НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри

_____ Олександр КОВАЛЬ

«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп'ютерне геометричне моделювання процесів та систем”

спеціальності 122 “Комп’ютерні науки”

на тему: “ Управління представленнями гетерогенних даних ”

Виконав:
студент IV курсу, групи ТР-72
Ковшар Руслан Олегович _____

Керівник:
Доцент,
Стативка Юрій Іванович _____

Рецензент:
_____

_____ _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2021

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет: теплоенергетичний

Кафедра: автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти: перший рівень

Напрям підготовки: 122 Комп’ютерні науки

Спеціалізація: Комп'ютерне геометричне моделювання процесів та систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. В. Коваль
(підпис)
“ ” _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Ковшару Руслану Олеговичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи _Управління представленнями гетерогенних даних_____
_____
_____

керівник роботи__к.т.н., доцент Стативка Юрій Іванович_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24”05_ 2021_р. № 1267с

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи __набір показників діяльності наукової установи, показники рейтингових систем, мова програмування Java_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розглянути існуючі рейтингові системи, провести аналіз використаних в них показників, сформувати власну модель даних, створити рейтингову систему_____
_____
_____
_____
_____

_____
_____
_____

5. Перелік ілюстративного матеріалу
_Показники_використані_у_існуючих_системах._Архітектура_системи._Результати_тестування_та_приклади_використання_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "_10_"__жовтня_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	01.10.2020 р.	Виконано
2.	Вивчення та аналіз задачі	16.11.2020 р.	Виконано
3.	Розробка архітектури та загальної структури системи	20.02.2021 р.	Виконано
4.	Розробка структур окремих підсистем	12.04.2021 р.	Виконано
5.	Програмна реалізація системи	17.05.2021 р.	Виконано
6.	Оформлення пояснювальної записки	25.05.2021 р.	Виконано
7.	Захист програмного продукту	11.05.2021 р.	Виконано
8.	Передзахист	25.05.2021 р.	Виконано
9.	Захист	17.06.2021	Виконано

Студент _____ _____Ковшар Р.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Стативка Ю.І._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Записка містить 53 сторінки, 35 рисунків, 3 таблиці, 3 додатки та 7 посилань.

Мета роботи: створення системи для керування представленнями гетерогенних даних, а саме системи для побудови рейтингів, як різновиду таких систем.

Робота складається з п’яти розділів.

У першому розділі формується постановка задачі та наводиться інформацію щодо використаних у системі індикаторів.

У другому розділі оглядаються існуючі системи, проводиться їх аналіз і доводиться необхідність розробки системи.

У третьому розділі наводиться архітектура майбутньої системи, з вказанням технологічного стеку для розробки, функціональних можливостей системи та описом основних класів та архітектури в якій вони перебувають.

У четвертому розділі описується програмна реалізація системи. Вказується на розроблені модулі системи, проводиться їх опис та зазначає функції за які вони відповідають.

У п’ятому розділі демонструються приклади з використання системи, наводиться алгоритм взаємодії користувача з системою, і водночас є інструкцією з використання системи.

ABSTRACT

The note contains 53 page, 35 pictures, 3 tables, 3 appendices and 7 links.

Purpose of the work is to create a system for managing the representation of heterogeneous data, namely a system for building ratings as a kind of such systems.

The work consists of five sections.

In the first section the problem statement is formed and information on the indicators used in the system is given.

The second section examines existing systems, analyzes them and proves the need to develop a system.

The third section provides the architecture of the future system, indicating the technological stack for development, system functionality and a description of the main classes and architecture in which they are located.

The fourth section describes the software implementation of the system. Indicates the developed modules of the system, describes them and indicates the functions for which they are responsible.

The fifth section shows examples of how to use the system, provides an algorithm for user interaction with the system, and at the same time provides instructions for using the system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ 8
ВСТУП 9
1. ЗАДАЧА УПРАВЛІННЯ ПРЕДСТАВЛЕННЯМИ ГЕТЕРОГЕННИХ ДАНИХ 11
1.1. Постановка задачі 11
1.2. Індикатори 12
1.2.1. Поняття індикатора 12
1.2.2. Перелік індикаторів 12
2. СИСТЕМИ ОЦІНКИ РІВНЯ НАУКОВИХ УСТАНОВ 16
2.1. Scimago Institusion Rankings 16
2.2. Leiden Ranking 18
2.3. Shanghai Ranking 19
3. АРХІТЕКТУРА СИСТЕМИ 21
3.1. Технічний стек системи 21
3.2. Базова архітектура системи 22
3.3. Функції системи 23
3.4. Клас індикатора 25
3.5. Архітектура індикатора 28
3.6. Семантика полів індикатора 29
4. ПРОГРАМНА РЕАЛІЗАЦІЯ 31
4.1. Модуль агрегації 31
4.1.1. Керування індикаторами 32
4.1.2. Управління представленням індикаторів 32
4.1.3. Управління файловим представленням індикаторів 33
4.1.4. Отримання переліку полів індикаторів 33
4.1.5. Керування семантикою полів індикатора 34
4.2. Модуль рейтингів 35
4.2.1. Формування системного рейтингу 36
4.2.2. Формування користувацького рейтингу 38
4.3. Адміністративний модуль 38
5. ПРИКЛАД ВИКОРИСТАННЯ СИСТЕМИ 40
5.1. Swagger 40
5.2. Тестування системи 43

5.2.1. Створення індикатора 44
5.2.2. Представлення індикатора 44
5.2.3. Отримання переліку полів індикатора 45
5.2.4. Файлове представлення індикатора 46
5.2.5. Створення обмежень для полів 48
5.2.6. Отримання рейтингу 48
5.2.7. Адміністративні функції 50
5.3. Інсталювання системи 51
ВИСНОВКИ 52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53
ДОДАТОК А 54
ДОДАТОК Б 56
ДОДАТОК В 65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

Гетерогенні дані – різнорідні дані, дані що мають різну структуру

Індикатор – показник діяльності, науково-технічної установи та певним напрямком

MVC – Module – View - Controller

REST - Representational State Transfer

Ендпоінт – точка входу до REST систем

SOLID - 5 основних принципів об'єктно-орієнтованого програмування і проектування

JSON - JavaScript Object Notation

CSV - Comma-Separated Values

ВСТУП

Діяльність науково-технічних установ вимагають точної оцінки результатів своєї діяльності. Така оцінка може стати як маркером успішної роботи та розвитку установи, так і свідчити про те, що певний заклад перебуває у кризовому стані і потребує реформування.

Оцінка рівня розвитку має бути чіткою і прозорою, давати однозначну відповідь про теперішній стан інституцій. Для такої оцінки використовують системи індикаторів, тобто маркерів що показують стан конкретного напрямку діяльності установи. Такими індикаторами можуть бути: кількість міжнародних публікацій, індекс впливовості, індекс цитування та інші. Поєднання і аналіз таких даних можуть надати над певний рейтинг установ.

Побудова таких систем – це нетривіальна задача, яка містить у собі: збір даних, їх аналіз, та представлення. На даний момент існують немало сервісів, що виконують таку задачу, наприклад Scimago Institusion Rankings [1], Leiden Ranking [2], Shanghai Ranking [3] тощо. Кожна з наведених вище систем являє собою рейтинг інституцій, що створений на основі власних алгоритмів або ж з використанням лише певного набору індикаторів, що, у власну чергу не дає змогу оцінити діяльність установи по іншому набору.

Такий підхід не дає можливості однозначно оцінити діяльність закладів за ознаками, які є актуальними для користувача у даний момент. Тому і виникає потреба у розробці системи, яка б дозволяла обирати актуальний для користувача набір індикаторів та обчислювати її рівень за алгоритмами, що відповідають стану та політиці розвитку множини установ у масштабах університету, галузі чи глобальному.

Мета роботи полягає у розробці засобів управління гетерогенними даними для потреб оцінки рівня діяльності наукової установи.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

1. розглянути приклади побудови популярних системи оцінки діяльності наукових інституцій;

2. визначити склад та характеристики потенційно необхідних даних для побудови моделі даних;
3. сформулювати функціональні вимоги та обмеження програмної реалізації з врахуванням динамічності складу та неповноти даних, варіативності алгоритмів обчислення;
4. розробити архітектуру та програмний прототип системи;
5. протестувати роботу прототипу системи.

Об’єктом роботи є управління представленнями гетерогенних даних.

Предмет роботи - засоби управління і представлення різнорідних та неповних даних з неуніфікованою структурою.

1. ЗАДАЧА УПРАВЛІННЯ ПРЕДСТАВЛЕННЯМИ ГЕТЕРОГЕННИХ ДАНИХ

Метою дипломної роботи є створення системи для управління представленнями гетерогенних даних. А саме систему для побудови рейтингів, оскільки такі системи найкраще відповідають визначенню гетерогенних даних. Рейтинг є досить зручним інструментом для оцінки діяльності науково-технічних установ за певними напрямками, як в порівняннями з іншими системами так і у розрізі розвитку однієї системи. Для формування рейтингів використовуються індикатори, тобто показники діяльності закладу у певному напрямку діяльності. Існує багато готових рішень, тобто систем-рейтингів, наприклад Scimago Institusion Rankings, Leiden Ranking, Shanghai Ranking тощо. Але усі ці системи формують рейтинги за своїм алгоритмом і не дозволяють користувачеві сформувати рейтинг за альтернативним алгоритмом.

Першочерговим завданням роботи було дослідження існуючих систем та програмних засобів, що можуть забезпечити динамічну вибірку даних. У результаті дослідження, було знайдено відповідний засіб, мову що дозволяє створювати запити та виконувати їх з існуючими даними, розробка компанії Facebook – GraphQL [4].

1.1. Постановка задачі

Створення системи включає в себе наступні задачі:

- Вивчення характеристик даних, що використовуються у існуючих рейтингових системах.
- Формування власної моделі даних
- Створення засобу для керування даними
- Створення засобу для динамічної вибірки даних
- Створення засобу для побудови рейтингів

1.2. Індикатори

Центральним поняттям системи, що може дати чітку відповідь що до розвитку установи, є поняття індикатора. Показники можуть бути обрані як уже з існуючого переліку так і бути побудовані, відповідно до накладених на них умов.

1.2.1. Поняття індикатора

Системи індикаторів, тобто показників рівня розвитку установи чи її підрозділу у певній сфері діяльності, можуть бути використані для оцінки рівня розвитку інституції в цілому. Здебільшого індикатори використовуються для сприяння позиціонування суб'єкта наукової діяльності, для його самоідентифікації, виявлення міжнародних взаємозв'язків, оцінки нематеріальних активів, а також для прийняття стратегічних рішень що до подальшого розвитку. Побудова показників відбувається в результаті аналізу даних про науково-технічну та інноваційну діяльність.

У конкретних випадках конструювання індикаторів виконується для забезпечення однозначного представлення діяльності установи та/чи її підрозділів у межах певної концептуальної моделі, тобто характеристик якими можна описати цю діяльність.

Оскільки різні установи можуть мати різні моделі оцінки або ж можуть відрізнятися від поточного плану розвитку або статусу закладу, то показників мають відповідати наступним умовам:

1. Показник має наводити явний опис або надати посилання на поняття, що він характеризує.
2. Показник має наводити перелік видів діяльності, що орієнтуються на нього.
3. Показник має бути доцільним з огляду на якість, та доступність даних.
4. Показник має наводити зрозумілий сенс та обмеження.

1.2.2. Перелік індикаторів

Наведений перелік індикаторів повністю відповідає наведеним вимогам (пункт 1.2.1) та подібний до переліку регламентованого Європейською науковою фундацією:

1) Обсяги зовнішнього фінансування
   - Вказує на: можливість залучення фінансування з недержавних джерел.

- Значення: Обсяги; Перевищення над рівнем державного фінансування; загальний бюджет організації.
- Розподіл за: галузями науки; країною; роками.
- Абревіатура: ПЗФ

2) Ефективність закордонного фінансування
   - Вказує на: ефективність залученого іноземного фінансування
   - Значення: Кількість публікацій без іноземних співавторів за проектами з іноземним фінансуванням.
   - Розподіл за: галузями науки; країною; типом/назвою фінансуючого суб'єкта, роками.
   - Абревіатура: ПЕФ

3) Міжнародне співавторство
   - Вказує на: рівень міжнародного наукового співробітництва
   - Значення: Частка робіт з міжнародними співавторами
   - Розподіл за: галузями науки; предметними областями; країною; роками.
   - Абревіатура: ПМС

4) Залучення закордонних науковців
   - Вказує на: рівень привабливості установи та на здатність залучати різноманітні/свіжі сили до міжнародного наукового співробітництва
   - Значення: Частка залучених закордонних дослідників від загального числа дослідників.
   - Розподіл за: типом залучення (постійні/непостійні); галузями науки; країною, звідки були залучені; роками.
   - Абревіатура: ПЗН

5) Міжнародна мобільність

- Вказує на здатність до інтеграції з міжнародним науковим середовищем та обсяги додаткових ресурсів.
- Значення: Кількість закордонних дослідників, які приїхали до організації; Їх частка до загальної кількості дослідників в організації; Кількість дослідників організації, які перейшли до закордонної чи міжнародної організації; Їх частка до загальної кількості дослідників в організації; Динаміка потоку — частка притік/відтік.
- Розподіл за: галузями науки; країною прибуття/відбуття; типом мобільності (два тижні-три місяці, три місяці і більше); науковим званням/ступенем; роками.
- Абревіатура: ПММ

6) Бюджет спільних наукових програм або проектів
   - Вказує на здатність до інтеграції у міжнародні програми чи проекти з розподілом функцій серед учасників з різних країн.
   - Значення: Обсяги фінансування спільних міжнародних проектів/програм; Загальний бюджет організації; Обсяги державного фінансування організації.
   - Розподіл за: галузями науки; роками.
   - Абревіатура: ПСП

7) Міжнародне використання дослідницької інфраструктури організації
   - Вказує на наявність та масштаб дослідницької інфраструктури, яка становить міжнародний інтерес.
   - Значення: Кількість іноземних користувачів інфраструктури; Кількість днів використання іноземними дослідниками.
   - Розподіл за: сферою досліджень; країнами, дослідники яких використовували інфраструктуру; типами об’єктів інфраструктури; роками.
   - Абревіатура: ПДІ

8) Участь у закордонних структурах залучення дослідників
   - Вказує на інтеграцію до міжнародного процесу залучення наукових працівників.
   - Значення: Частка кількості працівників організації, які працюють за кордоном від загальної кількості працівників організації, зайнятих підбором персоналу.
   - Розподіл за: науковим статусом/ступенем працівника; галузями науки; країною; роками.
   - Абревіатура: ПМЗ

9) Іноземні фахівці у оцінці наукових досліджень
   - Вказує на досвід та сприйняття організацією закордонних експертів з метою посилення міжнародної актуальності досліджень
   - Значення: Частка кількості іноземних експертів від загальної кількості учасників, залучених до оцінки наукових досліджень
   - Розподіл за: рівнем оцінювання (проектна група, підрозділ, організація); галузями науки; країною установи, в якій працюють експерти; роками.
   - Абревіатура: ПМО

2. СИСТЕМИ ОЦІНКИ РІВНЯ НАУКОВИХ УСТАНОВ

Для перегляду та аналізу існуючих індикаторів науково-технічних установ існує вже чимало систем. Здебільшого такі системи створюють рейтинги установ за певним набором індикаторів. Такі системи використовують певні алгоритми та фіксований, обраний розробниками набір індикаторів. Ці системи представляють результат розрахунку, а не інструмент для аналізу та планування розвитку наукової інституції. Розглянемо декілька прикладів таких систем.

2.1. Scimago Institusion Rankings

Першою системою є Scimago Institusion Rankings. Дана система дозволяє отримати рейтинги компаній, урядових установ, установ системи охорони здоров'я, університетів тощо.

