НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

"На правах рукопису"
УДК 004.4

«До захисту допущено»
В.о. зав.кафедри
_____ Олександр КОВАЛЬ
"___" _____202_ р.

Магістерська дисертація

За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення
на тему: Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

Виконав: студент 2 курсу, групи ТВ-12мп
Макаренко Антон Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник д.т.н., доцент Коваль О. В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент к.т.н. Сенченко В. Р. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
в.о. зав. кафедри
_____ Олександр КОВАЛЬ
(підпис)
«____» _____ 202_р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Макаренку Антону Олександровичу
(прізвище, ім'я, по батькові)
1. Тема дисертації: Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері
Науковий керівник д.т.н., доцент Коваль О. В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "_7_" _листопада_ 2022 року №4067-с
2. Строк подання студентом дисертації 5 грудня 2022 року
3. Вихідні дані до роботи обчислювальна система, список наукометричних показників, обчислювальна система із можливістю подальшого розширення функціоналу
4. Перелік питань, які потрібно розробити проаналізувати схему зберігання необхідних даних у базі даних, розробити архітектуру веб-сервісу, інтегрувати його у загальну обчислювальну систему
5. Орієнтований перелік ілюстративного матеріалу архітектура веб-сервісу, архітектура засобу контейнеризації Docker, архітектура оркестратора контейнерів Kubernetes, результати тестування та розгортання веб-сервісу
6. Дата видачі завдання «_1_» _жовтня_ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.10.2021	виконано
2	Дослідження предметної області	01.10.2021 - 01.11.2021	виконано
3	Постановка вимог до проєктування системи	01.11.2021 - 15.11.2021	виконано
4	Дослідження існуючих рішень	15.11.2021 - 01.12.2021	виконано
5	Підготовка публікацій	01.12.2021 - 09.12.2021	виконано
6	Розробка програмного продукту	05.03.2022 - 07.10.2022	виконано
7	Тестування	07.10.2022 - 11.11.2022	виконано
8	Захист програмного продукту	17.10-2022 – 21.10.2022	виконано
9	Підготовка магістерської дисертації	22.10.2022 – 20.11.2022	виконано
10	Передзахист	21.11.2022 - 25.11.2022	виконано
11	Захист	19.12.2022 – 23.12.2022	виконано

Студент _____ _Макаренко А. О.

(підпис) (прізвище та ініціали)

Науковий керівник _____ _Коваль О. В._____

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Актуальність теми. Актуальність теми дисертації полягає у необхідності оцінки рівня ефективності наукової діяльності за допомогою показників рівня її міжнародної взаємодії у науково-технічній сфері.

Мета роботи і завдання дослідження. Метою роботи є дослідження показників рівня міжнародного співробітництва та реалізація веб-сервісу для їхнього обчислення**.**

Об'єкт дослідження. Інформаційні технології для побудови, розгортання та підтримки веб-сервісів.

Предмет дослідження. Веб-сервіс обчислення показників рівня міжнародної взаємодії наукової організації в науково-технічній сфері.

Наукова новизна одержаних результатів. Розроблений веб-сервіс, що проводить обчислення показників рівня міжнародної взаємодії наукової організації у науково-технічній сфері. Веб-сервіс був створений за допомогою сучасних програмних інструментів та розгорнутий у хмарі, що дозволяє користуватися ним із будь-якого місця та у будь-який час.

Практичне значення одержаних результатів. Результати проведеної роботи представлено у вигляді готової обчислювальної системи, здатної надати необхідні для користувача дані про показники міжнародної взаємодії наукової установи.

Структура та обсяг магістерської роботи. Магістерська дисертація складається зі вступу, п'яти розділів, висновку, переліку посилань з 56 найменувань, п'яти додатків. Дисертація містить 16 рисунків, 15 таблиць. Повний обсяг магістерської дисертації складає 130 сторінок.

Ключові слова: веб-сервіс, база даних, наукові показники, обчислювальна система, обробка даних.

ABSTRACT

Topic's relevance. The relevance of the topic of the dissertation lies in the need to assess the level of effectiveness of scientific activity using indicators of the level of its international interaction in the scientific and technical sphere.

The aim of the research. The aim of the research is to study indicators of the level of international cooperation and implement a web service for their calculation.

The object of research. Information technologies for building, deploying and maintaining web services.

The subject of research. Web service for calculating indicators of the level of international interaction of a scientific organization in the scientific and technical sphere.

Scientific innovation. A web service was developed for calculation of indicators of the level of international interaction of a scientific organization in the scientific and technical sphere. The service was created using modern software tools and deployed in the cloud, which allows users to use it from anywhere and at any time.

Practical value. The results of the work are presented in the form of a ready-made computer system capable of providing the user with the necessary data on the indicators of the scientific institution's international interaction.

Structure and scope of the master's thesis. The master's thesis consists of an introduction, five chapters, a conclusion, a list of references from 56 titles, five appendices. The thesis contains 16 figures, 15 tables. The full volume of the master's thesis is 130 pages.

Keywords: web service, database, scientific indicators, computing system, data processing.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

API – application programming interface, спосіб взаємодії користувача із програмною системою

МСНО – міжнародне співробітництво наукової організації / наукових організацій

НД – наукова діяльність

ІЦ – індекс цитування

h-index – індекс Гірша

ІФ – імпакт-фактор

SNIP – Source-Normalized Impact per Paper

SJR – SCIMago Journal Ranking

НТС – науково-технічна сфера

МНТС – міжнародне науково-технічне співробітництво

WoS – Web of Science

GS – Google Scholar

URI – uniform resource identifier, уніфікований ідентифікатор ресурсу

URL – uniform resource locator, уніфікований локатор ресурсу

IoC – inversion of control, принцип побудови програми, при якому її частини отримують потік керування (викликаються) із загальної спільновикористовуваної бібліотеки

DI – dependency injection, шаблон проектування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи IoC отримання залежностей

MVC – Model-View-Controller, архітектурний шаблон для проектування додатків

ORM – object-relational mapping, технологія програмування, яка зв'язує бази даних з концепціями об’єктно-орієнтованих мов програмування

JMS – Java messaging service

POJO – plain old Java object

MS – Microsoft

SQL – structured query language, декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними базами даних

БД – база даних / бази даних

СУБД – система управління базою даних / базами даних

MSSQL – Microsoft SQL Server

ETL – Extract, Transform, Load

CLI – command line interface

AZ – active zone

JSON – JavaScript Object Notation, текстовий формат обміну даними між комп'ютерами

EKS – Elastic Kubernetes Service

IAM – Identity and Access Management

AWS CLI – Amazon Web Services Command Line Interface, програма, що надає доступ до інструментів AWS через командний рядок

ВСТУП

Наука – це одна зі сфер діяльності людства, що полягає у отриманні нових знань про оточуючий людей світ. Наукова діяльність відбувається за допомогою процесів збирання, систематизації, оновлення, аналізу нових та уже існуючих фактів, синтезу нових знань, що дозволяють передбачати перебіг подій та будувати між ними причинно-наслідкові зв'язки.

Вчений – представник науки, який здійснює діяльність з формування наукової картини світу, чия наукова діяльність та кваліфікація у тій чи іншій формі отримали визнання з боку наукової спільноти. Основною формальною ознакою визнання кваліфікації є публікація матеріалів досліджень в авторитетних наукових виданнях та доповіді на авторитетних наукових конференціях.

Вчені формують наукові товариства, які відіграють важливу роль для розвитку науки. Найбільш активна роль належить тим товариствам, основним завданням яких є обмін інформацією про проведені дослідження та отримані результати за допомогою наукових конференцій, публікацій у періодичних виданнях тощо.

Наукові дослідження часто фінансуються через конкурсний процес, у якому оцінюються потенційні дослідницькі проекти. При тому такі проекти можуть реалізовуватися як однією установою, так і групою установ, що взаємодіють одна із одною. Рівень того, наскільки взаємодія тої чи іншої установи є високою, визначає те, наскільки ефективно установа зможе працювати над проектом і, відповідно, наскільки високі шанси у проекту пройти конкурсний відбір.

Серед можливих типів взаємодії варто окремо виділити міжнародну взаємодію. Міжнародна взаємодія є складнішою, аніж взаємодія вітчизняних установ або внутрішня взаємодія співробітників установи між собою. Різні стандарти роботи, різна мова можуть мати великий вплив на те, як співробітники різних установ, розташованих у різних країнах, взаємодіють одне із одним. З іншого боку, саме міжнародна взаємодія надає найбільший спектр можливостей

для наукових організацій. Установи із різних країн мають різний науковий досвід, різну демографію, різні підходи до проведення досліджень. Усе це дозволяє проводити наукову діяльність із максимальною користю для кожного, хто приймає участь у міжнародній науковій взаємодії.

Магістерська дисертація складається із п’яти розділів. У першому розділі описано задачу обчислення показників рівня міжнародної взаємодії організації у науково-технічній сфері, а також проаналізовано існуючі рішення для поставленого завдання. Другий розділ розглядає та описує обрані засоби розробки для створюваного веб-сервісу. Третій розділ наводить опис програмної реалізації та розгортання компонентів веб-сервісу. Четвертий розділ містить системні вимоги для запуску програмного продукту та демонструє його роботу. У п’ятому розділі наведено ідею та стратегію розвитку створеної системи як стартап-проекту.

Завдяки створеному веб-сервісу можливо визначити те, яким є рівень міжнародної взаємодії організації у науково-технічній сфері.

1 ЗАДАЧА ОБЧИСЛЕННЯ ПОКАЗНИКІВ РІВНЯ МІЖНАРОДНОГО СПІВРОБІТНИЦТВА

1.1 Постановка задачі обчислення показників рівня міжнародного співробітництва

Світова спільнота прагне постійного розвитку, використовуючи для цього всі можливі засоби. Ні одна науково-технічна організація світу не здатна досягти вищого рівня розвитку використовуючи ресурси винятково власної країни. Для подолання цієї проблеми виникла потреба у об'єднанні зусиль науково-технічних організацій із різних країн.

Причини для міжнародного співробітництва наукових організацій у науково-технічній сфері можна розділити на внутрішні та зовнішні [1, с. 6]. Внутрішньою причиною є підвищення якості науки. Це стосується як наукових досліджень, так і академічного навчання, яке можуть проводити наукові організації. Не менш важливими є зовнішні причини, що стосуються глобальних наукових проблем та викликів. Складність багатьох із них вимагає комплексного мультидисциплінарного наукового підходу.

Метою магістерської роботи є реалізація системи обчислення показників рівня міжнародного співробітництва наукової організації у науково-технічній сфері. Для досягнення зазначеної мети було поставлено наступні задачі:

- виконати аналіз переваг та недоліків міжнародного науково-технічного співробітництва
- дослідити існуючі наукометричні показники
- проаналізувати існуючі системи обчислення показників рівня міжнародного співробітництва
- визначити які додаткові показники можуть бути використані для обчислення показників рівня міжнародного співробітництва

- розробити веб-сервіс для обчислення показників рівня міжнародного співробітництва

Вхідні дані для системи надаються із окремо створеної реляційної бази даних. До вхідних даних належить інформація про науковців, що працюють у тій чи іншій організації, про проведені конференції, наявні публікації, записи про науковців організації у інших системах обчислення показників.

Створений веб-сервіс повинен обчислювати визначені показники МНТС для організації чи науковця, використовуючи вищеописані вхідні дані, та надавати зручний для користувач API.

1.2 Переваги та недоліки міжнародного співробітництва у науково-технічній сфері

МНТС має власні переваги та недоліки. До переваг міжнародного співробітництва можна віднести те, що воно покращує не лише якість самих досліджень, а і можливості їх практичного застосування. На щастя для світової спільноти, рівень МСНО у НТС підвищується. Згідно із джерелом [1, с. 7], одинадцять із тринадцяти досліджених країн продемонстрували підвищення рівня міжнародної взаємодії у період між 2004 та 2011 роками.

Недоліки МНТС пов’язані із наступними ризиками [1, с. 7]:

- погіршення наукової якості
- зловживання науковими знаннями

Погіршення наукової якості може виникнути через вимоги до публікацій робіт. Чим більше робіт необхідно публікувати для учасників МНТС, тим менше часу для ретельного аналізу окремої роботи і тим більший ризик погіршення її якості.

Зловживання науковими знаннями пов’язане із взаємодією із організаціями чи країнами, що мають сумнівні наміри та/або репутацію. Це створює наступні ризики:

• Сприяння економічному, промисловому та військовому шпигунству. Науковці не завжди усвідомлюють з якою метою або до яких наслідків може призвести їх праця та якими методами будуть розповсюджуватися їх роботи, окрім публікацій та конференцій.

• Пряма чи непряма взаємодія із країнами, що знаходяться під міжнародними санкціями. Так, компоненти іранських безпілотних літальних апаратів містять компоненти, вироблені Китаєм, Німеччиною, Сполученими Штатами, Сполученим Королівством, Австрією, Японією, Швецією, Польщею та Україною [2, 3].

• Можливість шантажу науковців, що може призвести до компроментації чутливої інформації, розкриття секретних чи конфіденційних файлів.

• Можливість прямої чи непрямої підтримки неприйнятних для людей умов, таких як співпраця із країнами із репресивним режимом.

1.3 Аналіз існуючих наукометричних показників

Наукометрія – це наука, що займається статистичним дослідження структури і динаміки наукової інформації [4]. Відповідно, наукометричні показники є тими статистичними даними, що описують структуру і динаміку наукової інформації.

Наукометричні показники можна класифікувати наступним чином [4]:

• кількісні показники НД

• якісні показники впливу публікацій на інформаційний масив

• комплексні показники, які враховують кількісні та якісні критерії оцінки наукової діяльності

Ключовими показниками для визначення результативності наукової діяльності є показники публікаційної активності вченого чи організації та рейтингові показники періодичних видань, у яких публікуються роботи.

До показників публікаційної активності відносяться:

• загальна кількість публікацій

• індекс цитування публікацій

- індекс Гірша

До рейтингових показників періодичних видань відносяться:

- імпакт-фактор
- Source-Normalized Impact per Paper
- SCIMago Journal Ranking

Загальна кількість публікацій є найбільш узагальненим наукометричним показником. Він показує те, яка кількість опублікованих робіт була проіндексована системою.

Індекс цитування є ключовим показником, створеним Інститутом наукової інформації, що широко використовується у всьому світі [5]. ІЦ показує скільки разів публікація певного автора була процитована у роботах інших авторів, випущених у певний рік. Індекс цитування є кількісним показником, тобто він показує лише кількість цитувань і не гарантує високої якості процитованої роботи. Однак чим вищий ІЦ у роботи, тим більшою популярністю вона користується серед науковців і, відповідно, тим більша ймовірність її високого рівня її якості.

Індекс Гірша обчислюється на основі розподілу цитувань робіт автора. Вчений має індекс h, якщо h із його чи її N_p робіт мають по h чи більше цитувань, а інші (N_p - h) робіт мають не більше h цитувань кожна [6]. Іншими словами, щоб мати індекс h вченому необхідно мати h робіт, кожна із яких процитована щонайменше h разів. Якщо вчений опублікував 10 робіт, на кожну із яких посилалися по одному разу, то його h-індекс становить 1. З іншого боку, якщо у вченого 5 робіт, на кожну із яких посилалися по 5 разів, то його h-індекс становить 5.

Перевагою індексу Гірша є те, що він дає приблизно однаковий результат як для вченого із великою кількістю малоцитованих робіт, так і для науковця із малою кількістю популярних робіт. Високим h-індекс є для тих наукових співробітників, які регулярно роблять нові публікації та чиї роботи часто цитуються іншими науковими співробітниками.

Індекс Гірша не завжди здатен дати адекватну оцінку значущості вченого. Якщо кар'єра вченого через ті чи інші причини була короткою, його h-індекс завжди буде малим, не залежно від того, наскільки значущими є його роботи. Так, французький математик Еварист Галуа назавжди залишиться із індексом 2 не зважаючи на вплив його робіт на сучасну математику.

Імпакт-фактор – частота, з якою середня стаття в журналі цитувалася в певний рік. ІФ використовується для вимірювання важливості чи рейтингу журналу шляхом підрахунку кількості цитувань його статей. Розрахунок базується на дворічному періоді та передбачає ділення кількості цитувань статей на кількість статей, які потенційно можуть бути процитованими [7].

Source-Normalized Impact per Paper (нормалізований вплив джерела на статтю) є складною метрикою, яка враховує відмінності у практиці цитування у окремих галузях шляхом порівняння цитувань кожного журналу на публікацію із потенціалом цитування у його галузі. Потенціал цитування визначається як набір публікацій, які цитують цей журнал. Таким чином, SNIP вимірює вплив контекстуального цитування та дає можливість прямого порівняння журналів у різних предметних галузях, оскільки цінність одного цитування є більшою для журналів у галузях, де цитування є менш імовірними, і навпаки [8].

SCIMago Journal Ranking є рейтингом, що обчислюється на основі інформації у системі Scopus. SJR є аналогічним до ІФ показником. Журнал наділяє власним «престижем», або статусом, інші журнали, цитуючи опубліковані в них матеріали. Фактично це означає, що цитата з джерела з відносно високим показником SJR має більшу цінність, ніж цитата з джерела з нижчим показником SJR.

1.4 Аналіз існуючих систем обчислення показників рівня міжнародного співробітництва

При аналізі існуючих систем обчислення показників рівня міжнародного співробітництва варто у першу чергу згадати наукометричні бази даних.

Наукометрична база даних – це бібліографічна і реферативна база даних з інструментами для відстеження цитованості статей, опублікованих у наукових виданнях [9]. Високий індекс цитування вчених, що працюють у певній науково-технічній організації, говорить про високий рівень ефективності та результативності організації у цілому.

