной политики
Удаление	Удалённый источник перестаёт возвращаться в retrieval после завершения pipeline
ACL	Чужие документы отсутствуют в результатах, prompt, кэше и артефактах
Диагностика	Пользователь видит область поиска, статус индекса и отсутствие нужных данных

В R1 knowledge-функция ограничивается выбранными файлами; общий долговременный RAG поставляется в R2. Reranker подключается только при подтверждённом улучшении качества и с учётом дополнительной стоимости.

Доступность embedding API и его пропускная способность задают ограничения индексации. Сайзинг портала учитывает очередь и локальный parsing, но не включает серверы модели embeddings.

20. Код и изолированные рабочие среды

Coding-задание получает отдельную рабочую среду, ограниченный набор файлов и профиль ресурсов. Пользователь выбирает зарегистрированный репозиторий либо загружает архив по политике. Результат — diff/patch, журнал проверок и разрешённые артефакты. Автоматическое изменение production-среды не входит в базовую функцию.

Начальный профиль песочницы: 2 vCPU, 4–8 GB RAM, 20 GB рабочего диска и deadline до 60 минут; это настраиваемые пределы, проверяемые на пилоте. Тяжёлый профиль 4–8 vCPU/16 GB требует отдельного права и ресурсной квоты.

Область	Правило
Изоляция	Отдельный контейнер/VM, непривилегированный код, ограничение PID/CPU/RAM/диска
Сеть	Прямой интернет запрещён; доступ только к назначенным внутренним mirrors и endpoints
Секреты	Ключи моделей, LDAP, SMTP и управления платформой в среду не передаются
Docker	Общий host socket недоступен; создание сред через изолированный executor/control service
Git	Разрешённые репозитории, минимальные права, отдельное подтверждение push/PR
Инструменты сборки	Заранее подготовленные и версионизируемые образы
Завершение	Сохранить доступные артефакты; закрыть процессы; очистить среду по retention
Совместимость	Проверить shell, компиляторы, браузерные тесты и ограничения runtime

Для усиления контейнерной изоляции рассматривается gVisor или отдельные VM. Выбор зависит от совместимости тестов, доступного runtime и требований организации. Готовый harness не освобождает от ограничения среды [S12].

Без внутренних mirrors агент не сможет загрузить зависимости при сборке. Для проектов должны быть доступны согласованные npm/Python/Go/OS/container зависимости. Пользователь не получает право подключить произвольный интернет-проху ради одного теста.

Если нужен специализированный coding-runtime, OpenHands SDK проходит тот же eval-набор. Самодельные файловые адаптеры и интеграция executor допустимы; собственный универсальный агентный цикл не разрабатывается.

21. MCP, инструменты и плагины

Шлюз инструментов публикует реестр проверенных возможностей. MCP используется как стандартный протокол интеграции там, где он соответствует задаче. Регистрация сервера не означает разрешение всех его tools всем пользователям [S13].

Контракт расширения

Manifest содержит ID, версию, описание, входную/выходную schema, endpoint, required permissions, классы данных, сетевые назначения, секреты, ресурсный профиль и семантику внешних эффектов. Секреты задаются ссылками. Автор предоставляет health-check, контрактные тесты и инструкцию offline-поставки.

Этап	Обязательная проверка
Регистрация	Владелец, лицензия/источник, endpoint, зависимости и версия
Проверка	Schema, auth, SSRF, ресурсные пределы, negative ACL tests
Публикация	Разрешённые пространства и инструменты, режим согласования
Обновление	Совместимость контрактов и возможность rollback
Отключение	Запрет новых вызовов; понятное поведение активных заданий

Вызов выполняется с идентичностью `run`, установленной сервером. Поля `user_id/group_id` из сгенерированных моделью аргументов не считаются доказательством полномочий. Downstream-система должна проверять права; обычный сервисный ключ с неограниченным доступом нельзя выдавать за делегированную пользовательскую авторизацию.

Раздельно контролируются чтение, изменение, удаление и внешний обмен данными. Согласование применяется к конкретной операции, а не ко всему tool на неопределённое время. Изменение параметров после согласования требует нового решения.

Пользовательские плагины исполняются в отдельных сервисах/средах. Произвольная загрузка кода в процесс Go API запрещена. На старте не нужен открытый marketplace: достаточно корпоративного реестра и workflow публикации.

22. SMTP и уведомления

Системные уведомления и пользовательская отправка письма — две независимые функции. Уведомления о завершении задания не должны давать пользователю право отправить произвольное письмо от имени платформы.

Функция	Политика
Системное уведомление	Утверждённый sender, шаблон, адрес пользователя и минимальный контент
Черновик письма	Создание артефакта без сетевой отправки
Отправка пользователем	Tool permission, разрешённый sender и адресаты, проверка вложений
Внешние адресаты	Отдельное разрешение по классу данных и доменам
Автоматическая рассылка	Сервисная идентичность, лимит количества, owner и аудит

SMTP relay и TLS настраиваются администратором. Почтовые credentials доступны только почтовому сервису. Письмо проходит проверку получателей, размера, MIME и классификации вложений. При необходимости UI показывает точное содержимое перед согласованием.

Для отправки используется outbox и уникальный ID бизнес-операции. Журнал различает запланировано, передано relay, отклонено и неопределённый результат. Успешная передача relay не равна подтверждённому прочтению письма. При неоднозначном результате автоматические повторные отправки ограничены регламентом.

Предлагаемый стартовый лимит — до 20 пользовательских писем в день на человека; фактический предел устанавливает организация. Массовая внешняя рассылка не включается по умолчанию.

23. Единая администрация и оркестраторы

Пользователь получает одну административную панель, но за ней разделены управление приложением, исполнение заданий и размещение сервисов. Один оркестратор не заменяет правила доступа или каталог модельных API.

Уровень	Инструмент	Что администрируется
Продукт / control plane	Go API и панель портала	Подключения, помощники, права, бюджеты, инструменты, прайс и политики
Агентные задания	Temporal и интеграция runtime	Очереди, timeout, retries, расписания, отмена и восстановление
Базовая инфраструктура	Ansible + Compose	Inventory, конфигурация VM, контейнеры, версии и обновление
Динамическое размещение, опция	Nomad или Kubernetes	Placement, capacity, автоматическое перемещение контейнеров

Глобальная конфигурация

Настройки имеют области: инсталляция → пространство → помощник → задание. Нижний уровень может ужесточать ограничения; расширять права без делегирования нельзя. Изменение проходит валидацию, preview влияния, запись новой revision, применение и проверку. Секреты хранятся отдельно и не возвращаются в export.

В панели доступны LDAP mappings, зарегистрированные endpoints, разрешённые сетевые маршруты, бюджеты, цены, профили runtime, resource pools, SMTP, retention, feature flags и правила согласования. Для изменения критических политик предусмотрен дополнительный review, если это требует организация.

Показываются активная и желаемая конфигурации, drift и результат последнего применения. Глобальная настройка не редактирует скрыто произвольные файлы сервера. Инфраструктурное действие передаётся ограниченному deployment runner с утверждённым playbook и параметрами; web API не получает root SSH ко всем VM.

Выбор инфраструктурного инструмента

Базовый вариант — Compose + Ansible на нескольких VM. Он понятнее для старта и не требует Kubernetes. Его предел — статическое размещение сервисов: отказоустойчивость обеспечивают готовые реплики и запас мощности; Ansible добавляет или восстанавливает узлы [S06, S21].

Nomad рассматривается, если потребуется автоматическое placement без Kubernetes. Он поставляется как единый бинарник, но HA, ACL/TLS, сеть, storage и секреты всё равно требуют настройки. Условия лицензии выбранной версии проверяются отдельно; нельзя называть их автоматически Apache/MIT [S22–S23].

