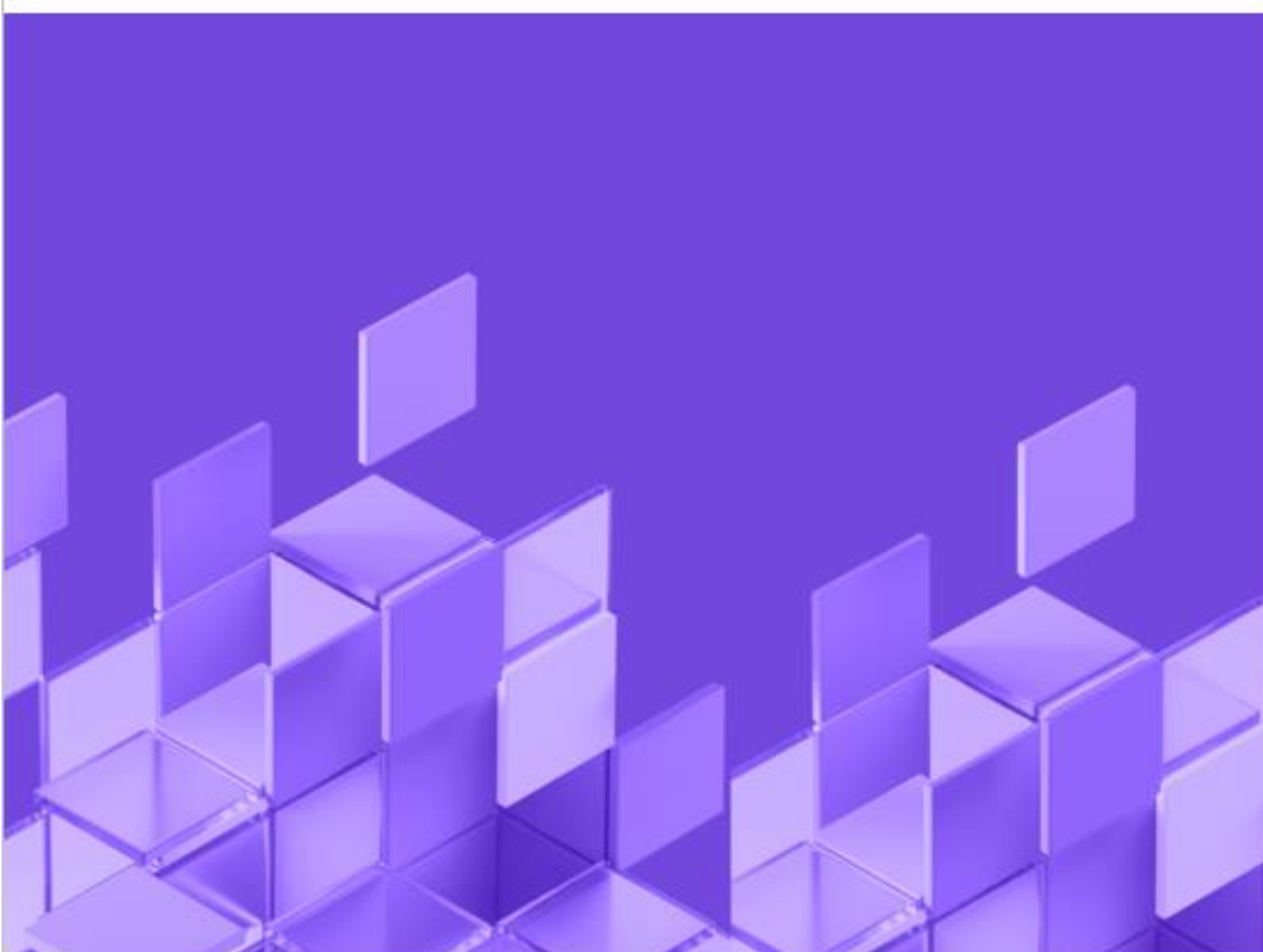
НЕСТ | МОДУС

ФУНКЦИОНАЛЬНОЕ ОПИСАНИЕ СИСТЕМЫ



Глоссарий

Термины и сокращения приведены в Таблицах 1 и 2 настоящего документа.

Таблица 1.1. Термины и сокращения

1	Event-driven architecture, Событийно-ориентированная архитектура	Парадигма построения архитектуры ПО вокруг создания, определения, потребления и реакции на события.
2	Микросервис, Микросервисная архитектура	Подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном закрытом пространстве и коммуницирует с остальными, используя легковесные механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды.
3	Kubernetes	Система оркестрации контейнеров, для автоматизации развёртывания, масштабирования контейнеризированных приложений и управления ими.
4	Pod, Под	Базовая единица для управления и запуска приложений в рамках Kubernetes, содержит один или несколько контейнеров, которым гарантирован запуск на одном узле, обеспечивается разделение ресурсов и единый IP-адрес в кластере.
5	Container, Контейнер	Метод виртуализации, при котором ядро операционной системы Worker Node кластера Kubernetes поддерживает в себе несколько изолированных экземпляров пространства пользователя (контейнеров). Эти экземпляры с точки зрения пользователя полностью идентичны отдельному экземпляру операционной системы. Обеспечивается полная изолированность контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга.
6	Quarkus	Java фреймворк от компании Red Hat (IBM), разработанный для развёртывания в Kubernetes. В основе Quarkus — Vert.x, Netty, поверх которых используются реактивные фреймворки и расширения.
7	Angular	фреймворк от компании Google для создания бесшовных веб-приложений.
8	Lit	фреймворк для создания быстрых, легких веб-компонентов.
9	Контейнер (как способ поставки)	Готовый, упакованный программный продукт, поставляемый «как есть»; на него распространяется вендорская поддержка и обновление из репозитория продукта согласно лицензионному соглашению. Самостоятельная доработка этих сервисов не предусматривается, их возможности зафиксированы их текущей реализацией, то есть дают лишь то, что предоставляется в их API и правилах использования, описанных в документации.
10	Приложение (как способ поставки)	Опенсорс компонент, передаваемый «как есть» в рамках свободной лицензии с полноценными правами root. На него распространяется вендорская поддержка и обновление из репозитория продукта согласно лицензионному соглашению при условии, что заказчик не вносит в него самостоятельные изменения. При внесении изменений вендорская поддержка перестает действовать на компонент, в который они были внесены.

11	Сервис (НОТА МОДУС)	Программа, предоставляющая API для работы с бизнес-объектами. Является минимальной единицей для развертывания.
12	Метод (НОТА МОДУС)	Операция предоставления сервисом для получения и/или обработки данных.
13	Бизнес-объект (НОТА МОДУС)	набор свойств сущностей, позволяющих хранить информацию о ней.
14	Поле, атрибут (НОТА МОДУС)	Свойство бизнес-объекта у которого есть тип данных.
15	Экран (НОТА МОДУС)	Набор структурированных визуальных элементов/ визуальных представлений.
16	Виджет (НОТА МОДУС)	Визуальный компонент интерфейса, может вмещать в себя более простые компоненты.
17	Индекс (НОТА МОДУС)	Элемент для полнотекстового поиска. У индексов есть атрибуты, которые могут настраиваться.
18	Бизнес-процесс (НОТА МОДУС)	Создание связей между компонентами приложения.
19	Пользователь (НОТА МОДУС)	Человек или робот, который работает с платформой.
20	Роль (НОТА МОДУС)	Набор бизнес-задач, решаемых пользователем и определяющих его права доступа к сервисам.
21	Группа доступа (НОТА МОДУС)	Набор прав для обеспечения доступа к записям данных сущности.

Введение

Назначение документа

Документ «Описание модулей ядра НОТА МОДУС» содержит описание ключевых решений по реализации НОТА МОДУС в части системной архитектуры, а именно программных платформ, решений типовых задач, взаимодействия сервисов, инфраструктурных сервисов и пр.

Документ предназначен для фиксации проектных решений по практическим вопросам реализации доработок системы.

Предполагается, что данный документ будет актуализироваться по мере накопления опыта применения описанных в нем решений и возникновения новых потребностей, при этом все другие документы проекта не должны вступать с ним в противоречие.

Для чего не предназначен документ

Документ «Описание модулей ядра НОТА МОДУС» не описывает используемые аппаратные средства, их конфигурацию, технические параметры, сетевые настройки, имена серверов, ip-адреса, процесс установки. Подробности технической конфигурации и развёртывания описаны в документе «Руководство администратора».

Функциональные и технические требования к бизнес-сервисам не рассматриваются в данном документе и должны быть описаны в Технических заданиях на сервисы.

Место документа

Документ «Описание модулей ядра НОТА МОДУС» входит в состав документов, описывающих аспекты архитектуры НОТА МОДУС на всех стадиях проекта; по порядку изложения основывается на «Концептуальной архитектуре НОТА МОДУС».

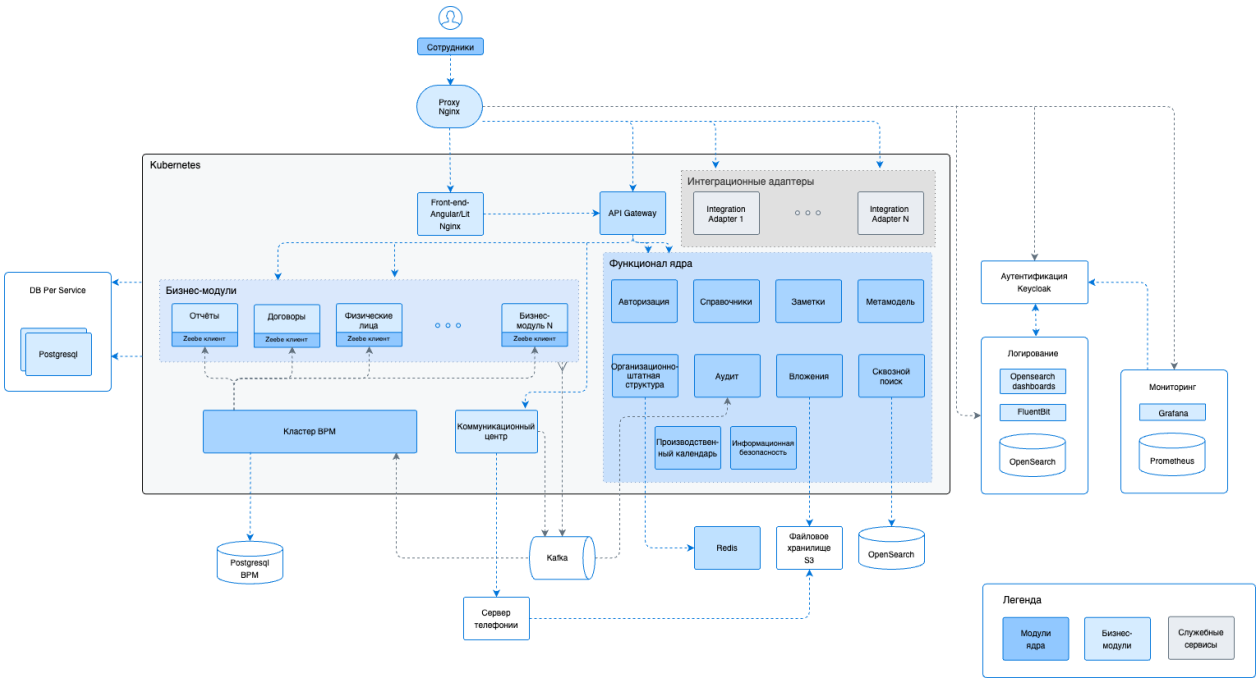
Целевая аудитория

Целевая аудитория данного документа:

- Команда проекта НОТА МОДУС: менеджеры, архитекторы, технологи и разработчики.
- Сотрудники других проектов, как выставляющих требования, так и дорабатывающих сервисные модули НОТА МОДУС: архитекторы, технологи и разработчики со стороны подрядчиков.
- Сотрудники службы поддержки НОТА МОДУС: администраторы серверов и другого ПО, составляющего НОТА МОДУС.

Модули и компоненты системы

Диаграмма модулей и компонентов



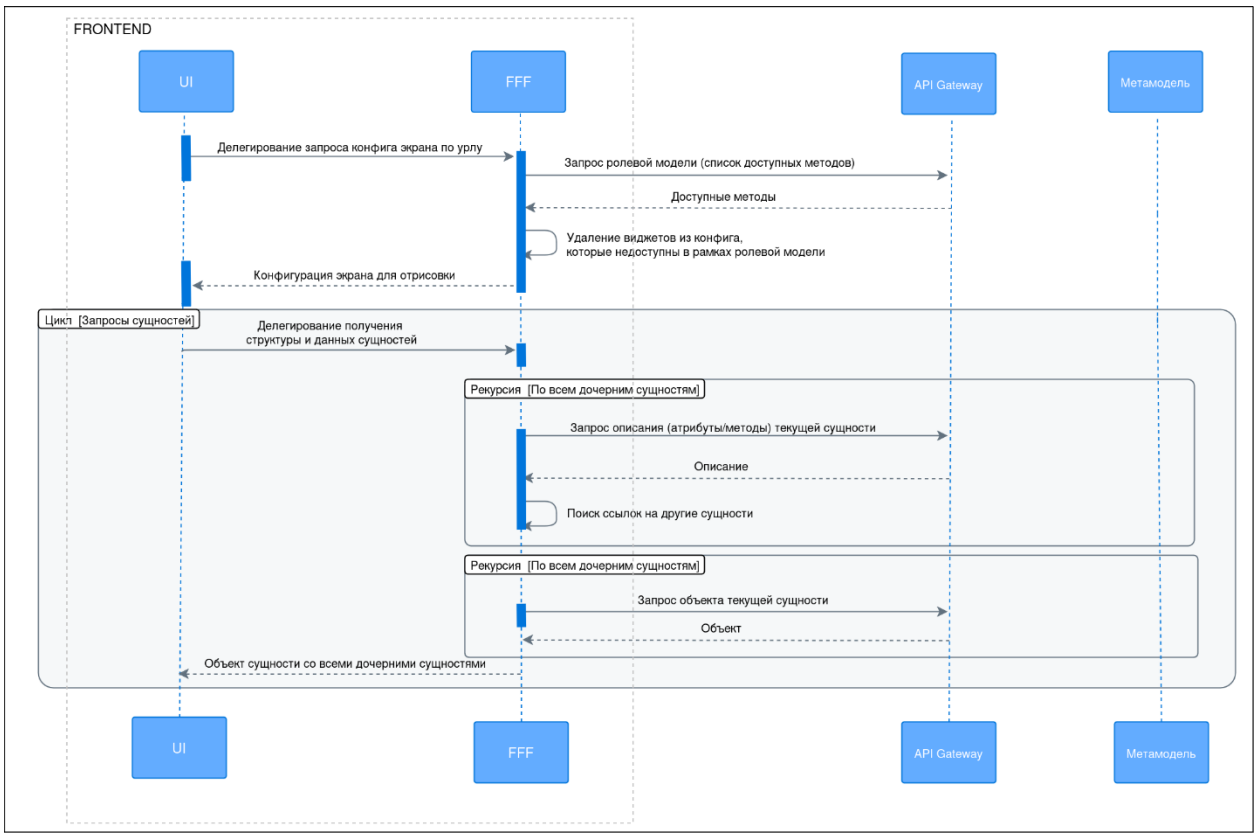
Описание модулей и компонентов

Frontend Angular/Lit Nginx

Frontend For Front-end (FFF) – это прослойка на стороне Frontend между API Gateway и UI, которая умеет запрашивать и подтягивать данные БО из метамодели (запрос модулей БО), а также фильтровать и дополнять их данными, через запросы к API Gateway (запрос справочников, данных о БО и фильтрация данных).

FFF позволяет быстрее и проще подтягивать нужные данные из Метамодели, фильтруя и дополняя их данными их API Gateway. При этом пользователь сразу получает готовые данные и видит их на фронте, а также получает возможности их создания, редактирования и удаления (FFF в данном случае выступает в роли агрегатора данных и действий с ними). Видимым результатом для пользователя являются действия, отображаемые на FE в виде структур, данных и справочников.

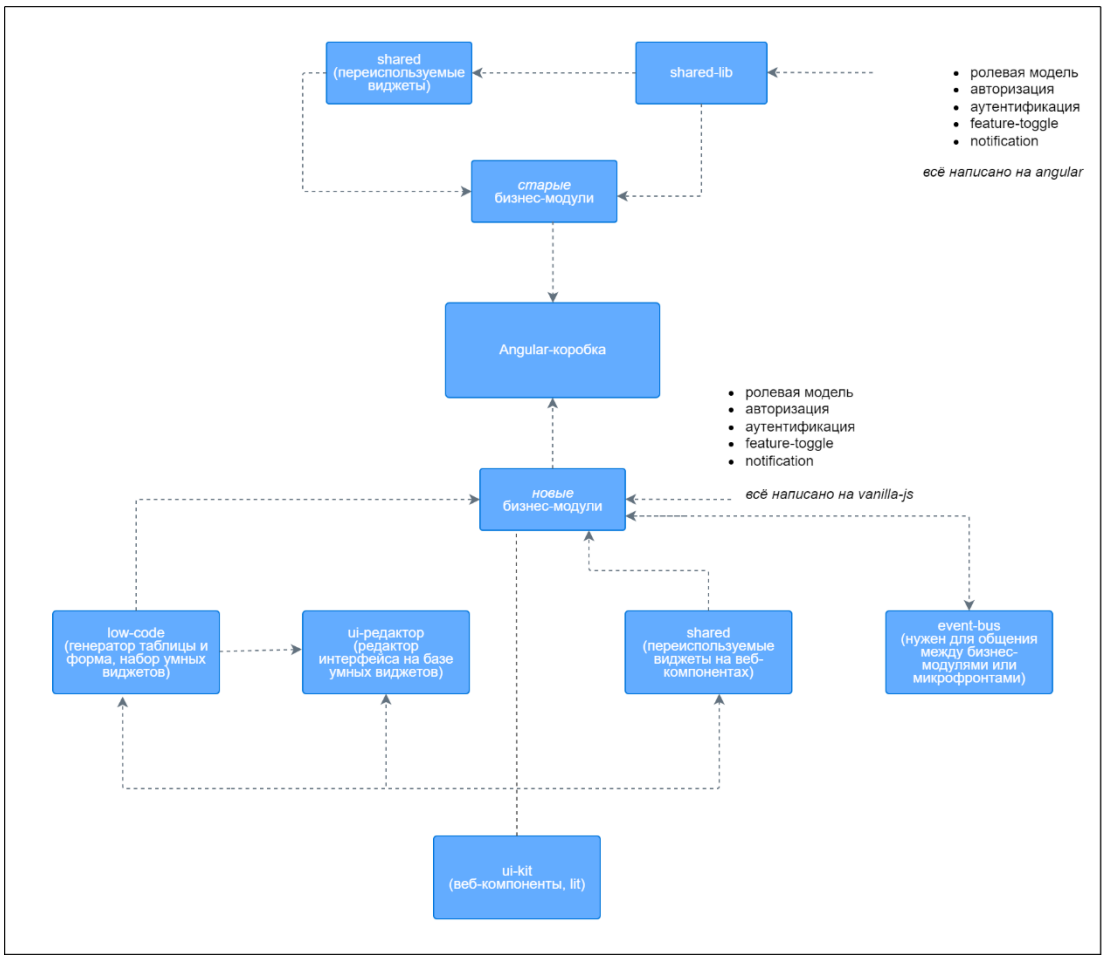
Схема работы FFF на примере запроса Get List (получения данных и структуры списка сущностей по наименованию бизнес-объекта)



UI приложения генерируется на web-серверах Nginx, разворачиваемых как микросервис с автоматическим масштабированием контейнеров в зависимости от текущей нагрузки.

Процесс сборки приложения Frontend представлен на схеме ниже

- Архитектура. Frontend. Синхронизация с проектами
- Описание архитектуры по разделам. Frontend



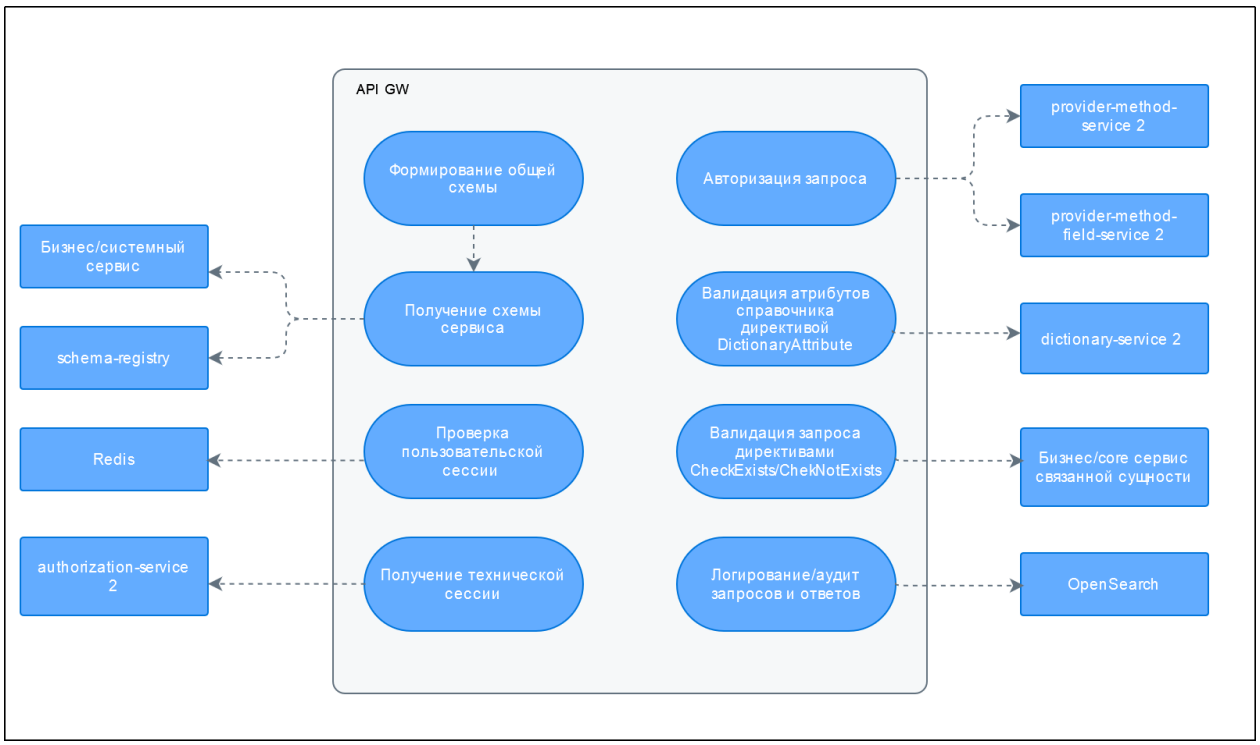
API Gateway

API Gateway представляет собой шлюз, который используется для проксирования вызовов по единому адресу на сервисы backend.