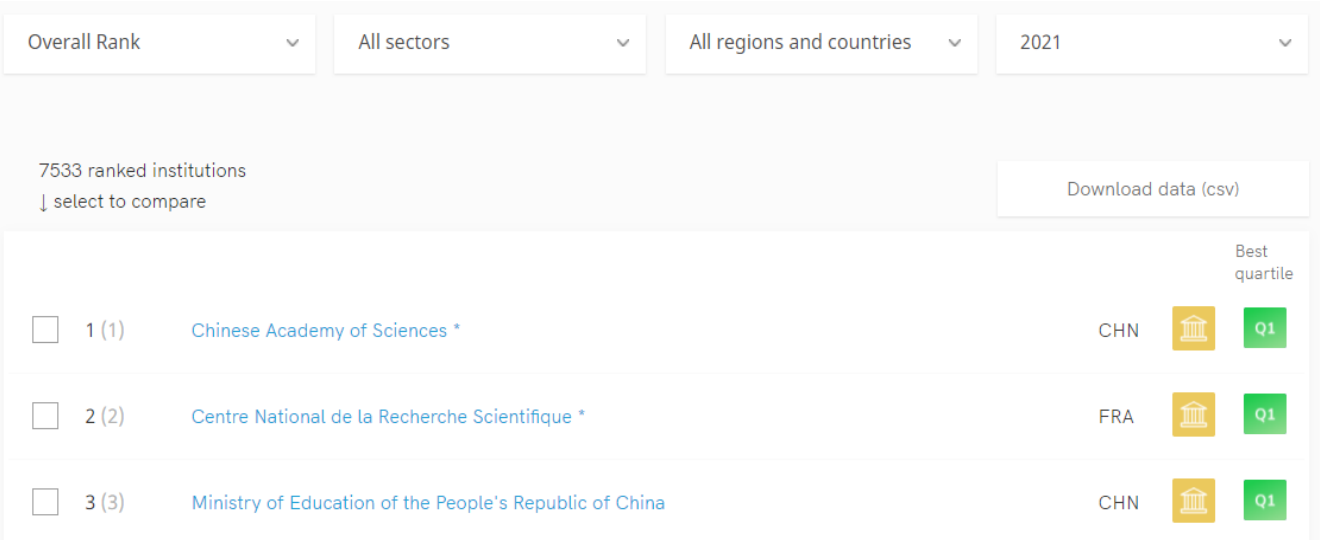

Рисунок 2.1 – інтерфейс Scimago Institusion Rankings

Рейтинг інституцій Scimago (SIR) - це рейтинг науково-технічних установ. Він поєднує у собі три різні набори показників:

- показники засновані на результатах наукових досліджень,
- показники засновані на результатах інновацій
- показники засновані на соціальному впливі, що вимірюється їхньою

видимістю в Інтернеті.

Показники Scimago поділяються на три групи:

- призначені для відображення наукових характеристик
- призначені для економічних характеристик
- призначені для соціальних характеристик

Таблиця 2.1 - індикатори, що включає в себе SIR

Критерій	Індикатор	Код	Вага, %
Наукові дослідження	Нормалізований вплив	NI	13
	Досконаліть з керівництвом	EwL	8
	Вихід	O	8
	Наукове керівництво	L	5
	Не власні журнали	NotOJ	3
	Власні журнали	OB	3
	Досконаліть	Exc	2
	Високоякісні публікації	Q1	2
	Міжнародне співробітництво	IC	2
	Відкритий доступ	OA	2
	Пул наукових талантів	STP	2
Інновації	Інноваційні знання	IK	10
	Патенти	PT	10
	Технологічний вплив	TI	10
Соціальний вплив	Альтметрика	AM	10
	Вхідні посилання	BN	5
	Веб-розмір	WS	5

SIR включає в себе як кількісні, так і не кількісні показники; тобто на показники можуть впливати і не впливати розміри установ. Виходячи з чого, можна прийти до висновку, що SIR надає загальний рейтинг з тієї чи іншої сфери діяльності установ.

2.2. Leiden Ranking

Наступною системою є Leiden Ranking. Leiden Ranking пропонує складний набір індикаторів, що забезпечують статистику на рівні університетів щодо:

- наукового впливу
- співпраці
- публікації у відкритому доступі.

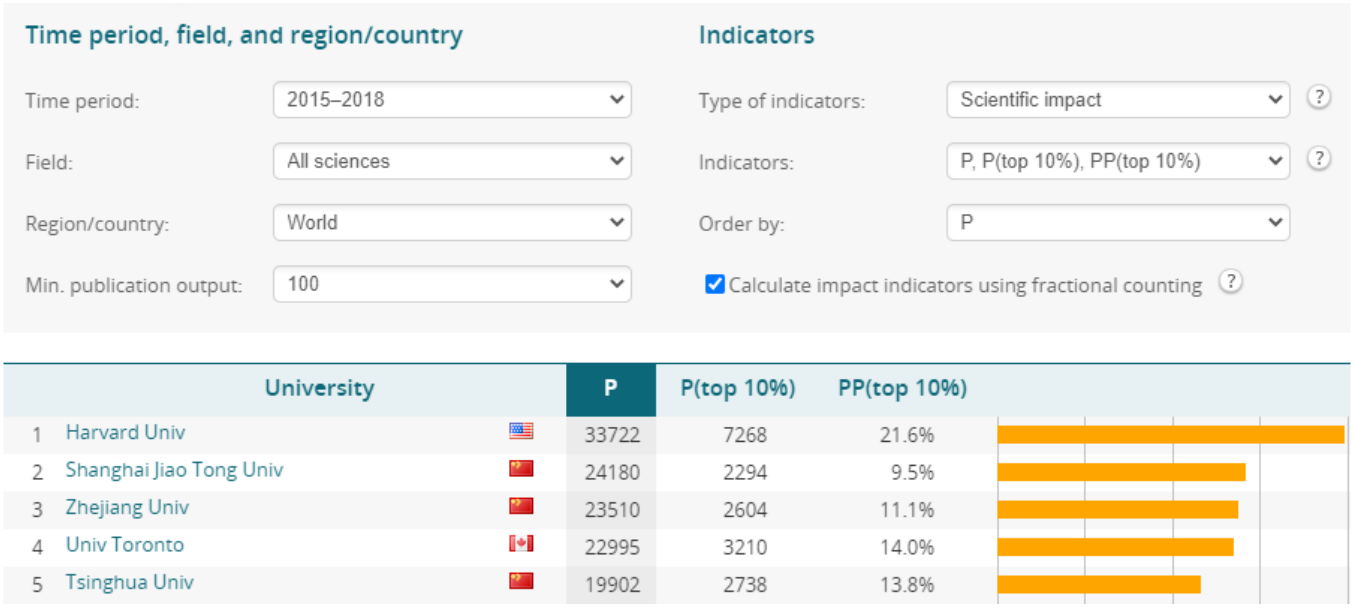

Рисунок 2.2 – інтерфейс Leiden Ranking

Рейтинг Лейдена включає в себе два типи показників:

- кількісний
- не кількісний

Кількісні показники будуються шляхом підрахунку абсолютної кількості публікацій. Не кількісні отримуються шляхом обчислення частки публікацій університету з певною властивістю.

Виходячи з цього, можна прийти до висновку, що коли використовуються не кількісні індикатори, то університети можуть мати досить високі результати незалежно від свого розміру. У випадку з кількісними показниками, університети з більшим обсягом публікацій, як правило, мають кращі результати, ніж університети з меншим обсягом публікацій.

2.3. Shanghai Ranking

Останньою системою в огляді є Shanghai Ranking. Дана система, як і оглянуті у пунктах 2.1 та 2.2, представляє собою рейтинг інституцій.

World Rank	Institution*	By location: All	National/Regional Rank	Total Score	Score on: Alumni
1	Harvard University		1	100.0	100.0
2	Stanford University		2	74.2	43.8
3	University of Cambridge		1	70.6	79.5
4	Massachusetts Institute of Technology (MIT)		3	69.6	71.4
5	University of California, Berkeley		4	65.8	64.9
6	Princeton University		5	61.1	59.0
7	Columbia University		6	58.6	59.5

Рисунок 2.3 – інтерфейс Shanghai Ranking

Shanghai Ranking класифікує науково-технічні установи за багатьма індикаторами наукової діяльності. Серед них можуть бути: випускники та співробітники, які отримують міжнародні премії, високо цитовані співробітники, проіндексовані за індексом цитування статті, та показники академічної успішності на душу населення.

Установа, що набрала найвищий бал за певним показником, отримує бал 100, інші установи отримують бал, розрахований як відсоток від найбільшої оцінки. Коригування показників відбувається за допомогою статистичних методів за допомогою певних ваг. Приклад показників та їх ваг наведений у таблиці 1.2.

Таблиця 1.2 - індикатори та ваги, що включає в себе Shanghai Ranking

Критерій	Індикатор	Код	Вага, %
Якість освіти	Випускники установи, що отримують міжнародні премії	Alumni	10
Якість факультету	Співробітники установи, що отримують міжнародні премії	Award	20
	Високо цитовані дослідники	HiCi	20
Результат дослідження	Доклади, опубліковані в “Nature and Science”	N&S	20
	Доклади, проіндексовані в Індексі наукових посилань та Індексі цитування соціальних наук	PUB	20
Ефективність на душу населення	Навчання на душу населення в установі	PCP	10

3. АРХІТЕКТУРА СИСТЕМИ

Першим кроком у розробці будь-якої інформаційної системи є її проектування та створення майбутньої архітектури проекту. Але ще до початку етапу проектування йде етап дослідження та вибору технологій завдяки яким буде побудована система. Технічний стек був обраний спираючись на: поставлене завдання, аналіз існуючих рішень та досвід у створенні та підтримці інформаційних систем.

3.1. Технічний стек системи

У якості основного інструменту у розробці було обрано мову програмування Java. На прийняття такого висновку посприяли такі плюси мови як:

- кросплатформенність (майбутня система не буде залежати від операційної системи)
- легкість інсталяцій та підтримки
- можливість використання дешевих серверів
- велика спільнота
- особистий досвід роботи

Для пришвидшення часу розробки та отримання вже готового базового функціоналу системи було прийнято рішення використати розробку компанії Pivotal сімейство фреймворків створених на мові програмування Java - Spring Framework.

Наступним кроком стало обирання сховища для даних. Основна задача системи вимагає динамічної схеми даних, тобто дані можуть дещо змінюватися від користувача до користувача. Для вирішення таких проблем найкраще підходить нереляційний підхід, тому у якості бази даних остаточний вибір пав на MongoDB.

Також для вирішення поставленої задачі необхідно визначитись з тим чи створювати своє рішення для реалізації динамічної вибірки даних чи використати вже існуючий інструмент. У результаті досліджень та тестів, було прийняте рішення

використати розробку компанії Facebook GraphQL. По-перше, інструмент безкоштовний отже його використання ніяким чином не збільшить ціну на продукт, по-друге, використання існуючого засобу дозволить значно зменшити час на розробку та тестування продукту.

Отже, у результаті маємо наступний технічний стек системи:

- мова програмування – Java [5]
- фреймворк – Spring [6]
- база даних – MongoDB [7]
- додаткова бібліотека - GraphQL

3.2. Базова архітектура системи

У основу архітектури системи було покладено принцип мікросервісності. Даний підхід передбачає використання максимально дрібних, мало зв’язних між собою, і легкозмінюваних модулів, тобто сервісів.

Такий підхід дозволить проводити розробку та тестування модулів системи незалежно один від одного, проводити легку інсталяцію продукту, легко виконувати розширення системи та додання нового функціоналу та замінювати модулі системи.

Основним модулем системи є агрегаційний модуль. Він буде виконувати наступні функції:

- керування індикаторами
- керування семантикою полів індикаторів
- отримання переліку використаних полів індикаторів
- динамічна вибірка даних індикаторів
- вивантаження даних про індикатори у файл

Наступним модулем системи є модуль побудови рейтингу. Він має функцію побудови рейтингу. Рішення про винесення функції формування рейтингу у окремий модуль було прийнято за для забезпечення можливого подальшого розвитку та

розширення системи.

Ще один модуль, модуль адміністратора. Модуль виконує функцію моніторингу усієї системи, тобто активні модулі, яке навантаження мають, скільки ресурсів використано, тощо.

3.3. Функції системи

Виходячи із поставленої мети та завдань роботи, та проаналізувавши існуючі аналоги систем, що використовуються для обробки індикаторів та побудови рейтингів, було прийняте рішення, що розроблена система буде мати наступні функції:

- керування індикаторами (створення, перегляд, редагування та видалення індикаторів)
- керування семантикою полів індикаторів (створює обмеження для даних, що є полем індикатора)
- отримання переліку використаних полів індикаторів (демонструє список усіх полів, що містять у собі індикатори)
- динамічна вибірка даних індикаторів (дає змогу отримувати лише необхідні користувачеві поля)
- вивантаження даних про індикатори у файл (створення представлення індикаторів у зручному файловому представленні)

Усі зазначені вище функції відображені на діаграмі прецедентів системи (рис 3.1). Також система має ще одну функцію, функцію моніторингу. Дану функцію я не включив до діаграми прецедентів, оскільки вона не стосується предметної області. Моніторинг дозволить користувачеві дізнатися поточний стан системи, а саме: використані ресурси, навантаження, підняті модулі тощо.

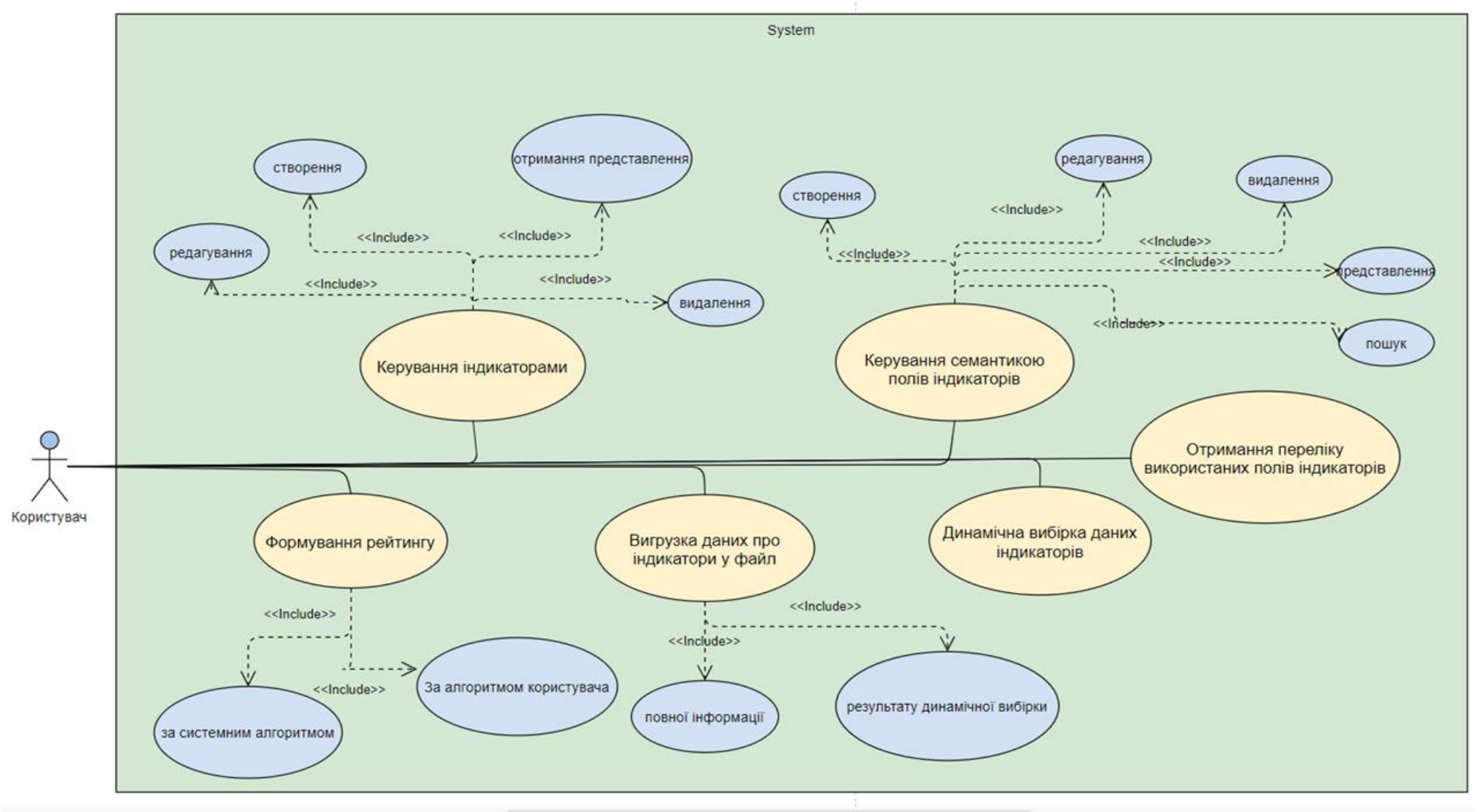

Рисунок 3.1 – UseCase діаграма систем

3.4. Клас індикатора

Оскільки розроблена система працює з індикаторами, та має більшість функцій пов’язаними с ними, то можна прийти до висновку що сутність індикатора, є центральною сутністю системи. Отже, для коректного відображення бізнес логіки необхідне правильне її представлення у системі.