До світових наукометричних систем належать наступні [10, 11]:

- Scopus
- Web of Science
- PubMed
- ERIC
- IEEE Xplore
- ScienceDirect
- Directory of Open Access Journals (DOAJ)
- Google Scholar
- Index Copernicus (IC)

Більш повний список систем, що займаються обчисленням наукометричних показників, можна знайти за джерелом [12].

1.4.1 База даних Scopus

Scopus – це реферативна база даних, що містить рецензовану літературу, таку як наукові журнали, книги та матеріали конференцій. Scopus надає повний огляд світових досліджень у сферах науки, технологій, медицини, соціальних наук, мистецтва та гуманітарних наук. Призначений Scopus для широкого кола користувачів: від науковців, що прагнуть наукових проривів, до науково-технічних установ і державних організацій, що займаються оцінкою та аналізом досліджень [13].

Scopus є одною із найбільших наукометричних баз даних світу. По всьому світу Scopus використовують понад 3000 академічних, державних і корпоративних

установ [13]. База даних містить близько 1,8 мільярда цитованих посилань, 17 мільйонів профілів, 335 тисячі книг та книжкових серій, 7 тисяч видавців та 94 тисячі зв'язаних профілів [14].

Для того, щоб мати видимий профіль, автору, присутньому у системі Scopus, необхідно мати дві та більше публікації. Якщо публікація у автора лише одна, його чи її профіль є прихованим. Самі профілі надають наступну інформацію [15]:

- варіанти імені автора
- перелік місць роботи автора
- кількість публікацій автора
- роки публікаційної активності автора
- галузі досліджень автора
- посилання на основних співавторів
- загальна кількість цитувань публікацій автора
- загальна кількість джерел, на які посилається автор
- індекс Гірша автора тощо

Для перегляду авторів із однією публікацією користувачам бази даних необхідно користуватися підпискою. Крім того, кількість результатів пошуку обмежена до 20 для користувачів, що безкоштовно користуються Scopus.

Scopus самостійно не об'єднує показники автора, якщо відрізняється автор має різну транслітерацію імені чи місце роботи. Для виправлення даної проблеми автору необхідно самостійно зробити об'єднання записів за допомогою процедури, описаної у [16].

1.4.2 Платформа Web of Science

Web of Science – наукометрична платформа, яка надає доступ до даних про посилання та цитування наукових робіт із періодичних наукових видань, конференцій та інших наукових робіт із різних дисциплін.

Web of Science містить 1,5 млрд посилань на публікації, 74,8 млн записів, із яких 10,1 млн із відкритим доступом, інформацію про більш ніж 21 тисячу міжнародних наукових журналів із 254 дисциплін [17].

WoS є не просто каталогом наукових видань. Це вибіркова, структурована та збалансована база даних із повними посиланнями на цитування та розширеними метаданими, яка підтримує широкий спектр інформаційних цілей.

Можна виділити два наступних компоненти WoS: Core Collection та Platform. Їх порівняння наведено у таблиці 1.1 [18].

Таблиця 1.1 – Порівняння Core Collection та Platform

	Core Collection	Platform
Опис	Індекси цитування, що представляють зв'язки між науковими дослідницькими статтями, знайденими у всесвітньо значущих журналах, книгах і роботах у галузі науки, соціальних наук, мистецтва та гуманітарних наук. Основна колекція WoS — це стандартний набір даних, що лежить в основі показників впливу на журнал, які можна знайти в Journal Citation Reports, і показників інституційної ефективності, які можна знайти в InCites.	Платформа, що надає доступ до міждисциплінарних і регіональних індексів цитування, індексів спеціалізованих предметів, індексу патентної родини та індексу наборів наукових даних. WoS забезпечує загальну мову пошуку, середовище навігації та структуру даних, що дозволяє дослідникам здійснювати широкий пошук у різних ресурсах і використовувати зв'язки цитування для переходу до відповідних результатів дослідження.

Продовження таблиці 1.1

Кількість видань	Більше 20 900 видань + інформація про наукові книги та конференції.	Більше 34 200 видань + інформація про наукові книги, конференції та набори даних.
Покриття	• Понад 75 мільйонів записів • Більше 101 000 книг Понад 8 мільйонів конференцій	• 155 мільйонів записів • 39,3 мільйонів патентних родин (більше 70 мільйонів патентів) 7,3 мільйона наборів даних
Охоплені ресурси	• Індекс наукового цитування • Індекс цитування соціальних наук • Індекс цитування гуманітарних наук • Індекс цитування матеріалів конференції • Індекс цитування книги • Індекс цитування нових джерел	Core Collection + наступні: • Індекс цитування BIOSIS • База даних китайського наукового цитування • Індекс цитування SciELO • Індекс цитування даних • CABI: Тези CAB і глобальне здоров'я • FSTA – науковий ресурс про харчові продукти • Medline • KCI – база даних корейських журналів • Зоологічні записи

Продовження таблиці 1.1

Охоплений період часу	• Наука: 1900-сучасність • Соціальні науки: 1900-сучасність • Гуманітарні науки: 1975-сучасність • Праці: 1990-сучасність • Книги: 2005-сучасність • Індекс цитування нових джерел: 2005-сучасність	• Журнальна література: 1800-сучасність • Патенти: 1963-сучасніть • Повне індексування цитованих посилань для всього вмісту WoS Core Collection • Індекс цитування для SciELO, BISOS • Весь вміст включає час цитування з WoS Core Collection і платформи Citation Sources
Індексація авторів	Індексуються всі автори з усіх публікацій.	• WoS Core Collection: усі автори індексуються для всіх публікацій. • Інші ресурси: індексування авторів залежить від ресурсу.
Частота оновлення	Щодня (з понеділка по п’ятницю).	Кожен набір даних оновлюється за власним графіком, від щоденного до щомісячного.
Індексація установ	Усі автори, приналежні до установ, індексуються. Варіанти закладів і батьківські й дочірні відображається та пов’язуються з назвою закладу.	Індексація приналежних до установ авторів залежить від аналізованого набору даних.

Продовження таблиці 1.1

Аналіз цитування	Підрахунок цитувань і обчислення h-індексу авторів. «Гарячі» та «Чисто цитовані» статті (статті у верхніх процентилях відповідно до року, галузі та типів документів) доступні з інтеграції основних наукових показників. Фактори впливу журналу та квартилі ефективності журналу доступні через інтеграцію звітів про цитування журналу.	Підрахунок цитувань і обчислення h-індексу авторів. «Гарячі» та «Чисто цитовані» статті (статті у верхніх процентилях відповідно до року, галузі та типів документів) доступні з інтеграції основних наукових показників.
Керований словник	Поля ключових слів включають ключові слова автора та «Ключові слова плюс», які витягуються із заголовків цитованих статей. Контрольоване індексування забезпечується для приналежності закладів (батьківське/дочірнє відображення).	Контрольований словниковий пошук надається для Medline, Inspec, FSTA, BIOSIS, Zoological Record.

1.4.3 Пошукова система Google Scholar

Google Scholar або Google Академія – це пошукова система, що призначена для зручного пошуку та індексації широкого спектру наукової літератури. GS є зручний тим, що надає можливість централізованого пошуку широкого спектру наукових робіт: статті, тези, книги, реферати, судові рішення від академічних

видань, професійних спільнот, онлайн репозиторіїв, університетів та веб-сайтів [19].

Google Scholar оцінює документи за стандартами наукової спільноти, аналізуючи весь текст кожного документу, коли він був опублікований, хто його автор, а також те, коли востаннє документ був процитований у іншій науковій літературі.

Google Scholar дозволяє користувачам шукати цифрові або фізичні копії статей. GS індексує повні тексти журнальних статей, технічні звіти, препринти, дисертації, книги та інші документи, включаючи вибрані веб-сторінки. Оскільки багато результатів пошуку Google Scholar посилаються на комерційні видання, більшість користувачів матимуть доступ лише до анотації та деталей цитування статті. Повний доступ до статті буде платним. Найрелевантніші результати для ключових слів, для яких ведеться пошук, будуть перераховані першими в порядку рейтингу автора, кількості посилань, пов’язаних із ним, і їх відповідності іншій науковій літературі, а також рейтингу публікації, у якій опубліковано журнал.

Недоліком Google Scholar є відсутність даних про його охоплення. Частина наукових видавництв не дозволяє GS індексувати власні публікації. Наприклад, публікації від журналу Elsevier не були включені до індексування системою Google Scholar до 2007 року, коли журнал зробив частину вмісту ScienceDirect доступним для GS [20].

Перевагою системи є її відкритість та доступність. Якщо користувачеві необхідно отримати базову інформацію про автора чи його роботи, то GS надає усе для цього необхідне. Якщо ж потрібні більш детальні дані, то необхідно мати або комерційну підписку на ресурси, із яких бере інформацію Google Scholar, або скористатися іншими системами, такими як Scopus або Web of Science.

1.5 Додаткові показники рівня міжнародного співробітництва

Будуючи власну систему обчислення показників МСНО не варто нехтувати тим, що уже є зробленим. Так, нова система може вираховувати те, скільки у автора тієї чи іншої організації публікацій у наукометричних базах даних, таких як Web of Science, Scopus, Google Scholar тощо. Звідти можна брати й додаткову інформацію, таку як цитування та інші параметри.

Не менш важливими є кількість міжнародних подій, у яких приймали участь науковці організації. До міжнародних наукових подій можна віднести конференції, де була представлена інформація про завершені роботи науковців або проміжні результати наукових проектів організації. Також можна вираховувати кількість опублікованих для подій тез або кількість проведених науковими співробітниками виступів.

Окрім безпосередньо наукової діяльності, наукова установа може мати освітні програми, із яких також можна брати показники. Такими показниками можуть бути кількість захищених дисертацій (магістерських, кандидатських, докторських) та участь студентів наукової організації у міжнародних подіях.

Висновки до розділу 1

У даному розділі була поставлена задача до дипломної роботи. Було визначено які основні переваги та недоліки МСНО, проаналізовано існуючі системи обчислення рівня МСНО та визначено які показники можуть бути використані для виконання задачі дипломної роботи.

Перевагою МСНО є можливість покращення якості досліджень та розширення спектру їх можливого застосування. До недоліків можна віднести потенційне погіршення наукової якості за наявності кількісних, але не якісних вимог до виконання робіт та свідомий чи несвідомий ризик роботи над небезпечними чи загрозливими проектами.

Систем, що обчислюють рівень показників для МНТС, є велика кількість. До поширених можна віднести Scopus, Web of Science, Google Scholar. Кожна із систем має свій функціонал, своє наповнення та власне обчислення показників.

Додатковим показниками, які можна обчислити на основі наявних даних, є кількість робіт у наукометричних базах даних, кількість міжнародних подій, у яких приймали участь науковці організації, кількість захищених дисертацій та кількість міжнародних подій, у яких приймали участь студенти наукової організації.

2 МЕТОДИ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ ОБЧИСЛЕННЯ ПОКАЗНИКІВ РІВНЯ МІЖНАРОДНОГО СПІВРОБІТНИЦТВА

2.1 API сервер

Для побудови веб-сервісу необхідно, перш за все, створити API-сервер. API використовується для надання доступу до сервісів чи інформації програми за допомогою набору заздалегідь створених ресурсів, таких як методи, об’єкти, URI. Використовуючи дані ресурси, інші додатки мають змогу отримати доступ до даних чи сервісів без необхідності реалізовувати специфічні об’єкти чи процедури. API є важливим елементом для багатьох сучасних архітектур програмного забезпечення, оскільки він надає високорівневі абстракції, які полегшують виконання програмних задач, підтримують дизайни розподілених та модульних систем та сприяють повторному використанню уже написаного коду [21].

Для розробки веб-сервісів та побудови необхідного API є безліч інструментів, доступних сучасному розробнику. Серед прикладів можна назвати Django, що базується на мові програмування Python і дозволяє створювати комплексні веб-додатки. Недоліком даного інструменту є його невисока швидкодія [22]. Іншим прикладом є Actix Web, що базується на сучасній низькорівневій мові програмування Rust. Actix Web, так само, як і Django, дозволяє створювати комплексні системи, але через новітність та складність мови програмування Rust, Actix Web є доволі складним у опануванні і вимагає від розробника достатньо великого досвіду і гарного розуміння низькорівневого програмування. З іншого боку, Actix Web є одним із найшвидших із точки зору швидкодії програми інструментів, поступаючись лише інструментам на C++ та іншим фреймворкам на Rust [22].

Інструментом, що має вищу за Django швидкодію [22] та дозволяє створювати складні веб-додатки є Spring Framework, який базується на мові

програмування Java. Spring Boot, що є надбудовою над фреймворком Spring для спрощення його конфігурації і роботи з ним, використовується у 66% випадків як сервер додатків серед розробників на Java станом на 2021 рік [23]. Через високий рівень підтримки, наявність детальної документації та прийнятний рівень швидкодії Spring був обраний як інструмент розробки веб-сервісу.

2.1.1 Spring Framework

Як правило, Spring описують як полегшений інструментарій для створення додатків Java. Із цього твердження випливає два нюанси. По-перше, за допомогою Spring можна створювати будь-які додатки: автономні додатки, веб-сервіси, додатки промислового рівня. По-друге, полегшеність інструментарія означає не зменшену кількість класів або зайнятої пам'яті, а визначає основний принцип інструментарію Spring: мінімальний вплив. Spring є легким у тому значенні, що розробнику необхідно внести небагато змін у код програми, якщо вони взагалі необхідно. Якщо розробник вирішить припинити використання Spring, зробити це буде досить просто [24].

Особливості фреймворку Spring, такі як IoC, AOP і керування транзакціями, роблять його унікальним серед інших фреймворків. Нижче наведено деякі з найважливіших особливостей Spring [25]:

- Контейнер IoC – основний контейнер, який використовує шаблон DI або IoC для неявного надання посилання на об'єкт у класі під час виконання. Цей шаблон діє як альтернатива шаблону service locator. Контейнер IoC містить код асемблера, який керує конфігурацією об'єктів програми. Фреймворк Spring надає два пакети, а саме org.springframework.beans і org.springframework.context, які допомагають забезпечити функціональність контейнера IoC.

- Інструментарій доступу до даних. Дозволяє розробникам використовувати Persistence API, такі як JDBC і Hibernate, для роботи із базою даних. Це допомагає у вирішенні проблем для розробника, таких як взаємодія із підключенням до бази

даних; пересвідчення, що з'єднання завершене; робота із винятками; керування транзакціями. Інструментарій доступу до даних також дозволяє розробнику легко писати код для доступу до бази даних у будь-якій частині програмі.

• Фреймворк Spring MVC. Дозволяє створювати веб-додатки на основі архітектури MVC. Усі запити, зроблені користувачем, спочатку проходять через контролер, а потім надсилаються до різних представлень, тобто до різних сторінок JSP або сервлетів. Обробка форм і функції перевірки форми Spring MVC можна легко інтегрувати з усіма популярними технологіями перегляду, такими як ISP, Jasper Report, FreeMarker і Velocity.

• Керування транзакціями. Допомагає керувати транзакціями програми, не впливаючи на її код. Ця структура забезпечує Java Transaction API (JTA) для глобальних транзакцій, керованих сервером додатків, і локальних транзакцій, керованих за допомогою JDBC Hibernate, Java Data Objects (JDO) або інших API доступу до даних. Це дозволяє розробнику моделювати широкий спектр транзакцій на основі декларативного та програмного керування транзакціями Spring.

• Spring Web Service. Генерує кінцеві точки та визначення веб-служб на основі класів Java, якими, однак, важко керувати у програмі. Щоб вирішити цю проблему, Spring Web Service надає багаторівневі підходи, якими окремо керує синтаксичний аналіз розширюваної мови розмітки (XML). Spring забезпечує ефективне відображення для передачі вхідного запиту XML-повідомлення до об'єкта та розробника для легкого розподілу XML-повідомлення (об'єкта) між двома машинами.

• Рівень абстракції JDBC. Допомагає користувачам легко та ефективно виправляти помилки. Програмний код JDBC можна скоротити, якщо цей рівень абстракції реалізовано у веб-додатку. Усі винятки SQLException транслюються в клас DataAccessException. Виняток доступу до даних Spring не стосується JDBC, тому об'єкти доступу до даних (DAO) не прив'язані лише до JDBC.

• Фреймворк Spring TestContext. Надає засоби модульного та інтеграційного тестування для додатків Spring. Крім того, структура Spring TestContext надає

спеціальні функції інтеграційного тестування, такі як керування контекстом і кешування DI тестових фікстур, а також керування транзакційним тестуванням із семантикою відкату за замовчуванням.

Spring Framework складається з функцій, організованих у приблизно 20 модулів. Ці модулі згруповані в базовий контейнер, доступ до даних/інтеграцію, веб, AOP (аспектно-орієнтоване програмування), інструментарій і тест, як показано на рисунку 2.1 [26].

