МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

ИНФОРМАЦИОННАЯ СИСТЕМА СБОРА И АНАЛИЗА ДАННЫХ ОБ ОБЩЕОБРАЗОВАТЕЛЬНЫХ УЧРЕЖДЕНИЯХ

Студент 8.06.2024 г. Горынина Анастасия Валерьевна
(подпись, дата) (ФИО полностью)

Руководитель 8.06.2024 г. Синев М.П.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 8.06.2024 г. Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент 10.06.2024 Мартыненко А.И. зав.каф. Программирование
(подпись, дата) (фамилия, инициалы) (должность, место работы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой _____ Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 14.06.24 № 19)

Секретарь ГЭК _____ Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«13» сентября 2022г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Горынина Анастасия Валерьевна гр.22ВВИм1 факультета ВТ

2.Тема работы «Информационная система сбора и анализа данных об общеобразовательных учреждениях»

Тема утверждена приказом ПГУ №765/о от «13» сентября 2022 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры ВТ Синев М.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Проектирование и разработка библиотеки, реализующей функционал поиска в нереляционной базе данных в существующей системе сбора и анализа данных об общеобразовательных учреждениях. Программное обеспечение должно отвечать требованиям спецификации FHIR. Входными данными методов библиотеки будут URL-адреса с query params в составе. Должны быть предоставлены следующие возможности: предоставлять сведения об общеобразовательных организациях, реализовывать быстрый поиск по критериям в нереляционной документоориентированной базе данных под управлением СУБД MongoDB, Необходимо реализовать валидацию входных параметров для исключения ошибок, связанных с человеческим фактором.

5. Перечень подлежащих разработке вопросов
Требуется провести анализ предметной области: изучить спецификацию FHIR и способы реализации быстрого поиска в нереляционных базах данных, рассмотреть входные (URL-адреса) и агрегирующие MongoDB запросы в контексте грамматики, спроектировать средства трансляции. Изучить вспомогательные технологии.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	15%	13.09.2022 – 16.12.2022	
2	Анализ требований	10%	17.12.2022 – 14.02.2023	
3	Выбор технологий	4%	15.02.2023 – 06.03.2023	
4	Проектирование транслятора	30%	07.03.2023 – 01.10.2023	
5	Разработка функционала библиотеки	25%	02.10.2023 – 01.03.2024	
6	Тестирование и отладка	10%	02.03.2024 – 03.05.2024	
7	Оформление пояснительной записки	6%	04.05.2024 – 31.05.2024	

Дата выдачи задания «13» сентября 2022 г.

Руководитель магистерской диссертации _____ 13.09.2022г
(подпись, дата)

Задание к исполнению принял студент _____ 13.09.2022г
(подпись, дата)

Работу к защите допустить
Декан факультета _____ А.Р. Ризаков
11.06.2024 *(подпись, дата)*

РЕФЕРАТ

Работа содержит 120 листов, 5 рисунков, 45 таблиц, 8 источников, 2 приложения

КОНТЕКСТНО-СВОБОДНАЯ ГРАММАТИКА, ПОИСК, ПРОДУКЦИОННАЯ МОДЕЛЬ ЗНАНИЙ, ПРОДУКЦИОННЫЕ ПРАВИЛА ВЫВОДА, ТРАНСЛЯТОР, ЦИФРОВИЗАЦИЯ ГОСУДАРСТВЕННЫХ УСЛУГ, JAVA, FHIR, MONGODB, REST

Цель работы – проектирование и реализация механизма поиска экземпляров ресурсов в нереляционной базе данных под управлением СУБД MongoDB в соответствии со спецификацией FHIR.

Объектом разработки является библиотека, обеспечивающая основную систему необходимым функционалом поиска по критериям.

В процессе выполнения работы был проведен анализ поисковых запросов и запросов к базе данных MongoDB с точки зрения грамматики, выявлены правила продукции, разработана библиотека поиска на основе продукционной модели знаний.

В результате выполнения работы была создана библиотека, отвечающая предъявляемым к ней требованиям.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 4
ВВЕДЕНИЕ 5
1 Анализ предметной области 8
1.1 Описание системы учета данных об общеобразовательных учреждениях 8
1.2 Поиск в рамках спецификации FHIR 9
1.2.1 Тип параметра String 10
1.2.2 Тип параметра Token 10
1.2.3 Тип параметра Reference 12
1.2.4 Тип параметра Date или Datetime 12
1.2.5 Обработка ошибок 13
1.3 Параметры поиска системы 14
1.3.1 Общие параметры 14
1.3.2 Специальные параметры 15
1.3.3 Параметры, определяющие результат поиска 24
1.4 Теоретическое обоснование транслятора 25
1.5 Правила трансляции 32
2 Разработка машины вывода 56
3 Тестирование 66
3.1 Позитивное функциональное тестирование 66
3.2 Негативное функциональное тестирование 81
ЗАКЛЮЧЕНИЕ 83
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 84
ПРИЛОЖЕНИЕ А Листинг программы 85
ПРИЛОЖЕНИЕ Б Презентация 114

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяют следующие сокращения и обозначения

ЕГРИП – Единый государственный реестр индивидуальных предпринимателей

ЕГРЮЛ – Единый государственный реестр юридических лиц

Рособрнадзор – Федеральная служба по надзору в сфере образования и науки

СУБД – Система управления базами данных

API (Application Programming Interface) – интерфейс программирования приложения

FHIR (Fast Healthcare Interoperability Resources) – ресурсы быстрого взаимодействия в сфере здравоохранения; название стандарта HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

JSON (JavaScript Object Notation) – текстовый формат обмена данными

REST (REpresentational State Transfer) – передача репрезентативного состояния; архитектурный стиль взаимодействия компонентов распределённого приложения в сети

XML (eXtensible Markup Language) – расширяемый язык разметки; текстовый формат обмена данными

UML (Unified Modeling Language) — унифицированный язык моделирования

URI (Uniform Resource Identifier) – унифицированный (единообразный) идентификатор ресурса

URL (Uniform Resource Locator) – единообразный указатель местонахождения ресурса

ВВЕДЕНИЕ

В современном постиндустриальном обществе информационные технологии стали одним из инструментов повышения эффективности государственного и муниципального управления. Активно развиваются технологии управления бюджетной, налоговой системами, технологии государственных информационных порталов, реализующих взаимодействие граждан и органов управления через компьютерные сети, технологии накопления информации в единых государственных базах данных, технологии муниципальных информационных систем, информационных систем жилищно-коммунальной сферы (в направлении создания единых расчетно-кассовых центров), геоинформационных систем (в направлении создания земельных и городских кадастров), электронного документооборота.

Главным российским ИТ-проектом последних лет стал переход к оказанию государственных услуг в электронном виде. С 1 июля 2012 года все регионы и муниципалитеты начали постепенный переход на электронное межведомственное взаимодействие [1]. В рамках данного проекта было написано решение по информатизации сферы образовательных услуг – информационная система «Реестр организаций, осуществляющих образовательную деятельность».

Система учета данных об общеобразовательных учреждениях разработана на языке Java, фреймворк Spring [2,3], на основе спецификации FHIR. FHIR – это стандарт обмена данными в области здравоохранения, опубликованный компанией HL7 [4]. Спецификация FHIR описывает набор ресурсов и несколько различных фреймворков для обмена ресурсами между различными системами в форматах XML, JSON, RDF. Спецификация носит общий характер, за счет чего имеет широкую применимость. FHIR определяет множество достаточно общих типов данных для использования в ресурсах. Спецификация имеет полную общедоступную документацию: ознакомиться с составом полей типов данных Meta, Identifier, Reference, Coding и других использованных в системе типов возможно на официальном сайте стандарта.

FHIR описывается как RESTful спецификация, основанная на общепринятом на отраслевом уровне использовании термина REST [5]. FHIR основан на стандартизации структур ресурсов и интерфейсов, это может рассматриваться как нарушение принципов REST, но является ключом к обеспечению согласованной совместимости между различными системами.

Для каждого типа ресурса определен один и тот же набор операций, которые могут использоваться для управления ресурсами, в их числе поиск экземпляров ресурса по параметрам.

Цель исследования: создание механизма поиска экземпляров ресурсов в нереляционной базе данных под управлением СУБД MongoDB в соответствии со спецификацией FHIR.

Объект исследования: электронное взаимодействие государственных и региональных ведомств, в чьей юрисдикции полностью или частично находится сбор, анализ, обработка и хранение данных об организациях, осуществляющих образовательную деятельность.

Предмет исследования: механизм поиска в системе сбора и анализа данных об общеобразовательных учреждениях.

Задачи исследования:

– проанализировать входные и агрегирующие запросы в контексте грамматики,

– определить состав поисковых параметров системы, их класс по спецификации FHIR, ассоциированные поля,

– выявить правила трансляции в направлении от входного URL-запроса с поисковыми параметрами к агрегирующим запросам к СУБД MongoDB,

– спроектировать механизм трансляции,

– определить используемые технологии разработки,

– разработать и протестировать механизм поиска в составе системы сбора и анализа данных об общеобразовательных учреждениях.

Методы и подходы: для достижения поставленных целей в работе применяются методы системного анализа и объективности, метод комплексного исследования; объектно-ориентированное программирование на языке JAVA, применение фреймворков для быстрой разработки, использование паттернов проектирования.

Теоретическая и практическая значимость результатов: в процессе исследования были выведены продукционные правила трансляции с опорой на типы FHIR, типы ассоциированных полей, вложенность и кратность. Написан и отлажен механизм поиска, основанный на продукционной модели знаний. Решение обеспечит функционал поиска по критериям в нереляционной базе данных под управлением СУБД MongoDB в соответствии со спецификацией FHIR, будет интегрировано в систему сбора и анализа данных об общеобразовательных учреждениях и может быть использовано в иных системах с похожим стеком технологий.

1 Анализ предметной области

1.1 Описание системы учета данных об общеобразовательных учреждениях

Реестр представляет собой массив реестровых записей, каждая из которых описывает сведения об одной организации (филиале) или об одном индивидуальном предпринимателе. Концептуальная модель реестровой записи представлена в виде диаграммы объектов на языке UML. Между объектами установлены два типа связей: направленная композиция от головного класса ресурса к вспомогательному классу и направленные ассоциации (ссылки) на экземпляры других ресурсов. Зеленым цветом выделены экземпляры прикладных ресурсов, белые – экземпляры вспомогательных классов BackboneElement.

Центральный объектом является реестровая запись. Она содержит ссылку на общие сведения об организации или на демографические данные индивидуального предпринимателя. Экземпляр лицензии содержит ссылку на реестровую запись. Аналогичную ссылку содержат приложение к лицензии, свидетельство о государственной аккредитации образовательной программы, приложение к свидетельству, штатное расписание, место осуществления образовательной деятельности, ученические места. Приложение к лицензии содержит ссылку на лицензию, приложение к свидетельству содержит ссылку на свидетельство. Экземпляр ресурса Event идентифицирует основание создания или изменения одного или нескольких экземпляров ресурсов. Например, таким основанием может служить выписка из ЕГРЮЛ или ЕГРИП, приказ о выдаче лицензии и т.д.

Концептуальная модель реестровой записи приведена на рисунке 1.

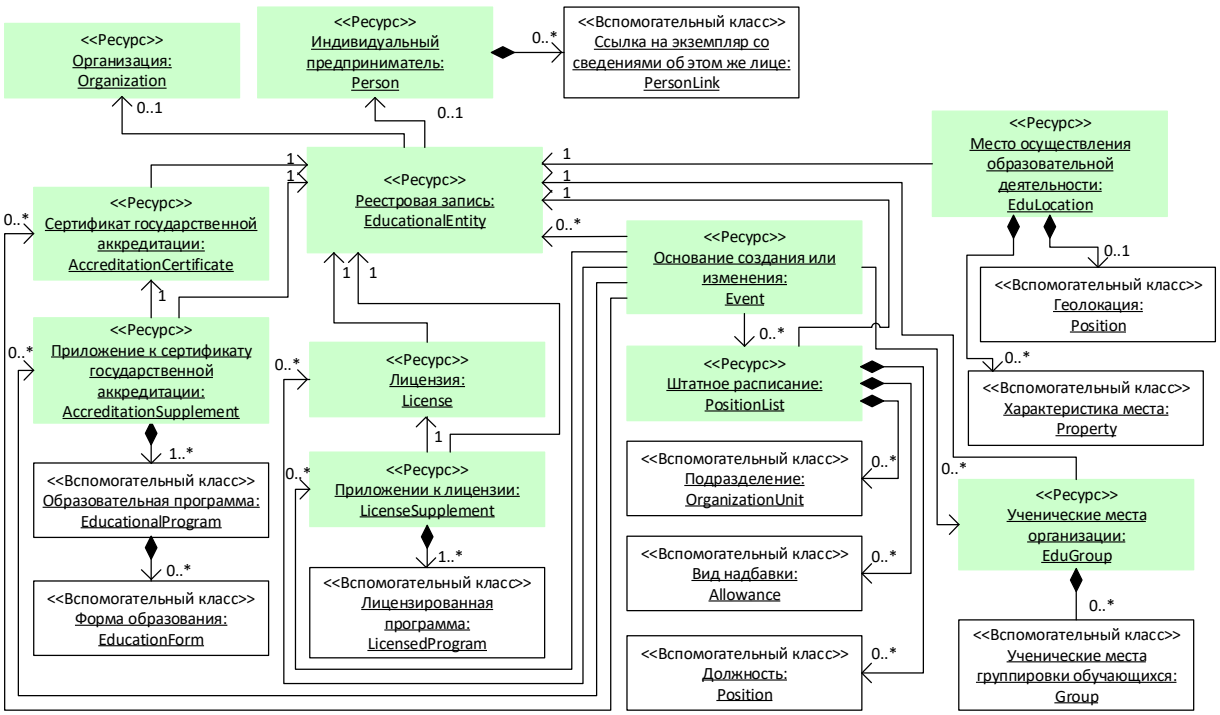

Рисунок 1 – Концептуальная модель реестровой записи

1.2 Поиск в рамках спецификации FHIR

Спецификация FHIR является RESTful стандартом. Выше были изложены поля одиннадцати ресурсов: EducationalEntity, Organization, Person, License, LicenseSupplement, AccreditationCertificate, AccreditationSupplement, PositionList, PupilPlaces, EduLocation, Event – поиск осуществляется GET-запросом к конечной точке ресурса:

[GET] baseurl/Resource{?name=value&...{&name=value}}

В query params запроса не передается ничего (поиск без ограничений) или заносятся пары параметров и значений вида name=value. Параметры в спецификации FHIR подразделяются на 3 основных категории: общие для всех ресурсов, специальные для ресурса и параметры, определяющие результат поиска. Общие и специальные параметры могут быть четырех основных типов: String, Token, Reference, Date/Datetime – тип в данном случает обозначает поведение сервера при обработке параметра. Параметры, определяющие результат поиска, влияют на отображение найденного массива документов: постраничный вывод, состав полей, дополнительно включаемые документы, настройки сортировки и другое.

1.2.1 Тип параметра String

Простой регистронезависимый поиск по строке. По умолчанию поле соответствует строковому запросу, если значение поля равно указанному значению параметра или начинается с него.

Запись параметра: [parameter]=[string], где string – любая цепочка символов.

Параметр строкового поиска может ссылаться на типы HumanName и Address из пространства имен “http://hl7.org/fhir”, в таком случае поиск охватывает элементы типа string и не охватывает такие элементы, как use и period. Для типа HumanName сервером должен быть осуществлен поиск по полям HumanName.family, HumanName.given, HumanName.middle, HumanName.text. Документ удовлетворяет условию, если хотя бы одно из значений приведенных полей начинается с значения параметра.

Аналогично для Address: поиск должен быть расширен на поля Address.city, Address.district, Address.postalCode, Address.street, Address.text.

Поведение сервера может изменяться модификаторами :contains и :exact. Модификатор :contains возвращает результаты, которые включают указанное значение параметра в любом месте искомого поля.

Запись параметра: [parameter]:contains=[string]

Модификатор :exact возвращает результаты, соответствующие всему предоставленному параметру, поиск чувствителен к регистру.

Запись параметра: [parameter]:exact=[string]

1.2.2 Тип параметра Token

Параметр обеспечивает поиск точного соответствия в строке символов, потенциально ограниченной URI. В основном он используется для типа данных Coding или Identifier из пространства имен “http://hl7.org/fhir”.

Тип Coding представляет собой кодированное значение из системы кодирования и кода, соответственно поиск осуществляется по паре Coding.system и Coding.code. Примером может служить регион в составе адреса, заполняемый кодом из справочника «Административный субъект

Российской Федерации (регион РФ)» (системой будет url https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Common-Ssrf-1).

Тип Identifier это цифровая или буквенно-цифровая строка, связанная с отдельным объектом или сущностью в данной системе. Как правило, идентификаторы используются для подключения содержимого ресурсов к внешнему контенту, доступному в других платформах или протоколах. Например, идентификатор реестровой записи составляется из пары Identifier.system=https://id.obrnadzor.gov.ru/school и Identifier.value=GUID, идентификатор присваивается Рособрнадзором.

Синтаксис параметра подразумевает четыре варианта обработки значений пары системы и кода и приведен в таблице 1.

Параметры типа Token также используются для строковых полей, где необходимы точные соответствия. В этих случаях часть URI не используется (первый вариант записи), поиск реализуется как регистрозависимый и, по сути, исполняется как если бы это был строковый параметр с модификатором :exact.

Логические значения также подразумевают точное соответствие, но поиск реализуется регистронезависимым.

Таблица 1 – Синтаксис параметра типа Token

Номер	Вариант записи пары	Реализуемое сервером условие истинности
1	[parameter]=[code]	Значение параметра [code] соответствует значению Coding.code или Identifier.value независимо от значения системного свойства
2	[parameter]=[system]|[code]	Значение параметра [code] соответствует значению Coding.code или Identifier.value, а значение параметра [system] соответствует системному свойству идентификатора или кодировки

Продолжение таблицы 1

Номер	Вариант записи пары	Реализуемое сервером условие истинности
3	[parameter]=|[code]	Значение параметра [code] соответствует значению Coding.code или Identifier.value, а системное свойство отсутствует
4	[parameter]=[system]|	Значение параметра [system] соответствует системному свойству идентификатора или кодировки, код или идентификатор не имеет значения

1.2.3 Тип параметра Reference

Параметр относится к ссылкам между ресурсами. Например, поиск всех реестровых записей, где предметом ссылки является конкретная организация, где организация выбрана по идентификатору.

Запись параметра: [parameter]=[id], где id – системный идентификатор базы данных MongoDB.

Поведение возможно изменить, применив модификатор :identifier, в этом случае сервер осуществит поиск по полю Reference.Identifier с процедурами обработки, соответствующими параметру типа Token.

Запись параметра:

[parameter]:identifier=[code]

[parameter]:identifier=[system]|[code]

[parameter]:identifier=|[code]

[parameter]:identifier=[system]|

1.2.4 Тип параметра Date или Datetime

Параметр выполняет поиск по дате или дате и времени. Формат даты – yyyy-MM-dd, формат даты и времени – yyyy-MM-dd’T’HH:mm:ss.SSS[Z|(+|-)hh:mm].

Запись параметра по умолчанию [parameter]=[datetime] подразумевает равенство, однако всего синтаксис параметра включает 12 вариантов обработки значений: временные диапазоны определяются префиксами ne, gt,

lt, ge, le и типом Date в качестве значения параметра – реализация описана в таблице 2.

Таблица 2 - Синтаксис параметра типа Date

Вариант записи	Реализуемое сервером условие истинности
[parameter]=[datetime] или [parameter]=eq[datetime]	Равенство значения параметра и поля документа
[parameter]=[date] или [parameter]=eq[date]	Параметр устанавливает период, состоящий из всех моментов времени этих суток, в который может попасть значение поля документа
[parameter]=ne[datetime]	Неравенство значения параметра и поля документа
[parameter]=ne[date]	Параметр устанавливает период, состоящий из всех моментов времени этих суток, в который не может попасть значение поля документа
[parameter]=le[datetime]	Поле документа является моментом времени ранее или равным datetime
[parameter]=le[date]	Поле документа является моментом времени ранее или равным последнему моменту времени в сутках date
[parameter]=ge[datetime]	Поле документа является моментом времени позднее или равным datetime
[parameter]=ge[date]	Поле документа является моментом времени позднее или равным первому моменту времени в сутках date
[parameter]=lt[datetime]	Поле документа является моментом времени ранее datetime
[parameter]=lt[date]	Поле документа является моментом времени ранее первого момента времени в сутках date
[parameter]=gt[datetime]	Поле документа является моментом времени позднее datetime
[parameter]=gt[date]	Поле документа является моментом времени позднее последнего момента времени в сутках date

1.2.5 Обработка ошибок

Если серверу не удается выполнить поисковый запрос, он должен вернуть ошибку: коды состояния HTTP 4xx и 5xx и сообщение об ошибке OperationOutcome с подробным описанием причины сбоя.

Стоит уточнить, что пустой результат поиска не является сбоем.

Сервер может получать параметры от клиента, которые он не распознает, или может получать параметры, которые он распознают, но не поддерживает (либо в целом, либо для конкретного ресурса). В таком случае сервер должен выдать ошибку 400 Bad Request.

1.3 Параметры поиска системы

1.3.1 Общие параметры

Параметры поиска, общие для всех ресурсов, описываются спецификацией FHIR, однако в системе учета данных об общеобразовательных учреждениях нет необходимости в реализации абсолютно всех общих параметров.

Описание общих параметров системы приведено в таблице 3.

Таблица 3 – Общие параметры поиска экземпляров ресурсов

Имя	Тип	Описание	Выражение	Тип поля документа
_id	Reference	Логический идентификатор экземпляра ресурса	Resource.id	String
_lastUpdated	Date	Дата и время последнего изменения экземпляра ресурса	Resource.meta.lastUpdated	LocalDateTime
_tag	Token	Тег, примененный к экземпляру ресурса	Resource.meta.tag	Coding[]
_profile	Token	Профиль экземпляра ресурса	Resource.meta.profile	String[]
_source	Token	Источник данных	Resource.meta.source	String

Параметры ориентированы на поля метаданных, включаемых в каждый документ, а потому для этой категории параметров можно использовать модификатор :missing. В таком случае в выводе должны быть документы, где ассоциированное с параметром поле отсутствует.

1.3.2 Специальные параметры

В системе учета данных об общеобразовательных организациях были выделены 11 ресурсов, для каждого ресурса выделены критерии поиска. Были выявлены: имя параметра, используемое в query params, тип параметра по спецификации FHIR, описание параметра – вкладываемый в параметр смысл, выражение и тип поля документа.