За для виконання цієї задачі було розроблено клас індикатор. Що включає в себе 59 полів, що є відображенням певних характеристик у напрямках діяльності науково-технічної установи та додаткового засобу для додання власних полів, тобто власних, необхідних користувачеві характеристик.

Поля, що є запропоновані для використання, було обрано та використано згідно до міжнародних стандартів та оглянутих у розділі 2 існуючих системах.

Оскільки система передбачає супровід певної установи під час її діяльності та розвитку, то необхідно забезпечити користувача засобом для оперативного додання нових показників що до діяльності системи у нових напрямках для оперативної оцінки. Саме тому необхідно забезпечити користувача засобом для попереднього створення власних полів індикатора. Засіб повинен давати можливість створювати показники будь-якої складності та можливість змінювати їх у будь-який момент.

Усі наведені вище характеристики класу індикатора наведені UML-діаграмі класу індикатора (рис. 3.2-3.3)

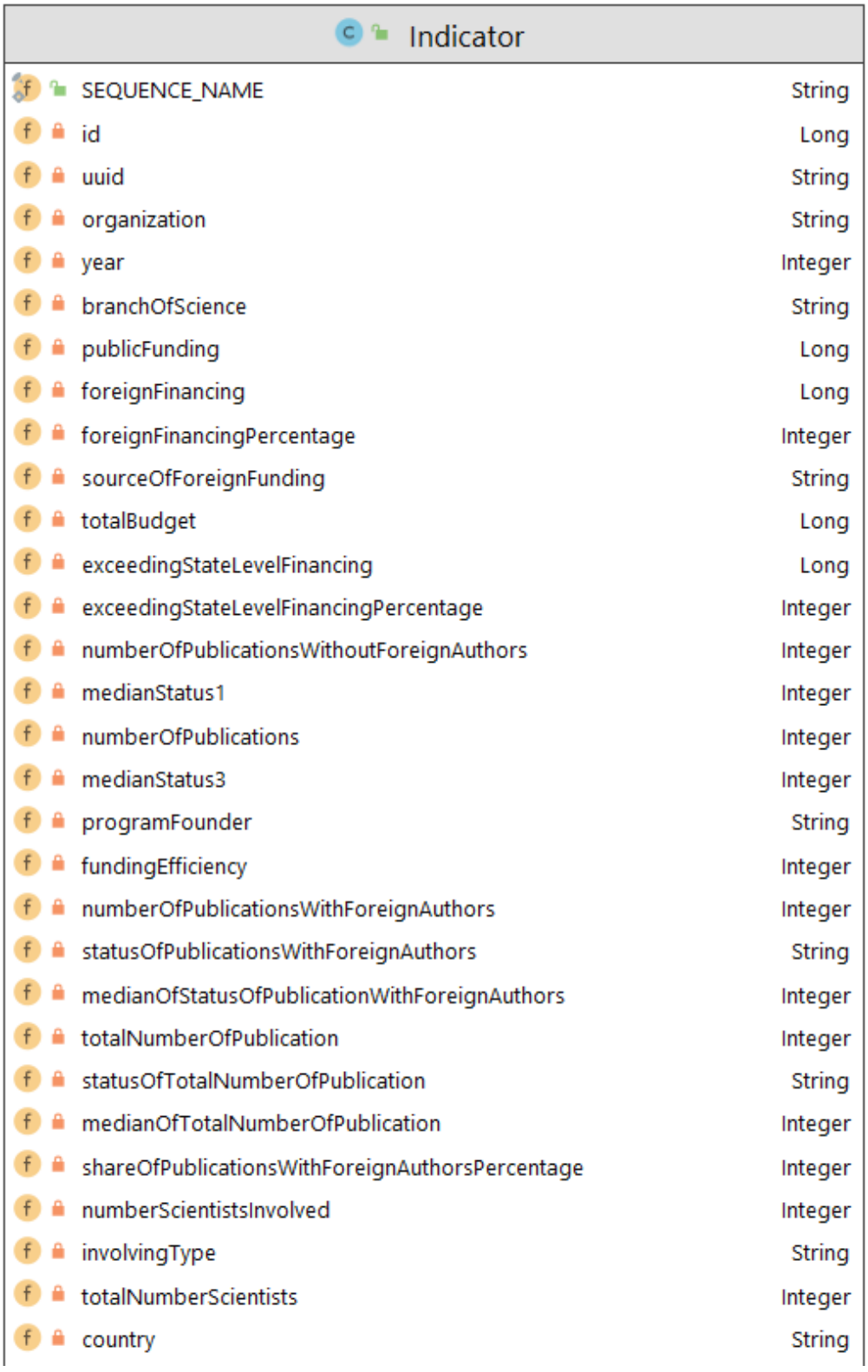

Рисунок 3.2 – Клас Indicator

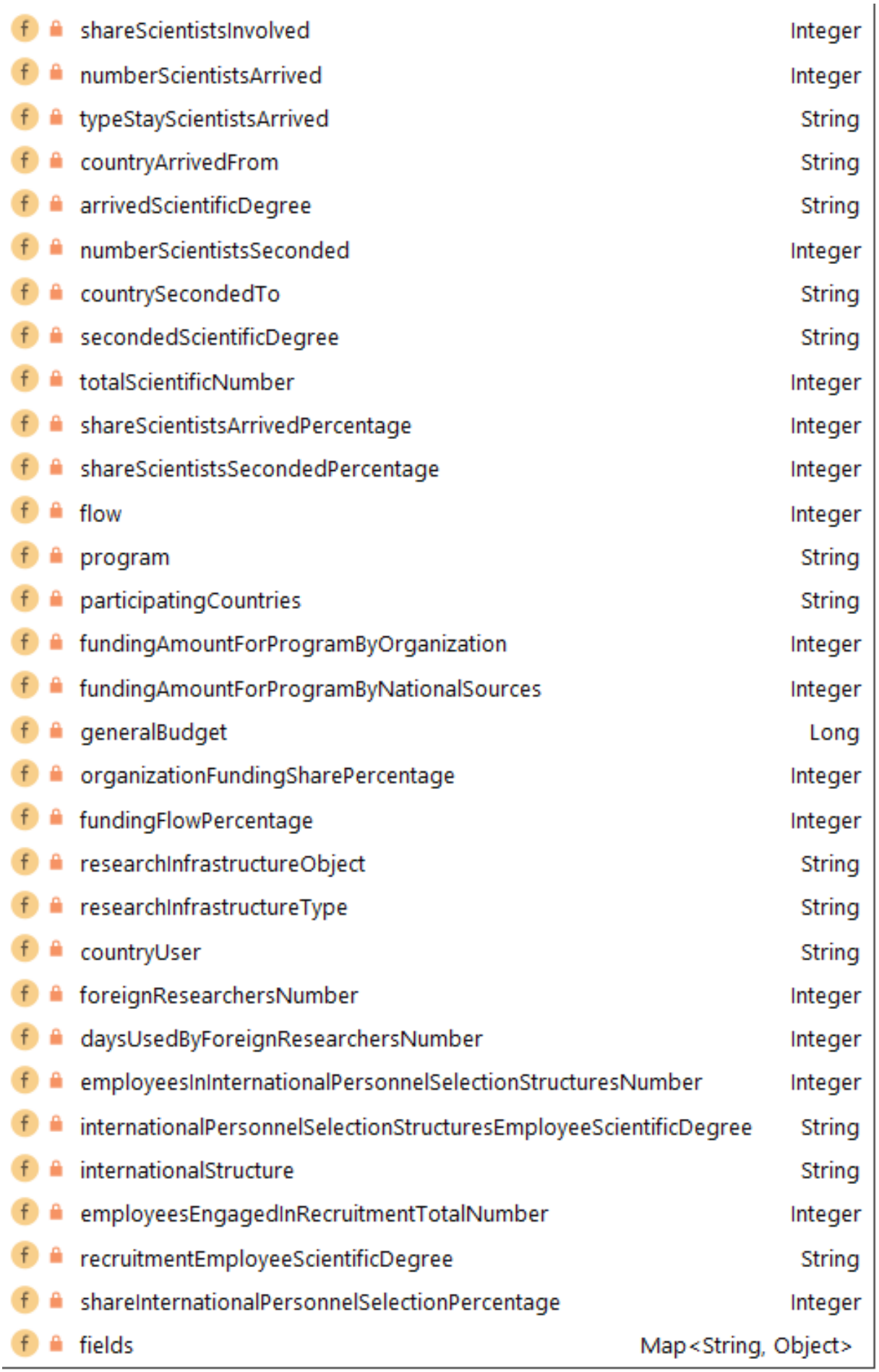

Рисунок 3.3 – Клас Indicator

Наведена діаграма демонструє перелік полів індикатора та засіб для створення користувацьких полів. Окрім полів клас має і класичні методи доступу до даних полів, але вони не були наведені.

Засобом для оперування користувацькими полями стала структура даних Map. Дана структура передбачає збереження даних у вигляді ключ - значення. Ключом є строка, тобто назва доданого поля, а значенням може бути будь-який об’єкт. Така структура дозволить будувати будь-яку необхідну користувачеві схему даних.

3.5. Архітектура індикатора

Власне сам клас індикатора, що описаний у пункті 3.4, перебуває у наступній архітектурі системи (рис. 3.4).

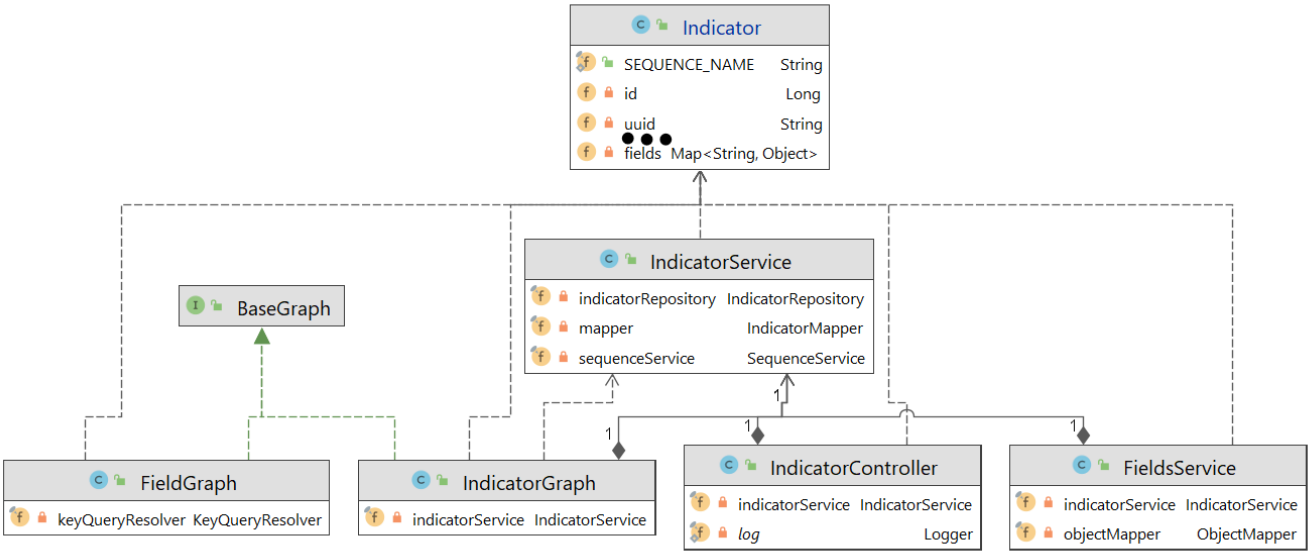

Рисунок 3.4 – UML діаграма класів компонента для обробки індикаторів

Зображена архітектура відповідає підходу MVC (Model – View - Controller). Тобто, для виконання дій з індикатором є контролер, що приймає команди, у даному конкретному випадку це http запити, оброблює їх та передає до рівня моделі. На цьому рівні діє сервіс, що виконує усі необхідні дії з індикатором. Сам сервіс має у собі класи помічники, що беруть на себе окремі функції. Наприклад, класи що виконують динамічну вибірку даних, або ж додатковий сервіс, що отримує усі поля з

індикатора. Таке роздроблення системи відповідає принципам SOLID.

Класи реалізують інтерфейси, що забезпечують легку та швидку модифікацію системи, кожен клас виконує лише одну поставлену задачу, що пришвидшує тестування та супровід системи, компоненти є легкозамінюваними.

3.6. Семантика полів індикатора

Оскільки розроблена система забезпечує користувача засобом для створення власних характеристик для індикаторів, то необхідно також забезпечити користувача засобом для обмежень або верифікації даних, що можуть знаходитися усередині цих полів.

З цією метою було спроектовано наступну схему (рис. 3.5).

Ця схема відображує структуру, за допомогою якої буде надаватись семантика для полів індикатора. Для кожного поля буде виділено окремий клас FieldMetadata, що включає в себе id поля котре він характеризує, тип даних, що це поле містить, та список фільтрів, що працюють за вказаним оператором (AND, OR). Фільтр являє собою клас, що складається з оператора порівняння та значення з яким буде порівняне значення поля.

Маючи дану структуру, користувач може легко обмежувати та валідувати дані, що містяться у певному полі.

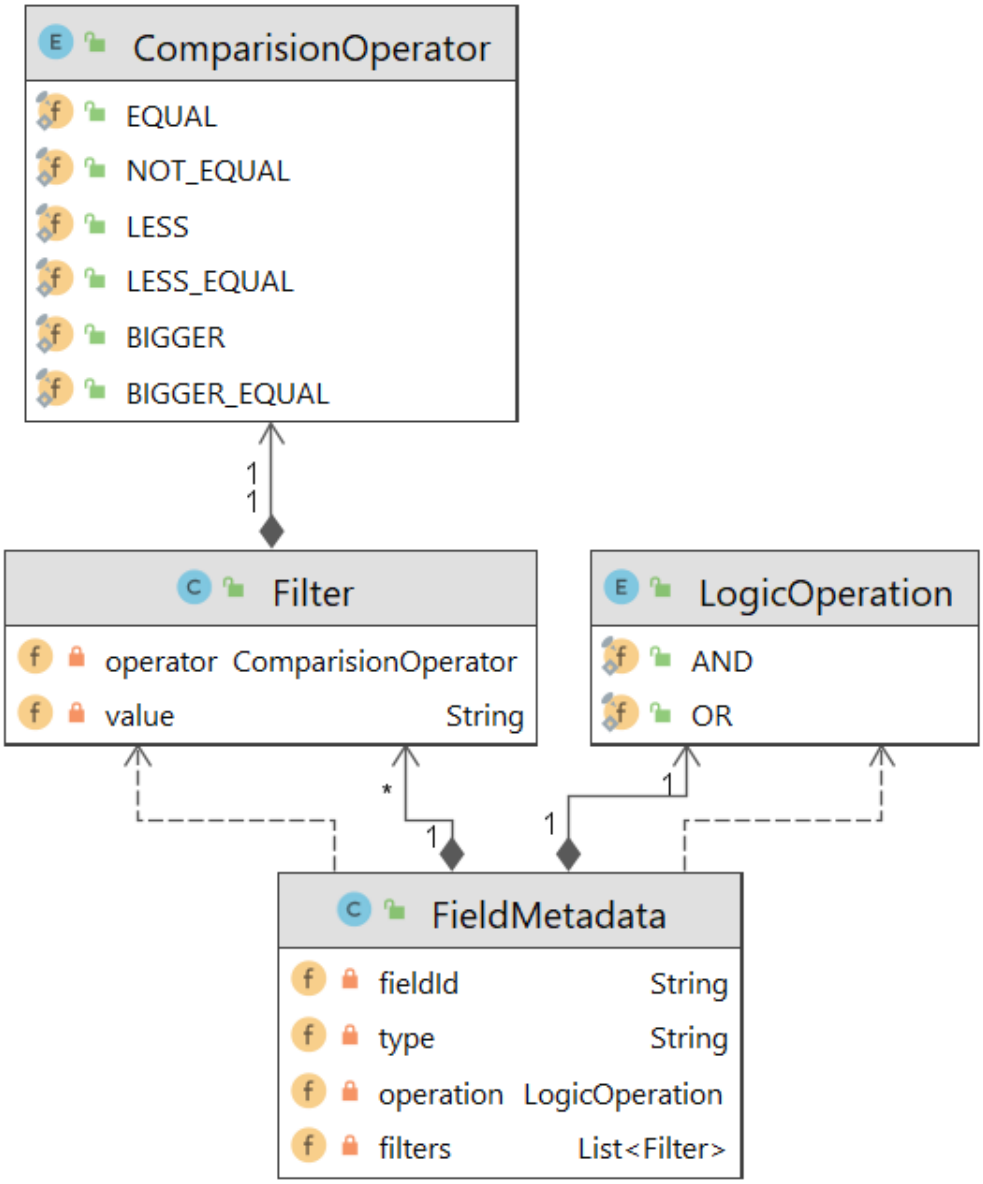

Рисунок 3.5 – UML діаграма класів для обмеження полів

4. ПРОГРАМНА РЕАЛІЗАЦІЯ

Під час основного етапу розробки, вся розробка велась спираючись на розроблену архітектуру описану у розділі 3.

