Микросервисный фреймворк бизнес-логики Verdi 3.0

Инструкция по эксплуатации

embedika

2025 г.

1. Общие сведения о программе

Микросервисный фреймворк бизнес-логики Verdi 3.0 - это платформа, предназначенная для ускоренной разработки приложений бизнес-логики, обеспечивающая единый подход к реализации сервисов с соблюдением требований надежности, масштабируемости и безопасности.

Она позволяет подключать и объединять микросервисы, написанные на разных языках программирования (Java, Scala, Python, C#, C++), и обеспечивает их согласованную работу.

Совместима с программной платформой LAMP (Language Analysis Microservice Platform), предназначенной для работы с текстовыми данными.

После подключения и настройки необходимых модулей разработчики могут строить распределенные микросервисные приложения, которые включают в себя все ключевые аспекты бизнес-логики: от регистрации пользователей и настройки прав доступа до сложных процессов обработки данных, рассылки уведомлений и интеграции с внешними системами.

Платформа состоит из миксросервисов, предоставляющих следующие функциональные возможности:

- Alexandrina: сервис справочников - управление справочными значениями Системы.
- Api-gateway: сервис маршрутизации - точка входа в систему, выполняет отправку команд в сервисы платформы.
- Archiver: сервис архивации файлов - создание и просмотр архивов файлов.
- Buzzer: сервис уведомлений - управление сервисами отправки уведомлений и отправка уведомлений.
- Buzzer-subscription: сервис подписок - управление подписками пользователей и настройками соответствия событий уведомлениям.
- Buzzer-email: сервис уведомлений на электронную почту - управление группами, шаблонами и настройками уведомлений на почту.
- Buzzer-personal-page: сервис уведомлений в личном кабинете - управление секциями ЛК и шаблонами для уведомлений, получение списка уведомлений.
- Data-model: сервис модели данных - управление типами и схемами хранимых данных.
- Erika: сервис печатных форм - конвертация файлов и создание отчетов по шаблону.
- Indexation: сервис индексаций - управление индексациями данных (хранимых в сервисе модели данных) с целью их обработки в системе полнотекстового поиска или семантического и содержательного анализа с помощью системы LAMP.
- Integration: сервис интеграций - управление внешними клиентами и их доступами к данным модели данных Системы.

- Mon: сервис аутентификации - управление пользователями Системы, их группами и статусами.
- Nestor: сервис журналирования - фиксация и просмотр событий Системы.
- Oberto: сервис системы прав - управление правами и доступами пользователей к функционалу Системы.
- Attila: сервис композитных правил - управление правилами, которые далее декомпозируются в правила Oberto.
- Tech-constants: сервис технического справочника - просмотр и управление внутренними справочными значениями различных сервисов Системы.
- UI-storage: сервис хранения UI - управление визуальным стилем Системы.

В следующем разделе данной инструкции перечислены команды сервисов платформы с более подробным описанием всех функциональностей, которыми можно воспользоваться через API интерфейс.

2. Взаимодействие с сервисами фреймворка

2.1. Alexandrina: сервис справочников

Сервис для хранения и работы с редко меняющимися справочными данными, имеющими одинаковую структуру (обычно название и код) и списочный характер.

При помощи команд реализованы возможности:

- просмотр списка справочников и списка элементов справочника;
- создание, редактирование, удаление справочников и элементов справочников;
- скачивание и загрузка справочников.

2.1.1. Создание справочника (createCatalog)

Для создания справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createCatalog
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Строковый код справочника.	Длина не более 32. Уникальное значение.
title	String	Да	-	Человекочитаемое название справочника.	Длина не более 256.
metadata	JSON	Да	-	Дополнительные данные справочника - объект JSON.	-
isProtected	Boolean	Нет	true	Признак защиты записи от изменений.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного справочника в виде строки формата UUID.

Данные созданного справочника можно получить с помощью команд «getCatalogByCode», «listCatalogs» или «searchCatalogs», описанных в данной инструкции.

2.1.2. Получение данных справочника (getCatalogByCode)

Для получения данных справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getCatalogByCode
- Content-Type: application/json
- Request body: String, код существующего справочника (атрибут «code»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении справочника;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.1.3. Изменение справочника (updateCatalog)

Для изменения данных справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateCatalog
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	UUID	Да	-	ID справочника.	Существующий в сервисе идентификатор.
code	String	Да	-	Строковый код справочника.	Длина не более 32. Уникальное значение.
title	String	Да	-	Человекочитаемое название	Длина не более 256.

				справочника.	
metadata	JSON	Да	-	Дополнительные данные справочника - объект JSON.	-
version	Int	Да	-	Версия, полученная при запросе данных справочника. Нужна для оптимистичных блокировок.	Совпадает с текущей версией справочника.
isProtected	Boolean	Нет	true	Признак защиты записи от изменений.	-

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененного справочника можно получить с помощью команд «getCatalogByCode», «listCatalogs» или «searchCatalogs», описанных в данной инструкции.

2.1.4. Удаление справочника (removeCatalog)

Для удаления справочника и всех его элементов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/removeCatalog
- Content-Type: application/json
- Request body: UUID, идентификатор существующего справочника (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.1.5. Получение полного списка справочников (listCatalogs)

Для получения полного списка справочников сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listCatalogs

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о справочнике:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении справочника;

- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.1.6. Получение списка справочников с поддержкой фильтрации (searchCatalogs)

Для получения списка справочников сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/searchCatalogs
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. code;
        ii. title.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
code	InSetQuery, LikeQuery
title	InSetQuery, LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи справочников, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными о справочнике:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении справочника;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.1.7. Создание элемента справочника (createCatalogItem)

Для создания элемента справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createCatalogItem

- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
catalogId	UUID	Да	-	ID справочника.	Существующий в сервисе идентификатор.
code	String	Да	-	Строковый код элемента.	Длина не более 32. Уникальное значение.
title	String	Да	-	Человекочитаемое название элемента.	Длина не более 10000.
metadata	JSON	Да	-	Дополнительные данные элемента - объект JSON.	-
isProtected	Boolean	Нет	true	Признак защиты записи от изменений.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного элемента в виде строки формата UUID.

Данные созданного элемента можно получить с помощью команд «getCatalogItemByCode» или «listItemsByCatalog», описанных в данной инструкции.

2.1.8. Получение данных элемента справочника (getCatalogItemByCode)

Для получения данных элемента справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getCatalogItemByCode
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
catalogCo	String	Да	-	Код справочника.	Существующий

de					в сервисе код справочника.
itemCode	String	Да	-	Код элемента справочника.	Существующий в указанном справочнике код элемента.

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении элемента;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.1.9. Изменение элемента справочника (updateCatalogItem)

Для изменения данных элемента справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateCatalogItem
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	UUID	Да	-	ID элемента.	Существующий в сервисе идентификатор.
code	String	Да	-	Строковый код элемента.	Длина не более 32. Уникальное значение.
title	String	Да	-	Человекочитаемое название элемента.	Длина не более 10000.
metadata	JSON	Да	-	Дополнительные данные справочника - объект JSON.	-
version	Int	Да	-	Версия, полученная при запросе данных элемента. Нужна для оптимистичных	Совпадает с текущей версией элемента.

				блокировок.	
isProtected	Boolean	Нет	true	Признак защиты записи от изменений.	-

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененного элемента можно получить с помощью команд «getCatalogItemByCode» или «listItemsByCatalog», описанных в данной инструкции.

2.1.10. Удаление элемента справочника (removeCatalogItem)

Для удаления элемента справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/removeCatalogItem
- Content-Type: application/json
- Request body: UUID, идентификатор существующего элемента справочника (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.1.11. Получение списка элементов справочника с поддержкой фильтрации (listItemsByCatalog)

Для получения списка элементов справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listItemsByCatalog
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
data	UUID	Да	-	ID справочника.	Существующий в сервисе идентификатор.

search	Search	Да	-	Параметры сортировки, фильтрации и пейджинга списочного запроса.	-

Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:

- Доступные поля для сортировки:
  a. code;
  b. created;
  c. modified.
- Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id	InSetQuery, LikeQuery
catalogId	InSetQuery, LikeQuery
code	InSetQuery, LikeQuery
title	InSetQuery, LikeQuery
archived	InSetQuery
created	InSetQuery, LikeQuery, RangeQuery
modified	InSetQuery, LikeQuery, RangeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи элементов указанного справочника, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными об элементе:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении элемента;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.1.12. Выгрузка справочников в формат JSON (exportCatalogs)

Для выгрузки данных справочников и их элементов в файл формата JSON необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/exportCatalogs
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
ids	List[UUID]	Да	-	Список ID справочников.	Существующие в сервисе идентификаторы.
exclude	Boolean	Нет	false	Признак исключения из выгрузки указанных справочников: ● false - выгружаются только справочники с идентификаторами, указанными в «ids»; ● true - выгружаются все справочники, кроме указанных в «ids».	-

При успешном выполнении команды в теле ответа поле «value» содержит относительный URL сформированного JSON-файла в файловом хранилище.

В файле содержаться следующие поля:

- «version» - равно «1.0»;
- «dataType» - равно «catalogs»;
- «data» - содержит все данные справочников и их элементов, за исключением системных атрибутов (например, идентификаторов). Данные справочников и их элементов в сервисе можно получить с помощью команд «listCatalogs» и «listItemsByCatalog», описанных в данной инструкции.

2.1.13. Преобразование данных справочников в формат для импорта (parseCatalogs)

Для преобразования данных справочников в формат для импорта необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/parseCatalogs
- Content-Type: application/json
- Request body: UUID, идентификатор JSON-файла в файловом хранилище в бакете (название бакета согласно переменной окружения ALEXANDRINA_TEMP_BUCKET). JSON должен состоять из полей согласно формату выгрузки справочников (команда «exportCatalogs», описанная в данной инструкции).

При успешном выполнении команды в теле ответа поле «value» содержит значение поля «data» JSON-файла, идентификатор которого был указан.

2.1.14. Импорт справочников (importCatalogs)

Для импорта справочников и их элементов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/importCatalogs
- Content-Type: application/json
- Request body: Массив JSON-объектов, содержащих следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Строковый код справочника.	Длина не более 32.
title	String	Да	-	Человекочитаемое название справочника.	Длина не более 256.
metadata	JSON	Да	-	Дополнительные данные справочника - объект JSON.	-
isProtected	Boolean	Нет	true	Признак защиты записи от изменений.	-
ignoreProtection	Boolean	Нет	false	Игнорирование признака защиты	

				справочника от изменений при импорте данных (т.е. справочник будет обновлен независимо от значения признака).	
items	List[CatalogItemExchangeDTO]	Да	-	Список элементов справочника.	-

Тип данных **CatalogItemExchangeDTO** - структура данных сервиса справочников, которая используются для описания данных элемента справочника для импорта.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Строковый код элемента.	Длина не более 32.
title	String	Да	-	Человекочитаемое название элемента.	Длина не более 10000.
metadata	JSON	Да	-	Дополнительные данные элемента - объект JSON.	-
isProtected	Boolean	Нет	true	Признак защиты записи от изменений.	-
archived	Boolean	Да	-	Признак архивности элемента.	-

Логика работы команды зависит от поля «code»:

- уникальное значение - создается новый справочник;
- существующее значение - изменяется соответствующий справочник, в т.ч. добавляются/обновляются его элементы.

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект с пустым массивом в поле «failedImporte», что свидетельствует об отсутствии ошибок импорта.

Данные импортированных справочников и их элементов можно получить с помощью команд «getCatalogByCode» и «listItemsByCatalog», описанных в данной инструкции.

2.2. Api-gateway: сервис маршрутизации

2.2.1. Синхронный запрос (sync)

Для синхронного выполнения команды необходимо знать ее название и формат данных, которые она принимает. Далее необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/**sync**/<command_name>
    a. где <command_name> - название команды, которая будет вызвана синхронным способом
- Content-Type и Request body: зависят от формата данных, которые принимает команда

Результат выполнения команды будет доступен сразу в ответе. Тип ответа - JSON или файл, зависит от типа команды. Формат JSON зависит от конкретной выполняемой команды.

2.2.2. Асинхронный запрос (async)

Для асинхронного выполнения команды необходимо знать ее название и формат данных, которые она принимает. Далее необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/**async**/<command_name>
    a. где <command_name> - название команды, которая будет вызвана синхронным способом
- Content-Type и Request body: зависят от формата данных, которые принимает команда

В результате асинхронного запроса будет содержаться только одна строка с «id» команды, которая начала выполняться. Стоит отметить, что ответ будет JSON-строкой, а значит, содержать свое значение в кавычках.

Данные выполнения команды можно получить с помощью команды «status», описанной в данной инструкции.

2.2.3. Получение статуса (status)

Для получения статуса выполнения команды, вызванной асинхронным способом, необходимо выполнить следующий запрос:

- Метод: GET
- URL: http://<IP-адрес_BM>:8083/api/send/status/<command_id>

- где <command_id> - идентификатор команды, который был получен в ответе при асинхронном вызове

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными о выполнении команды, вызванной асинхронным способом:

- «status» - статус выполнения команды;
- «timestamp» - временная метка последнего обновления статуса;
- «value» - возвращается в случае, когда команда завершила выполнение («status» равен «Completed»). Значение и его структура зависят от логики команды, которая была вызвана асинхронно.

Также нужно иметь в виду, что если запросить статус сразу же после ответа асинхронного запроса, то может вернуться ошибка 404. Это вызвано тем, что статус команды не успевает дойти до сервиса CommandStatus через kafka. Механизм пуллинга должен это учитывать.

2.2.4. Загрузка файла (upload)

Для загрузки файла в файловое хранилище необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/upload
- Content-Type: multipart/form-data
- Request body: поле «file», в котором указывается файл для загрузки

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными о файле:

- идентификатор файла;
- URL файла в файловом хранилище.

Скачать загруженный файл можно с помощью команды «download», описанной в данной инструкции.

2.2.5. Скачивание файла (download)

Для скачивания файла из файлового хранилища необходимо выполнить следующий запрос:

- Метод: GET
- URL: http://<IP-адрес_BM>:8083/api/download/<fileUrl>
- где <fileUrl> - URL файла в файловом хранилище

При успешном выполнении команды ответ формируется как файл.

2.3. Archiver: сервис архивации файлов

2.3.1. Архивация файлов из файлового хранилища (archiver_kafka_archiveFromUrls)

Для создания архива из файлов хранилища необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/archiver_kafka_archiveFromUrls
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
archiveName	String	Да	-	Название архива.	-
filesUrls	List[String]	Да	-	Список URL файлов хранилища, которые необходимо архивировать.	-

При успешном выполнении команды в теле ответа поле «value» содержит URL архива и статус его формирования.

Данные архива можно получить с помощью команды «archiver_http_listArchives», описанной в данной инструкции.

2.3.2. Архивация файлов объектов сущностей сервиса Data-model с поддержкой фильтрации (archiver_kafka_archiveFromDataModelSearch)

Для создания архива из файлов хранилища, которые относятся к объектам сущностей сервиса Data-model необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/archiver_kafka_archiveFromDataModelSearch
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обяза	Значение	Описание	Ограничения

		тельное	по умолчанию		
archiveName	String	Да	-	Название архива.	-
includedFiles	String	Нет	“main”	Тип файлов, которые должны быть архивированы.	Одно из допустимых значений: • all - все файлы; • main - файлы, указанные в полях объекта согласно настройке сущности; • additional - файлы, указанные в дополнительных полях объекта.
additionalFilesFilter	List[String]	Нет	-	Перечень наименований дополнительных полей, файлы которых необходимо архивировать.	-
search	Search	Да	-	Параметры сортировки, фильтрации и списочного запроса. Не поддерживается пагинация.	-

Search - универсальный тип параметров списочного запроса. Для данной команды не поддерживается пагинация и сортировка, но поддерживается фильтрация.

Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
fileId	InSetQuery
size	InSetQuery
name	InSetQuery, LikeQuery

extension	InSetQuery, LikeQuery
url	InSetQuery, LikeQuery
created	InSetQuery, RangeQuery
modified	InSetQuery, RangeQuery
md5	InSetQuery

При успешном выполнении команды в теле ответа поле «value» содержит URL архива и статус его формирования.

Данные архива можно получить с помощью команды «archiver_http_listArchives», описанной в данной инструкции.

2.3.3. Получение списка архивов с поддержкой фильтрации (archiver_http_listArchives)

Для получения списка архивов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/archiver_http_listArchives
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. archiveUrl;
        ii. userId;
        iii. status.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
status	InSetQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи архивов, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными об архиве: URL архива; идентификатор пользователя, создавшего архив; статус; URL всех файлов, входящих в состав архива; список объектов, к которым относятся файлы (код сущности и идентификатор объекта этой сущности).

2.4. Buzzer: сервис уведомлений

Сервис для регистрации и дальнейшего управления сервисами отправки, а также пользовательские настройки относительно уведомлений.

При помощи команд реализованы возможности:

- регистрация, изменение, удаление и получение списка сервисов отправки;
- создание, изменение, удаление и получение списка пользовательских настроек;
- отправка уведомления.

2.4.1. Регистрация сервиса отправки (createSendingBuzzer)

Для регистрации сервиса отправки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createSendingBuzzer
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор сервиса отправки.	-
topic	String	Да	-	Очередь Kafka для сервиса отправки, в которую необходимо перенаправлять уведомления.	-
version	Number	Нет	-1	Версия описания.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор зарегистрированного сервиса в виде строки.

Данные зарегистрированного сервиса можно получить с помощью команды «listSendingBuzzer», описанной в данной инструкции.

2.4.2. Изменение сервиса отправки (updateSendingBuzzer)

Для изменения сервиса отправки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateSendingBuzzer
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор сервиса отправки.	Существующий в сервисе идентификатор.
topic	String	Да	-	Очередь Kafka для сервиса отправки, в которую необходимо перенаправлять уведомления.	-
version	Number	Да	-	Версия, полученная при запросе данных сервиса отправки. Нужна для оптимистичных блокировок.	Совпадает с текущей версией сервиса отправки.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененного сервиса отправки можно получить с помощью команды «listSendingBuzzer», описанной в данной инструкции.

2.4.3. Удаление сервиса отправки (deleteSendingBuzzer)

Для удаления сервиса отправки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteSendingBuzzer
- Content-Type: application/json

- Request body: String, идентификатор существующего сервиса отправки (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.4.4. Получение полного списка зарегистрированных сервисов (listSendingBuzzer)

Для получения полного списка зарегистрированных сервисов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listSendingBuzzer

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со всеми атрибутами, переданными пользователем: все данные, указанные при регистрации или изменении сервиса.

2.4.5. Создание пользовательской настройки уведомления (createSettingBuzzer)

Для создания пользовательской настройки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createSettingBuzzer
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Строковый код уведомления.	-
title	List[String]	Да	-	Список сервисов для отправки данного уведомления.	Существующий сервис отправки.

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной пользовательской настройки уведомления в виде строки формата UUID.

Данные созданной пользовательской настройки можно получить с помощью команды «listSettingBuzzer», описанной в данной инструкции.