Выражение в общем смысле представлено ресурсами и полями документа, например, EducationalEntity.identifier, но может принимать вид условного выражения, к примеру, Person.telecom.where(system=’email’), тогда поле telecom.system в параметре token должно быть предзаполнено значением email. Также могут быть выражения вида Organization.address.city | Organization.address.district | Organization.address.postalCode | Organization.address.street | Organization.address.text, тогда условием истинности будет являться совпадение с любым из объявленных полей. У ссылочных параметров есть уточнение в скобках на какой тип ресурса (коллекцию) ссылается идентификатор в поле.

Тип поля документа указывает класс объекта или объектов, указанных в выражении. Представление в виде списка отмечается символами квадратными скобками.

Специальные параметры поиска экземпляров ресурсов системы приведены в таблице 4.

Таблица 4 – Специальные параметры поиска экземпляров ресурсов системы

Имя	Тип	Описание	Выражение	Тип поля документа
Ресурс EducationalEntity				
identifier	Token	Идентификатор реестровой записи	EducationalEntity.identifier	Identifier
entity	Reference	Идентификатор субъекта реестровой записи	EducationalEntity.entity (Organization или Person)	Reference
parent	Reference	Идентификатор реестровой записи	EducationalEntity.parent (EducationalEntity)	Reference

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
		головной организации, осуществляющей образовательную деятельность		
member	Reference	Идентификатор участника (учредителя)	EducationalEntity.member (Organization или Person)	Reference[]
predecessor	Reference	Идентификатор правопредшественника	EducationalEntity.predecessor (Organization)	Reference[]
status	Token	Статус реестровой записи	EducationalEntity.status	Enum
mainеconomicactivity	Token	Основной вид экономической деятельности	EducationalEntity.mainEconomicActivity	Coding
otherеconomicactivity	Token	Дополнительный вид экономической деятельности	EducationalEntity.otherEconomicActivity	Coding[]
Ресурс Organization				
active	Token	Поиск с целью проверки «Активна ли запись организации»	Organization.active	Boolean
address	String	Поиск, определяемый сервером, при котором совпадение ищется с любым строковым элементом объекта Address, включая city, district, postalCode, street и/или text	Organization.address.city | Organization.address.district | Organization.address.postalCode | Organization.address.street | Organization.address.text	String
address-city	String	Город, указанный в адресе	Organization.address.city	String
address-country	Token	Страна, указанная в адресе	Organization.address.country	Coding
address-postalcode	String	Почтовый индекс, указанный в адресе	Organization.address.postalCode	String

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
address-state	Token	Регион, указанный в адресе	Organization.address.state	Coding
address-use	Token	Код использования, указанный в адресе	Organization.address.use	Enum
identifier	Token	Любой идентификатор организации (но не идентификатор уполномоченного органа, присвоившего идентификатор)	Organization.identifier	Identifier[]
name	String	Часть наименования организации	Organization.name | Organization.alias	String
partof	Reference	Организация, частью которой является данная организация	Organization.partOf (Organization)	Reference
type	Token	Код типа организации	Organization.type	CodeableConcept[].Coding[]
executive name	String	Фамилия, имя, отчество руководителя	Organization.executiveName	String
executive position	String	Должность руководителя	Organization.executivePosition	String
Ресурс Person				
address	String	Поиск, определяемый сервером, при котором совпадение ищется с любым строковым элементом объекта Address, включая city, district, postalCode, street и/или text	Person.address.city | Person.address.district | Person.address.postalCode | Person.address.street | Person.address.text	String
address-city	String	Город, указанный в адресе	Person.address.city	String

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
address-country	Token	Страна, указанная в адресе	Person.address.country	Coding
address-postalcode	String	Почтовый индекс, указанный в адресе	Person.address.postalCode	String
address-state	Token	Регион, указанный в адресе	Person.address.state	Coding
address-use	Token	Код использования, указанный в адресе	Person.address.use	Coding
email	Token	Адрес электронной почты	Person.telecom.where(system=’email’)	ContactPoint[]
identifier	Token	Идентификатор лица	Person.identifier	Identifier[]
name	String	Поиск, определяемый сервером, при котором совпадение ищется с любым строковым элементом в объекте типа HumanName, включая фамилию (family), имя (give), отчество (middle) и/или текст (text)	Person.name.family | Person.name.give | Person.name.middle | Person.name.text	String
phone	Token	Номер телефона	Person.telecom.where(system=‘phone’)	ContactPoint[]
telecom	Token	Поиск по значению любой контактной информации	Person.telecom	ContactPoint[]
link	Reference	Поиск по ссылке на другой экземпляр Person со сведениями об этом же лице	Person.link.target (Person)	BackboneElement[].Reference
Ресурс License				
identifier	Token	Идентификатор лицензии	License.identifier	Identifier
start	Date	Дата начала действия лицензии	License.identifier.period.start	ZonedDateTime

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
end	Date	Дата прекращения действия лицензии	License.identifier.period.end	ZonedDateTime
assigner	String	Организация, присвоившая идентификатор	License.identifier.assignerDisplay	String
address	String	Местонахождение юридического лица или его филиала	License.address	String
mediaid	String	Серия и номер бланка или идентификатор электронной лицензии	License.mediaId	String
entity	Reference	Идентификатор реестровой записи лицензиата	License.educationalEntity (EducationalEntity)	Reference
replaces	Reference	Идентификатор заменяемой лицензии	License.insteadOfLicense (License)	Reference
type	Token	Тип лицензии	License.type	Coding
status	Token	Статус лицензии	License.status	Coding
Ресурс LicenseSupplement				
entity	Reference	Ссылка на реестровую запись лицензиата или его филиала	LicenseSupplement.educationalEntity (EducationalEntity)	Reference
identifier	Token	Идентификатор приложения	LicenseSupplement.identifier	Identifier
license	Reference	Ссылка на лицензию, к которой выдано приложение	LicenseSupplement.license (License)	Reference
location	String	Адрес места осуществления образовательной деятельности	LicenseSupplement.eduLocation	String[]
mediaid	String	Серия и номер бланка или идентификатор электронного приложения	LicenseSupplement.mediaId	String[]

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
status	Token	Статус приложения к лицензии	LicenseSupplement.status	Coding
level	Token	Уровень образования (лицензируемый вид образовательной деятельности)	LicenseSupplement.licensedProgram.level	BackboneElement[].Coding
Ресурс AccreditationCertificate				
identifier	Token	Идентификатор свидетельства	AccreditationCertificate.identifier	Identifier
start	Date	Дата начала действия свидетельства	AccreditationCertificate.identifier.period.start	ZonedDateTime
end	Date	Дата прекращения действия свидетельства	AccreditationCertificate.identifier.period.end	ZonedDateTime
assigner	String	Аккредитационный орган	AccreditationCertificate.identifier.assignerDisplay	Identifier.String
mediaid	String	Серия и номер бланка или идентификатор электронного свидетельства	AccreditationCertificate.mediaId	String
entity	Reference	Идентификатор реестровой записи субъекта свидетельства	AccreditationCertificate.educationalEntity (EducationalEntity)	Reference
replaces	Reference	Идентификатор заменяемого свидетельства	AccreditationCertificate.insteadOfCertificate (AccreditationCertificate)	Reference
status	Token	Статус свидетельства	AccreditationCertificate.status	Coding
Ресурс AccreditationSupplement				
entity	Reference	Идентификатор реестровой записи субъекта приложения к свидетельству	AccreditationSupplement.educationalEntity (EducationalEntity)	Reference
identifier	Token	Идентификатор приложения	AccreditationSupplement.identifier	Identifier

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
certificate	Reference	Идентификатор свидетельства об аккредитации	AccreditationSupplement.accreditationCertificate (AccreditationCertificate)	Reference
mediaid	String	Серия и номер бланка или идентификатор электронного свидетельства	AccreditationSupplement.mediaId	String[]
level	Token	Уровень образования (образовательная программа)	AccreditationSupplement.educationalProgram.level	BackboneElement[].Coding
status	Token	Статус приложения к свидетельству	AccreditationSupplement.status	Coding
program-status	Token	Статус образовательной программы	AccreditationSupplement.educationalProgram.status	BackboneElement[].Coding
Ресурс PositionList				
url	Token	Канонический идентификатор штатного расписания, представленный в форме URI	PositionList.url	String
version	String	Версия штатного расписания	PositionList.version	String
identifier	Token	Идентификатор штатного расписания	PositionList.identifier	Identifier
entity	Reference	Ссылка на реестровую запись субъекта штатного расписания	PositionList.educationalEntity(EducationalEntity)	Reference
academyear	Token	Академический год	PositionList.academYear	String
position-code	Token	Код должности	PositionList.position.code	BackboneElement[].Code
position-name	String	Наименование должности	PositionList.position.name	String

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
position-unit	Token	Код подразделения	PositionList.position.organizationUnit	BackboneElement[].Code
status	Token	Статус штатного расписания	PositionList.status	Enum
Ресурс EduGroup				
url	Token	Канонический идентификатор состава ученических мест, представленный в форме URI	EduGroup.url	String
version	String	Версия состава ученических мест	EduGroup.version	String
identifier	Token	Идентификатор состава ученических мест	EduGroup.identifier	Identifier
entity	Reference	Идентификатор реестровой записи субъекта состава ученических мест	EduGroup.educationalEntity(EducationalEntity)	Reference
academyear	Token	Академический год	EduGroup.academYear	String
group-type	Token	Тип группировки обучающихся	EduGroup.group.type	BackboneElement[].Coding
group-name	String	Наименование группировки	EduGroup.group.name	BackboneElement[].String
status	Token	Статус состава ученических мест	EduGroup.status	Enum
Ресурс EduLocation				
identifier	Token	Идентификатор местонахождения	EduLocation.identifier	Identifier[]
name	String	Наименование местонахождения	EduLocation.name | EduLocation.alias	String[]
custodian	Reference	Идентификатор организации, ответственной за	EduLocation.managingOrganization (Organization)	Reference

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
		предоставление и содержание местонахождения		
entity	Reference	Идентификатор реестровой записи субъекта, использующего местонахождение	EduLocation.educationalEntity (EducationalEntity)	Reference
physicaltype	Token	Физическая форма места осуществления образовательной деятельности	EduLocation.physicalType	CodeableConcept.Coding[]
address	String	Поиск, определяемый сервером, при котором совпадение ищется с любым строковым элементом объекта Address, включая city, district, postalCode, street и/или text	EduLocation.address.city | EduLocation.address.district | EduLocation.address.postalCode | EduLocation.address.street | EduLocation.address.text	String
address-city	String	Город	EduLocation.address.city	String
address-country	Token	Страна	EduLocation.address.country	Coding
address-postalcode	String	Почтовый индекс	EduLocation.address.postalCode	String
address-state	Token	Регион	EduLocation.address.state	Coding
status	Token	Статус доступности места осуществления образовательной деятельности	EduLocation.status	Enum
Ресурс Event				
type	Token	Тип события	Event.type	Coding

Продолжение таблицы 2

Имя	Тип	Описание	Выражение	Тип поля документа
reason	Token	Причина события	Event.reason	CodeableConcept.Coding[]
docid	Token	Идентификатор документа-основания для события	Event.docId	Identifier
docname	String	Наименование документа-основания для события	Event.docName	String
docdate	Date	Дата принятия документа-основания для события	Event.docDate	LocalDate
subject	Reference	Субъект события	Event.subject.subject (Licence, LicenceSupplement, EduGroup, PositionList, EducationalEntity, AccreditationCertificate, AccreditationSupplement)	BackboneElement[].Reference

Параметры ориентированы на конкретные поля документов, и потому для этой категории параметров можно использовать модификатор :missing. В таком случае в выводе должны быть документы, где ассоциированное с параметром поле отсутствует.

1.3.3 Параметры, определяющие результат поиска

Параметры _count и _offset определяют настройки постраничного вывода, если не передаются в запросе, то выставляются значения по умолчанию _count=10 и _offset=0.

Параметр _sort определяет сортировку. Сортировка осуществляется только по полям, связанным с поисковыми параметрами. Сортировка по убыванию включает знак минус перед именем параметра. Пример записи:

_sort=[(-)parameter{,(-)parameter,…,(-)parameter}].

Параметры _include и _revinclude включают в вывод документы из других коллекций, и строятся на основе ссылок Reference. Параметр _include включает в вывод документы, на которые ссылается найденный документ исходной коллекции:

_include=[baseResource]:[referenceParameterOfBaseResource]

Параметр _revinclude включает в вывод документы, которые ссылаются на найденные документы исходной коллекции:

_revinclude=[anotherResource]:[referenceParameterOfAnotherResource]

Параметр _elements определяет состав полей документа. Значениями параметра будут поля документа, которые необходимо включить в вывод: _elements=[field{,field,…,field}]. Системный идентификатор документа выводится всегда.

Параметр _format диктует формат вывода: json или xml. В отсутствие параметра вывод будет осуществлен в формате json.

1.4 Теоретическое обоснование транслятора

Каждый язык – это множество цепочек символов над некоторыми алфавитом. Определяя алфавит, конкретизируем множество допустимых символов. Кроме алфавита язык предусматривает задание правил построения допустимых цепочек, поскольку обычно далеко не все цепочки над алфавитом принадлежат языку.

Слово или лексическая единица (лексема) языка – эта конструкция, которая состоит из элементов алфавита языка и не содержит в себе других конструкций. И те, и другие в общем виде являются цепочками символов, но предусматривают некоторые правила построения. Таким образом, необходимо указать эти правила – задать язык [6]. Поскольку перечислить все допустимые цепочки языка не представляется возможным, то зададим язык указанием способа порождение цепочки языка (заданием грамматики языка).

Формально грамматика G определяется как четверка G(VT, VN, P, S), где:

VT – множество терминальных символов;

VN – множество нетерминальных символов $VN \cap VT = \emptyset, VN \cup VT = V$;

P – множество правил (продукций) вида $\alpha \to \beta$, где $\alpha \in V^+, \beta \in V^*$;

S – целевой (начальный) символ грамматики $S \in VN$.

Грамматика запроса с поисковыми параметрами Gu представляет собой следующую запись.

Gu({a, b, c .. z, A, B, C .. Z, /, -, _, |, 0, 1, 2 .. 9, =, &, @}, {Query, Pair, Sort_value, Elements_value, Boolean_parameter, Parameter, Value, Uri_parameter, Time_parameter, Sort_parameter, Format_parameter, Elements_parameter, Page_parameter, String_parameter, Include_parameter, Include_value, Reference_parameter, Time_value, Format_value, Page_value, String_value, Boolean_value, Reference_value, Uri_value, Word, Symbol, Word16, Symbol16, Datetime, Timemodifier, Time, Date, Y, M, D, I60, I24, I1000, Number, Digit}, Pu, Query)

Pu:

Query -> Pair | Query&Pair | λ

Pair -> Uri_parameter=Uri_value | Time_parameter=Time_value | Sort_parameter=Sort_value | Format_parameter=Format_value | String_parameter=String_value | Boolean_parameter=Boolean_value | Include_parameter=Include_value | Reference_parameter=Reference_value | Elements_parameter=Elements_value | Page_parameter=Page_value

Sort_value -> Parameter | -Parameter

Elements_value -> Parameter

Boolean_parameter -> Parameter:missing

Parameter -> Uri_parameter | Time_parameter | Sort_parameter | Format_parameter | String_parameter | Boolean_parameter | Include_parameter | Reference_parameter | Elements_parameter | Page_parameter

Value -> Uri_value | Time_value | Sort_value | Format_value | String_value | Boolean_value | Include_value | Reference_value | Elements_value | Page_value

Uri_parameter -> _id

Time_parameter -> _lastUpdated | start | end | docdate

Sort_parameter -> _sort

Format_parameter -> _format

Elements_parameter -> _elements

Page_parameter -> _count | _offset

String_parameter -> status | maineconomicactivity | othereconomicactivity | active | address | address-city | address-country | address-postalcode | address-state | address-use | identifier | name | type | executivename | executiveposition | email | phone | telecom | assigner | mediaid | status | level | program-status | url | version | academyear | position-code | position-name | position-unit | physicaltype | reason | docid | docname

Include_parameter -> _include | _revinclude

Include_value-> Resourse:Reference_parameter

Reference_parameter -> parent | member | predecessor | partof | link | entity | replaces | entity | license | certificate | custodian | subject

Time_value -> Date | Datetime | TimemodifierDate | TimemodifierDatetime

Format_value -> xml

Page_value -> _count | _offset

String_value -> Word

Boolean_value -> true | false

Reference_value -> Word16

Uri_value -> Word16

Word -> Symbol | WordSymbol

Symbol -> a | b | c .. z | A | B | C .. Z | / | - | ‘|’ | @ | 0 | 1 | 2 .. 9

Word16 -> Symbol16 | Word16Symbol16

Symbol16 -> 0 | 1| 2 .. 9 | a | b | c | d | e | f

Datetime -> DateTTime

Timemodifier -> ne | lt | gt | le | ge

Time -> I24:I60:I60.I1000Z

Date -> Y-M-D

Y -> Number

M -> 01 | 02 | 03 .. 12

D -> 01 | 02 | 03 .. 31

I60 -> 0 | 1 | 2 .. 59

I24 -> 0 | 1 | 2 .. 23

I1000 -> 0 | 1 | 2 .. 999

Number -> Digit | NumberDigit

Digit -> 0 | 1 | 2 .. 9

Грамматика Gm представляет собой следующую запись. Данная грамматика описывает некоторое подможество цепочек языка агрегаций MongoDB.

Gm({a, b, c .. z, A, B, C .. Z, /, \, -, _, |, 0, 1, 2 .. 9, $, “, ”, ‘{’, ‘}’, ^, ‘[’, ‘]’ , @}, {Query, Pipeline, Match_stage, Sort_stage, Skip_stage, Limit_stage, Sort_query, Sort_direction, Match_query, And_expr, And_addition, First_expression, Or_expr, Or_addition, Expression, String_expr, Address_expr, Name_expr, Exist_expr, Bool_expr, Date_expr, Date_operator, Token_expr, Tokenend_field, Backbone_array_expr, Array_expr, Reference_expr, Backbone_field, Token_field, Reference_field, Field, String_field, Bool_field, Boolean, Date_field, Word, Symbol, Word16, Symbol16, Datetime, Time, Date, Y, M, D, I60, I24, I1000, Number, Digit}, Pm, Query)

Pm:

Query -> {“aggregate” : ”_ _collection_ _”, “pipeline” : [Pipeline]}

Pipeline -> Match_stage Sort_stage Skip_stage Limit_stage

Match_stage -> {“$match” : {Match_query}}, | λ

Sort_stage -> {“$sort” : {Sort_query}}, | λ

Skip_stage -> {“$skip” : Number},

Limit_stage -> {“$limit” : Number}

Sort_query -> “Field” : Sort_direction | Sort_query, “Field” : Sort_direction

Sort_direction -> -1 | 1

Match_query -> Expression | And_expr

And_expr -> “$and” : [{First_expression}, {First_expression}And_addition]

And_addition -> λ | , {First_expression} | And_addition, {First_expression}

First_expression -> Or_expr | Expression

Or_expr -> “$or” : [{Expression}Or_addition]

Or_addition -> λ | , {Expression} | Or_addition, {Expression}

Expression -> String_expr | Address_expr | Name_expr | Exist_expr | Bool_expr | Date_expr | Token_expr | Reference_expr | Backbone_expr

String_expr -> “String_field” : “Word” | “String_field” : {“$regularExpression” : {“pattern” : “\\Word\\E”, “options” : “i”}} | “String_field” : {“$regularExpression” : {“pattern” : “^\\Word\\E”, “options” : “i”}}

Address_expr -> "$or" : [{ "address.city" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "address.district" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "address.postalCode" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "address.street" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "address.text" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}]

Name_expr -> "$or" : [{ "name.family" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "name.given" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "name.middle" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}, { "name.text" : { "$regularExpression" : { "pattern" : "^\\Word\\E", "options" : "i"}}}

Exist_expr -> “Field” : {“$exists” : Boolean}

Bool_expr -> “Bool_field” : Boolean

Date_expr -> “Date_field” : {“Date_operator” : {“$java” : “Datetime”}} | “$and” : [{Date_expr}, {Date_expr}] | “$or” : [{Date_expr}, {Date_expr}]

Date_operator -> $gte | $lt

Token_expr -> “Tokenend_field” : “Word” | “Token_field.Tokenend_field” : “Word” | “$and” : [{“system” : “Word”, “Tokenend_field” : “Word”}] | “$and” : [{“Token_field.system” : “Word”}, {“Token_field.Tokenend_field” : “Word”}] | “$and” : [{“system” : {“$exist” : false}, “Tokenend_field” : “Word”}] | “$and” : [{“Token_field.system” : {“$exist” : false}}, {“Token_field.Tokenend_field” : “Word”}] | “system” : “Word” | “Token_field.system” : “Word”

Tokenend_field -> value | code

Backbone_array_expr -> “Backbone_field” : {Array_expr} | “Backbone_field” : {“$elemMatch” : {Backbone_array_expr}}

Array_expr -> “$elemMatch” : {Token_expr} | “Reference_field” : {Array_expr} | “Token_field” : {Array_expr}

Reference_expr -> “Reference_field.ref.$id” : {“$oid” : “Word16”} | “_id” : {“$oid” : “Word16”}

Backbone_field -> educationalProgram | position | group | subject

Token_field -> identifier | maineconomicactivity | othereconomicactivity | address.country | address.state | type | email | telecom | Reference_field.identifier

Reference_field -> entity | parent | member | predecessor | partOf | link.target | license | certificate | custodian | subject.subject

Field -> meta.id | meta.lastUpdated | meta.source | meta.tag | meta.profile | identifier | entity | parent | member | predecessor | status | mainEconomicActivity | active | address | otherEconomicActivity | address.city | address.country | address.postalCode | address.state | address.use | name | alias | partOf | type | executiveName| executivePosition | telecom | link.target | identifier.period.start | identifier.period.end | identifier.assignerDisplay | mediaId | educationalEntity | insteadOfLicense | status | license | eduLocation | licensedProgram.level | insteadOfCertificate | accreditationCertificate | educationalProgram.level | educationalProgram.status | url | version | academYear | group.type | group.name | managingOrganization | physicalType | type | reason | docId | docName | docDate | subject.subject

String_field -> status | address.city | address.postalCode | address.use | name | alias | executiveName | executivePosition | identifier.assignerDisplay | address | mediaId | eduLocation | url | version | academYear | position.name | position.organizationUnit | group.name | docId | docName

Bool_field -> active

Boolean -> true | false

Date_field -> identifier.period.start | identifier.period.end | docDate

Word -> Symbol | WordSymbol

Symbol -> a | b | c .. z | A | B | C .. Z | / | - | ‘|’ | @ | 0 | 1 | 2 .. 9

Word16 -> Symbol16 | Word16Symbol16

Symbol16 -> 0 | 1| 2 .. 9 | a | b | c | d | e | f

Datetime -> DateTTime

Time -> I24:I60:I60.I1000

Date -> Y-M-D

Y -> Number

M -> 01 | 02 | 03 .. 12

D -> 01 | 02 | 03 .. 31

I60 -> 0 | 1 | 2 .. 59

I24 -> 0 | 1 | 2 .. 23

I1000 -> 0 | 1 | 2 .. 999

Number -> Digit | NumberDigit

Digit -> 0 | 1 | 2 .. 9

Запросы к mongoDB выполняются в пределах исходной коллекции, связанной с конечной точкой URL запроса.