4.1. Модуль агрегації

Даний модуль виконує функціональність зазначену у пункті 3.2, та має наступну пакетну схему (рис 4.1).

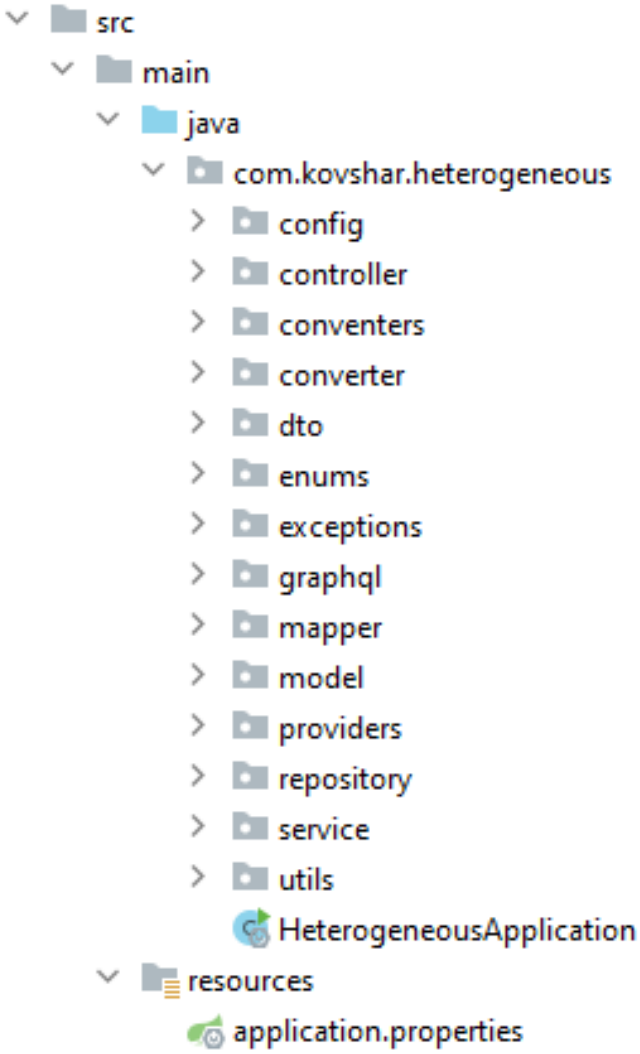

Рисунок 4.1 – пакети агрегаційного модуля

4.1.1. Керування індикаторами

Однією з основних функцій системи є функція керування індикатором. Даний модуль виконує її методами REST ендпоінтів. Для керування забезпечено набір CRUD(create-read-update-delete) операцій над індикатором (рис. 4.2.)

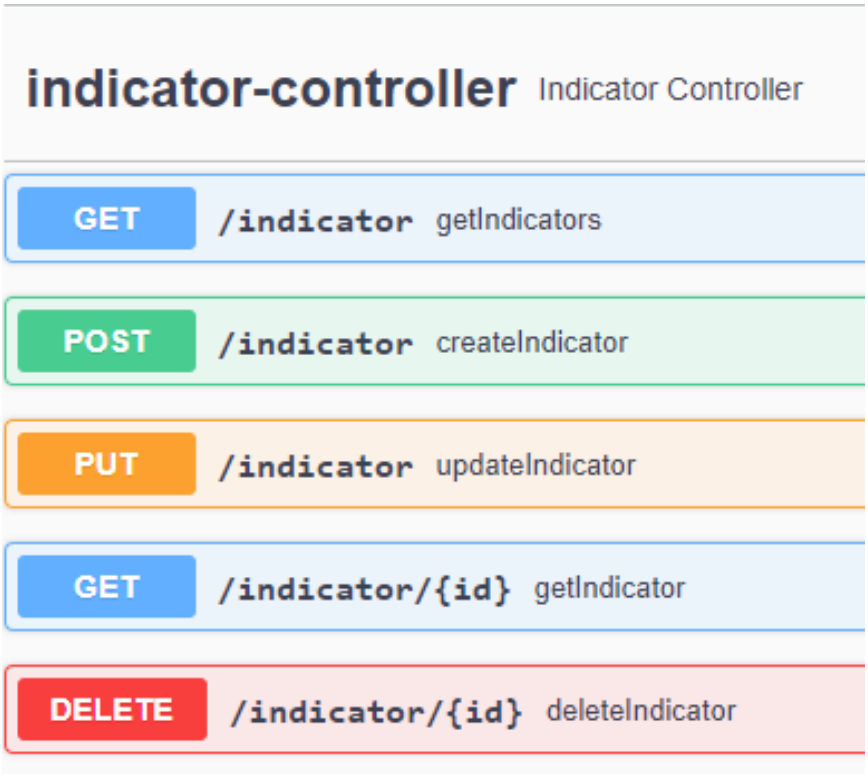

Рисунок 4.2 – методи для роботи з індикатором

На рисунку представлені методи:

- getIndicators – дозволяє отримати список усіх наявних індикаторів
- createIndicator – дозволяє створити новий індикатор
- updateIndicator – дозволяє оновити існуючий індикатор
- getIndicator – дозволяє отримати один індикатор за вказаним id
- deleteIndicator – дозволяє видалити існуючий індикатор

4.1.2. Управління представленням індикаторів

Окрім керування індикаторами, агрегаційним модуль забезпечує функцію представлення індикаторів.

Реалізовано цю функцію наступним чином. Користувач має декілька способів

отримати представлення індикаторів. Першим є метод getIndicators наведений на рис 4.2., або ж користувач може виконати GraphQL запит та отримати перелік лише необхідних індикаторів із необхідними полями (рис 4.3.).

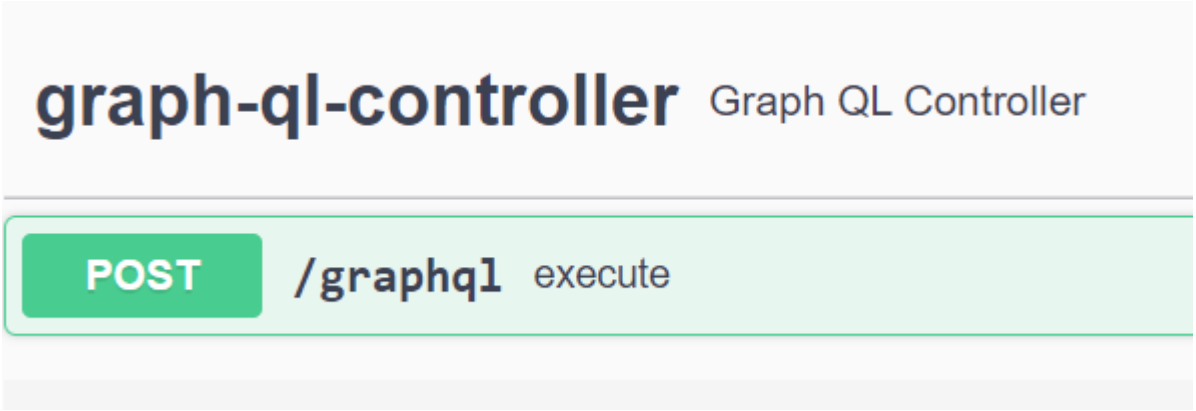

Рисунок 4.3 – методи для виконання GraphQL запиту

4.1.3. Управління файловим представленням індикаторів

Методи наведені у пункті 4.1.2 дають представлення для веб систем. Також існують методи, що використовують дані отримані наведеними методами та формують з них текстові файли. Наразі система підтримує два типи файлів:

- JSON
- CSV

Для отримання текстових даних з усією інформацією про індикатори або отриманих після виконання GraphQL запиту необхідно виконати відповідній метод з вказанням розширення файлу яке підтримується (рис 4.4.).

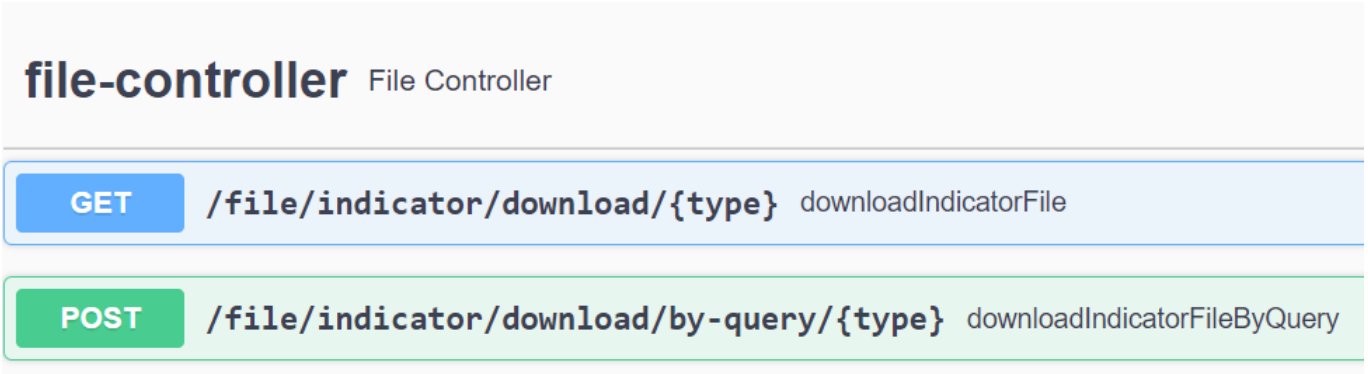

Рисунок 4.4 – методи файлового представлення індикаторів

4.1.4. Отримання переліку полів індикаторів

Для виконання GraphQL запитів, що були описані вище, необхідно зазначити які поля ми хочемо отримати, для цього необхідно знаки які взагалі поля наявні в системі.

Це необхідно оскільки схема даних динамічна і може постійно змінюватися.

З цією метою було розроблено метод для отримання повного переліку полів, наявних у індикаторах (рис. 4.5.). Метод обирає як системні поля, що містить кожен індикатор, так і додані користувацькі поля. Якщо ж поля мають вкладеність, буде проведений пошук у глиб та вибрані усі поля до самого глибшого рівня. Поля з різним рівнем вкладеності роздідені точкою між назвами предка та нащадків.

Рисунок 4.5 – метод отримання переліку полів індикаторів

4.1.5. Керування семантикою полів індикатора

Оскільки наша система має на меті створення рейтингів, як запропонованих системних так і користувацьких, то необхідно забезпечити чесність та об’єктивність цього рейтингу з точки зору даних на яких він буде будуватися. З цією метою було розроблено метод, що дозволить управляти класом, що був описаний у пункті 3.6.

Розроблений метод, виконує CRUD операції над класом FieldMetadata, і також виконує операцію пошуку метадати за заданими ідентифікаторами полів (рис 4.6.).

На рисунку представлені методи:

- getFieldMetadatas – дозволяє отримати список усіх наявних метадат
- createFieldMetadata – дозволяє створити нову метадату
- updateFieldMetadata – дозволяє оновити існуючу метадату
- getFieldMetadata – дозволяє отримати одну метадату за вказаним id
- deleteFieldMetadata – дозволяє видалити існуючу метадату
- searchMetadataByFieldsId – дозволяє отримати перелік метадат за вказаними идентифікаторами полів

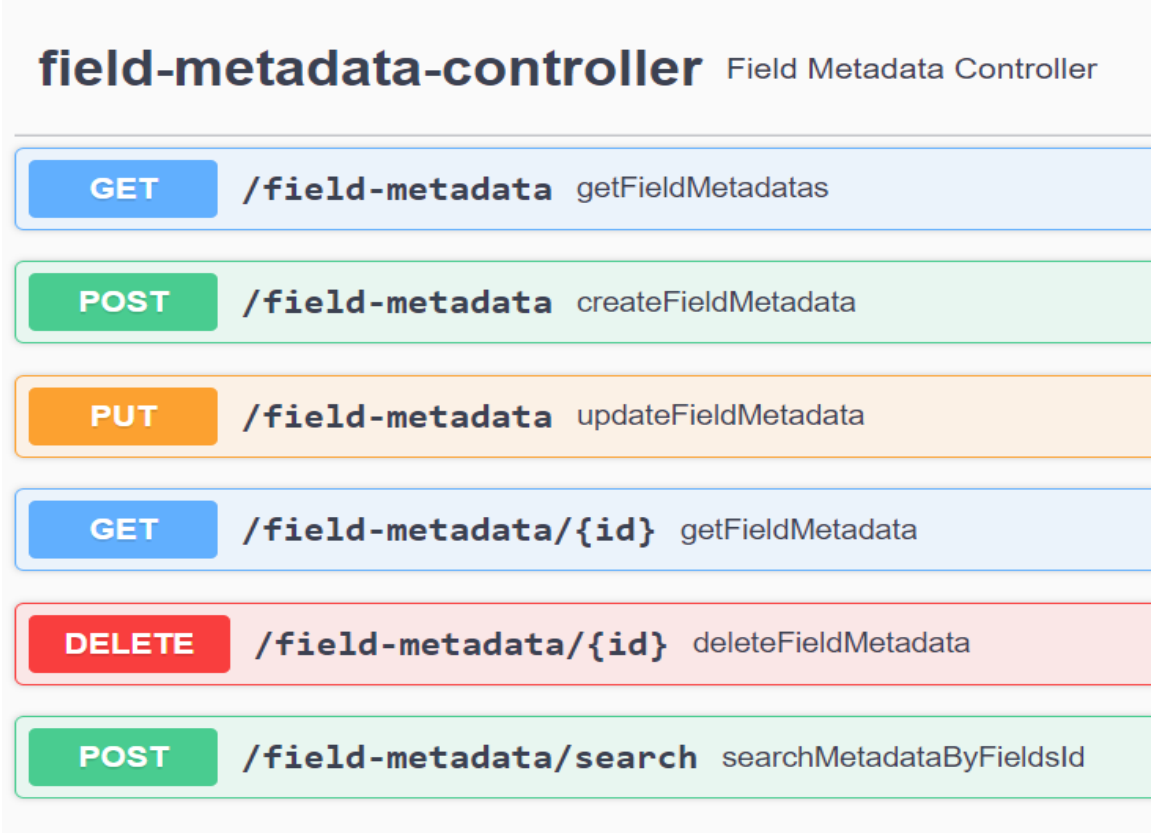

Рисунок 4.6 – методи управління семантикою полів

4.2. Модуль рейтингів

Даний модуль використовується для побудови рейтингів, як запропонований системою так і користувацький. Даний модуль має таку схему пакетів (рис. 4.7.).