Задачи:

1. Публикация API
2. Авторизация вызовов API
3. Валидация схем сервисов
4. Проксирование запросов с FE
5. Логирование запросов/ответов сервисов
6. Проверка справочных значений
7. Проверка связанных сущностей

Функциональные возможности

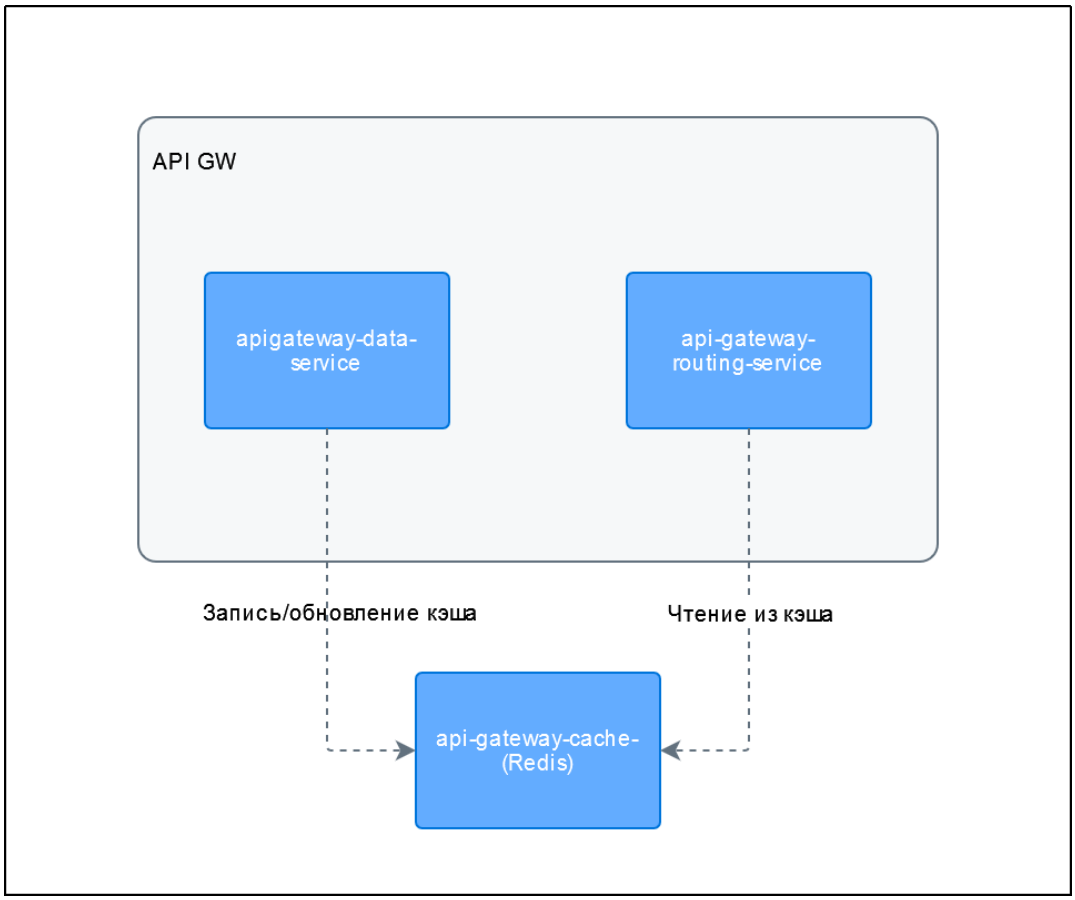


	Функция	Описание
1	Формирование общей схемы	Объединение схем используется для того, чтобы можно было резолвить методы в коде: • определять, какая у метода должна быть структура и валидировать её средствами библиотеки • в клиенте, при ручной проверке, строить запрос, используя автоподстановку Для получения схемы сервиса, шлюз выполняет запрос в schema-registry (при условии, если конфиге был указан блок schema-registry), иначе выполняет запрос в целевой сервис.
2	Проверка пользовательской сессии	Шлюз проверяет сессионные данные пользователя: • получает текущую сессию из запроса; • подтверждает существование данной сессии
3	Получение технической сессии	Выполняется запрос во внутренний кэш приложения, если сессия обновлялась давно, то выполняется запрос в authorization-service (в кэше при этом сессия обновляется)
4	Авторизация запроса	Выполняется поверка прав доступа к вызываемому методу. Проверка выполняется в 2 этапа: 1. Шлюз выполняет запрос в provider-method-service (gRPC getMethods) для получения ролей

		пользователя и доступных методов/саб-методов с ролями 2. Шлюз выполняет запрос в provider-method-field-service (gRPC getMethodFields) для получения доступных атрибутов по ролям Если пользователей не имеет по своей роли доступа к запрашиваемому ресурсу - авторизация не выполнится
5	Валидация атрибутов справочника директивой DictionaryAttribute	Используется для проверки переданных кодов атрибутов по справочникам в dictionary-service. Чтобы проверка была выполнена на уровне шлюза, в контракте объявляется директива DictionaryAttribute
6	Валидация запроса директивами CheckExists/CheckNotExists	Используется для проверки связанной сущности. Чтобы проверка была выполнена на уровне шлюза, в контракте объявляется директива
7	Логирование/аудит запросов и ответов	Шлюз логирует все вызовы, проходящие через него. События, подлежащие логированию, сохраняются OpenSearch

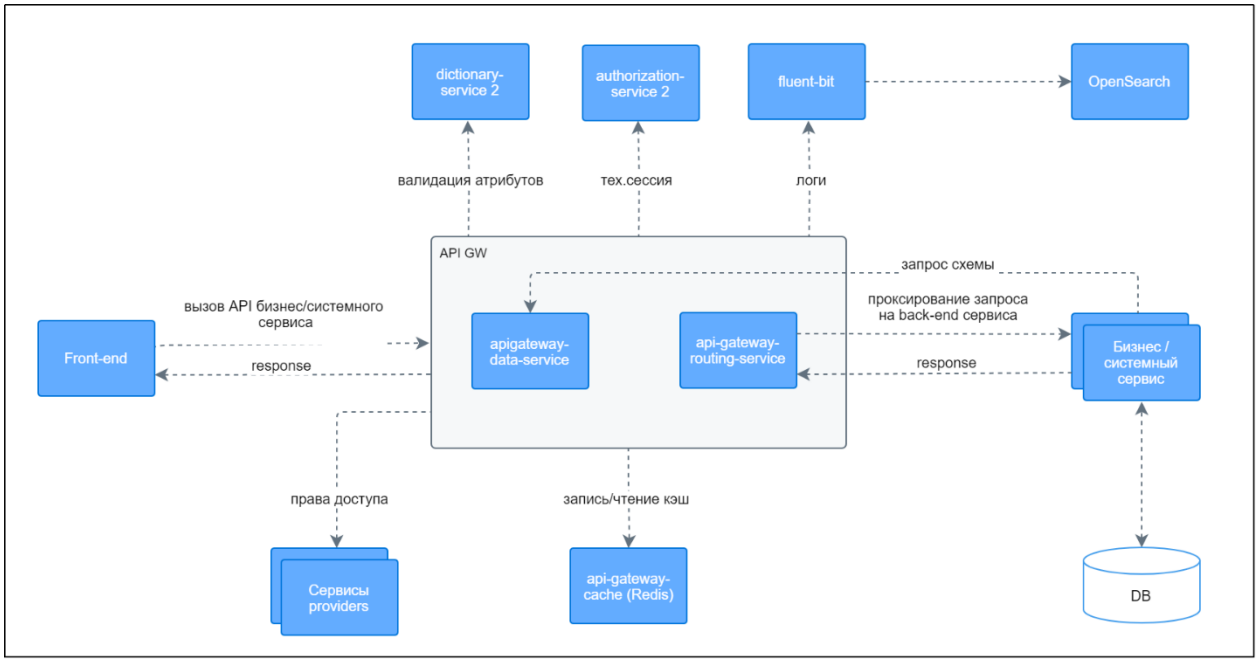
Архитектура

Диаграмма компонентов



	Компонент	Описание	Статус
1	api-gateway-data-service	Сервис, выполняющий следующие задачи: • обращение в бизнес-сервисы для получения контрактов • объединение контрактов в один общий • сохранение в кэш (схемы + связи) • обновление кэша (схемы + связи)	Существует
2	api-gateway-routing-service	Сервис, выполняющий следующие задачи: • чтение данных из кэша • роутинг сервисов на внутренние back-end сервисы на основе запроса	Существует
3	api-gateway-cache	КЭШ Redis	Существует

Диаграмма взаимодействия API GW с другими сервисами



Integration adapters

Микросервисы - адаптеры, обеспечивающие интеграцию с внешними системами.

В коробочную установку НОТА МОДУС не включаются никакие интеграционные адаптеры, так как особенности конфигурации внешних систем обладают слишком большим разнообразием для формирования единого подходящего под все случаи инструмента, даже в случае интеграции с одной и той же системой. Архитектура НОТА МОДУС подразумевает, что в рамках конкретного внедрения системы будет определён scope необходимых конкретному заказчику интеграций, и эти адаптеры будут разработаны под конкретные

требования. Такие микросервисы-адаптеры при этом являются кастомной разработкой, и создаются не на low-code. Примеры интеграционных адаптеров, разработанных в рамках различных проектов внедрения НОТА МОДУС, включают в себя адаптеры к SalesForce, SAP, DaData, Unisender, 1С и другие.

Модули ядра

Авторизация

Модуль "Авторизация" позволяет настроить пользователю определенные разрешения, права доступа и привилегии в системе.

Задачи модуля:

1. Разграничение доступа к ресурсам системы на уровне endpoint (доступные методы и саб-методы).
2. Разграничение доступа к ресурсам на уровне экранных форм и виджетов.
3. Разграничение доступа по полям.

Функциональные возможности

1. Ввод и редактирование методов/саб-методов.
2. Ввод и редактирование ролей.
3. Экспорт методов с опцией полной или пакетной (по заданным критериям) выгрузки в формате JSON.
4. Импорт методов в формате JSON.
5. Настройка endpoint для пользовательских ролей.
6. Настройка экранов и виджетов для пользовательских ролей.
7. Построение иерархии по элементам UI.
8. Запрет на отображение атрибутов пользователю.
9. Запрет на редактирование отдельных атрибутов пользователю.

Архитектура

Модуль "Авторизация" состоит из четырех микросервисов:

1. provider-method-service (Сервис управления методами и саб-методами для доступа к ресурсам).
2. provider-responsibility-service (Сервис управления пользовательскими ролями).
3. provider-view-service (Сервис управления экранными формами ролей).
4. provider-method-field-service (Сервис разграничения прав по полям).

База данных для микросервисов одна (PostgreSQL), каждый сервис использует свой набор таблиц. Сервисы предоставляют API на GQL, а также grpc для удаленного вызова процедур, через который происходит взаимодействие с другими сервисами.

Организационно-штатная структура

Модуль "ОШС" представляет из себя иерархию Организаций ОШС, Подразделений ОШС и связанных Позиций ОШС, на основании которой Пользователи получают доступ к данным CRM

Задачи модуля:

1. Контроль за изменениями в структуре ОШС и доступе сотрудников
2. Отражение актуальной информации о составе компании

Функциональные возможности

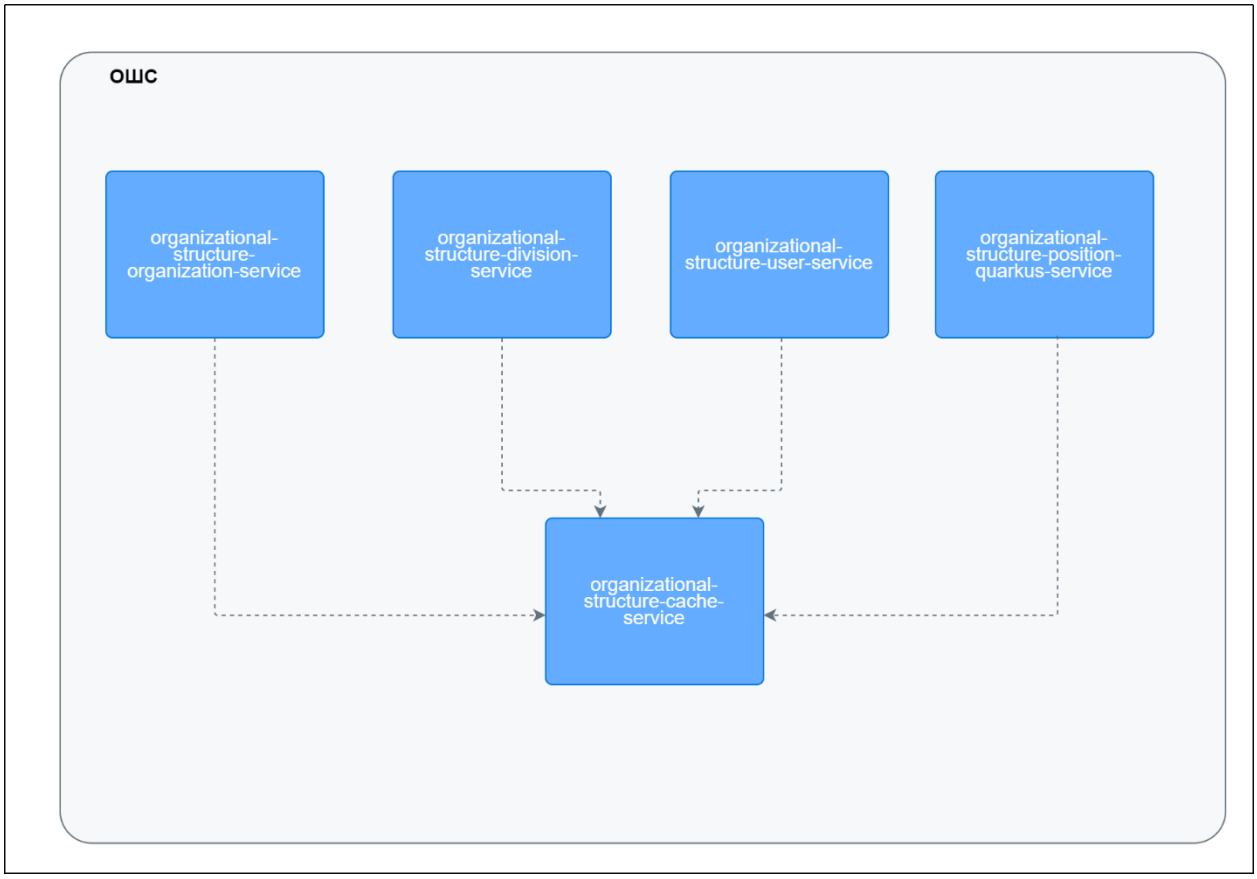
1. Ввод и редактирование организаций
2. Ввод и редактирование подразделений
3. Ввод и редактирование пользователей
4. Ввод и редактирование позиций
5. Просмотр списков, возможность быстрой сортировки и фильтрации единиц ОШС

6. Настройка видимости тех или иных объектов системы в зависимости от заданной Организации (ОШС)

Архитектура

Модуль "ОШС" состоит из 5 микросервисов. В качестве хранения единиц ОШС используется база данных - PostgreSQL. Сервисы предоставляют API на GQL, gRPC и REST (считается depricated - устаревшим), через который происходит взаимодействие с другими сервисами.

Диаграмма компонентов



	Список компонентов	Описание
1	organizational-structure-organization-service	Сервис, обеспечивающий работу с организациями
2	organizational-structure-division-service	Сервис, обеспечивающий работу с департаментами
3	organizational-structure-position-quarkus-service	Сервис, обеспечивающий работу с позициями
4	organizational-structure-user-service	Сервис, обеспечивающий работу с пользователями
5	organizational-structure-cache-service	Сервис для кэширования и быстрого получения сущностей и структур ОШС

Модуль "ОШС" представляет из себя иерархию Организаций ОШС, Подразделений ОШС и связанных Позиций ОШС, на основании которой Пользователи получают доступ к данным CRM

Задачи модуля:

1. Контроль за изменениями в структуре ОШС и доступе сотрудников
2. Отражение актуальной информации о составе компании

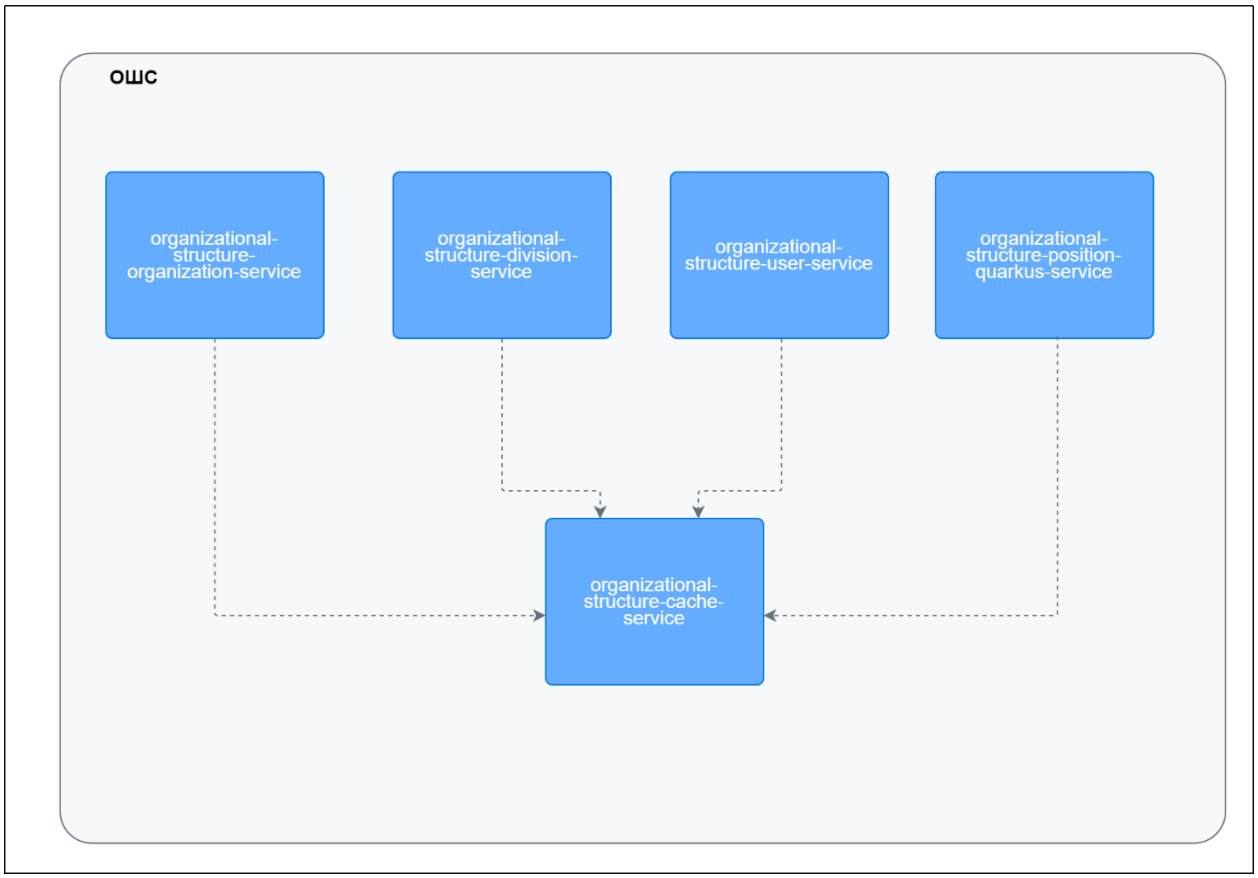
Функциональные возможности

1. Ввод и редактирование организаций
2. Ввод и редактирование подразделений
3. Ввод и редактирование пользователей
4. Ввод и редактирование позиций
5. Просмотр списков, возможность быстрой сортировки и фильтрации единиц ОШС
6. Настройка видимости тех или иных объектов системы в зависимости от заданной Организации (ОШС)

Архитектура

Модуль "ОШС" состоит из 5 микросервисов. В качестве хранения единиц ОШС используется база данных - PostgreSQL. Сервисы предоставляют API на GQL, gRPC и REST (считается depricated - устаревшим), через который происходит взаимодействие с другими сервисами.