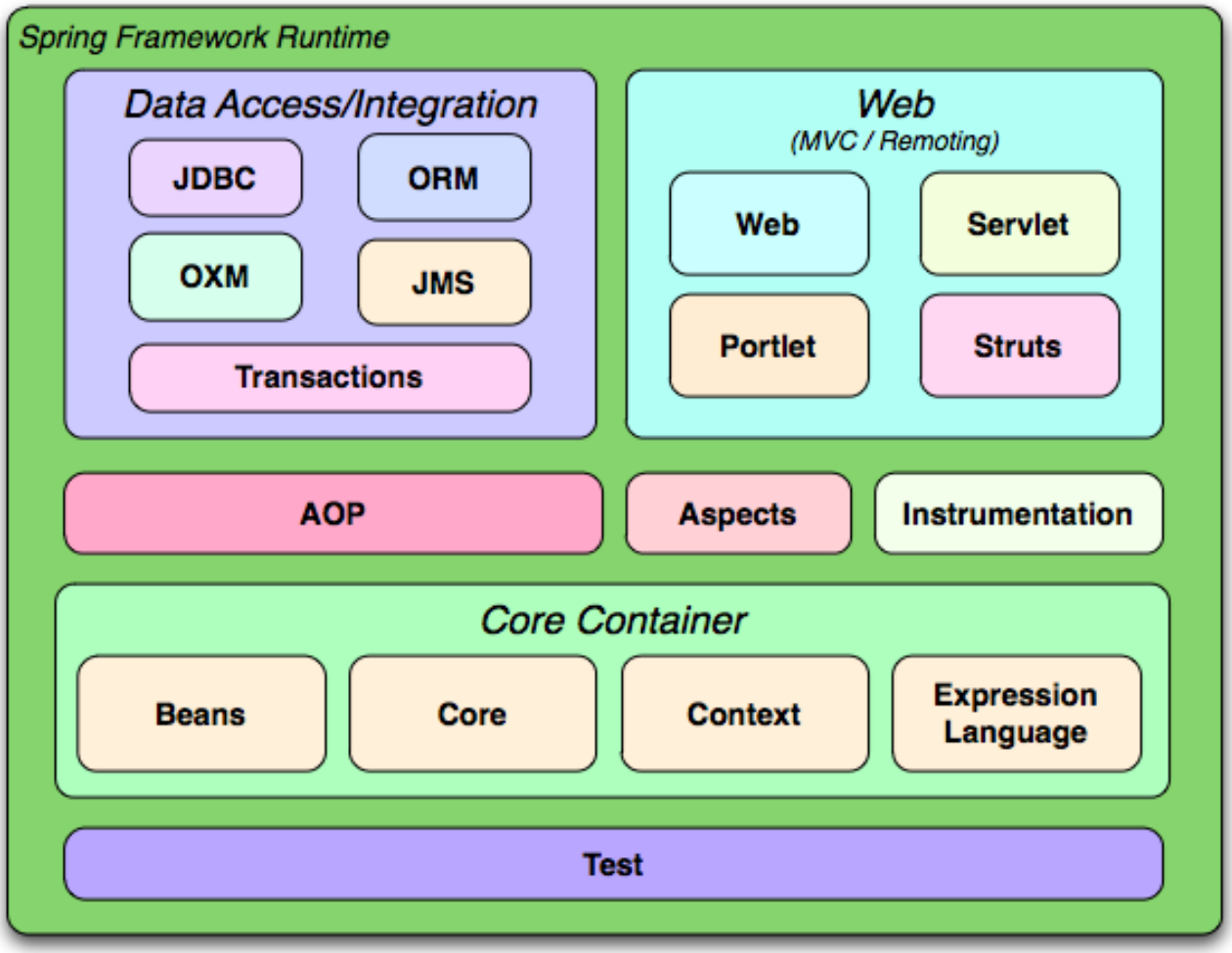


Рисунок 2.1 – Архітектура Spring Framework

Контейнер Core складається з модулів Core, Beans, Context і Expression Language. Модулі Core і Beans забезпечують фундаментальні частини фреймворку, включно з функціями IoC і Dependency Injection. BeanFactory – це складна

реалізація шаблону фабрики, що усуває потребу в програмних одиночках і дозволяє відокремити конфігурацію та специфікацію залежностей від фактичної логіки програми.

Модуль Context будується на основі, утвореній модулями Core і Beans. Модуль Context успадковує свої функції від модуля Beans і додає підтримку для інтернаціоналізації (з використанням, наприклад, пакетів ресурсів), розповсюдження подій, завантаження ресурсів і прозорого створення контекстів за допомогою, наприклад, контейнером сервлетів. Модуль Context також підтримує такі функції Java EE, як EJB, JMX і базове віддалене спілкування. Інтерфейс ApplicationContext є центром модуля Context.

Модуль Expression Language надає потужну мову виразів для запитів і маніпулювання графом об'єктів під час виконання. Мова підтримує налаштування та отримання значень властивостей, призначення властивостей, виклик методів, доступ до контексту масивів, колекцій та індексаторів, логічних і арифметичних операторів, іменованих змінних і отримання об'єктів за іменами з контейнера IoC Spring. Модуль також підтримує проектування списків, селекцію, а також загальні агрегації списків.

Рівень доступу до даних/інтеграції складається з модулів JDBC, ORM, OXM, JMS і Transaction.

Модуль JDBC надає рівень абстракції JDBC, який усуває потребу у монотонному кодуванню JDBC і роботою із кодами помилок, специфічних для вендора бази даних.

Модуль ORM забезпечує рівні інтеграції для популярних API об'єктно-реляційного відображення, такі як JPA, JDO, Hibernate та iBatis. Використовуючи пакет ORM, користувач може використовувати дані фреймворки у поєднанні з усіма іншими функціями, які пропонує Spring, такими як функція керування декларативними транзакціями.

Модуль OXM забезпечує рівень абстракції, який підтримує реалізацію відображення Object/XML для JAXB, Castor, XMLBeans, JiBX і XStream.

Модуль JMS містить функції для створення та споживання повідомлень.

Модуль Transaction підтримує програмне та декларативне керування транзакціями для класів, які реалізують спеціальні інтерфейси, і для всіх користувацьких POJO.

Веб-рівень складається з модулів Web, Web-Servlet, Web-Struts і Web-Portlet. Модуль Web надає базові функції веб-орієнтованої інтеграції, такі як функція завантаження файлів із кількома частинами та ініціалізація контейнера IoC за допомогою слухачів сервлетів і контексту веб-орієнтованої програми. Він також містить веб-частини підтримки віддаленого доступу Spring.

Модуль Web-Servlet містить реалізацію MVC Spring для веб-додатків. Spring MVC забезпечує чітке відокремлення між кодом моделі домену і веб-формами та інтегрується з усіма іншими функціями Spring Framework.

Модуль Web-Struts містить додаткові класи для інтеграції класичного веб-рівня Struts у програму Spring.Подібна підтримка є застарілою із виходом Spring версії 3.0 [26].

Модуль Web-Portlet забезпечує реалізацію MVC для використання в середовищі портлетів і є віддзеркаленням функціональності модуля Web-Servlet.

Модуль AOP від Spring забезпечує реалізацію аспектно-орієнтованого програмування, сумісну з AOP Alliance, що дозволяє визначати перехоплювачі методів і точки розрізів. Використовуючи функціональні можливості метаданих на рівні джерела можливо включити поведінкову інформацію у код програми у спосіб, подібний до атрибутів .NET. Окремий модуль Aspects забезпечує інтеграцію з AspectJ. Модуль Instrumentation забезпечує підтримку інструментів класу та реалізацію завантажувача класів для використання на певних серверах додатків.

Модуль Test підтримує тестування компонентів Spring за допомогою JUnit або TestNG. Він забезпечує послідовне завантаження Spring ApplicationContexts і кешування цих контекстів. Модуль також надає об’єкти mock, які можна використовувати для ізольованого тестування коду.

2.1.2 Spring Boot

Незважаючи на потужність і всеосяжність Spring Framework, інструмент все одно потребує значного часу та знань для налаштування та розгортання додатків Spring. Spring Boot полегшує роботу зі Spring Framework за допомогою трьох важливих можливостей: автоконфігурація, впертий підхід та автономність програм [27].

Автоконфігурація означає, що програми ініціалізуються із попередньо встановленими залежностями, які розробнику не потрібно налаштовувати вручну. Оскільки Spring Boot має вбудовані можливості автоконфігурації, він автоматично налаштовує базову конфігурацію Spring Framework і пакети сторонніх розробників на основі налаштувань розробника. Незважаючи на те, що значення за замовчуванням можуть бути змінені після ініціалізації, функція автоконфігурації Spring Boot дає змогу швидко розпочати розробку програм на основі Spring і зменшує ймовірність помилок, допущених розробником.

Spring Boot використовує впертий підхід до додавання та налаштування початкових залежностей на основі потреб проекту. Дотримуючись власного рішення, Spring Boot вирішує, які пакети встановити та які значення за замовчуванням використовувати, замість того, щоб вимагати від розробника самостійно приймати ці рішення та налаштовувати проект вручну.

Spring Boot допомагає розробникам створювати програми, які просто запускаються. Зокрема, це дозволяє створювати автономні програми, які запускаються самі по собі, не покладаючись на зовнішній веб-сервер, шляхом вбудовування веб-сервера, такого як Tomcat або Netty, у програму під час процесу ініціалізації. У результаті розробник чи користувач може запустити програму на будь-якій платформі, просто натиснувши команду «Виконати».

Найбільшими перевагами використання Spring Boot порівняно зі Spring Framework є простота використання та швидша розробка. Теоретично це відбувається за рахунок більшої гнучкості, яку розробник втрачає від роботи зі

Spring Boot. Але на практиці, якщо розробнику не потрібна дуже специфічна конфігурація, використання Spring Booth варте компромісу.

2.2 База даних Microsoft SQL Server

MS SQL Server – це система керування реляційною базою даних, розроблена Microsoft. Інструмент створено для виконання основних функцій зберігання даних для потреб інших програм. Систему можна запускати на локальному комп'ютері або на віддаленій машині.

Будучи комерційним інструментом, MS SQL Server є однією з найбільш популярних реляційних СУБД, на додачу до MySQL, PostgreSQL і Oracle. Рівень популярності MSSQL порівняно із іншими БД зображено на рисунку 2.2 [28].

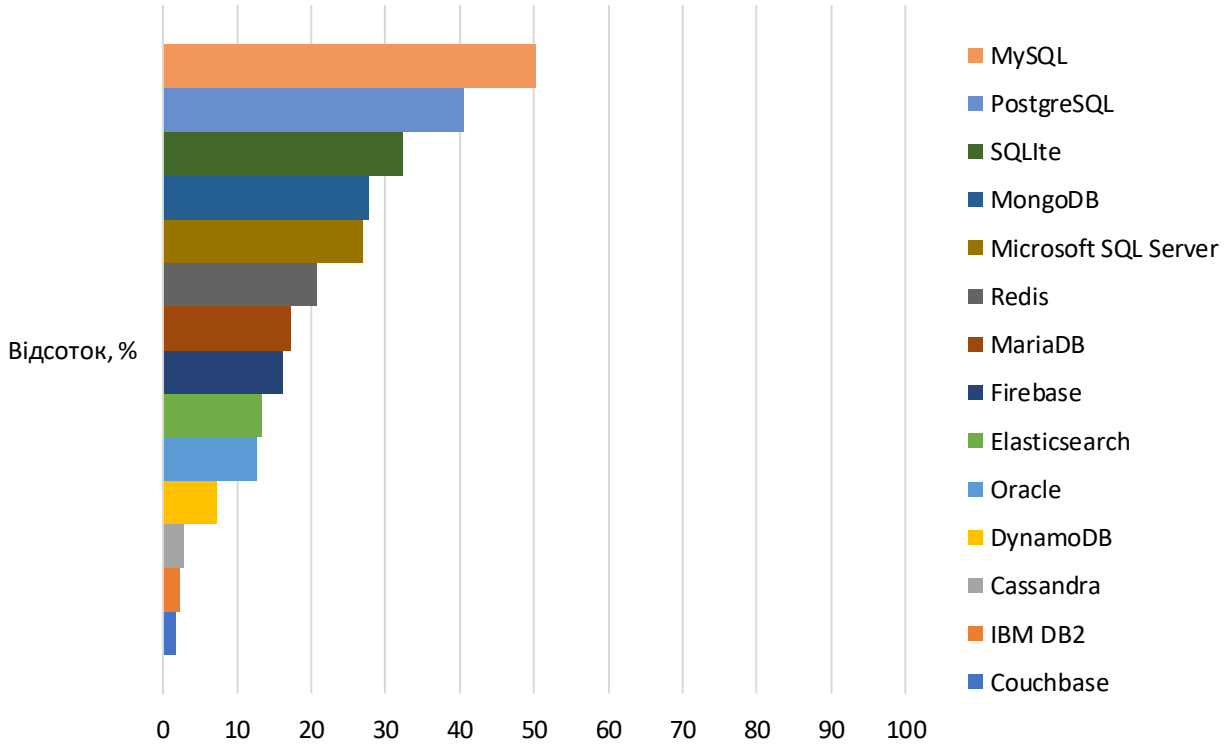


Рисунок 2.2 – Рівень популярності баз даних

Microsoft SQL Server добре справляється із ефективним зберіганням, зміною та керуванням реляційними даними. Для взаємодії з базами даних SQL Server

інженери БД зазвичай використовують мову Transact-SQL (T-SQL), яка є розширенням стандарту SQL.

SQL Server працює в архітектурі клієнт-сервер, тому підтримує два типи компонентів: робочу станцію та сервер. Компоненти робочої станції встановлюються на кожному пристрої користувача SQL Server. Вони є лише інтерфейсами для взаємодії з компонентами сервера. Серверні компоненти встановлюються централізовано на сервер [29].

Microsoft SQL Server надає широкий набір версій із різними функціями. Наприклад, версія Express із безкоштовною базою даних пропонує інструменти початкового рівня, які ідеально підходять для навчання та створення настільних або невеликих серверних програм. Версія Developers дозволяє створювати та тестувати програми, містить деякі корпоративні функції, але не містить ліцензії на робочий сервер. Для більших проектів також існують версії Web, Standard і Enterprise із різним ступенем адміністративних можливостей і рівнів обслуговування. Опис та порівняння версій MSSQL наведено у таблиці 2.1 [30].

Таблиця 2.1 – Версії Microsoft SQL Server

Версія	Опис
Enterprise	Версія преміум-класу. SQL Server Enterprise надає комплексні високоякісні можливості центру обробки даних із надзвичайно високою продуктивністю, необмеженою віртуалізацією та наскрізною бізнес-аналітикою, забезпечуючи високий рівень обслуговування для критично важливих робочих навантажень і доступ кінцевого користувача до аналізу даних.

Продовження таблиці 2.1

Standard	Версія SQL Server Standard забезпечує базове керування даними та базу даних бізнес-аналітики для відділів і невеликих організацій для запуску їх програм, а також підтримує загальні інструменти розробки для локальних і хмарних систем, забезпечуючи ефективне керування базами даних із мінімальними витраченими ресурсами.
Web	Версія SQL Server Web – це варіант із низькою вартістю володіння для веб-хостерів і веб-VAP, що забезпечує масштабованість, доступність і можливості керування для малих і великих ресурсів.
Developer	Версія SQL Server Developer дозволяє розробникам створювати будь-які програми на основі SQL Server. Вона включає в себе всі функції версії Enterprise, але ліцензована для використання лише як система розробки та тестування, а не як робочий сервер.
Express	Версія Express – це безкоштовна база даних початкового рівня, яка підходить для навчання та створення настільних і невеликих серверних програм, керованих даними. Це найоптимальніший вибір для незалежних постачальників програмного забезпечення, розробників і людей, чиїм хобі є створення клієнтських програм. Якщо користувачеві потрібні розширені функції бази даних, SQL Server Express можна оновити до більш просунутих версій SQL Server.

З фокусом на переважно комерційні рішення, MSSQL надає достатню кількість додаткових бізнес-функцій. Можливість вибору додаткових необов'язкових компонентів дозволяє будувати ETL-рішення, формувати базу знань і здійснювати очищення даних. Крім того, MSSQL надає інструменти для

загального адміністрування даних, онлайн-аналітичної обробки та інтелектуального аналізу даних, а також опції для створення звітів і візуалізації.

Microsoft SQL Server містить ряд недоліків, через які певним користувачам може бути складно або неможливо користуватися системою для зберігання даних і роботи з ними. Цими недоліками є висока ціна, нечіткі умови ліцензії та складний процес налаштування.

MSSQL Server використовується в основному для корпоративних продуктів, тому він є одним із найдорожчих рішень. Версія Enterprise коштує понад 14 000 доларів США за ядро, яке продається у вигляді двох ключових пакетів [31].

Іншим недоліком є процес ліцензування, який постійно змінюється. Стратегію ціноутворення важко зрозуміти, а елементи, включені в конкретне видання, є плаваючими, мають тенденцію змінюватися від одного варіанту до іншого.

Для початківців, яким доводиться працювати з великими наборами даних, робота з оптимізацією запитів і налаштуванням продуктивності може бути проблематичною. Оскільки процес не такий очевидний, він може створити значні вузькі місця на ранньому етапі розробки.

2.3 Розгортання

Розгортання додатків, також відоме як розгортання програмного забезпечення, – це процес інсталяції, налаштування, оновлення та ввімкнення однієї програми або набору програм, які роблять програмну систему доступною для використання, як-от надання певної URL-адреси на сервері.

Розгортання програми є одним із найважливіших етапів процесу розробки програмного забезпечення, оскільки стратегія, яка використовується для створення, тестування та розгортання, безпосередньо впливатиме на те, наскільки швидко програма може реагувати на зміни в налаштуваннях або вимогах до складових [32].

Існує безліч способів розгортати програму: вручну, за допомогою віртуальних машин, за допомогою інструментів контейнеризації, за допомогою оркестраторів контейнерів тощо. Розгортання вручну є найменш зручним та найбільш трудомістким способом, оскільки вимагає ручного налаштування кожного компоненту програми. Якщо програму потрібно перемістити чи скопіювати на іншу машину, ручне налаштування потрібно провести заново. Єдиною перевагою даного методу є відсутність необхідності стороннього програмного забезпечення для віртуалізації чи контейнеризації.

Розгортання за допомогою віртуальних машин дозволяє уникнути трудомісткого процесу ручного налаштування, якщо є необхідність у дублюванні чи перенесенні програми. Крім того, програми для віртуалізації, такі як VMware чи VirtualBox, дозволяють робити знімки віртуальної машини, за допомогою яких можна відкотитися до необхідного стану [33, 34]. Недоліком даного методу є великий розмір розгорнутих віртуальних машин, оскільки кожна віртуальна машина містить повноцінну операційну систему із власним ядром, ресурсами, процесами та іноді власним графічним інтерфейсом.