Kubernetes предлагается только при наличии команды/готовой корпоративной платформы или подтверждённой необходимости. Его возможности не являются обязательным условием заявленных функций портала.

24. Сетевая архитектура и матрица доступов

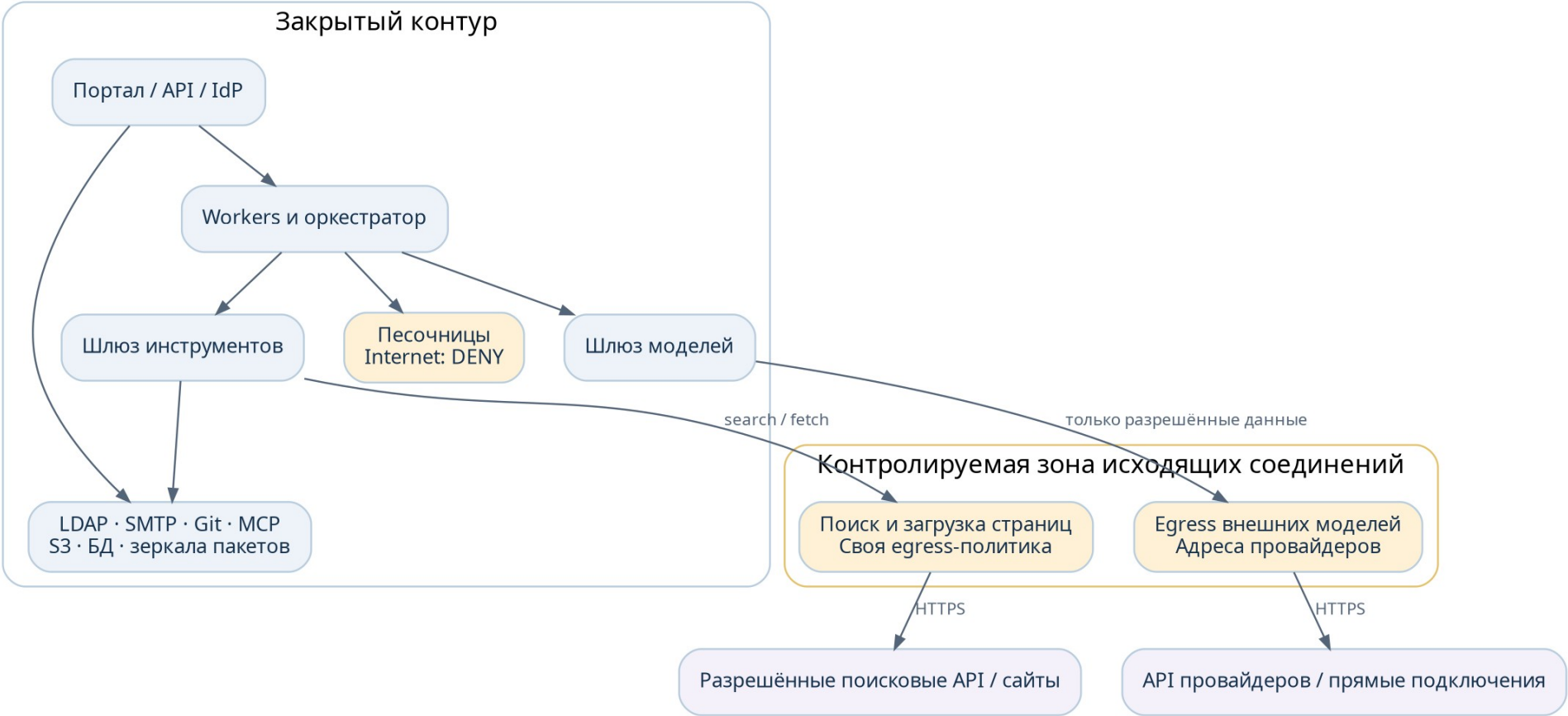


Рисунок 5. Раздельные исходящие направления: внешние модели и веб-поиск. Workers и песочницы не получают общего выхода.

Все назначения регистрируются. По умолчанию соединения между зонами запрещены; открываются только строки утверждённой матрицы. В таблице приведены типовые порты приложения; конкретный проху, S3 и модельный endpoint могут использовать иной согласованный порт.

Источник	Назначение	Протокол / порт	Назначение связи
Браузер	LB портала и IdP	HTTPS 443	UI, API, OIDC, SSE
LB	API / IdP	HTTPS, зарегистрированный порт	Проверенные внутренние backends
IdP	LDAP/AD	LDAPS 636; либо утверждённый StartTLS	Read-only пользователи и группы
API / runtime / сервисы	PostgreSQL	TCP 5432 + TLS	Только своя БД/роль
API / workers	Temporal frontend	TCP 7233 + TLS	Задания, события и polling
Workers	Модельный шлюз	HTTPS 443	Разрешённые модельные вызовы
Модельный шлюз	Внутренний API модели	Утверждённый endpoint	Конкретное подключение
Модельный шлюз	Egress проху моделей	Корпоративный проху-порт	Только модельный маршрут
Egress проху	API провайдеров	HTTPS 443	Allowlist провайдеров
Workers	Шлюз инструментов	HTTPS 443	Привязанные к run вызовы
Шлюз инструментов	Search/fetch	HTTPS 443	Отдельный сервис
Search/fetch	Интернет через свой проху	HTTPS 443; HTTP только по исключению	Поисковые engines и разрешённые страницы
Шлюз инструментов	MCP / Git / корпоративные API	Только зарегистрированные порты	Разрешённые операции
Почтовый сервис	SMTP relay	Обычно TCP 587 + TLS	Уведомления и разрешённые письма
API / workers	S3	Обычно HTTPS 443	Свои buckets и object scopes
Worker executor	Песочница	Закрытый управляющий endpoint	Запуск, события, файлы и остановка
Песочницы	Назначенные внутренние mirrors	HTTPS, согласованные endpoints	Сборка без интернета

Источник	Назначение	Протокол / порт	Назначение связи
Узлы	Корпоративные DNS / NTP	TCP/UDP 53; UDP 123	Разрешённые серверы
Узлы	Registry / пакетные зеркала	HTTPS 443	Offline-поставка и зависимости
Deployment runner	Управляемые VM	SSH 22 из административной зоны	Утверждённые Ansible-операции
Мониторинг	Метрики компонентов	Разрешённые metrics endpoints	Отдельная сервисная идентичность

Дополнительные служебные связи

Внутренние порты Temporal между его компонентами, PostgreSQL replication, выбранного HA-механизма, S3 и мониторинга фиксируются по конкретной версии в deploy inventory. Для self-hosted PostgreSQL HA может требоваться отдельный quorum/control store; его соединения добавляются явно.

В варианте Nomad: HTTP API обычно 4646, RPC 4647, gossip TCP/UDP 4648; окончательная конфигурация проверяется по документации версии. В варианте K3s: 6443 для управляющих узлов, 2379–2380 между etcd servers и выбранные CNI-порты. Эти опциональные строки не открываются при базовом Compose-развёртывании [S07, S24].

Compose networks и контейнерные настройки сами по себе не доказывают запрет egress. Применяются host firewall и ограничения на границе сети; учитываются Docker forwarding/NAT, IPv6, DNS и прокси. Запрет проверяется из самой песочницы и worker. Обход через общий HTTP-tool не допускается.

25. Хранилища, секреты и сроки хранения

PostgreSQL хранит каталоги, права, задания, проводки и аудит. Большие документы, snapshots, checkpoints с крупными вложениями и артефакты размещаются в S3 либо связываются с объектами S3. У каждого сервиса отдельные credentials и минимальная роль.