BackboneElement – это базовый тип вспомогательных классов, в данной системе их десять: EducationalProgram (AccreditationSupplement), EducationForm (AccreditationSupplement), Group (EduGroup), Position (EduLocation), Property (EduLocation), Subject (Event), LicensedProgram (LicenseSupplement), PersonLink (Person), Position (PositionList),

OrganizationUnit (PositionList). Экземпляры подобных классов всегда объединяются в список в классе ресурса или ином вспомогательном классе.

Параметры _include и _revinclude выбиваются из общей картины поскольку подразумевают обращение к другой, неисходной коллекции. Такие запросы не могут быть выполнены в пределах одной агрегации без задействования дополнительной памяти, поэтому будут выполнены по завершении основной агрегации [7].

Если в параметрах не поступили настройки постраничного вывода, то запрос должен исполниться с этапами {“$skip” : 0}, {“$limit” : 10} – первые десять найденных документов.

По итогу, задача преобразования запроса в форме URL в запрос к драйверу MongoDB сводится к разработке транслятора, преобразующего строку, соответсвующую грамматике Gu в строку, построенную на основе грамматики Gm и первоочередной задачей встает разработка правил трансляции.

1.5 Правила трансляции

Продукционная модель знаний или модель, основанная на правилах, позволяет представить знания в виде предложений типа «Если (условие), то (действие)» [8].

Под «условием» (антецедентом) понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, под «действием» (консеквентом) – действия, выполняемые при успешном исходе поиска. Действия могут быть как промежуточными, так и терминальными или целевыми, завершающими работу системы.

Данные – это факты, хранящиеся в базе фактов, на основании которых запускается машина вывода.

Правила трансляции строятся на основе грамматик Gu и Gm, базируются в первую очередь на типах параметров спецификации FHIR: Reference, Date, String, Token – типы диктуют общую структуру входного запроса и выходного агрегирующего, допустимость модификаторов и их обработку. Затем

анализируется тип поля, с которым ассоциирован параметр: тип параметра String может относиться к типам поля String, Address, HumanName, тип параметра Token может относиться к Coding, Identifier, ContactPoint, CodeableConcept.Coding, Reference.Identifier, BackboneElement. Reference.Identifier и так далее. Для типа Token вложенность и кратность поля, ассоциированного с параметром, имеет значение – в агрегирующий запрос пишется дополнительный оператор $elemMatch. После обработки общих и специальных параметров накладываются ограничения или расширения контролирующих вывод параметров.

Следовательно, фактами в данной модели являются тип параметра, тип, вложенность и кратность ассоциированного поля, а условием – сочетание некоторых заранее оговоренных фактов.

Действием является трансляция строки с использованием грамматики Gu в строку на основе грамматики Gm – реакция на некоторое входное значение.

Факты системы собираются из разных источников: часть из них заносится в словари HashMap на этапе написания или адаптации кода библиотеки, часть является метаданными объектов системы и выясняется в процессе работы приложения.

Продукционные правила вывода указаны в таблицах 5-43.

Таблица 5 – Правило 1

Характеристика	Значение
Условие	Запрос без параметров
Описание поведения сервера (правило)	Исполнение без ограничений вывода, но с постраничным выводом (по умолчанию – вывод десяти первых элементов)
Пример параметра	-
Запись примера запроса с использованием грамматики Gu	отсутствие параметров

Продолжение таблицы 5

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{ "aggregate" : "__collection__", "pipeline" : [{ "$skip" : 0}, { "$limit" : 10}]}

Дальнейшие правила описывают только этап конвейера агрегации match, и не включают обязательные этапы skip и limit, но подразумевают их присутствие.

Таблица 6 – Правило 2

Характеристика	Значение
Условие	Параметр типа String ассоциирован с полем типа String
Описание поведения сервера (правило)	Исполнение по умолчанию: регистронезависимый поиск документов, где начало поля включает значение параметра
Пример параметра	Licence.mediaid
Запись примера запроса с использованием грамматики Gu	mediaid=736599
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "mediaId" : { "$regularExpression" : { "pattern" : "^\\Q736599\\E", "options" : "i"}}}}

Таблица 7 – Правило 3

Характеристика	Значение
Условие	Параметр типа String с модификатором :contains, параметр ассоциирован с полем типа String
Описание поведения сервера (правило)	Регистронезависимый поиск документов, где значение поля включает значение параметра
Пример параметра	Licence.mediaid
Запись примера запроса с использованием грамматики Gu	mediaid:contains= 736599

Продолжение таблицы 7

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "mediaId" : { "$regularExpression" : { "pattern" : "\\Q736599\\E", "options" : "i"}}}}

Таблица 8 – Правило 4

Характеристика	Значение
Условие	Параметр типа String с модификатором :exact, параметр ассоциирован с полем типа String
Описание поведения сервера (правило)	Поиск документов, где значение поля равно значению параметра, регистрозависимый
Пример параметра	Licence.mediaid
Запись примера запроса с использованием грамматики Gu	mediaid:exact= 736599
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "mediaId" : "736599"}}

Таблица 9 – Правило 5

Характеристика	Значение
Условие	Параметр типа String ассоциирован с полем типа Address
Описание поведения сервера (правило)	Для типа Address сервером должен быть осуществлен регистронезависимый поиск по полям Address.city, Address.district, Address.postalCode, Address.street, Address.text. Документ удовлетворяет условию, если хотя бы одно из значений приведенных полей начинается с значения параметра
Пример параметра	Person.address
Запись примера запроса с использованием грамматики Gu	address=45

Продолжение таблицы 9

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$or" : [{ "address.city" : { "$regularExpression" : { "pattern" : "^\\Q45\\E", "options" : "i"}}}, { "address.district" : { "$regularExpression" : { "pattern" : "^\\Q45\\E", "options" : "i"}}}, { "address.postalCode" : { "$regularExpression" : { "pattern" : "^\\Q45\\E", "options" : "i"}}}, { "address.street" : { "$regularExpression" : { "pattern" : "^\\Q45\\E", "options" : "i"}}}, { "address.text" : { "$regularExpression" : { "pattern" : "^\\Q45\\E", "options" : "i"}}}]}}

Таблица 10 – Правило 6

Характеристика	Значение
Условие	Параметр типа String с модификатором :contains, параметр ассоциирован с полем типа Address
Описание поведения сервера (правило)	Для типа Address сервером должен быть осуществлен регистронезависимый поиск по полям Address.city, Address.district, Address.postalCode, Address.street, Address.text. Документ удовлетворяет условию, если хотя бы одно из значений приведенных полей включает значение параметра
Пример параметра	Person.address
Запись примера запроса с использованием грамматики Gu	address=школ
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$or" : [{ "address.city" : { "$regularExpression" : { "pattern" : "\\Qшкол\\E", "options" : "i"}}}, { "address.district" : { "$regularExpression" : { "pattern" : "\\Qшкол\\E", "options" : "i"}}}, { "address.postalCode" : { "$regularExpression" : { "pattern" : "\\Qшкол\\E", "options" : "i"}}}, { "address.street" : { "$regularExpression" : { "pattern" : "\\Qшкол\\E", "options" : "i"}}}, { "address.text" : { "$regularExpression" : { "pattern" : "\\Qшкол\\E", "options" : "i"}}}]}}

Таблица 11 – Правило 7

Характеристика	Значение
Условие	Параметр типа String с модификатором :exact, параметр ассоциирован с полем типа Address
Описание поведения сервера (правило)	Для типа Address сервером должен быть осуществлен регистрозависимый поиск по полям Address.city, Address.district, Address.postalCode, Address.street, Address.text. Документ удовлетворяет условию, если хотя бы одно из значений приведенных полей равно значению параметра
Пример параметра	Person.address
Запись примера запроса с использованием грамматики Gu	address:exact=456710
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$or": [{ "address.city": "456710" }, { "address.district": "456710" }, { "address.postalCode": "456710" }, { "address.street": "456710" }, { "address.text": "456710" }]}}

Таблица 12 – Правило 8

Характеристика	Значение
Условие	Параметр типа String ассоциирован с полем типа HumanName
Описание поведения сервера (правило)	Для типа HumanName сервером должен быть осуществлен регистронезависимый поиск по полям HumanName.family, HumanName.given, HumanName.middle, HumanName.text. Документ удовлетворяет условию, если хотя бы одно из значений приведенных полей начинается с значения параметра
Пример параметра	Person.name
Запись примера запроса с использованием грамматики Gu	name=иванов
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$or" : [{ "name.family" : { "$regularExpression" : { "pattern" : "^\\Qиванов\\E", "options" : "i"}}}, { " name.given " : { "$regularExpression" : { "pattern" : "^\\Q45иванов\\E", "options" : "i"}}}, { " name.middle " : { "$regularExpression" : { "pattern" : "^\\Qиванов\\E", "options" : "i"}}}, { "name.text" : {

Продолжение таблицы 12

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	"$regularExpression" : { "pattern" : "^\\Qиванов\\E", "options" : "i"}}}]}}

Параметры, ассоциированные с полями типов Address и HumanName находятся в одной категории в плане выходного выражения, а потому правила для модификаторов не выделяются – они будут аналогичны правилам 6 и 7.

Таблица 13 – Правило 9

Характеристика	Значение
Условие	Параметр типа Date ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – инстант
Описание поведения сервера (правило)	Равенство значения параметра и поля документа
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=2024-04-10T20:55:00.468Z или _lastUpdated=eq2024-04-10T20:55:00.468Z
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$java": 2024-04-10T20:55:00.468 }}}

Таблица 14 – Правило 10

Характеристика	Значение
Условие	Параметр типа Date с префиксом ne ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – инстант
Описание поведения сервера (правило)	Неравенство значения параметра и поля документа
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=ne2024-04-10T20:55:00.468Z

Продолжение таблицы 14

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$lt": { "$java": 2024-04-10T20:55:00.468 }, "$gte": { "$java": 2024-04-10T20:55:00.469 }}}}

Таблица 15 – Правило 11

Характеристика	Значение
Условие	Параметр типа Date с префиксом le ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – инстант
Описание поведения сервера (правило)	Поле документа является моментом времени ранее или равным значению параметра
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=le2024-04-10T20:55:00.468Z
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$lt": { "$java": 2024-04-10T20:55:00.469 }}}}

Таблица 16 – Правило 12

Характеристика	Значение
Условие	Параметр типа Date с префиксом ge ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – инстант
Описание поведения сервера (правило)	Поле документа является моментом времени позднее или равным значению параметра
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=ge2024-04-10T20:55:00.468Z

Продолжение таблицы 16

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$gte": { "$java": 2024-04-10T20:55:00.468 }}}}

Таблица 17 – Правило 13

Характеристика	Значение
Условие	Параметр типа Date с префиксом lt ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – инстант
Описание поведения сервера (правило)	Поле документа является моментом времени ранее значения параметра
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=lt2024-04-10T20:55:00.468Z
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$lt": { "$java": 2024-04-10T20:55:00.468 }}}}

Таблица 18 – Правило 14

Характеристика	Значение
Условие	Параметр типа Date с префиксом gt ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – инстант
Описание поведения сервера (правило)	Поле документа является моментом времени позднее значения параметра
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=gt2024-04-10T20:55:00.468Z

Продолжение таблицы 18

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$gte": { "$java": 2024-04-10T20:55:00.469 }}}}

Таблица 19 – Правило 15

Характеристика	Значение
Условие	Параметр типа Date ассоциирован с полем типа LocalDate, LocalDateTime или ZonedDateTime, входное значение – дата
Описание поведения сервера (правило)	Параметр устанавливает период, состоящий из всех моментов времени этих суток, в который может попасть значение поля документа
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=2024-04-10 или _lastUpdated=eq2024-04-10
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$gte": { "$java": 2024-04-10T00:00 }, "$lt": { "$java": 2024-04-11T00:00 }}}}

Таблица 20 – Правило 16

Характеристика	Значение
Условие	Параметр типа Date с префиксом ne ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – дата
Описание поведения сервера (правило)	Параметр устанавливает период, состоящий из всех моментов времени этих суток, в который не может попасть значение поля документа
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=ne2024-04-10

Продолжение таблицы 20

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$gte": { "$java": 2024-04-11T00:00 }, "$lt": { "$java": 2024-04-10T00:00 }}}}

Таблица 21 – Правило 17

Характеристика	Значение
Условие	Параметр типа Date с префиксом le ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – дата
Описание поведения сервера (правило)	Поле документа является моментом времени ранее или равным последнему моменту времени в сутках передаваемой даты
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=le2024-04-10
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$lt": { "$java": 2024-04-11T00:00 }}}}

Таблица 22 – Правило 18

Характеристика	Значение
Условие	Параметр типа Date с префиксом ge ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – дата
Описание поведения сервера (правило)	Поле документа является моментом времени позднее или равным первому моменту времени в сутках передаваемой даты
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=ge2024-04-10

Продолжение таблицы 22

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$gte": { "$java": 2024-04-10T00:00 }}}}

Таблица 23 – Правило 19

Характеристика	Значение
Условие	Параметр типа Date с префиксом lt ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – дата
Описание поведения сервера (правило)	Поле документа является моментом времени ранее первого момента времени в сутках передаваемой даты
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=lt2024-04-10
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$lt": { "$java": 2024-04 -10T00:00 }}}}

Таблица 24 – Правило 20

Характеристика	Значение
Условие	Параметр типа Date с префиксом gt ассоциирован с полем типа LocalDate, LocalDateTime, ZonedDateTime, входное значение – дата
Описание поведения сервера (правило)	Поле документа является моментом времени позднее последнего момента времени в сутках передаваемой даты
Пример параметра	Organization._lastUpdated
Запись примера запроса с использованием грамматики Gu	_lastUpdated=gt2024-04-10

Продолжение таблицы 24

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{"$match": { "meta.lastUpdated": { "$gte": { "$java": 2024-04-11T00:00 }}}}

Таблица 25 – Правило 21

Характеристика	Значение
Условие	Параметр типа Reference ассоциирован с полем типа Reference
Описание поведения сервера (правило)	Ссылочное поле документа указывает на документ с системным идентификатором, равным значению параметра
Пример параметра	EducationalEntity.entity
Запись примера запроса с использованием грамматики Gu	entity=6568952d1d3c443c7202e611
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "entity.ref.$id" : { "$oid" : "6568952d1d3c443c7202e611"}}}

Таблица 26 – Правило 22

Характеристика	Значение
Условие	Параметр типа Reference ассоциирован с полем типа String
Описание поведения сервера (правило)	Системный идентификатор документа равен значению параметра
Пример параметра	EducationalEntity._id
Запись примера запроса с использованием грамматики Gu	_id=6568952d1d3c443c7202e612
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "_id" : { "$oid" : "6568952d1d3c443c7202e612"}}}

Таблица 27 – Правило 23

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа String

Продолжение таблицы 27

Характеристика	Значение
Описание поведения сервера (правило)	Значение поля документа соответствует значению параметра, регистрозависимый поиск
Пример параметра	EduGroup.academyear
Запись примера запроса с использованием грамматики Gu	academyear=2014
Запись примера запроса с использованием грамматики Gm	{"$match": { " academYear ": "2014" }}

Обработка параметра, ассоциированного с полем типа String, и обработка параметра типа String с модификатором :exact совпадают.

Таблица 28 – Правило 24

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Enum
Описание поведения сервера (правило)	Значения поля документа совпадает со значением параметра, регистрозависимый поиск. Значение параметра ограничено перечислением, допустимые значения приводятся в описании ассоциированного поля в таблице с составом элементов ресурса
Пример параметра	EducationalEntity.status
Запись примера запроса с использованием грамматики Gu	status=active
Запись примера запроса с использованием грамматики Gm	{"$match": { "status": "ACTIVE" }}

Таблица 29 – Правило 25

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Boolean

Продолжение таблицы 29

Характеристика	Значение
Описание поведения сервера (правило)	Значения поля документа совпадает со значением параметра, регистронезависимый поиск.
Пример параметра	Organization.active
Запись примера запроса с использованием грамматики Gu	active=true
Запись примера запроса с использованием грамматики Gm	{"$match": { "active": true }}

Таблица 30 – Правило 26

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Identifier или ContactPoint, запись 1
Описание поведения сервера (правило)	Значение параметра соответствует значению value независимо от значения системного свойства
Пример параметра	EducationalEntity.identifier
Запись примера запроса с использованием грамматики Gu	identifier=6bd09cef-e125-a186-1c14-258d99ec0bda
Запись примера запроса с использованием грамматики Gm	{ "$match": { "identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda" }}

Таблица 31 – Правило 27

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Identifier или ContactPoint, запись 2
Описание поведения сервера (правило)	Значение параметра до вертикальной черты соответствует значению system, значение параметра после вертикальной черты – значению value
Пример параметра	EducationalEntity.identifier

Продолжение таблицы 31

Характеристика	Значение
Запись примера запроса с использованием грамматики Gu	identifier=https://id.obrnadzor.gov.ru/school|6bd09cef-e125-a186-1c14-258d99ec0bda
Запись примера запроса с использованием грамматики Gm	{ "$match": { "$and" : [{ "identifier.system" : "https://id.obrnadzor.gov.ru/school"}, { "identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}

Таблица 32 – Правило 28

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Identifier или ContactPoint, запись 3
Описание поведения сервера (правило)	Значение параметра соответствует значению value, а системное свойство отсутствует
Пример параметра	EducationalEntity.identifier
Запись примера запроса с использованием грамматики Gu	identifier=|6bd09cef-e125-a186-1c14-258d99ec0bda
Запись примера запроса с использованием грамматики Gm	{ "$match": { "$and" : [{ "identifier.system" : {"$exist" : false}}, { "identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}

Таблица 33 – Правило 29

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Identifier или ContactPoint, запись 4
Описание поведения сервера (правило)	Значение параметра соответствует системному полю system, идентификатор value не имеет значения
Пример параметра	EducationalEntity.identifier
Запись примера запроса с использованием грамматики Gu	identifier=https://id.obrnadzor.gov.ru/school|

Продолжение таблицы 33

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{ "$match": { "identifier.system" : "https://id.obrnadzor.gov.ru/school" }}

Для параметра типа Token, ассоциированного с типом Coding, актуальны правила 26-29, однако поле value меняется на поле code.

В списке параметров системы, описанных в таблице П, указаны два параметра с условным выражением: email и phone. Параметр типа Token, ассоциированный с полем типа ContactPoint, в таком случае имеет предзаполненное системное поле значениями “EMAIL” и “PHONE” соответственно – значения взяты из условных выражений. Оба параметра этого рода относятся к ресурсу Person и должны обработаться согласно правилу 30. При этом URI не указывается в принципе – актуальна первая форма записи параметра.

Таблица 34 – Правило 30

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа ContactPoint, выполнение с предзаполненным системным полем
Описание поведения сервера (правило)	Значение параметра соответствует значению value, значение system соответствует предзаполненному значению
Пример параметра	Person.email
Запись примера запроса с использованием грамматики Gu	email=Mivanov1960@mail.ru
Запись примера запроса с использованием грамматики Gm	{"$match": { "$and" : [{ "telecom.system" : "EMAIL"}, { "telecom.value" : "Mivanov1960@mail.ru"}]}}

Правила 26-30 могут видоизменяться в зависимости от вложенности ассоциированного поля, тогда имя поля в запросе с использованием грамматики Gm будет включать имя вышестоящего объекта. Например, в

правиле 27 для параметра, ассоциированного с типом Identifier, запись запроса такова:

{ "$match": { "$and" : [{ "identifier.system" : "https://id.obrnadzor.gov.ru/school"}, { "identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}

При использовании модификатора :identifier на параметре типа Reference обработка меняется на сходную обработке типа Token, а ассоциированное поле меняется на вложенное поле identifier. При этом запись запроса включит имя поля верхнего уровня и будет такова:

{ "$match": { "$and" : [{ "entity.identifier.system" : "https://id.obrnadzor.gov.ru/school"}, { "entity.identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}

Таблица 35 – Правило 31

Характеристика	Значение
Условие	Параметр типа Reference ассоциирован с полем типа Reference, используется модификатор :identifier
Описание поведения сервера (правило)	Параметр обрабатывается как тип Token, используя вложенное поле identifier класса Reference. Валидны 4 формы записи и 4 варианта обработки, указанные в правилах 26-29, здесь приведена вторая форма записи
Пример параметра	EducationalEntity.entity
Запись примера запроса с использованием грамматики Gu	entity:identifier=https://id.obrnadzor.gov.ru/school|6bd09cef-e125-a186-1c14-258d99ec0bda
Запись примера запроса с использованием грамматики Gm	{ "$match": { "$and" : [{ "entity.identifier.system" : "https://id.obrnadzor.gov.ru/school"}, { "entity.identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}

Поскольку в обработке параметра типа Token подразумевается единовременное выполнение двух выражений для одного объекта, то запросы, касающиеся полей-массивов, должны включать оператор $elemMatch в своем

составе, при этом синтаксис оператора требует вынести верхнее поле за пределы оператора.