Контейнерізація – це пакування додатку лише з бібліотеками операційної системи і залежностями, необхідними для запуску програми, що створює єдиний легкий виконуваний файл, який називається контейнером та узгоджено працює при будь-якій інфраструктурі. Контейнери є більш портативними та ресурсоефективними, ніж віртуальні машини. Контейнери є де-факто стандартними обчислювальними одиницями сучасних хмарних програм [35].

Концепції контейнеризації та ізоляції процесів існують протягом десятиліть, але саме поява в 2013 році Docker Engine прискорила впровадження технології. Сьогодні організації все частіше використовують контейнеризацію для створення нових програм і модернізації існуючих програм для хмари. Так, 61% користувачів технології контейнеризації відмітили, що вони використовували контейнери у 50% чи більше випадків при розробці нового програмного забезпечення протягом 2018-2020 років [36].

Контейнери часто називають легкими, оскільки вони мають спільне із комп'ютером ядро операційної системи і не вимагають додаткових ресурсів для нової операційної системи в кожній програмі. Контейнери за своєю суттю мають меншу місткість, ніж віртуальна машина, і потребують менше часу на запуск, що дозволяє працювати набагато більшій кількості контейнерів із тією ж обчислювальною потужністю, що й одна віртуальна машина. Це підвищує ефективність серверів і, у свою чергу, зменшує витрати на сервери й ліцензування.

Оркестрація контейнерів – це процес автоматизації більшої частини процесів операційної системи, необхідних для виконання контейнерних робочих навантажень і служб. Це включає в себе широкий спектр речей, необхідних розробникам для керування життєвим циклом контейнерів, включаючи надання, розгортання, масштабування у більшу чи меншу сторону, роботу із мережею, балансування навантаження тощо.

Оскільки контейнери легкі та ефемерні за своєю природою, їх запуск може легко стати великою проблемою. Зокрема, у поєднанні з мікросервісами, кожен з яких зазвичай працює у власних контейнерах, використання контейнеризованих програм може призвести до роботи з сотнями чи тисячами контейнерів, особливо під час створення та експлуатації будь-якої великомасштабної системи. Це може створити значну складність, якщо керувати програмами вручну. Оркестрація контейнерів робить дану операційну складність керованою для розробки, оскільки вона забезпечує декларативний спосіб автоматизації значної частини роботи.

При використанні оркестратора контейнерів зручно працювати із хмарними сервісами, скільки хмарні провайдери, такі як Amazon чи Microsoft надають готові рішення із керування контейнерами [37, 38].

Засобом для контейнеризації програми дипломної роботи був обраний інструмент Docker. У якості оркестратора контейнерів був обраний інструмент Kubernetes. Для хмарних обчислень був обраний Amazon Web Services.

2.3.1 Система контейнеризації Docker

Docker надає можливість пакувати та запускати програму в ізольованому середовищі. Ізоляція дозволяє запускати багато контейнерів одночасно на одній машині. Контейнери є легкими та містять усе необхідне для запуску програми.

Коли Docker був створений, він мав монолітну архітектуру. Сучасна архітектура Docker зображена на рисунку 2.3 [39].

Архітектура Docker розділена на наступні три компоненти:

- Docker Engine (dockerd)
- docker-containerd (containerd)
- docker-runc (runc)

Docker Engine складається з демона Docker, інтерфейсу API та Docker CLI. Демон Docker (dockerd) працює безперервно як служба dockerd systemd. Він відповідає за створення образів Docker (Docker images). Щоб керувати зображеннями та запускати контейнери, dockerd викликає API docker-container.

Containerd – це демон, який відповідає за завантаження образів Docker і їх запуску як контейнерів. Containerd надає API для отримання інструкцій від служби Docker.

Runc – це середовище виконання контейнера, яке відповідає за створення просторів імен і контрольних груп, необхідних для контейнера, та запуск команд контейнера всередині створених просторів імен.

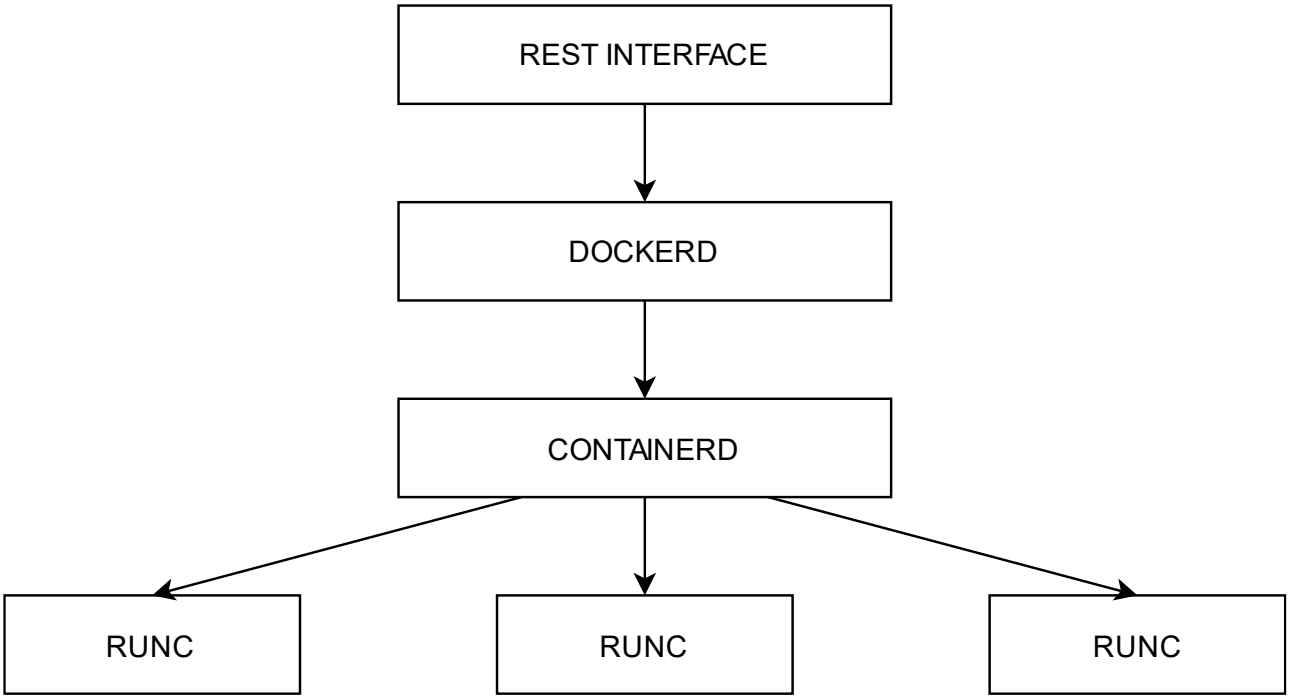


Рисунок 2.3 – Архітектура Docker

2.3.2 Оркестратор контейнерів Kubernetes

Kubernetes – це інструмент для керування життєвим циклом контейнерних програм організації. Це певного роду метапроцес, який надає можливість автоматизувати розгортання та масштабування багатьох контейнерів одночасно. Контейнери, що запускають одну програму, групуються разом. Дані контейнери діють як копії та служать для балансування навантаження вхідних запитів. Оркестратор контейнерів контролює групи даних контейнерів, гарантуючи, що вони працюють правильно [40].

На рисунку 2.4 зображено архітектуру Kubernetes, що містить наступні компоненти: контрольна площина, робочі вузли, поди, користувацький інтерфейс, CLI та інші.

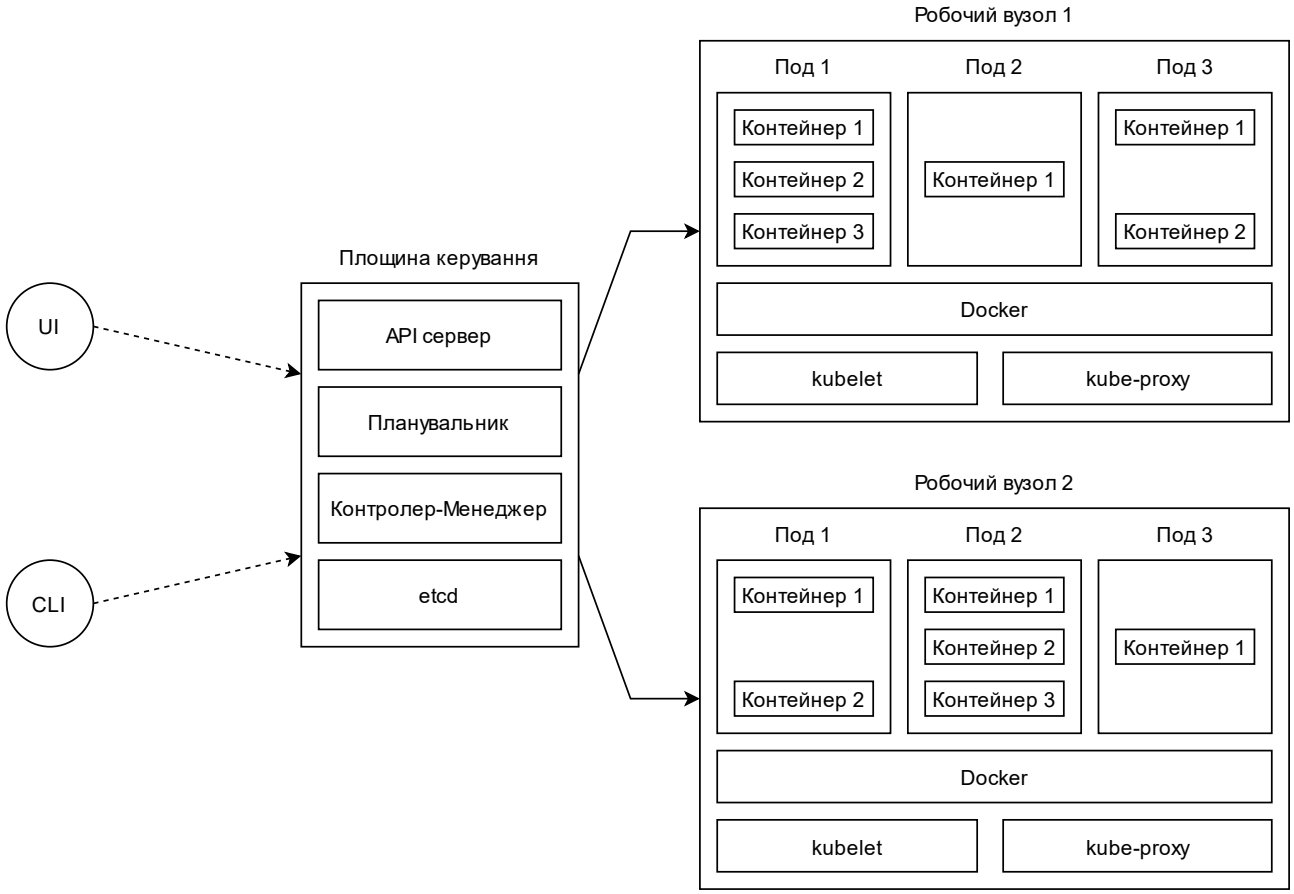


Рисунок 2.4 – Архітектура Kubernetes

Под Kubernetes – це група контейнерів і найменший блок, яким керує Kubernetes. Модулі мають єдину IP-адресу, яка застосовується до кожного контейнера в пакеті. Контейнери в модулі спільно використовують ресурси, такі як пам’ять чи сховище. Це дозволяє обробляти окремі контейнери Linux всередині поди разом як єдину програму. Поширеним явищем є под із одним контейнером, де програма чи служба є єдиним процесом. При ускладненні системи, коли кілька процесів повинні працювати разом, використовуючи спільні ресурси, багатоконтейнерні модулі спрощують конфігурацію розгортання порівняно з налаштуванням спільних ресурсів між контейнерами самостійно.

Вузол Kubernetes – це віртуальна чи фізична машина, яка керує подами та запускає їх. Подібно до того, як поди містять окремі контейнери, які працюють разом, вузол містить поди, які функціонують разом.

Площина керування Kubernetes є основною точкою входу для адміністраторів і користувачів для керування вузлами. Операції виконуються через HTTP-виклики або через інтерфейс командного рядка. Площина керування контролює взаємодію Kubernetes із іншими програмами.

Сервер API надає кластеру Kubernetes інтерфейс REST. Усі операції з модулями, службами тощо виконуються програмно через кінцеві точки серверу API.

Планувальник відповідає за призначення роботи різним вузлам. Він стежить за потужністю ресурсів і гарантує, що продуктивність робочого вузла знаходиться у прийнятних межах.

Контролер-менеджер відповідає за те, щоб спільний стан кластера працював належним чином. Диспетчер контролерів контролює контролери, які реагують на події, такі як вихід вузла із ладу.

Kubelet відстежує стан поди, щоб розуміти чи всі контейнери працюють. Кожні кілька секунд він надсилає до площини управління спеціально сформоване повідомлення. Якщо контролер реплікації не отримує це повідомлення, вузол позначається як несправний.

Kube-proxy направляє трафік, що надходить до вузла від сервісу. Kube-proxy пересилає запити до необхідних контейнерів.

etcd – це розподілене сховище типу ключ-значення, яке Kubernetes використовує для обміну інформацією про загальний стан кластера. Крім того, вузли можуть звертатися до даних глобальної конфігурації, що зберігаються у etcd, щоб проводити налаштування після перезапуску.

2.3.3 Хмарний сервіс Amazon Web Services

Сервіс Amazon Web Services почав пропонувати компаніям інформаційні послуги формі веб-сервісів у 2006 році [41]. Однією з ключових переваг хмарних обчислень є можливість замінити авансові капітальні витрати на інфраструктуру

низькими змінними витратами, які масштабуються відповідно до потреб бізнесу. Завдяки хмарним обчисленням компаніям більше не потрібно планувати та закуповувати сервери чи інші елементи IT-інфраструктури за тижні чи місяці до випуску інформаційного продукту. Хмари дозволяють миттєво запустити сотні чи тисячі серверів за лічені хвилини.

Хмара AWS охоплює 90 зон доступності в 28 географічних регіонах по всьому світу. Також компанія Amazon оголосила про плани щодо створення додаткових 21 зони доступності та 7 регіонів AWS у Австралії, Канаді, Індії, Ізраїлі, Новій Зеландії, Іспанії та Таїланді [41].

Безпека в AWS починається із основної інфраструктури. Усі дані, що надходять через глобальну мережу AWS, яка з'єднує центри обробки даних і регіони, автоматично шифруються на фізичному рівні, перш ніж покинути захищені об'єкти.

AWS забезпечує високу доступність мережі. Кожен регіон повністю ізольований і складається з кількох AZ, які є повністю ізольованими одне від одного. За потреби є можливість розділу програм між кількома AZ в одному регіоні. Крім того, рівні керування AWS і консоль керування AWS розподіляються між регіонами та включають регіональні кінцеві точки API, які розроблені для безпечної роботи.

Глобальна інфраструктура AWS створена для продуктивності. Регіони AWS пропонують низьку затримку, низьку втрату пакетів і високу загальну якість мережі. Це досягається за допомогою повністю резервованої волоконно-оптичної магістралі мережі 100 GbE, яка терабіти пропускної здатності між регіонами. AWS Local Zones і AWS Wavelength забезпечують високу продуктивність для додатків, що вимагають мінімальних затримок.

Глобальна інфраструктура AWS надає компаніям високу гнучкість та доступ до масштабованості хмари. Компанії можуть швидко нарощувати необхідні ресурси, розгортаючи сотні чи навіть тисячі серверів за лічені хвилини.

Глобальна інфраструктура AWS дає можливість обирати як і де виконувати робочі навантаження. Якщо є необхідність у запуску програми для всього світу, є можливість обрати будь-який із регіонів AWS або AZ. Якщо є потреба у мінімальних затримках програми для мобільних пристроїв і кінцевих користувачів, можна обрати AWS Local Zones або AWS Wavelength. Якщо програмі достатньо локального запуску, існує AWS Outposts.

Висновки до розділу 2

У даному розділі були проаналізовані методи реалізації веб-сервісу обчислення показників рівня міжнародного співробітництва. Для реалізації веб-сервісу необхідно створити сервер API. У якості інструменту для створення серверу API був обраний Spring Boot завдяки високому рівню підтримки від розробників, зручній конфігурації та побудові, а також прийнятній швидкодії.

Для зберігання даних, необхідних для веб-сервісу, використовується база даних Microsoft SQL Server. MSSQL є поширеним рішенням для розробників, що працюють над комерційними системами. Microsoft надає широкий вибір можливих варіантів своєї бази даних, починаючи від безкоштовних версій, що гадають мінімальний функціонал для розробників, і закінчуючи версіями Enterprise та Developer, які дають максимально великий спектр доступних функцій. Enterprise є платним рішенням для промислових додатків, Developer є безкоштовним рішенням для розробників, що хочуть ознайомитися із функціями Microsoft SQL Server.

Для розгортання додатку були обрані інструменти Docker та Kubernetes. Docker дозволяє створювати та запускати додатки у ізольованому середовищі, що називається контейнерами. При роботі із комерційними додатками складність роботи із контейнерами за допомогою лише Docker буде зростати пропорційно кількості контейнеризованих сервісів. Для спрощення роботи розробнику використовується оркестратор контейнерів Kubernetes, що автоматично займається

розгортанням контейнеризованих сервісів, їх масштабуванням, виділенням ресурсів, знищенням тощо.

При роботі із Kubernetes зручно користуватися послугами хмарних провайдерів. У такому випадку немає необхідності вручну встановлювати і налаштовувати Kubernetes та Docker, планувати та закуповувати фізичне обладнання у якості робочої станції. При використанні хмарного провайдера все це уже зроблено, а компанія може сконцентруватися на розробці та підтримці продукту, не переймаючись за програмну чи фізичну підтримку обладнання.