Ledger не редактируется вручную для исправления расхода: применяется корректирующая проводка с основанием. Конфигурация и опубликованные версии помощников сохраняются для воспроизводимости задания. История событий не равна бесконечному хранению всей модельной переписки.

Данные	Предлагаемый стартовый retention	Особенности
Диалоги и обычные результаты	90 дней	Пользовательская/пространственная настройка
Рабочие артефакты и web snapshots	30 дней	Закреплённые результаты могут храниться дольше
Временные среды	До 24 часов после завершения	Диск и процессы очищаются по регламенту
Checkpoints	30 дней после завершения	Активные задания не удаляются
Метрики высокой детализации	30 дней	Долговременные агрегаты отдельно
Аудит и бюджетные проводки	180 дней или корпоративный норматив	Значение согласуется до production
Базы знаний	До удаления/архивирования владельцем	Версии и backup по общей политике

Это проектные значения, не нормативные требования. Юридические и корпоративные сроки предоставляет заказчик. Отдельно определяется срок хранения резервных копий и процедура удаления из восстановленной системы.

Секреты доступны через внутренний secret service либо защищённую конфигурацию сервисов с ограниченными файловыми правами. В R1 не обязательно разворачивать новый Vault, если корпоративный secret store уже есть. Обязательны masking, ротация, исключение из логов/export и отсутствие ключей в пользовательской среде.

Backup охватывает PostgreSQL, S3/артефакты, необходимые конфигурации и ключевой материал по отдельному защищённому процессу. Наличие replication не заменяет backup от случайного удаления или повреждения данных.

26. Развёртывание и отказоустойчивость без обязательного Kubernetes

Базовый production-вариант — несколько VM с закреплёнными ролями и Compose-проектами. Ansible устанавливает одинаковые версии, конфигурацию и firewall. Load balancer направляет трафик к здоровым API/IdP. Workers разных VM получают задания из общих Temporal task queues; отдельные очереди обслуживают чат, документы и код.

Профиль	Что обеспечивает	Ограничение
Один сервер, разработка	Проверка UI, API и runtime на локальных контейнерах	Не используется как HA-система
Разделённый пилот	Приложение, данные, выполнение и поиск на отдельных VM	Часть сервисов остаётся без резервной реплики
Multi-VM production	API/IdP replicas, резерв workers, HA данных, backup	Нет автоматического placement контейнеров между VM
Nomad, опция	Динамическое размещение и восстановление workloads	Отдельный control cluster, политика сети, secrets/storage и проверка лицензии
Kubernetes, опция	Корпоративный стандарт и существующие platform services	Принимается только при готовой эксплуатации или подтверждённой потребности

Для PostgreSQL предпочтителен существующий корпоративный HA-сервис. Если его нет, в проект инфраструктуры включаются primary, replicas и готовый механизм failover с quorum, fencing и проверкой split-brain. Это отдельный обязательный результат этапа HA, а не скрытое свойство Compose. В качестве кандидата можно проверить Patroni либо стандарт организации; конкретная схема фиксируется ADR.

Temporal в production разворачивается по поддерживаемой схеме с несколькими экземплярами сервисов и отдельным persistence. Replica count не заменяет проверку отказа и доступности БД. У разных сервисов отдельные базы и connection pools; тяжёлый retrieval не должен блокировать ledger.

Запас N+1 означает, что после потери одного узла соответствующего пула остаётся ресурс для согласованного числа заданий. Очередь ограничивает новую нагрузку при нехватке capacity. Рост пула выполняется Ansible; динамический autoscaling не обещается в базовой поставке.

Предлагаемые цели: доступность портала 99,9% в месяц; RPO при восстановлении из backup до 15 минут; RTO до двух часов. Они подлежат согласованию и испытанию. Для acknowledged транзакций при одиночном отказе RPO=0 допускается только при проверенной synchronous replication/failover-конфигурации. Внешние модельные API и корпоративные системы имеют отдельные зависимости и показатели; end-to-end доступность измеряется независимо.

27. Сайзинг серверов приложения

Расчёт исключает inference, GPU, обучение и серверы embeddings. Подключаемые API моделей рассматриваются как внешняя зависимость. Сайзинг приложения определяется открытыми сессиями, числом задач, тяжестью parsing и количеством активных песочниц.

Допущения нагрузки

Профиль	Сессии на пике	Одновременные runs	Coding-среды
100 пользовател ей	10–20	5–10	3–5
1000 пользовател ей	100–200	20–50	10–20
3000 пользовател ей	300–600	60–120	30–60

Это проектные сценарии, не нормативные коэффициенты. При преимущественно чатовой работе сервисных серверов понадобится меньше, чем при большом объёме OCR, сборок и браузерных тестов. Расход CPU idle agent, ожидающего модельный API, нельзя приравнять к компиляции проекта.

Пилот: четыре VM

Роль	Узлы	vCPU	RAM	Диск	Размещаемые компоненты
Приложение	1	8	32 GB	200 GB SSD	API/UI, IdP, шлюзы и небольшой Temporal
Данные	1	8	32 GB	0,5–1 TB SSD	PostgreSQL и пилотное файловое хранилище
Выполнение	1	16	64 GB	500 GB SSD	Runtime, parsing и ограниченные песочницы
Search/fetch	1	4	8–16 GB	100 GB SSD	Отдельный поиск с разрешённым egress

Для пилота разрешено объединить роль данных с приложением при снижении изоляции и отсутствии HA. Это не должно стать незаметным production-решением. Если корпоративные IdP/БД/S3 доступны, они уменьшают число локальных компонентов.

Распределённая эксплуатация: ориентиры на один узел

Роль	Узлы	vCPU	RAM	Диск	Комментарий
API / IdP / gateway	2–3	8–16	32 GB	200 GB SSD	Повторяющиеся stateless/поддерживаемые сервисные роли
Temporal / platform	3	8	16–32 GB	200 GB SSD	Могут совмещаться с сервисными VM после измерения
Runtime / parsing	2–4	16	64 GB	500 GB SSD	Отдельные пулы очередей и ограничение OCR
PostgreSQL, если свой	3	8–16	32–64 GB	1–2 TB NVMe	Primary/replicas, готовый failover и отдельный quorum по схеме
Coding-среды	4–6	32	128 GB	1 TB NVMe	Для 30–60 базовых сред с проверенным запасом N+1
Search/fetch	2	4–8	16 GB	200 GB SSD	Разные отказы egress и инструмента
Наблюдаемость	1–2	8	32 GB	1–2 TB SSD	Retention и ingest определяют диск; важны внешние alerts
Registry / delivery	1–2	4–8	16 GB	1–2 TB SSD	Доступность обновлений; локальный кэш образов
LB, если свой	2	2–4	4–8 GB	50 GB SSD	Либо существующий корпоративный сервис

Таблица — роли для верхнего профиля; её строки нельзя механически считать обязательными отдельными VM. API и platform-роли можно объединять. Coding-pool не нужен при отсутствии coding-функции. Для 100–1000 пользователей production достаточно меньших runtime/coding-пулов при сохранении числа реплик, нужного для отказоустойчивости.

Как изменяются пулы при росте аудитории

Нагру зка	Service VM: API/IdP/gateways + Temporal	Runtime/parsing VM	Coding VM	Search VM
100 пользо вателей, produ ction	3 × 8–16 vCPU / 32 GB	2 × 16 vCPU / 64 GB	2 × 16 vCPU / 64 GB	2 × 4 vCPU / 16 GB
1000 пользо вателей	3 × 16 vCPU / 32 GB	2–3 × 16 vCPU / 64 GB	2–3 × 32 vCPU / 128 GB	2 × 4–8 vCPU / 16 GB
3000 пользо вателей	3 × 16 vCPU / 32–64 GB	3–4 × 16 vCPU / 64 GB	4–6 × 32 vCPU / 128 GB	2 × 8 vCPU / 16 GB

Эта таблица задаёт стартовые профили с совмещёнными service-ролями, а не добавляется к предыдущей. БД (обычно три узла), LB (два, если нет корпоративного), storage, delivery и наблюдаемость учитываются дополнительно по предыдущей таблице. Три service VM позволяют заранее разместить поддерживаемые реплики Temporal и IdP, однако их схемы распределения и зависимости проверяются отдельно. Размещение приложений не создаёт автоматически HA базы данных.