Диаграмма компонентов



	Список компонентов	Описание
1	organizational-structure-organization-service	Сервис, обеспечивающий работу с организациями

2	organizational-structure-division-service	Сервис, обеспечивающий работу с департаментами
3	organizational-structure-position-quarkus-service	Сервис, обеспечивающий работу с позициями
4	organizational-structure-user-service	Сервис, обеспечивающий работу с пользователями
5	organizational-structure-cache-service	Сервис для кэширования и быстрого получения сущностей и структур ОШС

Метамодель

Модуль "Метамодели" позволяет описать предметную область бизнеса для её хранения и автоматизации. В модуле описывается:

- Приложение - это набор модулей для реализации единой функциональности. Примеры приложений: CRM, ERP, MES, HR, Help Desk, Документооборот, ТОиР. Приложение задаёт область имён для модулей, объектов, сервисов, процессов
- Модуль - это семантическое объединение сервисов, экранов, объектов в одну группу, реализующее законченный бизнес-функционал. Примеры модулей: Сопровождение сделок, Управление лидами, CPQ
- Бизнес-объект - это набор свойств сущностей, позволяющих хранить информацию о ней (в рамках платформы это одна таблица в БД)
- Атрибут - это свойство бизнес-объекта, у которого есть следующие свойства: наименование, локализация, тип данных, признак: ключевое, признак: обязательность, признак: только для чтения, справочник.
- Сервис - это программа, предоставляющая API для работы с бизнес-объектами, является минимальной единицей для развертывания. В одном модуле может быть несколько сервисов, набор сервисов определяет функциональность модуля
- Метод действия API - это операция, предоставленная сервисом для получения и/или обработки данных (методы GraphQL или Job worker для кластера ZeeBe)

Задачи модуля:

1. Хранение описания предметной области
2. Настройка описания БД
3. Настройка low-code сервисов
4. Создание контракта стандартного API
5. Формирование спецификации для генерации кода low-code сервиса
6. Предоставление описания предметной области другим модулям системы
7. Миграция описания предметной области
8. Описание методов API с помощью подмодуля "Редактор API" и их хранение в системе

Функциональные возможности:

1. Администрирование приложения
2. Администрирование модуля
3. Администрирование БО
4. Администрирование сервиса
5. Администрирование методов API в сервисе
6. Импорт приложения в формате JSON
7. Экспорт приложения в формате JSON

Аутентификация

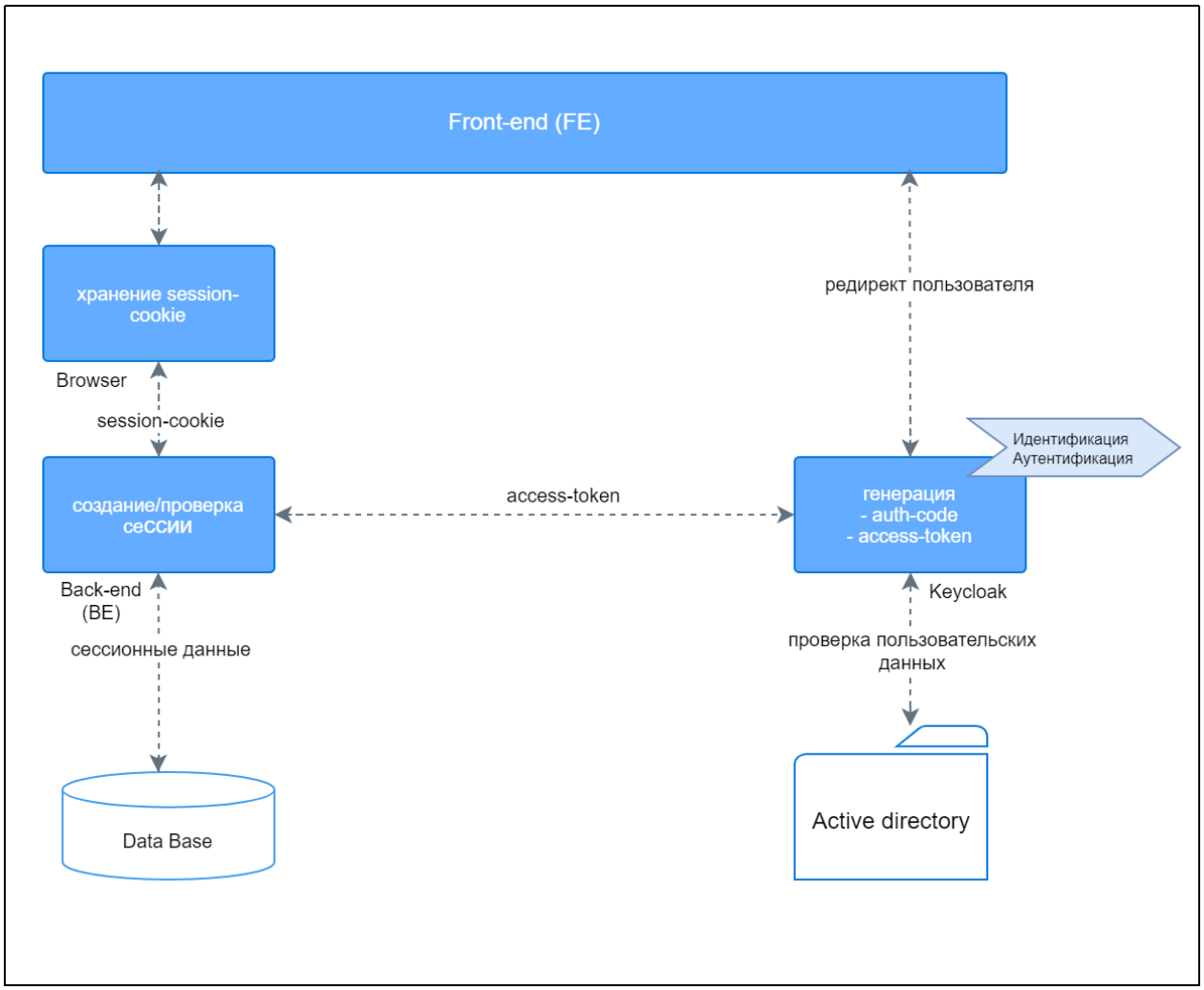
Модуль "Аутентификация" защищает от несанкционированного доступа к CRM.

Задачи модуля:

1. выполнение базового входа в CRM посредством интеграции с Keycloak

2. создание пользовательской сессии
3. выполнение logout
4. удаление пользовательской сессии
5. контроль актуальности пользовательской сессии

Архитектурная схема

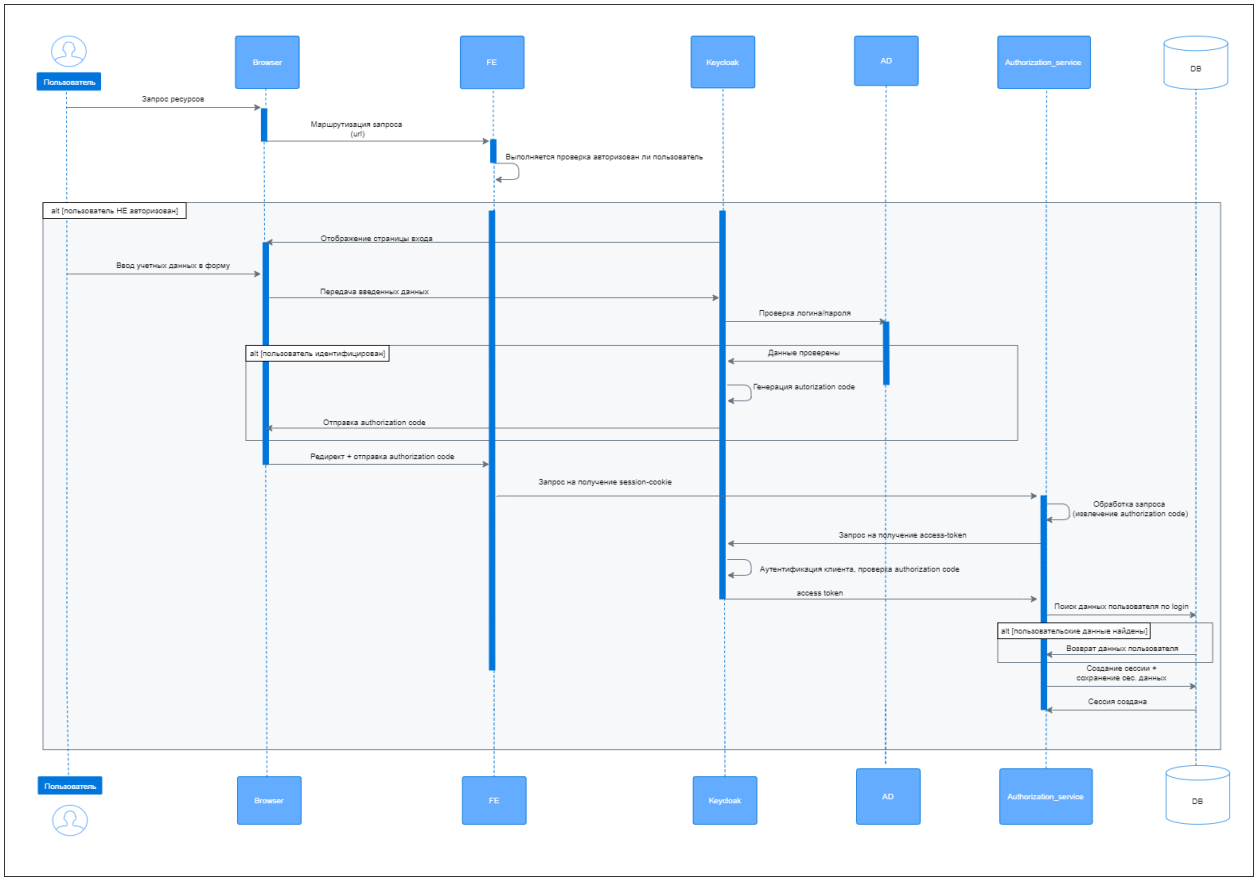


Описание к схеме

Участники	Назначение
Front-end (FE)	отвечает за: • редирект на необходимые URL back-end сервисов (через API GW), а также в Keycloak (авторизация) • отрисовку пользовательского интерфейса
Browser	отвечает за: • редирект запросов пользователя

	• сохранение и передачу session cookie
Keycloak	отвечает за: • идентификацию и аутентификацию пользователя в CRM
Active directory	отвечает за: • хранение данных пользователя (логин, пароль и др) • участвует в процессе идентификации пользователя
Back-end (BE)	отвечает за: • работу API • получение данных из Keycloak • создание сессий и сохранение сессионных/пользовательских данных в DB
Data Base	отвечает за: • хранение сессий CRM • хранение пользовательских данных

Интеграционное flow



Описание процесса аутентификации пользователя

1. Пользователь обращается к url CRM (например, https://crm-dev.t1-consulting.ru)
2. Браузер обрабатывает запрос пользователя и маршрутизирует его на FE
3. На FE выполняется проверка - авторизован пользователь или нет. Если нет то FE маршрутизирует пользователя на стартовую страницу CRM,
   которая является страницей в Keycloak (при этом url для редиректа в Keycloak хранится в ConfigMap на FE)
4. Пользователь вводит логин-пароль в форме аутентификации и нажимает на кнопку "Войти"
5. Выполняется запрос в Keycloak, где:
   a. происходит идентификация пользователя (проверяется совпадают ли пришедшие данные с данными в Active Directory)
   b. генерируется код авторизации (authorization code)
6. Keycloak выполняется редирект на FE с кодом авторизации
7. FE, имея код авторизации, выполняет запрос в authorization-service для получения session-cookie
8. BE обрабатывает запрос, извлекает авторизационный код, чтобы обменять его на access-token (токен доступа)
9. BE выполняет запрос в Keycloak для получения access-token
10. Keycloak обрабатывает запрос и ищет login пользователя по access-token
11. Keycloak возвращает в BE данные пользователя (login)
12. BE производит поиск login пользователя в DB
13. BE создает сессию CRM и возвращает session-cookie на FE
14. Получив session-cookie, FE может выполнять необходимые запросы в BE сервисы
15. Завершение процесса Аутентификации пользователя

Фабрика

Модуль "Фабрики" создаёт сервис, БД для сервиса, настраивает окружение для его работы с другими модулями.

Модуль состоит из следующих сервисов:

- factory-service - сервис, который отвечает за создание проекта сервиса в Git, сконфигурированного с помощью low-code
- db-managment-service - сервис, который отвечает за создание баз данных в СУБД Postrgesql для low-code сервисов

Задачи модуля:

1. Генерация кода по описанию сервиса из метамодели
2. Создание коннекторов и job-worker'ов в low-code сервисах
3. Создание HTTP клиента, чтобы можно было взаимодействовать с прокси методом в сервисах Поиска
4. Интеграция с Git для публикации проекта
5. Создание БД для сервиса
6. Интеграция с K8s для публикации секретов

Модули ядра (контейнеры)

1. Генерация методов для стандартного API
   a. create;
   b. update;
   c. multiUpdate;
   d. get;
   e. filter;
   f. delete;
   g. listAttachment - если активированы вложения в БО.
2. Реализация логики стандартных полей
   a. created_by;
   b. updated_by;
   c. created_at;
   d. updated_at;
   e. active.
3. Реализация механизма проверки доступов к записям согласно ОШС
4. Генерация jobWorkers для модуля управления бизнес процессами
5. Настройка producer kafka для BPM

6. Создание клиента к модулю поиска
7. Генерация заглушек для кастомных методов
8. Разделение кода сгенерированного factory-service от кода написанного разработчиком
9. Создание пользователя, пароля, и БД. Сохранение этой информации в секретах K8s
10. Настройка параметров БД и создание топиков Kafka для сбора истории изменения

Связи ресурсов

Модуль "Мониторинг связей ресурсов" предоставляет пользователям данные о ресурсах и их связях на дату.

Предпосылки создания модуля: Для разворачивания системы на разных контурах, для внесения изменений в сервисы нужно знать зависимости между ними.
Модуль позволит сократить время на установку системы, её обслуживание, и изменение

Задачи модуля:

1. Предоставлять список ресурсов и диаграмму зависимостей на текущую/выбранную дату
2. Отображать отдельный ресурс и его зависимости в виде текста/графа
3. Сравнивать ресурсы по изменениям состава зависимостей между текущей датой и выбранной датой контура
4. Сравнивать ресурсы по изменениям состава зависимостей между контурами

Функциональные возможности:

1. Просмотр списка текущих ресурсов. Типы ресурсов: микросервис, поднятый K8S; пакет фронта; бизнес-процесс из модуля bpm; топик Kafka в брокере сообщений; данные необходимые для работы текущей инсталляции платформы; неизвестный тип.
2. Просмотр графа зависимостей между ресурсами. Типы связей: rest - вызов по HTTP; grpc - вызов по GRPC; graphql - вызов по HTTP; согласно спецификации graphQL; kafka outgoing - отправка исходящих сообщений; kafka incoming - приём входящих сообщений; link - непрямая зависимость от данных сервиса для получения данных и проверки данных при изменении и создании записей, источник метамодель; dictionary - непрямая зависимость от данных бизнес справочников, источник метамодель
3. Переход по связанным ресурсам на графе зависимостей
4. Фильтрация ресурсов по названию, типу, по дате
5. Просмотр круговой диаграммы зависимостей, настройка фильтров для неё
6. Сравнение зависимостей ресурсов между текущей датой и выбранной датой контура
7. Выгрузка файла состояния стенда для сравнения состояний стендов
8. Загрузка файла состояния стенда для сравнения состояний стендов

Бизнес-справочники

Модуль "Бизнес-справочники" позволяет организовать хранение данных**,** имеющих одинаковую структуру и списочный характер, с возможностью использования значений справочников в карточке бизнес-сущностей, например, категории контактов Физических Лиц, ОПФ Компаний, статусы этапов Сделки и другое.

Задачи модуля:

1. Хранение справочников и его атрибутов
2. Настройка справочников и его атрибутов
3. Построение иерархичных справочников
4. Настройка видимости атрибутов в зависимости от организации (Бизнес-единицы)
5. Миграция справочников между стендами

Функциональные возможности

1. Ввод и редактирование справочников и атрибутов
2. Ввод и редактирование иерархичных справочников
3. Просмотр списков, возможность быстрой сортировки и фильтрации справочников и атрибутов
4. Настройка видимости тех или иных атрибутов справочника в зависимости от заданной Организации (единица ОШС)
5. Экспорт справочников системы с опцией полной или пакетной (по заданным критериям) выгрузки в формате JSON
6. Импорт справочников в формате JSON

7. Поддержка мультиязычности атрибутов

Архитектура

Модуль "Бизнес-справочники" состоит из одного сервиса dictionary-service, который обеспечивает всю работу с системными справочниками в системе. В качестве места хранения справочников, атрибутов и локализаций используется база данных - PostgreSQL. Сервис предоставляет API на GQL, gRPC и REST(считается depricated), через который происходит взаимодействие с другими сервисами. В качестве кэширующего сервера используется Redis:

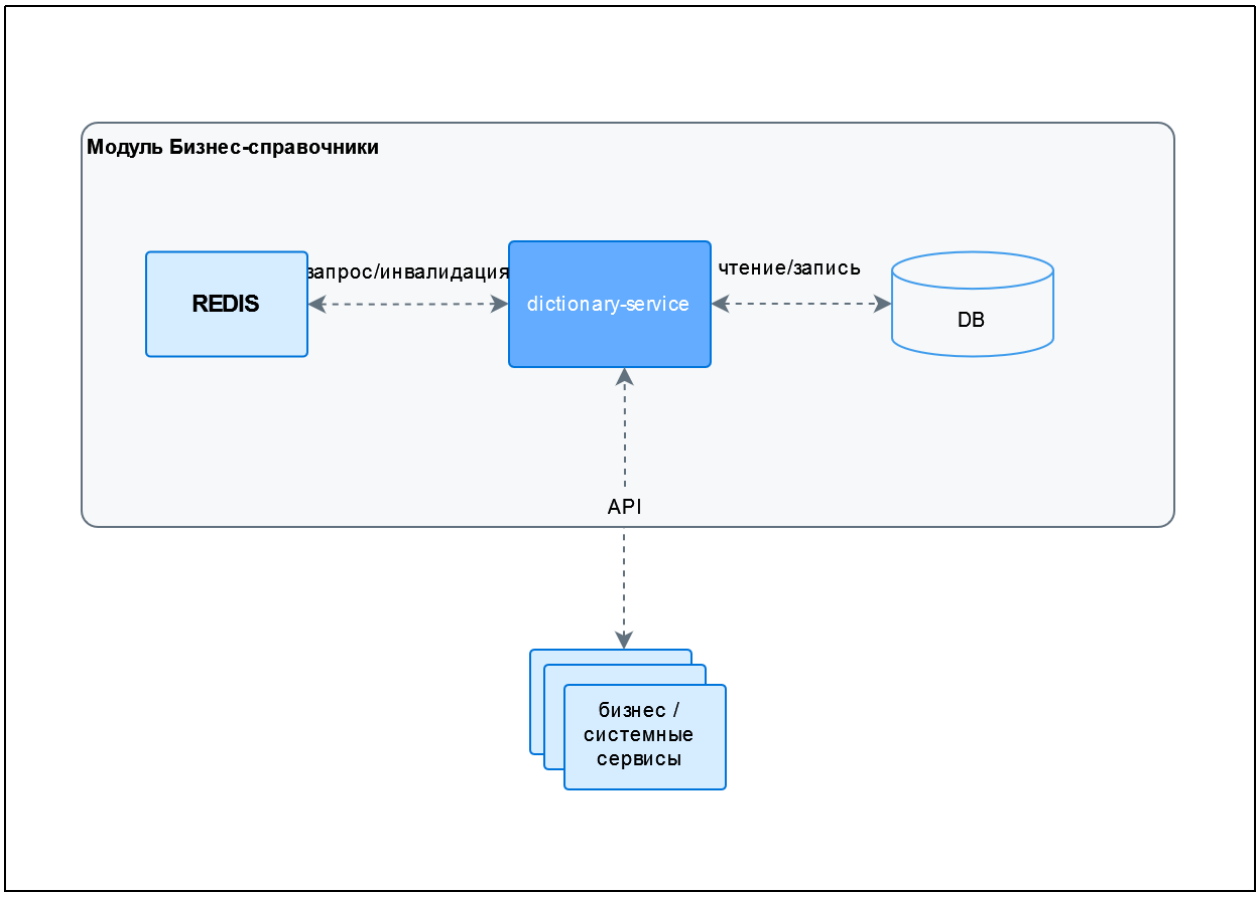


Схема "Модель зависимости атрибутов от ОШС (БЕ)"

Функционал позволяет регулировать видимость атрибута в зависимости от привязанной Организации (т.е. атрибут будет доступен для просмотра в той БЕ, для которой он был настроен). К одному атрибуту можно привязать > 1 Организации.

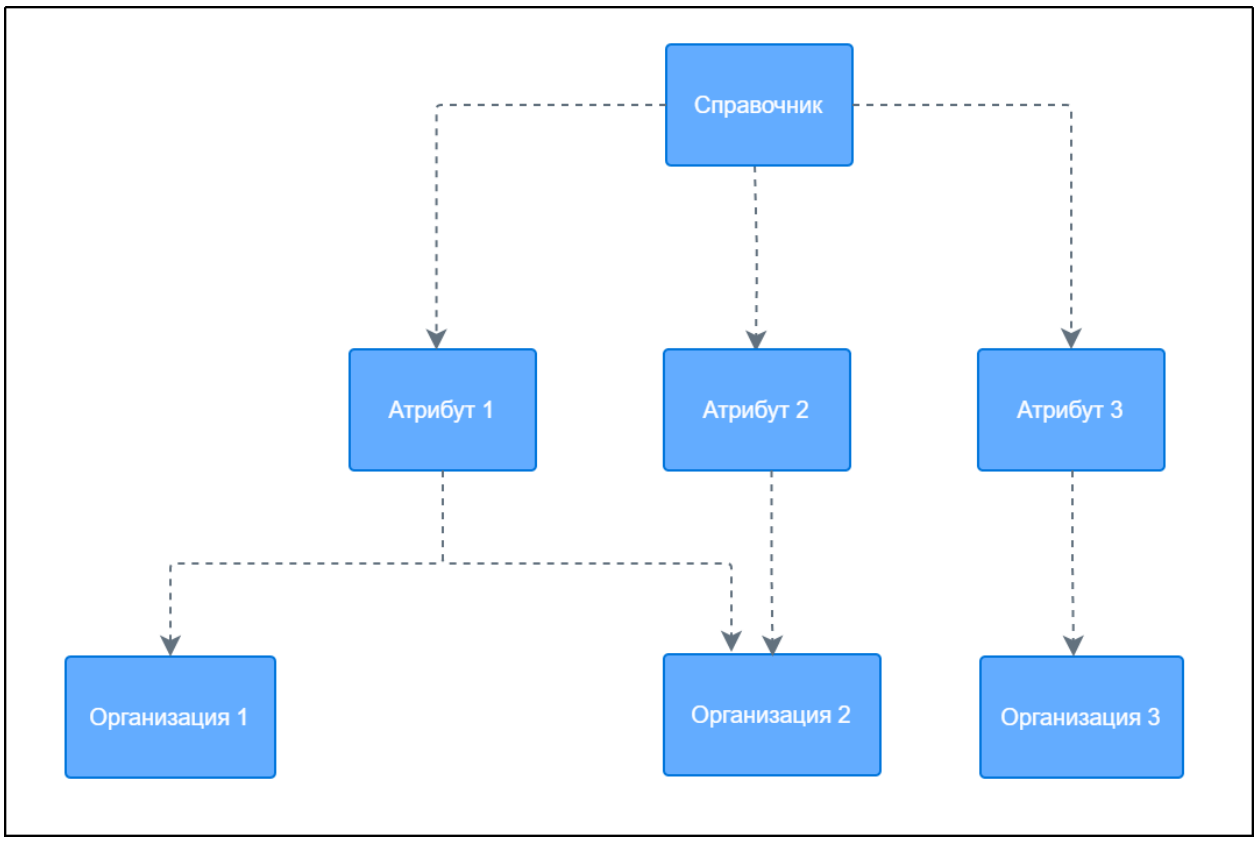


Схема "Модель иерархичных справочников"

Функционал позволяет настроить иерархичные справочники (к атрибуту настраиваются дочерние/родительские атрибуты других справочников). Уровни - без ограничений.