2.4.6. Изменение пользовательской настройки уведомления (updateSettingBuzzer)

Для изменения пользовательской настройки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateSettingBuzzer
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	UUID	Да	-	ID пользовательской настройки уведомления.	Существующий в сервисе идентификатор.
code	String	Да	-	Строковый код уведомления.	-
title	List[String]	Да	-	Список сервисов для отправки данного уведомления.	Существующий сервис отправки.
version	Number	Да	-	Версия, полученная при запросе данных настройки. Нужна для оптимистичных блокировок.	Совпадает с текущей версией настройки.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной пользовательской настройки уведомления можно получить с помощью команды «listSettingBuzzer», описанной в данной инструкции.

2.4.7. Удаление пользовательской настройки уведомления (deleteSettingBuzzer)

Для удаления пользовательской настройки уведомления необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteSettingBuzzer
- Content-Type: application/json
- Request body: UUID, идентификатор существующей пользовательской настройки уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.4.8. Получение полного списка пользовательских настроек уведомлений (listSettingBuzzer)

Для получения полного списка пользовательских настроек уведомлений сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listSettingBuzzer

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о пользовательских настройках уведомлений:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении пользовательской настройки уведомления;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.4.9. Отправка уведомления (sendNotification)

Для отправки уведомления пользователю необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/sendNotification
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Строковый код уведомления.	Существующий в сервисе код уведомления.
userId	UUID	Да	-	Идентификатор пользователя, которому будет	Существующий в сервисе идентификатор

				отправлено уведомление.	пользователя.
data	JSON	Да	-	Данные уведомления.	-

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное отправление.

2.5. Buzzer-subscription: сервис подписок

Сервис для создания, управления и хранения подписок пользователей и настроек соответствия бизнес событий и необходимых уведомлений.

При помощи команд реализованы возможности:

- создание, изменение, удаление и получение списка настроек для события;
- создание, изменение, удаление и получение списка подписок пользователей.

2.5.1. Создание настройки для события (createEventSettingsBuzzer-subscription)

Для создания настройки для события необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createEventSettingsBuzzer-subscription
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Тип сущности модели данных.	Комбинация «entityType» и «eventType» каждой настройки должна быть уникальной.
eventType	String	Да	-	Тип события.	Комбинация «entityType» и «eventType» каждой настройки

					должна быть уникальной.
enabled	Boolean	Да	-	Активность подписки.	-
notification Code	String	Да	-	Код уведомления.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной настройки в виде строки формата UUID.

Данные созданной настройки можно получить с помощью команд «getEventSettingsBuzzer-subscription» или «listEventSettingsBuzzer-subscription», описанных в данной инструкции.

2.5.2. Получение данных настройки для события (getEventSettingsBuzzer-subscription)

Для получения данных настройки для события необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getEventSettingsBuzzer-subscription
- Content-Type: application/json
- Request body: UUID, идентификатор существующей настройки для события (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении настройки для события;
- системные атрибуты: данные, сгенерированные сервисом автоматически (например, идентификатор).

2.5.3. Изменение настройки для события (updateEventSettingsBuzzer-subscription)

Для изменения данных настройки для события необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateEventSettingsBuzzer-subscription
- Content-Type: application/json

- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	UUID	Да	-	ID настройки события.	Существующий в сервисе идентификатор.
entityType	String	Да	-	Тип сущности модели данных.	Комбинация «entityType» и «eventType» каждой настройки должна быть уникальной.
eventType	String	Да	-	Тип события.	Комбинация «entityType» и «eventType» каждой настройки должна быть уникальной.
enabled	Boolean	Да	-	Активность подписки.	-
notificationCode	String	Да	-	Код уведомления.	-
version	Int	Да	-	Версия, полученная при запросе данных настройки события. Нужна для оптимистичных блокировок.	Совпадает с текущей версией настройки события.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной настройки для события можно получить с помощью команд «getEventSettingsBuzzer-subscription» или «listEventSettingsBuzzer-subscription», описанных в данной инструкции.

2.5.4. Удаление настройки для события (deleteEventSettingsBuzzer-subscription)

Для удаления настройки для события необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteEventSettingsBuzzer-subscription
- Content-Type: application/json
- Request body: UUID, идентификатор существующей настройки для события (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.5.5. Получение полного списка настроек для событий (listEventSettingsBuzzer-subscription)

Для получения полного списка настроек для событий сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listEventSettingsBuzzer-subscription

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о настройках для событий:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении настройки для события;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.5.6. Создание подписки пользователя (createSubscriptionBuzzer-subscription)

Для создания подписки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createSubscriptionBuzzer-subscription
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Тип сущности модели данных.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки

					должна быть уникальной.
entityId	String	Да	-	Идентификатор объекта сущности.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки должна быть уникальной.
eventType	String	Нет	-	Тип события.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки должна быть уникальной.
enabled	Boolean	Да	-	Активность подписки.	-
userId	UUID	Да	-	Идентификатор пользователя, который подписывается на событие.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки должна быть уникальной.

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной подписки в виде строки формата UUID.

Данные созданной подписки можно получить с помощью команд «getSubscriptionBuzzer-subscription» или «listSubscriptionBuzzer-subscription», описанных в данной инструкции.

2.5.7. Получение данных подписки пользователя (getSubscriptionBuzzer-subscription)

Для получения данных подписки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getSubscriptionBuzzer-subscription
- Content-Type: application/json
- Request body: UUID, идентификатор существующей подписки (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении подписки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.5.8. Изменение подписки пользователя (updateSubscriptionBuzzer-subscription)

Для изменения подписки пользоватля необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateSubscriptionBuzzer-subscription
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	UUID	Да	-	ID подписки.	Существующий в сервисе идентификатор.
entityType	String	Да	-	Тип сущности модели данных.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки должна быть уникальной.
entityId	String	Да	-	Идентификатор объекта сущности.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки должна быть уникальной.
eventType	String	Нет	-	Тип события.	Комбинация «entityType», «entityId», «eventType» и

					«userId» каждой настройки должна быть уникальной.
enabled	Boolean	Да	-	Активность подписки.	-
userId	UUID	Да	-	Идентификатор пользователя, который подписывается на событие.	Комбинация «entityType», «entityId», «eventType» и «userId» каждой настройки должна быть уникальной.
version	Int	Да	-	Версия, полученная при запросе данных подписки пользователя. Нужна для оптимистичных блокировок.	Совпадает с текущей версией подписки

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной подписки пользователя можно получить с помощью команд «getSubscriptionBuzzer-subscription» или «listSubscriptionBuzzer-subscription», описанных в данной инструкции.

2.5.9. Удаление подписки пользователя (deleteSubscriptionBuzzer-subscription)

Для удаления подписки пользователя необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteSubscriptionBuzzer-subscription
- Content-Type: application/json
- Request body: UUID, идентификатор существующей подписки пользователя (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.5.10. Получение полного списка подписок пользователей (listSubscriptionBuzzer-subscription)

Для получения полного списка подписок пользователей необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listSubscriptionBuzzer-subscription

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о подписках пользователей:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении подписки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.6. Buzzer-email: сервис уведомлений на электронную почту

Сервис для настройки и сбора уведомлений для отправки по электронной почте.

При помощи команд реализованы возможности:

- создание, изменение, удаление и получение списка настроек уведомления;
- создание, изменение, удаление и получение списка групп уведомлений;
- создание, изменение, удаление и получение списка шаблонов уведомлений;
- создание, изменение, удаление и получение списка рассылок уведомлений;
- создание и удаление шаблонов рассылок уведомлений.

2.6.1. Создание настройки для уведомления (createSettingEmail)

Для создания настройки для уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createSettingEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор	Существующий

				настройки уведомления.	код уведомления.
deliveryId	String	Да	-	Идентификатор рассылки.	Существующий идентификатор.
groupId	String	Да	-	Идентификатор группы уведомлений.	Существующий идентификатор.
version	Int	Нет	1	Версия настройки.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор настройки уведомления в виде строки.

Данные созданной настройки уведомления можно получить с помощью команд «getSettingEmail» и «listSettingEmail», описанных в данной инструкции.

2.6.2. Получение данных настройки уведомления (getSettingEmail)

Для получения данных настройки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getSettingEmail
- Content-Type: application/json
- Request body: String, идентификатор существующей настройки (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении настройки уведомления;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.3. Изменение настройки уведомления (updateSettingEmail)

Для изменения данных настройки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateSetting
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор настройки уведомления.	Существующий код уведомления.
deliveryId	String	Да	-	Идентификатор рассылки.	Существующий идентификатор.
groupId	String	Да	-	Идентификатор группы уведомлений.	Существующий идентификатор.
version	Int	Да	-	Версия, полученная при запросе данных настройки уведомления. Нужна для оптимистичных блокировок.	Совпадает с текущей версией настройки уведомления.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной настройки уведомления можно получить с помощью команд «getSettingEmail» и «listSettingEmail», описанных в данной инструкции.

2.6.4. Удаление настройки уведомления (deleteSettingEmail)

Для удаления настройки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteSettingEmail
- Content-Type: application/json
- Request body: Stirng, идентификатор существующей настройки уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.6.5. Получение полного списка настроек уведомлений (listSettingEmail)

Для получения полного списка настроек уведомлений сервиса необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/listSettingEmail

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о настройке уведомлений:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении настройки уведомления;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.6. Создание группы уведомлений (createGroupEmail)

Для создания группы уведомлений необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createGroupEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор группы уведомлений.	Уникальное значение.
name	String	Да	-	Название группы.	-
version	Int	Нет	1	Версия группы уведомлений.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной группы уведомлений в виде строки.

Данные созданной группы уведомлений можно получить с помощью команд «getGroupEmail» и «listGroupEmail», описанных в данной инструкции.

2.6.7. Получение данных группы уведомлений (getGroupEmail)

Для получения данных группы уведомлений необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getGroupEmail

- Content-Type: application/json
- Request body: String, идентификатор существующей группы уведомлений (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении группы уведомлений;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.8. Изменение группы уведомлений (updateGroupEmail)

Для изменения данных группы уведомлений необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateGroupEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор группы уведомлений.	Существующий в сервисе идентификатор.
name	String	Да	-	Название группы.	-
version	Int	Да	-	Версия, полученная при запросе данных группы уведомлений. Нужна для оптимистичных блокировок.	Совпадает с текущей версией группы уведомлений.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной группы уведомлений можно получить с помощью команд «getGroupEmail» и «listGroupEmail», описанных в данной инструкции.

2.6.9. Удаление группы уведомлений (deleteGroupEmail)

Для удаления группы уведомлений необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteGroupEmail
- Content-Type: application/json
- Request body: Stirng, идентификатор существующей группы уведомлений (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.6.10. Получение полного списка групп уведомлений (listGroupEmail)

Для получения полного списка групп уведомлений сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listGroupEmail

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о группе уведомлений:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении группы уведомлений;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.11. Создание шаблона уведомления (createTemplateEmail)

Для создания шаблона уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createTemplateEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор шаблона уведомления.	Уникальное значение.

title	String	Да	-	Название шаблона уведомления.	-
content	String	Да	-	Шаблон для рендеринга уведомлений в виде html разметки.	-
version	Int	Нет	1	Версия группы уведомлений.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного шаблона уведомления в виде строки.

Данные созданного шаблона уведомления можно получить с помощью команд «getTemplateEmail» и «listTemplateEmail», описанных в данной инструкции.

2.6.12. Получение данных шаблона уведомления (getTemplateEmail)

Для получения данных шаблона уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getTemplateEmail
- Content-Type: application/json
- Request body: String, идентификатор существующего шаблона уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении шаблона;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.13. Изменение данные шаблона уведомления (updateTemplateEmail)

Для изменения данных шаблона уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateTemplateEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор шаблона уведомления.	Существующий в сервисе идентификатор.
title	String	Да	-	Название шаблона уведомления.	-
content	String	Да	-	Шаблон для рендеринга уведомлений в виде html разметки.	-
version	Int	Да	-	Версия, полученная при запросе данных шаблона уведомления. Нужна для оптимистичных блокировок.	Совпадает с текущей версией шаблона уведомления.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененного шаблона уведомлений можно получить с помощью команд «getTemplateEmail» и «listTemplateEmail», описанных в данной инструкции.

2.6.14. Удаление шаблона уведомления (deleteTemplateEmail)

Для удаления шаблона уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteTemplateEmail
- Content-Type: application/json
- Request body: String, идентификатор существующего шаблона уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.6.15. Получение полного списка шаблонов уведомлений (listTemplateEmail)

Для получения полного списка шаблонов уведомлений сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listTemplateEmail

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о шаблоне уведомлений:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении шаблона уведомлений;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.16. Создание рассылки уведомления (createDeliveryEmail)

Для создания рассылки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createDeliveryEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор рассылки.	Уникальное значение.
schedule	String	Да	-	Расписание отправления рассылки.	Строка для cron в формате {{minutes}} {{hours}}.

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной рассылки уведомления в виде строки.

Данные созданной рассылки уведомления можно получить с помощью команд «getDeliveryEmail» или «listDeliveryEmail», описанных в данной инструкции.

2.6.17. Получение данных рассылки уведомления (getDeliveryEmail)

Для получения данных рассылки уведомления необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/getDeliveryEmail
- Content-Type: application/json
- Request body: String, идентификатор существующей рассылки уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении рассылки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.18. Изменение рассылки уведомления (updateDeliveryEmail)

Для изменения данных рассылки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateDeliveryEmail
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор рассылки.	Уникальное значение.
schedule	String	Да	-	Расписание отправления рассылки.	Строка для cron в формате {{minutes}} {{hours}}
version	Long	Да	-	Версия, полученная при запросе данных версии рассылки. Нужна для оптимистичных блокировок.	Совпадает с текущей версией рассылки.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной рассылки можно получить с помощью команд «getDeliveryEmail» или «listDeliveryEmail», описанных в данной инструкции.

2.6.19. Удаление рассылки уведомления (deleteDeliveryEmail)

Для удаления рассылки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteDeliveryEmail
- Content-Type: application/json
- Request body: String, идентификатор существующей рассылки уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.6.20. Получение полного списка рассылок уведомлений (listDeliveryEmail)

Для получения полного списка рассылок уведомлений сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listDeliveryEmail

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о рассылке уведомлений:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении рассылки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.6.21. Создание шаблона отправки уведомления (createDeliveryTemplate)

Для создания шаблона отправки уведомления необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createDeliveryTemplate
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения

templateId	String	Да	-	Строковый идентификатор шаблона уведомления.	Существующий в сервисе идентификатор.
deliveryId	String	Да	-	Строковый идентификатор рассылки уведомления.	Существующий в сервисе идентификатор.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное создание.

Данные созданного шаблона отправки уведомления можно получить с помощью команды «deliveryTemplatesEmail», описанной в данной инструкции.

2.6.22. Получение списка шаблонов уведомлений по указанной рассылке (deliveryTemplatesEmail)

Для получения списка шаблонов уведомлений, привязанных к указанной рассылке (т.е. для указанной рассылки создавался шаблон отправки уведомления) необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deliveryTemplatesEmail
- Content-Type: application/json
- Request body: String, идентификатор существующей рассылки уведомления (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о шаблонах уведомлений, которые привязаны к указанной рассылке:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении шаблона;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7. Buzzer-personal-page: сервис уведомлений в личном кабинете

Сервис для настройки и сбора уведомлений для отображения в личном кабинете пользователя.

При помощи команд реализованы возможности:

- создание, изменение, удаление и получение списка разделов ЛК для уведомлений;

- создание, изменение, удаление и получение списка настройки уведомлений в ЛК;
- создание, изменение, удаление и получение списка шаблонов уведомлений в ЛК;
- получение списка уведомлений и изменение их статуса.

2.7.1. Создание раздела ЛК для уведомлений (createSectionPp)

Для создания раздела необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createSectionPp
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор раздела.	Уникальное значение.
name	String	Да	-	Название раздела.	-
version	Int	Нет	1	Версия раздела. Нужна для оптимистичных блокировок.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного раздела в виде строки.

Данные созданного раздела можно получить с помощью команд «getSectionPp» или «listSectionPp», описанных в данной инструкции.

2.7.2. Получение данных раздела ЛК для уведомлений (getSectionPp)

Для получения данных раздела необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getSectionPp
- Content-Type: application/json
- Request body: String, идентификатор существующего раздела (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении раздела;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7.3. Изменение раздела ЛК для уведомлений (updateSectionPp)

Для изменения данных раздела необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateSectionPp
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор раздела.	Существующий в сервисе идентификатор.
name	String	Да	-	Название раздела.	-
version	Int	Да	-	Версия, полученная при запросе данных раздела. Нужна для оптимистичных блокировок.	Совпадает с текущей версией раздела.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененного раздела можно получить с помощью команд «getSectionPp» или «listSectionPp», описанных в данной инструкции.

2.7.4. Удаление раздела ЛК для уведомлений (deleteSectionPp)

Для удаления раздела необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteSectionPp

- Content-Type: application/json
- Request body: String, идентификатор существующего раздела (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.7.5. Получение полного списка разделов ЛК для уведомлений (listSectionPp)

Для получения полного списка разделов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listSectionPp

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о справочнике:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении раздела;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7.6. Создание настройки уведомления в ЛК (createSettingPp)

Для создания настройки уведомления в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createSettingPp
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор настройки уведомления.	Существующий код уведомления.
sectionId	String	Да	-	Идентификатор раздела.	Существующий идентификатор.
version	Int	Нет	1	Версия настройки уведомления в ЛК. Нужна для оптимистичных	-

				блокировок.	

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной настройки уведомления в ЛК в виде строки.

Данные созданной настройки можно получить с помощью команд «getSettingPp» или «listSettingPp», описанных в данной инструкции.

2.7.7. Получение данных настройки уведомления в ЛК (getSettingPp)

Для получения данных справочника необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getSettingPp
- Content-Type: application/json
- Request body: String, код существующей настройки уведомления в ЛК (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении настройки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7.8. Изменение настройки уведомления в ЛК (updateSettingPp)

Для изменения данных настройки уведомления в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateSettingPp
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор настройки уведомления.	Существующий в сервисе код уведомления.

sectionId	String	Да	-	Идентификатор раздела.	Существующий в сервисе идентификатор.
version	Int	Да	-	Версия, полученная при запросе данных настройки. Нужна для оптимистичных блокировок.	Совпадает с текущей версией настройки.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной настройки уведомлений в ЛК можно получить с помощью команд «getSettingPp» или «listSettingPp», описанных в данной инструкции.

2.7.9. Удаление настройки уведомлений в ЛК (deleteSettingPp)

Для удаления настройки уведомлений в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteSettingPp
- Content-Type: application/json
- Request body: String, код существующей настройки уведомления в ЛК (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.7.10. Получение полного списка настроек уведомлений в ЛК (listSettingPp)

Для получения полного списка настроек уведомлений в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listSettingPp

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о настройке:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении настройки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7.11. Создание шаблона для настройки уведомления в ЛК (createTemplatePp)

Для создания шаблона для настройки уведомления в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createTemplatePp
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор шаблона.	Уникальное значение.
settingId	String	Да	-	Строковый идентификатор настройки уведомления в ЛК.	Существующий в сервисе идентификатор настройки.
title	String	Да	-	Заголовок.	-
content	String	Да	-	Шаблон рендеринга уведомления.	-
version	Int	Нет	1	Версия настройки шаблона. Нужна для оптимистичных блокировок.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного шаблона в виде строки.