У якості хмарного провайдера був обраний Amazon, що володіє хмарним сервісом Amazon Web Services. AWS має широкий спектр регіонів. Дата-центри AWS розташовані на п’яти континентах у 28 регіонах світу. Сумарно AWS надає 90 зон, якими можуть скористатися клієнти Amazon.

3 РОЗРОБКА ВЕБ-СЕРВІСУ ОБЧИСЛЕННЯ ПОКАЗНИКІВ РІВНЯ МІЖНАРОДНОГО СПІВРОБІТНИЦТВА

3.1 Архітектура веб-сервісу

Веб-сервіс будується за дорпомогою архітектурного шаблону Model-View-Controller (MVC). Цей тристоронній поділ програми передбачає відокремлення частин, які представляють модель основного домену програми, від того, як модель представлена користувачеві, і від того, як користувач з нею взаємодіє.

MVC – це тристоронній шаблон, за допомогою якого об’єкти різних класів беруть на себе операції, пов’язані з доменом програми (модель), відображенням стану програми (вид) і взаємодії користувача з моделлю та представленням (контролером) [42]. На рисунку 3.1 зображено схему шаблону Model-View-Controller.

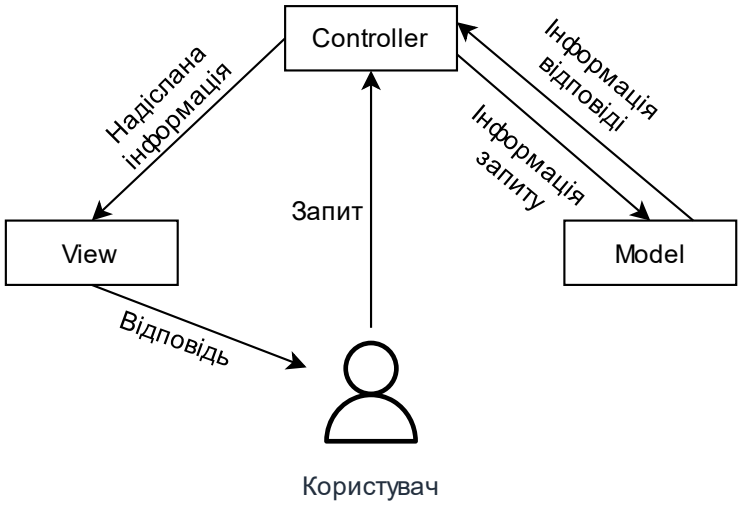


Рисунок 3.1 – Схема шаблону Model-View-Controller

Робота Model полягає в тому, щоб керувати даними. Незалежно від того, чи надходять дані з бази даних, API чи об’єкта JSON, обов’язком Model є керування даними.

Робота View полягає в тому, щоб визначати те, що користувач бачитиме на своєму екрані та як.

Робота Controller полягає в отриманні, зміні та наданні даних користувачеві. Іншими словами, Controller є сполучною ланкою між View і Model.

На рисунку 3.2 зображено структуру веб-сервісу. Кореневим пакетом є пакет kpi.ipze.tv.makarenko. Всередині кореневого пакету містяться пакети controller, dto, entity, repository, service.

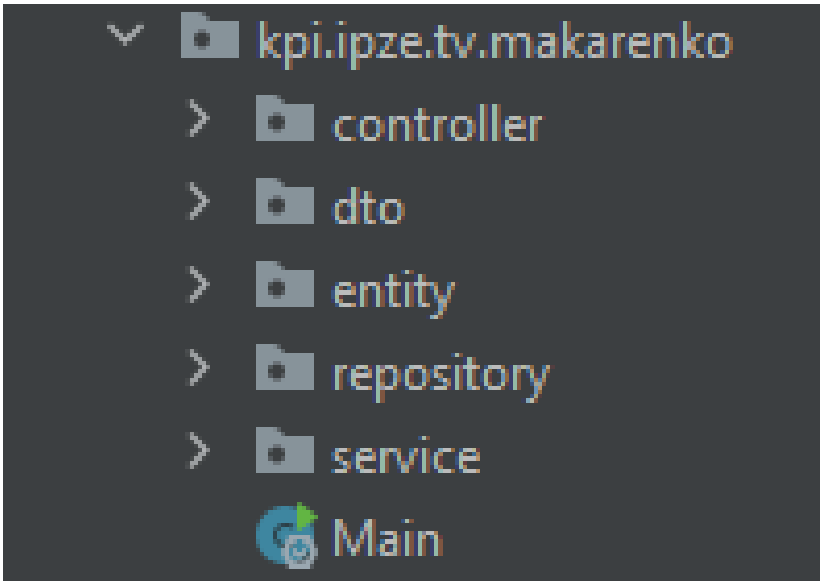


Рисунок 3.2 – Структура веб-сервісу для обчислення показників МСНО у НТУ

Пакет controller містить класи-контролери, що використовуються для формування кінцевих точок веб-сервісу. Контролери інтерпретують введені користувачем дані та перетворюють їх на модель, що представляється користувачеві.

Пакети entity та dto містять класи, що є класами-моделями та класами-відображеннями відповідно. Веб-сервіс повертає користувачеві дані, сформовані на основі значень полів класів у пакеті entity у вигляді JSON. Сформований JSON має ту ж структуру, що і класи у пакеті dto.

Пакет repository містить необхідні інтерфейси для взаємодії зі сховищем даних. За допомогою методів інтерфейсів пакету repository формуються SQL-

запити до бази даних MSSQL. Запити можуть бути як автоматично згенерованими за допомогою можливостей Spring Data, так і сформовані вручну за допомогою анотації @Query.

Пакет service містить класи, що відповідають за бізнес-логіку сервісу. Класи пакету service отримують інформацію із бази даних за допомогою інтерфейсів пакету repository, виконують необхідні обчислення і повертають обраховані значення, які потім використовуються класами-контролерами із пакету controller.

Клас Main є точкою запуску програми. Main використовує вбудовані у Spring Boot інструменти для аналізу classpath, сканування компонентів додатку та, за необхідності, додання відсутніх конфігураційних елементів [43].

3.2 Обробка даних

3.2.1 Отримання даних із бази даних

Необхідна інформація для обчислення показників береться із бази даних, схема якої зображена на рисунку 3.1. Серед наведених таблиць сервісу необхідними для обчислення МСНО у НТУ є таблиці Worker, Organization, CollectedData та Achievement. Для репозиторію необхідні інтерфейси для формування запитів у дані таблиці. Назвемо їх WorkerRepository, OrganizationRepository, CollectedDataRepository та AchievementRepository. Повний код інтерфейсів розміщено у додатку А.

Інструменти Spring Data JPA надають змогу автоматизовано будувати запити до бази даних. Так, примітивні запити, такі як отримання даних за допомогою первинного ключа, а також велика частина більш складних запитів можуть бути побудовані за допомогою спеціально названих методів, як це продемонстровано у [44].

Якщо у розробника виникає потреба у запиті до бази даних, який не здатен бути побудований за допомогою спеціально названих методів інтерфейсу-

репозиторію, можливо скористатися анотацією @Query. За замовчуванням @Query використовує JPQL, але також є можливість використовувати чистий SQL [45].

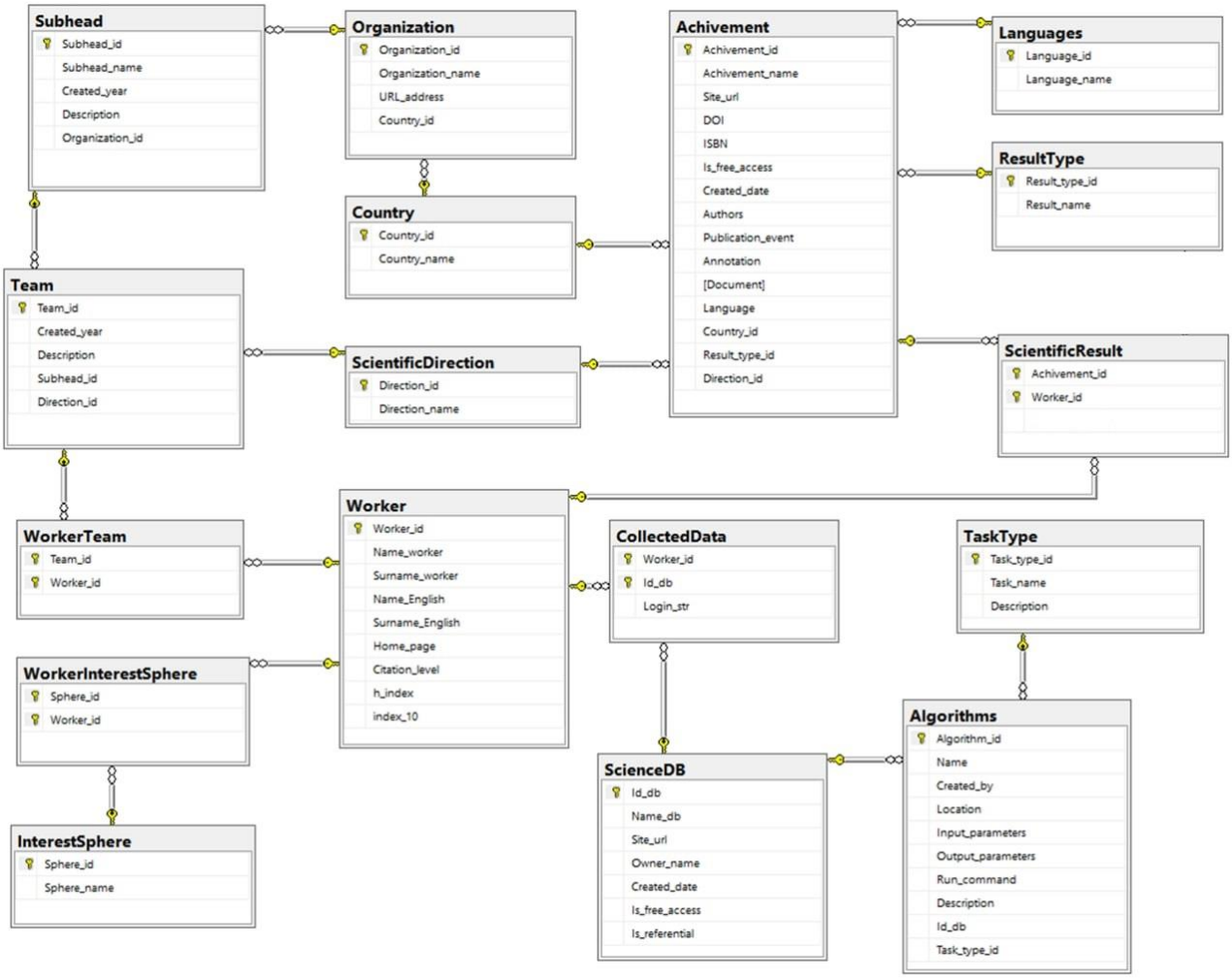


Рисунок 3.3 – Схема бази даних для обчислення показників МСНО у НТУ

Інтерфейсам WorkerRepository, OrganizationRepository та CollectedDataRepository для формування своїх запитів достатньо того функціоналу, який надає інструмент, описаний у [44]. Інтерфейс WorkerRepository містить метод findByNameAndSurname, який повертає Optional<Worker>. У випадку, якщо знайдено працівника у базі даних із заданим ім’ям та прізвищем, Optional буде містити об’єкт типу Worker із полями, що містять отримані із бази даних значення. Якщо працівника із заданим ім’ям та прізвищем у базі даних немає, Optional буде пустим.

Аналогічно до WorkerRepository, OrganizationRepository містить метод findByName, що повертає Optional<Organization>. Метод findByName робить запит до бази даних для отримання інформації про організацією із зазначеною назвою. Якщо організації із зазначеним іменем не існує, повернений Optional буде пустим.

Інтерфейс CollectedDataRepository містить метод countByWorkerIn, що приймає колекцію працівників як вхідний параметр. Метод робить запит на підрахунок співробітників із колекції, що містяться у базі даних.

Інтерфейс AchievementRepository наступні містить методи:

- countByWorkersAndResultTypeNameIgnoreCase
- countByWorkersAndPublicationEventContainingIgnoreCase
- countByWorkersAndPublicationEventContainingIgnoreCaseAndNotCountryNameIgnoreCase

Усі вищеназвані методи, не зважаючи на спеціально сформовану назву, використовують анотацію @Query для формування запиту до бази даних. Кожна із анотацій методів використовує JPQL.

Метод countByWorkersAndResultTypeNameIgnoreCase приймає множину співробітників та рядок, що є назвою результату, як вхадні параметри. На виході метод повертає список цілих чисел, кожне із яких є кількістю досягнень певного співробітника.

Метод countByWorkersAndPublicationEventContainingIgnoreCase приймає множину співробітників та назву події як вхідні параметри. На виході метод повертає список цілих чисел, кожне із яких є кількістю подій певного співробітника.

Останній метод приймає три параметри на вхід: множину співробітників, ім’я наукової події, назва країни. У результаті виконання методу повертається список цілих чисел, кожне із яких є кількістю подій співробітника, які опубліковані не у вказаній країні.

3.2.2 Обробка даних сервісами веб-додатку

Обробка отриманих із бази даних відбувається за допомогою сервісів, що містять бізнес-логіку веб-додатку. Сервіси містяться у пакеті service. При роботі зі Spring Boot можливо використовувати як інтерфейси для створення сервісів, так і безпосередньо класи без імплементацій інтерфейсів. Для певних випадків використання інтерфейсів є більш бажаним рішенням [46].

Веб-додаток містить два наступні сервісні інтерфейси для обчислення показників: OrganizationService та WorkerService. Для кожного із інтерфейсів є відповідний клас-імплементація. Програмний код класів-імплементацій для зазначених інтерфейсів наведено у додатку Б.

WorkerService містить наступні методи:

- amountInScienceDb
- hIndex
- citationLevel
- index10
- amountOfPublishedThesesOrPresentations
- amountOfForeignPublications
- allParams

Кожен із методів приймає на вхід ім’я та прізвище науковця як вхідні параметри. Це потрібно для пошуку науковця у базі даних за допомогою WorkerRepository. Якщо науковець у базі даних присутній, метод виконує свою бізнес-логіку та повертає необхідне значення. Якщо науковця із заданим ім’ям та прізвищем у базі даних немає, метод повертає значення по замовчуванню, яке, як правило, є нулем або пустим значенням.

amountInScienceDb виконує обрахунок того, скільки записів є про науковця у наукометричних базах даних, таких як Scopus або Google Scholar. Для цього він використовує метод countByWorkerIn із WorkerRepository. Особливістю countByWorkerIn є те, що вхідним параметром є множина значень співробітників,

а не єдиний співробітник. Для обходу даного обмеження можна скористатися статичним методом of із класу Set, який на вхід приймає від одного до безлічі об’єктів. Прописавши Set.of(worker) можна створити множину співробітників, що містить одного співробітника. Цю множину можна використати як вхідний параметр для методу countByWorkerIn, який порахує кількість записів про співробітника.

hIndex повертає індекс Гірша для співробітника із указаним ім’ям та прізвищем. Якщо співробітника у базі даних не знайдно, повертається нульове значення. Аналогічно до цього, методи citationLevel та index10 повертають рівень цитування вченого та кількість його публікацій, що має щонайменше десять цитувань відповідно.

amountOfPublishedThesesOrPresentations виконує обчислення кількості опублікованих вченим тез чи презентацій. Для цього метод використовує WorkerRepository для отримання окремо кількості презентацій та кількості тез науковця. Оскільки дані збираються лише про одного науковця, метод countByWorkersAndResultTypeNameIgnoreCase із інтерфейсу WorkerRepository повертатиме список, що міститиме лише одне значення, тому достатньо скористатися методом get із вхідним параметром 0 для отримання даного значення. amountOfPublishedThesesOrPresentations повертає суму кількості презентацій вченого та кількісті тез ученого.

amountOfForeignPublications повертає кількість закордонних публікацій науковця. Для цього робиться виклик методу інтерфейсу WorkerRepository countByWorkersAndPublicationEventContainingIgnoreCaseAndNotCountryNameIgnoreCase, де на вхід подається множина, що містить науковця, подія publication та ім’я країни Ukraine. Метод поверне кількість публікацій вказаного науковця, що були випущені не в Україні. Аналогічно із countByWorkersAndResultTypeNameIgnoreCase, на виході буде отримано список із єдиним значенням, тому можна скористатися методом get із вхідним параметром 0.

allParams повертає клас типу WorkerParamsDto, що містить усі обраховані показники для конкретного співробітника. Для цього WorkerParamsDto використовує паттерн Builder. Builder дозволяє встановлювати значення полів класу без необхідності у використанні конструктора класу або спеціальних методів-сеттерів. Якщо клас містить велику кількість полів, значення яких необхідно ініціалізувати, використання конструктора або спеціальних методів-сеттерів є незручним. Шаблон проектування Builder дозволяє у довільному порядку встановити значення полів та отримати повністю заініціалізований клас на виході. Джерело [47] містить детальнішу інформацію про шаблон проектування Builder.

allParams використовує усі вищезазначені методи WorkerService для ініціалізації полів об'єкту типу WorkerParamsDto. Якщо співробітника із вказаним ім'ям та прізвищем не існує, поля об'єкту класу WorkerParamsDto будуть містити нульові або пусті значення.