Базовая coding-среда для расчёта — 2 vCPU / 4 GB RAM; тяжёлые сборки получают иной профиль. При резерве 25% на узел и N+1 число coding VM рассчитывается как $\text{ceil}(\text{одновременные среды} \times \text{vCPU среды} / (\text{vCPU узла} \times 0,75)) + 1$, затем проверяется по RAM и диску. Для 60 сред и 32-vCPU узлов получается шесть VM; для 30 — четыре. Это расчёт допуска ресурсов, не обещание быстрогодействия конкретной сборки.

Корпоративное S3 и backup storage предполагаются внешними сервисами. При их отсутствии требуется отдельный расчёт storage: исходные файлы + артефакты + версии + retention + резервные копии + репликационный overhead. Пользователи сами по себе не определяют дисковый объём.

Сеть: начальный ориентир 1 Gbit/s для приложения; для интенсивной передачи файлов/backup между data и execution-пулами проверить 10 Gbit/s. Для БД важнее измеренная latency/fsync/IOPS, чем только размер диска.

28. Offline-установка, обновление и восстановление

Поставка должна полностью устанавливаться при запрете общего интернета. Внешний подготовительный build-процесс собирает разрешённые компоненты, проверяет происхождение и передаёт комплект в контур по регламенту. Production runtime не скачивает зависимости при старте.

Состав комплекта	Что фиксируется
Container images	Версия, digest, архитектура, источник и подпись/контрольная сумма
ОС и инструменты	Пакеты Docker/Compose, Python/Go tools, CA и системные зависимости
Automation	Ansible collections, playbooks, inventory templates и миграции
Offline-зависимости	Wheels, npm/Go dependencies, parser/OCR assets и образы песочниц
Конфигурация	Templates без секретов, схема версии и перечень обязательных параметров
Проверки	Preflight, smoke, network-deny probes, restore и upgrade tests
Эксплуатация	Инструкции, SBOM, known issues, rollback и changelog

Проектируемый интерфейс установки

Следующие команды описывают будущий deployment-инструмент; в репозитории они ещё не реализованы. Имена могут измениться при реализации, но операции должны сохраняться.

```
portalctl validate-bundle --bundle ./release-bundle
portalctl preflight --inventory ./production.yml
portalctl install --inventory ./production.yml --offline
portalctl verify --inventory ./production.yml
portalctl backup --inventory ./production.yml
portalctl upgrade --inventory ./production.yml --version X.Y.Z
portalctl restore --inventory ./recovery.yml --backup BACKUP_ID
```

Preflight проверяет версии ОС, диск, DNS, время, CA, доступы, права каталогов и отсутствие конфликтующих портов. Install импортирует images/пакеты, применяет конфигурацию и миграции, создаёт начальные сущности и запускает проверку. Повтор install не должен дублировать бюджеты, пользователей или настройки.

Upgrade использует health-check, согласованное окно, резервную копию и поузловое обновление где это возможно. Миграции следуют expand/contract. Rollback бинарников не гарантирует rollback необратимой миграции: совместимость фиксируется в манифесте; восстановление данных является отдельной операцией.

Disaster recovery выполняется на чистой инфраструктуре с проверкой целостности, ledger, ACL и активных заданий. Восстановление не должно случайно возобновить уже выполненную рассылку или другой внешний эффект.

29. Наблюдаемость, аудит и безопасность

Оператор наблюдает API latency/errors, очередь, активные runs, доступность endpoints, usage completeness, резервы бюджета, document pipeline, sandbox capacity, replication и backup. End-to-end trace связывает API request, run, model call, tool call и артефакт. Экспорт телеметрии разрешён только утверждённым внутренним получателям.

Событие	Обязательные поля	Ограничение содержимого
Вход/блокировка	Principal, IdP, время, результат, источник	Не хранить пароль или токен
Изменение политики	Автор, scope, revision, результат	Секреты маскируются
Модельный вызов	Run, connection, price version, usage, duration	Prompt/response не журналировать полностью по умолчанию
Инструмент	Principal, tool/version, разрешение, статус	Аргументы редактируются по schema redaction
Sharing	Объект, grant/revoke, получатель, роль, срок	Не раскрывать содержание документа
Инфраструктурная операция	Approved playbook/version, инициатор, targets, результат	Без произвольного shell и root-secret в UI

Основные угрозы: межпользовательская утечка, prompt injection из документов/веба, SSRF, обход квот, дублирование внешнего действия, escape песочницы, компрометация connector, утечка секретов, вредоносное обновление, неконтролируемый egress и переполнение ресурсов.

Контрмеры включают серверную авторизацию, разделение trust zones, read/write permissions инструментов, изоляцию execution, ограничения сети, проверку supply chain и тесты отрицательных сценариев. Данные из документа не получают полномочия system-инструкции. Вывод модели не принимается как доказательство разрешения действия.

Отдельный документ «Пакет для службы информационной безопасности» содержит подробную модель угроз, сетевые правила, реестр мер, перечень испытаний, регламенты и форму согласования. Он не является сертификатом соответствия и должен быть дополнен нормами конкретной организации.

30. Проверки и критерии приёмки

Приёмка включает функциональность, безопасность, экономику и эксплуатацию.

Производительность модели измеряется отдельно от overhead портала; нагрузочные тесты приложения используют согласованный mock API и небольшой проверяемый набор реальных вызовов, чтобы не смешивать пределы внешнего провайдера с пределами портала.

ID	Проверка	Критерий
A01	Два типа модельных подключений	Работают внутренний API и разрешённый внешний API; секреты скрыты
A02	Возможности API	Неподдерживаемый tool/JSON/streaming блокируется до выполнения
A03	LDAP и блокировка	Группа даёт разрешение; отзыв действует в согласованный срок
A04	Разделение данных	Чужие файлы, chunks, события, экспорт и названия закрытых объектов не раскрываются
A05	Общий бюджет	Параллельные calls не расходуют один резерв дважды; ledger сверяется
A06	Аналитика	Filters/aggregates соответствуют ledger/events; share и use различаются
A07	Помощники	Версия закрепляется в run; публикация не передаёт права автора
A08	Продолжительное задание	Закрытие UI не останавливает run; worker fault обрабатывается
A09	Интернет-изоляция	Workers/sandbox не выходят в интернет и не используют обходной проху
A10	Search/fetch	Разрешённый search работает; SSRF/redirect/private IP блокируются
A11	Код	Отдельная среда, тесты, diff и ограничения ресурсов
A12	MCP/SMTP	Авторизация на вызове; запрещённый адресат/endpoint отклоняется
A13	Offline install	Установка на чистых VM без общего интернета; повтор идемпотентен
A14	Upgrade/restore	Проверенный сценарий миграции и восстановления из backup
A15	Экономика	Отчёт качества и всех costs по runtime; нет скрытых retries

ID	Проверка	Критерий
A16	Темизация UI	Цвета/иконки/плотность сохраняются; contrast и layout проверены
A17	Глобальное управление	Revision, права и аудит; инфраструктурная операция ограничена playbook

Предлагаемые ориентиры overhead приложения: p95 обычного control API до 500 ms при согласованной нагрузке; SSE-событие показывается до одной секунды после сохранения; запуск готового worker до пяти секунд без очереди и cold sandbox. Эти ориентиры не включают генерацию модели, parsing большого файла и ожидание capacity.