Данные созданного шаблона можно получить с помощью команд «getTemplatePp» или «listTemplatePp», описанных в данной инструкции.

2.7.12. Получение данных шаблона для настройки уведомления в ЛК (getTemplatePp)

Для получения данных шаблона для настройки уведомления в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getTemplatePp
- Content-Type: application/json

- Request body: String, идентификатор существующего шаблона (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении шаблона;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7.13. Изменение шаблона для настройки уведомления в ЛК (updateTemplatePp)

Для изменения шаблона для настройки уведомления в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateTemplatePp
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор шаблона.	Существующий в сервисе идентификатор шаблона.
settingId	String	Да	-	Строковый идентификатор настройки уведомления в ЛК.	Существующий в сервисе идентификатор настройки.
title	String	Да	-	Заголовок.	-
content	String	Да	-	Шаблон рендеринга уведомления.	-
version	Int	Да	-	Версия, полученная при запросе данных шаблона. Нужна для оптимистичных блокировок.	Совпадает с текущей версией шаблона.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененного шаблона можно получить с помощью команд «getTemplatePp» или «listTemplatePp», описанных в данной инструкции.

2.7.14. Удаление шаблона для настройки уведомления в ЛК (deleteTemplatePp)

Для удаления справочника и всех его элементов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteTemplatePp
- Content-Type: application/json
- Request body: String, идентификатор существующего шаблона (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на успешное удаление.

2.7.15. Получение полного списка шаблонов для настройки уведомлений в ЛК (listTemplatePp)

Для получения полного списка шаблонов настройки уведомлений в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listTemplatePp

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о шаблоне:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении шаблона;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.7.16. Получение списка уведомлений в ЛК текущего пользователя с поддержкой фильтрации (searchPPNotifications)

Для получения списка уведомлений в ЛК пользователя необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/searchPPNotifications
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения

search	Search	Да	-	Параметры сортировки, фильтрации и пейджинга списочного запроса.	-
markRead	Boolean	Нет	false	Признак необходимости отмечать найденные уведомления как прочитанные.	-
markRead	Boolean	Нет	false	Признак необходимости рендеринга уведомлений.	-

Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:

- Доступные поля для сортировки:
    a. created;
    b. isRead.
- Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
sectionId	InSetQuery
isRead	InSetQuery
id	InSetQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи уведомлений пользователя в ЛК, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными об уведомлении:

- атрибуты, заполненные при отправлении уведомления (например, код уведомления, идентификатор пользователя и данные уведомления);
- системные атрибуты: данные, сгенерированные сервисом автоматически (например, идентификатор, дата сохранения, отметка о прочтении и рендер уведомления).

2.7.17. Получение списка уведомлений в ЛК любого пользователя с поддержкой

фильтрации (auditSearchPPNotifications)

Для получения списка уведомлений в ЛК пользователя необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/auditSearchPPNotifications
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
search	Search	Да	-	Параметры сортировки, фильтрации и пейджинга списочного запроса.	-
markRead	Boolean	Нет	false	Признак необходимости отмечать найденные уведомления как прочитанные.	-
markRead	Boolean	Нет	false	Признак необходимости рендеринга уведомлений.	-

Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:

- Доступные поля для сортировки:
  a. created;
  b. isRead.
- Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
userId	InSetQuery
sectionId	InSetQuery
isRead	InSetQuery
id	InSetQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи уведомлений в ЛК пользователей Системы, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными об уведомлении:

- атрибуты, заполненные при отправлении уведомления (например, код уведомления, идентификатор пользователя и данные уведомления);
- системные атрибуты: данные, сгенерированные сервисом автоматически (например, идентификатор, дата сохранения, отметка о прочтении и рендер уведомления).

2.7.18. Изменение статуса уведомлений в ЛК (changeNotificationStatus)

Для изменения статуса уведомлений в ЛК необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/changeNotificationStatus
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
ids	List[UUID]	Да	-	Список идентификаторов уведомлений.	Существующие в сервисе идентификаторы.
isRead	Boolean	Да	-	Статус уведомления, который необходимо присвоить уведомлению: ● true - «Прочитано»; ● false - «Не прочитано».	-
allNotifications	Boolean	Да	-	Логика изменения статуса уведомлений: ● true - изменить статус у всех уведомлений, кроме	-

				• указанных; • false - изменить статус только у указанных уведомлений. .	

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

Данные уведомлений можно получить с помощью команд «searchPPNotifications» или «auditSearchPPNotifications», описанных в данной инструкции.

2.8. Data-model: сервис модели данных

Сервис настройки модели данных Системы: настройка типов сущностей и их полей, а также отношений между объектами сущностей. В сервисе хранятся данные объектов сущностей и информация о связях и отношениях между ними.

При помощи команд реализованы возможности:

- обновление модели данных;
- получение списка типов сущностей, которые составляют модель данных;
- создание, изменение, удаление объектов и получение списка объектов;
- создание, удаление отношений и получение списка отношений;
- создание, удаление связей и получение списка связей.

2.8.1. Обновление модели данных (createDataModel)

Для обновления модели данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createDataModel
- Content-Type: application/json
- Request body: Список JSON-объектов одного из поддерживаемых типов операций
- Поддерживаемые типы операций:
- Тип данных **CreateEntity** - структура данных сервиса модели данных, которая используются для создания новой сущности:

Поле	Тип	Обязательн	Значение по	Описание	Ограничения

		ое	умолчанию		
operation	String	Да	-	Наименование операции.	Одно поддерживаемое значение - CreateEntity.
entityType	String	Да	-	Строковый код типа сущности.	-
title	String	Да	-	Человекочитаемое название типа сущности.	-
description	String	Нет	-	Описание типа сущности.	-
settings	JSON	Нет	-	Произвольные настройки типа сущности.	-

- Тип данных **UpdateEntity** - структура данных сервиса модели данных, которая используются для изменения сущности:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
operation	String	Да	-	Наименование операции.	Одно поддерживаемое значение - UpdateEntity.
entityType	String	Да	-	Строковый код типа сущности.	Существующий код типа сущности в сервисе.
title	String	Да	-	Человекочитаемое название типа сущности.	-
description	String	Нет	-	Описание типа сущности.	-
settings	JSON	Нет	-	Произвольные настройки типа сущности.	-

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное обновление модели данных сервиса.

Данные сущностей модели данных можно получить с помощью команды «listDataModels», описанной в данной инструкции.

2.8.2. Получение полного списка сущностей модели данных (listDataModels)

Для получения полного списка справочников сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listDataModels

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о сущностях:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении сущности;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор или описание)*.

2.8.3. Получение примера объекта сущности (getExampleObject)

Для получения примера объекта сущности необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getExampleObject
- Content-Type: application/json
- Request body: String, код существующего типа сущности (атрибут «entityType» при создании, «id» при получении списка)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибут «data» - JSON-объект, в котором:
    a. ключи - коды полей, которые были указаны при добавлении поля в сущность модели данных
    b. значения - примеры данных для соответствующего поля, которые были указаны при добавлении поля в сущность модели данных;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.8.4. Создание объекта сущности (saveObject)

Для создания объекта сущности модели данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/saveObject

- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код сущности.	Существующий код в сервисе.
source	String	Да	-	Источник данных для объекта.	-
attributes	JSON	Да	-	Объект JSON, структура которого соответствует списку полей сущности.	Указаны все обязательные поля сущности. Значение полей соответствует всем ограничениям для поля сущности.

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного объекта в виде строки формата UUID.

Данные созданного объекта можно получить с помощью команды «listObjects», описанной в данной инструкции.

2.8.5. Изменение объекта сущности (updateObject)

Для изменения объекта сущности модели данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/updateObject
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код сущности.	Существующий код в сервисе.
objectId	UUID	Да	-	ID объекта.	Существующий в сервисе идентификатор.

attributes	JSON	Да	-	Объект JSON, структура которого соответствует списку полей сущности.	Указаны все обязательные поля сущности. Значение полей соответствует всем ограничениям для поля сущности.
version	Int	Да	-	Версия, полученная при запросе данных объекта. Нужна для оптимистичных блокировок.	Совпадает с текущей версией объекта.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное изменение.

Данные измененного объекта можно получить с помощью команды «listObjects», описанной в данной инструкции.

2.8.6. Удаление объекта сущности (deleteObject)

Для удаления объекта сущности модели данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteObject
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код сущности.	Существующий код в сервисе.
objectId	UUID	Да	-	ID объекта.	Существующий в сервисе идентификатор.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.8.7. Получение списка объектов сущности с поддержкой фильтрации (listObjects)

Для получения списка объектов сущности необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listObjects
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код типа сущности.	Существующий в сервисе код сущности.
search	Search	Да	-	Параметры сортировки, фильтрации и пейджинга списочного запроса.	-

Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:

- Доступные поля для сортировки:
    a. id;
    b. externalid;
    c. title;
    d. modified;
    e. created.
- Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id или objectid	InSetQuery
externalid	InSetQuery
title	InSetQuery, LikeQuery
source	InSetQuery
created	InSetQuery, RangeQuery
modified	InSetQuery, RangeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная

страница содержит записи объектов указанного типа сущности, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными об объекте:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении объекта;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.8.8. Сохранение отношения между объектами (saveRelation)

Для сохранения отношения между объектами необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/saveRelation
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entity1Type	String	Да	-	Строковый код типа сущности первого объекта.	Существующий в сервисе код сущности.
object1Id	UUID	Да	-	Идентификатор первого объекта.	Существующий в сервисе идентификатор объекта.
entity2Type	String	Да	-	Строковый код типа сущности второго объекта.	Существующий в сервисе код сущности.
object2Id	UUID	Да	-	Идентификатор второго объекта.	Существующий в сервисе идентификатор объекта.
entity1Field	String	Да	-	Виртуальное поле отношения из первой сущности.	-
source	String	Да	-	Источник данных для отношений. Для введенных вручную - manual.	-

При успешном выполнении команды в теле ответа поле «value» значение «true», указывающее на успешное сохранение.

Данные отношений объекта можно получить с помощью команды «listAllRelations», описанной в данной инструкции.

2.8.9. Удаление всех отношений объекта с объектами перечисленных сущностей (deleteRelations)

Для удаления всех отношений объекта со всеми объектами перечисленных типов сущностей необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteRelations
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код сущности.	Существующий код в сервисе.
objectId	UUID	Да	-	ID объекта.	Существующий в сервисе идентификатор.
relatedEntityTypes	List[String]	Да	-	Перечень кодов сущностей, отношение с объектами которых будет удалено.	Существующие коды в сервисе.

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными об удаленных отношениях:

- атрибуты, переданные пользователем: все данные, указанные при сохранении отношения;
- системные атрибуты: данные, сгенерированные сервисом автоматически (например, идентификатор).

2.8.10. Получение всех отношений объекта (listAllRelations)

Для получения всех отношений объекта необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/listAllRelations
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код сущности.	Существующий код в сервисе.
objectId	UUID	Да	-	ID объекта.	Существующий в сервисе идентификатор.

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о всех отношениях объекта:

- атрибуты, переданные пользователем: все данные, указанные при сохранении отношения;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.8.11. Создание связи между объектами (createObjectLink)

Для создания связи между объектами необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createObjectLink
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entity1Type	String	Да	-	Строковый код типа сущности первого объекта.	Существующий в сервисе код сущности.
object1Id	UUID	Да	-	Идентификатор первого объекта.	Существующий в сервисе идентификатор объекта.

entity2Type	String	Да	-	Строковый код типа сущности второго объекта.	Существующий в сервисе код сущности.
object2Id	UUID	Да	-	Идентификатор второго объекта.	Существующий в сервисе идентификатор объекта.
linkType	String	Да	-	Тип связи, строковый код.	-
source	String	Да	-	Источник данных для связи. Для введенных вручную - manual.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной связи в виде строки формата UUID.

Данные созданной связи можно получить с помощью команды «getObjectLinks», описанной в данной инструкции.

2.8.12. Удаление связи между объектами (deleteObjectLink)

Для удаления связи между объектами необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteObjectLink
- Content-Type: application/json
- Request body: UUID, идентификатор существующей связи между объектами (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.8.13. Получение всех связей объекта (getObjectLinks)

Для получения всех связей объекта необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getObjectLinks
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Строковый код сущности.	Существующий код в сервисе.
objectId	UUID	Да	-	ID объекта.	Существующий в сервисе идентификатор.
forwardLinks	Boolean	Да	-	Признак получения прямых или обратных связей. Если true, тогда связи прямые, тогда искомый объект - это первый объект из связи (object1). Если false, тогда искомый объект - это второй объект из связи (object2)	-

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о всех связях объекта:

- атрибуты, переданные пользователем: все данные, указанные при создании связи;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.9. Erika: сервис печатных форм

Сервис для обработки файлов с целью их конвертации или формирования отчета по заданному шаблону на основе данных.

При помощи команд реализованы возможности:

- конвертация в PDF;
- извлечение переменных из файла шаблона;
- генерация отчета путем замены переменных внутри файла с шаблоном на соответствующие данные.

2.9.1. Конвертация файла в PDF (convertToPdf)

Для конвертации файла в PDF необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/convertToPdf

- Content-Type: application/json
- Request body: URL файла для конвертации в файловом хранилище. Поддерживаются файлы следующих расширений: doc, docx, txt.

При успешном выполнении команды в теле ответа поле «value» содержит URL файла в формате PDF в файловом хранилище.

2.9.2. Извлечение переменных из файла (extractVars)

Для извлечения переменных из файла необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/extractVars
- Content-Type: application/json
- Request body: URL файла для извлечения переменных в файловом хранилище. Поддерживаются файлы следующих расширений: doc, docx, txt.

При успешном выполнении команды в теле ответа поле «value» содержит строку с сериализованным в JSON списком переменных из файла.

2.9.3. Создание отчета (createReport)

Для создания отчета необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createReport
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
mode	String	Да	-	Режим генерации отчета.	Одно поддерживаемое значение - Single.
name	String	Нет	Согласно наименованию файла, указанного в templateFile.	Имя файла с отчетом. Расширение (и имя, если не указано) подставляется от шаблона.	-
templateFi	String	Да	-	URL файла шаблона	-

le				в файловом хранилище. Поддерживаются файлы следующих расширений: doc, docx, txt.	
json	JSON	Да	-	Произвольные данные для подстановки в отчет вместо соответствующих переменных в шаблоне.	-

При успешном выполнении команды в теле ответа поле «value» содержит URL файла в файловом хранилище. Содержимое файла аналогично файлу шаблону, но в нем заменены переменные согласно указанным данным.

2.10. Indexation: сервис индексаций

Сервис для управления процессом индексации данных, хранимых в сервисе модели данных, с целью их обработки внешними сервисами или плагинами.

При помощи команд реализованы возможности:

- создание, получение и удаление описания индексаций;
- запуск индексации согласно ее описанию, просмотр списка уже запущенных индексаций и статистики по ним.

2.10.1. Создание описания индексации (createIndexationDescription)

Для создания описания индексации необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createIndexationDescription
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор описания	Существующий код типа сущности

				индексации.	сервиса модели данных. Уникальное значение.
steps	List[Step]	Да	-	Список шагов описания индексации.	-

Тип данных **Step** - структура данных сервиса индексаций, которая используются для описания шагов в описании индексации.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
children	List[Step]	Да	-	Зависящие шаги. Не будут запущены, если не пройден текущий шаг.	-
payload	StepPayload	Да	-	Описание шага	-
label	String	Нет	-	Произвольный лейбл шага. Может быть полезен при отладке.	-

Тип данных **StepPayload** - структура данных сервиса индексаций, которая используются для описания логики шага индексации. Структура зависит от типа индексации: Lamp, Plugin.

Тип данных **Lamp** - структура данных сервиса индексаций, для описания типа индексации в сервисах LAMP всех объектов индексируемой сущности. Может фиксировать отдельные результаты в сервисе модели данных. Этот шаг для каждого объекта индексируемой сущности и каждого i-го элемента массива, указанного в «fileRoutes», извлекает из поля «filesRoutes[i].field» информацию о файле, с помощью этой информации достает файл из хранилища, и отправляет запросы с содержимым этого файла к рутам Lamp, согласно «fileRoutes.actions.lampRoute».

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
filesRoutes	List[FileRoutes]	Да	-	Список файловых полей сущности для индексации.	-

Тип данных **FileRoutes** - структура данных сервиса индексаций, которая используется для описания логики индексации отдельно взятого файлового поля сущности в сервисах LAMP.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
field	String	Да	-	Имя поля, где хранится файл, для которого будет запущена индексация.	-
saveContentTo	String	Нет		Ключ для сохранения результата работы «extraction в data-model в поле «data.[значение field].additional	
actions	List[FileRoute]			Список конкретных индексаций в LAMP.	

Тип данных **FileRoute** - структура данных сервиса индексаций, которая используется для описания логики индексации в сервисах LAMP.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
lampRoute	String	Да	-	Lamp роут.	Одно из доступных значений: Extraction, Keyword, Plagiarism, Similarity, ExtractionPdfium
routeResult	RouteResult	Нет	-	Обработка результатов от Lamp.	-

Тип данных **RouteResult** - структура данных сервиса индексаций, которая используется для описания логики сохранения результата индексации в сервисах LAMP.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
resultName	String	Да	-	Ключ для фиксации результата, по которому следующие шаги могу получить этот результат.	Одно из доступных значений: Extraction, Keyword, Plagiarism, Similarity,

					ExtractionPdfium
saveToDM	Boolean	Нет	false	Флаг, указывающий нужно ли сохранить результат в сервис модели данных.	-

Тип данных **Plugin** - структура данных сервиса индексаций, для описания типа индексации в плагинах интеллектуализации. Доступные на данный момент:

- «plagiarismPlugin» - построение ссылок между объектами на основе содержательных пересечений;
- «similarityPlugin» - построение ссылок между объектами близкими по смыслу;
- «taggingPlugin» - назначение тэгов файлам на основе их расширения.

Для «plagiarismPlugin» и «similarityPlugin» шаг должен быть запущен после обработки всех передаваемых «fields» с помощью шага Lamp (в шаге Lamp обрабатываемое поле указывается в «filesRoutes.field»). В шаге Lamp в «filesRoutes.actions» должны присутствовать действия «Plagiarism» и «Similarity» соответственно.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
topic	String	Да	-	Ключ для фиксации результата, по которому следующие шаги могу получить этот результат.	Кафка топик целевого плагина, по которому произойдет связь с конкретным плагином. Топик необходимого плагина хранится в сервисе технического справочника в константе с соответствующим кодом.
fields	List[String]	Да	-	Список файловых полей для индексации.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданного описания индексации, равный указанному значению в поле «id».

Данные созданного описания индексации можно получить с помощью команды «getIndexationDescription», описанной в данной инструкции.