Для наглядности снова обратимся к правилу 27, для параметра, ассоциированного с типом Identifier, запись запроса такова:

{ "$match": { "$and" : [{ "identifier.system" : "https://id.obrnadzor.gov.ru/school"}, { "identifier.value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}

Если параметр ассоциирован с полем-списком типа Identifier, то запрос с использованием грамматики Gm будет выглядеть иначе:

{ "$match": { "identifier" : { "$elemMatch" : { "$and" : [{ "system" : "https://id.obrnadzor.gov.ru/school"}, { "value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}}}

Таблица 36 – Правило 32

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа Identifier[]
Описание поведения сервера (правило)	Параметр обрабатывается как тип Token, валидны 4 формы записи и 4 варианта обработки, указанные в правилах 26-29, здесь приведена вторая форма записи
Пример параметра	Organization.identifier
Запись примера запроса с использованием грамматики Gu	identifier=https://id.obrnadzor.gov.ru/school|6bd09cef-e125-a186-1c14-258d99ec0bda
Запись примера запроса с использованием грамматики Gm	{ "$match": { "identifier" : { "$elemMatch" : { "$and" : [{ "system" : "https://id.obrnadzor.gov.ru/school"}, { "value" : "6bd09cef-e125-a186-1c14-258d99ec0bda"}]}}}}

Правило 33 диктует обработку параметра, ассоциированного с полем-списком типа Coding, находящемся в объекте типа CodeableConcept, оно демонстрирует сочетание правил вложенности и кратности.

Таблица 37 – Правило 33

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа CodeableConcept
Описание поведения сервера (правило)	Параметр обрабатывается как тип Token, валидны 4 формы записи и 4 варианта обработки, указанные в правилах 26-29, здесь приведена вторая форма записи
Пример параметра	Event.reason
Запись примера запроса с использованием грамматики Gu	reason=https://nsi.edu.gov.ru/CodeSystem/CodeSystem-OK018-24-1|01
Запись примера запроса с использованием грамматики Gm	{ "$match": { "reason.coding" : { "$elemMatch" : { "$and" : [{ "system" : "https://nsi.edu.gov.ru/CodeSystem/CodeSystem-OK018-24-1"}, { "code" : "01"}]}}}}

Таблица 38 – Правило 34

Характеристика	Значение
Условие	Параметр типа Token ассоциирован с полем типа CodeableConcept[]
Описание поведения сервера (правило)	Параметр обрабатывается как тип Token, валидны 4 формы записи и 4 варианта обработки, указанные в правилах 26-29, здесь приведена вторая форма записи
Пример параметра	Organization.type
Запись примера запроса с использованием грамматики Gu	type=https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Edu-EduOrganizationType-1|01 02 01
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "type" : { "$elemMatch" : { "coding" : { "$elemMatch" : { "$and" : [{ "system" : "https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Edu-EduOrganizationType-1"}, { "code" : "01 02 01"}]}}}}}}

Классы под общим названием BackboneElement чаще всего собираются в списки в классах ресурсов и могут иметь внутри себя объекты всех вышеперечисленных классов: Identifier, Identifier[], Coding, Coding[], CodeableConcept, CodeableConcept[], иной BackboneElement[] и другие.

Таблица 39 – Правило 35

Характеристика	Значение
Условие	На любом общем или специальном параметре может быть использован модификатор :missing
Описание поведения сервера (правило)	Документ удовлетворяет условию missing=true, если поле, ассоциированное с поступившим параметром, отсутствует в документе, и наоборот соответственно
Пример параметра	Organization.type
Запись примера запроса с использованием грамматики Gu	type:missing=true
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "type" : { "$exists" : false}}}

Общие и специальные параметры могут иметь несколько значений после знака равно, записанных через запятую, это расценивается как выражение логического ИЛИ. Необходимая степень соответствия определяется правилами 2-35: для параметра типа Token и параметра типа String, ассоциированных с типом String, условия истинности выражения будут различны. Параметры типа Token могут независимо друг от друга иметь различные формы записи, параметры типа Date могут иметь различные префиксы также независимо друг от друга.

Таблица 40 – Правило 36

Характеристика	Значение
Условие	Логическое ИЛИ нескольких значений одного параметра
Описание поведения сервера (правило)	Документ удовлетворяет условию, если значение поля соответствует хотя бы одному из приведенных значений параметра
Пример параметра	EduGroup.academyear
Запись примера запроса с использованием грамматики Gu	academyear=2014,2015

Продолжение таблицы 40

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$or" : [{ "academYear" : "2014"}, { "academYear" : "2015"}]}}

Запрос с использованием грамматики Gu может включать несколько общих и/или специальных параметров, записанных через амперсанд, в таком случае соответствующие выражения с использованием грамматики Gm записываются согласно правилам 2-36 и объединяются в логическое И.

Таблица 41 – Правило 37

Характеристика	Значение
Условие	Логическое И нескольких параметров
Описание поведения сервера (правило)	Документ удовлетворяет условию, если документ соответствует всем приведенным параметрам
Пример параметра	EduGroup.academyear, EduGroup.version
Запись примера запроса с использованием грамматики Gu	academyear=2014,2015&version=v1
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$and" : [{ "$or" : [{ "academYear" : "2014"}, { "academYear" : "2015"}]}, { "version" : { "$regularExpression" : { "pattern" : "^\\Qv1\\E", "options" : "i"}}}]}}

После формирования списка документов к выводу применяются параметры, включающие в вывод дополнительные документы, связанные с найденными ссылочным механизмом.

Таблица 42 – Правило 38

Характеристика	Значение
Условие	Параметр определяет включение в вывод документов, на которые ссылается найденный документ исходной коллекции. Значением параметра является имя исходного ресурса и через двоеточие – ссылочное поле типа Reference исходного ресурса

Продолжение таблицы 42

Характеристика	Значение
Описание поведения сервера (правило)	Ссылочное поле, переданное в значении параметра, хранит системный идентификатор документа исходной или иной коллекции, документ с этим идентификатором должен быть включен в вывод. Запрос выполняется вне основной агрегации над коллекцией ресурса, указанного в ссылке DBHref
Пример параметра	_include
Запись примера запроса с использованием грамматики Gu	_include=EducationalEntity:entity
Запись примера запроса с использованием грамматики Gm	{ "_id" : { "$in" : [{"$oid" : "6568952d1d3c443c7202e600"}]}}

Таблица 43 – Правило 39

Характеристика	Значение
Условие	Параметр определяет включение в вывод документов, которые ссылаются на найденный документ исходной коллекции. Значением параметра является имя ресурса и через двоеточие – ссылочное поле типа Reference этого ресурса, которое хранит идентификатор документа исходной коллекции
Описание поведения сервера (правило)	Ссылочное поле, переданное в значении параметра, хранит системный идентификатор документа исходной коллекции, документ иной коллекции с таким значением ссылочного поля должен быть включен в вывод. Запрос выполняется вне основной агрегации, над коллекцией ресурса, указанного в значении параметра
Пример параметра	_revinclude
Запись примера запроса с использованием грамматики Gu	_revinclude=License:entity

Продолжение таблицы 43

Характеристика	Значение
Запись примера запроса с использованием грамматики Gm	{ "educationalEntity.ref.$id" : { "$in" : [{"$oid" : "6568952d1d3c443c7202e603"}]}}

2 Разработка машины вывода

Механизм поиска, приведенный в данной работе, реализуется как библиотека на языке Java, которая в дальнейшем будет встроена в общий проект системы учета данных об общеобразовательных учреждениях. Библиотеку будут использовать несколько микросервисов системы.

На этапе выявления правил перевода было решено, что в первую очередь из строки запроса необходимо извлечь поступившие параметры, модификаторы и префиксы. Далее выделить ассоциированные поля, выяснить их тип, вложенность и для параметров типа Token кратность ассоциированного поля. Затем параметры стоит разбить на категории обработки, похожие на типы FHIR, но уже по ассоциированному типу:

– String или Enum,

– Decimal или иной числовой,

– LocalDateTime, LocalDate, ZonedDateTime,

– Boolean,

– Reference,

– Identifier, Coding, ContactPoint, CodeableConcept, BackboneElement.

В этих категориях реализовывать обработку согласно правилам.

После обработки общих и специальных параметров накладываются ограничения или расширения контролирующих вывод параметров, при этом параметры _count, _sort, _offset реализуются в конвейере агрегации к исходной коллекции, параметры _include и _revinclude требуют обращения к иным коллекциям, параметр _elements реализуется программно, параметр _format реализуется системой вне библиотеки поиска.

Общий алгоритм работы представлен на рисунках 2-4.

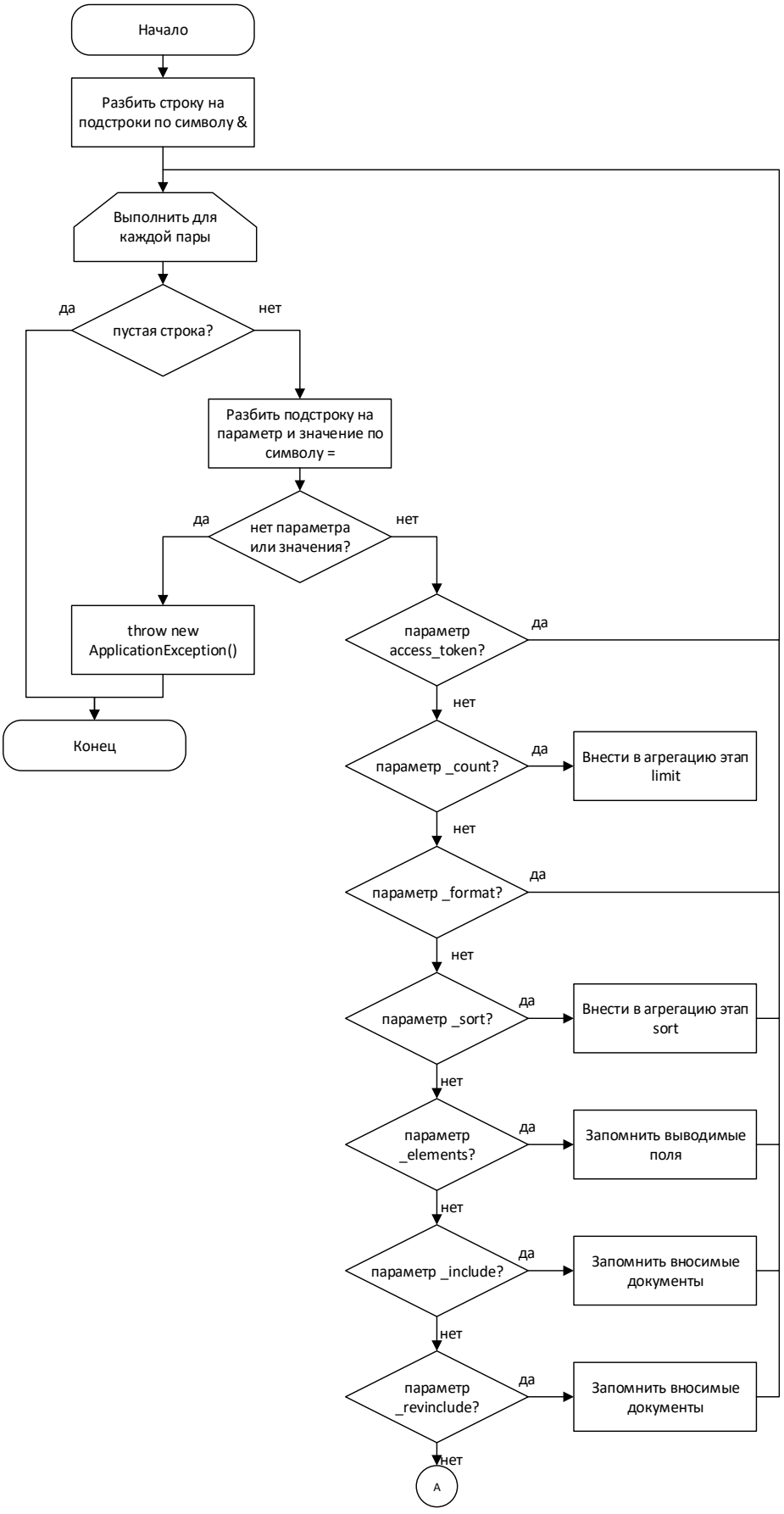

Рисунок 2 – Общая схема алгоритма

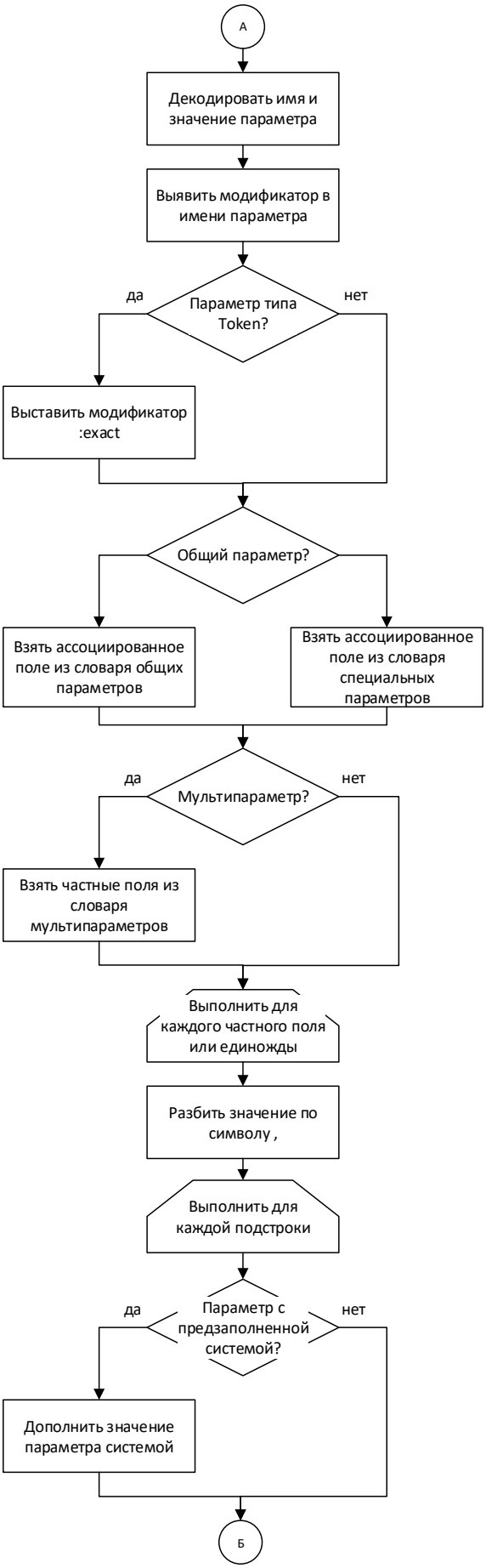

Рисунок 3 – Общая схема алгоритма, продолжение

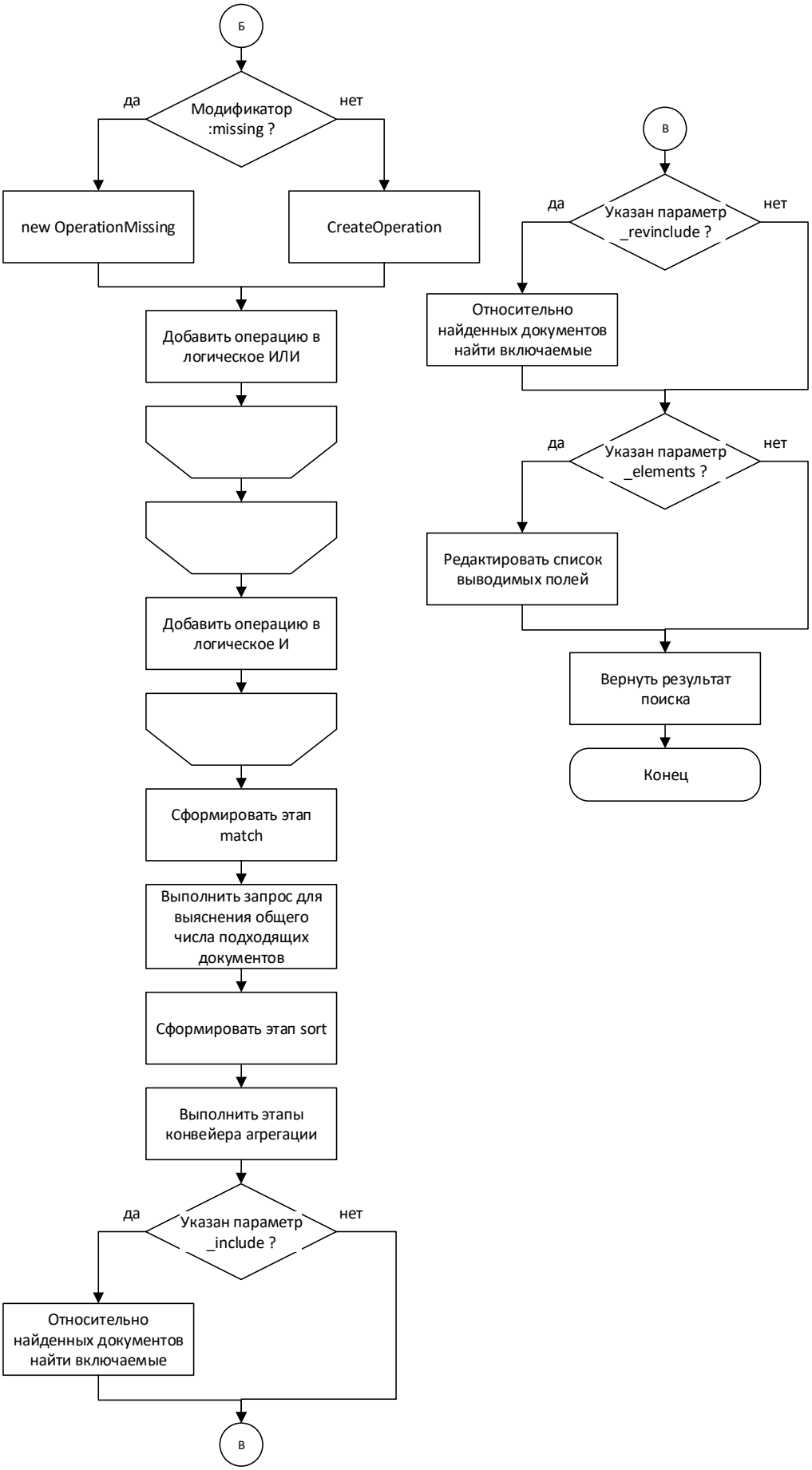

Рисунок 4 – Общая схема алгоритма, продолжение

База знаний включает справочники параметров и ассоциированных с нииим полей:

– params – все специальные параметры системы и ассоциированные с ними поля,

– generalParams – общие для всех ресурсов параметры,

– tokenParams – параметры типа Token, ассоциированные со строковыми полями,

– compositeParams – параметры типа Token с предзаполненым системным полем,

– multiParams – параметры, которые подразумевают проверку нескольких полей, например, ассоциированные с полем типа HumanName.

Метаданные о типе ассоциированного поля, вложенности и кратности выясняются выясняются в процессе работы приложения с использованием механизма Java Reflection API.

По вышеупомянутым справочникам ведется проверка типа параметра и выявление ассоциированных с ним полей, следующие этапы проверки условия правил проводятся в методе CreateOperation() приведенном ниже. Обработка параметров типа Token подразумевает вложенность поля, которая разрешается рекурсией метода createOperationToken(), он также приведен ниже.

```
private Operation createOperation(Class<?> clazz, Modifier modifier,
String param, String valueWithPrefix) {
  Prefix prefix = null;
  Operation operation = null;
  Class<?> paramType = getParamType(clazz, param);
  try {
    if (paramType.equals(BigDecimal.class) ||
Number.class.isAssignableFrom(paramType)) {
      BigDecimal value;
      if (valueWithPrefix.length() > 2) {
        prefix = Prefix.getByValue(valueWithPrefix.substring(0, 2));
      }
      if (prefix != null) {
        value = new BigDecimal(valueWithPrefix.substring(2));
      } else {
        value = new BigDecimal(valueWithPrefix);
      }
        operation = new OperationCompareDecimal(param, prefix, value);
      } else if (paramType.equals(String.class)||paramType.isEnum()) {
        if (paramType.isEnum()) {
```

```
          Method getValue = null;
          try {
            getValue = paramType.getDeclaredMethod("getValue");
          } catch (Exception ignore) {
          }
          String findValue = valueWithPrefix;
          if (getValue != null) {
            for (Object enumConstant : paramType.getEnumConstants()) {
              String value = (String) getValue.invoke(enumConstant);
              if (value.equals(findValue)) {
                findValue = enumConstant.toString();
                break;
              }
            }
          }
          operation = new OperationCompareString(param, EXACT,
findValue);
        } else {
          if ("id".equals(param))
            operation = new OperationCompareId(valueWithPrefix);
          else
            operation = new OperationCompareString(param, modifier,
valueWithPrefix);
        }
      } else if (paramType.equals(Boolean.class)) {
        Boolean value = null;
        if (valueWithPrefix.equalsIgnoreCase("true"))
          value = true;
        else if (valueWithPrefix.equalsIgnoreCase("false"))
          value = false;
        if (value != null)
          operation = new OperationCompareBoolean(param, value);
        else
          operation = new OperationCompareString(param, EXACT,
valueWithPrefix);
      } else if (paramType.equals(LocalDate.class) ||
paramType.equals(LocalDateTime.class) ||
paramType.equals(ZonedDateTime.class) ||
paramType.equals(Period.class)) {
        String value = valueWithPrefix;
        prefix = Prefix.getByValue(valueWithPrefix.substring(0, 2));
        if (prefix != null) {
          value = valueWithPrefix.substring(2);
        }
        if (paramType.equals(Period.class))
          operation = new OperationComparePeriod(param, prefix,
value);
        else
          operation = new OperationCompareDateTime(param, prefix,
value);
      } else if (paramType.equals(Reference.class) && modifier !=
Modifier.IDENTIFIER) {
        operation = new OperationCompareReference(param,
valueWithPrefix);
      } else {
        UUID uuid = UUID.randomUUID();
```

```
        String[] systemAndCode = (uuid + valueWithPrefix +
uuid).split("\\|");
        String code, system;
        if (systemAndCode.length == 1) {
          code = systemAndCode[0].replaceAll(uuid.toString(), "");
          system = null;
        } else {
          system = systemAndCode[0].replaceAll(uuid.toString(), "");
          code = systemAndCode[1].replaceAll(uuid.toString(), "");
        }
        operation = this.createOperationToken(clazz, "", param,
system, code);
      }
      if (operation == null) {
        throw new ApplicationException("The structure is specified
instead of a field");
      }
    } catch (Exception e) {
      throw new ApplicationException(messageUtil.getMessage(
"search.ValueNotParsed", valueWithPrefix, param), List.of("url"));
    }
    return operation;
}

private Operation createOperationToken(Class<?> clazz, String prefix,
String param, String system, String code) {
  String head, tail = "";
  String[] params = param.split(("\\."));
  if (params.length > 1){
    tail = param.replace(params[0] + ".", "");
  }
  head = params[0];
  Class<?> paramType = getParamType(clazz, head);
  Boolean isList = isList(clazz, head);
  if (paramType.equals(Coding.class)) {
    return new OperationCompareToken(prefix + head, isList, "system",
"code", system, code);
  } else if (paramType.equals(Identifier.class) ||
paramType.equals(ContactPoint.class)) {
    return new OperationCompareToken(prefix + head, isList, "system",
"value", system, code);
  } else {
    if (paramType.equals(Reference.class)) {
      tail = "identifier";
    } else if (paramType.equals(CodeableConcept.class)) {
      tail = "coding";
    }
    Operation operation;
    if (isList) {
      operation = this.createOperationToken(paramType, "", tail,
system, code);
      operation = new OperationList(prefix + head, operation);
    } else {
      operation = this.createOperationToken(paramType, prefix + head +
".", tail, system, code);
    }
    return operation;
```

```
  }
 }
}
```

Методы OperationCompareString(), OperationCompareBoolean(), OperationComparePeriod(), OperationCompareToken(), OperationCompareId(), OperationCompareDecimal(), OperationCompareReference(), OperationCompareDateTime(), отвечают за создание выражения на языке драйвера базы данных MongoDB.