Нагрузочные профили: 20, 200 и 600 открытых сессий; отдельно 10, 50 и 120 runs; отдельно кодовые среды. Измеряется скорость постановки задания, DB contention, очередь, память, аналитический lag и поведение после потери узла. Требования окончательно фиксируются до R3.

31. Порядок реализации и результаты этапов

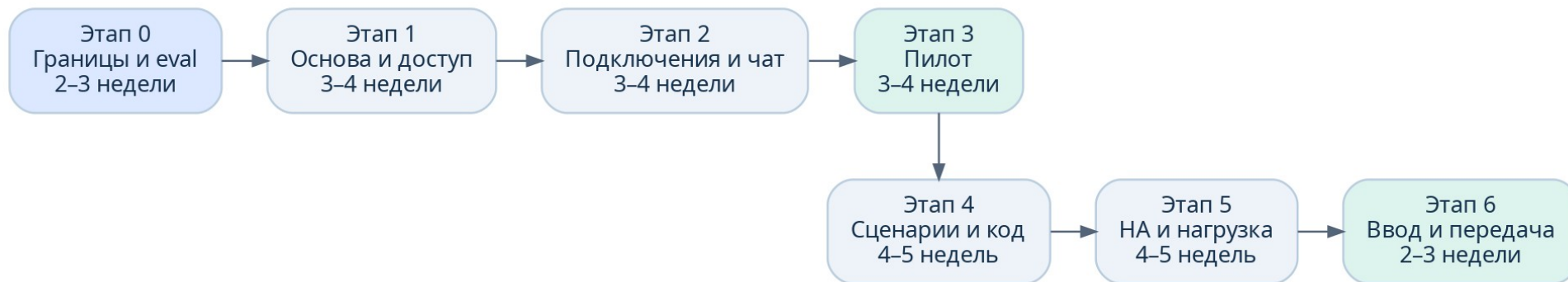


Рисунок 6. Предлагаемая последовательность этапов. Сроки — диапазоны оценки, календарь начинается после готовности команды и среды.

Полный план вынесен в отдельный DOCX. Здесь фиксируются контрольные точки, чтобы описание продукта и план не расходились.

Этап	Результат	Условие перехода
0. Границы и eval	Согласованные требования, прототип UI, тестовый корпус, ADR runtime/deployment	Закрыты вопросы прав, API и базовых сценариев
1. Основа	Repository/CI, API/UI shell, OIDC, spaces, secret handling, offline skeleton	Отрицательные ACL-тесты и установка работают
2. Подключения и чат	Каталог API, streaming, версии помощников, ledger и аналитика	Два типа подключений и групповой бюджет проверены
3. Пилот	Документы, search, один workflow, события и monitoring	Пользователи завершают сквозные задания; ИБ допускает пилот
4. Функциональная версия	RAG, сценарии, согласования, код, MCP, SMTP и sharing	Контрактные тесты и экономический regression пройдены
5. Распределение и НА	Multi-VM deployment, capacity, upgrade/restore и fault tests	Согласованная нагрузка и восстановление подтверждены
6. Ввод	Документация, обучение, эксплуатационные регламенты и backlog	Приёмка, владельцы и каналы поддержки назначены

Оценка базовых работ — 21–28 недель; с резервом на инфраструктуру, согласование и неизвестные интеграции — 24–32 недели. Оценка пилота R1 — 11–15 недель при наличии необходимых API/LDAP и команды. Это не обещание сроков; этап 0 уточняет backlog и capacity команды.

Предлагаемый состав: технический лидер/backend, второй backend-разработчик, frontend-разработчик, runtime/integration-разработчик, QA и DevOps; участие UX, ИБ и владельцев корпоративных сервисов. Часть ролей может совмещаться с изменением сроков. Детальный план содержит зависимости, владельцев и критерии готовности.

32. Риски, предположения и открытые решения

Вопрос / риск	Предлагаемое действие	Когда решить
Неизвестная concurrent-нагрузка	Измерить пилот; отдельно считать чат, parsing и code	До фиксации R3
Экономика harness	Eval на одной модели; стоимость принятого результата	Этап 0 и каждый runtime-upgrade
Возможности внешних API	Контрактные тесты usage/tools/streaming/cancel	До публикации подключения
Сложность Kubernetes	Compose + Ansible по умолчанию; Nomad/K8s только по критериям	Этап 0
Недоступность LDAP/SMTP/Git	Mock для разработки и тестовый endpoint владельца	Этап 1
Общий бюджет при параллельности	Ledger reservations, неизвестный usage и fault tests	Этап 2
Нет корпоративного S3/HA PostgreSQL	Отдельный infrastructure backlog и ресурсный расчёт	До R3
Неполные mirrors для code	Каталог технологий и заранее подготовленные sandbox images	До coding-пилота
License/Enterprise functions	Проверить выбранные версии и нужные возможности	До включения компонента в поставку
Privacy аналитики	Роли, минимальные события, export ACL и retention	До пользовательского пилота
Темизация ухудшает читаемость	Ограниченные design tokens и проверка контраста	Этап 1
Несогласованные требования ИБ	Передать отдельный пакет, провести review и закрыть замечания	Этап 0 → R3

Не требуется согласовывать все будущие возможности до первой строки кода. Необходимо заранее закрыть границы данных, сетевые пути, идентичности и критерии приёма. Неизвестные размеры нагрузки и точные цены не блокируют реализацию учёта и совместимого каталога подключений.

33. Источники и терминология

Проверка документации: 02.10.2026. Источники подтверждают возможности компонентов и публичные архитектурные подходы; собственные проектные требования, сроки, ресурсы и численные пределы являются предложениями этого документа. Версии и лицензии повторно проверяются перед поставкой.

Ссылка	Первичный источник
S01	Deep Agents: возможности и контекст
S02	LangGraph: сохранение состояния
S03	Temporal: self-hosted deployment
S04	LiteLLM: бюджеты и ограничения
S05	Keycloak: администрирование и LDAP
S06	Docker Compose в production
S07	K3s: offline-установка
S08	SearXNG: исходящие соединения
S09	Docling: обработка документов
S10	OpenHands Software Agent SDK
S11	pgvector: retrieval и фильтрация
S12	Deep Agents: ограничения backend
S13	MCP: рекомендации по безопасности
S14	OpenAI Agents SDK: исполнение в приложении
S15	Dify: лицензия
S16	Pydantic AI: официальный обзор
S17	Anthropic: построение эффективных агентов
S18	Anthropic: Claude Agent SDK
S19	Anthropic: архитектура Managed Agents
S20	Anthropic: multi-agent research
S21	Ansible: управление Compose v2
S22	Nomad: архитектура и назначение

Ссылка	Первичный источник
S23	Nomad: лицензия
S24	K3s: сетевые требования
S25	Temporal: выполнение Activities

Точные порты необязательного Nomad-профиля подтверждаются deployment review выбранной версии.

Harness — готовый агентный цикл и работа с контекстом/tools. **Runtime** — процесс, исполняющий этот цикл. **Workflow** — управляемый сценарий. **Control plane** — управление политиками и конфигурацией. **Ledger** — журнал бюджетных операций. **ACL** — права объекта. **Egress** — исходящий сетевой доступ. **RAG** — поиск доступных фрагментов для формирования ответа. **Run** — конкретное выполнение задания. **Sharing** — выдача доступа или публикация; **usage** — фактическое использование. **RPO/RTO** — допустимая потеря данных и время восстановления.

34. Дизайн портала и панели управления

Интерфейс состоит из двух связанных режимов: «Портал» для сотрудников и «Управление» для администраторов/операторов. Переход между ними находится в постоянной навигации. Право входа в режим управления не даёт автоматического чтения пользовательских документов.