2.10.2. Получение описания индексации (getIndexationDescription)

Для получения описания индексации необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getIndexationDescription
- Content-Type: application/json
- Request body: String, идентификатор существующего справочника (атрибут «code»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании описания индексации;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

2.10.3. Удаление описания индексации (deleteIndexationDescription)

Для удаления описания индексации необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/deleteIndexationDescription
- Content-Type: application/json
- Request body: String, идентификатор существующего описания индексации(атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «1», указывающее на успешное удаление.

2.10.4. Запуск индексации (startIndexation)

Для удаления описания индексации необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/startIndexation
- Content-Type: application/json

- Request body: String, идентификатор существующего описания индексации (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор запущенной индексации в виде строки формата UUID.

2.10.5. Получение статистики по индексации (indexationStatistics)

Для получения статистики по индексации необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/indexationStatistics
- Content-Type: application/json
- Request body: UUID, идентификатор существующей индексации (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- статус индексации;
- список объектов, по которым все шаги были успешно выполнены;
- сводная информация об ошибках в формате «Тип шага:кол-во провалившихся шагов»;
- сводная информация об ошибках, в формате «идентификатор объекта:типы провалившихся шагов».

2.10.6. Получение полного списка индексаций (getIndexations)

Для получения полного индексаций сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/getIndexations

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными об индексации:

- атрибут, содержащий полное описание индексации на момент ее запуска;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.11. Integration: сервис интеграций

Сервис для регулирования доступов клиентов к внешнему API для импорта данных в сервис модели данных.

При помощи команд реализованы возможности:

- регистрация клиентов и их просмотр;
- обновление списка доступных типов данных сервиса модели данных и получение этого списка;
- выдача прав клиентам на работу с определенным типом данных.

2.11.1. Регистрация клиента (integration_CreateClient)

Для регистрации клиента необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/integration_CreateClient
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Код клиента.	-
title	String	Да	-	Наименование клиента.	-
allowFiles	Boolean	Да	-	Признак разрешения загрузки файлов клиентом.	-
metadata	JSON	Да	-	Дополнительные данные клиента в свободном формате.	-

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешную регистрацию.

Данные зарегистрированного клиента можно получить с помощью команды «integration_ListClients», описанной в данной инструкции.

2.11.2. Получение полного списка клиентов (integration_ListClients)

Для получения полного списка клиентов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/integration_ListClients

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о клиенте:

- атрибуты, переданные пользователем: все данные, указанные при регистрации клиента;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор и токен для доступа внешнего API)*.

2.11.3. Обновление типов данных согласно сервису модели данных (integration_RefreshEntityTypes)

Для обновления списка типов данных для интеграции согласно актуальному списку типов данных сервиса модели данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/integration_RefreshEntityTypes

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное обновление.

Данные обновленного списка типов данных можно получить с помощью команды «integration_ListEntityTypes», описанной в данной инструкции.

2.11.4. Получение полного списка доступных типов данных (integration_ListEntityTypes)

Для получения полного списка доступных для интеграции типов данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/integration_ListEntityTypes

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов с атрибутами о типе данных, которые были переданы пользователем: все данные, указанные при создании или изменении типа данных сервиса модели данных.

2.11.5. Предоставление клиенту доступа к типу данных (integration_AllowEntityType)

Для предоставления клиенту доступа к типу данных необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/integration_AllowEntityType
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
clientId	String	Да	-	Строковый код клиента.	Существующий код зарегистрированного клиента.
entityTypeId	String	Да	-	Строковый код типа данных.	Существующий код типа данных.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное предоставление доступа.

Данные о доступных для клиента типах данных можно получить с помощью команды «integration_ListEntityTypesForClient», описанной в данной инструкции.

2.11.6. Получение полного списка типов данных доступных для клиента (integration_ListEntityTypesForClient)

Для получения полного списка типов данных доступных для клиента необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/integration_ListEntityTypesForClient

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов с атрибутами доступных для клиента типов данных, которые были переданы пользователем: все данные, указанные при создании или изменении типа данных сервиса модели данных.

2.12. Mon: сервис аутентификации

Сервис для хранения информации о пользователях, объединения их в группы, а также их регистрации и аутентификации.

При помощи команд реализованы возможности:

- регистрации пользователя;
- изменение статуса пользователя;
- аутентификация на основе логина и пароля;
- создание, удаление и получение групп пользователей;
- добавление и удаления пользователей из групп;
- получение и изменение данных пользователей.

2.12.1. Регистрация пользователя (mon_http_RegisterUser)

Для регистрации пользователя необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_RegisterUser
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
email	String	Да	-	Email пользователя.	-
password	String	Да	-	Пароль, который будет использован для аутентификации.	-
info	UserRegistrationInfo	Да	-	Дополнительная информация о пользователе	-

Тип данных **UserRegistrationInfo** - структура данных сервиса аутентификации, которая используются для описания дополнительных данных пользователя, указывающихся при регистрации.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
firstName	String	Да	-	Имя пользователя.	-
lastName	String	Да	-	Фамилия пользователя.	-
middleName	String	Нет	-	Отчество пользователя.	-

data	JSON	Нет	-	Дополнительные данные о пользователе - объект JSON.	-

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со сгенерированным идентификатором пользователя и информацией о том, было ли отправлено письмо.

Данные зарегистрированного пользователя можно получить с помощью команды «mon_http_ListUsers», описанной в данной инструкции.

2.12.2. Изменение статуса пользователя (mon_http_ChangeUserStatus)

Для изменения статуса пользователя необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_ChangeUserStatus
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
userId	UUID	Да	-	Идентификатор пользователя	Существующий в сервисе идентификатор.
status	String	Да	-	Необходимый статус пользователя.	Одно из списка допустимых значений: Pending, Activated, Deactivated.

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными о пользователе:

- атрибуты, переданные пользователем: все данные, указанные при регистрации или изменении пользователя;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.12.3. Изменение данных пользователя (mon_http_UpdateUserInfo)

Для изменения статуса пользователя необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_UpdateUserInfo
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	UUID	Да	-	Идентификатор пользователя	Существующий в сервисе идентификатор.
info	UserInfo	Да	-	Информация о пользователе.	

Тип данных **UserInfo** - структура данных сервиса аутентификации, которая используются для описания дополнительных данных пользователя.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
firstName	String	Нет	-	Имя пользователя.	-
lastName	String	Нет	-	Фамилия пользователя.	-
middleName	String	Нет	-	Отчество пользователя.	-
data	JSON	Нет	-	Дополнительные данные о пользователе - объект JSON.	-

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными о пользователе:

- атрибуты, переданные пользователем: все данные, указанные при регистрации или изменении пользователя;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.12.4. Получение списка пользователей с поддержкой фильтрации (mon_http_ListUsers)

Для получения списка пользователей необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_ListUsers
- Content-Type: application/json
- Request body: отсутствует или имеет структуру Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. userId;
        ii. status;
        iii. email;
        iv. lastName;
        v. firstName;
        vi. middleName;
        vii. modified.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
userId	InSetQuery
status	InSetQuery
email	InSetQuery, LikeQuery
modified	InSetQuery, RangeQuery
lastName	InSetQuery, LikeQuery
firstName	InSetQuery, LikeQuery
middleName	InSetQuery, LikeQuery
fio	TsQuery
data	LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи пользователей, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными о пользователе:

- атрибуты, переданные пользователем: все данные, указанные при регистрации или изменении пользователя и его статуса;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.12.5. Создание группы пользователей (mon_http_CreateGroup)

Для создания группы пользователей необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_CreateGroup
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Строковый идентификатор группы.	Уникальное значение.
name	String	Да	-	Человекочитаемое название группы.	-
description	String	Нет	-	Описание группы.	-

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными о группе:

- атрибуты, переданные пользователем: все данные, указанные при создании группы;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, дата последней модификации)*.

2.12.6. Удаление группы пользователей (mon_http_DeleteGroup)

Для удаления группы пользователей необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_DeleteGroup
- Content-Type: application/json

- Request body: String, идентификатор существующей группы (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.12.7. Получение списка групп пользователей с поддержкой фильтрации (mon_http_ListGroups)

Для получения списка групп пользователей необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_ListGroups
- Content-Type: application/json
- Request body: отсутствует или имеет структуру Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. groupId;
        ii. name;
        iii. description;
        iv. modified;
        v. count.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
groupId	InSetQuery, LikeQuery
name	InSetQuery, LikeQuery
description	InSetQuery, LikeQuery
modified	InSetQuery, LikeQuery, RangeQuery
count	InSetQuery, LikeQuery, RangeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи групп пользователей, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными о группе:

- атрибуты, переданные пользователем: все данные, указанные при создании группы пользователей;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, дата последней модификации)*.

2.12.8. Добавление пользователей в группы (mon_http_AddUsersToGroup)

Для добавления пользователей в группы необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_AddUsersToGroup
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
groupId	List[String]	Да	-	Список идентификаторов групп пользователей.	Существующие в сервисе идентификаторы.
users	List[UUID]	Да	-	Список идентификаторов пользователей.	Существующие в сервисе идентификаторы.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное добавление пользователей в группы.

Данные пользователей и групп, в которые они добавлены, можно получить с помощью команды «mon_http_ListUsersWithGroups», описанной в данной инструкции.

2.12.9. Удаление пользователей из групп (mon_http_RemoveUsersFromGroup)

Для удаления пользователей из групп необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_RemoveUsersFromGroup
- Content-Type: application/json

- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
groupId	List[String]	Да	-	Список идентификаторов групп пользователей.	Существующие в сервисе идентификаторы.
users	List[UUID]	Да	-	Список идентификаторов пользователей.	Существующие в сервисе идентификаторы.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление пользователей из групп.

Данные пользователей и групп, в которых они состоят, можно получить с помощью команды «mon_http_ListUsersWithGroups», описанной в данной инструкции.

2.12.10. Получение списка пользователей с информацией о группах с поддержкой фильтрации (mon_http_ListUsersWithGroups)

Для получения списка пользователей с информацией о группах, в которые они добавлены, необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_ListUsersWithGroups
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. userId;
        ii. status;
        iii. email;
        iv. lastName;
        v. firstName;
        vi. middleName;
        vii. count.

b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
userId	InSetQuery
status	InSetQuery
email	InSetQuery, LikeQuery
modified	InSetQuery, RangeQuery
lastName	InSetQuery, LikeQuery
firstName	InSetQuery, LikeQuery
middleName	InSetQuery, LikeQuery
fio	TsQuery
data	LikeQuery
count	InSetQuery, LikeQuery, RangeQuery
groupId	InSetQuery, AllInSetQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи о пользователях и их группах, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными о пользователе и его группах:

- атрибуты, переданные пользователем: все данные, указанные при регистрации/создании или изменении пользователя или группы;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.12.11. Аутентификация по логину и паролю (mon_http_LoginByCredentials)

Для аутентификации по логину и паролю необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_LoginByCredentials
- Content-Type: application/json

- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
email	String	Да	-	Email пользователя.	Email существующего в сервисе пользователя со статусом «Activated».
password	String	Да	-	Пароль пользователя.	Корректный пароль пользователя, email которого указан в поле «email».

При успешном выполнении команды в теле ответа поле «value» содержит значение «null» и в заголовках ответа вернулось значение в формате «Set-Cookie:jwt_token=<значение>; Expires=<значение>; Path=/; HttpOnly», указывающее на успешную аутентификацию.

Данные аутентифицированного пользователя можно получить с помощью команды «mon_http_MyData», описанной в данной инструкции.

2.12.12. Получение данных пользователя на основе JWT (mon_http_MyData)

Для получения данных пользователя на основе JWT необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/mon_http_MyData
- Request Headers: Cookie: jwt_token=<значение>, где <значение> - JWT, полученный в результате прохождения аутентификации («mon_http_LoginByCredentials»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными о пользователе, согласно JWT:

- атрибуты, переданные пользователем: все данные, указанные при регистрации или изменении пользователя;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.13. Nestor: сервис журналирования

Сервис для фиксирования, хранения и предоставления доступа к информации о совершенных пользователями действиях, влияющих на данные в других сервисах.

При помощи команд реализованы возможности:

- создание события;
- просмотр списка событий.

2.13.1. Создание события (createEvent)

Для создания события необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/createEvent
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
eventDate	Long	Да	-	Дата события. Ожидается в виде Timestamp.	-
eventType	String	Да	-	Тип события.	-
userId	UUID	Да	-	Идентификатор связанного с событием пользователя. В т.ч. допустимо значение в виде пустой строки - автоматически будет сконвертировано в идентификатор анонимного пользователя ("00000000-0000-0000-0000-000000000000").	-
eventCategory	String	Да	-	Статус события.	-

application	String	Да	-	Идентификатор связанного с событием приложения.	-
data	String	Да	-	Данные события.	-
dependencies	List[Dependency]	Да	-	Иные связанные с событием сущности.	-

Тип данных **Dependency** - структура данных сервиса журналирования, которая используются для описания зависимости события с другими

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityClass	String	Да	-	Класс зависимости.	-
entityId	String	Да	-	Идентификатор зависимости.	-
dependencyType	String	Да	-	Тип зависимости.	-

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со значением «success» в поле «result», что свидетельствует об отсутствии ошибок создания.

Данные созданного события можно получить с помощью команды «searchEvents», описанной в данной инструкции.

2.13.2. Получение списка событий с поддержкой фильтрации (searchEvents)

Для получения списка событий необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/searchEvents
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. id;
        ii. eventDate;

iii. storeDate;
iv. eventType;
v. userId;
vi. eventCategory;
vii. application;
viii. dependencies.entityClass;
ix. dependencies.entityId;
x. dependencies.dependencyType.

b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id	InSetQuery
eventDate	InSetQuery, RangeQuery
storeDate	InSetQuery, RangeQuery
eventType	InSetQuery, LikeQuery
userId	InSetQuery, LikeQuery
eventCategory	InSetQuery, LikeQuery
application	InSetQuery, LikeQuery
data	InSetQuery, LikeQuery, TsQuery
renderedData	LikeQuery, TsQuery
dependencies.entityClass	InSetQuery, AllInSetQuery, LikeQuery
dependencies.entityId	InSetQuery, AllInSetQuery, LikeQuery
dependencies.dependencyType	InSetQuery, AllInSetQuery, LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи событий, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными о событии:

- атрибуты, переданные пользователем: все данные, указанные при создании события;

- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.14. Oberto: сервис системы прав

Сервис предназначен для создания и хранения настроек доступов к различным ресурсам, а также проверки доступности пользователей к ресурсам на основе этих настроек.

При помощи команд реализованы возможности:

- создание и удаление версий политик;
- получение списка политик доступа;
- проверка доступа.

2.14.1. Создание версии политики (oberto_UpdatePolicyV2)

Для создания версии политики необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/oberto_UpdatePolicyV2
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
policyId	String	Да	-	Идентификатор политики.	Уникальное значение.
version	String	Да	-	Версия политики согласно стандарту семантического версионирования.	Уникальное значение для указанного идентификатора политики. Значение соответствует регулярному выражению «\d{1,16}.\d{1,16}.\d{1,16}»
priority	Int	Нет	-	Приоритет политики: чем меньше, тем раньше она будет проверена. Без	-

				указания приоритета - будет проверяться после всех политик с приоритетом.	
description	String	Да	-	Описание политики.	-
rules	List[PolicyRuleDto]	Да	-	Список правил политики.	-

Тип данных **PolicyRuleDto** - структура данных сервиса системы прав, которая используются для описания правила политики.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
ruleId	String	Да	-	Идентификатор правила.	Уникальное значение в рамках политики.
rule	String	Да	-	Правило применения эффекта разрешения или запрета.	Соответствует формату описания правил, описанному ниже в данной инструкции.
priority	Int	Да	-	Приоритет правила: чем меньше, тем раньше оно будет проверено.	-

Формат описания правил представляет собой 3 части, разделенных пробелом:

- «Effect»: может содержать одно из двух ключевых слов: permin, deny
- «Actions»: содержит список действий, на которые будет распространяться правило. Список правил начинается со слова «Actions», заключен в скобки и разделен запятыми;
- «IfClause»: содержит условия для пользователя и целевого ресурса, при выполнении которых будет применяться правило. Условия начинаются со слова «if» и разделены на группы. Внутри каждой группы могут быть другие группы условий или само условие. Доступные операции для сравнения значений:
    a. «==» - два значения равны;
    b. «>=» - левое значение больше или равно правому;
    c. «<=» - левое значение меньше или равно правому;

d. «contains» - левое значение содержит подстроку, равную правой;

e. «startsWith» - левое значение начинается со строки, что справа;

f. «endsWith» - левое значение заканчивается строкой, что справа.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное создание версии политики.

Данные созданной политики можно получить с помощью команд «oberto_ListPolicyRecordsV2», описанных в данной инструкции.

2.14.2. Удаление версии политики (oberto_DeletePolicyV2)

Для удаления версии политики необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/oberto_DeletePolicyV2
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
id	String	Да	-	Идентификатор политики.	Существующий в сервисе идентификатор.
version	String	Да	-	Версия политики.	-

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление версии политики.

2.14.3. Удаление всех версий политики (oberto_DeleteAllPolicyVersionsV2)

Для удаления всех версий политики необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/oberto_DeleteAllPolicyVersionsV2
- Content-Type: application/json
- Request body: String, идентификатор существующей политики (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.14.4. Получение полного списка политик (oberto_ListPolicyRecordsV2)

Для получения полного списка политик необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/oberto_ListPolicyRecordsV2

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов с основными атрибутами политики, указанными при создании или изменении политики.

2.14.5. Проверка доступа (oberto_AuthorizeV2)

Для проверки доступа действий для конкретного ресурса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/oberto_AuthorizeV2
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
resourceId	String	Нет	-	Идентификатор конкретного объекта с типом «resourceType», для которого проверяются «actions».	-
resourceType	String	Нет	-	Название типа ресурса, для которого проверяются «actions».	-
actions	List[String]	Да	-	Список идентификаторов действий	-
attributes	Map[String, List[Strin	Нет	-	Произвольный список атрибутов ресурса (можно	-

	g]]			задавать атрибуты любого ресурса).	

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект, в котором:

- ключами являются значения, указанные во входных данных в поле «actions»;
- значениями является результат проверки доступа в виде значения «true» или «false».

2.15. Attila: сервис композитных правил

Сервис для упрощения работы с правилами сервиса системы прав. Предоставляет возможность создавать высокоуровневые правила доступа к объектам системы, которые затем транслирует в сервис системы прав декомпозируя их в более простые правила, подходящие для работы в Oberto.

При помощи команд реализованы возможности:

- получение списка команд;
- получение списка ресурсов и списка их полей;
- получение списка действий.

2.15.1. Получение списка команд с поддержкой фильтрации (attila_ListCommandAuthorizationInfo)

Для получения списка названий команд, для которых можно настроить доступ, необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/attila_ListCommandAuthorizationInfo
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. resource;
        ii. ruleaction;
        iii. uiaction;
        iv. commandName.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id	InSetQuery, LikeQuery
commandname	InSetQuery, LikeQuery
resource	InSetQuery, LikeQuery
ruleaction	InSetQuery, LikeQuery
uiaction	InSetQuery, LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи команд, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке - наименование команды, к которой можно настроить доступ.