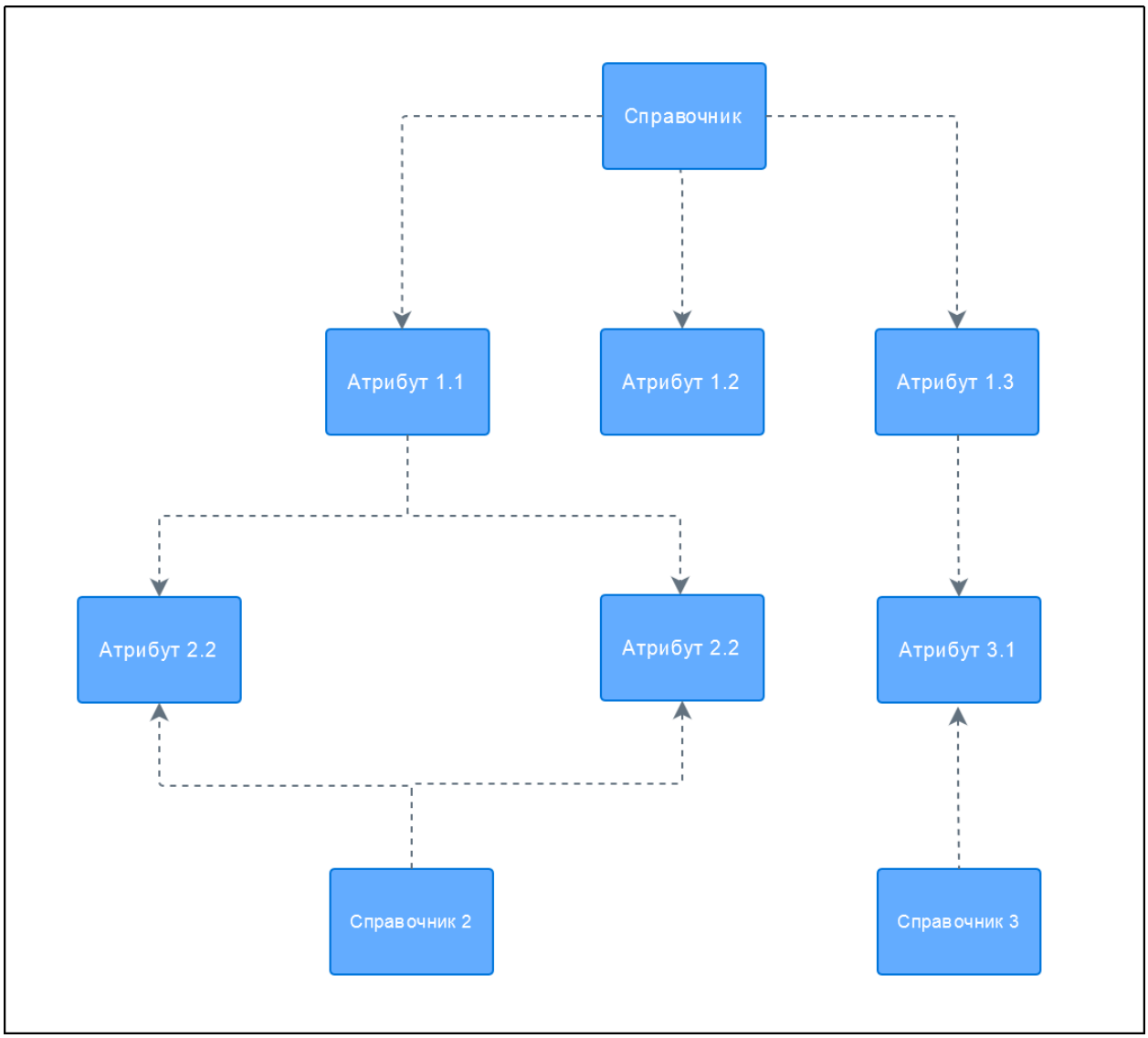


Диаграмма контейнеров

Данная диаграмма отражает, как Модуль "Бизнес-справочники" взаимодействует с пользователем и другими системами, а также разбивку Модуля "Бизнес-справочники" на взаимосвязанные контейнеры.

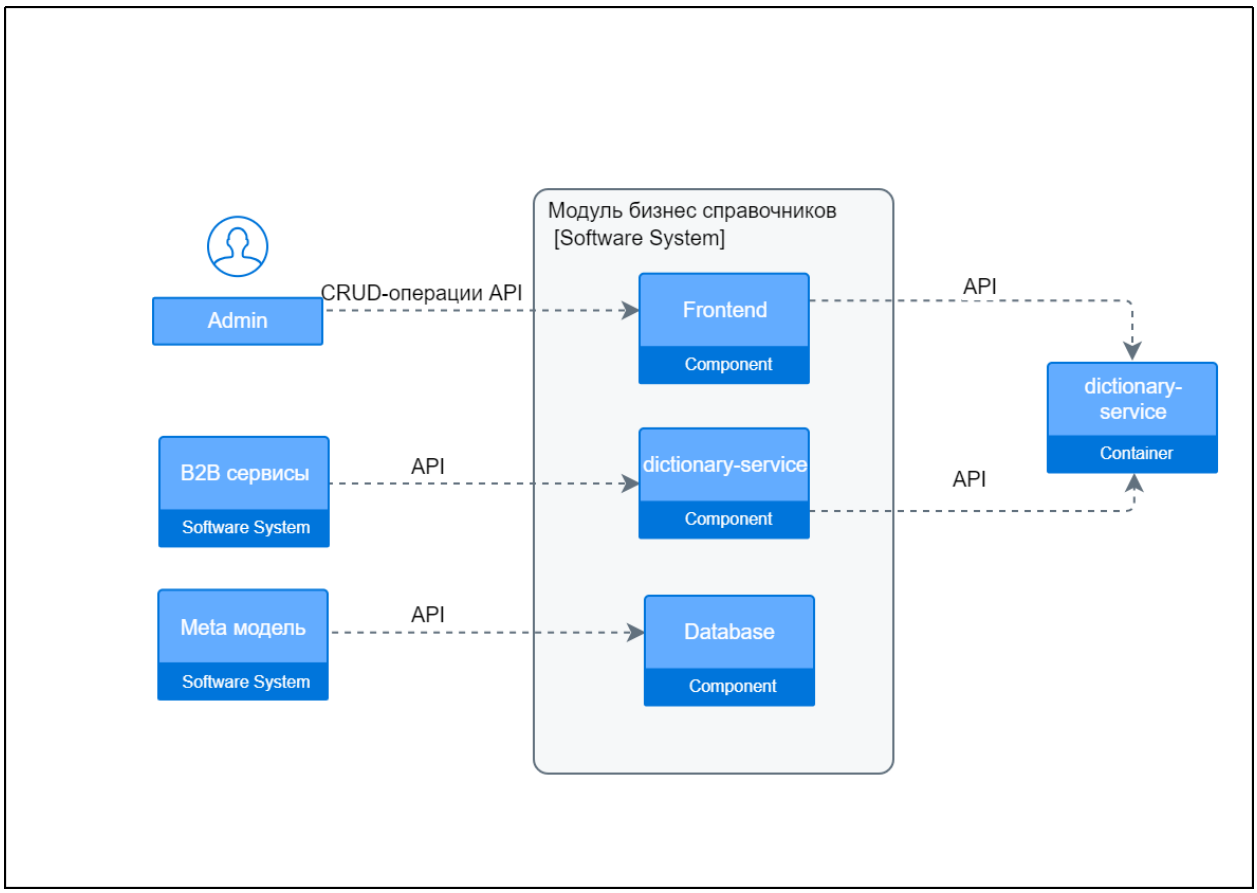
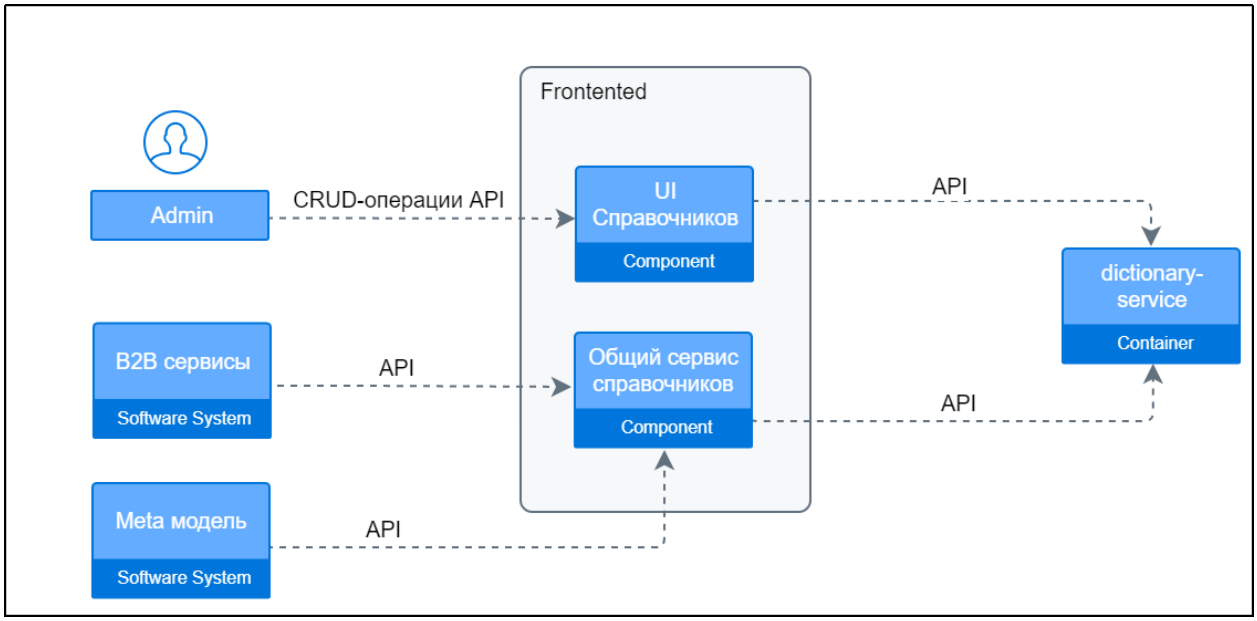


Диаграмма компонентов

Данная диаграмма показывает, как контейнер Frontend Модуль "Бизнес-справочников" разделяется на взаимосвязанные компоненты и отражает взаимодействие с пользователем и другими системами.



Описание к диаграммам

Admin	Frontend (UI Справочников)	Dictionary-service	Database	Администрирование справочников и его атрибутов через UI	Интеграция выполняется посредством вызова API (любой из методов), которое предоставляет dictionary-service. Триггером вызова API с FE служат пользовательские действия (например, сохранение записи при создании/изменении справочника и др.). После успешной обработки запроса на стороне dictionary-service выполняется запрос в БД на чтение или запись данных
B2B сервисы	Frontend (Общий сервис справочников)	Dictionary-service	Database	Получение справочников и их атрибутов для отрисовки в виджетах бизнес модулей и последующего использования в бизнес-процессах	Интеграция выполняется посредством вызова API (метод getDictionaries), которое предоставляет dictionary-service. На FE реализован общий сервис для получения справочников, которым пользуются большинство бизнес-модулей. Общий сервис получает данные непосредственно с dictionary-service, который в свою очередь выполняет чтение необходимых справочников из БД
Meta model	Frontend (Общий сервис справочников)	Dictionary-service	Database	Получение справочников для описания контракта сервиса (для типов данных dictionary) при генерации сервисов на технологии Low-code	Интеграция выполняется посредством вызова API (метод getDictionaries), которое предоставляет dictionary-service. Для получения справочников Meta model обращается в общий сервис на FE, который в свою очередь отправляет запрос в dictionary-service и выполняет чтение необходимых справочников из БД
API GW	Dictionary-service		Database	Валидирование справочных значений	Интеграция выполняется посредством grpc метода по валидации, который предоставляет dictionary-service. Для выполнения проверки справочных полей в спецификации сервиса, API GW обращается в dictionary-service, который в свою очередь выполняет запрос справочных значений из БД

Заметки

Модуль "Заметки" позволяет пользователям загружать и хранить в CRM неструктурированную дополнительную информацию в виде текста, графических изображений, ссылок.

Задачи модуля:

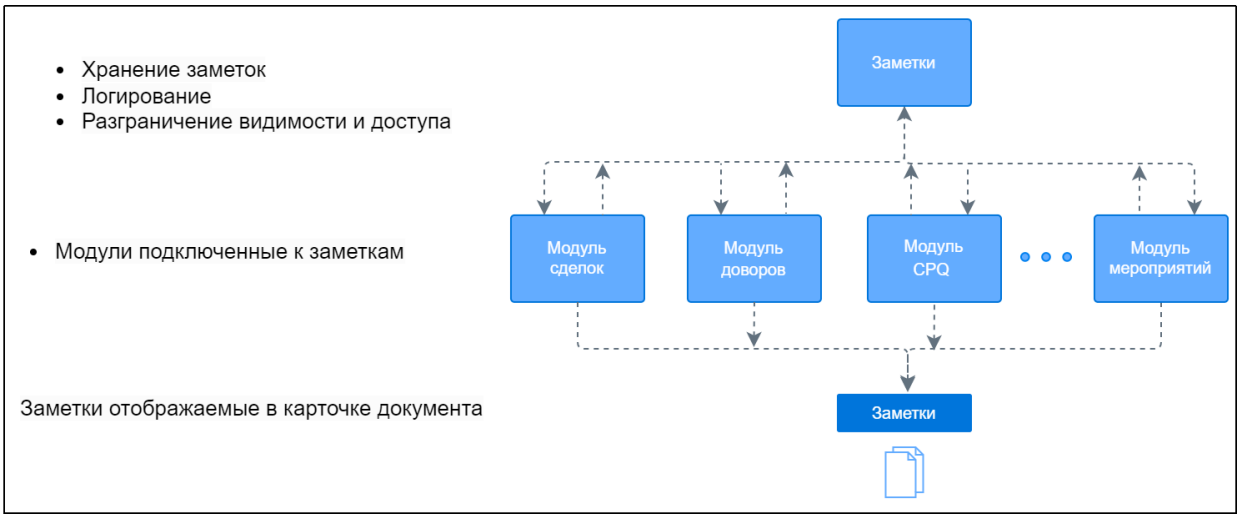
1. Загружать, хранить, редактировать заметки в системе.
2. Просматривать сохраненные заметки.
3. Осуществлять поиск по сохраненным заметкам.

Функциональные особенности:

- Модуль "Заметки" не является самостоятельным модулем, а подключается к родительской бизнес-сущности (например, сделка, компания, физическое лицо), и доступен в карточке бизнес-сущности в виде виджета.
- Доступ к виджету "Заметки" возможен, только совместно с доступом в карточку родительской бизнес-сущности. В таком случае пользователю доступны все заметки в карточке родительской бизнес-сущности.
- Для создания заметки используется онлайн редактор.

- Удаление ранее созданной записи доступно только автору записи и администратору.

Архитектура



Вложения

Модуль "Вложения" позволяет пользователям загружать и хранить в CRM документы в виде файлов большинства расширений (за исключением исполняемых).

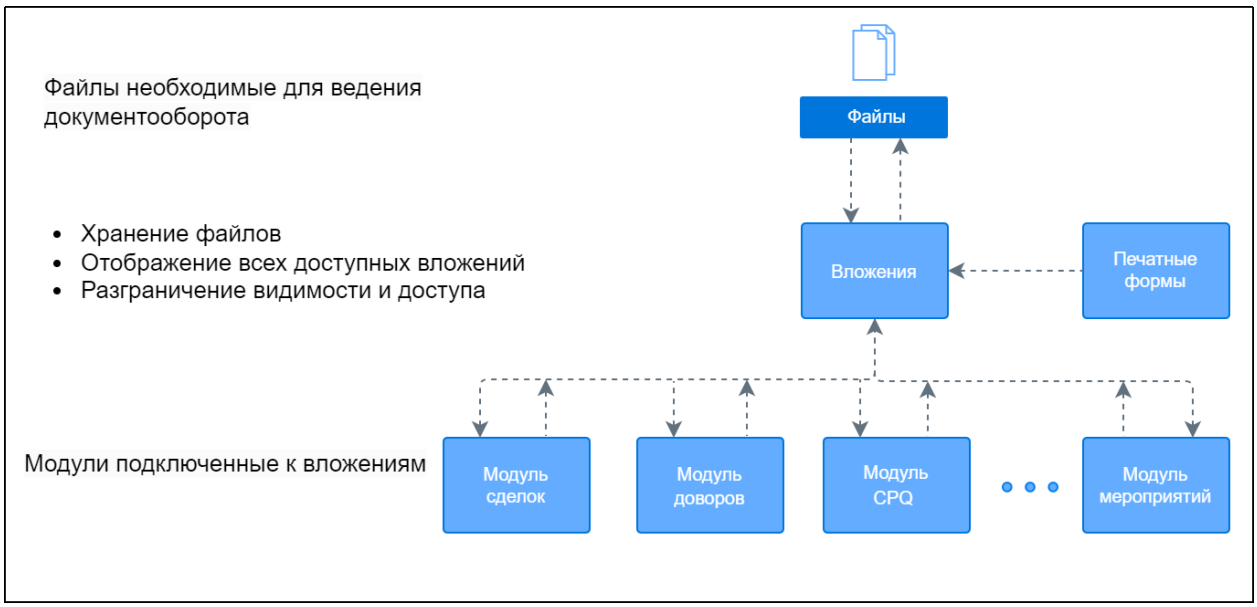
Задачи модуля:

1. Загружать, хранить, редактировать документы в системе.
2. Скачивать и просматривать сохраненные документы.
3. Осуществлять поиск по сохраненным вложениям.

Функциональные особенности:

- Модуль "Вложения" не является самостоятельным модулем, а подключается к родительской бизнес-сущности (например, сделка, компания, физическое лицо), и доступен в карточке бизнес-сущности в виде виджета.
- Доступ к виджету "вложения" возможен, только совместно с доступом в карточку родительской бизнес-сущности. В таком случае пользователю доступны все вложения в карточке родительской бизнес-сущности.
- Также реализован общий список вложений. Есть возможность назначить доступ к этому списку согласно ролевой модели. Общий список вложений возможно использовать для поиска необходимого вложения как базу знаний.
- Модуль "Вложения" является местом хранения сгенерированных по шаблонам документов Полученные таким образом документы являются полноценными и над ними доступны все действия, производимые над стандартными вложениями.
- Удаление ранее созданной записи доступно только автору записи и администратору.

Архитектура



Сквозной поиск

Определение:

Модуль "Поиска" состоит из 2-х частей: сервиса поиска и сервиса индексации. Сервис поиска формирует поисковые запросы, дополняя их необходимыми для фильтрации по уровню доступа данными, а сервис индексации, в свою очередь, собирает и публикует в поисковые индексы изменения, произошедшие в бизнес-объекте. Модуль поиска позволяет проводить сквозной поиск через все внесенные в поисковую систему документы, независимо от их принадлежности к тому или иному бизнес-объекту. Простой поиск позволяет быстро найти бизнес-объект и перейти на его карточку. Расширенный поиск позволяет представить в табличной форме информацию по конкретному типу бизнес-объекта, применить сложные фильтры на его поля, подвергнуть данные сортировке, при необходимости сделать экспорт.

Задачи модуля:

Сервис индексации

Функциональные:

- Добавление, обновление и удаление данных по бизнес-объектам на основе событий в этих объектах;
- Обогащение корневого документа данными из связанных бизнес-объектов;
- Обновление и удаление из корневого документа данных о связанных бизнес-объектах;
- Обогащение документа пользовательскими данными;
- Переиндексация данных по запросу;
- Интеграция с OpenSearch для публикации данных;
- Интеграция с сервисом предоставления данных аудита для извлечения данных по бизнес-объектам.