Портал

Левая панель: рабочая область, помощники, сценарии, документы, инструменты, подключения и аналитика. Центр: диалог, редактор или список. Правая панель контекста появляется только когда полезна — источники, версии, артефакты и лимиты задания. В длинном диалоге input остаётся доступным, а progress не вытесняет ответ.

Помощники отображаются карточками с назначением, владельцем, версией и областью публикации. Editor имеет разделы «Инструкция», «Модель», «Инструменты», «Знания», «Ограничения» и «Доступ». Сохранение черновика и публикация версии разделены. Пользователь видит только доступные tools, без списка скрытых запрещённых ресурсов.

Экран подключений показывает внутренний/внешний маршрут, тип API, health, capabilities и разрешённые группы. Полный endpoint доступен уполномоченному администратору; ключ не показывается. Аналитика имеет вкладки «Расход», «Использование», «Инструменты», «Документы» и «Совместный доступ».

Управление

Разделы: обзор, задания, очереди/workers, инфраструктура, политики, глобальная конфигурация, оформление и аудит. В обзоре — доступность сервисов, backlog, budget warnings и незакрытые согласования. Задание имеет timeline, закреплённые версии, usage, разрешённые артефакты и действия остановки/продолжения по роли.

Инфраструктурный экран показывает VM, закреплённые роли, версии, capacity, maintenance и drift. В Compose-профиле добавление worker означает применение inventory через утверждённый deployment process, а не обещание автоматически создать VM. Оркестратор Nomad/Kubernetes отображается только если такой профиль реально установлен.

Настраиваемая визуальная система

Параметр	Уровень настройки	Ограничение
Название, логотип и акцент	Организация / пространство по разрешению	Проверенные изображения; никакого исполняемого HTML/SVG
Light/dark/system	Пользователь; глобальный default	Сохранить контраст и состояния disabled/error
Цвета	Design tokens: surface/text/accent/border/status	Проверка WCAG-контраста; semantic colors сохраняют смысл
Иконки	Утверждённые наборы и mapping пунктов	Accessible labels, одинаковая оптическая ширина

Параметр	Уровень настройки	Ограничение
Плотность и размеры	Comfortable/compact, размер текста и панели	Минимальная кликабельная зона и перенос текста
Скругления и карточки	Настройки темы	Не менять layout произвольным пользовательским CSS
Форматы	Язык, дата, timezone и валюта	Не менять валюту фактического ledger без конвертации

Все экраны используют общие tokens. Новая тема предварительно просматривается на портале, аналитике и панели управления. Администратор может вернуть прошлую revision; личные настройки не переопределяют ограничения безопасности. В интерактивном макете используются собственные SVG-иконки, синтетические данные и локальное сохранение настройки оформления.

Состояния и взаимодействия

Для каждого экрана проектируются loading, empty, error, forbidden, partial-data и success. Действие с внешним эффектом имеет review параметров; отмена подтверждается. Фильтры аналитики сохраняются при drill-down и возврате. Таблицы имеют поиск, сортировку/пагинацию в production и доступный export. Клавиатурная навигация, видимый focus и подписи иконок обязательны.

Предоставленный HTML-макет является проектным артефактом, а не подключённым production-порталом. Он нужен для согласования структуры, визуальной системы и основных взаимодействий до реализации API.

Рабочая область: визуальный макет

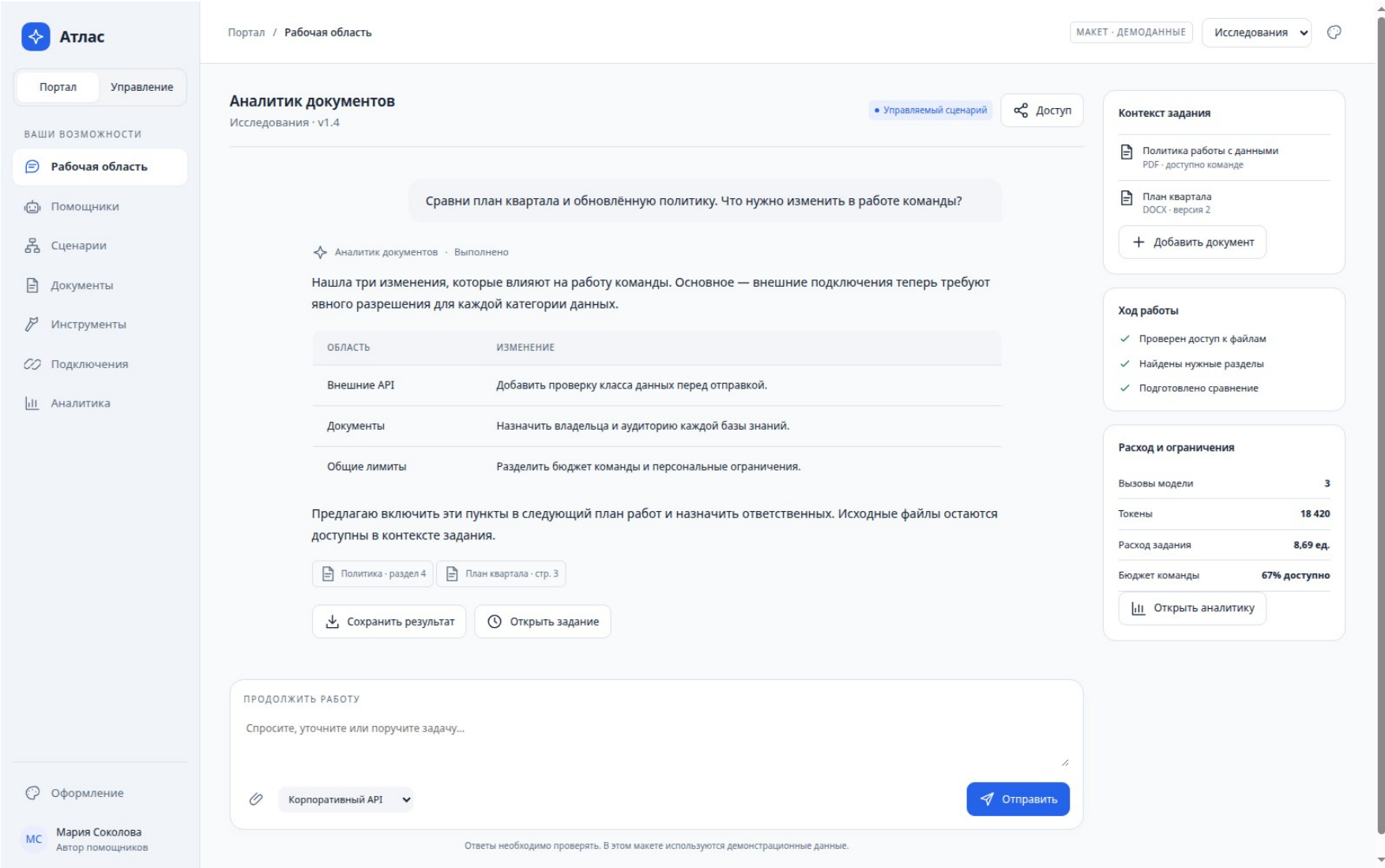


Рисунок 7. Диалог, контекст документов, источники и бюджет задания. Макет на синтетических данных.