2.15.2. Получение списка ресурсов с поддержкой фильтрации (attila_ListResourceTypeAuthorizationInfo)

Для получения списка ресурсов, для которых можно настроить доступ, необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/attila_ListResourceTypeAuthorizationInfo
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. resource;
        ii. ruleaction;
        iii. uiaction;
        iv. commandName.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id	InSetQuery, LikeQuery
commandname	InSetQuery, LikeQuery

resource	InSetQuery, LikeQuery
ruleaction	InSetQuery, LikeQuery
uiaction	InSetQuery, LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи ресурсов, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке - наименование ресурса, к которому можно настроить доступ.

2.15.3. Получение списка полей ресурсов с поддержкой фильтрации (attila_GetResourceTypeFields)

Для получения списка полей ресурсов, для которых можно настроить доступ, необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/attila_GetResourceTypeFields
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды не поддерживается пагинация, но поддерживается сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. fieldname;
        ii. title;
        iii. resourceType.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
resourcetype	InSetQuery, LikeQuery
fieldname	InSetQuery, LikeQuery
title	InSetQuery, LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект, в котором:

- ключ - доступный ресурс;
- значение ключа - JSON-объект, в котором ключи это название полей, а значение это их человекочитаемый заголовок.

2.15.4. Получение списка действий с поддержкой фильтрации (attila_ListRuleActionAuthorizationInfo)

Для получения списка действий необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/attila_ListRuleActionAuthorizationInfo
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. resource;
        ii. ruleaction;
        iii. uiaction;
        iv. commandName.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id	InSetQuery, LikeQuery
commandname	InSetQuery, LikeQuery
resource	InSetQuery, LikeQuery
ruleaction	InSetQuery, LikeQuery
uiaction	InSetQuery, LikeQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи действий, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке - наименование действия, к которому можно настроить доступ.

2.16. Tech-constants: сервис технического справочника

Сервис технического справочника предназначен для настройки и хранения констант - данных сервисов, которые относятся к сквозным функциональностям.

При помощи команд реализованы возможности:

- добавление, обновление, получение, удаление сервисов;
- добавление, обновление, получение, удаление типов констант;
- добавление, обновление, получение, удаление ручных констант.

2.16.1. Добавление или обновление сервиса (techConstants_UpsertService)

Для добавления или обновления сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_UpsertService
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Код сервиса.	Длина не более 32.
name	String	Да	-	Наименование сервиса.	Длина не более 128.
description	String	Нет	-	Описание сервиса.	-

Логика работы команды зависит от поля «code»:

- уникальное значение - добавляется новый сервис;
- существующее значение - обновляется соответствующий сервис.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное добавление или обновление сервиса.

Данные добавленного или обновленного сервиса можно получить с помощью команды «techConstants_ListServices», описанной в данной инструкции.

2.16.2. Удаление сервиса (techConstants_DeleteService)

Для удаления сервиса необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_DeleteService
- Content-Type: application/json
- Request body: String, идентификатор существующего сервиса (атрибут «code»), по которому нет констант

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.16.3. Получение полного списка сервисов (techConstants_ListServices)

Для получения полного списка сервисов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_ListServices

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов с атрибутами сервиса, переданными пользователем: все данные, указанные при добавлении или обновлении сервиса.

2.16.4. Добавление или обновление типа констант (techConstants_UpsertConstantType)

Для добавления или обновления типа констант необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_UpsertConstantType
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Код типа констант.	Длина не более 32. Уникальное значение.
name	String	Да	-	Наименование типа констант.	Длина не более 128.
description	String	Нет	-	Описание типа констант.	-

Логика работы команды зависит от поля «code»:

- уникальное значение - добавляется новый тип констант;
- существующее значение - обновляется соответствующий тип констант.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное добавление или обновление типа констант.

Данные добавленного или обновленного типа констант можно получить с помощью команды «techConstants_ListConstantTypes», описанной в данной инструкции.

2.16.5. Получение полного списка типов констант (techConstants_ListConstantTypes)

Для получения полного списка сервисов необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_ListConstantTypes

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов с атрибутами типа, переданными пользователем: все данные, указанные при добавлении или обновлении типа констант.

2.16.6. Удаление типа констант (techConstants_DeleteConstantType)

Для удаления сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_DeleteConstantType
- Content-Type: application/json
- Request body: String, идентификатор существующего типа констант (атрибут «code»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.16.7. Добавление или обновление ручной константы (techConstants_UpsertManualConstant)

Для добавления или обновления ручной константы необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_UpsertManualConstant
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Код константы.	Длина не более 64. Комбинация «code» и «constantType» каждой константы должна быть уникальной.
constantType	String	Да	-	Код типа констант, к которому относится текущая константа.	Длина не более 32. Существующий код типа констант. Комбинация «code» и «constantType» каждой константы должна быть уникальной.
service	String	Да	-	Код сервиса, к которому относится текущая константа.	Длина не более 32. Существующий код сервиса.
name	String	Да	-	Наименование константы.	Длина не более 128.
description	String	Нет	-	Описание константы.	-
settings	JSON	Нет	-	Дополнительные произвольные настройки константы.	-

Логика работы команды зависит от комбинации «code» и «constantType»:

- уникальная комбинация - добавляется новая константа;
- существующее значение - обновляется соответствующая константа.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное добавление или обновление константы.

Данные добавленной или обновленной константы можно получить с помощью команды «techConstants_ListAllConstants», описанной в данной инструкции.

2.16.8. Удаление ручной константы (techConstants_DeleteManualConstant)

Для удаления ручной константы необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_DeleteManualConstant
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
code	String	Да	-	Код константы.	Длина не более 64. Существующая комбинация «code» и «constantType».
constantType	String	Да	-	Код типа констант, к которому относится текущая константа.	Длина не более 32. Существующая комбинация «code» и «constantType».

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.16.9. Получение списка констант с поддержкой фильтрации (techConstants_ListAllConstants)

Для получения списка констант необходимо выполнить следующий запрос:

- Метод: POST

- URL: http://<IP-адрес_BM>:8083/api/send/sync/techConstants_ListAllConstants
- Content-Type: application/json
- Request body: Search - универсальный тип параметров списочного запроса. Для данной команды поддерживается пагинация и сортировка с фильтрацией по следующим данным:
    a. Доступные поля для сортировки:
        i. code;
        ii. name;
        iii. constantType.name;
        iv. source.
    b. Доступные поля для фильтрации и виды фильтров по ним:

Поле	Виды фильтров
id	InSetQuery
code	InSetQuery, LikeQuery
name	InSetQuery, LikeQuery
constantType	InSetQuery
source	InSetQuery
service	InSetQuery

При успешном выполнении команды в теле ответа поле «value» содержит объект типа Page - стандартную страницу списка данных фреймворка. Данная страница содержит записи констант, соответствующие переданным параметрам фильтрации, сортировки и пагинации. Каждая запись в списке представляет собой JSON-объект со следующими данными о константе:

- атрибуты, переданные пользователем: все данные, указанные при добавлении или обновлении константы;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, дата последнего обновления)*.

2.17. UI-storage: сервис хранения UI

Сервис предназначен для хранения настроек форм объектов, реестров, карточек объектов и т.д.

При помощи команд реализованы возможности:

- создание, изменение и удаление форм;
- получение форм;

- получение списка типов форм;
- создание, изменение и удаление ключей и значений глобальных настроек;
- получение глобальных настроек.

2.17.1. Получение полного списка типов форм (UIStorage_http_ListFormTypes)

Для получения полного списка всех существующих типов форм необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_http_ListFormTypes

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов с атрибутами всех стандартных типов форм (значение атрибутов в полях «code», «title» и «description»).

2.17.2. Создание формы (UIStorage_kafka_CreateForm)

Для создания формы необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_CreateForm
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
entityType	String	Да	-	Тип сущности.	Длина не более 127. Не пустое. Комбинация «entityType» и «formType» каждой формы должна быть уникальной.
formType	String	Да	-	Код типа формы.	Длина не более 127. Значение должно быть из списка типов форм сервиса. Не пустое. Комбинация

					«entityType» и «formType» каждой формы должна быть уникальной.
formData	JSON	Да	-	Данные формы.	-

При успешном выполнении команды в теле ответа поле «value» содержит идентификатор созданной формы в виде строки формата UUID.

Данные созданной формы можно получить с помощью команд «UIStorage_http_GetForm» или «UIStorage_http_ListForms», описанных в данной инструкции.

2.17.3. Получение данных формы (UIStorage_http_GetForm)

Для получения данных формы необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_http_GetForm
- Content-Type: application/json
- Request body: UUID, идентификатор существующей формы (атрибут «id»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со следующими данными:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении формы;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.17.4. Изменение формы (UIStorage_kafka_UpdateForm)

Для изменения данных формы необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_UpdateForm
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения

id	UUID	Да	-	ID формы.	Существующий в сервисе идентификатор.
formData	JSON	Да	-	Данные формы.	-
version	Int	Да	-	Версия, полученная при запросе данных формы. Нужна для оптимистичных блокировок.	Совпадает с текущей версией формы.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной формы можно получить с помощью команд «UIStorage_http_GetForm» или «UIStorage_http_ListForms», описанных в данной инструкции.

2.17.5. Удаление формы (UIStorage_kafka_DeleteForm)

Для удаления формы необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_DeleteForm
- Content-Type: application/json
- Request body: String : Id (UUID), идентификатор существующей формы

При успешном выполнении команды в теле ответа поле «value» содержит значение «1», указывающее на успешное удаление.

2.17.6. Получение полного списка форм (UIStorage_http_ListForms)

Для получения списка форм сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_http_ListForms

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о форме:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении формы;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, идентификатор)*.

2.17.7. Создание глобальной настройки (UIStorage_kafka_CreateKey)

Для создания глобальной настройки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_CreateKey
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
key	String	Да	-	Ключ глобальной настройки.	Длина не более 64. Не пустое. Уникальное значение.
value	JSON	Да	-	Значение глобальной настройки.	-
title	String	Да	-	Название глобальной настройки.	Длина не более 127 символов.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное создание.

Данные созданной глобальной настройки можно получить с помощью команд «UIStorage_http_ListKeys» и «UIStorage_http_GetValue», описанных в данной инструкции.

2.17.8. Получение данных глобальной настройки (UIStorage_http_GetValue)

Для получения данных глобальной настройки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_http_GetValue
- Content-Type: application/json
- Request body: String, код существующей глобальной настройки (атрибут «key»)

При успешном выполнении команды в теле ответа поле «value» содержит JSON-объект со значением глобальной настройки (поле «value» глобальной настройки).

2.17.9. Изменение глобальной настройки (UIStorage_kafka_UpdateKey)

Для изменения глобальной настройки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_UpdateKey
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
key	String	Да	-	Ключ глобальной настройки.	Существующий в сервисе ключ. Длина не более 64.
value	JSON	Да	-	Значение глобальной настройки.	-
title	String	Да	-	Название глобальной настройки.	Длина не более 127.
version	Int	Да	-	Версия, полученная при запросе данных глобальной настройки. Нужна для оптимистичных блокировок.	Совпадает с текущей версией глобальной настройки.

При успешном выполнении команды в теле ответа поле «value» содержит число 1, указывающее на количество обновленных записей.

Данные измененной глобальной настройки можно получить с помощью команд «UIStorage_http_ListKeys» и «UIStorage_http_GetValue», описанных в данной инструкции.

2.17.10. Изменение значения глобальной настройки (UIStorage_kafka_UpdateValue)

Для изменения значения глобальной настройки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_UpdateValue
- Content-Type: application/json
- Request body: JSON, содержащий следующую информацию:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
key	String	Да	-	Ключ глобальной настройки.	Существующий в сервисе идентификатор. Длина не более 64.
value	JSON	Да	-	Значение глобальной настройки.	-
version	Int	Да	-	Версия, полученная при запросе данных глобальной настройки. Нужна для оптимистичных блокировок.	Совпадает с текущей версией глобальной настройки.

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное изменение.

Данные измененной глобальной настройки можно получить с помощью команд «UIStorage_http_ListKeys» и «UIStorage_http_GetValue», описанных в данной инструкции.

2.17.11. Удаление глобальной настройки (UIStorage_kafka_DeleteKey)

Для удаления глобальной настройки необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_kafka_DeleteKey
- Content-Type: application/json
- Request body: String, идентификатор существующей глобальной настройки (атрибут «key»)

При успешном выполнении команды в теле ответа поле «value» содержит значение «true», указывающее на успешное удаление.

2.17.12. Получение полного списка глобальных настроек (UIStorage_http_ListKeys)

Для получения полного списка глобальных настроек сервиса необходимо выполнить следующий запрос:

- Метод: POST
- URL: http://<IP-адрес_BM>:8083/api/send/sync/UIStorage_http_ListKeys

При успешном выполнении команды в теле ответа поле «value» содержит список JSON-объектов со следующими данными о глобальных настройках:

- атрибуты, переданные пользователем: все данные, указанные при создании или изменении глобальной настройки;
- системные атрибуты: данные, сгенерированные сервисом автоматически *(например, версия)*.

3. Общие типы данных

Структуры данных, которые используются в нескольких фреймоврках - выделяются в отдельные общие модели.

3.1. Search

Для некоторых команд предусматривается фильтрация, сортировка и пагинация результата. Для этих целей был добавлен тип данных Search, который передается в качестве параметров к команде.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
query	Filter query	Да	-	Строка с запросом, в котором указывается комбинация фильтров. Для запросов без фильтрации - пустая строка.	Значения в строке должны быть указаны в поле context.
context	Filter context	Да	-	Контекст фильтров запроса. Объект, в котором ключ - это название фильтра, а значение - это значение фильтра. Для запросов без фильтрации - пустой объект.	-

sorting	Sorting	Нет	Сортировка по полю code в порядке возрастания:
			{
			"fieldName": "code",
			"order": "asc"
			}
				Параметры сортировки данных.	-

paging	Paging	Нет	Первая страница с 10 элементами:
			{
			"page": 1,
			"count": 10,
			}
				Параметры пейджинга.	-

3.2. Filter query

Является строкой с запросом для фильтрации при выполнении команды.

Запрос состоит из названий атрибутов фильтруемого ресурса и способов их комбинирования:

- «фильтр для поля» - название атрибута у ресурса, по которому предполагается поиск. Обычно это ресурс из результата команды.
- «комбинация фильтров» - бинарные операции «||» и «&&» с возможностью группировки через круглые скобки «(» и «)».

3.3. Filter context

С помощью контекста передаются параметры для фильтрации полей из «query».

Контекст является JSON объектом, который представляет собой тип Map[String, Json], где:

- поле JSON объекта является ключом для Map[String, Json] и должно быть упомянуто в поле «query» (т.е. иметь название атрибута ресурса);
- значение поля JSON объекта является контекстом для фильтра FilterQueryContext. Каждый фильтр имеет свой формат.

3.4. FilterQueryContext

Общий тип для всех видов фильтров. Существует несколько реализаций, у которых может поддерживаться несколько форматов для записи условия.

3.4.1. InSetQuery

Фильтрация атрибута ресурса по списку значений. Если значение атрибута ресурса является массивом, то условие фильтрации возвращает «true», если этот массив и список значений для условия фильтрации пересекаются хотя бы по одному элементу. Иначе условие фильтрации возвращает «true», если одиночное значение из атрибута соответствует хотя бы одному значению из фильтра.

Данный вид фильтра формируется в виде JSON объекта со следующими полями:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
values	List[String]	Да	-	Список значений для условия фильтрации.	-
kind	String	Да	-	Вид фильтра.	Может принимать

					только значение «any».
negation	Boolean	Нет	false	Необходимость отрицания результата фильтрации.	-

3.4.2. AllInSetQuery

Фильтрация массива из атрибута ресурса по списку значений. Условие фильтрации возвращает «true», если массив из атрибута содержит все значения из фильтра. Работает только для полей, являющихся массивом.

Данный вид фильтра формируется в виде JSON объекта со следующими полями:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
values	List[String]	Да	-	Список значений для условия фильтрации.	-
kind	String	Да	-	Вид фильтра.	Может принимать только значение «all».
negation	Boolean	Нет	false	Необходимость отрицания результата фильтрации.	-

3.4.3. LikeQuery

Фильтрация атрибута ресурса по шаблону. Условие фильтрации возвращает «true», если значение из атрибута соответствует шаблону.

Данный вид фильтра соответствует фильтру «like» из SQL и формируется в виде JSON объекта со следующими полями:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
likequery	String	Да	-	Шаблон для условия фильтрации.	-
kind	String	Да	-	Вид фильтра.	Может принимать только значение «like».
negation	Boolean	Нет	false	Необходимость	-

				отрицания результата фильтрации.	

В поле «likequery», помимо слов для условия, поддерживаются специальные символы:

- «%» - неограниченное кол-во любых символов;
- «_» - только один любой символ, который можно использовать несколько раз подряд.

3.4.4. TsQuery

Фильтрация атрибута ресурса по условие для полнотекстового поиска PostgreSQL. Условие фильтрации возвращает «true», если значение из атрибута соответствует условию. Поддерживаются следующие операторы:

- «&» - обозначает «И»;
- «|» - обозначает «ИЛИ»;
- «:*» - обозначает начало слова;
- «!» - обозначает отрицание условия.
- Для логического объединения нескольких операторов могут использоваться скобки «(» и «)».

Данный вид фильтра формируется в виде JSON объекта со следующими полями:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
values	String	Да	-	Условия полнотекстового поиска для фильтрации.	-
kind	String	Да	-	Вид фильтра.	Может принимать только значение «ts».
negation	Boolean	Нет	false	Необходимость отрицания результата фильтрации.	-

3.4.5. RangeQuery

Фильтрация атрибута ресурса по условию вхождения в интервал из значения фильтра. Условие фильтрации: from <= targetValue <= to.

Данный вид фильтра соответствует фильтру «like» из SQL и формируется в виде JSON объекта со следующими полями:

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
from	Long	Нет	-	Значение, с которого начинается интервал для фильтрации (входит в интервал).	-
to	Long	Нет	-	Значение, с которым заканчивается интервал для фильтрации (входит в интервал).	-
kind	String	Да	-	Вид фильтра.	Может принимать только значение «range».
negation	Boolean	Нет	false	Необходимость отрицания результата фильтрации.	-

Если поля from и to не указаны, то условие всегда вернет true.

3.5. Sorting

Тип данных для передачи параметров сортировки списка элементов.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
fieldName	String	Да	-	Название поля для сортировки.	-
order	String	Да	-	Направление сортировки.	Допустимые значения: asc и desc (для сортировки по возрастанию и убыванию соответственно).

3.6. Paging

Тип данных для передачи параметров пейджинга списка.

Поле	Тип	Обязательное	Значение по умолчанию	Описание	Ограничения
page	Int	Да	-	Номер запрашиваемой страницы. Страницы нумеруются с 1.	Значение больше 0.
count	Int	Да	-	Количество элементов на странице.	Значение больше или равно 0.

3.7. Page

Страница списка элементов (часть списка с примененным пейджингом).

Поле	Тип	Описание
items	List[JSON]	Список элементов на запрошенной странице.
total	Int	Общее количество элементов.

4. Список тестов и целевые результаты для экспертной проверки

Для проверки корректности функционирования программы перед её дальнейшей эксплуатацией необходимо провести тестирование сервисов Verdi на тестовом наборе данных.