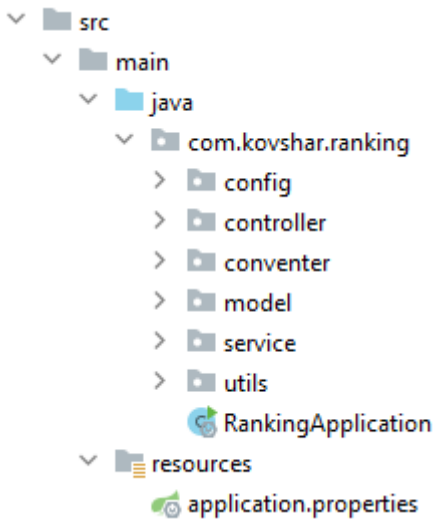

Рисунок 4.7 – пакети модуля рейтингів

Для реалізації поставлених функцій було реалізовано відповідні методи (рис. 4.8):

- createDefaultRanking – дозволяє створити системний рейтинг
- createUserRanking – дозволяє сформувати рейтинг за формулою від користувача

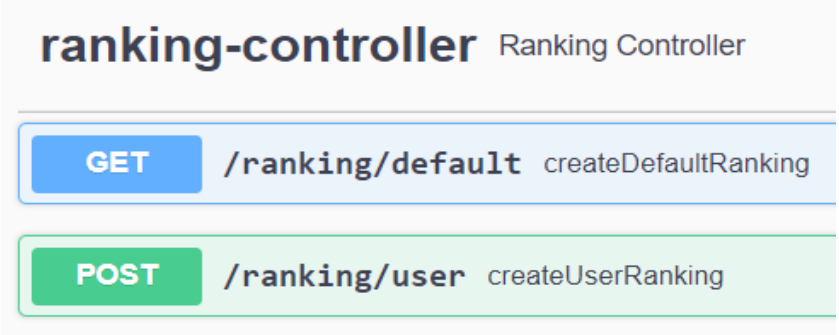

Рисунок 4.8 – методи для створення рейтингів

4.2.1. Формування системного рейтингу

Система пропонує користувачеві сформувати запропонований рейтинг. Рейтинг формується за принципом зваженої суми. Тобто, із системних полів індикаторів було обрано перелік полів згідно з міжнародними стандартами. Ці показники множаться на певне число, вагу, і з отриманих значень формується зважена сума, що і є значенням що береться при формуванні рейтингу. Перелік показників, ваг та зваженої суми наведено у таблиці 4.1. За наведеною таблицею, система обирає з індикаторів поля, наведені в таблиці, валідує згідно з наведеною метадатою полів, та за формулою сформованою відповідно до таблиці обраховує значення рейтингу, сортує їх, і формує остаточний рейтинг.

Таблиця 4.1 - таблиця полів та ваг для формування системного рейтингу

Загальний	Level_1			Level_2		
Зважена сума	Показник	Вага	Значення	Показник	Вага	Значення
	Міжнародне наукове співробітництво	0.25	Зважена сума	Міжнародне співавторство	0.8	Частка публікацій з іноз. співавторами
				Міжнародне використання дослідницької інфраструктури	0.2	Ресурс об'єкта в людино-днях на рік
	Інтеграція з міжнародним дослідницьким середовищем	0.25	Зважена сума	Залучення закордонних науковців	0.5	Частка залучених
				Міжнародна мобільність	0.5	Частка відряджених
	Фінансова складова	0.25	Зважена сума	Обсяги зовнішнього фінансування	0.35	Частка закордонного фінансування
				Ефективність закордонного фінансування	0.3	Ефективність фінансування
				Бюджет спільних наукових програм або проектів	0.35	Частка від загального бюджету
	Міжнародна спрямованість адміністрування	0.25	Зважена сума	Участь у закордонних структурах залучення дослідників	0.5	Частка працівників організації у міжнар. структурах від загальної кількості зайнятих підбором персоналу
				Іноземні фахівці у оцінці наукових досліджень	0.5	Частка іноземних експертів

4.2.2. Формування користувацького рейтингу

Також система дозволяє формувати рейтинг за заданою користувачем формулою. Метод формування користувацького рейтингу, приймає на вхід формулу, за якою і буде формуватися рейтинг. У формулі користувач повинен задати ваги, суму та назви полів, що будуть використовуватися при побудові (рис. 4.9). Надіслана користувачем формула приймається системою, використовує її замість формули наведеної у пункті 4.2.1.

```
0.25 * (0.8 * shareOfPublicationsWithForeignAuthorsPercentage + 0.2 * foreignResearchersNumber *
daysUsedByForeignResearchersNumber / manDaysPerYear) +
0.25 * (0.5 * shareScientistsInvolved + 0.5 * shareScientistsSecondedPercentage) +
0.25 * (0.35 * foreignFinancingPercentage + 0.3 * fundingEfficiency + 0.35 *
organizationFundingSharePercentage) +
0.25 * (0.5 * shareInternationalPersonnelSelectionPercentage + 0.5 * shareOfForeignExpert)
```

Рисунок 4.9– приклад формули для створення користувацького рейтингу

4.3. Адміністративний модуль

Для забезпечення збору статистики, актуалізації даних про модулі та їх роботу, про використання системою ресурсів було створено адміністративний модуль. Даний модуль має користувацький інтерфейс, у якому демонструє інформацію про систему та її модулі, що працюють на даний момент часу (рис. 4.10).

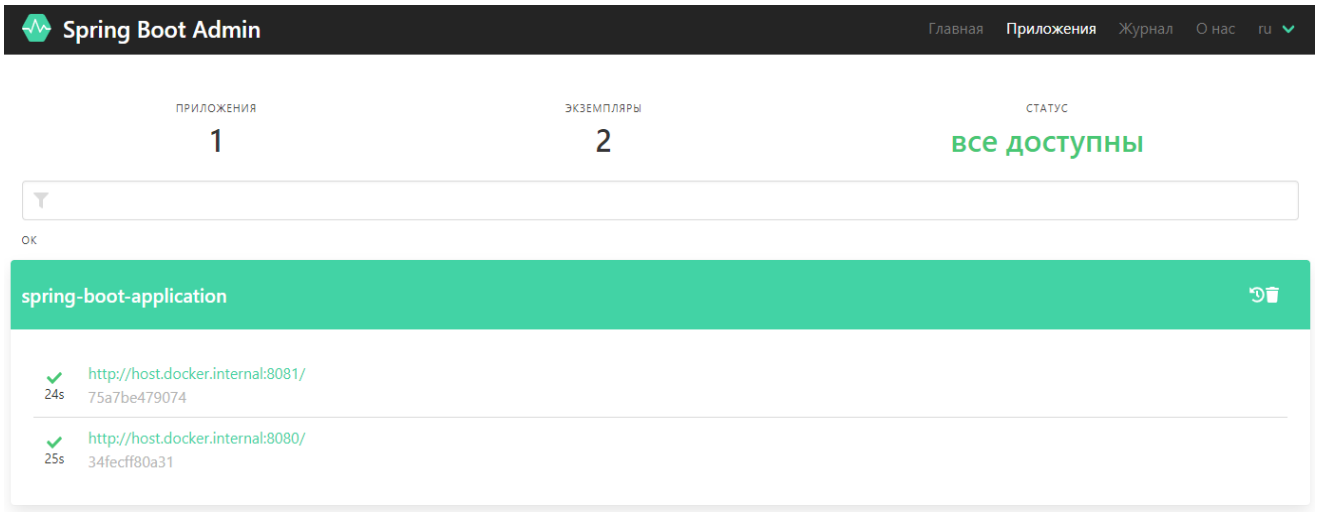

Рисунок 4.10– інтерфейс адміністративного модуля

Натиснувши на певний модуль, ми потрапимо на сторінку з детальною інформацією що до обраного модуля системи (рис. 4.11).

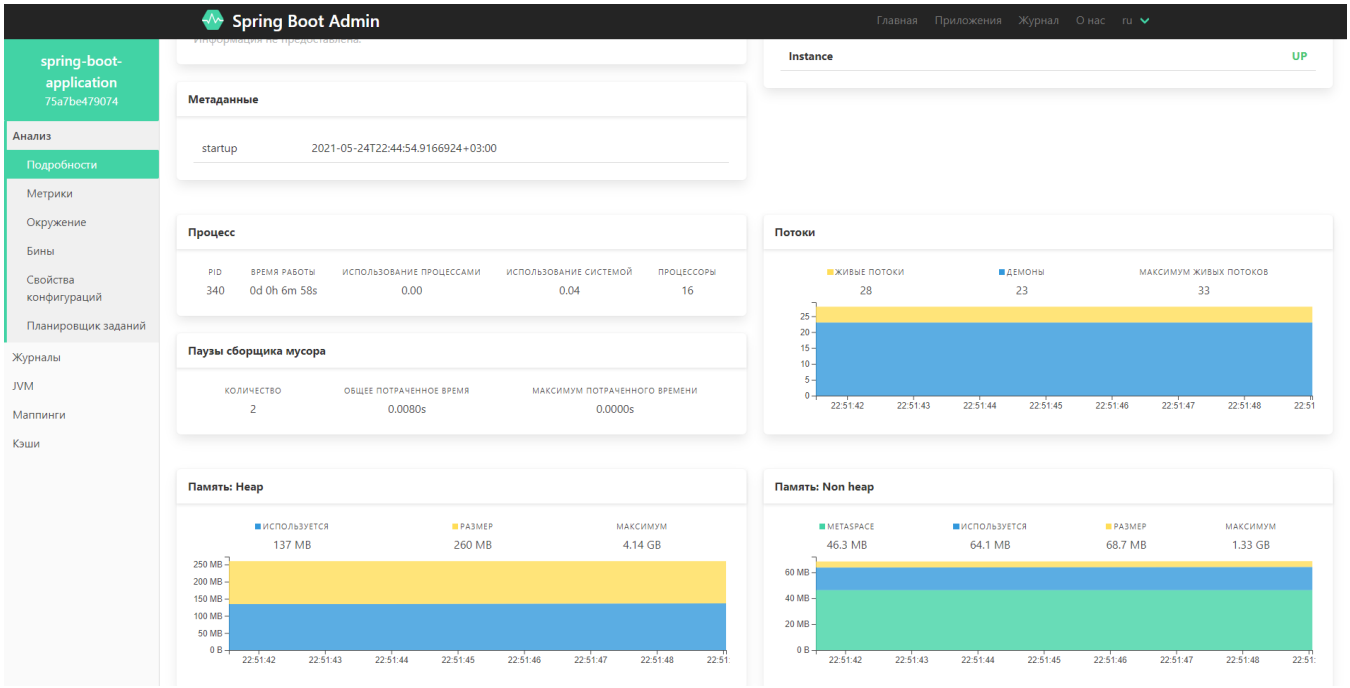

Рисунок 4.11 – інтерфейс детальної інформації що до модуля

Детальний інтерфейс дозволяє користувачу отримати інформацію про:

- класи використані у системі
- кеши системи
- властивості конфігурації
- міграції баз даних
- дані про роботу модуля
- трасування http запитів (за замовчуванням за останні 100 запитів)
- додаткова інформація про модуль
- параметри логування
- інформація про метрики
- сесії користувачів
- інформація про системні потоки

Отже, використання адміністративного модуля допоможе користувачеві отримати необхідні дані що до роботи системи в цілому, провести аналіз, та виявити можливі проблеми та швидко їх усунути не витрачаючи час на залучення спеціалістів для аналізу та дебагу системи або її оточення.

5. ПРИКЛАД ВИКОРИСТАННЯ СИСТЕМИ

5.1. Swagger

Усі наступні приклади використання системи будуть наведені за допомогою застосунку Swagger. Цей застосунок, що входить до складу системи і використовується як автоматична документація REST-API. Усі ендпоінти системи, тобто її методи, будуть автоматично відображені у цьому застосунку. Відповідно кожен модуль має Swagger, та документує свої методи. Автоматична документація знаходиться за наступним шляхом (рис. 5.1.).

Рисунок 5.1 –шлях доступу до Swagger

Відповідно, що якщо певний модуль піднятий не локально, а на виділеному сервері і має іншу IP адресу або доменне ім’я, то шлях localhost:8080 має бути заміненій відповідну на адресу серверу та порт на котрому працює модуль.

Як було сказано вище, даний засіб демонструє перелік усіх наявних методів системи, і також їх детальний опис (рис. 5.2 – 5.3).

Опис включає в себе наступні характеристики:

- шлях методу
- HTTP-метод для виклику
- назва методу
- приклад вхідних даних (якщо необхідні)
- приклад вихідних даних
- тип вхідних даних
- тип вихідних даних
- інформація про код відповіді

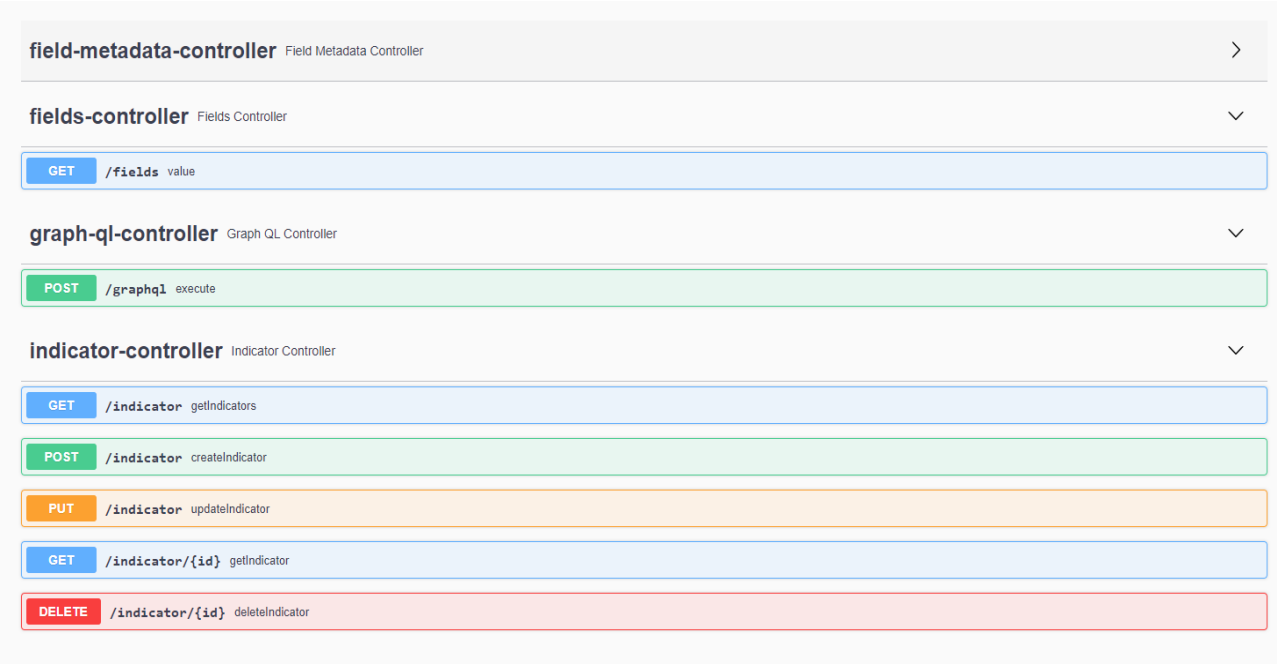

Рисунок 5.2 –перелік ендпоінтів агрегаційного модуля

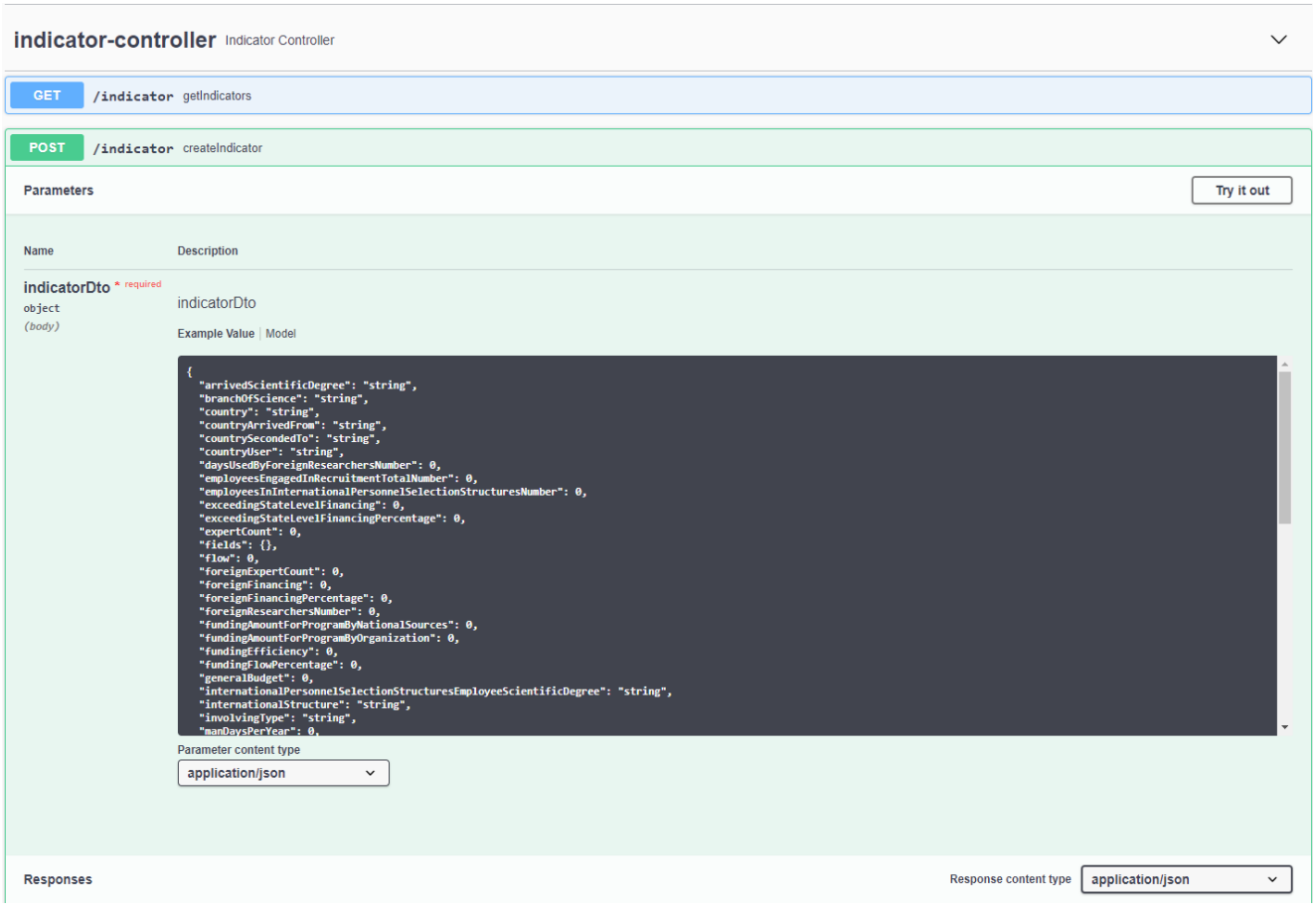

Рисунок 5.3 –детальний інформація про обраний метод

Окрім інформації, Swagger дозволяє також виконати обраний запит. Натиснувши на кнопку Try it out (рис. 5.4) ми потрапляємо до меню виконання запиту (рис. 5.5), де ми можемо редагувати вхідні дані, та власне виконати сам запит та отримати

відповідь.