Аналитика: визуальный макет

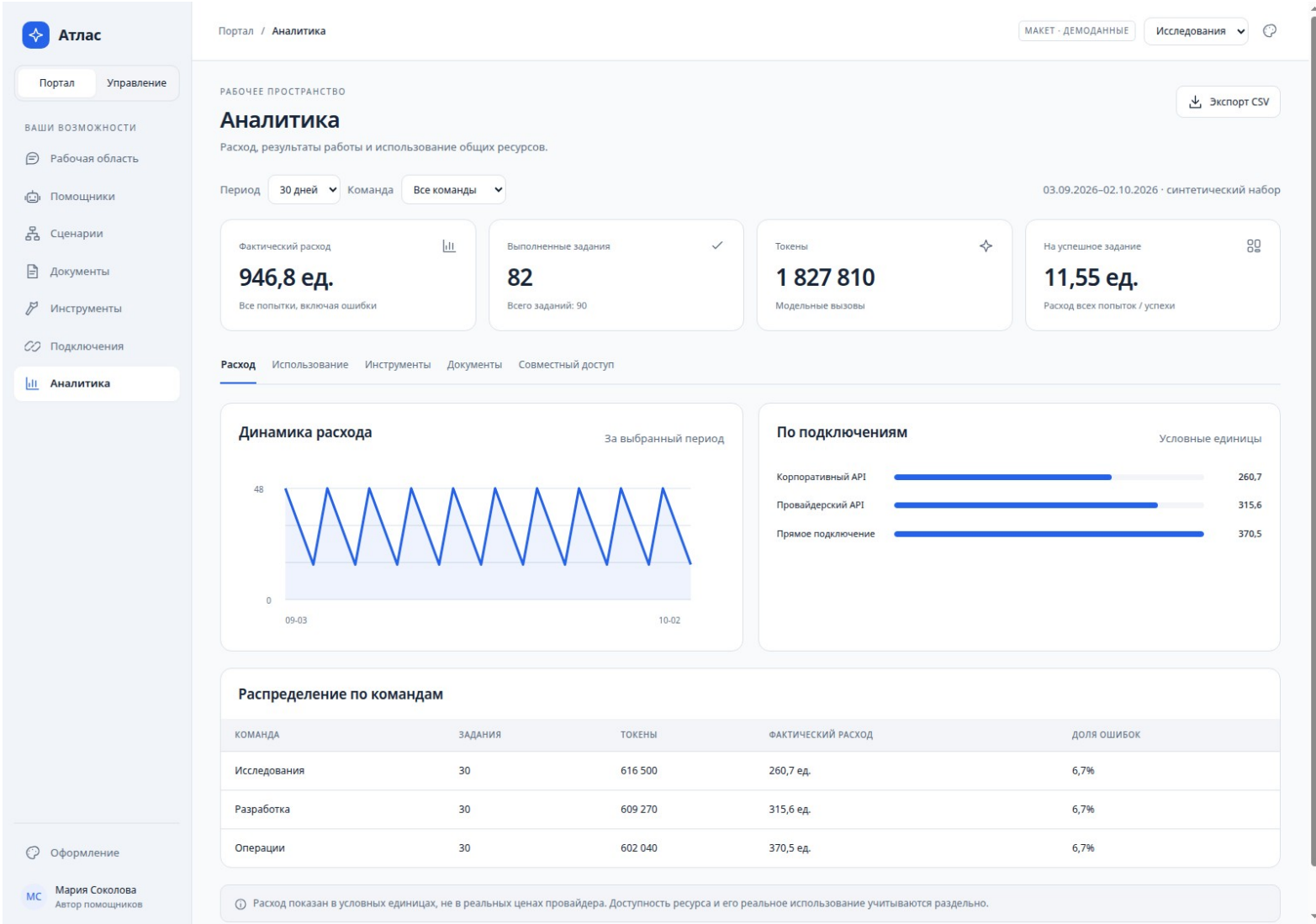


Рисунок 8. Динамика расхода и распределение по API-подключениям; цены условные.

Управление: визуальный макет

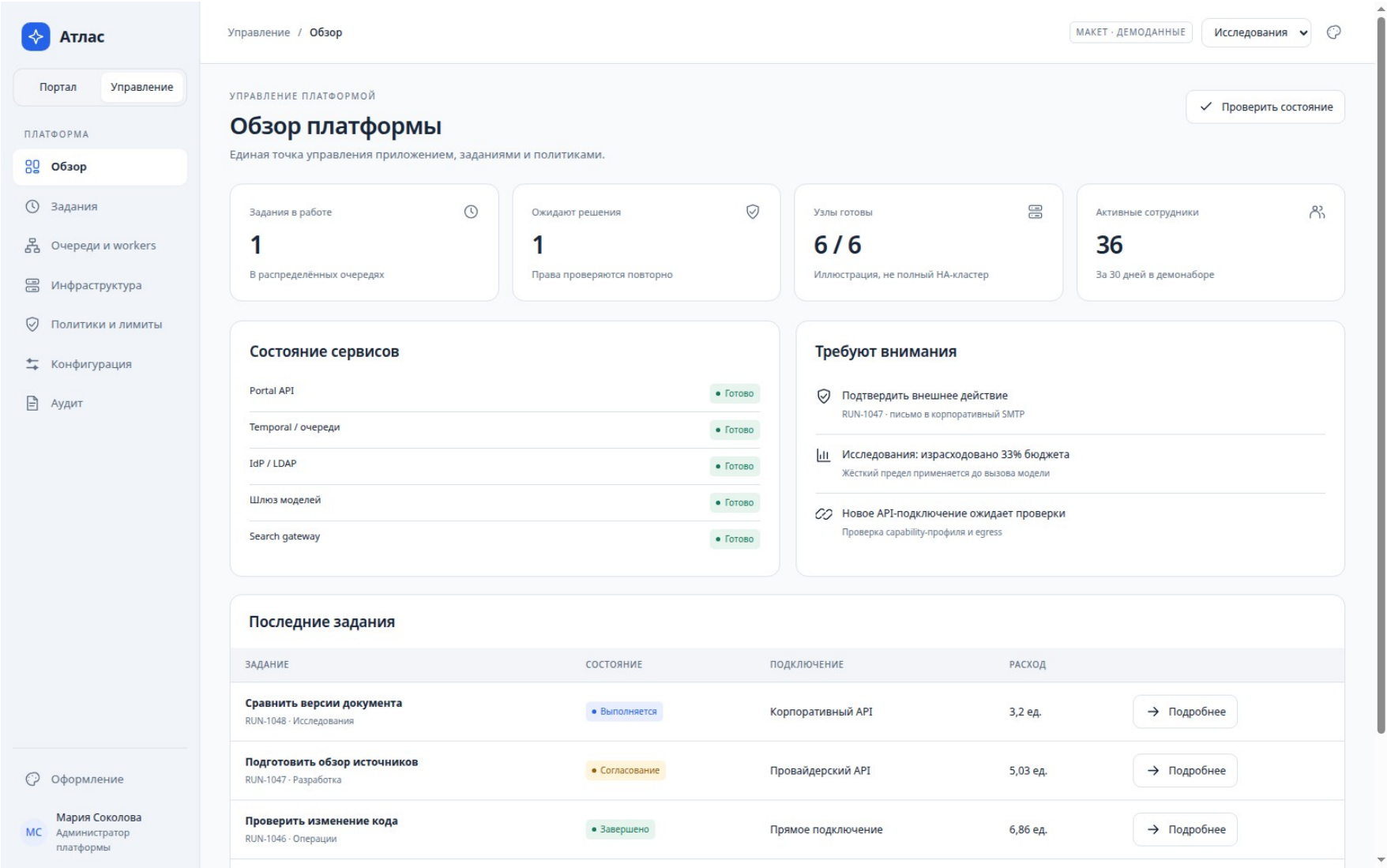


Рисунок 9. Обзор приложения: сервисы, задания, согласования и предупреждения.