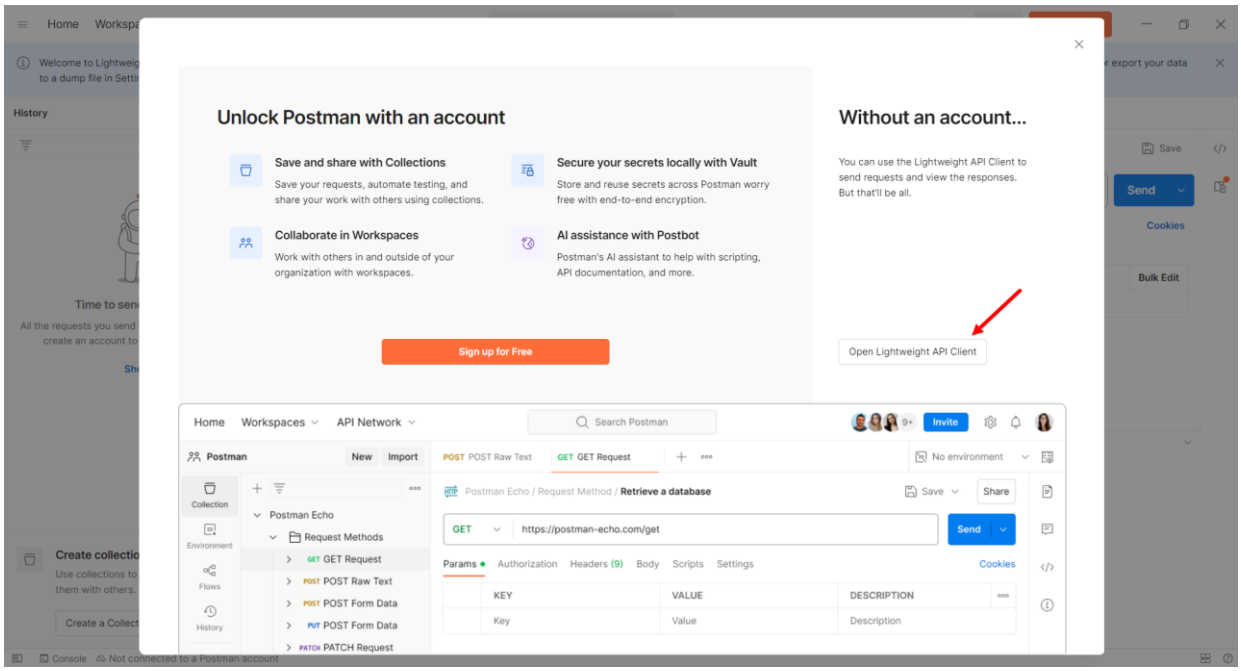
4.1. Общая инструкция для выполнения тестов

Тестирование функционала заключается в отправке GET и POST запросов на определенные URL и получения ожидаемых ответов. При этом используется интерфейс сервиса REST API.

4.1.1. Настройка инструмента для выполнения тестов

Для отправки GET и POST-запросов необходимо воспользоваться программой Postman, которую можно скачать с официального сайта: https://www.postman.com/downloads/

После установки и открытия программы, откроется её основной интерфейс. Без входа в аккаунт и без подключения Workspace (рабочего пространства) будет доступно только локальное создание и отправка запросов в режиме Lightweight API Client, без возможности сохранения запросов, коллекций (наборов запросов) и других данных на сервере Postman. Для выполнения тестов, расположенных ниже, будет достаточно использования режима Lightweight API Client.

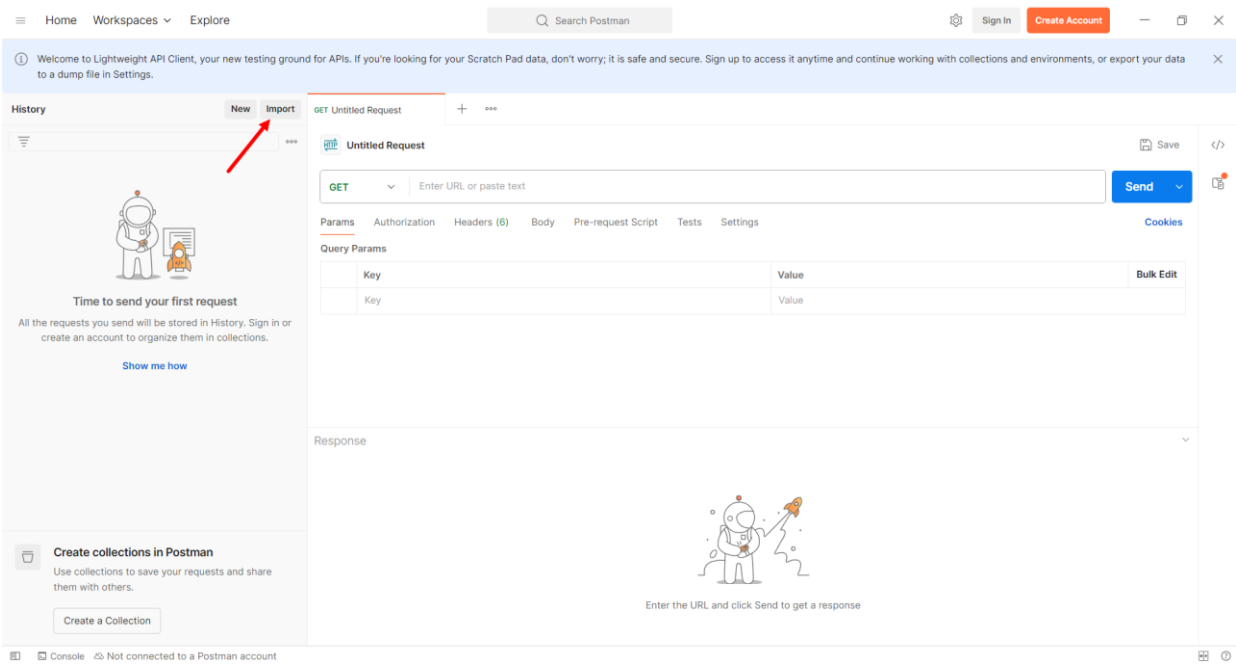


4.1.2. Шаги для выполнения тестов

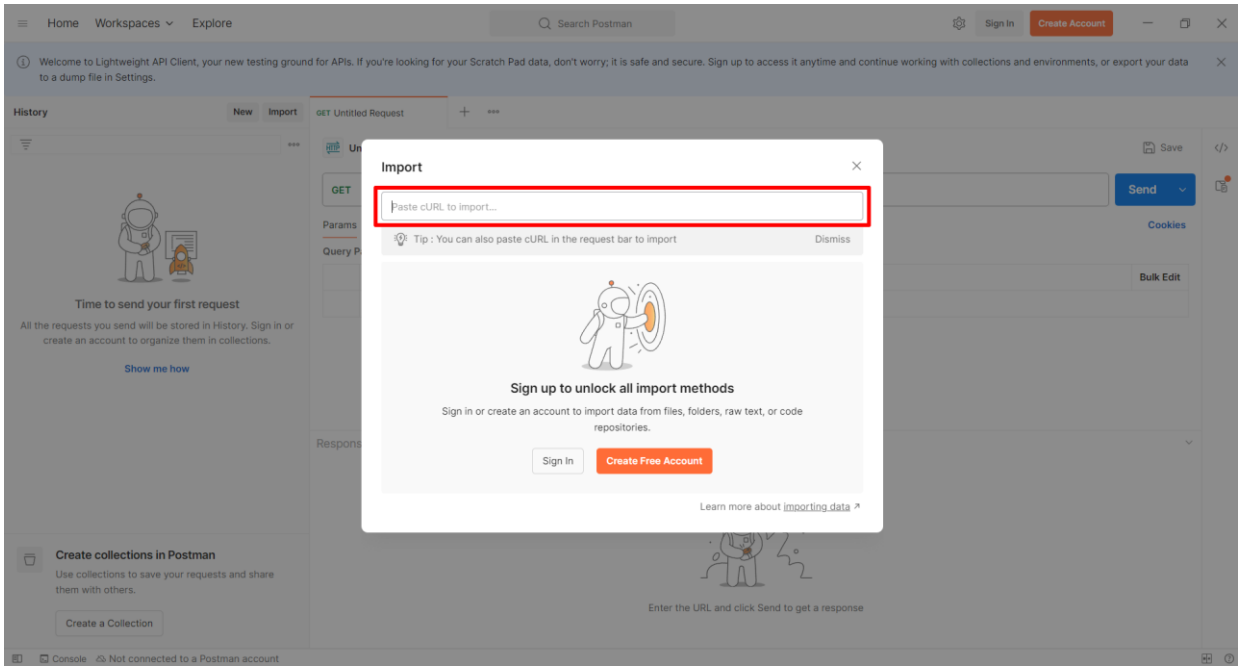
Для выполнения каждой из проверок, расположенных ниже, необходимо выполнить следующие шаги:

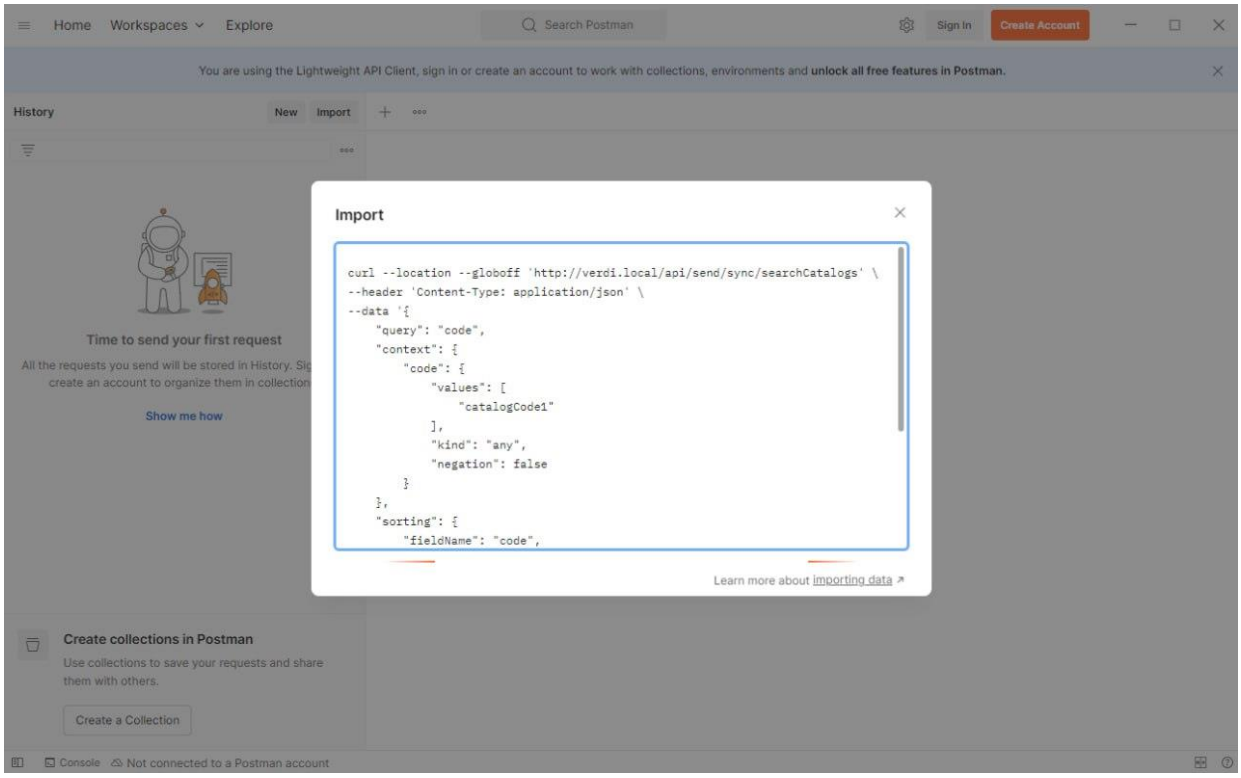
1. Скопировать cURL запроса из используемых для тестирования данных проверяемой команды. В тех случаях, когда для проверки необходимо выполнение нескольких cURL-запросов — их необходимо запускать строго в том порядке, в котором они указаны в описании теста. При этом для каждого из указанных cURL необходимо повторить последовательность шагов №1-6.

2. В Postman нажать на кнопку «Import».

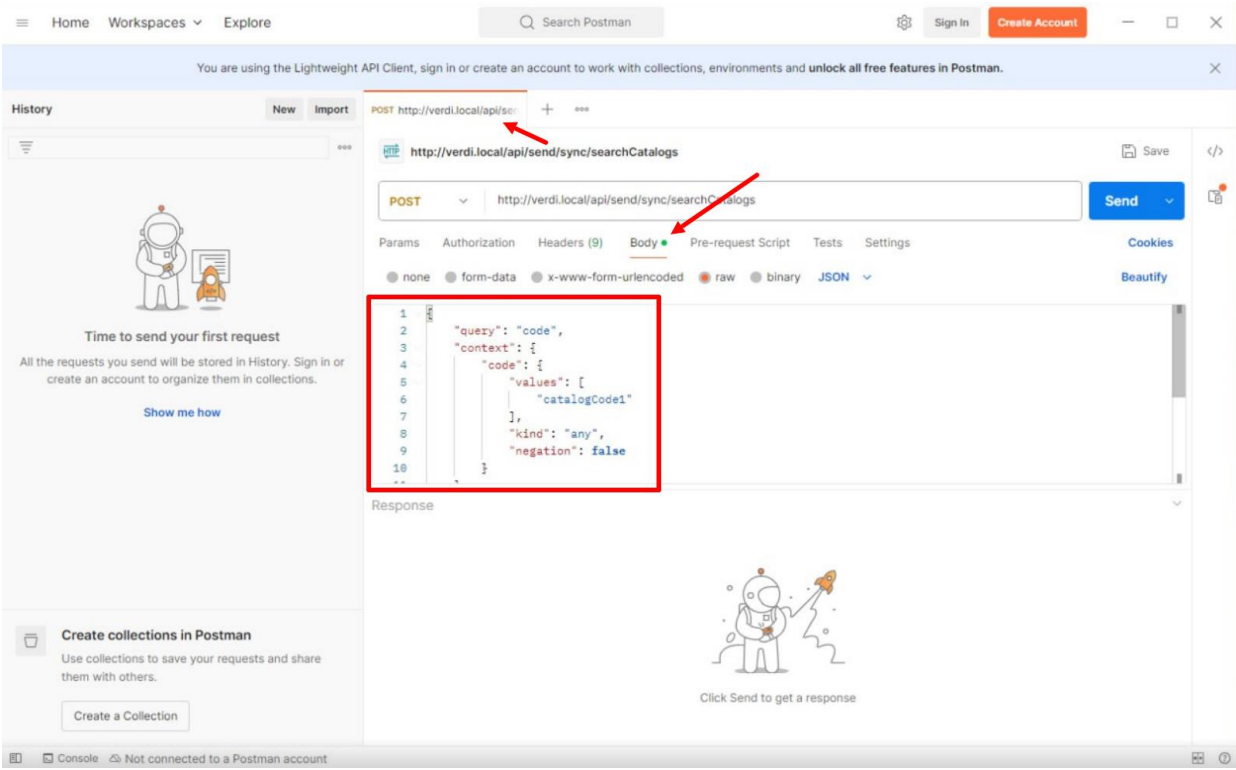


3. В открывшемся окне в поле ввода «Paste cURL to import…» вставить cURL, который был скопирован на шаге 1. Введённый cURL будет автоматически обработан и преобразован в POST или GET-запрос.

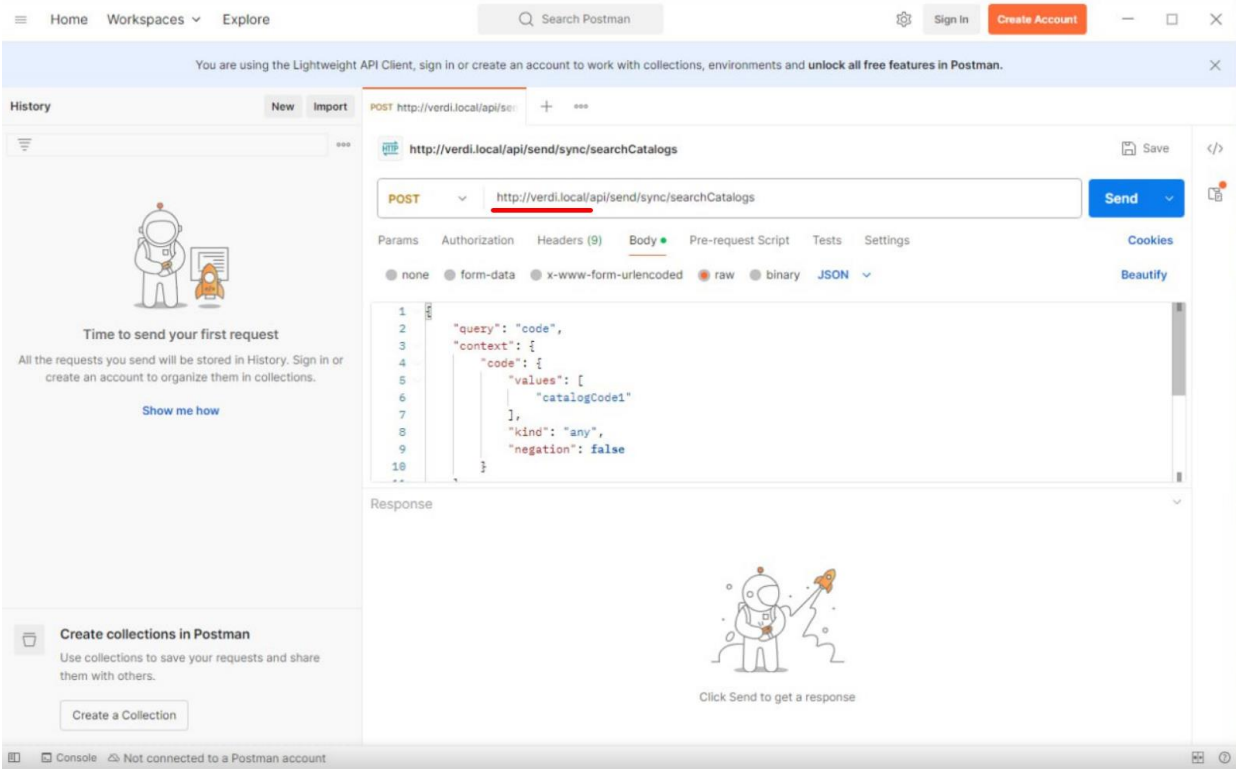




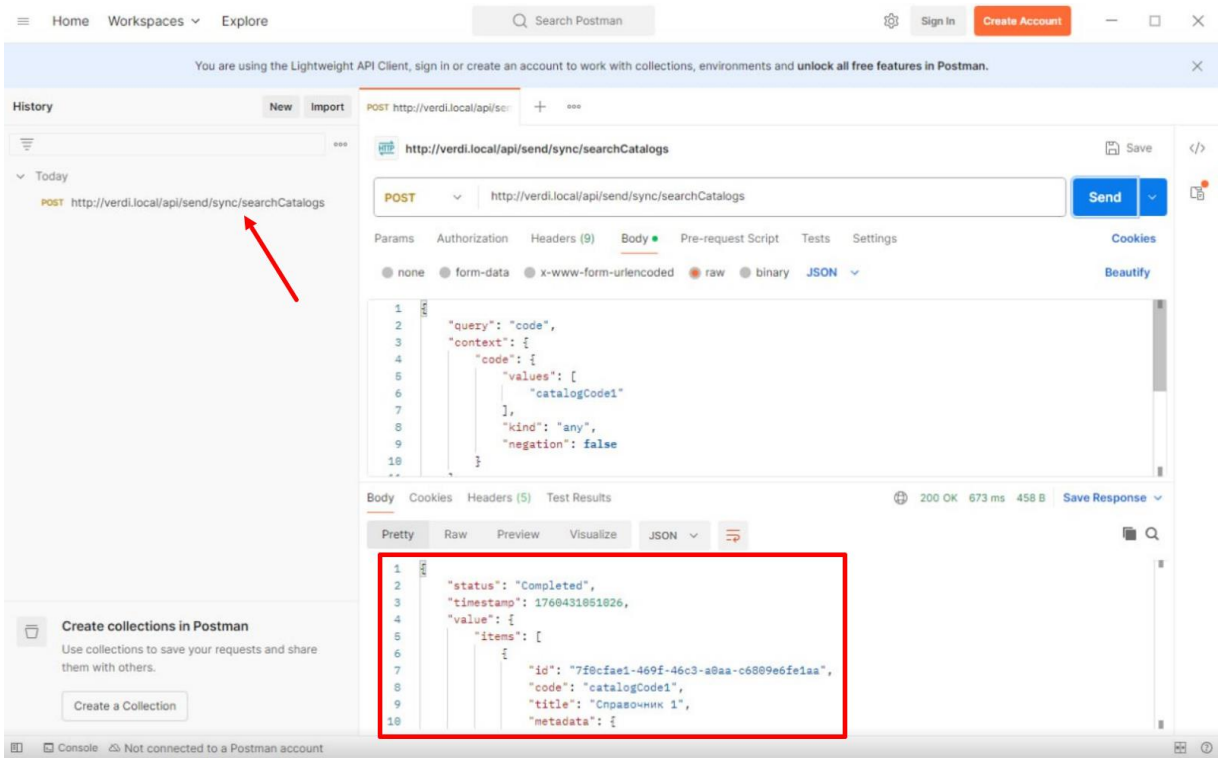
4. После преобразования cURL-запроса в интерфейсе Postman появится новая вкладка с импортированным запросом. На вкладке «Body» запроса можно убедиться, что тело запроса соответствует содержимому исходного cURL-запроса.