OrganizationService містить наступні методи:

- amountOfWorkersInInternationalConferences
- amountOfDefendedDissertations
- amountOfWorkersInInternationalEvents
- amountOfPublishedThesesOrPresentations
- amountInScienceDb
- amountOfForeignPublications
- allParams

Кожен із методів приймає на вхід ім'я організації у якості вхідного параметру. Це необхідно для пошуку організації у базі даних за її назвою за допомогою методу із інтерфейсу OrganizationRepository. Якщо організація у базі даних присутня, метод виконує власну частину бізнес-логіки і повертає обраховане значення. Якщо організації із заданою назвою у базі даних немає, метод повертає значення за замовчуванням, яке дорівнює нулю або пустому значенню.

amountOfWorkersInInternationalConferences обраховує кількість співробітників установи, що приймають чи приймали участь у міжнародних

конференціях. На додачу до інтерфейсу OrganizationRepository, метод використовує інтерфейс AchievementRepository. За допомогою методу countByWorkersAndPublicationEventContainingIgnoreCase, що належить AchievementRepository, повертається список цілих чисел, кожне із яких є кількістю міжнародних конференцій, у яких прийняв участь певний науковець. Оскільки повертається список цілих чисел, а не єдине ціле число, значення, що містяться у списку, необхідно просумувати для отримання фінального результату. Це можна зробити як і вручну у циклі, так і за допомогою Java Stream API. Java Stream API дозволяють писати функціональний програмний код, що дозволяє проводити операції над масивом даних без використання розробником циклів [48].

amountOfDefendedDissertations повертає кількість захищених у науковій установі дисертацій. Метод використовує OrganizationRepository та AchievementRepository для отримання інформації із бази даних. Аналогічно до amountOfWorkersInInternationalConferences, метод використовує Java Stream API для сумування отриманих із бази даних результатів і повернення єдиного цілого числа.

amountOfWorkersInInternationalEvents рахує у скількох міжнародних подіях брали участь співробітники організації. Для цього метод використовує OrganizationRepository для отримання інформації про організацію та AchievementRepository для отримання кількості міжнародних подій, у яких взяли участь співробітники організації. Аналогічно до попередніх методів, amountOfWorkersInInternationalEvents використовує Java Stream API для перетворення списку цілих чисел у єдине число, що є їх сумою.

Метод amountOfPublishedThesesOrPresentations у інтерфейсі OrganizationService призначений для того ж, що і однойменний метод у інтерфейсі WorkerService: він обраховує кількість опублікованих тез чи презентацій. Різниця полягає у тому, що метод інтерфейсу OrganizationService використовується для обрахунку тез чи презентацій усіх співробітників організації, а не лише одного конкретного співробітника. Аналогічно метод amountInScienceDb, що рахує

кількість матеріалів у наукометричних базах даних для всіх співробітників наукової установи, та метод amountOfForeignPublications, що рахує кількість закордонних публікацій для всіх науковців організації.

allParams використовує усі вищезазначені методи OrganizationService для ініціалізації полів об’єкту типу OrganizationParamsDto. Якщо організації із вказаною назвою не існує, поля об’єкту класу OrganizationParamsDto будуть містити нульові або пусті значення.

3.2.3 Кінцеві точки веб-додатку

Кінцеві точки у додатках, написаних із використанням Spring Framework, створюються за допомогою контролерів. Контролер – це Java-клас, помічений анотацією @Controller або @RestController. Анотація @RequestMapping вказує базовий шлях, за яким знаходяться усі методи контролера. Для того, щоб перейти до конкретного метода контролера, необхідно над методом вказати у анотації @GetMapping, @PostMapping, @PutMapping або @DeleteMapping шлях до методу. Шлях, вказаний у @RequestMapping класу та @GetMapping, @PostMapping, @PutMapping або @DeleteMapping методу утворюють кінцеву точку, за якою користувач може отримати результат конкретного методу, що знаходиться у конкретному класі-контролері.

Веб-сервіс містить контролери OrganizationController та WorkerController у пакеті controller. Кожен із контролерів відповідає за свою частину роботи. OrganizationController працює із усім, що пов’язано із параметрами організації, а WorkerService – із параметрами співробітників організації. Код класів контролерів наведено у додатку В.

Базовим шляхом до WorkerController є /worker. Методи, що містить WorkerController наведено у таблиці 3.1. Варто відмітити, що вхідні параметри для кожного із методів є однаковими. Вхідними параметрами є ім’я та прізвище

науковця (name, surname), для якого відбувається пошук у базі даних і для якого відбувається обчислення показників.

Таблиця 3.1 – Методи класу WorkerController

Назва	Шлях URL	Вихідні параметри
amountInScienceDb	/{name}-{surname}/amount-in-scientific-db	Mono<Integer>
hIndex	/{name}-{surname}/h-index	Mono<Float>
citationLevel	/{name}-{surname}/citation-level	Mono<Float>
index10	/{name}-{surname}/index-10	Mono<Float>
amountOfPublished ThesesOrPresentations	/{name}-{surname}/amount-of-published-theses-or-presentations	Mono<Integer>
amountOfForeign Publications	/{name}-{surname}/amount-of-foreign-publications	Mono<Integer>
allParams	/{name}-{surname}	Mono<WorkerParamsDto>

Обчислені значення загортаються у об’єкт типу Mono для підтримки асинхронності веб-сервісу. Mono – це спеціалізована версія Publisher, який видає щонайбільше один елемент за допомогою сигналу onNext, а потім завершується сигналом onComplete або onError [49].

Базовим шляхом до Organization є /organization. Методи, що містить OrganizationController, наведено у таблиці 3.2. Вхідним параметром для усіх методів є назва організації, для якої проводиться обчислення показників міжнародної взаємодії.

Таблиця 3.2 – Методи класу OrganizationController

Назва	Шлях URL	Вихідні параметри
amountInScienceDb	/{name}/amount-in-science-db	Mono<Integer>
amountOfWorkers InInternationalEvents	/{name}/amount-of-workers-in-international-events	Mono<Integer>
amountOf DefendedDissertations	/{name}/amount-of-defended-dissertations	Mono<Integer>
amountOfWorkersIn InternationalConferences	/{name}/amount-of-workers-in-international-conferences	Mono<Integer>
amountOfPublished ThesesOrPresentations	/{name}/amount-of-published-theses-or-presentations	Mono<Integer>
amountOf ForeignPublications	/{name}/amount-of-foreign-publications	Mono<Integer>
allParams	/{name}	Mono<WorkerParamsDto>

3.3 Розгортання веб-сервісу у хмарному середовищі AWS

3.3.1 Створення необхідних хмарних ресурсів

Для роботи із сервісом AWS необхідно мати зареєстрований на сайті aws.amazon.com акаунт. Акаунт із сайту www.amazon.com не підійде, не зважаючи на те, що обидва сайти належать Amazon. Перший сайт використовується для роботи із хмарним середовищем, другий – для онлайн-покупок.

Після реєстрації акаунту на aws.amazon.com необхідно створити кластер для роботи Kubernetes у хмарі Amazon, або Elastic Kubernetes Service. Для цього потрібно створити необхідну роль. Створенням ролей та політик займається сервіс AWS Identity and Access Management.

Для переходу на сторінку IAM можна скористатися пошуковим рядком у лівому верхньому куті веб-сторінки AWS Management Console. На лівій частині веб-сторінки IAM присутня вкладка Access management, де можна знайти посилання Roles для перегляду, редагування та створення ролей.

Для створення потрібної ролі вказуємо AWS service як Trusted entity type, у якості Use case вказуємо EKS - Cluster, як це показано на рисунку 3.4. Наступна сторінка дозволяє налаштувати дозволи ролі за допомогою політик. Серед наявних політик для ролі, що створюється, доступна лише AmazonEKSClusterPolicy. Остання сторінка дозволяє надати ім'я ролі, додати їй опис, а також проглянути і за необхідності змінити налаштування, що були зроблені у попередніх кроках.

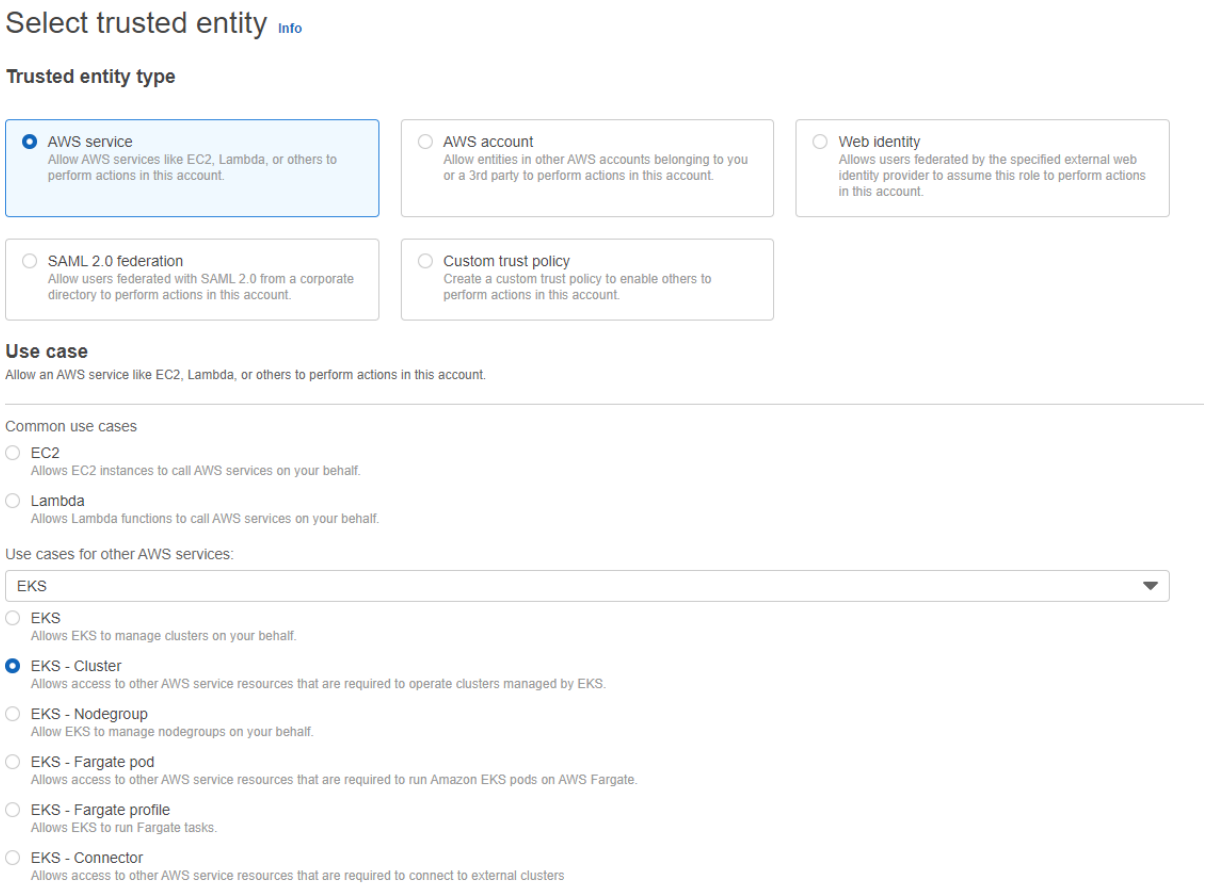


Рисунок 3.4 – Створення та початкове налаштування ролі для кластеру EKS

Для створення кластеру переходимо до сервісу EKS і натискаємо на Add cluster → Create. Початкова сторінка дозволяє налаштувати ім'я кластеру, вказати

версію Kubernetes, а також обрати необхідну роль. Ім’я може бути довільним, але змістовним, тому назвемо кластер kpi-service. Версію Kubernetes вкажемо як останню із доступних для вибору. У якості ролі оберемо ту, що була створена раніше. Після проведених кроків сторінка налаштувань буде мати вигляд, вказаний на рисунку 3.5. Мережеві налаштування та налаштування логування, що проводяться на наступних сторінках, можна залишити за замовчуванням.

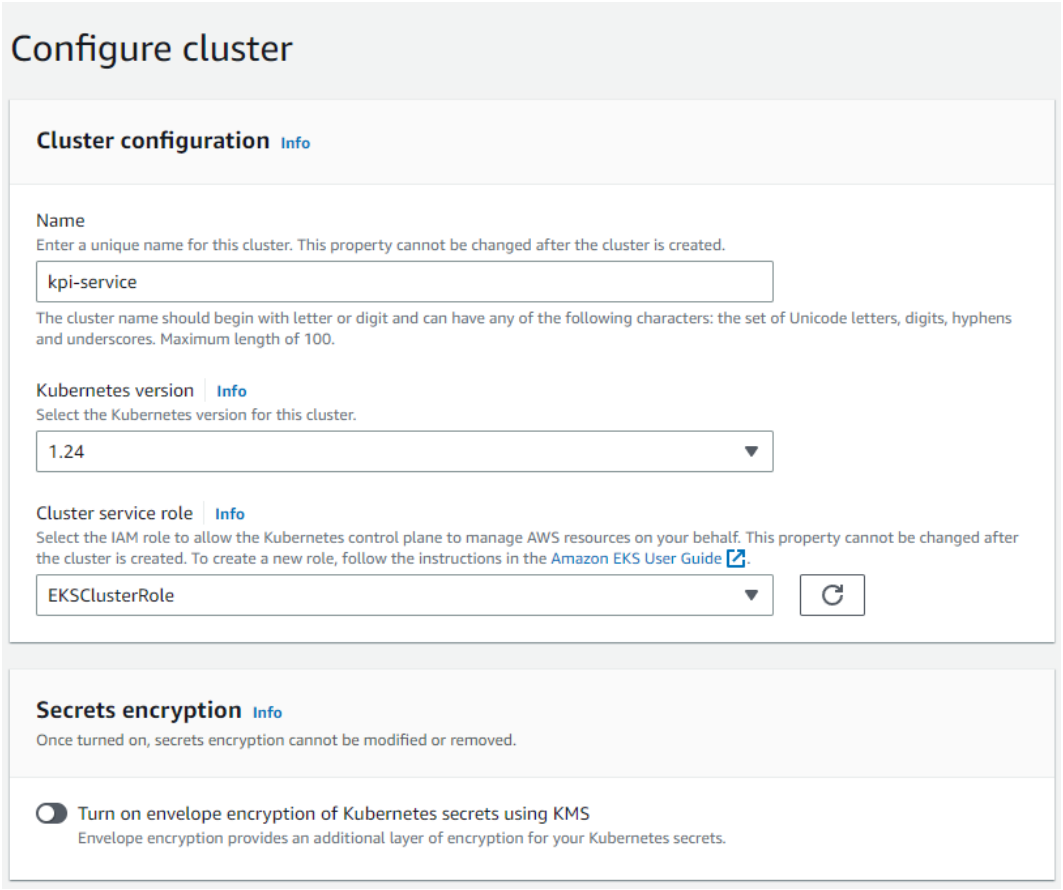


Рисунок 3.5 – Створення та початкове налаштування кластеру EKS

Наступним кроком є створення робочого вузла для Kubernetes. Для цього потрібно створити нову роль для EC2. Повторюючи вищеназвані кроки, створюємо нову роль із політиками AmazonEKSWorkerNodePolicy, AmazonEC2ContainerRegistryReadOnly та AmazonEKS_CNI_Policy.

Переходимо до створення нового робочого вузла. Знаходимо вкладку Compute на сторінці створеного кластеру kpi-service. Натискаємо Add node group.

Відкриється веб-сторінка для створення нової групи вузлів. Даємо назву новій групі робочих вузлів та вказуємо створену для вузлів роль. Наступний крок містить налаштування ресурсів для вузла. Якщо немає потреб у достатній обчислювальній потужності, рекомендується обрати мінімально можливі значення для економії коштів. Після вказання необхідних значень сторінка буде мати вигляд, вказаний на рисунку 3.6. Мережеві налаштування лишаємо за замовчуванням. Масштабування вказуємо мінімальне.

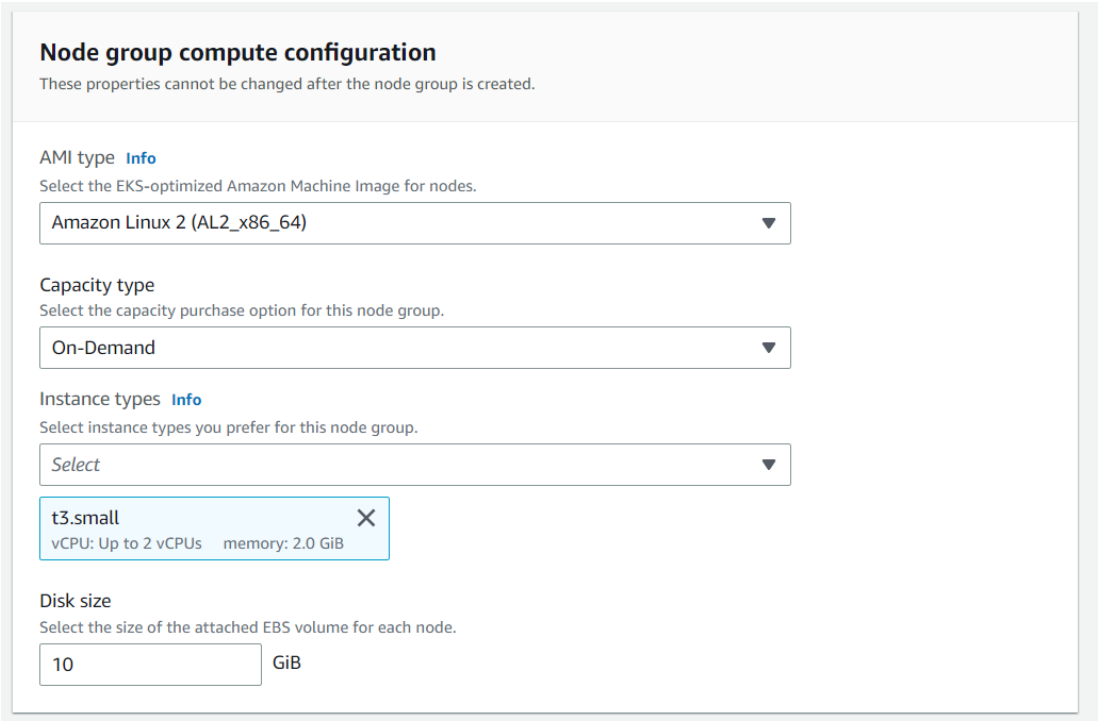


Рисунок 3.6 – Виділення ресурсів для групи робочих вузлів

Після усіх проведених налаштувань буде створена група вузлів із ім’ям kpi-service-node-group, а також робочий вузол, як це показано на рисунку 3.7.