Специальные и общие параметры определяют этап match в конвейере агрегации, после его выполнения исполняются еще два поисковых запроса, собранных методами getIncludeDocuments() и getRevincludeDocuments(), методы приведены ниже.

```
private List<Resource> getIncludeDocuments(Class<?> clazz,
List<String> includeList, List<Resource> documents) {
  List<Resource> includeDocuments = new ArrayList<>();
  includeList.forEach(p -> {
    String[] typeAndParam = p.split(":");
    if (typeAndParam.length < 2 || typeAndParam[0].equals(""))
      throw new
ApplicationException(messageUtil.getMessage("search.TypeMissed",
"_include"), List.of("url"));
    String type = typeAndParam[0];
    String param = typeAndParam[1];
    if (!clazz.getSimpleName().equals(type))
      throw new
ApplicationException(messageUtil.getMessage("search.IncorrectType",
type, "_include"), List.of("url"));
    param = customParamService.getCustomParam(clazz.getSimpleName(),
param);
    Class<?> paramClazz = getParamType(clazz, param);
    if (paramClazz.equals(Reference.class)) {
      Map<String, Set<String>> ids = new HashMap<>();
      for (Resource document : documents) {
        Object o = getValueByField(document, param);
        if (o instanceof List) {
          ((List<Reference<?>>) o).forEach(ref -> {
            if (ref != null) {
              String name = ref.getType();
              if (!ids.containsKey(name))
              ids.put(name, new HashSet<>());
              ids.get(name).add(ref.getRef().getId().toString());
            }
          });
        } else {
          Reference<?> ref = (Reference<?>) o;
          if (ref != null) {
            String name = ref.getType();
            if (!ids.containsKey(name))
              ids.put(name, new HashSet<>());
```

```
            ids.get(name).add(ref.getRef().getId().toString());
          }
        }
      }
      ids.forEach((name, idSet) -> {
        if (idSet.size() > 0) {
          Class<?> refType =
reflectionUtil.getClassBySimpleName(name);
          includeDocuments.addAll((List<Resource>)
mongoTemplate.find(Query.query(new Criteria("_id").in(idSet)),
refType));
        }
      });
    }
  });
  return includeDocuments;
}

private List<Resource> getRevincludeDocuments(List<String>
revincludeList, List<Resource> documents) {
  List<Resource> revincludeDocuments = new ArrayList<>();
  revincludeList.forEach(p -> {
    String[] typeAndParam = p.split(":");
    if (typeAndParam.length < 2 || typeAndParam[0].equals(""))
      throw new ApplicationException(messageUtil.getMessage(
"search.TypeMissed", "_revinclude"), List.of("url"));
    String type = typeAndParam[0];
    String param = typeAndParam[1];
    Class<?> clazz = reflectionUtil.getClassBySimpleName(type);
    param = customParamService.getCustomParam(clazz.getSimpleName(),
param);
    Class<?> paramClazz = getParamType(clazz, param);
    if (paramClazz.equals(Reference.class)) {
      Set<ObjectId> ids = new HashSet<>();
      documents.forEach(document -> ids.add(new
ObjectId(document.getId())));
      if (ids.size() > 0) {
        revincludeDocuments.addAll((List<Resource>)
mongoTemplate.find(Query.query(new Criteria(param +
".ref.$id").in(ids)), clazz));
      }
    }
  });
  return revincludeDocuments;
}
```

Часть кода, отвечающая за исполнение ограничений параметра _elements:

```
if (elements != null) {
  Set<String> elementSet = new HashSet<>();
  Collections.addAll(elementSet, elements.split(","));
  elementSet.add("id");
  Set<Field> fields = new HashSet<>(getAllFields(new LinkedList<>(),
clazz));
  Set<Field> nullFields = new HashSet<>();
  for (Field field : fields) {
```

```
    if (!elementSet.contains(field.getName()))
    field.setAccessible(true);
    nullFields.add(field);
  }
  for (Resource document : documents) {
    for (Field nullField : nullFields) {
      try {
        nullField.set(document, null);
      } catch (IllegalAccessException e) {
      }
    }
  }
}
```

Полный листинг библиотеки приведен в приложении А. Диаграмма классов представлена на рисунке 5.

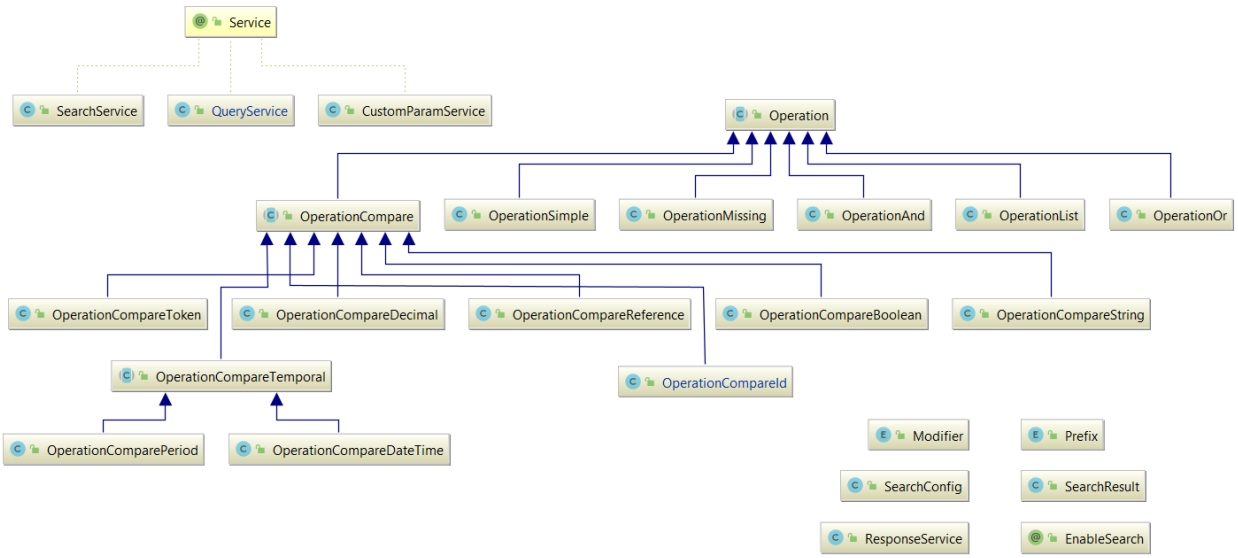

Рисунок 5 – Диаграмма классов библиотеки

3 Тестирование

3.1 Позитивное функциональное тестирование

Для тестирования используется REST-клиент Insomnia 2023.5.8.

Функциональное тестирование модуля библиотеки проводится на основе правил трансляции: ответ сервера должен соответствовать результату исполнения изложенного в правилах верного поведения сервера.

Библиотека поиска подключается к системе учета даннных об общеобразовательных учреждениях, базовым URL будет http://localhost:8005/org.

Перед тестированием небходимо получить токен авторизации, выполнив запрос:

[POST] http://localhost:8001/sso/oauth/token?client-id=org-app &scope=server&grant_type=client_credentials

--autorization Basic

--pass secret-org-app

--username org-app

Статус ответа должен быть 200 OK, тело ответа приведено ниже:

{ "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImVkdS1rZXktaWQifQ.eyJzY29wZSI6WyJzZXJ2ZXIiXSwiZXhwIjoxNzE3Mzk4MzkyLCJhdXRob3JpdGllcyI6WyJST0xFX1NFUlZJQ0UiXSwianRpIjoiMTA1MGQxNGUtOWFkMi00ZGMxLWIyZjYtNzgwNTUxZDFlM2FiIiwiY2xpZW50X2lkIjoib3JnLWFwcCJ9.X7uAA8zCPzDRXGBCVJV6_-M0_juIQn2e177fxlQIllf5w_Mrze3ZhroWraUb_cdsUqoS1yjd8GKj3-RTNtRpTLYlMRwrkOx_-BWAfFyhnOT-yxuv5G6Z53AAd2CkRS5nKnp937p1L2towAAIo5mOJ62cVs5bTorVzUeu3ovExGhhHlEHvQZ7_DmrvTg4mtzajt79jFLExeynEeIbJjjxxlbHyoLkr6u4OBV7moWd4xamgZ5tF9DQJR5fGKYut4u25u7yEsotfkoxoPcSCx8jsfgD2z2yVumI4G5uIE8oX6Rx0nXhHsKu5DIHhqStStCLKKfFCmj052zYKZhx7fa2Mg",
"token_type": "bearer",

"expires_in": 43199,
"scope": "server",
"jti": "1050d14e-9ad2-4dc1-b2f6-780551d1e3ab" }

Значение из поля access_token передется с запросами системы, авторизация Bearer, это не отражено в тестовых примерах, но подразумевается. В общем виде поисковый запрос без параметров будет выглядеть так:

[GET] http://localhost:8005/org/Resource

--autorization Bearer

--token access_token

Продукционные правила обозначают действия при условии получения некоего запроса. При выполнении запроса без параметров http://localhost:8005/org/EducationalEntity статус ответа будет 200 OK, тело ответа будет выглядеть следующим образом:

{
"resourceType": "Bundle",
"id": "8c14c87d-50ce-4c5d-824e-7fb09d7434f7",
"meta": {
"lastUpdated": "2024-06-03T09:12:58.785931+03:00"
},
"type": "searchset",
"total": 51,
"link": [
{
"relation": "self",
"url":
"http://localhost:8002/org/EducationalEntity?_offset=0&_count=10"
},
{
"relation": "next",

```
      "url":
"http://localhost:8002/org/EducationalEntity?_offset=10&_count=10"
     }
    ],
    "entry": [
     {
      "fullUrl":
"http://localhost:8002/org/EducationalEntity/6568952d1d3c443c7202e603",
      "resource": {
       "resourceType": "EducationalEntity",
       "id": "6568952d1d3c443c7202e603",
       "meta": {
        "versionId": "9",
        "lastUpdated": "2023-12-28T15:51:25.205+03:00"
       },
       "identifier": {
        "type": {
         "display": "Федеральный ИД"
        },
        "system": "https://id.obrnadzor.gov.ru/school",
        "value": "6404299c-b86d-4794-a99d-af18c9f775f1",
        "assignerDisplay": "Рособрнадзор"
       },
       "entity": {
        "type": "Organization",
        "identifier": {
         "system": "https://id.obrnadzor.gov.ru/school",
         "value": "6404299c-b86d-4794-a99d-af18c9f775f1"
        },
        "reference": "6568952d1d3c443c7202e600"
```

```
            },
            "type": {
              "system": "https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Edu-
EduOrganizationType-1",
              "code": "01 02 02",
              "display": "Образовательная организация высшего образования"
            },
            "member": [
              {
                "type": "Organization",
                "display": "ПИЛЬ МУНИЦИПАЛЬНЫЙ ОКРУГ"
              }
            ],
            "status": "entered-in-error"
          },
          "search": {
            "mode": "match"
          }
        },
      …
        {
      "fullUrl":
"http://localhost:8002/org/EducationalEntity/6568952d1d3c443c7202e6b7",
          "resource": {
            "resourceType": "EducationalEntity",
            "id": "6568952d1d3c443c7202e6b7",
            "meta": {
              "versionId": "0",
              "lastUpdated": "2023-11-30T16:59:09.174+03:00"
            },
```

```
      "identifier": {
        "type": {
          "display": "Федеральный ИД"
        },
        "system": "https://id.obrnadzor.gov.ru/school",
        "value": "6404299c-b86d-4794-a99d-af18c9f7757b",
        "assignerDisplay": "Рособрнадзор"
      },
      "entity": {
        "type": "Organization",
        "identifier": {
          "system": "https://id.obrnadzor.gov.ru/school",
          "value": "6404299c-b86d-4794-a99d-af18c9f7757b"
        },
        "reference": "6568952d1d3c443c7202e611"
      },
      "type": {
        "system": "https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Edu-
EduOrganizationType-1",
        "code": "01 02 01",
        "display": "Профессиональная образовательная организация"
      },
      "member": [
        {
          "type": "Organization"
        }
      ],
      "status": "active"
    },
    "search": {
```

```
        "mode": "match"
      }
    }
  ]
}
```


В выводе представлены первые десять документов, соответствующие критериям поиска без ограничений. Тело ответа по умолчанию в формате json.

Тело представлено ресурсом типа Bundle с type=searchset, при этом элемент link хранит сведения о постраничном выводе, entry – найденные документы, total – количество документов в базе, удовлетворяющих условиям поиска. Формат ответа совпадает с предъявляемыми требованиями.

В таблице 44 приведены тестовые примеры для проверки работы продукционных правил и результаты их выполнения. В тестовых примерах вывод ограничивается параметром _count до одного-трех документов в ответе.

Таблица 44 – Результаты выполнения позитивных тестов

Характеристика	Значение
Тестовый пример 1 – Правило 2	
URL запроса	http://localhost:8005/org/License?_count=1&mediaid=736599
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле mediaId равно “73659987” и значение поля начинается со значения параметра
Тестовый пример 2 – Правило 3	
URL запроса	http://localhost:8005/org/License?_count=1&mediaid:contains=36599
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле mediaId равно “73659987” и включает значение параметра
Тестовый пример 3 – Правило 4	
URL запроса	http://localhost:8005/org/License?_count=1&mediaid:exact=73659987
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле mediaId равно “73659987”, то есть равно значению параметра

Продолжение таблицы 44

Характеристика	Значение
Тестовый пример 4 – Правило 5	
URL запроса	http://localhost:8005/org/Person?_count=1&address=45
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где address.postalCode= “456710”. Документ удовлетворяет условию, так как хотя бы одно из полей address.city, address.district, address.postalCode, address.street, address.text начинается со значения параметра
Тестовый пример 5 – Правило 6	
URL запроса	http://localhost:8005/org/Person?_count=1&address:contains=школ
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где address.text= “456710, РОССИЯ, Челябинская область, Кунашакский муниципальный район, д. Бурино, ул. Школьная 6”. Документ удовлетворяет условию, так как хотя бы одно из полей address.city, address.district, address.postalCode, address.street, address.text включает значение параметра
Тестовый пример 6 – Правило 7	
URL запроса	http://localhost:8005/org/Person?_count=1&address:exact=456710
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где address.postalCode равно “456710”. Документ удовлетворяет условию, так как хотя бы одно из полей Address.city, Address.district, Address.postalCode, Address.street, Address.text равно значению параметра
Тестовый пример 7 – Правило 8	
URL запроса	http://localhost:8005/org/Person?_count=1&name=иванов
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где name.family равно “Иванов”, Документ удовлетворяет условию так как хотя бы одно из полей HumanName.family, HumanName.given, HumanName.middle, HumanName.text начинается с значения параметра

Продолжение таблицы 44

Характеристика	Значение
Тестовый пример 8 – Правило 9	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=2024-04-10T20:55:00.468
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2024-04-10T20:55:00.468+03:00”. Документ удовлетворяет условию, значение параметра равно значению поля
Тестовый пример 9 – Правило 10	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=ne2024-04-10T20:55:00.468Z
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2023-12-28T15:51:25.205+03:00”. Документ удовлетворяет условию, значение параметра не равно значению поля
Тестовый пример 10 – Правило 11	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=le2024-04-10T20:55:00.468
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2023-12-28T15:51:25.205+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени ранее или равным значению параметра
Тестовый пример 11 – Правило 12	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=ge2024-04-10T20:55:00.468
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2024-04-10T20:55:00.468+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени позднее или равным значению параметра
Тестовый пример 12 – Правило 13	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=lt2024-04-10T20:55:00.468
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2023-12-28T15:51:25.205+03:00”. Документ

Продолжение таблицы 44

Характеристика	Значение
	удовлетворяет условию так как поле документа является моментом времени ранее значения параметра
Тестовый пример 13 – Правило 14	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=gt2024-04-10T20:55:00.468
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2024-04-10T23:55:00.532+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени позднее значения параметра
Тестовый пример 14 – Правило 15	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=2024-04-10
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2024-04-10T20:55:00.468+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени в сутках значения параметра
Тестовый пример 15 – Правило 16	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=ne2024-04-10
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2023-12-28T15:51:25.205+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени не включенным в сутки параметра
Тестовый пример 16 – Правило 17	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=le2024-04-10
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2023-12-28T15:51:25.205+03:00”. Документ удовлетворяет условию так как поле документа является моментом моментом времени ранее или равным последнему моменту времени в сутках передаваемой даты

Продолжение таблицы 44

Характеристика	Значение
Тестовый пример 17 – Правило 18	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=ge2024-04-10
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2024-04-10T20:55:00.468+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени позднее или равным первому моменту времени в сутках передаваемой даты
Тестовый пример 18 – Правило 19	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=lt2024-04-10
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2023-12-28T15:51:25.205+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени ранее первого момента времени в сутках передаваемой даты
Тестовый пример 19 – Правило 20	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_lastUpdated=gt2024-04-10
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где meta.lastUpdated равно “2024-05-17T17:36:51.476+03:00”. Документ удовлетворяет условию так как поле документа является моментом времени позднее последнего момента времени в сутках передаваемой даты
Тестовый пример 20 – Правило 21	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&entity=65fad633c9e3f8764bdbc4b6
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где entity.reference равно “65fad633c9e3f8764bdbc4b6”. Документ удовлетворяет условию так как ссылочное поле entity указывает на документ с системным идентификатором, равным значению параметра

Продолжение таблицы 44

Характеристика	Значение
Тестовый пример 21 – Правило 22	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_id=65fad633c9e3f8764bdbc4ca
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где id равно “65fad633c9e3f8764bdbc4ca”. Документ удовлетворяет условию так как системный идентификатор документа совпадает со значеним параметра
Тестовый пример 22 – Правило 23	
URL запроса	http://localhost:8005/org/EduGroup?_count=1&academyear=2014
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле academYear равно “2014”. Документ удовлетворяет условию так как поле документа academYear соответствует значению параметра
Тестовый пример 23 – Правило 24	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&status=active
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле status равно “active”. Документ удовлетворяет условию так как поле документа совпадает со значением параметра
Тестовый пример 24 – Правило 25	
URL запроса	http://localhost:8005/org/Organization?_count=1&active=True
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле active равно “true”. Документ удовлетворяет условию так как поле документа совпадает со значением параметра, регистронезависимый поиск
Тестовый пример 25 – Правило 26	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&identifier=03b68eb8-169f-9a50-ba4e-e9b02a7897f0
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле identifier.value равно “03b68eb8-169f-9a50-ba4e-e9b02a7897f0”. Документ удовлетворяет условию так как поле документа identifier.value соответствует значению параметра независимо от значения системного свойства

Продолжение таблицы 44

Характеристика	Значение
Тестовый пример 26 – Правило 27	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&identifier=https://id.obrnadzor.gov.ru/school|03b68eb8-169f-9a50-ba4e-e9b02a7897f0
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле identifier.value равно “03b68eb8-169f-9a50-ba4e-e9b02a7897f0”, identitifer.system равно “https://id.obrnadzor.gov.ru/school”. Документ удовлетворяет условию так как поле документа identifier.system соответствует значению параметра до вертикальной черты, identifier.value соответствует значению параметра после вертикальной черты
Тестовый пример 27 – Правило 28	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&identifier=|6404299c-b86d-4794-a99d-af18c9f7757b
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле identifier.value равно “03b68eb8-169f-9a50-ba4e-e9b02a7897f0”, identitifer.system отсутствует. Документ удовлетворяет условию так как identifier.value соответствует значению параметра, а системное свойство отсутствует
Тестовый пример 28 – Правило 29	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&identifier=https://id.obrnadzor.gov.ru/school|
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле identitifer.system равно “ https://id.obrnadzor.gov.ru/school”. Документ удовлетворяет условию так как поле identifier.system соответствует значению параметра, идентификатор value не имеет значения
Тестовый пример 29 – Правило 30	
URL запроса	http://localhost:8005/org/Person?_count=1&email=Mivanov1960@mail.ru
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле telecom.system равно email, telecom.value равно “Mivanov1960@mail.ru”. Документ удовлетворяет условию так как

Продолжение таблицы 44

Характеристика	Значение
	поле документа value соответствует значению параметра, значение system соответствует предзаполненному значению
Тестовый пример 30 – Правило 31	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&entity:identifier=https://id.obrnadzor.gov.ru/school|2aaefbe0-0ef4-b9d4-62dc-642b244807a7
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где поле entity.identifier.system равно “https://id.obrnadzor.gov.ru/school”, entity.identifier.value равно “2aaefbe0-0ef4-b9d4-62dc-642b244807a7”. Документ удовлетворяет условию так как поле документа entity.identifier.system соответсвует значению параметра до вертикальной черты, entity.identifier.value соответсвует значению параметра после вертикальной черты: параметр обрабатывается как тип Token, используя вложенное поле identifier класса Reference. Валидны 4 формы записи и 4 варианта обработки, здесь приведена вторая форма записи
Тестовый пример 31 – Правило 32	
URL запроса	http://localhost:8005/org/Organization?_count=1&identifier=https://id.edu.gov.ru/Organization/ogrn|9023405162648
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где в поле identifier есть элемент с system равным “ https://id.edu.gov.ru/Organization/ogrn”, и value равным “9023405162648”. Документ удовлетворяет условию так как поле документа identifier.system соответсвует значению параметра до вертикальной черты, identifier.value соответсвует значению параметра после вертикальной черты. Валидны 4 формы записи и 4 варианта обработки, здесь приведена вторая форма записи
Тестовый пример 32 – Правило 33	
URL запроса	http://localhost:8005/org/Event?_count=1&reason=https://nsi.edu.gov.ru/CodeSystem/CodeSystem-OK018-24-1|01

Продолжение таблицы 44

Характеристика	Значение
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где в поле reason.system равно “https://nsi.edu.gov.ru/CodeSystem/CodeSystem-OK018-24-1”, а reason.code равно “01”. Документ удовлетворяет условию так как поле документа reason.system соответсвует значению параметра до вертикальной черты, reason.code соответсвует значению параметра после вертикальной черты. Валидны 4 формы записи и 4 варианта обработки, здесь приведена вторая форма записи
Тестовый пример 33 – Правило 34	
URL запроса	http://localhost:8005/org/Organization?_count=1&type=https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Edu-EduOrganizationType-1|01 02 01
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где в поле type.coding.system равно “https://nsi.edu.gov.ru/CodeSystem/CodeSystem-Edu-EduOrganizationType-1”, а type.coding.code равно “01 02 01”. Документ удовлетворяет условию так как поле документа system соответсвует значению параметра до вертикальной черты, code соответсвует значению параметра после вертикальной черты. Валидны 4 формы записи и 4 варианта обработки, здесь приведена вторая форма записи
Тестовый пример 34 – Правило 35	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&type:missing=true
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен документ, где отсутсвует поле type. Документ удовлетворяет условию так как поле type отсутсвует в документе
Тестовый пример 35 – Правило 36	
URL запроса	http://localhost:8005/org/EduGroup?_count=3&academyear=2014,2021
Результат тестового примера	Статус 200 OK. В ответе на запрос приведены 3 документа, где поле academYear равно 2014 или 2021. Три выведенных документа удовлетворяют условию так как поле academYear документов соответствует хотя бы одному из значений параметра

Продолжение таблицы 44

Характеристика	Значение
Тестовый пример 36 – Правило 37	
URL запроса	http://localhost:8005/org/EduGroup?_count=3&academyear=2014,2021&version=v1
Результат тестового примера	Статус 200 OK. В ответе на запрос приведены 3 документа, где поле academYear равно 2014 или 2021 и version равно v1. Три выведенных документа удовлетворяют условию так как поле academYear документов соответствует хотя бы одному из значений параметра и значение поля version равно значению параметра
Тестовый пример 37 – Правило 38	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_include=EducationalEntity:entity
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен один документ EducationalEntity, где “entity.reference” равен “6568952d1d3c443c7202e600”, и один документ Organization, где системный идентификатор “id” равен “6568952d1d3c443c7202e600”. Документы удовлетворяют условию так как в вывод была добавлена сущность Organization, чей системный идентификатор соответствует идентификатору в ссылочном поле entity
Тестовый пример 38 – Правило 39	
URL запроса	http://localhost:8005/org/EducationalEntity?_count=1&_revinclude=License:entity
Результат тестового примера	Статус 200 OK. В ответе на запрос приведен один документ EducationalEntity с “id” равным “6568952d1d3c443c7202e603”, и один документ License, где поле “educationalEntity.reference” равно “6568952d1d3c443c7202e603”. Документы удовлетворяют условию так как в вывод были добавлены все существующие документы License, которые ссылаются на найденный документ EducationalEntity

В изложенных тестовых примерах результаты выполнения соответствуют ожидаемому верному поведению сервера, функционал модуля поиска отвечает предъявляемым к нему требованиям.