5. Перед отправкой запроса следует убедиться в корректности адреса виртуальной машины, указанного в URL. При выполнении этого шага предполагается, что виртуальная машина уже настроена, запущена и полностью проинициализирована. Необходимо убедиться, что в URL запроса указан адрес «http://local.verdi», настроенный ранее в конце этапа «Запуск и получение IP-адреса». Пример подготовленного к отправке запроса:



6. Для отправки запроса необходимо нажать на кнопку «Send». В результате этого шага в области «History», расположенной слева, будет сохранён выполненный запрос, а в области «Response», расположенной в нижней части экрана во вкладке «Body» — будет отображён ответ на выполненный запрос.



7. Необходимо сравнить полученный результат в поле «value» ответа на запрос с планируемым ответом проводимого теста и убедиться в корректности работы программы.

4.2. Тесты и целевые результаты

Для проверки функционирования платформы достаточно убедиться в корректной работе нескольких ключевых сервисов. Далее по каждому сервису описаны тесты по нескольким его командам. Все команды вызываются через сервис Api-gateway *(помимо команд самого сервиса)*, имеющий единую структуру ответа - при успешном выполнении возвращается JSON объект со следующими данными:

- «status» - статус выполнения команды. Т.к. используются синхронные запросы, то всегда будет равен значению «Completed»
- «timestamp» - временная метка
- «value» - содержит итог выполнения команды и зависит от логики команды.

В силу единообразной структуры - в описании тестирования каждой команды в планируемом ответе описывается только значение поля «value» в теле ответа.

4.2.1. Alexandrina: сервис справочников

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.1.1. Тестирование создания справочника

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createCatalog' \
--header 'Content-Type: application/json' \
--data '{
   "code": "UniqueTestCode",
   "title": "Тестовый справочник",
   "metadata": {
      "shortTitle": "TC"
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданного справочника в виде строки формата UUID.

4.2.1.2. Тестирование получения списка справочников с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/searchCatalogs' \
--header 'Content-Type: application/json' \
--data '{
   "query": "code",
   "context": {
      "code": {
         "values": [
```

```
            "catalogCode1"
         ],
         "kind": "any",
         "negation": false
      }
   },
   "sorting": {
      "fieldName": "code",
      "order": "desc"
   },
   "paging": {
      "page": 1,
      "count": 5
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
       "items": [
         {
                  "id": "7f0cfae1-469f-46c3-a0aa-c6809e6fe1aa",
                  "code": "catalogCode1",
                  "title": "Справочник 1",
                  "metadata": {
                           "shortTitle": "C1"
                  },
                  "modified": 1759989864617,
                  "version": 1,
                  "isProtected": true,
                  "items": 0
         }
       ],
       "total": 1
}
```

4.2.1.3. Тестирование создания элемента справочника

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/createCatalogItem'
\
```

```
--header 'Content-Type: application/json' \
--data '{
   "catalogId": "a9a53147-1405-4868-8a3d-822464982f08",
   "code": "element1",
   "title": "Элемент 1",
   "metadata": {
      "shortTitle": "Э1"
   },
   "isProtected": false
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданного элемента справочника в виде строки формата UUID.

4.2.1.4. Тестирование получения списка элементов справочника с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/listItemsByCatalog'
\
--header 'Content-Type: application/json' \
--data '{
   "data": "98813efa-f662-4305-9461-b139b65641a6",
   "search": {
      "query": "code",
      "context": {
         "code": {
            "values": [
               "catalogCode3-1"
            ],
            "kind": "any",
            "negation": false
         }
      },
      "sorting": {
         "fieldName": "title",
         "order": "desc"
```

```
      },
      "paging": {
         "page": 1,
         "count": 5
      }
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "items": [
        {
                "id": "c2a8fc75-4e45-4b2e-87c9-f98749e184b4",
                "catalogId": "98813efa-f662-4305-9461-b139b65641a6",
                "code": "catalogCode3-1",
                "title": "Первый элемент",
                "archived": false,
                "metadata": {
                        "shortTitle": "П1"
                },
                "created": 1759990855039,
                "modified": 1759990855039,
                "version": 1,
                "isProtected": false
        }
      ],
      "total": 1
}
```

4.2.2. Api-gateway: сервис маршрутизации

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.2.1. Тестирование синхронного запроса

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/techConstants_UpsertService' \
--header 'Content-Type: application/json' \
--data '{
```

```
  "code": "externalService",
  "name": "Внешний сервис",
  "description": "Внешний сервис относительно платформы"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.2.2. Тестирование асинхронного запроса

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/async/techConstants_UpsertService' \
--header 'Content-Type: application/json' \
--data '{
  "code": "routeService",
  "name": "Сервис маршрутов",
  "description": "Сервис для настройки этапов для разных бизнес-процессов. Внешний сервис относительно платформы."
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ - не является структурой api-gateway и выглядит следующим образом:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор отправленной команды в виде строки формата UUID.

4.2.2.3. Тестирование получения статуса

Используемые для тестирования данные:

- Готовый cURL запроса для асинхронного вызова команды:

```
curl --location --globoff
'http://verdi.local/api/send/async/techConstants_UpsertService' \
--header 'Content-Type: application/json' \
--data '{
  "code": "electronicSignatureService",
  "name": "Сервис электронной подписи",
```

```
  "description": "Сервис для электронной подписи документов"
}'
```

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/status/{{commandID}}'
```

где {{commandID}} - идентификатор команды, полученный в результате асинхронного вызова команды.

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ: зависит от времени выполнения запроса после асинхронного вызова

- команда завершила свое выполнение - ответ в поле «value»:

```
true
```

- команда не завершила свое выполнение - ответ:

```
{
  "status": "InProcess",
  "timestamp": <timestamp>,
}
```

где <timestamp> - временная метка последнего обновления статуса.

4.2.2.4. Тестирование скачивания файла

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/download/datamodel/document/83df54b4-636c-
40e9-87d6-294c0a5ecda7'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста» (для данного теста необходимо нажать не на кнопку «Send», а открыть выпадающий список рядом с ней и нажать «Send and Download»)

Планируемый ответ - программа предлагает сохранить файл с наименованием «convert-example.docx»: при выборе расположения и подтверждении - на компьютер скачивается файл.

4.2.3. Archiver: сервис архивации файлов

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.3.1. Тестирование архивации файлов объектов сущностей сервиса Data-model с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/archiver_kafka_archiveFromDataModelSea
rch' \
--header 'Content-Type: application/json' \
--data '{
   "search": {
      "query": "entityType",
      "context": {
         "entityType": "document"
      }
   },
   "archiveName": "All files from documents"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "archiveUrl": "archive/All files from documents.zip",
      "status": "finished"
}
```

4.2.3.2. Тестирование получения списка архивов с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/archiver_http_listArchives' \
--header 'Content-Type: application/json' \
--data '{
   "query": "",
   "context": {},
   "sorting": null,
   "paging": null
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
        "archiveUrl": "archive/Documents.zip",
        "userId": "00000000-0000-0000-0000-000000000000",
        "status": "finished",
        "fileUrls": [
          "datamodel/document/83df54b4-636c-40e9-87d6-294c0a5ecda7"
        ],
        "entityObjectIdentities": null
}
```

4.2.4. Buzzer: сервис уведомлений

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.4.1. Тестирование получения полного списка зарегистрированных сервисов

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/listSendingBuzzer'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующие объекты:

```
[
  {
    "id": "Profile",
    "topic": "personal-page",
    "version": 1
  },
  {
    "id": "Email",
    "topic": "email",
    "version": 1
  }
```

```
]
```

4.2.4.2. Тестирование создания пользовательской настройки уведомления

Используемые для тестирования данные:

- Готовый cURL запроса для аутентификации пользователя:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_LoginByCredentials' \
--header 'Content-Type: application/json' \
--data-raw '{
   "email": "user@embedika.ru",
   "password": "js5;Z` r!?#33GygW"
}'
```

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/createSettingBuzzer' \
--header 'Content-Type: application/json' \
--data '{
   "code": "userSettingBuzzer",
   "sendingIds": [
      "Profile",
      "Email"
   ]
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданной пользовательской настройки уведомления в виде строки формата UUID.

4.2.4.3. Тестирование получения полного списка пользовательских настроек уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса для аутентификации пользователя:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_LoginByCredentials' \
--header 'Content-Type: application/json' \
--data-raw '{
   "email": "user@embedika.ru",
   "password": "js5;Z` r!?#33GygW"
}'
```

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/listSettingBuzzer'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "6bab06b0-e3b0-4253-a787-13f158cf4928",
      "code": "profileAndEmailSettingExample",
      "sendingIds": [
         "Profile",
         "Email"
      ],
      "userId": "1912826f-a12b-4470-a5e9-1c961ea7d731",
      "version": 1
   }
]
```

4.2.4.4. Тестирование отправки уведомления

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/sendNotification' \
--header 'Content-Type: application/json' \
--data '{
   "code": "profileNotification",
   "userId": "1912826f-a12b-4470-a5e9-1c961ea7d731",
   "data": {
      "field1": "Текст, отправленный в уведомлении"
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.5. Buzzer-subscription: сервис подписок

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.5.1. Тестирование создания настройки для события

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createEventSettingsBuzzer-subscription' \
--header 'Content-Type: application/json' \
--data '{
    "entityType": "Specification",
    "eventType": "Update",
    "enabled": true,
    "notificationCode": "SpecificationUpdated"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданных настроек для события в виде строки формата UUID.

4.2.5.2. Тестирование получения полного списка настроек для событий

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST 'http://verdi.local/api/send/sync/listEventSettingsBuzzer-subscription'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "72501f1e-d1e9-4ec7-aa1a-bfeddfe7317b",
      "entityType": "document",
      "eventType": "Update",
      "enabled": true,
      "notificationCode": "documentUpdated",
      "version": 1
   }
]
```

4.2.5.3. Тестирование создания подписки пользователя

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createSubscriptionBuzzer-subscription' \
--header 'Content-Type: application/json' \
--data '{
   "entityType": "Specification",
   "entityId": "7699ece1-89f1-400c-a30e-d442332f0f35",
   "eventType": "Update",
   "enabled": true,
   "userId": "1912826f-a12b-4470-a5e9-1c961ea7d731"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданной подписки пользователя в виде строки формата UUID.

4.2.5.4. Тестирование получения полного списка подписок пользователей

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/listSubscriptionBuzzer-subscription'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
  {
    "id": "f8561101-3d5d-443b-975e-65eddcd78d5c",
    "entityType": "document",
    "entityId": "61090aab-6130-4897-942d-efa35c4b24e0",
    "eventType": "Update",
    "enabled": true,
    "userId": "1912826f-a12b-4470-a5e9-1c961ea7d731",
    "version": 1
  }
]
```

4.2.6. Buzzer-email: сервис уведомлений на электронную почту

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.6.1. Тестирование создания настройки для уведомления

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createSettingEmail' \
--header 'Content-Type: application/json' \
--data '{
  "id": "emailSettingCode",
  "deliveryId": "exampleDeliveryId",
  "groupId": "exampleGroupId",
  "version": 1
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"emailSettingCode"
```

4.2.6.2. Тестирование получения полного списка настроек уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/listSettingEmail'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "exampleSettingCode",
      "deliveryId": "exampleDeliveryId",
      "groupId": "exampleGroupId",
      "version": 1
   }
]
```

4.2.6.3. Тестирование создания группы уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/createGroupEmail' \
--header 'Content-Type: application/json' \
--data '{
   "id": "emailGroupId",
   "name": "Email group",
   "version": 1
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"emailGroupId"
```

4.2.6.4. Тестирование получения полного списка групп уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/listGroupEmail'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "exampleGroupId",
      "name": "Example email group",
      "version": 1
   }
]
```

4.2.6.5. Тестирование создания шаблона уведомления

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createTemplateEmail' \
--header 'Content-Type: application/json' \
--data '{
   "id": "emailTemplateId",
   "title": "Email template",
   "content": "{{#groups}}{{#notifications}}{{#data}} <b>Example email
content</b>: <текст_шаблона> {{/data}}{{/notifications}}{{/groups}}",
   "version": 1
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
“emailTemplateId”
```

4.2.6.6. Тестирование получения полного

списка шаблонов уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/listTemplateEmail'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "exampleTemplate",
      "title": "Example template",
      "content": "{{#groups}}{{#notifications}}{{#data}} <b>Example
content</b>: <текст_шаблона> {{/data}}{{/notifications}}{{/groups}}",
      "version": 1
   }
]
```

4.2.6.7. Тестирование создания рассылки уведомления

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createDeliveryEmail' \
--header 'Content-Type: application/json' \
--data '{
   "id": "emailDeliveryId",
   "schedule": "0 12"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"emailDeliveryId"
```

4.2.6.8. Тестирование получения полного списка рассылок уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/listDeliveryEmail'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
  {
    "id": "exampleDeliveryId",
    "schedule": "0 12",
    "version": 1759994755895
  }
]
```

4.2.7. Buzzer-personal-page: сервис уведомлений в личном кабинете

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.7.1. Тестирование получения списка уведомлений в ЛК любого пользователя с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/auditSearchPPNotifications'
\
--header 'Content-Type: application/json' \
--data '{
  "search": {
    "query": "userId && id",
    "context": {
      "userId": {
        "values": [
          "1912826f-a12b-4470-a5e9-1c961ea7d731"
        ],
        "kind": "any",
        "negation": false
```

```
        },
        "id": {
          "values": [
            "cb87e6d1-f3ea-45b8-af79-df0b619df925"
          ],
          "kind": "any",
          "negation": false
        }
      },
      "sorting": {
        "fieldName": "created",
        "order": "asc"
      },
      "paging": {
        "page": 1,
        "count": 5
      }
    },
    "render": true
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
    "items": [
        {
            "id": "cb87e6d1-f3ea-45b8-af79-df0b619df925",
            "code": "profileNotification",
            "userId": "1912826f-a12b-4470-a5e9-1c961ea7d731",
            "data": {
                "field1": "Текст, отправленный в уведомлении"
            },
            "created": 1760017227631,
            "isRead": false,
            "render": "Содержимое уведомления: Текст, отправленный в уведомлении"
        }
    ],
    "total": 1
}
```

4.2.7.2. Тестирование создания раздела ЛК для уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createSectionPp' \
--header 'Content-Type: application/json' \
--data '{
   "id": "personalPageSectionId",
   "name": "Personal page section",
   "version": 1
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"personalPageSectionId"
```

4.2.7.3. Тестирование получения полного списка разделов ЛК для уведомлений

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/listSectionPp'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "exampleSectionId",
      "name": "Example section",
      "version": 1
   }
]
```

4.2.7.4. Тестирование создания настройки уведомления в ЛК

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createSettingPp' \
--header 'Content-Type: application/json' \
--data '{
   "id": "personalPageSettingId",
   "sectionId": "exampleSectionId",
   "version": 1
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"personalPageSettingId"
```

4.2.7.5. Тестирование получения полного списка настроек уведомлений в ЛК

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/listSettingPp'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "exampleSettingId",
      "sectionId": "exampleSectionId",
      "version": 1
   }
]
```

4.2.7.6. Тестирование создания шаблона для настройки уведомления в ЛК

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location 'http://verdi.local/api/send/sync/createTemplatePp' \
```

```
--header 'Content-Type: application/json' \
--data '{
   "id": "personalPageTemplateId",
   "settingId": "exampleSettingId",
   "title": "Personal page template",
   "content": "<b>Personal page content</b>: {{field1}}",
   "version": 1
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"personalPageTemplateId"
```

4.2.7.7. Тестирование получения полного списка шаблонов для настройки уведомлений в ЛК

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/listTemplatePp''
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
[
   {
      "id": "exampleTemplateId",
      "settingId": "exampleSettingId",
      "title": "Personal page example template",
      "content": "<b>Example personal page content</b>: {{field1}}",
      "version": 1
   }
]
```

4.2.8. Data-model: сервис модели данных

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.8.1. Тестирование обновления модели данных

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/createDataModel' \
--header 'Content-Type: application/json' \
--data '[
   {
      "operation": "CreateEntity",
      "entityType": "demonstrationEntity",
      "title": "Демонстрационная сущность",
      "description": "Объекты для демонстраций",
      "settings": {}
   }
]'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.8.2. Тестирование получения полного списка сущностей модели данных

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/listDataModels'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
      "id": "businessProcess",
      "title": "Бизнес-процесс",
      "description": "Различные бизнес-процессы компании",
      "settings": {}
}
```

4.2.8.3. Тестирование создания объекта сущности

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/saveObject' \
--header 'Content-Type: application/json' \
--data '{
   "entityType": "businessProcess",
   "source": "manual",
   "attributes": {
      "businessProcessTitle": "Управление цепочкой поставок"
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданного объекта в виде строки формата UUID.