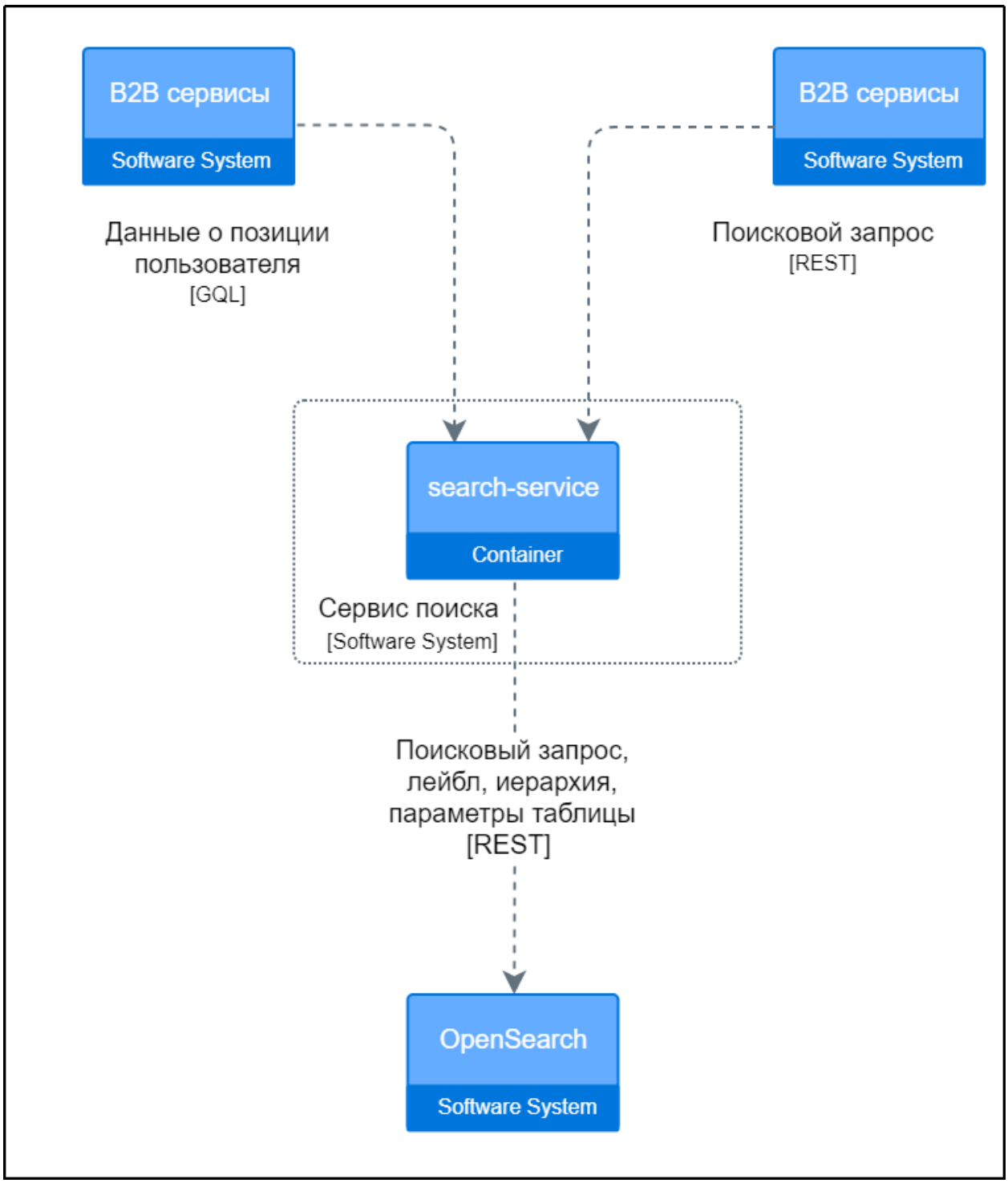
Сервис поиска

- Функциональные:
  - Позволяет осуществить поиск по всем атрибутам сущности;

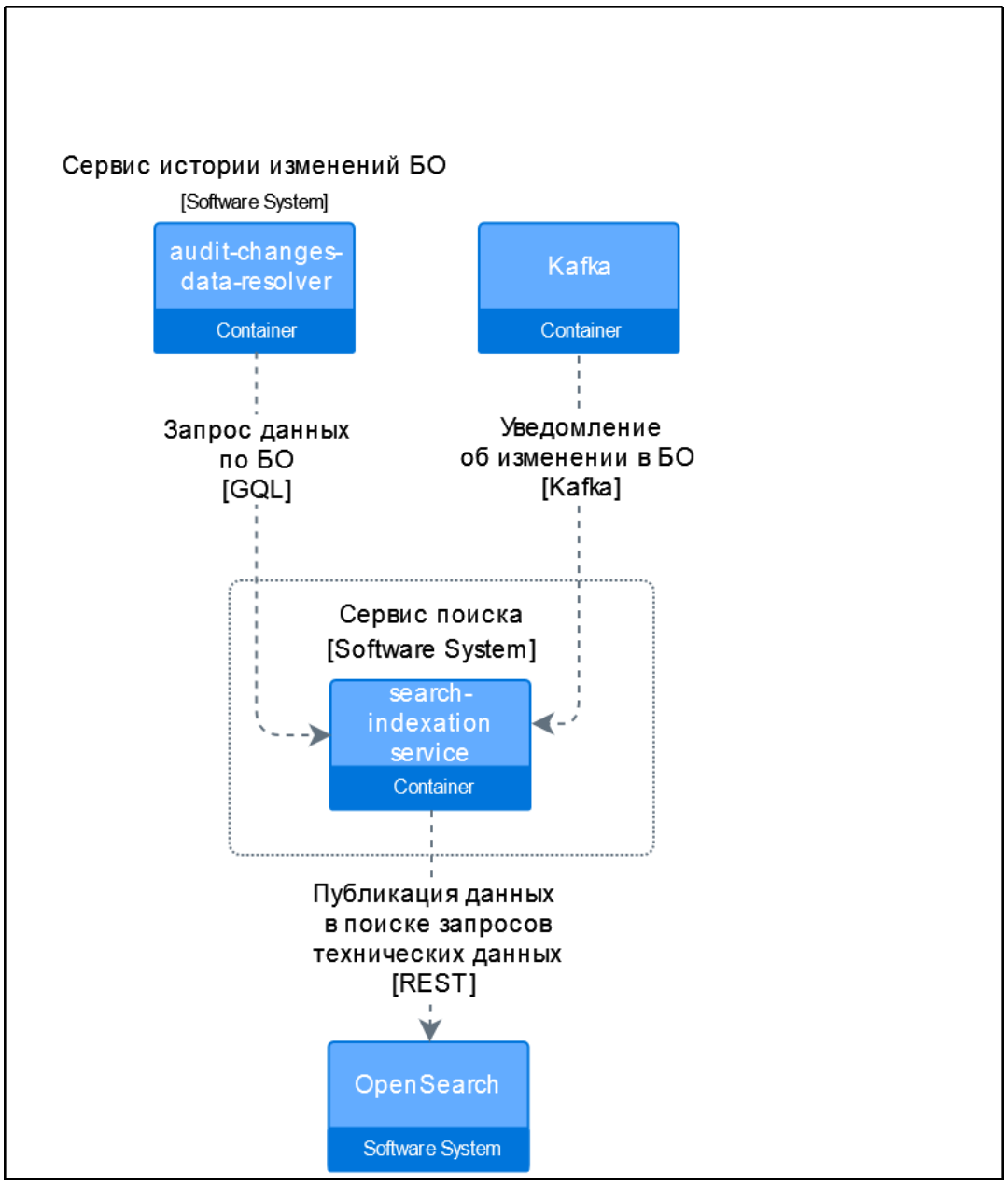
        - Позволяет осуществить поиск только по разрешённым пользователю сущностям согласно правам доступа;
        - Позволяет сохранить поисковые запросы;
        - Состоит из двух блоков:
            - Быстрый поиск:
                - Поиск для навигации по системе;
            - Расширенный поиск:
                - Поиск для навигации по системе;
                - Отображение сущностей, объединённых каким-либо общим признаком или набором общих признаков;
                - Экспорт найденных сущностей;
- Нефункциональные:
    - Предоставляет наиболее релевантные результаты поиска для пользователя;
    - Удобство составления поисковых запросов

Интеграции модуля

Сервис поиска интегрирован с сервисом ОШС для получения информации о БЕ и позиции пользователя, выполняющего запрос, а также имеет REST-интеграцию с поисковым движком OpenSearch для осуществления полнотекстового поиска.



- С Kafka для получения уведомлений о событиях в БО;
- С сервисом предоставления данных аудита для получения разыменованной информации по текущему БО и связанным с ним;
- С OpenSearch для получения информации по связанным документам, пользовательским полям и техническим таблицам;
- С OpenSearch для публикации данных по БО в поисковые индексы.

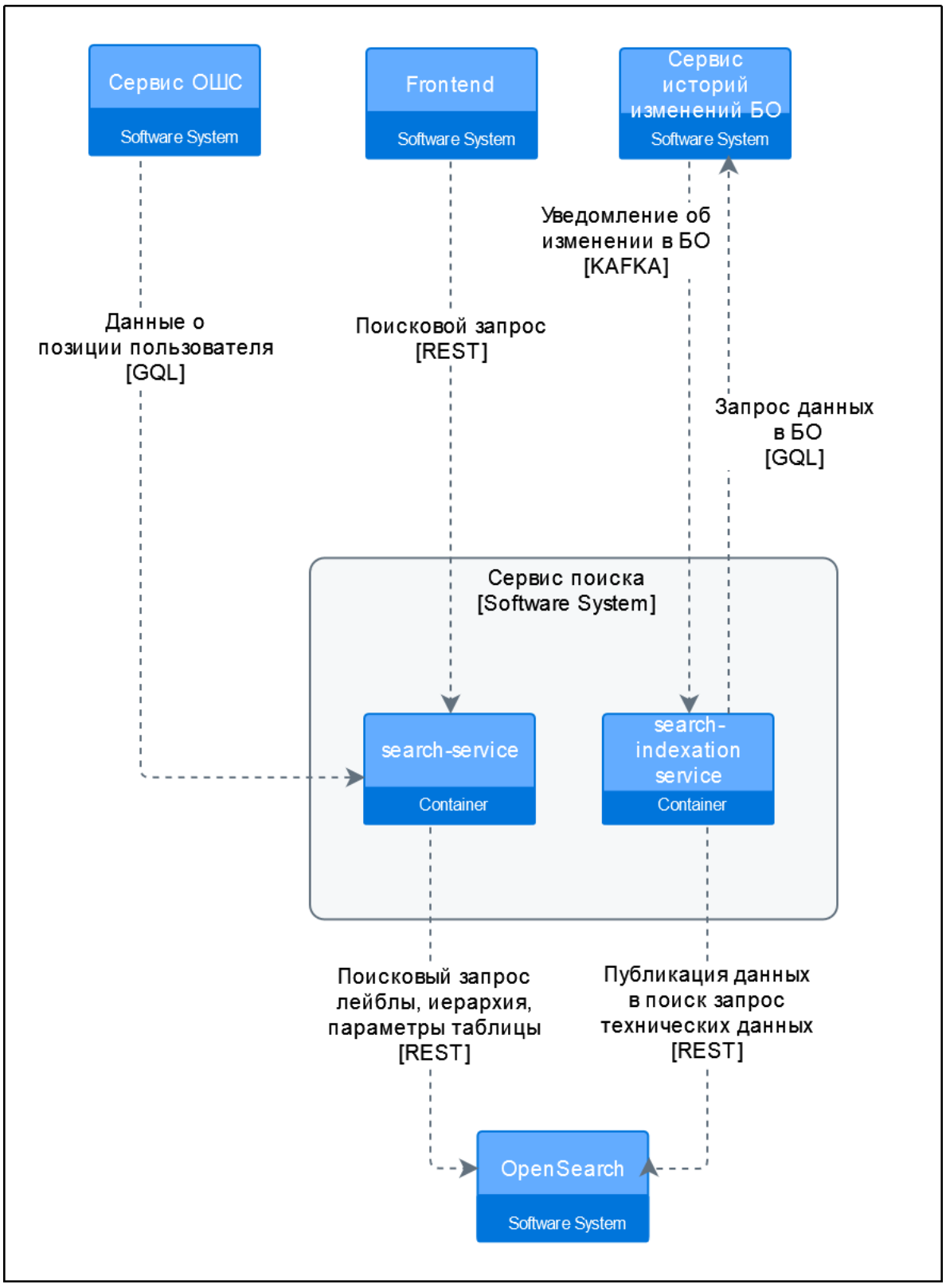


Архитектура

Модуль поиск состоит из 2-х раздельных сервисов, один из которых обеспечивает наполнение поиска данными, а второй непосредственно выполняет в них поиск. Сервис индексации имеет один GraphQL, через который можно принудительно заставить переиндексировать данные по определенному типу БО.

Принцип работы:

Для того чтобы включить индексацию, необходимо подключить БО к сервису "История изменений БО" (далее - Аудит). После подключения Аудит начинает уведомлять сервис индексации об изменениях в наблюдаемых БО, вся необходимая информация запрашивается сервисом индексации из сервиса предоставления данных аудита. Передаваемая информация уже денормализована, т.к. сервис предоставления данных аудита имеет все необходимые интеграции. Согласно данным о связанных БО и пользовательских полях сервис индексации обогащает документы необходимыми данными. Если БО удаляют\обновляют, то сервис индексации удаляет\обновляет соответствующий документ и в поисковом индексе, а также удаляет\обновляет информацию о нем в связанных документах.



Аудит изменение бизнес-объекта

Определение:

Модуль предназначен для осуществления унифицированного подхода к сбору изменений, происходящих в БО. Модуль обеспечивает унифицированный подход для решения хранения истории бизнес-объектов и

доступа к данным. Собранная история изменений может как логически, так и физически быть разнесена с базой данных источника.

Задачи модуля:

Сервис audit-changes предназначен для фиксирования изменений в подключенных БО. Захват изменений реализуется через Debezium с последующей публикацией в соответствующий топик Kafka. На топики Kafka подписан сервис audit-changes который приводит информацию об изменениях к необходимому формату и складывает в отдельную БД.

Функциональные требования:

- Отслеживание изменений в БО;
- Публикация данных об изменении в Kafka;
- Снятие снапшота данных при первичном подключении;
- Отказоустойчивость. Возможность получить данные после восстановления работы Kafka или сервиса audit-changes.

Сервис реализует доступ к данным модуля "История изменений БО", посредствам API GraphQL. Возвращает данные в разыменованном и денормализованном виде, подставив значения справочников, имена сущностей и ФИО пользователей вместо идентификаторов. Проводит фильтрацию данных согласно заданным параметрам.

Функциональные требования:

- Информация о составе БО извлекается из Метамодели;
- Разыменование сущностей (например, вместо company_id возвращать short_name по данной компании);
- Разыменование справочных значений (например, вместо "810" для справочника transaction currency возвращать "РУБ");
- Разыменование ФИО пользователя по его id или позиции;
- Фильтрация данных по дате;
- Фильтрация данных по полю;
- Сортировка данных.

Сервис debezium-connector-manager предназначен для автоматизации процесса подключения бизнес-объекта к "Истории изменений БО", и позволяет осуществлять создание, переконфигурацию и удаление коннекторов Debezium для захвата изменений в подключенных БО. В свою очередь, данные об изменениях получает подписанный на соответствующие топики Kafka сервис audit-changes, который приводит полученную информацию к необходимому формату и складывает в отдельную БД.

Функциональные требования:

- Автоматизации процесса подключения бизнес-объекта к аудиту;
- Создание коннектора Debezium при подключении аудита;
- Автоматическое обновления конфигурации коннектора Debezium;
- Удаление коннектора Debezium при отключении от аудита;
- Отслеживание статуса подключения коннектора Debezium.

Интеграции модуля:

БД CRM	Debezium	Kafka	Получение и публикация изменений БО	Интеграция на уровне БД. Триггером публикации данных в Kafka является изменение данных в наблюдаемых таблицах либо первичное подключение таблицы (снятие снапшота).
Kafka	audit-changes-service	БД Аудита	Преобразование и запись информации об изменении БО в БД Аудита	Интеграция на уровне топика Kafka. Триггером является появление новой информации в топике отслеживаемой таблицы.

uml

```
@startuml
!include <C4/C4_Container>

ContainerDb(crm_db, "БД CRM")
ContainerDb(audit_db, "БД Аудита")
System(debezium, "Debezium")
System(Kafka, "Kafka")
Container(audit_changes_service, "audit-changes-service")

Rel(crm_db, debezium,  "Снятие изменений")
Rel(debezium, Kafka,  "Публикация изменений")

Rel(Kafka, audit_changes_service, "Чтение и преобразование изменений")
Rel(audit_changes_service, audit_db, "Запись данных")

@enduml
```

audit-resolver-service	Meta-model	Получение лейблов, информации о справочниках, линках и именах БО	Интеграция выполняется посредством вызова API мета-модели (метод getMetamodelObjectsWithProperties). Вызов осуществляется по GraphQL протоколу.
audit-resolver-service	dictionary-service 2	Получение справочных значений	Интеграция выполняется посредством вызова API сервиса справочников. Вызов осуществляется по GQL протоколу.
audit-resolver-service	organizational-structure-service	ФИО пользователя по позиции и ФИО по идентификатору	Интеграция выполняется посредством вызова API сервиса ОШС (метод/graphql/users). Интеграция выполнена для получения ФИО пользователя по позиции. Вызов осуществляется по GraphQL протоколу.
FE	audit-resolver-service	Получение данных по изменениям БО	Интеграция выполняется посредством вызова API audit-resolver-service (метод POST getAuditChangesData). FE запрашивает изменения по БО и показывает результат пользователю.

uml

```
@startuml
!include <C4/C4_Container>

ContainerDb(audit_db, "БД Аудита")
Container(audit_resolver_service, "audit-resolver-service")
```

```
Container(meta_model, "Мета модель")
Container(dictionary_service, "Сервис справочников")
Container(organizational_sructure_service, "Сервис ОШС")
System(FE, "FE")

Rel(audit_resolver_service, audit_db, "", "")

Rel(audit_resolver_service, dictionary_service, "Получение справочных значений", "")
Rel(audit_resolver_service, meta_model,"Лейблы, имена\nсущностей, ссылки", "")
Rel(audit_resolver_service, organizational_sructure_service,"ФИО пользователей\n по позиции", "")

Rel(FE, audit_resolver_service,"Запрос информации\nпо изменениям", "")
@enduml
```

debezium-connector-manager-service	metamodel-service	Получение изменений по аудированию сущностей	Интеграция происходит через вызов debezium-connector-manager-service метода getMetaModelDeploymentsData
debezium-connector-manager-service	Kafka	• Получение событий деплоя • Направление статуса поднятия коннектора	• debezium-connector-manager-service подписан на топик Kafka с событием о деплое сервиса • debezium-connector-manager-service отправляет в топик Kafka информацию о статусе подключения коннектора
debezium-connector-manager-service	Debezium	Автоматизация подключения	Интеграция происходит посредством внесения debezium-connector-manager-service конфигурации коннекторам Debezium
debezium-connector-manager-service	db-management-service	• Получение прав на аудируемые таблицы • Применение команды Replica Identity	Интеграция происходит опционально (в зависимости от наличия на стенде db-management-service, который использовался для упрощения процесса подключения и тестирования). • Интеграция происходит через вызов debezium-connector-manager-service метода addAuditGrants • Интеграция происходит через вызов debezium-connector-manager-service метода setReplicaIdentityMode

uml

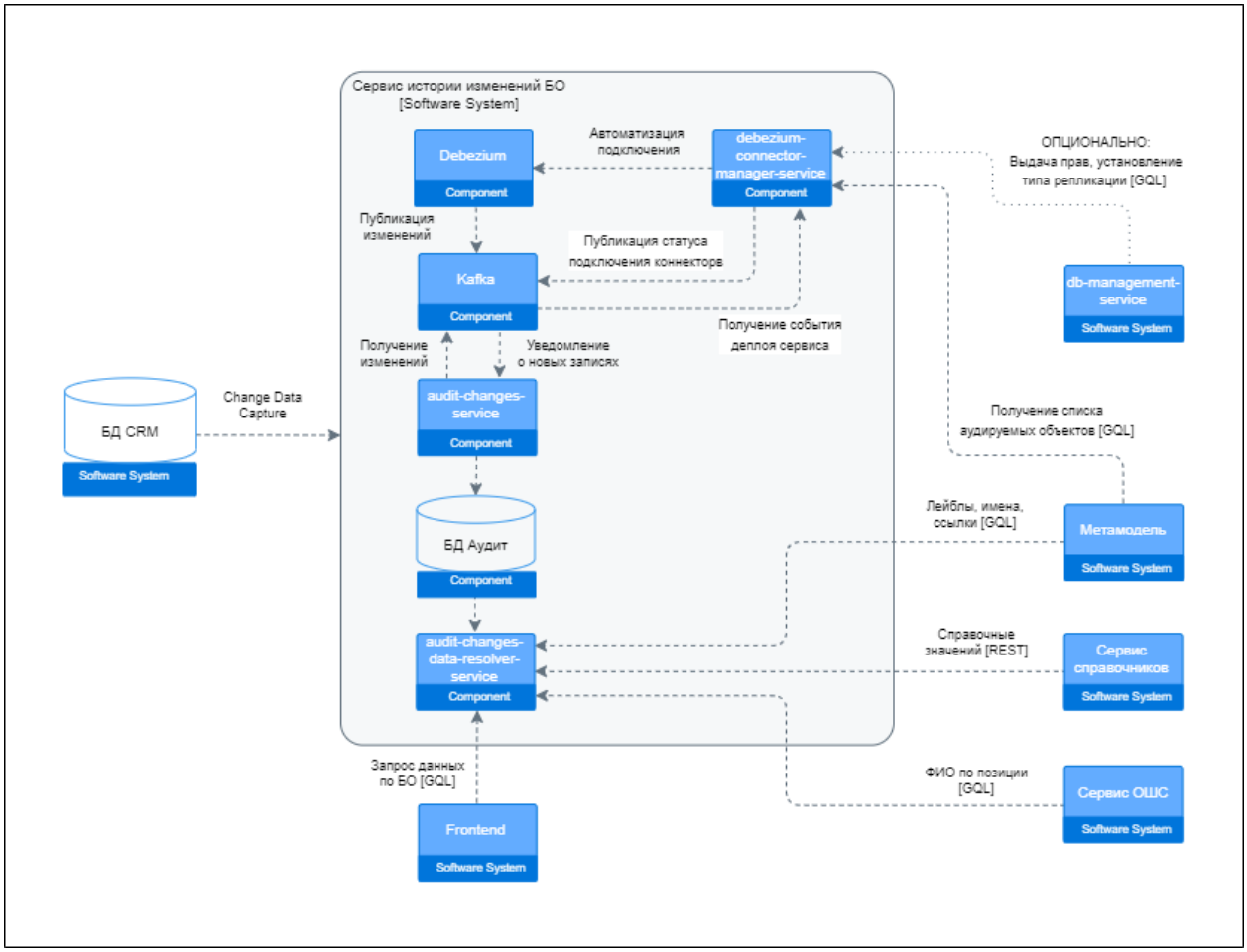
```
@startuml
!include <C4/C4>
!include <C4/C4_Context>
!include <C4/C4_Container>

ContainerDb(dbCRM, "db CRM")
System(metamodelservice, "metamodel-service")
System(dbmanagementservice, "db-management-service")

System_Boundary("auditmodule_boundary", "audit-module") {
  ContainerDb(auditmodule.dbAudit, "db Audit")
  Container(auditmodule.debeziumconnectormanagerservice, "debezium-connector-manager-service")
  Container(auditmodule.Debezium, "Debezium")
  Container(auditmodule.Kafka, "Kafka")
```

```
  Container(auditmodule.auditchangesservice, "audit-changes-service")
}

Rel(dbCRM, auditmodule.Debezium, "Change Data Capture")
Rel(auditmodule.Debezium, auditmodule.Kafka, "Публикация изменений")
Rel(auditmodule.Kafka, auditmodule.auditchangesservice, "Получение изменений")
Rel(auditmodule.auditchangesservice, auditmodule.dbAudit, "Запись изменений")
Rel(metamodelservice, auditmodule.debeziumconnectormanagerservice, "Передача изменений по аудированию сущностей")
Rel(dbmanagementservice, auditmodule.debeziumconnectormanagerservice, "Выдача прав")
Rel(auditmodule.debeziumconnectormanagerservice, auditmodule.Debezium, "Управление коннектором")
Rel(auditmodule.Kafka, auditmodule.debeziumconnectormanagerservice, "Получение события деплоя")
Rel(auditmodule.debeziumconnectormanagerservice, auditmodule.Kafka, "Публикация статуса поднятия коннектора")

@enduml
```

Архитектура:

Модуль "История изменений БО" состоит из 3-х сервисов.