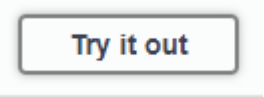

Рисунок 5.4 – кнопка для переходу до меню запиту

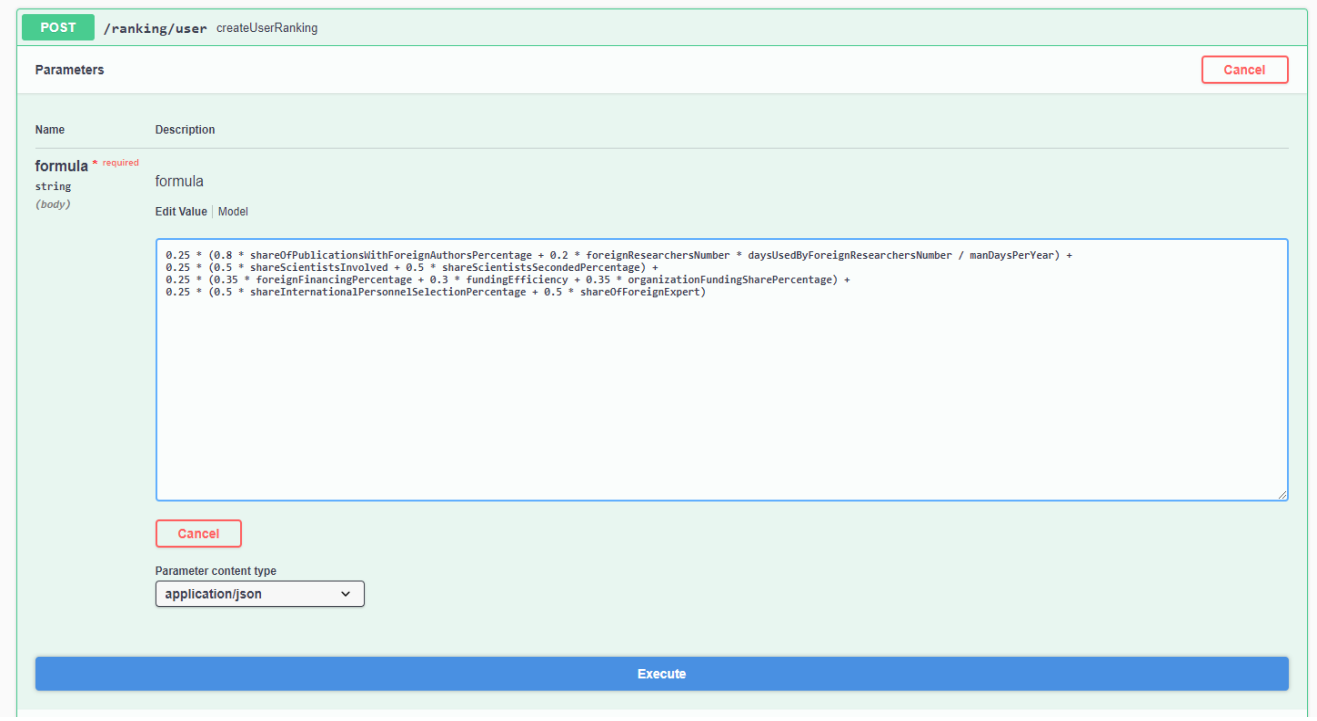

Рисунок 5.5 – меню запиту

Рисунок 5.5 демонструє приклад для виконання запиту для створення рейтингу за алгоритмом від користувача. На прикладі зображено, що буде виконано HTTP-метод POST на шлях /ranking/user з тілом типу JSON, вхідними даними слугує формула від користувача для створення рейтингу. Натиснувши кнопку Execute ми виконаємо заданий запит та отримаємо відповідь від системи (рис. 5.6). В результаті ми будемо мати сформований рейтинг за алгоритмом користувача.

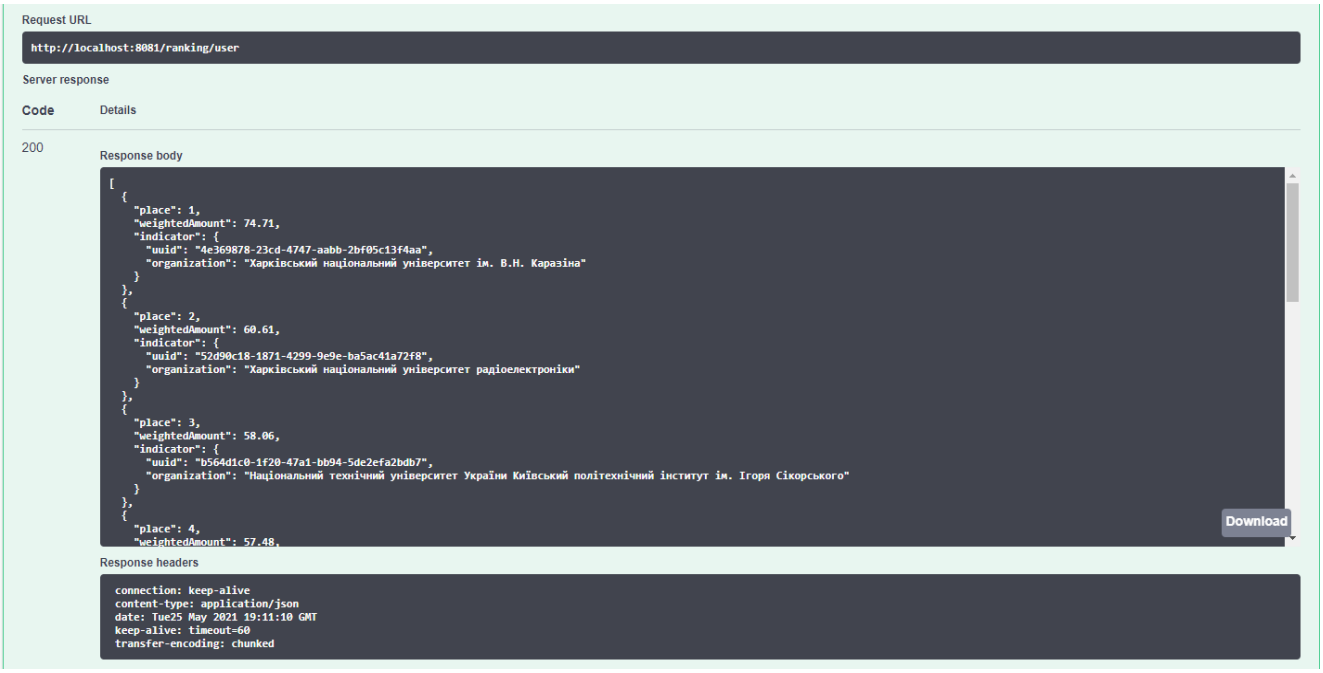

Рисунок 5.6 – результат запиту

5.2. Тестування системи

Наступні інструкції можуть бути прикладом з використання системи. Оскільки метою дипломної роботи було створення системи для побудови рейтингів, то всі дії продемонстровані у прикладах будуть поступово наближати до цієї мети, оскільки це і є кінцевою точкою системи.

Далі наведено наступні приклади використання функцій системи:

- Створення індикатора
- Представлення індикатора
- Отримання переліку полів індикатора
- Файлове представлення індикатора
- Створення обмежень для полів
- Отримання рейтингу
- Адміністративні функції

Система розроблена за принципом REST. Кожна функція системи це окремий метод

для виклику. У разі успішного виконання запиту система поверне відповідний очікуваний результат. У разі помилки система поверне статус код помилки, і також інформацію про помилку, також ця інформація буде наявна у лозі системи.

5.2.1. Створення індикатора

Для створення рейтингу необхідні дані на яких він буде побудований. У розробленій системі рейтинг будується на індикаторах певної установи. Отже, для того щоб отримати рейтинг, в першу чергу необхідно створити відповідні індикатори для певної установи.

З цією метою виконаємо метод для створення індикатора, з відповідним тілом (рис. 5.7.). У тілі вказуємо перелік полів із системних та також додамо своє поле, що використаємо при побудові рейтингу.


```
{
  "organization": "Національний технічний університет України Київський політехнічний інститут ім. Ігоря Сікорського",
  "foreignFinancingPercentage": 22,
  "fundingEfficiency": 41,
  "shareOfPublicationsWithForeignAuthorsPercentage": 67,
  "shareScientistsInvolved": 48,
  "shareScientistsSecondedPercentage": 45,
  "organizationFundingSharePercentage": 59,
  "foreignResearchersNumber": 9,
  "daysUsedByForeignResearchersNumber": 70,
  "manDaysPerYear": 44,
  "shareInternationalPersonnelSelectionPercentage": 18,
  "shareOfForeignExpert": 29,
  "fields": {
    "correlation": 10
  }
}
```


Рисунок 5.7 – тіло створеного індикатора

5.2.2. Представлення індикатора

Після створення індикатора, необхідно перевірити чи отримала наша система усі дані вірно. Для цього необхідно виконати запит та переглянути на наведені індикатори. Але система може мати їх багато, і на пошук необхідного індикатора може піти багато часу, тому метод для перегляду усіх індикаторів не дуже для цього підійде. Альтернативою може стати метод, що повертає індикатор за вказаним ідентифікатором, цей метод поверне саме той індикатор що нам і потрібен, але буде містити перелік усіх системних полів, і якщо вони не матимуть значень то будуть

мати значення null. Шукати та перевіряти 12 полів із 50 також не дуже зручно.

Для вирішення такої проблеми, було розроблено метод, на основі GraphQL запиту, що повертає лише ті дані, що ми запросили у системи.

Отже, для перегляду створеного нами індикатора, виконаємо GraphQL запит та отримаємо його результат (рис. 5.8.). Для отримання лише одного нашого індикатора вказуємо його ідентифікатор, а для отримання необхідного нам переліку полів, вказуємо лише ті поля що ми задали у пункті 5.2.1.. У результаті ми маємо дані відповідні з тими, що і були вказані при створенні.

Рисунок 5.8 – GraphQL запит та його результат

5.2.3. Отримання переліку полів індикатора

Під час користування системою, може виникнути проблема, що користувач забуде поля, що були додані, або ж не знатиме про поля додані іншим користувачем, але йому необхідно буде отримати їх представлення. А оскільки наведений у попередньому пункті GraphQL запит потребує на вхід перелік лише наявних у індикаторів полів, то необхідно було забезпечити користувача такою можливістю.

З цією метою було розроблено метод для отримання усіх наявних у системі полів. Якщо виконати даний метод, то у результаті, як і очікувалось, отримаємо перелік усіх полів (рис. 5.9.).

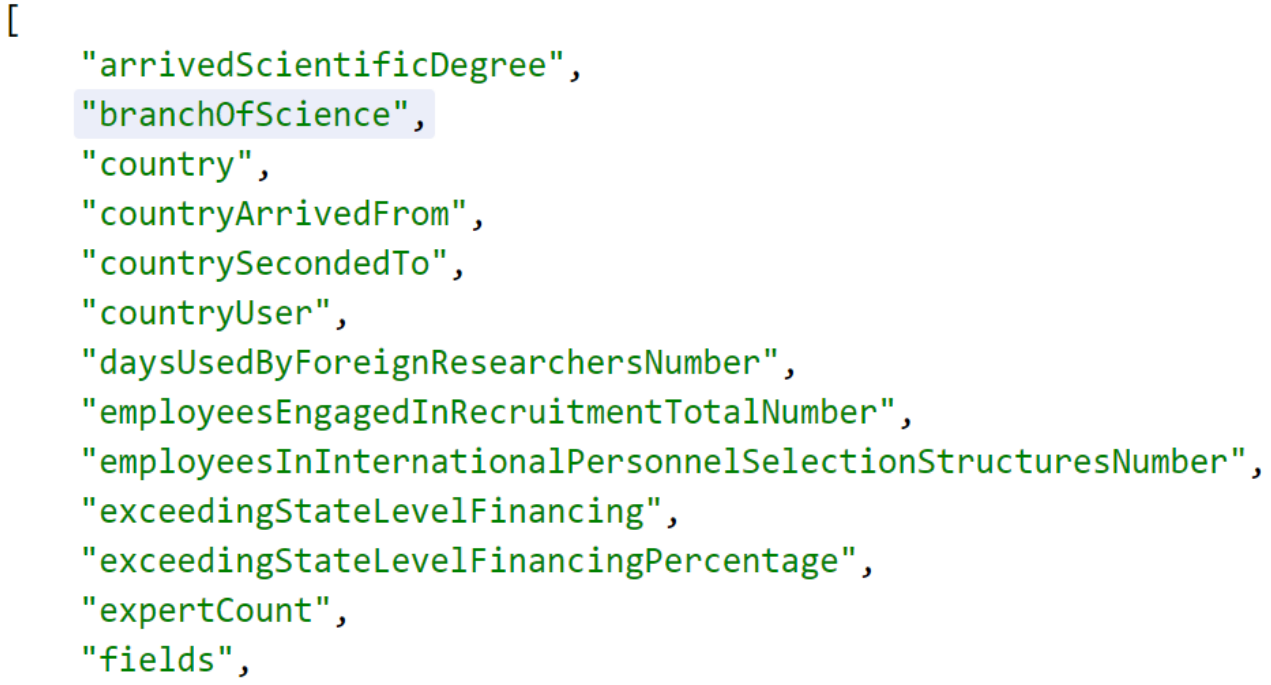

Рисунок 5.9 – результат методу для отримання усіх полів

5.2.4. Файлове представлення індикатора

Важливо зазначити, що під час роботи із системою у користувача може виникнути необхідність експортувати дані про індикатори у файлові вигляді. Ця необхідність була врахована, та було створено відповідно методи для експорту даних про індикатори у файл.

Система підтримує формування файлів типу JSON та CSV. Для формування файлу з інформацією про усі індикатори з повним набором даних, необхідно виконати відповідний метод та вказати тип бажаного файлу. На рис. 5.10. ми можемо бачити приклад JSON файлу, а на рис. 5.11. – CSV відповідно.