3.2 Негативное функциональное тестирование

Сервер обязан обрабатывать возникающие при поиске ошибки. Ошибки оборачиваются в ресурс OperationOutcome, где элемент issue хранит информацию об ошибке. Элемент severity будет иметь значение “error”, code может принимать значения “exception” (код ответа 500 Internal Server Error) и “invalid” (код ответа 400 Bad Request). Поле issue.details.text содержит текстовое описание ошибки.

Пример тела ответа в формате json:


```
{
 "resourceType": "OperationOutcome",
 "issue": [
  {
   "severity": "error",
   "code": "invalid",
   "details": {
    "text": "Параметр Organization:test не поддерживается"
   },
   "expression": [
    "url"
   ]
  } ] }
```


Тестовые примеры и результаты их выполнения приведены в таблице 45.

Таблица 45 – Результаты выполнения негативных тестов

Характеристика	Значение
Тестовый пример 1	
URL запроса	http://localhost:8002/org/Organization?test=value
Результат тестового примера	Статус 400 Bad Request. В ответе приведен ресурс OperationOutcome с severity=error, code=invalid, details.text=“Параметр Organization:test не поддерживается”. Ответ верный так как ресурс Organization действительно не имеет критерия поиска с именем “test”

Продолжение таблицы 45

Характеристика	Значение
Тестовый пример 2	
URL запроса	http://localhost:8002/org/Organization?type
Результат тестового примера	Статус 400 Bad Request. В ответе приведен ресурс OperationOutcome с severity=error, code=invalid, details.text="Не удалось разобрать значение null параметра type". Ответ верный так как не выполнены правила записи параметра
Тестовый пример 3	
URL запроса	http://localhost:8002/org/EducationalEntity?_include=entity
Результат тестового примера	Статус 400 Bad Request. В ответе приведен ресурс OperationOutcome с severity=error, code=invalid, details.text="В ссылочном параметре _include не указан тип ресурса". Ответ верный так как не выполнены правила записи значения ссылочного параметра
Тестовый пример 4	
URL запроса	http://localhost:8002/org/EducationalEntity?_include=EduGroup:entity
Результат тестового примера	Статус 400 Bad Request. В ответе приведен ресурс OperationOutcome с severity=error, code=invalid, details.text="Тип ресурса EduGroup в ссылочном параметре _include не поддерживается". Ответ верный так как EduGroup не входит в число валидных ресурсов для поля entity ресурса EducationalEntity
Тестовый пример 5	
URL запроса	http://localhost:8002/org/EducationalEntity?_id=65fad633c9e3f8764bdbc4сафф
Результат тестового примера	Статус 500 Internal Server Error. В ответе приведен ресурс OperationOutcome с severity=error, code=invalid, details.text="invalid hexadecimal representation of an ObjectId: [65fad633c9e3f8764bdbc4сафф]". Ответ верный так как не выполнены правила записи значения типа ObjectId

В изложенных тестовых примерах результаты выполнения соответствуют ожидаемому поведению сервера, функционал модуля поиска отвечает предъявляемым к нему требованиям.

ЗАКЛЮЧЕНИЕ

В рамках данной магистерской работе были выведены и зафиксированы две укорачивающие леволинейные контекстно-свободные грамматики, выведены продукционные правила вывода, разработан, отлажен и протестирован механизм продукционного вывода с опорой на типы FHIR, типы ассоциированных полей, вложенность и кратность. Библиотека соответствует предъявляемым к ней функциональным требованиям.

Решение обеспечивает функционал поиска по критериям в нереляционной базе данных под управлением СУБД MongoDB в соответствии со спецификацией FHIR, интегрировано в систему сбора и анализа данных об общеобразовательных учреждениях и может быть использовано в иных системах с похожим стеком технологий, в частности, в дальнейшем библиотека будет использоваться в системах «Реестры обучающихся» и «Реестры педагогических рабтников».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информатизация госорганов – URL: https://digital.gov.ru/ (дата обращения 12.02.2024)

2. Lea D. Concurrent Programming in Java: Design Principles and Patterns. – Addison Wesley, 2003

3. Cosmina I., Harrop R., Schaefer C., Ho C. Pro Spring 5. Thifth edition. – Apress, 2017. – 866 с.

4. FHIR v4.3.0 – URL: https://hl7.org/fhir/ (дата обращения 03.05.2024)

5. Webber J., Parastatidis S., Robinson I. REST in Practice. – O'Reilly Media, 2010. – 448 с.

6. Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. – СПб.: Питер, 2001. – 736 с.

7. https://www.mongodb.com/docs/ – URL: https://hl7.org/fhir/ (дата обращения 03.05.2024)

8. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем – СПб.:Питер, 2000 – 384 с

ПРИЛОЖЕНИЕ А

(обязательное)

Информационная система сбора и анализа данных об общеобразовательных учреждениях

Листинг программы

Файл «SearchService.java»

```
package package ru.lib.search;

import org.springframework.stereotype.Service;
import ru.lib.search.query.QueryService;
import ru.lib.types.Bundle;

@Service
public class SearchService {
    private final QueryService queryService;
    private final ResponseService responseService;

    public SearchService(QueryService queryService,
                         ResponseService responseService) {
        this.queryService = queryService;
        this.responseService = responseService;
    }

    public Bundle find(Class<?> clazz, String query) {
        SearchResult result = this.execute(clazz, query);
        return responseService.response(result);
    }

    private SearchResult execute(Class<?> clazz, String query) {
        if (query == null)
            query = "";
        return queryService.execute(clazz, query);
    }
}
```

Файл «QueryService.java»

```
package ru.lib.search.query;

import org.bson.types.ObjectId;
import org.springframework.data.domain.Sort;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.aggregation.Aggregation;
import
org.springframework.data.mongodb.core.aggregation.AggregationOperation
;
import
org.springframework.data.mongodb.core.aggregation.AggregationOptions;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.util.Pair;
import org.springframework.stereotype.Service;
import ru.lib.base.error.ApplicationException;
import ru.lib.base.util.MessageUtil;
import ru.lib.base.util.ReflectionUtil;
import ru.lib.search.ResponseService;
import ru.lib.search.SearchResult;
import ru.lib.search.query.operation.*;
import ru.lib.search.query.operation.compare.*;
```

```
import
ru.lib.search.query.operation.compare.temporal.OperationCompareDateTim
e;
import
ru.lib.search.query.operation.compare.temporal.OperationComparePeriod;
import ru.lib.types.*;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.math.BigDecimal;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZonedDateTime;
import java.util.*;
import static ru.lib.base.util.UrlUtil.urlDecode;
import static ru.lib.search.query.Modifier.EXACT;

@Service
public class QueryService {
    private final MongoTemplate mongoTemplate;
    private final MessageUtil messageUtil;
    private final ReflectionUtil reflectionUtil;
    private final CustomParamService customParamService;
    private final ResponseService responseService;

    private static final Map<String, String> generalParams = new
HashMap<>() {{
        put("_id", "id");
        put("_lastUpdated", "meta.lastUpdated");
        put("_tag", "meta.tag");
        put("_source", "meta.source");
        put("_profile", "meta.profile");
    }};

    private static final Set<String> tokenParams = new HashSet<>() {{
        add("_id");
        add("_profile");
        add("EducationalEntity:status");
        add("EducationalEntity:entity-type");
        add("PositionList:url");
        add("PositionList:academyear");
        add("PositionList:position-code");
        add("PositionList:position-unit");
        add("PositionList:status");
        add("EduGroup:url");
        add("EduGroup:academyear");
        add("EduGroup:status");
        add("EduLocation:status");
        add("EduProvider:status");
        add("EduProviderRole:status");
        add("EduProviderQualification:status");
        add("EduProviderRestriction:status");
        add("Endpoint:status");
        add("Pupil:status");
        add("PupilRole:status");
        add("NextOfKin:status");
```

```
    }};

    private final Map<String,  List<String>> compositeParams = new
HashMap<>() {{
        put("Person:email", List.of("telecom", "EMAIL"));
        put("Person:phone", List.of("telecom", "PHONE"));
    }};

    private final Map<String, List<String>> multiParams = new
HashMap<>() {{
        put("Organization:name", List.of("name",
                "alias"));
        put("EduLocation:name", List.of("name",
                "alias"));
        put("Organization:address", List.of("address.city",
                "address.district",
                "address.postalCode",
                "address.street",
                "address.text"));
        put("Person:address", List.of("address.city",
                "address.district",
                "address.postalCode",
                "address.street",
                "address.text"));
        put("EduLocation:address", List.of("address.city",
                "address.district",
                "address.postalCode",
                "address.street",
                "address.text"));
        put("Person:name", List.of("name.family",
                "name.given",
                "name.middle",
                "name.text"));
    }};

    public QueryService(MongoTemplate mongoTemplate,
                        MessageUtil messageUtil,
                        ReflectionUtil reflectionUtil,
                        CustomParamService customParamService,
                        ResponseService responseService) {
        this.mongoTemplate = mongoTemplate;
        this.messageUtil = messageUtil;
        this.reflectionUtil = reflectionUtil;
        this.customParamService = customParamService;
        this.responseService = responseService;
    }

    private boolean isToken(String className, String paramName) {
        return tokenParams.contains(paramName) ||
tokenParams.contains(className + ":" + paramName);
    }

    private Pair<String, String> getCompositeParamOrDefault(String
className, String param, String value) {
        String key = className + ":" + param;
        if (compositeParams.containsKey(key)) {
            param = compositeParams.get(key).get(0);
```

```
            value = compositeParams.get(key).get(1) + "|" + value;
        }
        return Pair.of(param,value);
    }

    public List<String> getMultiParam(String className, String
paramName) {
        return multiParams.getOrDefault(className + ":" + paramName,
List.of(paramName));
    }

    @SuppressWarnings("unchecked")
    public SearchResult execute(Class<?> clazz, String query) {
        List<Operation> andList = new ArrayList<>();
        long count = 10L;
        long offset = 0L;
        String sort = null;
        String elements = null;
        Map<String, String> params = new LinkedHashMap<>();
        List<String> includeList = new ArrayList<>();
        List<String> revincludeList = new ArrayList<>();
        for (String subQuery : query.split("&")) {
            if (subQuery.isEmpty())
                break;
            String[] paramAndValue = subQuery.split("=");
            if (paramAndValue.length < 2)
                throw new ApplicationException(

messageUtil.getMessage("search.ValueNotParsed", "null",
paramAndValue[0]), List.of("url"));
            if (paramAndValue[0].equals("access_token")) {
                continue;
            }
            List<Operation> orList = new ArrayList<>();
            params.put(paramAndValue[0], paramAndValue[1]);
            if (paramAndValue[0].equals("_count")) {
                count = Long.parseLong(paramAndValue[1]);
                if (count > responseService.getMaxPageSize())
                    count = responseService.getMaxPageSize();
                continue;
            }
            if (paramAndValue[0].equals("_offset")) {
                offset = Long.parseLong(paramAndValue[1]);
                continue;
            }
            if (paramAndValue[0].equals("_format")) {
                continue;
            }
            if (paramAndValue[0].equals("_sort")) {
                sort = paramAndValue[1];
                continue;
            }
            if (paramAndValue[0].equals("_elements")) {
                elements = paramAndValue[1];
                continue;
            }
            if (paramAndValue[0].equals("_include")) {
```

```
                includeList.add(urlDecode(paramAndValue[1]));
                continue;
            }
            if (paramAndValue[0].equals("_revinclude")) {
                revincludeList.add(urlDecode(paramAndValue[1]));
                continue;
            }
            String multiParam = paramAndValue[0];
            Modifier modifier = null;
            paramAndValue[0] = urlDecode(paramAndValue[0]);
            paramAndValue[1] = urlDecode(paramAndValue[1]);
            for (Modifier mod : Modifier.values()) {
                if (paramAndValue[0].contains(mod.getValue())) {
                    modifier = mod;
                    multiParam =
paramAndValue[0].replace(modifier.getValue(), "");
                    break;
                }
            }
            if (isToken(clazz.getSimpleName(), multiParam) && modifier
!= Modifier.MISSING)
                modifier = EXACT;
            if (generalParams.containsKey(multiParam))
                multiParam = generalParams.get(multiParam);
            else
                multiParam =
customParamService.getCustomParam(clazz.getSimpleName(), multiParam);
            for (String param : getMultiParam(clazz.getSimpleName(),
multiParam)) {
                for (String subValue : paramAndValue[1].split(",")) {
                    List<Operation> compositeList = new ArrayList<>();
                    Pair<String, String> paramAndValueWithSystem =
this.getCompositeParamOrDefault(clazz.getSimpleName(), multiParam,
subValue);
                    if
(!subValue.equals(paramAndValueWithSystem.getSecond())) {
                        param = paramAndValueWithSystem.getFirst();
                        subValue =
paramAndValueWithSystem.getSecond();
                    }
                    for (String simpleValue : subValue.split("\\$")) {
                        if (modifier != null &&
modifier.equals(Modifier.MISSING))
                            compositeList.add(new
OperationMissing(param, Boolean.parseBoolean(simpleValue)));
                        else
                            compositeList.add(createOperation(clazz,
modifier, param, simpleValue));
                    }
                    orList.add(compositeList.size() > 1 ? new
OperationAnd(compositeList) : compositeList.get(0));
                }
            }
            andList.add(orList.size() > 1 ? new OperationOr(orList) :
orList.get(0));
        }
        Criteria criteria = null;
```

```
        if (!andList.isEmpty()) {
            Operation operation = andList.size() > 1 ? new
OperationAnd(andList) : andList.get(0);
            criteria = operation.toCriteria();
        }
        long total = criteria != null ?
mongoTemplate.count(Query.query(criteria), clazz)
                : mongoTemplate.estimatedCount(clazz);
        List<Resource> documents = new ArrayList<>();
        List<Resource> includeDocuments = new ArrayList<>();
        if (count != 0) {
            List<AggregationOperation> aggregationOperations = new
ArrayList<>();

//aggregationOperations.add(SetOperation.builder().set("id").withValue
OfExpression("toString(_id)"));
            if (criteria != null) {

aggregationOperations.add(Aggregation.match(criteria));
            }
            if (sort != null) {
                List<Sort.Order> orders = new ArrayList<>();
                for (String param : sort.split(",")) {
                    Sort.Direction direction = Sort.Direction.ASC;
                    if (param.charAt(0) == '-') {
                        param = param.substring(1);
                        direction = Sort.Direction.DESC;
                    }
                    if (generalParams.containsKey(param))
                        param = generalParams.get(param);
                    else
                        param =
customParamService.getCustomParam(clazz.getSimpleName(), param);
                    orders.add(new Sort.Order(direction, param));
                }

aggregationOperations.add(Aggregation.sort(Sort.by(orders)));
            }
            aggregationOperations.add(Aggregation.skip(offset));
            aggregationOperations.add(Aggregation.limit(count));
            AggregationOptions aggregationOptions =
AggregationOptions.builder().allowDiskUse(true).build();
            Aggregation aggregation =
Aggregation.newAggregation(aggregationOperations).withOptions(aggregat
ionOptions);
            documents.addAll((List<Resource>)
mongoTemplate.aggregate(aggregation, getCollectionNameByClass(clazz),
clazz)
                    .getMappedResults());
            includeDocuments = getIncludeDocuments(clazz, includeList,
documents);

includeDocuments.addAll(getRevincludeDocuments(revincludeList,
documents));
            if (elements != null) {
                Set<String> elementSet = new HashSet<>();
                Collections.addAll(elementSet, elements.split(","));
```

```
                elementSet.add("id");
                Set<Field> fields = new HashSet<>(getAllFields(new
LinkedList<>(), clazz));
                Set<Field> nullFields = new HashSet<>();
                for (Field field : fields) {
                    if (!elementSet.contains(field.getName()))
                        field.setAccessible(true);
                    nullFields.add(field);
                }
                for (Resource document : documents) {
                    for (Field nullField : nullFields) {
                        try {
                            nullField.set(document, null);
                        } catch (IllegalAccessException e) {
                        }
                    }
                }
            }
        }
        params.put("_offset", Long.toString(offset));
        params.put("_count", Long.toString(count));
        HashMap<String, Resource> uniqueIncludeDocuments = new
HashMap<>();
        includeDocuments.forEach(document ->
uniqueIncludeDocuments.put(document.getId(), document));
        includeDocuments = new
ArrayList<>(uniqueIncludeDocuments.values());
        return new SearchResult(documents, total,
clazz.getSimpleName(), params, includeDocuments);
    }

    private Object getValueByField(Object o, String param) {
        Class<?> clazz = o.getClass();
        List<Object> objs = new ArrayList<>();
        objs.add(o);
        try {
            String[] fields = param.split("\\.");
            for (String s : fields) {
                Field field = getField(clazz, s).get();
                field.setAccessible(true);
                List<Object> newObjs = new ArrayList<>();
                for (Object obj : objs) {
                    obj = field.get(obj);
                    if (obj != null) {
                        if (obj instanceof List)
                            newObjs.addAll((List<?>) obj);
                        else
                            newObjs.add(obj);
                    }
                }
                objs = newObjs;
                clazz = getGeneric(clazz, s);
            }
            if (objs.isEmpty())
                return null;
            else if (objs.size() == 1)
                return objs.get(0);
```

```
            else
                return objs;
        } catch (Exception e) {
            throw new
ApplicationException(messageUtil.getMessage("search.ParamNotFound",
param), List.of("url"));
        }
    }

    private List<Resource> getIncludeDocuments(Class<?> clazz,
List<String> includeList, List<Resource> documents) {
        List<Resource> includeDocuments = new ArrayList<>();
        includeList.forEach(p -> {
            String[] typeAndParam = p.split(":");
            if (typeAndParam.length < 2 || typeAndParam[0].equals(""))
                throw new
ApplicationException(messageUtil.getMessage("search.TypeMissed",
"_include"), List.of("url"));
            String type = typeAndParam[0];
            String param = typeAndParam[1];
            if (!clazz.getSimpleName().equals(type))
                throw new
ApplicationException(messageUtil.getMessage("search.IncorrectType",
type, "_include"), List.of("url"));
            param =
customParamService.getCustomParam(clazz.getSimpleName(), param);
            Class<?> paramClazz = getParamType(clazz, param);
            if (paramClazz.equals(Reference.class)) {
                Map<String, Set<String>> ids = new HashMap<>();
                for (Resource document : documents) {
                    Object o = getValueByField(document, param);
                    if (o instanceof List) {
                        ((List<Reference<?>>) o).forEach(ref -> {
                            if (ref != null) {
                                String name = ref.getType();
                                if (!ids.containsKey(name))
                                    ids.put(name, new HashSet<>());

ids.get(name).add(ref.getRef().getId().toString());
                            }
                        });
                    } else {
                        Reference<?> ref = (Reference<?>) o;
                        if (ref != null) {
                            String name = ref.getType();
                            if (!ids.containsKey(name))
                                ids.put(name, new HashSet<>());

ids.get(name).add(ref.getRef().getId().toString());
                        }
                    }
                }
                ids.forEach((name, idSet) -> {
                    if (idSet.size() > 0) {
                        Class<?> refType =
reflectionUtil.getClassBySimpleName(name);
                        includeDocuments.addAll(
```

```
                                    (List<Resource>)
mongoTemplate.find(Query.query(new Criteria("_id").in(idSet)),
refType));
                    }
                });
            }
        });
        return includeDocuments;
    }

    private List<Resource> getRevincludeDocuments(List<String>
revincludeList, List<Resource> documents) {
        List<Resource> revincludeDocuments = new ArrayList<>();
        revincludeList.forEach(p -> {
            String[] typeAndParam = p.split(":");
            if (typeAndParam.length < 2 || typeAndParam[0].equals(""))
                throw new
ApplicationException(messageUtil.getMessage("search.TypeMissed",
"_revinclude"), List.of("url"));
            String type = typeAndParam[0];
            String param = typeAndParam[1];
            Class<?> clazz =
reflectionUtil.getClassBySimpleName(type);
            param =
customParamService.getCustomParam(clazz.getSimpleName(), param);
            Class<?> paramClazz = getParamType(clazz, param);
            if (paramClazz.equals(Reference.class)) {
                Set<ObjectId> ids = new HashSet<>();
                documents.forEach(document -> ids.add(new
ObjectId(document.getId())));
                if (ids.size() > 0) {
                    revincludeDocuments.addAll(
                            (List<Resource>) mongoTemplate.find(
                                    Query.query(new Criteria(param +
".ref.$id").in(ids)), clazz));
                }
            }
        });
        return revincludeDocuments;
    }

    private String getCollectionNameByClass(Class<?> clazz) {
        return
clazz.getDeclaredAnnotation(Document.class).collection();
    }

    private List<Field> getAllFields(List<Field> fields, Class<?>
type) {
        fields.addAll(Arrays.asList(type.getDeclaredFields()));

        if (type.getSuperclass() != null) {
            getAllFields(fields, type.getSuperclass());
        }

        return fields;
    }
```

```
    private Optional<Field> getField(Class<?> clazz, String fieldName)
{
        List<Field> fields = getAllFields(new LinkedList<>(), clazz);
        for (Field field : fields) {
            if (field.getName().equals(fieldName))
                return Optional.of(field);
        }
        return Optional.empty();
    }


    private Class<?> getGeneric(Class<?> clazz, String field) {
        Type type = getField(clazz, field).orElseThrow(() ->
                new RuntimeException("Field not
found")).getGenericType();
        while (type instanceof ParameterizedType) {
            ParameterizedType parameterizedType = ((ParameterizedType)
type);
            if (parameterizedType.getRawType().equals(List.class) ||
parameterizedType.getRawType().equals(Set.class)) {
                type = parameterizedType.getActualTypeArguments()[0];
            } else {
                return (Class<?>) parameterizedType.getRawType();
            }
        }
        return (Class<?>) type;
    }

    private Class<?> getParamType(Class<?> clazz, String param) {
        try {
            String[] fields = param.split("\\.");
            for (int i = 0; i < fields.length - 1; i++) {
                clazz = getGeneric(clazz, fields[i]);
            }
            return getGeneric(clazz, fields[fields.length - 1]);
        } catch (Exception e) {
            throw new
ApplicationException(messageUtil.getMessage("search.ParamNotFound",
param), List.of("url") );
        }
    }

    private boolean isList(Class<?> clazz, String param) {
        try {
            String[] fields = param.split("\\.");
            for (int i = 0; i < fields.length - 1; i++) {
                clazz = getGeneric(clazz, fields[i]);
            }
            Type type = getField(clazz, fields[fields.length -
1]).orElseThrow(() ->
                    new RuntimeException("Field not
found")).getGenericType();
            if (type instanceof ParameterizedType) {
                ParameterizedType parameterizedType =
((ParameterizedType) type);
                return
parameterizedType.getRawType().equals(List.class)
```

```
                        ||
parameterizedType.getRawType().equals(Set.class);
            } else {
                return false;
            }
        } catch (Exception e) {
            throw new
ApplicationException(messageUtil.getMessage("search.ParamNotFound",
param), List.of("url") );
        }
    }

    private Operation createOperation(Class<?> clazz, Modifier
modifier, String param, String valueWithPrefix) {
        Prefix prefix = null;
        Operation operation = null;
        Class<?> paramType = getParamType(clazz, param);
        try {
            if (paramType.equals(BigDecimal.class) ||
Number.class.isAssignableFrom(paramType)) {
                BigDecimal value;
                if (valueWithPrefix.length() > 2) {
                    prefix =
Prefix.getByValue(valueWithPrefix.substring(0, 2));
                }
                if (prefix != null) {
                    value = new
BigDecimal(valueWithPrefix.substring(2));
                } else {
                    value = new BigDecimal(valueWithPrefix);
                }
                operation = new OperationCompareDecimal(param, prefix,
value);
            } else if (paramType.equals(String.class) ||
paramType.isEnum()) {
                if (paramType.isEnum()) {
                    Method getValue = null;
                    try {
                        getValue =
paramType.getDeclaredMethod("getValue");
                    } catch (Exception ignore) {
                    }
                    String findValue = valueWithPrefix;
                    if (getValue != null) {
                        for (Object enumConstant :
paramType.getEnumConstants()) {
                            String value = (String)
getValue.invoke(enumConstant);
                            if (value.equals(findValue)) {
                                findValue = enumConstant.toString();
                                break;
                            }
                        }
                    }
                    operation = new OperationCompareString(param,
EXACT, findValue);
                } else {
```

```
                    if ("id".equals(param))
                        operation = new
OperationCompareId(valueWithPrefix);
                    else
                        operation = new OperationCompareString(param,
modifier, valueWithPrefix);
                }
            } else if (paramType.equals(Boolean.class)) {
                Boolean value = null;
                if (valueWithPrefix.equalsIgnoreCase("true"))
                    value = true;
                else if (valueWithPrefix.equalsIgnoreCase("false"))
                    value = false;
                if (value != null)
                    operation = new OperationCompareBoolean(param,
value);
                else
                    operation = new OperationCompareString(param,
EXACT, valueWithPrefix);
            } else if (paramType.equals(LocalDate.class) ||
paramType.equals(LocalDateTime.class)
                    || paramType.equals(ZonedDateTime.class) ||
paramType.equals(Period.class)) {
                String value = valueWithPrefix;
                prefix =
Prefix.getByValue(valueWithPrefix.substring(0, 2));
                if (prefix != null) {
                    value = valueWithPrefix.substring(2);
                }
                if (paramType.equals(Period.class))
                    operation = new OperationComparePeriod(param,
prefix, value);
                else
                    operation = new OperationCompareDateTime(param,
prefix, value);
            } else if (paramType.equals(Reference.class) && modifier
!= Modifier.IDENTIFIER) {
                operation = new OperationCompareReference(param,
valueWithPrefix);
            } else {
                UUID uuid = UUID.randomUUID();
                String[] systemAndCode = (uuid + valueWithPrefix +
uuid).split("\\|");
                String code, system;
                if (systemAndCode.length == 1) {
                    code =
systemAndCode[0].replaceAll(uuid.toString(), "");
                    system = null;
                } else {
                    system =
systemAndCode[0].replaceAll(uuid.toString(), "");
                    code =
systemAndCode[1].replaceAll(uuid.toString(), "");
                }
                operation = this.createOperationToken(clazz, "",
param, system, code);
            }
```

```
            if (operation == null) {
                throw new ApplicationException("The structure is
specified instead of a field");
            }
        } catch (Exception e) {
            throw new
ApplicationException(messageUtil.getMessage("search.ValueNotParsed",
valueWithPrefix, param), List.of("url"));
        }
        return operation;
    }

    private Operation createOperationToken(Class<?> clazz, String
prefix, String param, String system, String code) {
        String head, tail = "";
        String[] params = param.split(("\\."));
        if (params.length > 1){
            tail = param.replace(params[0] + ".", "");
        }
        head = params[0];
        Class<?> paramType = getParamType(clazz, head);
        Boolean isList = isList(clazz, head);

        if (paramType.equals(Coding.class)) {
            return new OperationCompareToken(prefix + head, isList,
"system", "code", system, code);
        } else if (paramType.equals(Identifier.class) ||
paramType.equals(ContactPoint.class)) {
            return new OperationCompareToken(prefix + head, isList,
"system", "value", system, code);
        } else {
            if (paramType.equals(Reference.class)) {
                tail = "identifier";
            } else if (paramType.equals(CodeableConcept.class)) {
                tail = "coding";
            }
            Operation operation;
            if (isList) {
                operation = this.createOperationToken(paramType, "",
tail, system, code);
                operation = new OperationList(prefix + head,
operation);
            } else {
                operation = this.createOperationToken(paramType,
prefix + head + ".", tail, system, code);
            }
            return operation;
        }
    }
}
```