- Сервис audit-changes-service работает в автоматическом режиме согласно конфигурации коннектора Debezium. Конфиг Debezium предается через POST запрос в Kafka. Не имеет энд поинтов.
- Сервис предоставления данных audit-resolver-service позволяет эффективно запрашивать, фильтровать и сортировать собранные данные. Имеет 2 GraphQL энд поинта, один для предоставления данных по изменениям для FE CRM, второй для предоставления source-данных для search-indexation-service.
- Сервис debezium-connector-manager-service предназначен для автоматизации процесса подключения БО к аудиту, путем создания, изменения или удаления коннекторов Debezium. Не имеет энд поинтов.



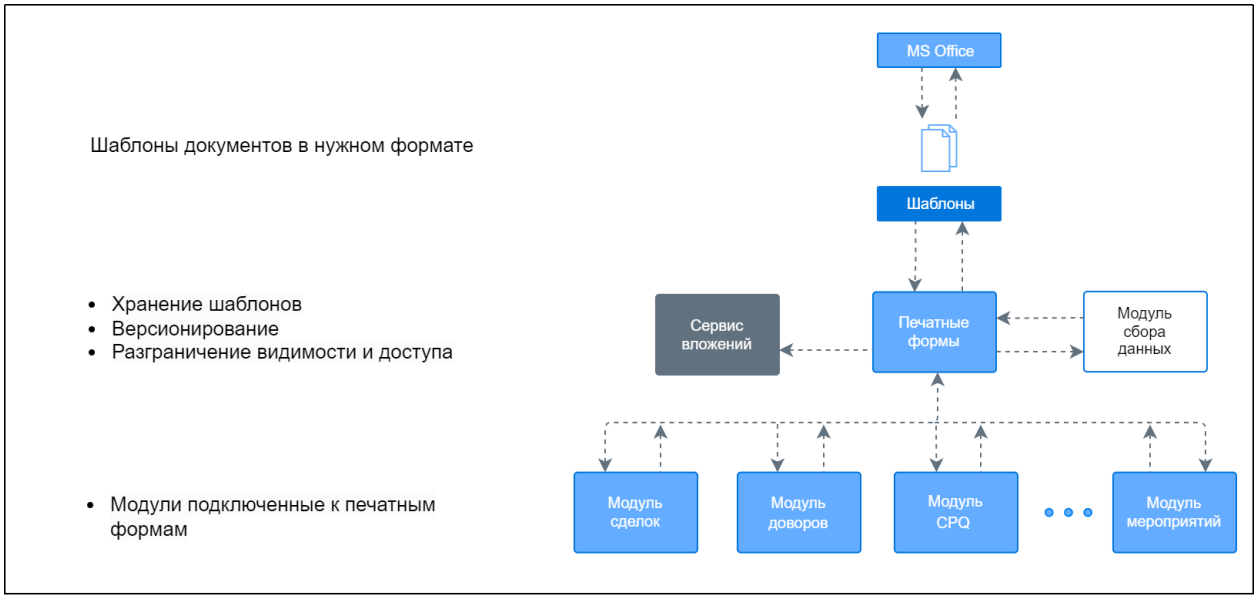
Генерация документов по шаблону

Генератор предназначен для создания шаблонов форм отчетов на основе шаблонов. Сервис генератора документов подключается к различным сервисам и автоматизирует процессы составления документов, таких как контракты, счета, инвойсы, отчеты и так далее. Для генерации используются шаблоны подготовленные в MS Office

Шаблон документа – это документ (файл указанного формата), который состоит из двух частей: статической и динамической. Статическая часть не изменяется, а в динамической части вместо данных содержатся названия переменных, значения которых формируются и записываются из атрибутов электронного документа или контекстных переменных процесса.

Реализовано автоматическое склонение и относительные даты в генерируемом документе.

Сохранение сгенерированных документов происходит в сервисе вложений



Аудит действия пользователей

Определение:

Аудит действий пользователей — это комплекс мероприятий, который отслеживает, кто из пользователей выполнил те или иные действия. Сведения об успешных и неуспешных действиях пользователей в информационной системе фиксируются в журнале событий ИБ с указанием даты и времени их совершения, а также логином пользователя. Внедрение системы мониторинга действий пользователей в компании позволяет фиксировать и оперативно реагировать на активность сотрудников.

Модуль аудита действий пользователя состоит из набора логеров действий пользователя, реализованных в местах, через которые проходят основные информационные потоки.

Лог Api Gateway - API GateWay реализует единую точку входа к API используемых системных и бизнес-сервисов CRM. В логе GateWay фиксируются все действия пользователя как на просмотр, так и на изменение системных либо бизнес-объектов, а также сами отраженные либо записанные данные.

Лог KeyCloak - ПО KeyCloak позволяет настроить различные способы аутентификации. Его лог содержит данные об успешной\неуспешной аутентификации.

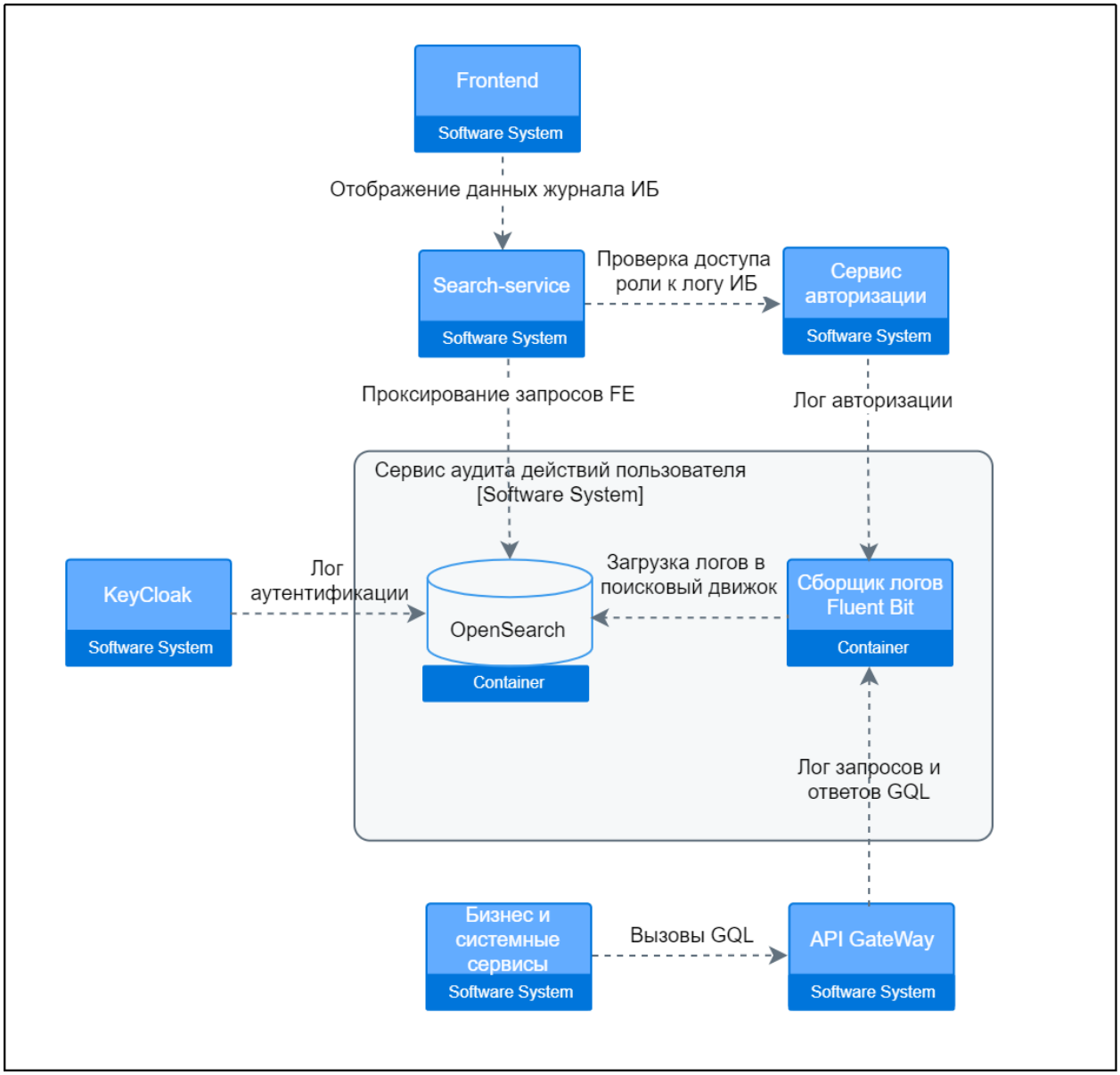
Лог Сервиса авторизации - сервис авторизации выдает пользователю, прошедшему аутентификацию, набор прав определенных в его профайле, а также ведет управление сессией пользователя. В логе сервиса авторизации отражается создание и завершение сессии пользователя.

Архитектура:

Модуль Аудит действий пользователя собирает данные:

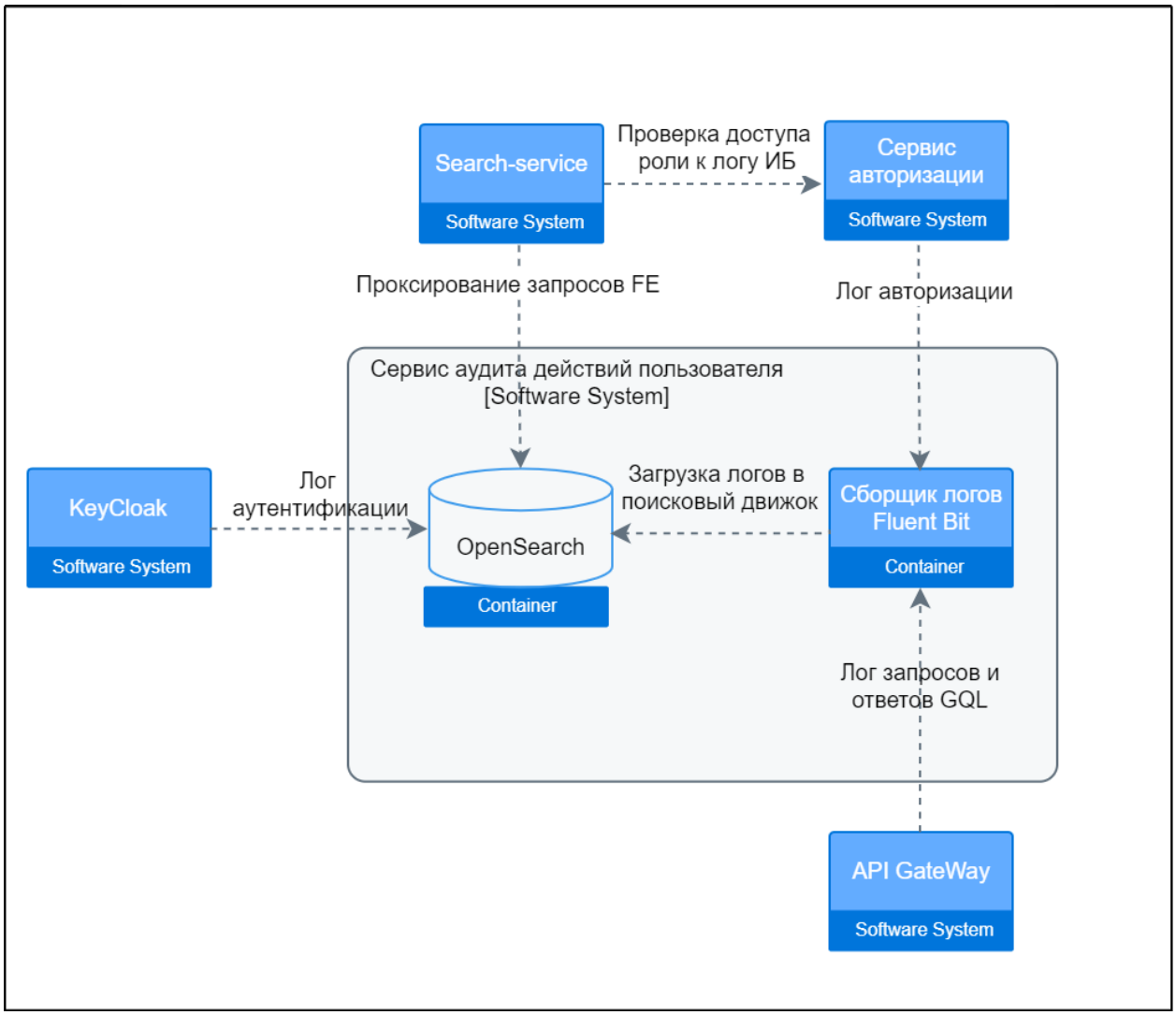
- События аутентификации - успешный или неуспешный логин;
- События авторизации;
- Действия пользователя в системе.

Модуль состоит из внутрисервисных логеров, сборщика логов Fluent Bit и системы хранения и поиска информации OpenSearch. Для доступа и разграничения доступа используются проксирующие вызовы search-service, которые перед выполнением проходят проверку прав в сервисе авторизации.



Интеграции модуля:

Лог Сервиса авторизации и Лог Api Gateway интегрированы с Fluent-bit, который осуществляет сбор, обработку и загрузку (пересылку) в OpenSearch лог-сообщений.



Сохранение пользовательских настроек UI

Определение:

Сервис предназначен для хранения в базе данных настроек пользователей, что позволяет избежать их потери при сбое или смене рабочего места. При работе Пользователя с системой сервис производит

автоматическое сохранение, изменение или удаление его UI настроек в БД, а также дублирует все указанные действия для локального хранилища браузера, что позволяет получать быстрый доступ к имеющимся настройкам без необходимости обращения к BE. После выхода Пользователя из системы все пользовательские настройки удаляются из локального хранилища браузера, но при этом сохраняются в базе данных. В начале новой сессии Пользователя все настройки имеющиеся в базе данных подтягиваются в локальное хранилище браузера, чем и достигается синхронизация пользовательских настроек на BE и FE.

Задачи сервиса:

- Хранение пользовательских настроек UI в базе данных;

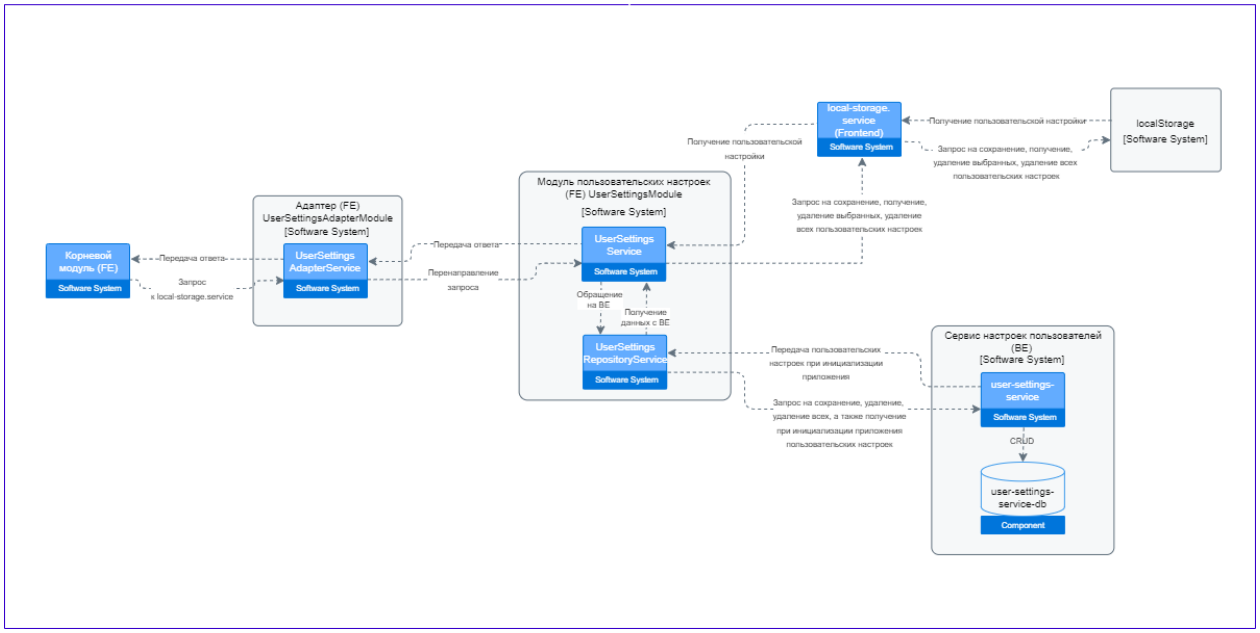
Функциональные особенности:

- Сервис автоматически отслеживает изменения пользовательских настроек и синхронизирует их с базой данных и локальным хранилищем браузера;
- Сервис использует локальное хранилище браузера для быстрого доступа к настройкам без обращения к серверу;
- При входе пользователя в систему, настройки из базы данных загружаются в локальное хранилище, при logout Пользователя локальное хранилище браузера очищается.

Архитектура:

Сервис настроек пользователей включает в себя:

- FE-модуль (UserSettingsModule), который отслеживает изменения в пользовательских настройках и синхронизирует их с локальным хранилищем браузера и backend-частью сервиса;
- FE-адаптер (UserSettingsAdapterModule), необходимый для внедрения сервиса без изменения всей прежней логики взаимодействия системы с локальным хранилищем браузера;
- BE-сервис (user-settings-service), главным образом отвечающий за сохранение пользовательских настроек в базу данных.



Правила информационной безопасности

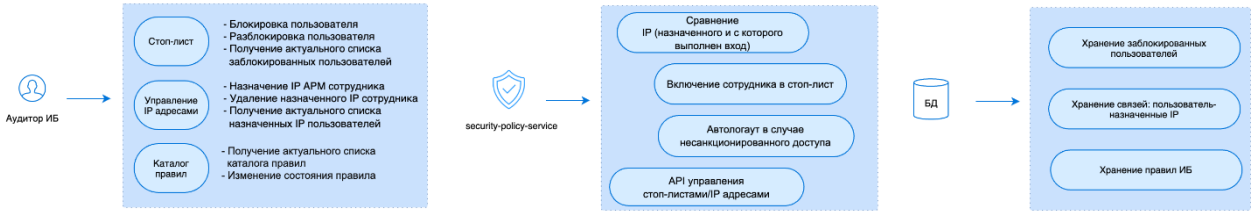
Назначение Модуля ИБ заключается в обеспечении безопасности и контроля доступа к системе. Реализация функций по предотвращению несанкционированного доступа и обеспечения соблюдение политик

безопасности происходит за счет ряда интеграций, которые направленны на автоматизацию информационной безопасности.