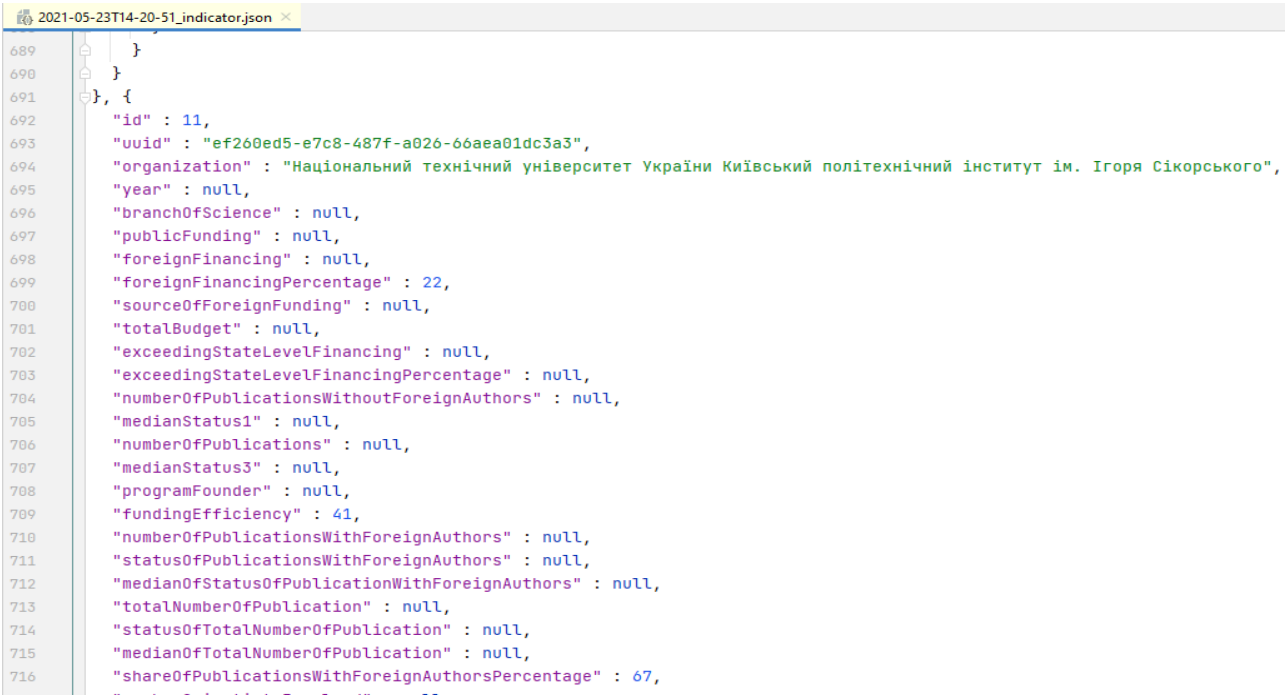

Рисунок 5.10 – приклад згенерованого JSON файлу

	ors	medianOfTotalNumberOfPublication	medianStatus1	medianStatus3	numberOfPublications	numberOfPublicationsWithForeignAuthors	numberOfPu
1		98	28	340	127	279	10
2		274	87	4	306	578	368
3		116	192	334	476	521	188
4		223	19	420	431	735	317
5		95	169	378	994	246	112
6		258	101	87	575	566	60
7		297	12	70	435	175	124
8		142	15	175	965	2	328
9		12	41	390	817	27	365
10		140	165	245	693	325	337
11		null	null	null	null	null	null

Рисунок 5.11 – приклад згенерованого CSV файлу

Але, можуть виникнути проблеми описані у пункті 5.2.2, або ж користувачеві буде необхідна певна інформація лише по певним індикатором. Для вирішення цієї проблеми було створено ще один метод для експорту у даних у файл. Цей метод приймає на вхід, такий же запит як і запит описаний у пункті 5.2.2 з вказанням бажаного типу файлу, але у результаті запиту буде сформований файл лише з тими даними, що були вказані користувачем. Рисунок 5.12 та 5.13 демонструють приклади файлів, створених на основі користувацьких запитів.

	daysUsedByForeignResearchersNumber	fields	fields.correlation	foreignFinancingPercentage	foreignResearchersNumber	fundingEfficiency	i
1	70	<null>	10	22	9	41	11

Рисунок 5.12 – приклад згенерованого CSV файлу за користувацьким запитом

	fields.correlation	fundingEfficiency	id	manDaysPerYear	organization	shareOfForeignExpert
1	10	41	11	44	Національний технічн…	29

Рисунок 5.13 – інший приклад згенерованого CSV файлу за користувацьким запитом

5.2.5. Створення обмежень для полів

Наступним кроком для формування рейтингу буде створення обмежень для полів. Це зроблено для забезпечення цілісності та об’єктивності рейтингу з точки зору даних на яких його буде побудовано. Для створення цих даних, необхідно викликати відповідний метод, та вказати там фільтр для даних полів (рис. 5.14).

```
[
  {
    "fieldId": "shareOfPublicationsWithForeignAuthorsPercentage",
    "type": "INT",
    "operation": "AND",
    "filters": [{"operator": "BIGGER_EQUAL", "value": "0"}, {"operator": "LESS_EQUAL", "value": "100"}]
  },
  {
    "fieldId": "foreignResearchersNumber",
    "type": "INT",
    "operation": "AND",
    "filters": [{"operator": "BIGGER_EQUAL", "value": "0"}, {"operator": "LESS_EQUAL", "value": "100"}]
  },
  {
    "fieldId": "daysUsedByForeignResearchersNumber",
    "type": "INT",
    "operation": "AND",
    "filters": [{"operator": "BIGGER_EQUAL","value": "0"},{"operator": "LESS_EQUAL","value": "100"}]
  },
  {
    "fieldId": "manDaysPerYear",
    "type": "INT",
    "operation": "AND",
    "filters": [{"operator": "BIGGER_EQUAL","value": "0"},{"operator": "LESS_EQUAL","value": "100"}]
  },
  {
    "fieldId": "shareScientistsInvolved",
    "type": "INT",
    "operation": "AND",
    "filters": [{"operator": "BIGGER_EQUAL","value": "0"},{"operator": "LESS_EQUAL","value": "100"}]
  },
```

Рисунок 5.14 – приклад створення обмежень для полів

5.2.6. Отримання рейтингу

Після виконання усіх вищенаведених операцій ми нарешті можемо отримати рейтинг. Для отримання системного рейтингу необхідно викликати відповідний метод та отримати рейтинг установ (рис. 5.15).

```
[
  {
    "place": 1,
    "weightedAmount": 111.38,
    "indicator": {
      "uuid": "c8608236-0244-4688-98cf-5d076320c88c",
      "organization": "Національний технічний університет України Київський політехнічний інститут ім. Ігоря Сікорського"
    }
  },
  {
    "place": 2,
    "weightedAmount": 65.14,
    "indicator": {
      "uuid": "5872cf1f-2fd1-4963-83db-74d49e0e4f0f",
      "organization": "Львівський національний університет ім. Івана Франка"
    }
  },
  {
    "place": 3,
    "weightedAmount": 58.78,
    "indicator": {
      "uuid": "0549fc28-64be-4808-949e-2de2417beae7",
      "organization": "Сумський державний університет"
    }
  },
  {
    "place": 4,
    "weightedAmount": 55.88,
    "indicator": {
```


Рисунок 5.15 – приклад готового системного рейтингу

Якщо ж користувач хоче отримати рейтинг за альтернативним алгоритмом, то необхідно викликати метод, що формуватиме рейтинг за формулою користувача та передати йому на вхід формулу (рис. 5.16.) у результаті виконання методу буде отримано рейтинг за формулою користувача (рис. 5.17.).

```
(0.25 *
(0.8 * shareOfPublicationsWithForeignAuthorsPercentage +
0.2 * foreignResearchersNumber * daysUsedByForeignResearchersNumber / manDaysPerYear) +
0.25 *
(0.5 * shareScientistsInvolved +
0.5 * shareScientistsSecondedPercentage) +
0.25 *
(0.35 * foreignFinancingPercentage +
0.3 * fundingEfficiency +
0.35 * organizationFundingSharePercentage) +
0.25 * (0.5 * shareInternationalPersonnelSelectionPercentage + 0.5 * shareOfForeignExpert))
* fields.correlation
```

Рисунок 5.16 – приклад формули для формування користувацького рейтингу

```
[
  {
    "place": 1,
    "weightedAmount": 1113.75,
    "indicator": {
      "uuid": "c8608236-0244-4688-98cf-5d076320c88c",
      "organization": "Національний технічний університет України Київський політехнічний інститут ім. Ігоря Сікорського"
    }
  },
  {
    "place": 2,
    "weightedAmount": 651.37,
    "indicator": {
      "uuid": "5872cf1f-2fd1-4963-83db-74d49e0e4f0f",
      "organization": "Львівський національний університет ім. Івана Франка"
    }
  },
  {
    "place": 3,
    "weightedAmount": 587.78,
    "indicator": {
      "uuid": "0549fc28-64be-4808-949e-2de2417beae7",
      "organization": "Сумський державний університет"
    }
  },
```


Рисунок 5.17 – приклад рейтингу на побудованого на основі користувацької формули

5.2.7. Адміністративні функції

Для одержання певної інформації про роботу системи, про використані нею ресурси або ж для певного профілювання системи, можна викликати відповідно адміністративний модуль системи та одержати необхідну інформацію (рис. 5.18.).

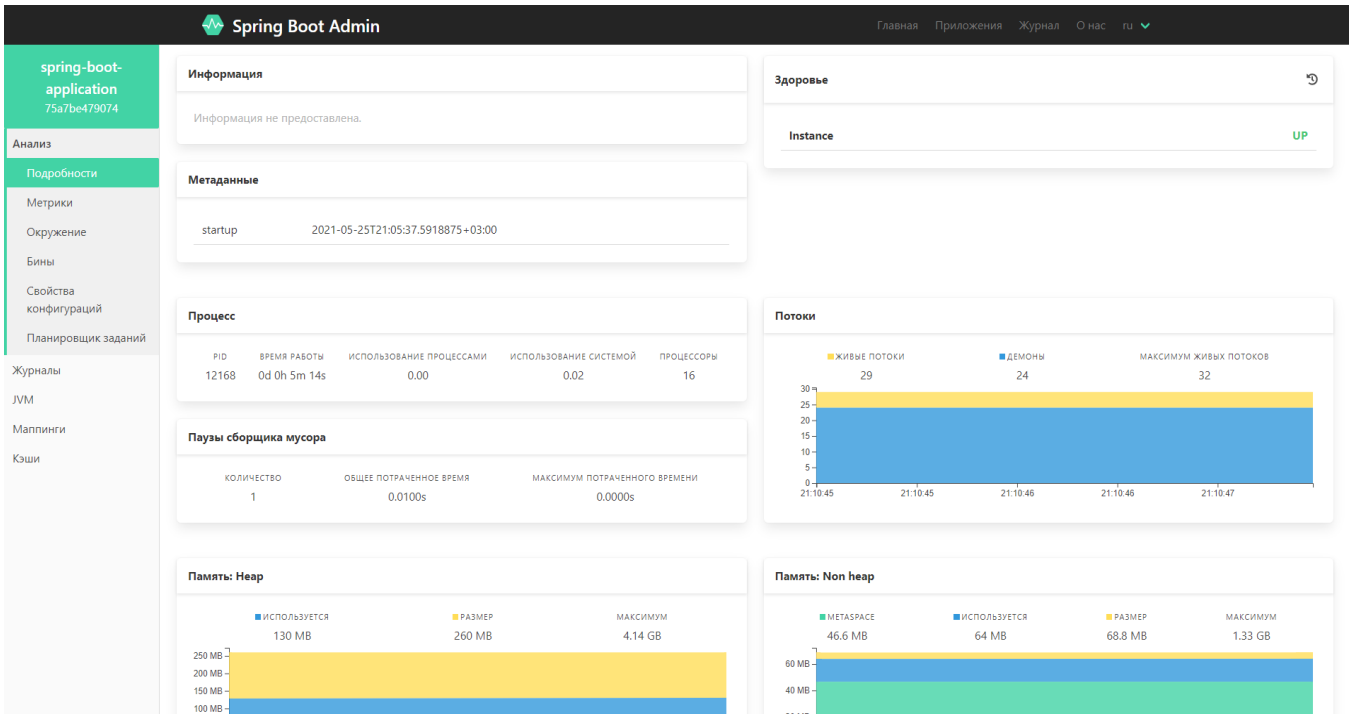

Рисунок 5.18 – приклад метрик системи

5.3. Інсталювання системи

Як було зазначено система складається з трьох модулів. Отже для коректної роботи системи та підтримки усіх вищезазначених функцій необхідно інсталювати усі модулі.

Для інсталяції усіх модулів на локальному комп'ютері необхідна бути встановлена MongoDB, Java та якщо у наявності вихідний код програми Maven.

Для формування виконуваного .jar файлу з вихідного коду системи необхідно виконати наступну команду: *mvn clean package*

Маючи виконуваний jar файл необхідно виконати команду *java –jar ${fileName}.jar*, де замість ${fileName} вказати назву файлу для запуску, для запуску модуля.

Усі вище наведені дії виконуються для усіх модулів системи.

У разі використання модулів на різних серверах необхідно перед запуском модуля у файлі application.properties:

- для агрегаційного модулю замінити параметр spring.boot.admin.client.url на шлях до адміністративного модуля.
- для модуля рейтингів замінити параметр spring.boot.admin.client.url на шлях до адміністративного модуля, і також замінити aggregation.application.root-path на шлях до агрегаційного модуля.

Для зміни порту для запуску модуля необхідно у файлі application.properties вказати параметр server.port на бажаний порт. Слід пам'ятати, що кожен модуль повинен бути запущений на окремому порту. За замовчуванням, усі модулі знаходяться за хостом – 127.0.0.1 та портах:

- 8080 – агрегаційний модуль
- 8081 – модуль рейтингів
- 8082 – адміністративний модуль

ВИСНОВКИ

Існуючі рейтингові системи, не дають можливості, яка б дозволяла обирати актуальний для користувача набір індикаторів та обчислювати рівень розвитку установи за алгоритмами, що відповідають стану та політиці розвитку множини установ.

Було сформовано власний набір показників, що дозволяють оцінювати системи за широким спектром напрямків і також було розроблено засіб для створення та використання нових показників, з використанням MongoDB як сховища для даних без чіткої структури та GraphQL для можливості керування представленнями такими даними.

Для забезпечення чіткості рейтингу, було розроблено засіб для накладення обмежень на дані, що використовуватимуться при побудові рейтингу. Засіб представлено у вигляді фільтрів, які повинні пройти дані перед побудовою рейтингу.

З наведених запропонованих показників було сформовано системний рейтинг, що враховує усі запропоновані показники, та наводить рейтинг, а отже і наводить оцінку за широким переліком напрямків розвитку установи.

Для забезпечення користувача можливістю самому формувати рейтинг та проводити оцінку за альтернативним алгоритмом, було розроблено метод для формування рейтингу за формулою від користувача.

Створена система пропонує власний рейтинг і, на відміну від інших систем, забезпечує користувача засобами для створення власних показників та побудови рейтингів на основі альтернативних алгоритмів. Усе це забезпечує великій потенціал до подальшого використання системи та входження її до класу передових рейтингових систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Scimago Institusion Rankings [Електронний ресурс] – Режим доступу до ресурсу: https://www.scimagoir.com
2. Leiden Ranking [Електронний ресурс] – Режим доступу до ресурсу: https://www.leidenranking.com
3. Shanghai Ranking [Електронний ресурс] – Режим доступу до ресурсу: http://www.shanghairanking.com
4. GraphQL [Електронний ресурс] – Режим доступу до ресурсу: https://graphql.org
5. Joshua B. Effective Java / Bloch Joshua., 2017. – 414 с. – (3rd edition)
6. Spring [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io
7. Chodorow K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage / Kristina Chodorow., 2013. – 432 с. – (Second edition).