Файл «Prefix.java»

```
package ru.lib.search.query;

public enum Prefix {
```

```
    EQ("eq"),
    NE("ne"),
    GT("gt"),
    LT("lt"),
    GE("ge"),
    LE("le");

    private final String value;

    Prefix(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }

    public static Prefix getByValue(String value) {
        for (Prefix prefix : Prefix.values()) {
            if (prefix.getValue().equals(value))
                return prefix;
        }
        return null;
    }
}
```

Файл «Modifier.java»

```
package ru.lib.search.query;

public enum Modifier {
    MISSING(":missing"),
    EXACT(":exact"),
    CONTAINS(":contains"),
    IDENTIFIER(":identifier");

    private final String value;

    Modifier(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}
```

Файл «CustomParamService.java»

```
package ru.lib.search.query;

import org.springframework.stereotype.Service;
import ru.lib.base.error.ApplicationException;
import ru.lib.base.util.MessageUtil;
import ru.lib.base.util.ReflectionUtil;
```

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Service
public class CustomParamService {

    private final MessageUtil messageUtil;
    private final ReflectionUtil reflectionUtil;

    public enum RegType {
        ORG,
        EDU,
        PUPIL
    }

    public CustomParamService(MessageUtil messageUtil,
                              ReflectionUtil reflectionUtil) {
        this.messageUtil = messageUtil;
        this.reflectionUtil = reflectionUtil;
    }

    private static final Map<RegType, Map<String, String>> params =
new HashMap<>() {{
        put(RegType.ORG, //ОООД
                new HashMap<>() {{
                    put("EducationalEntity:identifier", "identifier");
                    put("EducationalEntity:entity", "entity");
                    put("EducationalEntity:parent", "parent");
                    put("EducationalEntity:member", "member");
                    put("EducationalEntity:predecessor",
"predecessor");
                    put("EducationalEntity:status", "status");

                    put("EducationalEntity:maineconomicactivity",
"mainEconomicActivity");
                    put("EducationalEntity:othereconomicactivity",
"otherEconomicActivity");
                    put("EducationalEntity:entity-type",
"entity.type");

                    put("Organization:active", "active");
                    put("Organization:address", "address");

                    put("Organization:address-city", "address.city");
                    put("Organization:address-country",
"address.country");
                    put("Organization:address-postalcode",
"address.postalCode");
                    put("Organization:address-state",
"address.state");
                    put("Organization:address-use", "address.use");

                    put("Organization:identifier", "identifier");
                    put("Organization:name", "name");

                    put("Organization:partof", "partOf");
```

```
                    put("Organization:type", "type");

                    put("Organization:executivename",
"executiveName");
                    put("Organization:executiveposition",
"executivePosition");

                    put("Person:address", "address");

                    put("Person:address-city", "address.city");
                    put("Person:address-country", "address.country");
                    put("Person:address-postalcode",
"address.postalCode");
                    put("Person:address-state", "address.state");
                    put("Person:address-use", "address.use");
                    put("Person:email", "email");

                    put("Person:identifier", "identifier");
                    put("Person:name", "name");

                    put("Person:phone", "phone");

                    put("Person:telecom", "telecom");

                    put("Person:link", "link.target");

                    put("License:identifier", "identifier");

                    put("License:start", "identifier.period.start");
                    put("License:end", "identifier.period.end");

                    put("License:assigner",
"identifier.assignerDisplay");

                    put("License:address", "address");

                    put("License:mediaid", "mediaId");

                    put("License:entity", "educationalEntity");
                    put("License:replaces", "insteadOfLicense");

                    put("License:type", "type");
                    put("License:status", "status");

                    put("LicenseSupplement:entity",
"educationalEntity");

                    put("LicenseSupplement:identifier", "identifier");
                    put("LicenseSupplement:license", "license");

                    put("LicenseSupplement:location", "eduLocation");

                    put("LicenseSupplement:mediaid", "mediaId");

                    put("LicenseSupplement:status", "status");
```

```
                    put("LicenseSupplement:level",
"licensedProgram.level");

                    put("AccreditationCertificate:identifier",
"identifier");

                    put("AccreditationCertificate:start",
"identifier.period.start");
                    put("AccreditationCertificate:end",
"identifier.period.end");

                    put("AccreditationCertificate:assigner",
"identifier.assignerDisplay");

                    put("AccreditationCertificate:mediaid",
"mediaId");

                    put("AccreditationCertificate:entity",
"educationalEntity");

                    put("AccreditationCertificate:replaces",
"insteadOfCertificate");

                    put("AccreditationCertificate:status", "status");

                    put("AccreditationSupplement:entity",
"educationalEntity");

                    put("AccreditationSupplement:identifier",
"identifier");

                    put("AccreditationSupplement:certificate",
"accreditationCertificate");

                    put("AccreditationSupplement:mediaid", "mediaId");

                    put("AccreditationSupplement:level",
"educationalProgram.level");
                    put("AccreditationSupplement:status", "status");

                    put("AccreditationSupplement:program-status",
"educationalProgram.status");

                    put("PositionList:url", "url");
                    put("PositionList:version", "version");
                    put("PositionList:identifier", "identifier");

                    put("PositionList:entity", "educationalEntity");

                    put("PositionList:academyear", "academYear");

                    put("PositionList:position-code",
"position.code");
                    put("PositionList:position-name",
"position.name");
                    put("PositionList:position-unit",
"position.organizationUnit");
```

```
                    put("PositionList:status", "status");
                    put("EduGroup:url", "url");
                    put("EduGroup:version", "version");
                    put("EduGroup:identifier", "identifier");

                    put("EduGroup:entity", "educationalEntity");

                    put("EduGroup:academyear", "academYear");

                    put("EduGroup:group-type", "group.type");
                    put("EduGroup:group-name", "group.name");
                    put("EduGroup:location", "group.eduLocation");

                    put("EduGroup:status", "status");
                    put("EduLocation:identifier", "identifier");
                    put("EduLocation:name", "name");
                    put("EduLocation:alias", "name");

                    put("EduLocation:custodian",
"managingOrganization");
                    put("EduLocation:entity", "educationalEntity");

                    put("EduLocation:physicaltype", "physicalType");

                    put("EduLocation:address", "address");

                    put("EduLocation:address-city", "address.city");
                    put("EduLocation:address-country",
"address.country");
                    put("EduLocation:address-postalcode",
"address.postalCode");
                    put("EduLocation:address-state", "address.state");
                    put("EduLocation:status", "status");

                    put("Event:type", "type");
                    put("Event:reason", "reason");

                    put("Event:docid", "docId");
                    put("Event:docname", "docName");
                    put("Event:docdate", "docDate");
                    put("Event:subject", "subject.subject");
                }});
    }};

    public String getCustomParam(String className, String paramName) {
        String param = className + ":" + paramName;
        RegType regType = RegType.ORG;
        if (!params.get(regType).containsKey(param))
            throw new
ApplicationException(messageUtil.getMessage("search.ParamNotFound",
param), List.of("url"));
        return params.get(regType).get(param);
    }
}
```

Файл «OperationSimple.java»

```
package ru.lib.search.query.operation;

import org.springframework.data.mongodb.core.query.Criteria;

public class OperationSimple extends Operation{

    private final String param;
    private final Operation operation;


    public OperationSimple(String param, Operation operation) {
        this.param = param;
        this.operation = operation;
    }

    @Override
    public Criteria toCriteria() {
        return new
Criteria(param).andOperator(this.operation.toCriteria());
    }
}
```

Файл «OperationOr.java»

```
package ru.lib.search.query.operation;

import org.springframework.data.mongodb.core.query.Criteria;
import java.util.List;
import java.util.stream.Collectors;

public class OperationOr extends Operation {
    private final List<Operation> operationList;

    public OperationOr(List<Operation> operationList) {
        this.operationList = operationList;
    }

    @Override
    public Criteria toCriteria() {
        return new Criteria().orOperator(
                operationList.stream().map(operation ->
operation.toCriteria()).collect(Collectors.toList()));
    }
}
```

Файл «OperationMissing.java»

```
package ru.lib.search.query.operation;

import org.springframework.data.mongodb.core.query.Criteria;

public class OperationMissing extends Operation{
```

```
    private final String param;
    private final Boolean value;

    public OperationMissing(String param, Boolean value) {
        this.param = param;
        this.value = value;
    }

    @Override
    public Criteria toCriteria() {
        return new Criteria(this.param).exists(!this.value);
    }
}
```

Файл «OperationList.java»

```
package ru.lib.search.query.operation;

import org.springframework.data.mongodb.core.query.Criteria;

public class OperationList extends Operation{

    private final String param;
    private final Operation operation;

    public OperationList(String param, Operation operation) {
        this.param = param;

        this.operation = operation;
    }

    @Override
    public Criteria toCriteria() {
        return new
Criteria(this.param).elemMatch(this.operation.toCriteria());
    }
}
```

Файл «OperationAnd.java»

```
package ru.lib.search.query.operation;

import org.springframework.data.mongodb.core.query.Criteria;
import java.util.List;
import java.util.stream.Collectors;

public class OperationAnd extends Operation {
    private final List<Operation> operationList;

    public OperationAnd(List<Operation> operationList) {
        this.operationList = operationList;
    }

    @Override
    public Criteria toCriteria() {
```

```
        return new Criteria().andOperator(
                operationList.stream().map(operation ->
operation.toCriteria()).collect(Collectors.toList()));
    }
}
```

Файл «Operation.java»

```
package ru.lib.search.query.operation;

import org.springframework.data.mongodb.core.query.Criteria;


public abstract class Operation {

    public abstract Criteria toCriteria();

}
```

Файл «OperationCompareToken.java»

```
package ru.lib.search.query.operation.compare;

import org.springframework.data.mongodb.core.query.Criteria;

public class OperationCompareToken extends OperationCompare {
    private final String path;
    private final boolean isArray;
    private final String systemParam;
    private final String codeParam;
    private final String system;
    private final String code;

    public OperationCompareToken(String path, boolean isArray,
                         String systemParam, String codeParam,
                         String system, String code) {
        this.path = path;
        this.isArray = isArray;
        this.systemParam = systemParam;
        this.codeParam = codeParam;
        this.system = system;
        this.code = code;
    }

    @Override
    public Criteria toCriteria() {
        Criteria criteria;
        Criteria subCriteria;
        Criteria systemCriteria = null;
        Criteria codeCriteria = null;
        String systemParam = this.systemParam;
        String codeParam = this.codeParam;
        if (!this.isArray) {
            if (systemParam != null) {
                systemParam = this.path + "." + systemParam;
            }
```

```
            codeParam = this.path + "." + codeParam;
        }
        if (this.system != null) {
            if (this.system.isEmpty()) {
                systemCriteria =
Criteria.where(systemParam).exists(false);
            } else {
                systemCriteria =
Criteria.where(systemParam).is(this.system);
            }
        }
        if (!this.code.isEmpty()) {
            codeCriteria = Criteria.where(codeParam).is(this.code);
        }
        if (systemCriteria == null)
            subCriteria = codeCriteria;
        else if (codeCriteria == null)
            subCriteria = systemCriteria;
        else
            subCriteria = new Criteria().andOperator(systemCriteria,
codeCriteria);
        if (this.isArray) {
            criteria =
Criteria.where(this.path).elemMatch(subCriteria);
        } else {
            criteria = subCriteria;
        }
        return criteria;
    }
}
```

Файл «OperationCompareString.java»

```
package ru.lib.search.query.operation.compare;

import org.springframework.data.mongodb.core.query.Criteria;
import ru.lib.search.query.Modifier;
import java.util.regex.Pattern;

public class OperationCompareString extends OperationCompare {

    private final String param;
    private final Modifier modifier;
    private final Object value;

    public OperationCompareString(String param, Modifier modifier,
Object value) {
        this.param = param;
        this.modifier = modifier;
        this.value = value;
    }

    @Override
    public Criteria toCriteria() {
        Criteria criteria = new Criteria(this.param);
        String strValue = Pattern.quote((String) this.value);
```

```
        if (this.modifier != null) {
            switch (this.modifier) {
                case CONTAINS:
                    criteria = criteria.regex(strValue, "i");
                    break;
                case EXACT:
                    criteria = criteria.is(this.value);
            }
        } else {
            criteria = criteria.regex("^" + strValue, "i");
        }
        return criteria;
    }
}
```

Файл «OperationCompareReference.java»

```
package ru.lib.search.query.operation.compare;

import org.bson.types.ObjectId;
import org.springframework.data.mongodb.core.query.Criteria;

public class OperationCompareReference extends OperationCompare {

    private final String param;
    private final String value;

    public OperationCompareReference(String param, String value) {
        this.param = param;
        this.value = value;
    }

    @Override
    public Criteria toCriteria() {
        return Criteria.where(this.param + ".ref.$id").is(new
ObjectId(this.value));
    }
}
```

Файл «OperationCompareId.java»

```
package ru.lib.search.query.operation.compare;

import org.bson.types.ObjectId;
import org.springframework.data.mongodb.core.query.Criteria;

public class OperationCompareId extends OperationCompare{
    private final String value;

    public OperationCompareId(String value) {
        this.value = value;
    }

    @Override
    public Criteria toCriteria() {
```

```
        return Criteria.where("_id").is(new ObjectId(this.value));
    }
}
```

Файл «OperationCompareDecimal.java»

```
package ru.lib.search.query.operation.compare;

import org.springframework.data.mongodb.core.query.Criteria;
import ru.lib.search.query.Prefix;

public class OperationCompareDecimal extends OperationCompare {

    private final String param;
    private final Prefix prefix;
    private final Object value;

    public OperationCompareDecimal(String param, Prefix prefix, Object
value) {
        this.param = param;
        this.prefix = prefix != null ? prefix : Prefix.EQ;
        this.value = value;
    }

    @Override
    public Criteria toCriteria() {
        Criteria criteria = Criteria.where(this.param);
        switch (this.prefix) {
            case EQ:
                criteria = criteria.is(this.value);
                break;
            case NE:
                criteria = criteria.ne(this.value);
                break;
            case GT:
                criteria = criteria.gt(this.value);
                break;
            case LT:
                criteria = criteria.lt(this.value);
                break;
            case GE:
                criteria = criteria.gte(this.value);
                break;
            case LE:
                criteria = criteria.lte(this.value);
        }
        return criteria;
    }
}
```

Файл «OperationCompareBoolean.java»