Задачи модуля:

1. **Управление стоп-листом:** включает пользователей, которые были идентифицированы, как источники угроз безопасности
2. **Контроль доступа:** на этапе авторизация или выполнении любого действия в системе (сопровождаемый вызовом API сервиса) выполняются проверки безопасности по наличию пользователя в стоп-листе, сверки назначенного и зарегистрированного IP и др.
3. **Назначение IP-адреса:** АРМ сотрудника имеет выделенный IP с которого допустима работа с ресурсами организации (CRM)

Для гибкого управления, предусматривается пользовательский интерфейс - панель Аудитора ИБ, где он может назначить IP АРМ сотрудника, внести в стоп-лист или разблокировать необходимого пользователя, а также изменить состояние правила ИБ



Используемые правила ИБ:

check_user_stoplist

Запрет на авторизацию в CRM, если пользователь находится в стоп-листе

Кейс:
Ранее пользователь был добавлен в стоп-лист и сейчас выполняет попытку пройти аутентификацию в CRM

different_arm_ip

Блокировка пользователя с неразрешенным IP

Кейс: Пользователь выполнил вход в CRM с IP-адреса, отличным от IP-адреса
АРМ, с которого ему разрешена работа

active_session_ different_ip

Блокировка пользователя с измененным в рамках активной сессии IP

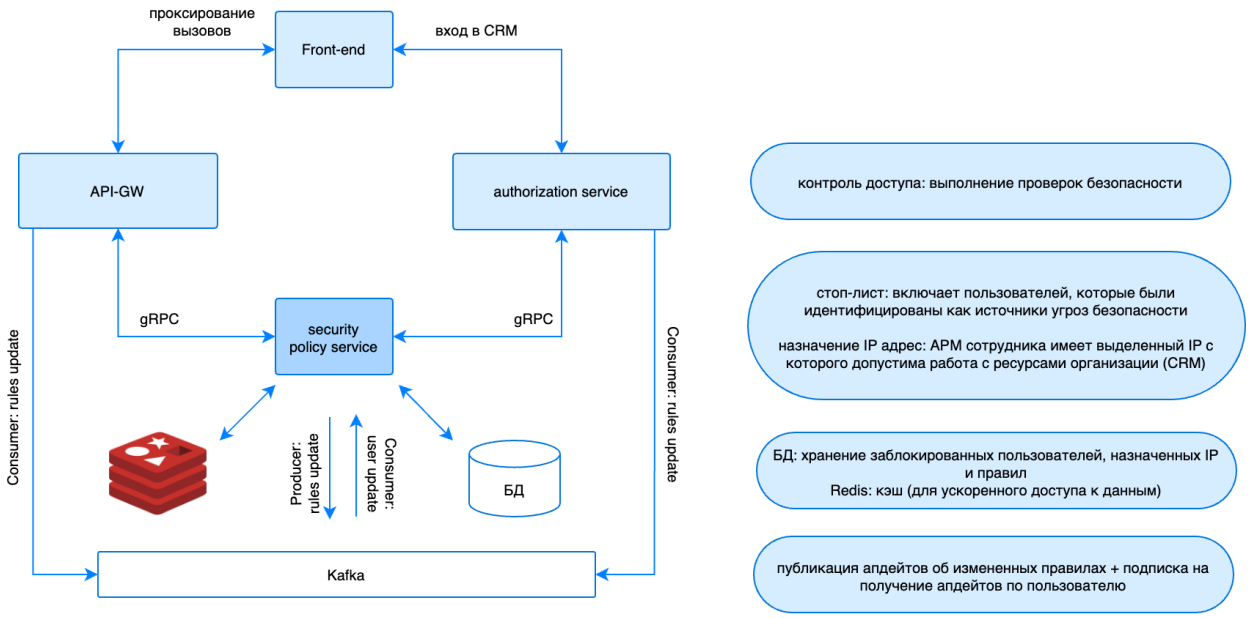
Кейс:
Пользователь успешно авторизовался в системе, но его IP был изменен в рамках пользовательской сессии (например, Аудитор ИБ обновил IP-адрес пользователя)

active_session_ re_auth

Блокировка пользователя при повторной аутентификации при активной сессии

Кейс:
Аутентификация в систему с разрешенным IP и выполнение повторной аутентификации с идентичным логином/паролем

Архитектура



Кластер BPM

Общее

Модуль "Управление бизнес-процессами" предназначен для реализации паттерна Оркестрация в микросервисной архитектуре.

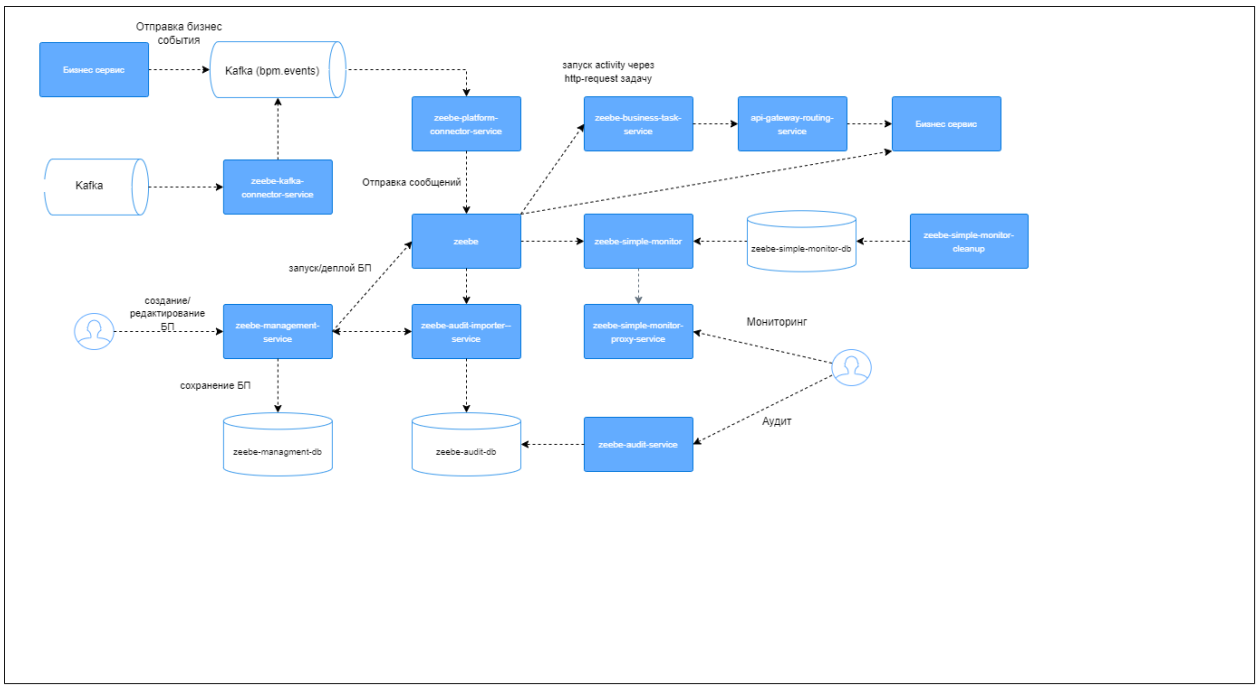
Задачи модуля:

1. Моделирование бизнес процессов в Нота Модус
2. Реализация интеграций микросервисов,
3. Создание внешних интеграций,
4. Автоматизация пользовательских процессов.

Особенности модуля:

1. Возможность написания процессов в нотации bpmn 2.0
2. Запуск процессов через события в кафку
3. Запуск процессов вызовом через апи
4. Запуск синхронных процессов
5. Запуск асинхронных процессов

Архитектура



Работа модуля разделена на 4 блока:

1. Создание/редактирование бизнес процесса. Деплой процесса в zeebe кластер.
2. Запуск и исполнение бизнес процесса
3. Мониторинг процесса

Создание/редактирование бизнес процесса. Деплой процесса в zeebe кластер

1. Интеграция с метамоделью сделана через ui редактор. Из метамодели получаются данные по модулям, объектам
2. Реализованы шаблонизированные обращения к эндпоинтам сервисов, которые могут быть использованы в процессе.
3. Для обеспечения передачи в контекст процесса глобальных переменных, которые задаются на уровне приложения используется zeebe-variables-service
4. Вся информация о созданном пользователем процессе хранится в zeebe-menegement-service

Запуск бизнес процесса

1. Запустить бизнес-процесс можно двумя способами:
    a. По триггеру из кафки
    b. Через апи zeebe-management-service
2. При запуске процесса через кафку
    a. zeebe-kafka-connector-service читает топик, указанный в конфигах и при наличии сообщений пересылает их в топик bpm.events
    b. zeebe-platform-connector-service читает топик bpm.events и принимает одно из следующих решений:
        i. Принятие решения о запуске БП (каждый БП привязан к тому или иному событию). Кроме простой привязки к событию можно задать фильтр - логическое выражение, в которое передается payload события.
        ii. Отправка сообщений в zeebe для "пробуждения" активных процессов, ожидающих нового значения атрибута, изменения атрибута или изменения атрибута с одного значения на другое.
    c. Схема события, отправляемого бизнес сервисом в топик bpm_events, описана в bpm-event-model

Наименование	Тип	Обязательный	Описание
id	int64	да	ID объекта - источника события. Например, если изменилась сделка, то ID = ID сделки
name	string	нет	Наименование события. В общем случае можно задавать произвольные имена событиям для CUD событий предполагается стандартное именование, по шаблону <имя_сущности>_<операция> например, opportunity_updated
userId	int64	да	ID пользователя - инициатора события. Для чего используется: один из примеров использования - необходимость отклонить запуск БП, если событие инициировано техническим пользователем.
paylod	Struct	нет	Произвольные данные. Для бизнес сервисов и стандартных операций следует придерживаться описанной ниже схемы
module	string	да	Наименование модуля
service	string	да	Наименование сервиса
object	string		Наименование объекта
actionType	string	да	Тип действия
correlationKey	string	нет	Ключ корреляции

d. Структура payload для события бизнес сервиса:

Тип события	Наименование	Тип	Описание
<entity>_updated	new	map<string, ?>	Коллекция атрибут-значение обновленного состояния объекта в виде ассоциативного массива. Может содержать только те данные, которые изменились.
	old	map<string, ?>	Коллекция атрибут-значение предыдущего состояния объекта в виде ассоциативного массива. Внимание! Если в структуре old не переданы параметры, но те же параметры переданы в new, то такие параметры будут считаться измененными и в zeebe будут отправлены соответствующие сообщения!
<entity>_created	new	map<string, ?>	Коллекция атрибут-значение созданного объекта в виде ассоциативного массива

e. **Пример:**
f. Событие обновления сделки
g. {

```
h.      "objectId": 4568,
i.      "name": "opportunity_updated",
j.   "userId": 1337,
k.      "payload": {
l.          "old": {},
m.          "new": {"opportunity_type": "hunter"}
n.      }
o. }
p.
```

3. При запуске процесса через API zeebe-management-service
    a. Для асинхронного запуска вызвать Карточка метода startZeebeManagementServiceProcessByName
    b. Для синхронного запуска вызвать Карточка метода startZeebeManagementServiceSynchProcessByName

Исполнение бизнес процесса

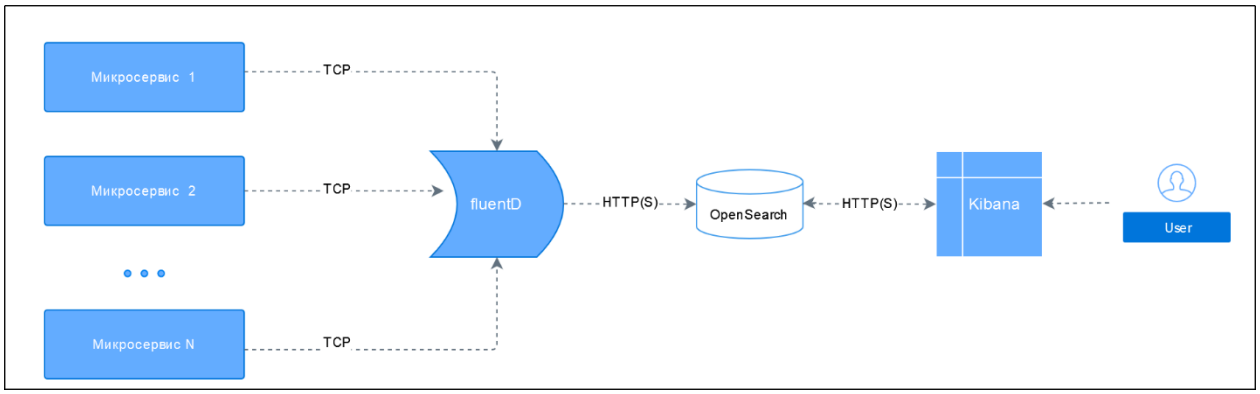
1. Во время исполнения бизнес процесс задействует коннекторы к lc-сервисам (создаются автоматически при генерации сервиса)
2. Также имеется интеграционный сервис zeebe-business-task-service. Он позволяет делать http-вызовы через api-gateway-routing-service к сервисам.

Мониторинг процесса

1. Для мониторинга бизнес процесса используется zeebe-simple-monitor-proxy-service, который обеспечивает авторизованной доступ к zeebe-simple-monitor.
2. Для отображения аудита по запущенным процессам используется zeebe-audit-service, за наполнение отвечает zeebe-audit-importer-service .
3. Для защиты от переполнения БД используется сервис zeebe-simple-monitor-cleanup

Логирование

Работа с логами системы осуществляется с помощью OpenSearch (Elasticsearch) + Fluentd+Kibana



Настройка подключения к FluentD производится командой DevOps при разворачивании контура. Параметры настройки сохраняются в переменные окружения: `logtcphost, logtcpport`

Дополнительные настройки логирования указываются в конфигурационном файле проекта (микросервиса/библиотеки)

```
log:
```

```
service-name: ИМЯ_СЕРВИСА

category:  // описание правил логирования для пакетов
  root:    // Правила для всех
    level: INFO  // уровень логирования
  "ru.tsc":// Правило для пакетов, начинающихся на ru.tsc
    level:
      tcp: TRACE // уровень логирования для отправки логов по TCP в fluentD
      console: INFO // уровень логирования для консоли
```

Вход-выход-ошибку для операций — **INFO/ERROR**. Допускается логировать какой-либо id на входе, количество изменённых сущностей на выходе, сообщение ошибки.

Запрос-ответ сервиса — **DEBUG/TRACE**. На DEBUG логируется URL, заголовки, параметры запроса и ответа, на TRACE добавляется логирование тела запроса. Если тело запроса — файл или бинарные данные, логировать не нужно в любом случае. Такое логирование в большинстве случаев реализуется в интерцепторах/wrapper-ах/фильтрах, которые есть в util-либах. В этот момент выставляются ThreadContext-параметры traceId, spanId, parentId.

Ошибка при вызове — **ERROR** — логируется весь стэктрейс. Такое логирование должно быть предусмотрено перед отправкой ответа, чтобы включить в него все вложенные ошибки.

Вызов внешнего сервиса — **DEBUG/TRACE/ERROR** - На DEBUG логируется URL, заголовки, параметры запроса и ответа, на TRACE добавляется логирование тела запроса. Для ошибки логируется только сообщение. При вызове в ThreadContext проставляется сквозной параметр **webRequestId** (**grpcRequestId** для gRPC), который должен быть в логах запроса/ответа/ошибки. При ошибке логируется только сообщение. Такое логирование в большинстве случаев реализуется в интерцепторах/wrapper-ах/фильтрах, которые есть в util-либах.

Запрос к БД - **DEBUG/TRACE/ERROR** — На DEBUG логируются параметры запроса и факт ответа (либо количество возвращаемых сущностей), на TRACE добавляется текст запроса и ответ в виде возвращаемой структуры. При вызове в ThreadContext проставляется сквозной параметр **queryId**. При ошибке логируется только сообщение. Логирование руками делать не нужно, всё есть в util-либах.

Запрос в кэш - **DEBUG/TRACE/ERROR** — На DEBUG логируется тело запроса и его тип. Тело запроса должно логироваться только на уровне TRACE. Для батча логируется количество запросов в батче. При вызове в ThreadContext проставляется сквозной параметр **redisRequestId**. Для получения одного значения по ключу и батча есть готовое логирование в util-либах, для остальных случаев его нужно делать руками (не забыв redisRequestId).

Отправка сообщения в Kafka - **DEBUG/TRACE/ERROR**. На DEBUG логируются заголовки и топик запроса, а также топик, оффсет и партиция после отправки сообщения. На TRACE логируется ещё тело сообщения. При вызове в ThreadContext проставляется сквозной параметр **kafkaRequestId**. При ошибке логируется только сообщение.

Список обязательных для логирования атрибутов:

sessionId	Идентификатор сессии пользователя
traceId	Сквозной идентификатор запроса
spanId	Идентификатор текущего шага выполнения запроса

parentId	Идентификатор предыдущего (родительского) шага выполнения запроса
userId	Идентификатор пользователя в системе
operationName	Наименование логической операции в рамках сервиса
webRequestId	Уникальный идентификатор запроса при межсервисном вызове
grpcRequestId	Уникальный идентификатор вызова удаленной процедуры по gRPC
queryId	Уникальный идентификатор запроса в БД
redisRequestId	Уникальный идентификатор запроса в Redis
kafkaRequestId	Уникальный идентификатор запроса при отправке сообщения в Kafka

Обязательность атрибутов для логирования в зависимости от точки входа указана в матрице заполнения атрибутов :

sessionId	x	x	x	x				x	
traceId	x	x	x	x					
spanId	x	x	x	x					
parentId	x	x	x	x					
userId	x	x	x	x				x	
operationName	x	x	x	x	x	x	x	x	
webRequestId	x	x		x					
grpcRequestId			x						
queryId	x	x	x	x					
redisRequestId	x	x	x	x					
kafkaRequestId					x				

Для корректного добавления атрибутов логирования необходимо использовать Mapped Diagnostic Context (MDC).

https://logging.apache.org/log4j/2.x/manual/thread-context.html

Мониторинг

Модуль мониторинга. В качестве инструмента сбора метрик используется Prometheus, для анализа и визуализации метрик Grafana, для отправки уведомлений Alert Manager

Postgresql

Объектно-реляционная система управления базами данных. Используется микросервисами для хранения данных, при этом под каждый разворачиваемый микросервис может выделяться как отдельная БД, так и разделяемая с другими сервисами.

Redis

Система управления базами данных класса NoSQL, работающая со структурами данных типа «ключ — значение». Служит для реализации кэша. В составе НОТА МОДУС Redis используется совместно с разными сервисами, конкретные варианты его использования приведены в описании конкретных сервисов.

Файловое хранилище S3

Элемент архитектуры, обеспечивающий хранение документов, файлов, вложений. Реализован на базе объектного хранилища MinIO, совместимого с Amazon S3 API.

OpenSearch

Программная поисковая система, реализующая возможность полнотекстового поиска, обеспечивающая возможность горизонтального масштабирования.

Kafka

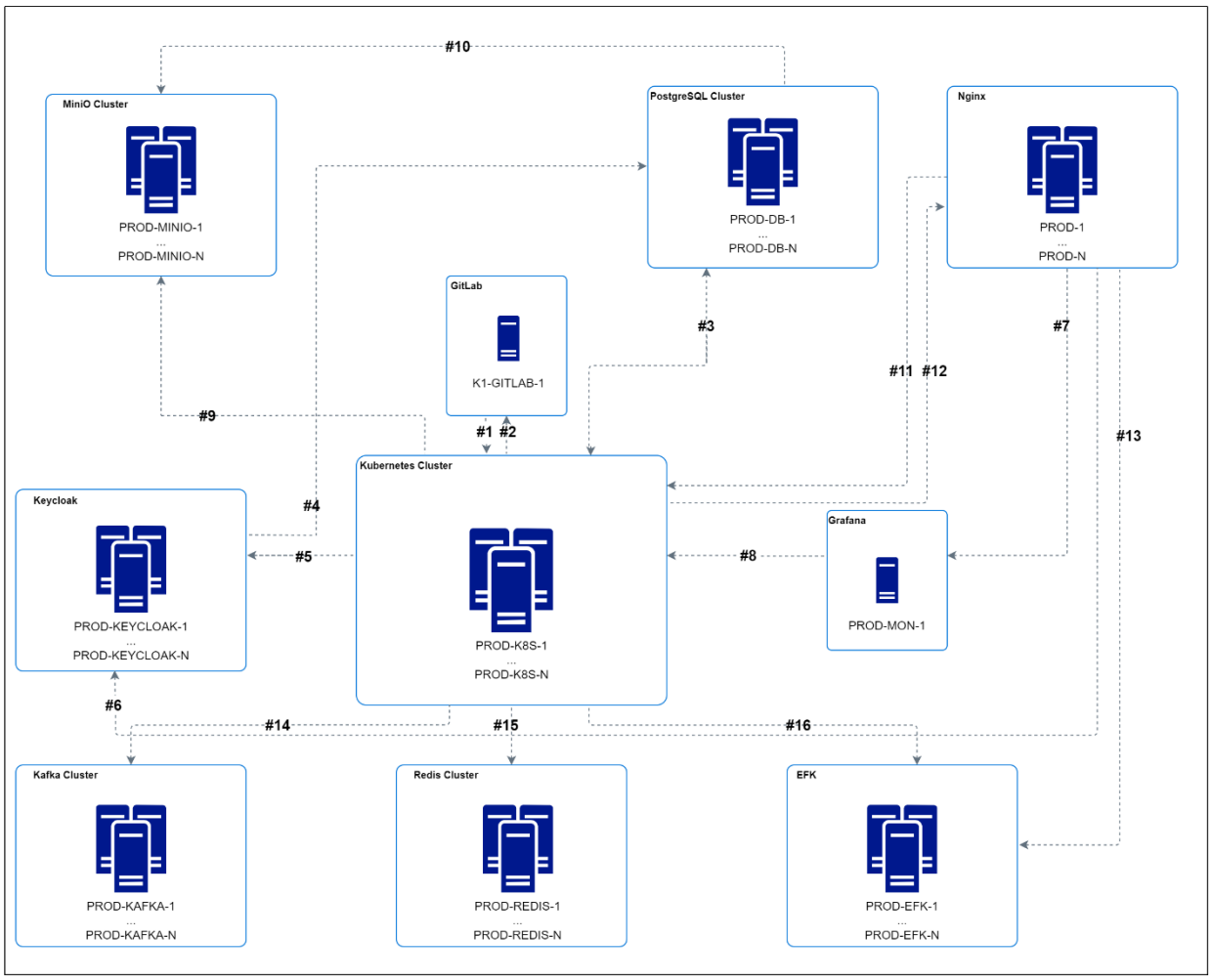
Программный брокер сообщений, реализующий в архитектуре возможность обмена информацией о наступлении событий. Есть два основных стандартных паттерна использования Kafka в составе НОТА МОДУС – для сбора от микросервисов аудируемых событий и передачи их в модуль аудита, и для интеграции бизнес-модулей T1CRM с модулем BPM. Эти паттерны подробнее изложены подробнее в описании соответствующих модулей.