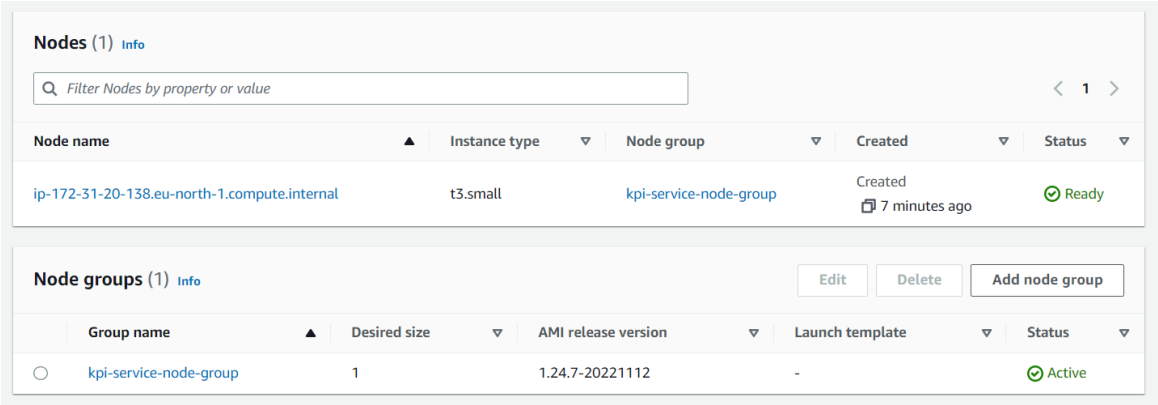


Рисунок 3.7 – Група вузлів та виділений робочий вузол

3.3.2 Розгортання бази даних

Найзручнішим способом розгортання ресурсів на Kubernetes є декларативний спосіб, тобто такий, де одразу описано весь бажаний результат. Kubernetes використовує спеціально сформовані yaml-файли, які містять повний опис ресурсу, що розгортається. Для бази даних необхідно два наступних ресурси: Service та Deployment. Вміст yaml-файлів для розгортання ресурсів бази даних наведено у додатку Г.

Для розгортання ресурсів бази даних зручно скористатися інструментом kubectl. Інструкції щодо налаштування AWS CLI та kubectl для роботи з EKS можна знайти за джерелом [50] та [51] відповідно.

На рисунку 3.8 зображено процес створення та перегляду ресурсів сервісу і розгортання на кластері Kubernetes у хмарному середовищі AWS. Ресурс сервісу надає IP, доступне всередині кластеру, а також у деяких випадах зовнішнє IP, доступне із інтернету. Ресурс розгортання відповідає за розгортання програми на кластері. Якщо READY чи AVAILABLE має значення нуль із певної кількості, це може свідчити про дві речі: або розгортання ще не завершено, або створені за допомогою розгортання поди не можуть запуститися.

```
PS C:\Users\makar\Documents\Університет\Anton-Makarenko\database> kubectl apply -f deployment.yml
deployment.apps/kpi-database created
PS C:\Users\makar\Documents\Університет\Anton-Makarenko\database> kubectl apply -f service.yml
service/kpi-database-service created
PS C:\Users\makar\Documents\Університет\Anton-Makarenko\database> kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
kpi-database   0/1     1            0           19s
PS C:\Users\makar\Documents\Університет\Anton-Makarenko\database> kubectl get service
NAME                   TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kpi-database-service   ClusterIP   10.100.250.1   <none>        1433/TCP   23s
kubernetes             ClusterIP   10.100.0.1     <none>        443/TCP    44h
PS C:\Users\makar\Documents\Університет\Anton-Makarenko\database> kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
kpi-database   1/1     1            1           57s
PS C:\Users\makar\Documents\Університет\Anton-Makarenko\database>
```

Рисунок 3.8 – Процес створення та перегляду ресурсів для роботи бази даних на кластері Kubernetes

3.3.3 Розгортання веб-додатку

Розгортання веб-додатку відбувається аналогічно до розгортання бази даних. Відмінність полягає у тому, що для веб-додатку потрібен інший тип сервісу. Для бази даних достатнім був звичайний сервіс, що визначав статичний ClusterIP. Доступ до веб-додатку потрібен не лише зсередини кластера, а й з Інтернету, тому для веб-додатку використовується сервіс типу LoadBalancer, який створює зовнішню IP-адресу. Вміст yaml-файлу, що описує конфігурацію сервісу для веб-додатку, а також вміст yaml-файлу для розгортання веб-додатку наведено у додатку Ґ.

На рисунку 3.9 зображено процес створення і перегляду ресурсів сервісу і розгортання для веб-додатку. Можна помітити, що був створений EXTERNAL-IP, який є DNS-адресою, за якою можна перейти і надіслати запит до веб-додатку.

```
PS C:\Users\makar\Documents\Університ\Anton-Makarenko> kubectl apply -f deployment.yml
deployment.apps/kpi-web-service created
PS C:\Users\makar\Documents\Університ\Anton-Makarenko> kubectl apply -f service.yml
service/kpi-web-service-service created
PS C:\Users\makar\Documents\Університ\Anton-Makarenko> kubectl get deployment
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
kpi-database      1/1     1            1           95s
kpi-web-service   0/1     1            0           13s
PS C:\Users\makar\Documents\Університ\Anton-Makarenko> kubectl get service
NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)        AGE
kpi-database-service      ClusterIP      10.100.250.1     <none>                                                                   1433/TCP       89s
kpi-web-service-service   LoadBalancer   10.100.154.208   a955aa3ea8eac436ca6d948eb52cd2dd-225615108.eu-north-1.elb.amazonaws.com   80:31947/TCP   8s
kubernetes                ClusterIP      10.100.0.1       <none>
          443/TCP        45h                    kubectl get deployment
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
kpi-database      1/1     1            1           111s
kpi-web-service   1/1     1            1           29s
PS C:\Users\makar\Documents\Університ\Anton-Makarenko>
```

Рисунок 3.9 – Процес створення та перегляду ресурсів для роботи веб-додатку на кластері Kubernetes

DNS-адреси, що генеруються хмарою Amazon, є незручними для звичайних користувачів. Щоб виправити даний недолік необхідно зареєструвати власне доменне ім’я та створити CNAME, що буде вказувати на нього. На рисунку 3.10 зображено записи DNS для antonmakarenko.site.

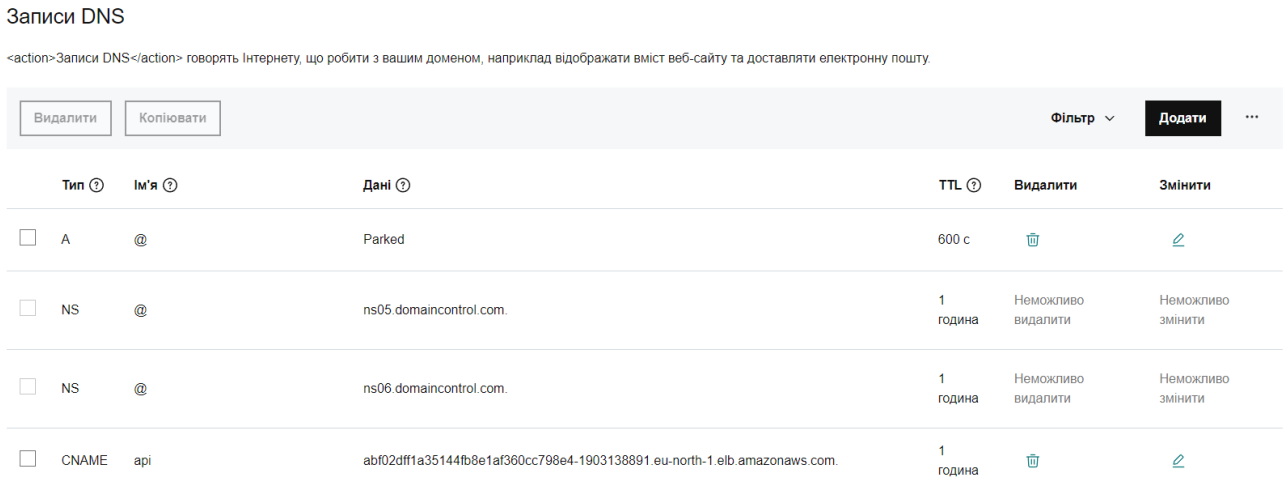


Рисунок 3.10 – Записи DNS для доменного імені antonmakarenko.site

Висновки до розділу 3

У даному розділі було ретельно описано архітектуру веб-сервісу, а також його програмну реалізацію, взаємодію із базою даних та розгортання за допомогою технології Kubernetes у хмарному середовищі Amazon Web Services.

Архітектурним паттерном для веб-сервісу обчислення показників МНТС організації було обрано Model-View-Controller. MVC дозволяє зручно розділити програму на частини. Частина Model відповідає за взаємодію із моделлю даних, які зберігаються у базі даних. Частина View відповідальна за перетворення отриманих із Model даних у той вид, який зручно подати користувачу (HTML-сторінка, JSON-об'єкт тощо). Частина Controller відповідальна за взаємодію користувача із системою, обробку запиту та виконання необхідної бізнес-логіки додатку.

Веб-додаток складається із наступних пакетів: controller, dto, entity, repository, service. Пакет controller містить два класи-контролери, за допомогою яких будуються кінцеві точки веб-додатку. Пакет dto містить класи, поля яких містять обчислені у відповідь на запит користувача значення і об'єкти яких повертаються користувачеві у вигляді JSON. Пакет entity містить класи, об'єкти яких міститимуть ті ж значення, що і таблиці бази даних, і чия структура повторює структуру зазначених таблиць бази даних. Пакет repository містить інтерфейси, за допомогою яких відбувається взаємодія із базою даних. Пакет service містить інтерфейси та класи-реалізації, чиї методи містять бізнес-логіку додатку. Класи та інтерфейси усіх пакетів, за винятком пакету dto, активно використовують інструменти, що надає фреймворк Spring.

У якості середовища, де буде працювати веб-сервіс, було обрано хмарне середовище Amazon Web Services. Було створено окремий кластер EKS, на якому були виділені ресурси для окремого робочого вузла, де будуть працювати компоненти веб-сервісу. У процесі створення кластеру та робочого вузла були налаштовані необхідні для їх роботи ролі.

Для розгортання компонентів веб-сервісу використовувалися інструменти AWS CLI та kubectl. Були створені service.yml та deployment.yaml для опису конфігурації для бази даних та веб-додатку. За допомогою AWS CLI та kubectl розгортання та сервіс були задіяні у середовищі AWS.

4 РОБОТА КОРИСТУВАЧА ІЗ СИСТЕМОЮ

4.1 Системні вимоги та запуск

Для взаємодії користувача із системою жодних особливих вимог немає. Використовувати можна будь-який обчислювальний пристрій: стаціонарний комп'ютер, ноутбук, мобільний телефон тощо. Головною вимогою до пристрою є можливість підключатися до інтернету, здатність надсилати запити та отримувати й оброблювати відповіді.

Для роботи веб-сервісу потрібен серверний комп'ютер із достатньою потужністю та можливістю масштабування на випадок збільшення оброблювальної інформації та/або збільшення потоку користувачів на сервер. Обом варіантам задовільняє використання послуг хмарного провайдера, у тому числі AWS.

4.2 Вхід у систему

Веб-сервіс є відкритим для будь-якого користувача, тому для входу у систему не потрібно вносити фінанси для оплати роботи із веб-сервісом чи реєструвати акаунт у системі.

Для використання веб-сервісу достатньо мати будь-який веб-клієнт: веб-браузер, платформа для API Postman, самостійно написаний клієнт на технологіях Rust Yew, JavaScript React, Java Swing із включеними бібліотеками для роботи із веб тощо. У цілях демонстрації роботи веб-сервісу була обрана платформа для API Postman через її гнучкість, широкий діапазон доступних функцій та активну підтримку зі сторони розробників. Postman містить платні тарифи, але навіть безкоштовного функціоналу буде достатньо для демонстрації роботи програмного продукту.

4.3 Демонстрація роботи веб-сервісу

Для перегляду показників конкретної організації використовується URL api.antonmakarenko.site/organization/<назва організації> (або <згенерований через AWS URL>/organization/<назва організації>). Якщо не вказувати який показник необхідно повернути у відповіді на запит, то поверненим буде об’єкт JSON, який буде містити усі обчислені показники, що продемонстровано на рисунку 4.1.

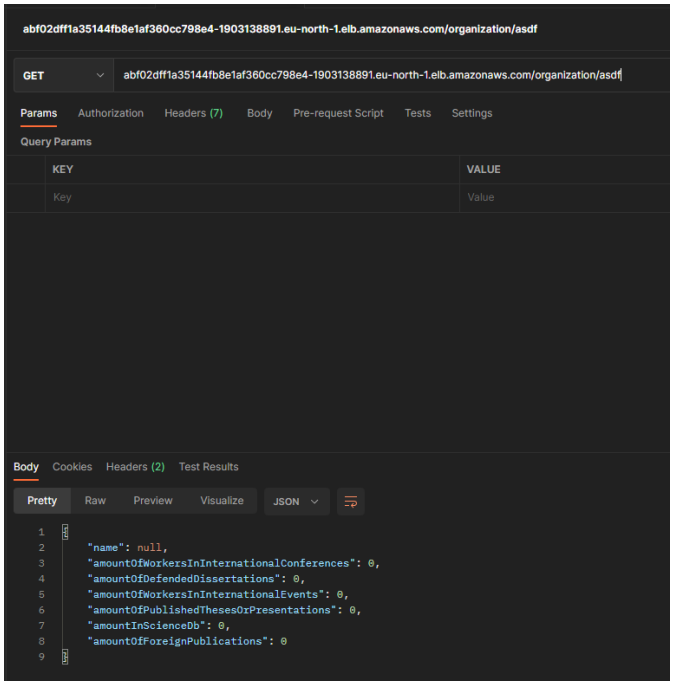


Рисунок 4.1 – Запит до кінцевої точки <AWS URL>/organization/<назва> та отримана відповідь

Для перегляду показників конкретної організації використовується URL api.antonmakarenko.site/worker/<ім’я>-<прізвище> (або <згенерований через AWS URL>/worker/<ім’я>-<прізвище>). Якщо не вказувати який показник необхідно повернути у відповіді на запит, то поверненим буде об’єкт JSON, який буде містити усі обчислені показники, що продемонстровано на рисунку 4.2.

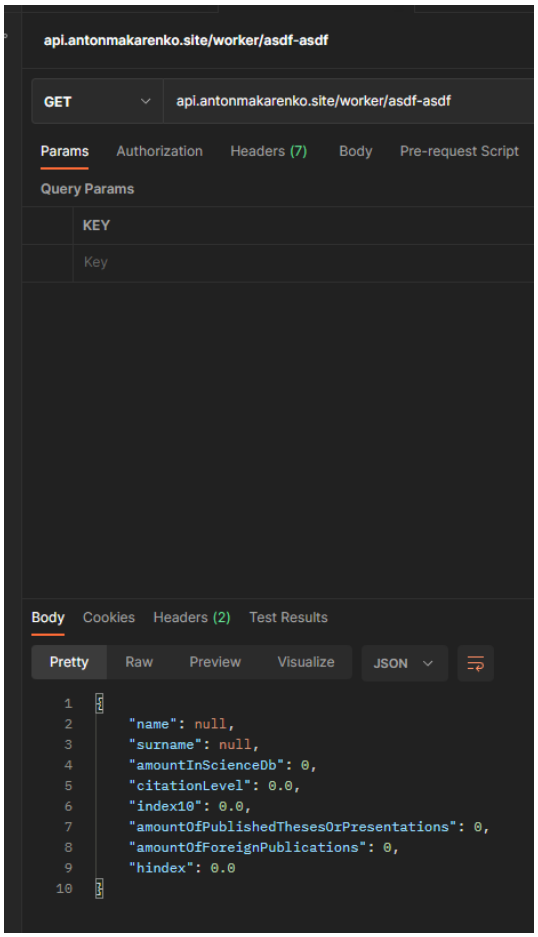


Рисунок 4.2 – Запит до кінцевої точки api.antonmakarenko.site/worker/asdf-asdf та отримана відповідь

У таблиці 4.1 наведено URL, за якими можна отримати той чи інший обчислений показник для організації або для конкретного співробтника. Варто відмітити, що api.antonmakarenko.site може бути замінений на AWS URL для будь-якого із зазначених URL.

Таблиця 4.1 – URL для отримання показників рівня міжнародної взаємодії організації.