4.2.8.4. Тестирование получения списка объектов сущности с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/listObjects' \
--header 'Content-Type: application/json' \
--data '{
   "entityType": "businessProcess",
   "search": {
      "query": "id",
      "context": {
         "id": {
            "values": [
               "93eec27f-331b-4cb2-a398-f191bb3aec29"
            ],
            "kind": "any",
            "negation": false
         }
```

```
    },
    "sorting": {
      "fieldName": "created",
      "order": "asc"
    },
    "paging": {
      "page": 1,
      "count": 10
    }
  }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
    "items": [
      {
            "id": "93eec27f-331b-4cb2-a398-f191bb3aec29",
            "externalId": null,
            "entityType": "businessProcess",
            "title": "businessProcess 93eec27f-331b-4cb2-a398-
f191bb3aec29",
            "source": "manual",
            "created": 1760017061145,
            "modified": 1760017061145,
            "data": {
                  "businessProcessTitle": "Обработка заказа клиента"
            },
            "additionalData": {},
            "metadata": {},
            "version": 1,
            "removed": false
      }
    ],
    "total": 1
}
```

4.2.9. Erika: сервис печатных форм

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.9.1. Тестирование конвертации файла в PDF

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/convertToPdf' \
--header 'Content-Type: application/json' \
--data-binary '"datamodel/document/83df54b4-636c-40e9-87d6-294c0a5ecda7"'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"pdf/<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданного файла с итогом конвертации в виде строки формата UUID.

4.2.9.2. Тестирование создания отчета (createReport)

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/createReport' \
--header 'Content-Type: application/json' \
--data '{
   "mode": "single",
   "name": "reportGeneratedByTemplate",
   "templateFile": "datamodel/document/2ba12482-d024-4726-87bd-3762c2926744",
   "json": {
      "documentName": "Порядок оформления отпусков",
      "documentType": "Регламент",
      "documentDescription": "Данный документ регламентирует последовательность действий сотрудника и компании для официального оформления ежегодного оплачиваемого отпуска - от подачи заявления до издания приказа"
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"report/<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданного отчета в виде строки формата UUID.

4.2.10. Indexation: сервис индексаций

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.10.1. Тестирование создания описания индексации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/createIndexationDescription' \
--header 'Content-Type: application/json' \
--data '{
   "id": "businessProcess",
   "steps": []
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"businessProcess"
```

4.2.10.2. Тестирование получения описания индексации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/getIndexationDescription' \
--header 'Content-Type: application/json' \
--data-binary '"document"'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "id": "document",
      "steps": [
        {
              "children": [],
              "payload": {
```

```
                        "filesRoutes": [
                                {
                                        "field": "documentFile",
                                        "saveContentTo": "documentFile_content",
                                        "actions": [
                                                {
                                                        "lampRoute": "Plagiarism",
                                                        "routeResult": null
                                                },
                                                {
                                                        "lampRoute": "Similarity",
                                                        "routeResult": null
                                                }
                                        ],
                                        "fieldSearch": null
                                }
                        ],
                        "type": "Lamp"
                },
                "extensionFilter": null,
                "label": ""
          }
        ],
        "version": 1
}
```


4.2.10.3. Тестирование запуска индексации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/startIndexation' \
--header 'Content-Type: application/json' \
--data-binary '"document"'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор запущенной индексации в виде строки формата UUID.

4.2.11. Integration: сервис интеграций

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.11.1. Тестирование получения полного списка клиентов

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/integration_ListClients'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
	"id": "externalClient",
	"title": "Внешний клиент",
	"token": "TR2Efj-FRvesMT588-
7x?ZOSpRidGIIbkYnkdGJsgsTV_0H7z6KO82F7c1O9zmjy",
	"allowFiles": true,
	"data": null,
	"blocked": false
}
```

4.2.11.2. Тестирование получения полного списка доступных типов данных

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/integration_ListEntityTypes'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
	"id": "document",
	"title": "Документ",
	"description": "Различные документы"
```

```
}
```

4.2.11.3. Тестирование предоставления клиенту доступа к типу данных

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/integration_AllowEntityType' \
--header 'Content-Type: application/json' \
--data '{
   "clientId": "integrationSystem",
   "entityTypeId": "document"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.11.4. Тестирование получения полного списка типов данных доступных для клиента

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/integration_ListEntityTypesForClient' \
--header 'Content-Type: application/json' \
--data-binary '"externalClient"'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
      "id": "document",
      "title": "Документ",
      "description": "Различные документы"
}
```

4.2.12. Mon: сервис аутентификации

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.12.1. Тестирование регистрации пользователя

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_RegisterUser' \
--header 'Content-Type: application/json' \
--data-raw '{
   "email": "test@embedika.ru",
   "password": "js5;Z` r!?#33GygW",
   "info": {
      "firstName": "Михаил",
      "lastName": "Михайлов",
      "middleName": "Михайлович"
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "userId": "<сгенерированный_UUID>",
      "verificationRequested": true
}
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор зарегистрированного пользователя в виде строки формата UUID.

4.2.12.2. Тестирование аутентификации по логину и паролю

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_LoginByCredentials' \
--header 'Content-Type: application/json' \
--data-raw '{
```

```
    "email": "user@embedika.ru",
    "password": "js5;Z` r!?#33GygW"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
null
```

4.2.12.3. Тестирование изменения статуса пользователя

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_ChangeUserStatus' \
--header 'Content-Type: application/json' \
--data-raw '{
    "userId": "c04f6e3a-f3f7-47b3-83ef-75e166a73ab2",
    "status": "Activated"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
        "id": "c04f6e3a-f3f7-47b3-83ef-75e166a73ab2",
        "email": "maximov@embedika.ru",
        "status": "Activated",
        "info": {
          "firstName": "Максим",
          "lastName": "Максимов",
          "middleName": "Максимович",
          "data": {}
        },
        "modified": <timestamp>,
        "modifiedBy": "<id_пользователя>",
        "reason": null,
        "deactivateAt": null
}
```

где <timestamp> — зафиксированная сервисом временная метка момента изменения статуса пользователя, а <id_пользователя> — идентификатор

пользователя (или анонимного пользователя), инициировавшего изменение статуса.

4.2.12.4. Тестирование создания группы пользователей

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_CreateGroup' \
--header 'Content-Type: application/json' \
--data '{
   "id": "experts",
   "name": "Эксперты",
   "description": "Группа пользователей, проводящих экспертную
проверку"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "id": "experts",
      "name": "Эксперты",
      "modified": <timestamp>,
      "description": "Группа пользователей, проводящих экспертную
проверку"
}
```

где <timestamp> - зафиксированная сервисом временная метка момента создания группы пользователей.

4.2.12.5. Тестирование получения списка групп пользователей с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_ListGroups' \
--header 'Content-Type: application/json' \
--data '{
   "query": "groupId",
   "context": {
```

```
      "groupId": {
        "values": [
          "businessAdmins"
        ],
        "kind": "any",
        "negation": false
      }
    },
    "sorting": {
      "fieldName": "modified",
      "order": "desc"
    },
    "paging": {
      "page": 1,
      "count": 10
    }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "items": [
        {
              "group": {
                    "id": "businessAdmins",
                    "name": "Бизнес-администраторы",
                    "modified": 1760005621380,
                    "description": "Пользователи с ограниченным
доступом к управлению Системой"
              },
              "usersCount": 0
        }
      ],
      "total": 1
}
```

4.2.12.6. Тестирование добавления пользователей в группы

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_AddUsersToGroup' \
```

```
--header 'Content-Type: application/json' \
--data '{
   "groupIds": [
      "admins"
   ],
   "users": [
      "c04f6e3a-f3f7-47b3-83ef-75e166a73ab2"
   ]
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.12.7. Тестирование получения списка пользователей с информацией о группах с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/mon_http_ListUsersWithGroups' \
--header 'Content-Type: application/json' \
--data '{
   "query": "userId",
   "context": {
      "userId":{
         "values": [
            "1912826f-a12b-4470-a5e9-1c961ea7d731"
         ],
         "kind": "any",
         "negation": false
      }
   },
   "sorting": {
      "fieldName": "email",
      "order": "asc"
   },
   "paging": {
      "page": 1,
      "count": 10
   }
```

```
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
	"items": [
	 {
		"user": {
			"id": "1912826f-a12b-4470-a5e9-1c961ea7d731",
			"email": "user@embedika.ru",
			"status": "Activated",
			"info": {
				"firstName": "Александр",
				"lastName": "Александров",
				"middleName": "Александрович",
				"data": {}
			},
			"modified": <timestamp>,
			"modifiedBy": "00000000-0000-0000-0000-
000000000000",
			"reason": null,
			"deactivateAt": null
		},
		"groups": [
			{
				"groupId": "users",
				"name": "пользователи"
			}
		]
	 }
	],
	"total": 1
}
```

где <timestamp> - зафиксированная сервисом временная метка момента последнего изменения пользователя.

4.2.13. Nestor: сервис журналирования

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.13.1. Тестирование создания события

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/createEvent' \
--header 'Content-Type: application/json' \
--data '{
   "eventDate": 1788238800000,
   "eventType": "exampleEventType",
   "userId": "00000000-0000-0000-0000-000000000000",
   "eventCategory": "Success",
   "application": "alexandrina",
   "data":
"{\"eventType\":\"exampleEventType\",\"payload\":\"example\",\"result\":true,\
"data\":null}",
   "dependencies": []
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
   "result": "success"
}
```

4.2.13.2. Тестирование получения списка событий с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff 'http://verdi.local/api/send/sync/searchEvents' \
--header 'Content-Type: application/json' \
--data '{
   "query": "id",
   "context": {
      "id": {
         "values": [
            "745c9f84-c909-40e6-ae91-45dd323e83b4"
         ],
         "kind": "any",
         "negation": false
      }
   },
   "sorting": {
      "fieldName": "eventDate",
      "order": "desc"
```

```
  },
  "paging": {
    "page": 1,
    "count": 5
  }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:


```
{
        "items": [
          {
                  "id": "745c9f84-c909-40e6-ae91-45dd323e83b4",
                  "eventDate": 1759997521833,
                  "storeDate": 1759997533188,
                  "eventType": "journalRegister",
                  "userId": "1912826f-a12b-4470-a5e9-1c961ea7d731",
                  "eventCategory": "Success",
                  "application": "Mon",
                  "data":
"{\"eventType\":\"journalRegister\",\"payload\":{\"email\":\"user@embedika.ru
\",\"info\":{\"firstName\":\"Александр\",\"lastName\":\"Александров\",\"middl
eName\":\"Александрович\",\"data\":null}},\"result\":{\"userId\":\"1912826f-
a12b-4470-a5e9-1c961ea7d731\",\"verificationRequested\":true},\"data\":null}",
                  "dependencies": [],
                  "renderedData": "Попытка регистрации пользователя
Александр Александров\n<div><b></b></div>\n"
          }
        ],
        "total": 1
}
```


4.2.14. Oberto: сервис системы прав

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.14.1. Тестирование создания версии политики

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/oberto_UpdatePolicyV2' \
--header 'Content-Type: application/json' \
--data '{
   "policyId": "Compare files",
   "version": "0.0.1",
   "priority": 10,
   "description": "Права для раздела \"Сравнение файлов\"",
   "rules": [
      {
         "ruleId": "ViewCompare",
         "priority": 1,
         "rule": "permit Actions(ViewCompare) if resource.entityType ==
'\"AdministrationSection'\" && anyOf User.groupId == anyOf
['\"Anonymous'\",'\"users'\",'\"businessAdmins'\",'\"admins'\"]",
         "filters": null
      }
   ]
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.14.2. Тестирование получения полного списка политик

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/oberto_ListPolicyRecordsV2'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
      "id": "default_AllowEverything",
      "priority": 1,
      "versions": [
        "0.0.1"
      ]
```

```
}
```

4.2.14.3. Тестирование проверки доступа

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/oberto_AuthorizeV2' \
--header 'Content-Type: application/json' \
--data '{
   "resourceType": "document",
   "actions": [
      "DataModel_SaveObject"
   ]
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "DataModel_SaveObject": true
}
```

4.2.15. Attila: сервис композитных правил

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.15.1. Тестирование получения списка команд с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/attila_ListCommandAuthorizationInfo' \
--header 'Content-Type: application/json' \
--data '{
   "query": "commandname",
   "context": {
      "commandname": {
         "values": [
```

```
            "createDataModel"
          ],
          "kind": "any",
          "negation": false
        }
    },
    "sorting": {
        "fieldName": "ruleAction",
        "order": "asc"
    },
    "paging": {
        "page": 1,
        "count": 10
    }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
        "items": [
          "createDataModel"
        ],
        "total": 1
}
```

4.2.15.2. Тестирование получения списка ресурсов с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/attila_ListResourceTypeAuthorizationInfo' \
--header 'Content-Type: application/json' \
--data '{
    "query": "",
    "context": {},
    "sorting": {
        "fieldName": "ruleAction",
        "order": "asc"
    },
    "paging": {
        "page": 1,
        "count": 100
```

```
    }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
        "items": [
          "",
          "Constant",
          "EntityFormSettings",
          "GlobalSetting",
          "Group",
          "indexation",
          "indexationDescription",
          "PersistFileDTO",
          "RequestDependent",
          "row",
          "System",
          "User"
        ],
        "total": 12
}
```

4.2.15.3. Тестирование получения списка действий с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/attila_ListRuleActionAuthorizationInfo' \
--header 'Content-Type: application/json' \
--data '{
   "query": "commandname",
   "context": {
      "commandname": {
         "values": [
            "createDataModel"
         ],
         "kind": "any",
         "negation": false
      }
   },
   "sorting": {
```

```
      "fieldName": "ruleAction",
      "order": "asc"
   },
   "paging": {
      "page": 1,
      "count": 10
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
      "items": [
        "dataModel_RequestDependent_CreateModel"
      ],
      "total": 1
}
```

4.2.16. Tech-constants: сервис технического справочника

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.16.1. Тестирование добавления или обновления сервиса

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/techConstants_UpsertService' \
--header 'Content-Type: application/json' \
--data '{
   "code": "exampleService",
   "name": "Пример сервиса",
   "description": "Любой сервис (внешний или один из сервисов Verdi)"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.16.2. Тестирование получения полного списка сервисов

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/techConstants_ListServices'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
        "code": "attila",
        "name": "Сервис композитных правил",
        "description": "Сервис для работы с композитными правилами
ограничения доступа"
}
```

4.2.16.3. Тестирование добавления или обновления типа констант

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/techConstants_UpsertConstantType' \
--header 'Content-Type: application/json' \
--data '{
    "code": "exampleConstantType",
    "name": "Пример типа констант",
    "description": "Логическая группировка констант по смыслу"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.16.4. Тестирование полного списка типов констант

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/techConstants_ListConstantTypes'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
        "code": "journalEventType",
        "name": "Тип события в журнале",
        "description": "Название и описание типа события в журнале"
}
```

4.2.16.5. Тестирование добавления или обновления ручной константы

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/techConstants_UpsertManualConstant' \
--header 'Content-Type: application/json' \
--data '{
    "code": "constant1",
    "constantType": "journalEventType",
    "service": "attila",
    "name": "Ручная константа 1",
    "description": "Описание ручной константы",
    "settings": {
        "additionalInfo": "Manual constant 1"
    }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.16.6. Тестирование получения списка констант с поддержкой фильтрации

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/techConstants_ListAllConstants' \
--header 'Content-Type: application/json' \
--data '{
   "query": "code && constantType",
   "context": {
      "code": {
         "values": [
            "createCatalog"
         ],
         "kind": "any",
         "negation": false
      },
      "constantType": {
         "values": [
            "journalEventType"
         ],
         "kind": "any",
         "negation": false
      }
   },
   "sorting": {
      "fieldName": "code",
      "order": "desc"
   },
   "paging": {
      "page": 1,
      "count": 5
   }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
{
       "items": [
         {
                  "code": "createCatalog",
                  "constantType": "journalEventType",
                  "service": "alexandrina",
                  "name": "Создание справочника",
                  "description": null,
                  "settings": {
```

```
                    "template": "<table>\n    <tr>\n        <td>Код</td>\n
<td>{{code}}</td>\n    </tr>\n    <tr>\n        <td>Название</td>\n
<td>{{title}}</td>\n    </tr>\n</table>\n"
                },
                "source": "Auto",
                "modified": <timestamp>
            }
        ],
        "total": 1
}
```

где <timestamp> - зафиксированная сервисом временная метка момента последнего изменения константы.

4.2.17. UI-storage: сервис хранения UI

Для проверки функционирования сервиса достаточно убедиться в корректной работе нескольких ключевых команд.

4.2.17.1. Тестирование получения полного списка типов форм

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
'http://verdi.local/api/send/sync/UIStorage_http_ListFormTypes'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
[
  {
    "code": "card",
    "title": "Карточка объекта",
    "description": "Карточка объекта"
  },
  {
    "code": "createForm",
    "title": "Форма создания",
    "description": "Форма создания"
  },
  {
    "code": "editForm",
    "title": "Форма редактирования",
```

```
    "description": "Форма редактирования"
  },
  {
    "code": "list",
    "title": "Реестр",
    "description": "Элемент списка в реестре объектов"
  },
  {
    "code": "filterPanel",
    "title": "Панель фильтров",
    "description": "Панель фильтров"
  }
]
```

4.2.17.2. Тестирование создания формы

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/UIStorage_kafka_CreateForm' \
--header 'Content-Type: application/json' \
--data '{
  "entityType": "businessProcess",
  "formType": "card",
  "formData": {
    "field1": "value1"
  }
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
"<сгенерированный_UUID>"
```

где <сгенерированный_UUID> - сгенерированный сервисом идентификатор созданной формы в виде строки формата UUID.

4.2.17.3. Тестирование получения полного списка форм

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff --request POST
```

```
'http://verdi.local/api/send/sync/UIStorage_http_ListForms'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:

```
{
        "id": "b47ccc6c-9abe-435a-93b4-72d31a5aa250",
        "entityType": "document",
        "formType": {
          "code": "card",
          "title": "Карточка объекта"
        }
}
```

4.2.17.4. Тестирование создания глобальной настройки

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --globoff
'http://verdi.local/api/send/sync/UIStorage_kafka_CreateKey' \
--header 'Content-Type: application/json' \
--data '{
   "key": "exampleColors",
   "value": {
      "color-bg": "#ffffff",
      "color-text": "#fcbe04"
   },
   "title": "Цвет"
}'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

```
true
```

4.2.17.5. Тестирование получения полного списка глобальных настроек

Используемые для тестирования данные:

- Готовый cURL запроса:

```
curl --location --request POST
'http://verdi.local/api/send/sync/UIStorage_http_ListKeys'
```

Для проведения теста необходимо повторить все шаги, описанные в разделе «Шаги для выполнения теста».

Планируемый ответ в поле «value»:

Список объектов, включающий как минимум следующий объект:


```
{
	"key": "user-colors",
	"value": [
	  {
		"color": "rgba(255, 55, 254, 1)"
	  }
	],
	"title": "Пользовательские цвета",
	"created": 1759995092985,
	"modified": 1759995092985,
	"lastUser": "00000000-0000-0000-0000-000000000000",
	"version": 1
}
```