Схема развёртывания системы

Типовой набор серверов, на которых разворачивается система, а также сетевых взаимодействий между ними, представлен на схеме ниже. Количество серверов в кластере для каждой конкретной установки, а также мощность серверов и тип (физические/виртуальные) зависит от планируемой нагрузки, и определяется либо экспертным заключением (рекомендуется для контуров разработки и тестирования), либо по результатам нагрузочного тестирования (рекомендуется для промышленных контуров).

Наименование серверов приведено на примере продуктивной среды, у прочих сред рекомендуется именовать сервера с другим префиксом.

Более полная информация о развёртывании системы приведена в документе «Инструкция по развёртыванию НОТА МОДУС»



Детализация сетевых взаимодействий

1	k1-gitlab-1	prod-k8s-1 ... prod-k8s-N	TCP: 6443	Раскатка сервисов Kubernetes из Гитлаба
2	prod-k8s-1 ... prod-k8s-N	k1-gitlab-1	TCP: 5050 8443 443	обращение сервисов фабрики к GitLab
3	prod-k8s-1 ... prod-k8s-N	prod-db-1 ... prod-db-N	TCP: 5432	Использование сервисами, развёрнутыми в Kubernetes БД PostgreSQL
4	prod-keycloak-1 ...	prod-db-1 ... prod-db-N	TCP: 5432	Использование серверами keycloak БД PostgreSQL

	prod-keycloak-N			
5	prod-1 ... prod-n	prod-keycloak-1 ... prod-keycloak-N	TCP: 8443 8080	Аутентификация через Keycloak сервисов в Kubernetes
6	prod-k8s-1 ... prod-k8s-N	prod-keycloak-1 ... prod-keycloak-N	TCP: 8443	Аутентификация через Keycloak пользователей
7	prod-1 ... prod-n	prod-mon-1	TCP: 3000	Вход пользователей в Grafana
8	prod-mon-1	prod-k8s-1 ... prod-k8s-N	TCP: 9090	Снятие метрик серверов Kubernetes для мониторинга
9	prod-k8s-1 ... prod-k8s-N	prod-minio-1 ... prod-minio-N	TCP: 9000	Обращение сервисами, развёрнутыми в Kubernetes, к файловому хранилищу MiniO
10	prod-1 ... prod-n	prod-minio-1 ... prod-minio-N	TCP: 9001	Обращение веб-сервиса Nginx к файловому хранилищу MiniO
11	prod-1 ... prod-n	prod-k8s-1 ... prod-k8s-N	TCP: 443 80	Обращение пользователей к сервисам в kubernetes
12	prod-k8s-1 ... prod-k8s-N	prod-1 ... prod-n	TCP: 443 8443 9000	Ответы от сервисов kubernetes к пользователям
13	prod-1 ... prod-n	prod-efk-1 ... prod-efk-N	TCP: 5601	Вход административного персонала в Elasticsearch

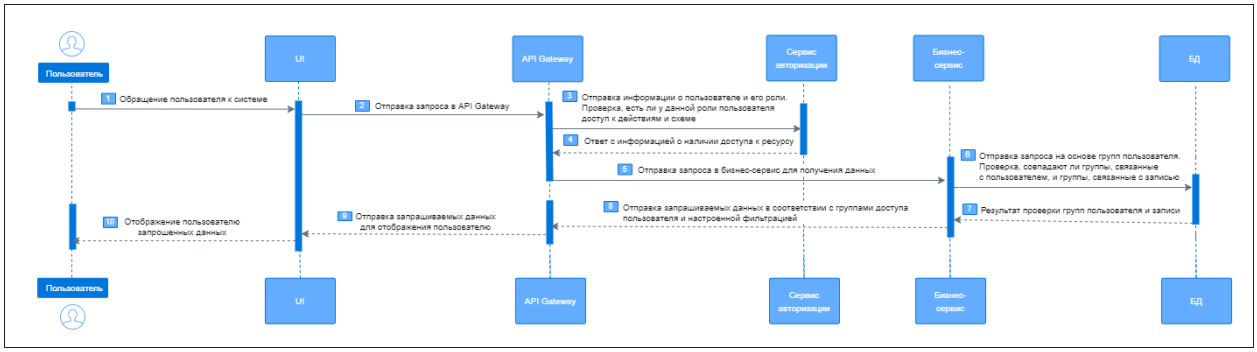
14	prod-k8s-1 ... prod-k8s-N	prod-kafka-1 ... prod-kafka-N	TCP: 9092	Обращение сервисами, развёрнутыми в Kubernetes, к очередям в kafka
15	prod-k8s-1 ... prod-k8s-N	prod-redis-1 ... prod-redis-N	TCP: 6379	Обращение сервисами, развёрнутыми в Kubernetes, к кешу redis
16	prod-k8s-1 ... prod-k8s-N	prod-efk-1 ... prod-efk-N	TCP: 30764	Обращение сервисами, развёрнутыми в Kubernetes, к elasticsearch

Ролевая модель и права доступа

Права доступа к системе проверяются на следующих уровнях:

1. Проверка доступа к элементам UI
2. Проверка доступа к действиям системы API (методам и атрибутам)
3. Проверка доступа к записям (ACL)

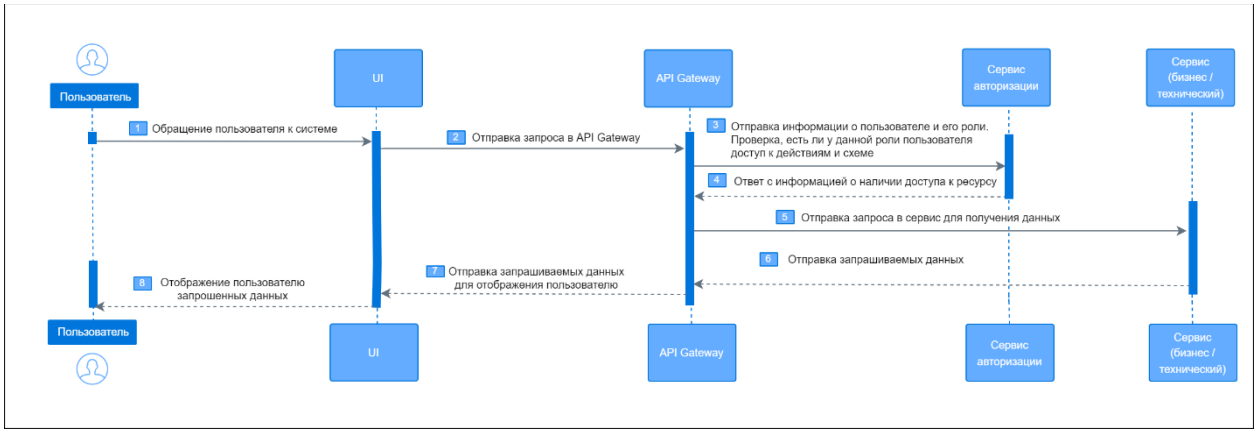
Проверка доступа к элементам UI (экраны/виджеты/элементы UI)



1	Пользователь	UI	Обращение пользователя к системе	
2	UI	API Gateway	Отправка запроса в API Gateway	
3	API Gateway	Сервис авторизации	Отправка информации о пользователе и его роли. Проверка, есть ли у данной роли пользователя доступ к действиям и схеме	
4	Сервис авторизации	API Gateway	Ответ с информацией о наличии доступа к к элементам UI:	

			• Если доступ отсутствует, то пользователю отобразится ошибка на экране • Если доступ имеется, то далее выполняется п.5	
5	API Gateway	Бизнес-сервис	Отправка запроса в бизнес-сервис для получения данных	
6	Бизнес-сервис	API Gateway	Отправка запрашиваемых данных	
7	API Gateway	UI	Отправка запрашиваемых данных для отображения пользователю	
8	UI	Пользователь	Отображение пользователю запрошенных данных	

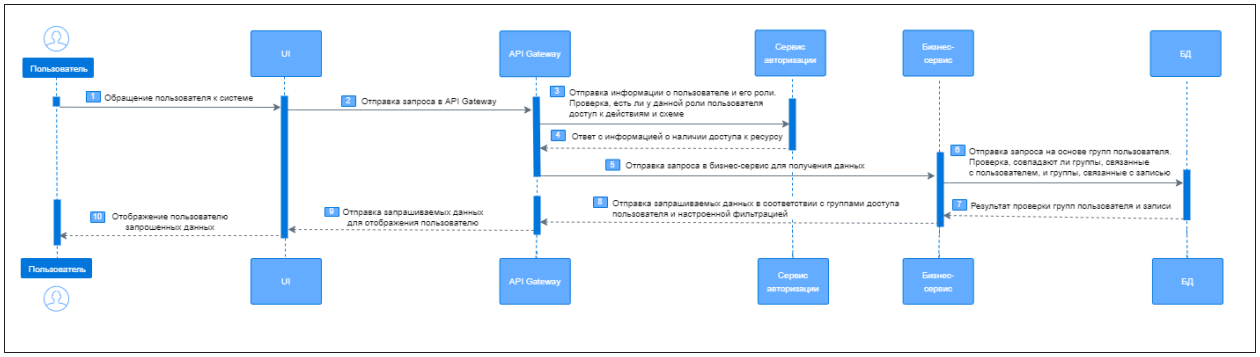
Проверка доступа к методам и атрибутам API



1	Пользователь	UI	Обращение пользователя к системе	
2	UI	API Gateway	Отправка запроса в API Gateway	
3	API Gateway	Сервис авторизации	Отправка информации о пользователе и его роли. Проверка, есть ли у пользователя данной роли доступ к действиям и схеме	
4	Сервис авторизации	API Gateway	Ответ с информацией о наличии доступа к ресурсу (действиям и схеме): • Если доступ отсутствует, то пользователю отобразится ошибка на экране • Если доступ имеется, то далее выполняется п.5	
5	API Gateway	Сервис (бизнес/технический)	Отправка запроса в сервис (бизнес/технический) для получения данных В технический сервис может обращаться пользователь с ролью администратор. У администратора не будет проверяться пересечение по группам. Будет проверяться только доступ к действиям	

6	Сервис (бизнес/технический)	API Gateway	Отправка запрашиваемых данных	
7	API Gateway	UI	Отправка запрашиваемых данных для отображения пользователю	
8	UI	Пользователь	Отображение пользователю запрошенных данных	

Проверка доступа к записям (ACL)



1	Пользователь	UI	Обращение пользователя к системе
2	UI	API Gateway	Отправка запроса в API Gateway
3	API Gateway	Сервис авторизации	Отправка информации о пользователе и его роли. Проверка, есть ли у данной роли пользователя доступ к действиям и схеме
4	Сервис авторизации	API Gateway	Ответ с информацией о наличии доступа к ресурсу (действиям и схеме): • Если доступ отсутствует, то пользователю отобразится ошибка на экране • Если доступ имеется, то далее выполняется п.5
5	API Gateway	Бизнес-сервис	Отправка запроса в бизнес-сервис для получения данных
6	Бизнес-сервис	База данных	Отправка запроса на основе групп пользователя. Проверка совпадают ли группы, связанные с пользователем, и группы, связанные с записью
7	База данных	Бизнес-сервис	Отправка результата проверки совпадения групп пользователя и записи: • Если ни одной группы нет, то пользователю отказывают в чтении данных этого бизнес объекта • Иначе пользователь получает данные бизнес объекта согласно своей роли (выполняется п.8)
8	Бизнес-сервис	API Gateway	Отправка запрашиваемых данных в соответствии с группами доступа пользователя и настроенной фильтрацией

9	API Gateway	UI	Отправка запрашиваемых данных для отображения пользователю
10	UI	Пользователь	Отображение пользователю запрошенных данных

Меры обеспечения информационной безопасности в системе

Требования и стандарты ИБ, реализованные в системе

Объекты защиты системы

Описание журналов аудита

В Системе фиксируются события, подлежащие аудиту информационной безопасности, с указанием даты и времени их совершения, а также логином пользователя: создание, удаление, редактирование бизнес-сущностей (в том числе создание пользователя и присвоение ему ролей), неуспешные попытки логического доступа и т.д.

Таблица 1. Список событий, подлежащих аудиту

1	Аутентификация пользователя в системе	Keycloak	· Попытка удачного входа в систему · Попытка неудачного входа в систему	• Время дата • Логин пользователя • IP адрес • Статус аутентификации (Успех/Неуспех) • HTTP код ответа
2	События авторизации пользователя (проверка полномочий пользователя)	Provider-method-service Provider-view-service	Все действия, связанные с авторизацией (осуществляется проверка возможности вызова метода согласно ролевой модели, фиксируются только неуспешные попытки)	• Время дата • Логин пользователя • IP адрес • ID пользователя • HTTP код ответа • URI к которому было обращение • Тип запроса (POST/GET и др.) • Тело запроса • Тело ответа
3	Создание записи, Удаление записи, Изменение записи	бизнес-сервисы	Требуется зафиксировать факт создания, изменения, удаления записей в БД системы, в том числе создание пользователей и назначение им ролей. Исключения- изменения, выполняющиеся от имени ТУЗ[2].	• Время дата • Логин пользователя • IP адрес • ID пользователя • Имя сервиса • HTTP код ответа • URI к которому было обращение • Имя метода • Тип запроса (POST/ GET и др.) • Тело запроса • Тело ответа

				• Название сущности • Список Id изменяемой/создаваемой/удаляемой записи • [Опционально] Дополнительная информация (Изменяемые поля)
4	Выход пользователя из системы по запросу пользователя по кнопке «Выйти»	session-service	Фиксировать событие выхода пользователя из системы по кнопке «Выйти»	• Время дата • Логин пользователя • IP адрес • ID пользователя • HTTP код ответа

Обеспечение безопасного хранения журнала аудита в БД

Для обеспечения безопасного хранения логов журнала аудита в БД реализовано, что ТУЗ, под которой ведется автоматическая запись логов в журнал аудита ИБ, имеет права только на запись без возможности обновления и удаления.

Журнал событий защищен от несанкционированного доступа не только через интерфейс системы, но и на уровне БД.

Для этого:

- Предоставляется ответственному специалисту по информационной безопасности отдельная учетная запись с привилегированным доступом к серверу с операционной системой, где будет расположена БД PostgreSQL, для целей администрирования сервера.
- Предоставляется учетная запись для просмотра каталога с архивами аудита хранилища S.
- Реализовано бэкапирование БД PostgreSQL в части журнала аудита ИБ и стоп-листа. Для этих целей Заказчику будет предоставлена привилегированная учетная запись с полным доступом только к БД PostgreSQL. При необходимости, Заказчик может настроить дополнительную служебную учетную запись с ограниченными правами для сотрудника, проводящего настройку бэкапирования.
- Реализована выгрузка записей из таблицы БД журнала аудита в файл в формате CSV. Записи аудита в БД храниться не менее 1 года (по усмотрению заказчика). CSV файл формируется за каждый месяц для записей таблицы журнала аудита старше 1 года. Для этого в системе реализовано соответствующее задание, периодичность запуска которого настраивается с помощью системных параметров. Запуск задания производится под технологической учетной записью БД и отдельной технологической записью S3, с правами на удаление записей из таблицы журнала аудита. Алгоритм работы периодического задания:
    1. При запуске задания происходит поиск записей в таблице аудита, у которых created_at < [Текущая дата] - 365 дней, где created_at – значение колонки «Дата создания записи».
    2. Выполняется выгрузка найденный записей в CSV файлы в файловое хранилище S
    3. Имя файла формируется по маске: audit_ib_YYYYMM.csv, где YYYYMM - год и месяц, за который производится выгрузка. Записи за каждый месяц выгружаются в отдельный файл.
    4. При успешной выгрузке файла происходит удаление выгруженных записей из таблицы журнала аудита в БД.
    5. Если формирование файла завершилось с ошибкой, удаление записей из таблицы аудита не производится. Текст ошибки логируется в систему сбора логов EFK.

Постановка пользователя в стоп-лист в случае подозрительных действий с его стороны.

Разработать механизм блокировки сеанса пользователя и добавление его в стоп-лист при подозрительных действиях. В стоп-листе Аудитор ИБ может узнать, по какой причине у пользователя была заблокирована УЗ. В стоп-листе отображается такая информация как: дата, время, IP-адрес, id пользователя, id подозрительного события, детали события. В момент подозрительной активности, пользователь будет проинформирован системной нотификацией, с описанием проблемы и рекомендацией обратиться к Аудитору ИБ. Список подозрительных событий, подлежащих аудиту, приведен в Табл.2. Такие события также фиксируются в Журнале аудита.

Таблица 2. Список подозрительных событий для стоп-листа

1	Изменение IP-адреса пользователя при наличии активной сессии. (кейс, когда пользователь прошел аутентификацию с одним IP, а при работе в системе IP изменился)	Блокировка пользователя, Постановка учетной записи в стоп-лист. Системная нотификация пользователя: «Обнаружены подозрительные действия. Необходимо выполнить выход и вход в систему обязательно через личный кабинет.»	• Время дата • Логин пользователя • IP адрес первоначальный • IP адрес текущий
2	Вход в систему с IP-адреса, отличным от IP-адреса АРМ, с которой обычно работает сотрудник. (кейс, когда за сотрудником закреплен IP-адрес его АРМ, а вход в систему осуществляется с другой машины)	Блокировка пользователя, Постановка учетной записи в стоп-лист. Системная нотификация пользователя: «Обнаружены подозрительные действия. Кто-то пытается войти в ваш аккаунт с другого рабочего места. Обратитесь к Аудитору ИБ»	• Время дата • Логин пользователя • IP адрес
3	Повторная аутентификация при наличии активной сессии. (кейс, когда пользователь уже прошел аутентификацию, а в момент работы в системе осуществляется повторная аутентификация в ту же систему с идентичными идентификаторами пользователя (логин/пароль)).	Разрыв всех сессий пользователя. Блокировка пользователя, Постановка учетной записи в стоп-лист. Системная нотификация пользователя: «Обнаружены подозрительные действия. Кто-то пытается войти в систему под вашим логином/паролем, сессия разорвана. Обратитесь к Аудитору ИБ»	• Время дата • Логин пользователя • IP адрес • ID пользователя

Требования к журналу аудита действий пользователя

Реализован отдельный Экран Аудитора ИБ с двумя вкладками:

- Журнал аудита ИБ
- Стоп-лист пользователей.

Атрибуты экрана «Журнал аудита ИБ» перечислены в Таблице 3., атрибуты экрана «Стоп-лист пользователей» перечислены в Таблице 4.

Доступ к вкладкам экрана Аудитора ИБ имеет только пользователь с ролью «Аудитор ИБ».

В журнале аудита реализована возможность, вручную выгрузить журнал за выбранный промежуток времени в формате CSV.

Возможные действия на вкладке Журнал аудита:

- Просмотр только в режиме чтения.
- Фильтрация.

Возможные действия на вкладке Стоп-лист:

- Фильтрация
- Возможность разблокировать пользователя из Стоп-листа путем нажатия на кнопку «Разблокировать»

Таблица 3. Атрибутивный состав вкладки Журнал аудита

№	Атрибут	Тип данных	Возможна фильтрация
1.	Дата/время	Дата время	да
2.	Сервис	строка	да
3.	Логин пользователя	строка	да
4.	ID пользователя	число	да
5.	ID События	число	да
6.	Сущность	строка	да
7.	IP адрес первоначальный	строка	да
8.	IP адрес текущий	строка	да
9.	Тип запроса (POST/GET…)	строка	да
10.	URI к которому было обращение	строка	да
11.	HTTP код ответа	строка	да
12.	Имя метода	строка	да
13.	Детали	строка	

Таблица 4. Атрибутивный состав вкладки Стоп-лист

№	Атрибут	Тип данных	Возможна фильтрация
1.	Дата/время	Дата время	да
2.	IP адрес	строка	
3.	ID пользователя	число	да
4.	Логин	строка	да
5.	ID события	число	

6.	Блокировка	флаг	да
7.	Детали	строка	
8.	Разблокировать	Кнопка, при нажатии на которую у записи в поле Блокировка устанавливается false, при этом пользователь разблокируется..	

Интеграция с централизованной системой для сбора логов SIEM

T1 CRM может являться источником событий для системы мониторинга ИБ Заказчика (SIEM).

Для этого необходимо реализовать возможность передачи событий в централизованную систему для сбора логов Заказчика.

Управление доступом

При осуществлении пользователями доступа в ИС предусматривается предварительная регистрация пользователей в ИС и разграничение прав.

У пользователя отсутствует возможность самостоятельного расширения прав доступа без акцепта администратора.

Разграничение доступа пользователей, в т.ч. технических, к ИС осуществляется на основе ролевой модели прав доступа.

Обеспечивается обязательная аутентификация и авторизация всех пользователей ИС, в т.ч. технических.

Прямой доступ пользователей системы к БД системы исключен.

Управление уязвимостями

Все системные хосты и сервисы ИС сканируются на наличие уязвимостей до начала промышленной эксплуатации.

Для ИС проведен анализ защищенности до начала промышленной эксплуатации.

Уязвимости средней и высокой критичности устраняются до начала промышленной эксплуатации.

Сканирование на наличие уязвимостей проводится после каждого существенного изменения (только для элементов инфраструктуры, которые были изменены).

Приоритеты в устранении обнаруженных уязвимостей в ходе периодических сканирований, выставляются в соответствии с критичностью указанных уязвимостей.

Веб-страница написана с учетом OWASP Top 10.

В процессе разработки ИС для выявления уязвимостей используются такие классы инструментов, как: SAST/SCA/DAST.