```
package ru.lib.search.query.operation.compare;

import org.springframework.data.mongodb.core.query.Criteria;
```

```
public class OperationCompareBoolean extends OperationCompare {

    private final String param;
    private final Boolean value;

    public OperationCompareBoolean(String param, Boolean value) {
        this.param = param;
        this.value = value;
    }

    @Override
    public Criteria toCriteria() {
        return Criteria.where(this.param).is(this.value);
    }
}
```

Файл «OperationCompare.java»

```
package ru.lib.search.query.operation.compare;

import ru.lib.search.query.operation.Operation;

public abstract class OperationCompare extends Operation {

}
```

Файл «OperationCompareTemporal.java»

```
package ru.lib.search.query.operation.compare.temporal;

import ru.lib.search.query.Prefix;
import ru.lib.search.query.operation.compare.OperationCompare;
import java.time.LocalDateTime;
import java.time.ZonedDateTime;
import java.time.temporal.ChronoUnit;

public abstract class OperationCompareTemporal extends
OperationCompare {

    private final String param;
    private final Prefix prefix;
    private final LocalDateTime localDateTimeStart;
    private final LocalDateTime localDateTimeEnd;

    public OperationCompareTemporal(String param, Prefix prefix,
String value) {
        this.param = param;
        this.prefix = prefix != null ? prefix : Prefix.EQ;
        String[] dateTimeAndZone = value.split("\\+");
        String[] dateAndTime = dateTimeAndZone[0].split("T");
        String[] date = dateAndTime[0].split("-");
        ChronoUnit chronoUnit = null;
        int year;
        int month = 1;
        int day = 1;
        switch (3 - date.length) {
            case 0:
```

```
                chronoUnit = ChronoUnit.DAYS;
                day = Integer.parseInt(date[2]);
            case 1:
                chronoUnit = chronoUnit == null ? ChronoUnit.MONTHS :
chronoUnit;
                month = Integer.parseInt(date[1]);
        }
        chronoUnit = chronoUnit == null ? ChronoUnit.YEARS :
chronoUnit;
        year = Integer.parseInt(date[0]);
        int hour = 0;
        int minute = 0;
        int second = 0;
        int millis = 0;
        if (dateAndTime.length > 1) {
            dateAndTime[1] = dateAndTime[1].split("Z")[0];
            String[] time = dateAndTime[1].split(":");
            hour = Integer.parseInt(time[0]);
            minute = Integer.parseInt(time[1]);
            chronoUnit = ChronoUnit.MINUTES;
            if (time.length > 2) {
                String[] secondAndMills = time[2].split("\\.");
                second = Integer.parseInt(secondAndMills[0]);
                chronoUnit = ChronoUnit.SECONDS;
                if (secondAndMills.length > 1) {
                    millis = Integer.parseInt(secondAndMills[1]);
                    chronoUnit = ChronoUnit.MILLIS;
                }
            }
        }
        String text = String.format("%4d-%02d-
%02dT%02d:%02d:%02d.%03d",
                year, month, day, hour, minute, second, millis);
        if (dateTimeAndZone.length > 1)
            text += "+" + dateTimeAndZone[1];
        else
            text += "+00:00";
        localDateTimeStart =
ZonedDateTime.parse(text).toLocalDateTime();
        localDateTimeEnd = localDateTimeStart.plus(1, chronoUnit);
    }

    protected String getParam() {
        return param;
    }

    protected Prefix getPrefix() {
        return prefix;
    }

    protected LocalDateTime getLocalDateTimeStart() {
        return localDateTimeStart;
    }

    protected LocalDateTime getLocalDateTimeEnd() {
        return localDateTimeEnd;
    }
```

```
}
```

Файл «OperationComparePeriod.java»

```
package ru.lib.search.query.operation.compare.temporal;

import org.springframework.data.mongodb.core.query.Criteria;
import ru.lib.search.query.Prefix;
import java.time.LocalDateTime;

public class OperationComparePeriod extends OperationCompareTemporal {

    public OperationComparePeriod(String param, Prefix prefix, String
value) {
        super(param, prefix, value);
    }

    @Override
    public Criteria toCriteria() {
        Criteria criteriaStart = Criteria.where(getParam() + ".start");
        Criteria criteriaEnd = Criteria.where(getParam() + ".end");
        Criteria criteria = null;
        LocalDateTime start = getLocalDateTimeStart();
        LocalDateTime end = getLocalDateTimeEnd();
        Prefix prefix = getPrefix();
        switch (prefix) {
            case EQ:
                criteria                    =                    new
Criteria().andOperator(criteriaStart.lt(end), criteriaEnd.gt(start));
                break;
            case NE:
                criteria                    =                    new
Criteria().orOperator(criteriaStart.gte(end), criteriaEnd.lte(start));
                break;
            case GT:
                criteria = criteriaStart.gte(end);
                break;
            case LT:
                criteria = criteriaEnd.lte(start);
                break;
            case GE:
                criteria = criteriaEnd.gt(end);
                break;
            case LE:
                criteria = criteriaStart.lt(start);
                break;
        }
        return criteria;
    }
}
```

Файл «OperationCompareDateTime.java»

```
package ru.lib.search.query.operation.compare.temporal;

import org.springframework.data.mongodb.core.query.Criteria;
import ru.lib.search.query.Prefix;
```

```
import java.time.LocalDateTime;

public class OperationCompareDateTime extends OperationCompareTemporal
{

    public OperationCompareDateTime(String param, Prefix prefix,
String value) {
        super(param, prefix, value);
    }

    @Override
    public Criteria toCriteria() {
        Criteria criteria = Criteria.where(getParam());
        LocalDateTime start = getLocalDateTimeStart();
        LocalDateTime end = getLocalDateTimeEnd();
        Prefix prefix = getPrefix();
        switch (prefix) {
            case EQ:
                criteria = new
Criteria().andOperator(criteria.gte(start), criteria.lt(end));
                break;
            case NE:
                criteria = new
Criteria().orOperator(criteria.lt(start), criteria.gte(end));
                break;
            case GT:
                criteria = criteria.gte(end);
                break;
            case LT:
                criteria = criteria.lt(start);
                break;
            case GE:
                criteria = criteria.gte(start);
                break;
            case LE:
                criteria = criteria.lt(end);
                break;
        }
        return criteria;
    }
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Информационная система сбора и анализа данных об общеобразовательных учреждениях

Презентация

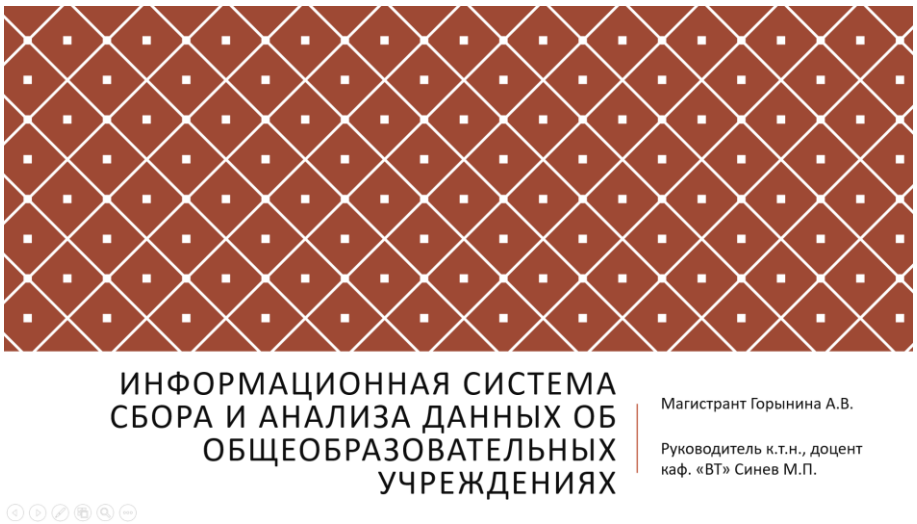

Рисунок Б.1 – Слайд 1

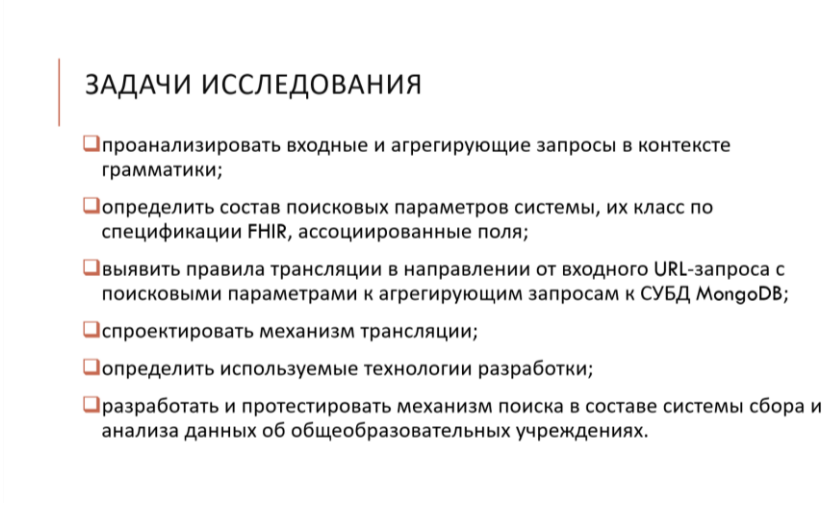

Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

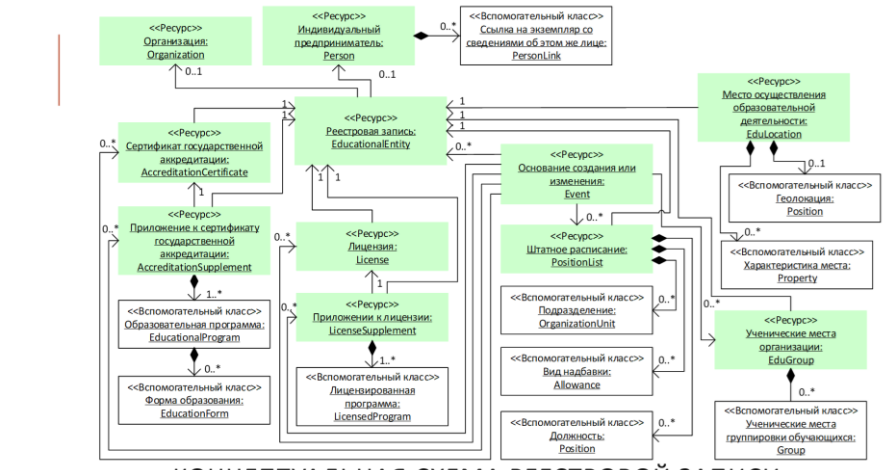

Рисунок Б.4 – Слайд 4

ПОИСК В РАМКАХ СПЕЦИФИКАЦИИ FHIR

[GET] baseurl/Resource{?name=value&...{&name=value}}

Параметры делятся на три категории: общие, специальные, параметры, определяющие результат поиска

Общие и специальные параметры могут быть четырех основных типов:

- String
- Token
- Reference
- Date/Datetime

тип в данном случает обозначает поведение сервера при обработке параметра

Рисунок Б.5 – Слайд 5

СПЕЦИАЛЬНЫЕ ПАРАМЕТРЫ ПОИСКА РЕСУРСА PERSON

Имя	Тип	Описание	Выражение	Тип поля документа
name	String	Поиск, определяемый сервером, при котором совпадение ищется с любым строковым элементом в объекте типа HumanName, включая фамилию (family), имя (give), отчество (middle) и/или текст (text)	Person.name.family | Person.name.give | Person.name.middle | Person.name.text	String
phone	Token	Номер телефона	Person.telecom.where(system= 'phone')	ContactPoint[]
telecom	Token	Поиск по значению любой контактной информации	Person.telecom	ContactPoint[]
link	Reference	Поиск по ссылке на другой экземпляр Person со сведениями об этом же лице	Person.link.target (Person)	BackboneElement[]. Reference
address-city	String	Город, указанный в адресе	Person.address.city	String

Рисунок Б.6 – Слайд 6

ЗАПРОСЫ С ТОЧКИ ЗРЕНИЯ ГРАММАТИКИ

Gu({a, b, c .. z, A, B, C .. Z, /, -, _, |, 0, 1, 2 .. 9, =, &, @}, {Query, Pair, Sort_value, Elements_value, Boolean_parameter, Parameter, Value, Uri_parameter, Time_parameter, Sort_parameter, Format_parameter, Elements_parameter, Page_parameter, String_parameter, Include_parameter, Include_value, Reference_parameter, Time_value, Format_value, Page_value, String_value, Boolean_value, Reference_value, Uri_value, Word, Symbol, Word16, Symbol16, Datetime, Timemodifier, Time, Date, Y, M, D, I60, I24, I1000, Number, Digit}, Pu, Query)

Gm({a, b, c .. z, A, B, C .. Z, /, \, -, _, |, 0, 1, 2 .. 9, $, “, ”, ‘{’, ‘}’, ^, ‘[’, ‘]’ , @}, {Query, Pipeline, Match_stage, Sort_stage, Skip_stage, Limit_stage, Sort_query, Sort_direction, Match_query, And_expr, And_addition, First_expression, Or_expr, Or_addition, Expression, String_expr, Address_expr, Name_expr, Exist_expr, Bool_expr, Date_expr, Date_operator, Token_expr, Tokenend_field, Backbone_array_expr, Array_expr, Reference_expr, Backbone_field, Token_field, Reference_field, Field, String_field, Bool_field, Boolean, Date_field, Word, Symbol, Word16, Symbol16, Datetime, Time, Date, Y, M, D, I60, I24, I1000, Number, Digit}, Pm, Query)

Рисунок Б.7 – Слайд 7

ПРАВИЛА ВЫВОДА. ЗАПРОС БЕЗ ПАРАМЕТРОВ

Условие	Запрос без параметров
Описание поведения сервера (правило)	Исполнение без ограничений вывода, но с постраничным выводом (по умолчанию – вывод десяти первых элементов)
Пример параметра	-
Запись примера запроса с использованием грамматики Gu	отсутствие параметров
Запись примера запроса с использованием грамматики Gm	{ "aggregate" : "__collection__", "pipeline" : [{ "$skip" : 0}, { "$limit" : 10}]}

Рисунок Б.8 – Слайд 8

ПРАВИЛА ВЫВОДА. ОПОРА НА ТИПЫ

Условие	Параметр типа Token ассоциирован с полем типа Boolean
Описание поведения сервера (правило)	Значения поля документа совпадает со значением параметра, регистронезависимый поиск.
Пример параметра	Organization.active
Запись примера запроса с использованием грамматики Gu	active=true
Запись примера запроса с использованием грамматики Gm	{"$match": { "active": true }}

Рисунок Б.9 – Слайд 9

ПРАВИЛА ВЫВОДА. ВЛОЖЕННОСТЬ И КРАТНОСТЬ

Условие	Параметр типа Token ассоциирован с полем типа CodeableConcept
Описание поведения сервера (правило)	Параметр обрабатывается как тип Token, валидны 4 формы записи и 4 варианта обработки, здесь приведена вторая форма записи
Пример параметра	Event.reason
Запись примера запроса с использованием грамматики Gu	reason=https://nsi.edu.gov.ru/CodeSystem/CodeSystem-OK018-24-1|01
Запись примера запроса с использованием грамматики Gm	{ "$match": { "reason.coding" : { "$elemMatch" : { "$and" : [{ "system" : "https://nsi.edu.gov.ru/CodeSystem/CodeSystem-OK018-24-1"}, { "code" : "01"}]}}}}

Рисунок Б.10 – Слайд 10

ПРАВИЛА ВЫВОДА. ЛОГИЧЕСКОЕ ИЛИ

Условие	Логическое ИЛИ нескольких значений одного параметра
Описание поведения сервера (правило)	Документ удовлетворяет условию, если значение поля соответствует хотя бы одному из приведенных значений параметра
Пример параметра	EduGroup.academyear
Запись примера запроса с использованием грамматики Gu	academyear=2014,2015
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$or" : [{ "academYear" : "2014"}, { "academYear" : "2015"}]}}

Рисунок Б.11 – Слайд 11

ПРАВИЛА ВЫВОДА. ЛОГИЧЕСКОЕ И

Условие	Логическое И нескольких параметров
Описание поведения сервера (правило)	Документ удовлетворяет условию, если документ соответствует всем приведенным параметрам
Пример параметра	EduGroup.academyear, EduGroup.version
Запись примера запроса с использованием грамматики Gu	academyear=2014,2015&version=v1
Запись примера запроса с использованием грамматики Gm	{ "$match" : { "$and" : [{ "$or" : [{ "academYear" : "2014"}, { "academYear" : "2015"}]}, { "version" : { "$regularExpression" : { "pattern" : "^\\Qv1\\E", "options" : "i"}}}]}}

Рисунок Б.12 – Слайд 12

ОРГАНИЗАЦИЯ БАЗЫ ЗНАНИЙ

База знаний включает справочники параметров и ассоциированных с ними полей:

- params – все специальные параметры системы и ассоциированные с ними поля,
- generalParams – общие для всех ресурсов параметры,
- tokenParams – параметры типа Token, ассоциированные со строковыми полями,
- compositeParams – параметры типа Token с предзаполненым системным,
- multiParams – параметры, которые подразумевают проверку нескольких полей, например, ассоциированные с полем типа HumanName.

Рисунок Б.13 – Слайд 13

АЛГОРИТМ РАБОТЫ МАШИНЫ ВЫВОДА

1 Извлечь параметры, модификаторы и префиксы

2 Отделить параметры, управляющие выводом

3 Выявить типы ассоциированных полей и разбить на категории

4 Применить параметры, управляющие выводом

- String или Enum,
- Decimal или иной числовой,
- LocalDateTime, LocalDate, ZonedDateTime,
- Boolean,
- Reference,
- Identifier, Coding, ContactPoint, CodeableConcept, BackboneElement.

Внутри категорий реализовывать обработку согласно правилам

Рисунок Б.14 – Слайд 14

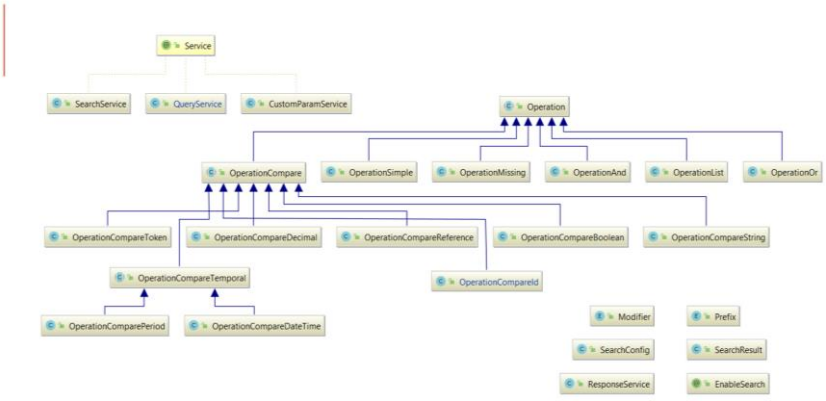

ДИАГРАММА КЛАССОВ

Рисунок Б.15 – Слайд 15

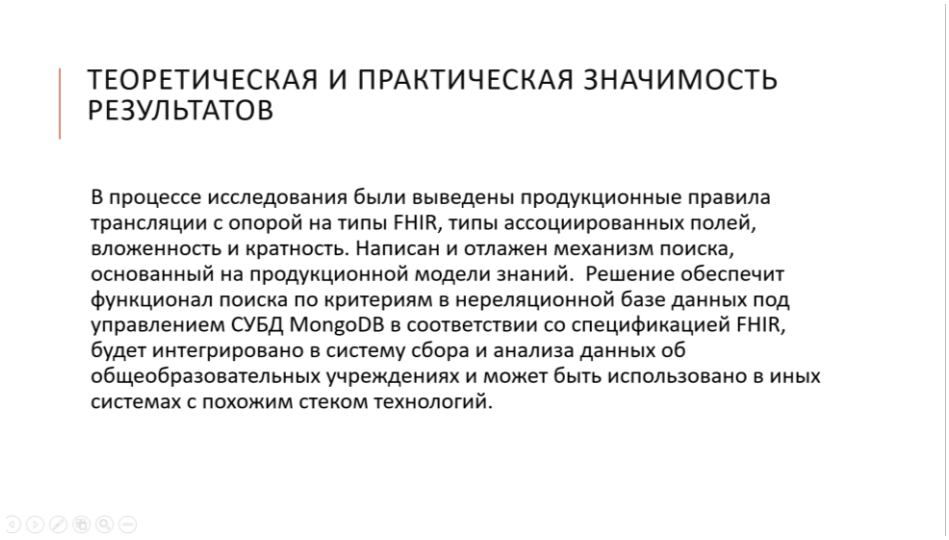

Рисунок Б.16 – Слайд 16

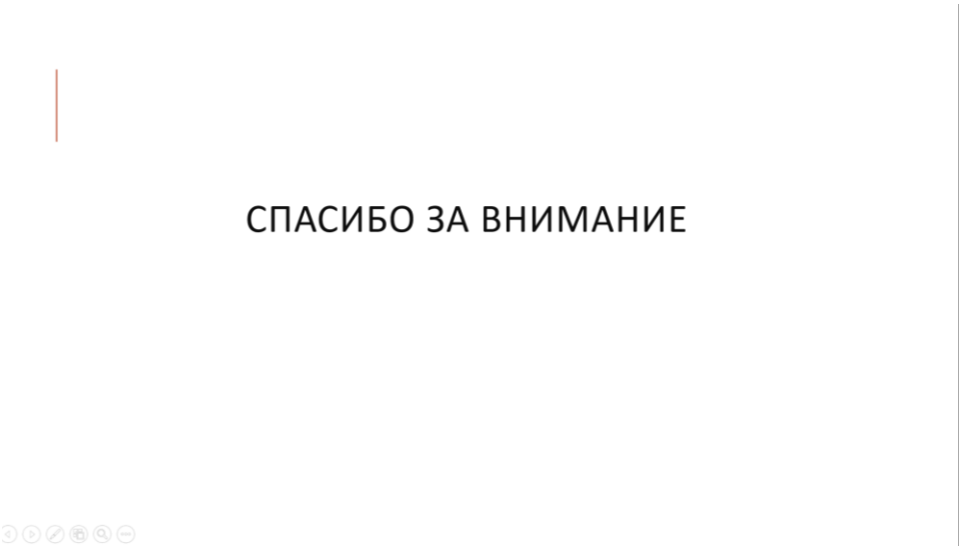

Рисунок Б.17 – Слайд 17