ДОДАТОК А

Управління представленнями гетерогенних даних

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР72259_21Б

Аркушів 2

Київ — 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР72259_21Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР72259_21Б 12-1	Indicator.java	Компонент, що містить індикатори установ та засіб для створення показників
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР72259_21Б 12-2	IndicatorService .java	Компонент, що виконує операції над класом індикатора
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР72259_21Б 12-3	IndicatorControl ler.java	Компонент, головна точка входу для виклику операцій над індикаторами
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР72259_21Б 12-4	DefaultRanking Service.java	Компонент, для побудови системного рейтингу установ

ДОДАТОК Б

Управління представленнями гетерогенних даних

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР72259_21Б 12-1

Аркушів 9

Київ — 2021

Indicator.java

```
package com.kovshar.heterogeneous.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.Date;
import java.util.Map;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Document(collection = "indicators")
@Builder
public class Indicator {
   @Transient
   public static final String SEQUENCE_NAME = "users_sequence";

   @Id
   private Long id;
   private String uuid;
   //ПЗФ
   private String organization;
   private Integer year;
   private String branchOfScience;
   private Long publicFunding;
   private Long foreignFinancing;
   private Integer foreignFinancingPercentage;
   private String sourceOfForeignFunding;
   private Long totalBudget;
   private Long exceedingStateLevelFinancing;
   private Integer exceedingStateLevelFinancingPercentage;
   //ПЕФ
   private Integer numberOfPublicationsWithoutForeignAuthors;
   private Integer medianStatus1;
   private Integer numberOfPublications;
   private Integer medianStatus3;
   private String programFounder;
   private Integer fundingEfficiency;
   //ПМС
   private Integer numberOfPublicationsWithForeignAuthors;
```

```
private String statusOfPublicationsWithForeignAuthors;
private Integer medianOfStatusOfPublicationWithForeignAuthors;
private Integer totalNumberOfPublication;
private String statusOfTotalNumberOfPublication;
private Integer medianOfTotalNumberOfPublication;
private Integer shareOfPublicationsWithForeignAuthorsPercentage;
//ПЗН
private Integer numberScientistsInvolved;
private String involvingType;
private Integer totalNumberScientists;
private String country;
private Integer shareScientistsInvolved;
//ПММ
private Integer numberScientistsArrived;
private String typeStayScientistsArrived;
private String countryArrivedFrom;
private String arrivedScientificDegree;
private Integer numberScientistsSeconded;
private String countrySecondedTo;
private String secondedScientificDegree;
private Integer totalScientificNumber;
private Integer shareScientistsArrivedPercentage;
private Integer shareScientistsSecondedPercentage;
private Integer flow;
//ПСП
private String program;
private String participatingCountries;
private Integer fundingAmountForProgramByOrganization;
private Integer fundingAmountForProgramByNationalSources;
private Long generalBudget;
private Integer organizationFundingSharePercentage;
private Integer fundingFlowPercentage;
//ПДІ
private String researchInfrastructureObject;
private String researchInfrastructureType;
private String countryUser;
private Integer foreignResearchersNumber;
private Integer daysUsedByForeignResearchersNumber;
private Integer manDaysPerYear;
//ПМЗ
private Integer employeesInInternationalPersonnelSelectionStructuresNumber;
private String internationalPersonnelSelectionStructuresEmployeeScientificDegree;
private String internationalStructure;
private Integer employeesEngagedInRecruitmentTotalNumber;
private String recruitmentEmployeeScientificDegree;
private Integer shareInternationalPersonnelSelectionPercentage;
//ПМО
private Integer foreignExpertCount;
private Integer expertCount;
```

```
        private Integer shareOfForeignExpert;

        private Map<String, Object> fields;

        @Override
        public String toString() {
            return "Indicator{" +
                    "id=" + id +
                    ", uuid='" + uuid + '\'' +
                    '}';
        }
}
```

IndicatorService.java

```
package com.kovshar.heterogeneous.service;

import com.kovshar.heterogeneous.dto.IndicatorDto;
import com.kovshar.heterogeneous.exceptions.IndicatorNotFoundException;
import com.kovshar.heterogeneous.mapper.IndicatorMapper;
import com.kovshar.heterogeneous.model.DatabaseSequence;
import com.kovshar.heterogeneous.model.Indicator;
import com.kovshar.heterogeneous.repository.IndicatorRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.query.Update;
import org.springframework.stereotype.Service;

import java.util.*;

import static org.springframework.data.mongodb.core.FindAndModifyOptions.options;
import static org.springframework.data.mongodb.core.query.Criteria.where;
import static org.springframework.data.mongodb.core.query.Query.query;

@Service
@RequiredArgsConstructor
public class IndicatorService {
    private final IndicatorRepository indicatorRepository;
    private final IndicatorMapper mapper;
    private final SequenceService sequenceService;

    public Indicator getById(Long id) {
        return indicatorRepository.findById(id)
                .orElseThrow(() -> new IndicatorNotFoundException(id));
    }

    public List<Indicator> getAllByIds(Long[] ids) {
        return indicatorRepository.findAllByIdIn(ids);
    }
```

```
    public List<Indicator> findAll() {
        return indicatorRepository.findAll();
    }

    public Indicator saveIndicator(IndicatorDto indicatorDto) {
        Indicator indicator = mapper.fromIndicatorDto(indicatorDto);
        indicator.setId(sequenceService.generateSequence(Indicator.SEQUENCE_NAME));
        indicator.setUuid(UUID.randomUUID().toString());
        return indicatorRepository.save(indicator);
    }

    public Indicator saveIndicator(Indicator indicator) {
        return indicatorRepository.save(indicator);
    }

    public Indicator deleteById(Long id) {
        Indicator indicator = getById(id);
        indicatorRepository.deleteById(id);
        return indicator;
    }
}
```

IndicatorController.java

```
package com.kovshar.heterogeneous.controller;

import com.kovshar.heterogeneous.dto.IndicatorDto;
import com.kovshar.heterogeneous.model.Indicator;
import com.kovshar.heterogeneous.service.IndicatorService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/indicator")
@RequiredArgsConstructor
@Slf4j
public class IndicatorController {
    private final IndicatorService indicatorService;

    @PostMapping(consumes = MediaType.APPLICATION_JSON_VALUE, produces =
MediaType.APPLICATION_JSON_VALUE)
    public Indicator createIndicator(@RequestBody IndicatorDto indicatorDto) {
        log.debug("Create indicator {}", indicatorDto);
        Indicator indicator = indicatorService.saveIndicator(indicatorDto);
        log.debug("Created indicator {}", indicator);
        return indicator;
```

```
    }

    @PutMapping(consumes = MediaType.APPLICATION_JSON_VALUE, produces =
MediaType.APPLICATION_JSON_VALUE)
    public Indicator updateIndicator(@RequestBody Indicator indicator) {
        log.debug("Update indicator {}", indicator);
        Indicator saveIndicator = indicatorService.saveIndicator(indicator);
        log.debug("Updated indicator {}", saveIndicator);
        return saveIndicator;
    }

    @GetMapping(produces = MediaType.APPLICATION_JSON_VALUE)
    public List<Indicator> getIndicators() {
        log.debug("Get all indicators");
        List<Indicator> all = indicatorService.findAll();
        log.debug("Get indicators {}", all);
        return all;
    }

    @GetMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    public Indicator getIndicator(@PathVariable Long id) {
        log.debug("Get indicator by id {}", id);
        Indicator indicator = indicatorService.getById(id);
        log.debug("Got indicator {}", indicator);
        return indicator;
    }

    @DeleteMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    public Indicator deleteIndicator(@PathVariable Long id) {
        log.debug("Delete indicator by id {}", id);
        Indicator indicator = indicatorService.deleteById(id);
        log.debug("Deleted indicator {}", indicator);
        return indicator;
    }
}
```

DefaultRankingService.java

```
package com.kovshar.ranking.service.impl;

import com.kovshar.ranking.conventer.JsonObjectConverter;
import com.kovshar.ranking.model.FieldMetadata;
import com.kovshar.ranking.model.Indicator;
import com.kovshar.ranking.model.IndicatorRating;
import com.kovshar.ranking.model.WightedIndicator;
import com.kovshar.ranking.model.dto.IndicatorDto;
import com.kovshar.ranking.service.AggregationRestService;
import com.kovshar.ranking.service.FieldsIdsFetcher;
import com.kovshar.ranking.service.MetadataValidator;
import com.kovshar.ranking.service.RankingService;
```

```
import lombok.RequiredArgsConstructor;
import net.objecthunter.exp4j.Expression;
import net.objecthunter.exp4j.ExpressionBuilder;
import org.json.JSONObject;
import org.springframework.stereotype.Service;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.*;
import java.util.function.Function;
import java.util.stream.Collectors;

import static com.kovshar.ranking.utils.FieldsUtils.getDataFromField;

@Service
@RequiredArgsConstructor
public class DefaultRankingService implements RankingService {
   private final AggregationRestService restService;
   private final MetadataValidator validator;
   private final JsonObjectConverter converter;
   private final FieldsIdsFetcher fieldsIdsFetcher;

   @Override
   public List<IndicatorRating> createDefaultSystemRanking() {
      List<Indicator> indicators = restService.fetchAllIndicators();
      List<WightedIndicator> wightedIndicators = getWightedIndicators(indicators,
this.toWightedIndicator());
      return createWightedIndicatorRating(wightedIndicators);
   }

   @Override
   public Object createUserRanking(String formula) {
      List<Indicator> indicators = restService.fetchAllIndicators();
      List<String> ids = fieldsIdsFetcher.fetchFieldsIds(formula);
      List<WightedIndicator> wightedIndicators = getWightedIndicators(indicators,
this.toFormulaWightedIndicator(formula, ids));
      return createWightedIndicatorRating(wightedIndicators);
   }

   private List<IndicatorRating> createWightedIndicatorRating(List<WightedIndicator>
wightedIndicators) {
      List<IndicatorRating> indicatorRatingList = new ArrayList<>(wightedIndicators.size());
      for (int i = 0; i < wightedIndicators.size(); i++) {
         WightedIndicator wightedIndicator = wightedIndicators.get(i);
         BigDecimal weightedAmount =
BigDecimal.valueOf(wightedIndicator.getWightedAmount()).setScale(2,
RoundingMode.HALF_UP);
         IndicatorDto indicatorDto = IndicatorDto.of(wightedIndicator.getIndicator());
         IndicatorRating indicatorRating = new IndicatorRating(i + 1, weightedAmount,
```

```
indicatorDto);
            indicatorRatingList.add(indicatorRating);
        }
        return indicatorRatingList;
    }

    private Function<Indicator, WightedIndicator> toFormulaWightedIndicator(String formula,
List<String> ids) {
        return i -> {
            validateIndicatorByDefaultRatingFields(i, ids);
            JSONObject jsonObject = converter.createJSONObject(i);
            Map<String, Double> variables = new HashMap<>();
            ids.forEach(fieldId -> {
                String value = getDataFromField(jsonObject, fieldId).toString();
                variables.put(fieldId, Double.valueOf(value));
            });
            Expression expression = new ExpressionBuilder(formula)
                    .variables(ids.toArray(new String[0]))
                    .build()
                    .setVariables(variables);
            double result = expression.evaluate();
            return new WightedIndicator(result, i);
        };
    }

    private List<WightedIndicator> getWightedIndicators(List<Indicator> indicators,
Function<Indicator, WightedIndicator> mapper) {
        return indicators.stream()
                .map(mapper)

.sorted(Comparator.comparingDouble(WightedIndicator::getWightedAmount).reversed())
                .collect(Collectors.toList());
    }

    private Function<Indicator, WightedIndicator> toWightedIndicator() {
        return i -> {
            double wightedAmount = calculateDefaultSystemWightedAmount(i);
            return new WightedIndicator(wightedAmount, i);
        };
    }

    private double calculateDefaultSystemWightedAmount(Indicator i) {
        List<String> fieldsIds = List.of(
                "shareOfPublicationsWithForeignAuthorsPercentage",
                "foreignResearchersNumber",
                "daysUsedByForeignResearchersNumber",
                "manDaysPerYear",
                "shareScientistsInvolved",
                "shareScientistsSecondedPercentage",
```

```
                "foreignFinancingPercentage",
                "fundingEfficiency",
                "organizationFundingSharePercentage",
                "shareInternationalPersonnelSelectionPercentage",
                "shareOfForeignExpert");
        validateIndicatorByDefaultRatingFields(i, fieldsIds);

        double pms = 0.8 * i.getShareOfPublicationsWithForeignAuthorsPercentage();
        double pdi = 0.2 * i.getForeignResearchersNumber() *
    i.getDaysUsedByForeignResearchersNumber() / i.getManDaysPerYear();
        double internationalScientificCooperation = pms + pdi;

        double pzn = 0.5 * i.getShareScientistsInvolved();
        double pmm = 0.5 * i.getShareScientistsSecondedPercentage();
        double internationalResearchEnvironmentIntegration = pzn + pmm;

        double pzf = 0.35 * i.getForeignFinancingPercentage();
        double pef = 0.3 * i.getFundingEfficiency();
        double psp = 0.35 * i.getOrganizationFundingSharePercentage();
        double financialComponent = pzf + pef + psp;

        double pmz = 0.5 * i.getShareInternationalPersonnelSelectionPercentage();
        double pmo = 0.5 * i.getShareOfForeignExpert();
        double internationalOrientationOfAdministration = pmz + pmo;

        return 0.25 * internationalScientificCooperation +
            0.25 * internationalResearchEnvironmentIntegration +
            0.25 * financialComponent +
            0.25 * internationalOrientationOfAdministration;
      }

      private void validateIndicatorByDefaultRatingFields(Indicator i, List<String> fieldsIds) {
        Map<String, FieldMetadata> fieldMetadataMap =
    restService.findMetadataByFieldsIds(fieldsIds);

        JSONObject jsonObject = converter.createJSONObject(i);
        fieldsIds.forEach(fieldId -> {
          String value = getDataFromField(jsonObject, fieldId).toString();
          FieldMetadata metadata = fieldMetadataMap.get(fieldId);
          validator.validateFieldByMetadata(fieldId, value, metadata);
        });
      }
}
```

ДОДАТОК В

Управління представленнями гетерогенних даних

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР72259_21Б 13-1

Аркушів 10

Київ — 2021

АНОТАЦІЯ

Додаток містить опис системи, що виконує керування представленнями гетерогенних даних, а саме системи для створення рейтингів, як системі що містить в собі дані, що можуть змінювати свою структуру з часом. За допомогою системи можна формувати рейтинги науково-технічних установ, як за запропонованим алгоритмом так і за альтернативними алгоритмами від користувача. Система виконує такі завдання:

- керування індикаторами
- керування семантикою полів
- отримання переліку використаних полів індикаторів
- динамічна вибірка даних індикаторів
- вивантаження даних про індикатори у файл

Система була розроблена мовою програмування Javas з використанням сімейств фреймворків Spring, бази даних MongoDB та бібліотеки GraphQL.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ 68
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 69
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ 70
4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ 71
5. ВИКЛИК І ЗАВАНТАЖЕННЯ 72
6. ВХІДНІ ДАНІ 73
7. ВИХІДНІ ДАНІ 74

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів системи управління представленнями гетерогенних даних, що виконує деякі завдання, визначенні в розділі 1. Додаток Б містить програмний код цих компонентів.

Для роботи кліентської частини потребує браузер, з підтримкою JavaScript.

Для роботу серверної частини, необхідний ПК або віртуальна машина з тактовою частотою процессора більше 1.5 Гц, та оперативною пам’яттю 1 ГБ.

Система розроблена за допомогою мови програмування Java у середовищі програмування IntellijIDEA.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблена система надає користувачу наступний функціонал:

- керування індикаторами
- керування семантикою полів
- отримання переліку використаних полів індикаторів
- динамічна вибірка даних індикаторів
- вивантаження даних про індикатори у файл

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, система складається з трьох модулів: агрегаційний модуль, модуль рейтингів та адміністративний модуль.

Агрегаційний модуль відповідає за показники науково-технічних установ, операції над ними, керування представленнями та обмеженнями даних.

Модуль рейтингів, будує рейтинги, як запропонований системою так і за алгоритмом від користувача.

Адміністративний модуль збирає інформацію щодо роботи системи та використання системою ресурсів машини.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання системи, користувач повинен мати персональний комп’ютер з тактовою частотою процесора більше 1.5 Гц, та оперативною пам’яттю не менше 1 ГБ. На комп’ютері повинна бути встановлена Java та MongoDB.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Вихідний код програми наведений на GitHub, та доступна за посиланням https://github.com/RuslanKovshar/Heterogeneous

Для встановлення та запуску програми необхідно виконати інструкцій наведені у пункті 5.3.

ВХІДНІ ДАНІ

Вхідна інформація для системи:

а) показники з певних напрямків діяльності науково-технічних установ

ВИХІДНІ ДАНІ

Вихідна інформація системи

- представлення даних
- рейтинг установ