Инфраструктура: визуальный макет

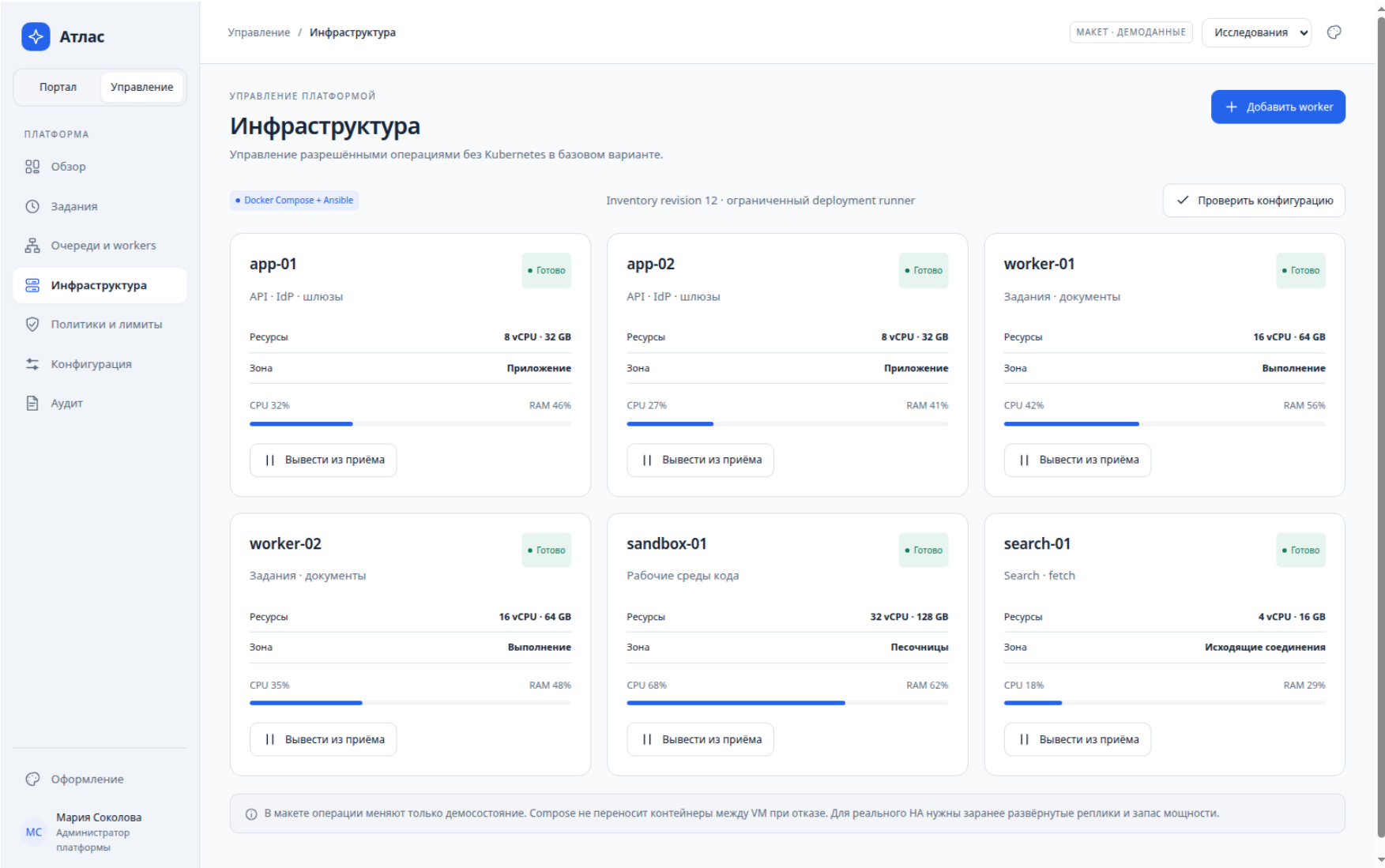


Рисунок 10. Разрешённые операции с узлами. Карточки иллюстративны и не заменяют production-инвентарь.

Настройки оформления: визуальный макет

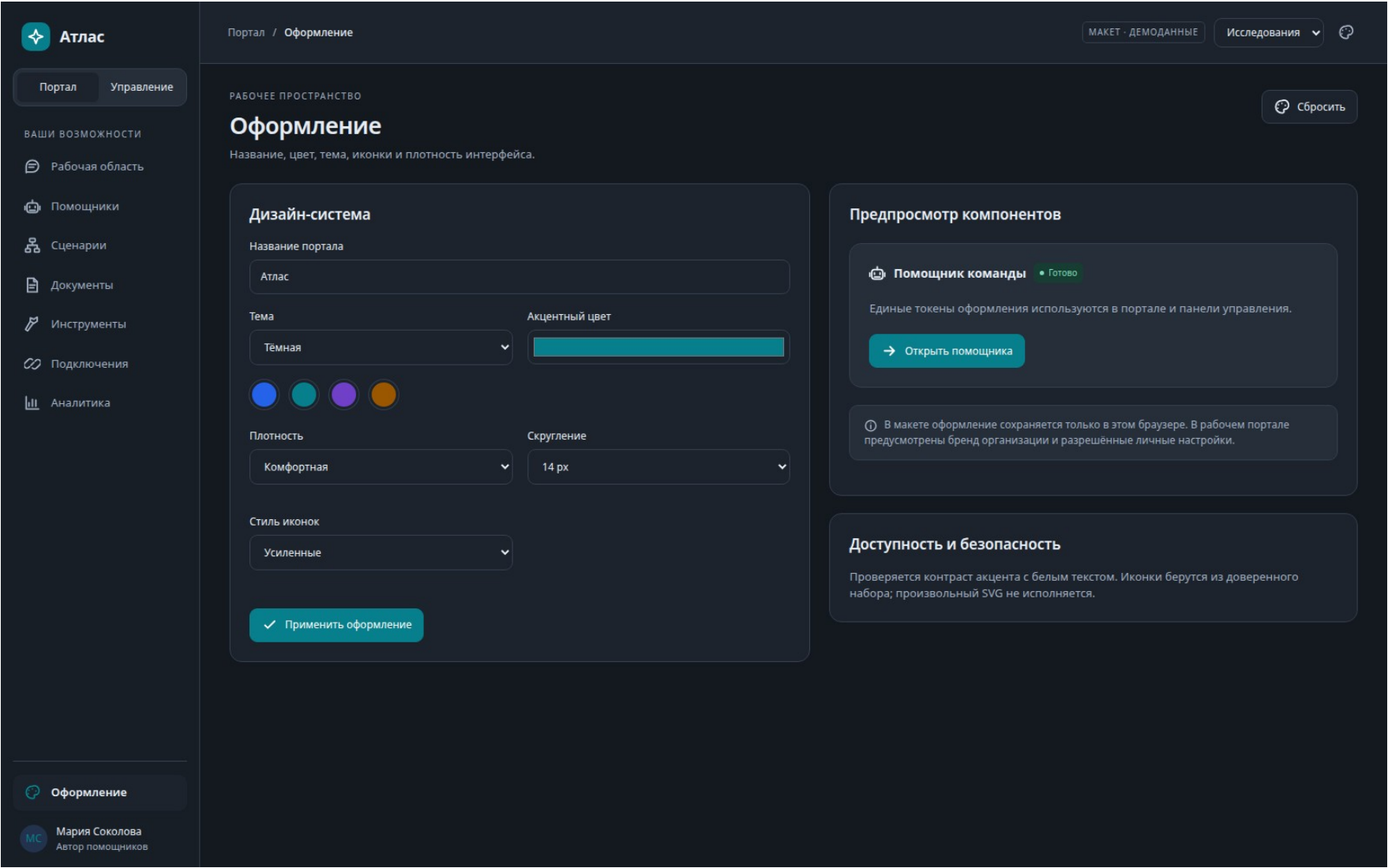


Рисунок 11. Настраиваемая тема: тёмный режим, бирюзовый акцент и усиленный набор иконок.