URL	Показник	Тип поверненого значення
api.antonmakarenko.site/organization/<name>/amount-of-workers-in-international-conferences	Кількість співробітників установи, що приймали участь у міжнародних конференціях	Ціле число
api.antonmakarenko.site/organization/<name>/amount-of-defended-dissertations	Кількість захищених дисертацій в установі	Ціле число
api.antonmakarenko.site/organization/<name>/amount-of-workers-in-international-events	Кількість співробітників установи, що приймали участь у міжнародних подіях	Ціле число
api.antonmakarenko.site/organization/<name>/amount-of-published-theses-or-presentations	Кількість опублікованих тез чи презентацій співробітниками установи	Ціле число
api.antonmakarenko.site/organization/<name>/amount-in-science-db	Кількість записів про співробітників установи у наукометричних базах даних	Ціле число
api.antonmakarenko.site/organization/<name>/amount-of-foreign-publications	Кількість закордонних публікацій співробітників установи	Ціле число

Продовження таблиці 4.1

api.antonmakarenko.site/worker/<name>-<surname>/amount-in-scientific-db	Кількість записів про науковця у наукометричних базах даних	Ціле число
api.antonmakarenko.site/worker/<name>-<surname>/h-index	Індекс Гірша науковця	Десяткове число
api.antonmakarenko.site/worker/<name>-<surname>/citation-level	Рівень цитування науковця	Десяткове число
api.antonmakarenko.site/worker/<name>-<surname>/index-10	Індекс i10 науковця	Десяткове число
api.antonmakarenko.site/worker/<name>-<surname>/ amount-of-published-theses-or-presentations	Кількість опублікованих тез чи презентацій науковця	Ціле число
api.antonmakarenko.site/worker/<name>-<surname>/ amount-of-foreign-publications	Кількість закордонних публікацій науковця	Ціле число

Висновки до розділу 4

У даному розділі була продемонстровна робота веб-сервісу. Були надані мінімальні вимоги для взаємодії користувача із веб-сервісом. Мінімальними вимогами є наявність обчислювального пристрою із можливістю підключення до мережі Інтернет.

Були показані приклади роботи веб-сервісу, а також наданий детальний список із URL-адресами для отримання тих чи інших показників для організації чи конкретного співробітника.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

У таблиці 5.1 наведено опис ідеї стартап-проекту.

Таблиця 5.1 – Ідея стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмне забезпечення для обчислення показників рівня МНТС наукової організації	Аналіз впливу наукової установи на розвиток світової науки	Отримання інформації про значущість наукової установи та її досліджень
	Аналіз ефективності науковців, що працюють у науковій установі	Отримання інформації про наукові рейтинги окремих науковців

Існують аналоги подібних систем. Одні аналоги надають розширений функціонал лише за внесення коштів. Інші аналоги є вузькоспеціалізованими і не покривають більшу частину напрямів наукової діяльності.

У таблиці 5.2 наведено порівняння розробленого веб-сервісу із аналогічними системами.

Таблиця 5.2 – Порівняння створеного веб-сервісу із існуючими системами

Система	URL	Наявність платних послуг	Можливість реєстрації акаунту
Розроблений веб-сервіс	antonmakarenko.site	Ні	Ні
Google Scholar	scholar.google.com	Ні	Ні*
Scopus	www.scopus.com	Так	Так
Web of Science	access.clarivate.com	Так	Так

* інформація береться із акаунту Google

5.2 Технологічний аудит проекту

Для здійснення технологічного аудиту необхідно визначити які технології використовуються для створення та обслуговування веб-сервісу і який рівень їхньої доступності. У таблиці 5.3 наведено результати аналізу технологій, що використовуються.

Таблиця 5.3 – Технології, що використовуються для побудови веб-сервісу та його роботи

Технологія	Опис	Доступність
Середовище розробки IntelliJ Idea	Інтегроване середовище розробки, що дозволяє писати, відлагоджувати та керувати життєвим циклом програм, написаних на мові програмування Java.	IntelliJ Idea Community Edition є безкоштовною та надає базовий функціонал для роботи із Java-додатками. IntelliJ Idea Ultimate містить розширений функціонал для зручної роботи із поширеними фреймворками та базами даних. IntelliJ Idea Ultimate коштує близько 600€ на рік або 60€ на місяць [53].
Програмний каркас Spring	Фреймворк для платформи Java. Підтримує інверсію керування та надає зручний інструментарій для роботи із компонентами веб-сервісу.	Spring Framework є проектом із відкритим вихідним кодом, доступний для безкоштовного використання будь-ким.

Продовження таблиці 5.3

База даних Microsoft SQL Server	Система зберігання даних, що розроблена та підтримується компанією Microsoft.	Залежно від версії, коштує від 0$ до 13 748$ або від 0$ за рік до 5 434$ за рік [54].
Інструмент керування контейнерами Docker	Програмне забезпечення для керування ізольованими контейнерами. Дозволяє створювати та розгортати написаний програмний продукт у середовищі контейнера.	Залежно від версії коштує від 0$ за місяць на одну людину до 24$ за місяць на одну людину [55].
Оркестратор контейнерів Kubernetes	Програмне забезпечення для управління життєвими циклами контейнеризованих додатків та їх ресурсів.	Kubernetes є програмним забезпеченням із відкритим кодом, доступним для безкоштовного використання.
Сервіс хмарних обчислень Amazon Web Services	Хмарна платформа, що надає користувачам обчислювальні потужності залежно від їх потреб	Цінова політика залежить від конкретного сервісу AWS. Детальніше із вартістю послуг можна ознайомитися за джерелом [56].

Залежно від бюджету проекту, а також особливих вимог до системи обираються необхідні тарифні плани вищезазначених технологій.

5.3 Аналіз ринкових можливостей стартап-проекту

Для аналізу ринкових можливостей програмного продукту необхідно у першу чергу визначити цільову аудиторію веб-сервісу та вимоги, які вона визначає.

У таблиці 5.4 наведено потенційних клієнтів для користування розробленим програмним забезпеченням та їхні вимоги.

Таблиця 5.4 – Цільова аудиторія розробленого веб-сервісу та її вимоги до програмного продукту

Цільова аудиторія	Мета використання	Вимоги до продукту
Комерційні підприємці та комерційні установи	Визначення ефективності наукової діяльності установи для можливого фінансування.	Зручний API, висока швидкодія, надійність та безперебійність.
Науковці та наукові установи	Визначення ефективності власної діяльності у науково-технічній сфері; визначення ефективності інших науковців чи наукових установ для розуміння можливостей потенційної співпраці.	Зручний API, висока швидкодія, доступна ціна для використання, достатня надійність та безперебійність.
Розробники стороннього програмного забезпечення	Створення програмного забезпечення із використанням API, яке надає веб-сервіс для подальшого комерційного чи некомерційного розповсюдження.	Зручний API, доступна ціна для використання, висока швидкодія, надійність та безперебійність.
Ентузіасти	Використання відкритого API для створення власних проектів за допомогою різного набору технологій та ресурсів.	Зручний API, низька ціна використання або повна її відсутність, достатня швидкодія, безперебійність та надійність.

Найменші вимоги до ціни користування веб-сервісом серед потенційних клієнтів висувають комерційні підприємці та комерційні установи, проте у них найвищі вимоги щодо швидкодії та надійності програмного продукту. Додаткові оптимізації для досягнення високої швидкодії та постійний моніторинг стану веб-сервісу можуть негативно вплинути на рентабельність продукту.

Найменші вимоги до веб-сервісу висувають ентузіасти, проте вони є тою групою клієнтів, які потенційно не готові платити за користування веб-сервісом.

Науковці, наукові установи та розробники стороннього програмного забезпечення є тими групами клієнтів, які, з одної сторони не висувають достатньо строгих вимог до роботи веб-сервісу, а з іншої – готові вносити достатні кошти за користування програмним продуктом.

Після визначення груп потенційних клієнтів необхідно визначити фактори загроз, що здатні перешкодити впровадженню веб-сервісу на ринок. У таблиці 5.5 вказаний список потенційних загроз, а також способи протидії ним.

Таблиця 5.5 – Фактори загроз

Загроза	Опис	Можлива реакція
Конкуренція	Можлива поява конкурентів.	Постійне удосконалення програмного продукту, його активна підтримка, додання нових компонентів та функцій, які виділяли б програмний продукт з-поміж його аналогів.
Зміна ринкових тенденцій	Поява більш досконалих програмних продуктів від конкурентів.	Оновлення програмного продукту, додання нового функціоналу чи повна заміна старого функціоналу на більш сучасний.

Продовження таблиці 5.5

Зменшення попиту	Зменшення попиту на програмний продукт.	Корекція цінової політики, зміна цільової аудиторії, адаптація старого функціоналу під нові реалії.
Зменшення репутації компанії	Дії конкурентів чи власні дії, що призводять до зменшення репутації компанії.	Виправлення власних недоліків, проведення рекламних кампаній з метою покращити репутацію компанії у суспільстві.

Окрім факторів загроз, існують фактори можливостей, що дозволяють покращити становище компанії та збільшити розповсюдження програмного продукту. У таблиці 5.6 наведено фактори можливостей для створеного веб-сервісу.

Таблиця 5.6 – Фактори можливостей

Можливість	Опис	Можлива реакція
Збільшення попиту	Збільшення попиту на програмний продукт.	Розповсюдження програмного продукту, вкладення додаткових коштів у маркетингові кампанії.
Відповідні тенденції ринку	Наукова спільнота потребує зручних та надійних інструментів для визначення ефективності наукової діяльності установ та окремих науковців.	Розповсюдження програмного продукту, вкладення додаткових коштів у маркетингові кампанії.

5.4 Розроблення ринкової стратегії програмного продукту

Для розроблення ринкової стратегії необхідно у першу чергу визначити стратегію охоплення ринку, тобто опис цільових груп потенційних споживачів. У таблиці 5.7 наведено аналіз цільових груп потенційних споживачів.

Таблиця 5.7 – Цільові групи потенційних споживачів

Цільова група	Опис потреби	Попит	Конкуренція	Складність входу у сегмент
Комерційні установи	Визначення рейтингів науковців та наукових установ для надання грантів на дослідження.	Високий	Помірно висока	Помірно висока
Наукові установи	Оцінка власної ефективності та ефективності інших наукових установ та науковців.	Високий	Висока	Середня
Розробники стороннього програмного забезпечення	Використання наданих API для створення стороннього програмного забезпечення.	Помірно високий	Середня	Помірно низька
Ентузіасти та аматори	Використання наданих API для створення стороннього програмного забезпечення.	Помірно низький	Низька	Низька

Групами із високим чи помірно високим попитом є комерційні та наукові установи, а також розробники стороннього програмного забезпечення, тому саме на них потрібно фокусуватися як на цільових аудиторіях.

Другим кроком є вибір стратегії конкурентної поведінки. У таблиці 5.8 наведено базову конкуренту поведінку.

Таблиця 5.8 – Визначення базової конкурентної поведінки

Чи є проект першопроходьцем на ринку?	Так
Чи є необхідність у пошуку нових споживачів?	Так
Чи є необхідність переманювати користувачів конкурентів?	Так
Чи є необхідність копіювання характеристик товарів конкурентів?	Так
Стратегія конкурентної поведінки	Зайняття конкурентної ніші

5.5 Розроблення маркетингової програми стартап-проекту

Для розробки маркетингової програми стартап-проекту необхідно сформувати маркетингову концепцію товару. У таблиці 5.9 наведено підсумки попереднього аналізу конкурентоспроможності товару.

Таблиця 5.9 – Ключові переваги концепції потенційного продукту

Потреба	Вигода від продукту	Ключові переваги
Зручність використання	Відсутність вимог щодо специфічних знань користувача.	Використання програмного забезпечення не вимагає від користувача додаткових дій для користування веб-сервісом .

Продовження таблиці 5.9

Інформація про показники МНТС	Можливість аналізу ефективності міжнародної діяльності наукової установи.	Простота та доступність системи.

Наступним кроком є розробка трирівневої маркетингової моделі продукту: ідея продукту, його фізичні складові та особливості процесу його надання. Інформація про трирівневу модель надано у таблиці 5.10.

Таблиця 5.10 – Трирівнева маркетингова модель продукту

Рівень	Опис
Ідея	Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері
Фізичні складові	Реалізовано веб-сервіс для обчислення показників МСНО за допомогою доступного фреймворку та реляційної бази даних
Надання та підтримка	Надання відбувається за допомогою хмарних сервісів AWS, постійне впровадження нових функцій

Захист ідеї власності може бути організований за допомогою захисту інтелектуальної власності або ноу-хау.

Цінові межі на програмний продукту залежать від набору функцій, які надаються веб-сервісом. Режим trial або тестовий режим на обмежений період часу є безкоштовним. Плата за режим із повним функціоналом може стягуватися помісячно малими сумами або одноразово великою сумою.

Висновки до розділу 5

У розділі було розглянуто пункти стартап-проекту: опис ідеї проекту, технологічний аудит проекту, аналіз ринкових можливостей проекту, розроблення ринкової стратегії проекту та розроблення маркетингової програми проекту.

ВИСНОВКИ

Під час роботи над магістерською дисертацією було досліджено показники рівня міжнародного співробітництва наукової організації у науково-технічній сфері. Було розроблено веб-сервіс, що програмно обчислює показники рівня міжнародної взаємодії.

Перший розділ магістерської дисертації містить опис досліджуваної проблеми та аналіз схожих до розроблювальної систем. До одних із найбільш поширених програмних продуктів можна віднести системи Scopus, Web of Science, Google Scholar. Scopus та Web of Science є системами, що містять платний функціонал та надають широке покриття наукових публікацій та робіт. Google Scholar є відкритою для користування системою, що індексує тексти наукових публікацій різних форматів та дисциплін. Також було визначено список додаткових показників, які обчислюватиме розроблювальний веб-сервіс.

Другий розділ описує технології, які використовуються для побудови веб-сервісу. Даними технологіями є програмний каркас Spring, реляційна база даних Microsoft SQL Server, засіб контейнеризації Docker, оркестратор контейнерів Kubernetes та хмарна служба Amazon Web Services. Програмний каркас Spring містить широкий набір інструментів, корисних для розробки веб-сервісів. Реляційна база Microsoft SQL Server має хорошу підтримку від корпорації Microsoft. Засіб контейнеризації Docker дозволяє створювати контейнеризовані додатки для їх зручного запуску. Оркестратор контейнерів Kubernetes гнучко керує життєвим циклом контейнеризованих додатків. AWS надає хмарні обчислення та сервіси за доступні ціни.

Третій розділ містить опис реалізації веб-сервісу. Архітектурним шаблоном для створення системи був обраний шаблон Model-View-Controller, який розділяє сервіс на три частини: Model, View, Controller. Для програмної реалізації використовувалися інструменти Spring Framework, що дозволяють із легкістю створювати та налаштовувати кожен із компонентів MVC. Було створено образи

бази даних та веб-додатку і вивантажено у репозиторій. Було написано файли конфігурацій для створення ресурсів, необхідних для розгортання програмного продукту за допомогою оркестратора контейнерів Kubernetes. Останнім кроком було розгортання за допомогою сервісів хмарного середовища Amazon Web Services.

Четвертий розділ містить системні вимоги до розгортання та користування веб-сервісом, а також описує його основні кінцеві точки і роботу із ними. Умовами до користування є наявність обчислювального пристрою із мінімальною потужністю та можливістю підключення до мережі Інтернет. Серверні потужності при розгортанні надаються службами Amazon Web Services. Кінцеві точки надають доступ користувача до обчислення конкретних показників або до обчислення усіх показників разом.

П’ятий розділ містить інформацію про концепцію стартап-проекту для розробленого веб-сервісу. У п’ятому розділі наведено опис ідеї проекту, технологічний аудит проекту, аналіз ринкових можливостей проекту, розроблення ринкової стратегії проекту та розроблення маркетингової програми проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. International Scientific Cooperation. Challenges and Predicaments. Options for Risk Assessment / typesetting: E. Bouma. Amsterdam : Royal Netherlands Academy of Arts and Sciences, 2014.

2. David Albright, Sarah Burkhard, Spencer Faragasso. Iranian Drones in Ukraine Contain Western Brand Components: Institute for Science and International Security, 2022. P. 6.

3. ІРАНСЬКИЙ МОХАДЖЕР - 6, БРОНЕТРАНСПОРТЕР BUSHMASTER, САУ КРАБ. URL: https://www.youtube.com/watch?v=MPU8ZrgU8lY.

4. Наукометричні показники. *Рейтинг сайтів КПІ ім. Ігоря Сікорського.* URL: https://webometr.kpi.ua/citation-data.

5. Індекси цитування та наукометрій БД. *Наукова бібліотека Національного університету кораблебудування імені адмірала Макарова.* URL: http://lib.nuos.edu.ua/%D1%96%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%B8-%D1%86%D0%B8%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D1%82%D0%B0-%D0%BD%D0%B0%D1%83%D0%BA%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D0%B9-%D0%B1%D0%B4/.

6. Jorge Hirsch. An index to quantify an individual's scientific research output : University of California, San Diego, 2005. P. 1.

7. Measuring Your Impact: Impact Factor, Citation Analysis, and other Metrics: Journal Impact Factor (IF). *Library of University of Illinois Chicago.* URL: https://researchguides.uic.edu/if/impact.

8. Measuring a journal's impact. URL: https://www.elsevier.com/authors/tools-and-resources/measuring-a-journals-impact.

9. Наукометричні бази даних. *Бібліотека Університету Ушинського*. URL: https://library.pdpu.edu.ua/index.php/pro-nas/naukovtsiam/275-naukometrichni-bazi-danikh.

10. The top list of academic research databases. URL: https://paperpile.com/g/academic-research-databases/.

11. Світові наукометричні бази даних. *Наукова бібліотека Чернівецького національного університету.* URL: http://library.chnu.edu.ua/?page=/ua/07services/04helpsci/0102scidb.

12. List of academic databases and search engines. URL: https://en.wikipedia.org/wiki/List_of_academic_databases_and_search_engines.

13. What is Scopus Preview? URL: https://service.elsevier.com/app/answers/detail/a_id/15534/supporthub/scopus/#tips/.

14. Coverage you can count on. URL: https://www.elsevier.com/solutions/scopus/how-scopus-works/content.

15. Scopus. *Науково-технічна бібліотека Тернопільського національного технічного університету імені Івана Пулюя.* URL: https://library.tntu.edu.ua/resources/peredplachenyj-dostup/scopus/.

16. Корекція профілів. *Науково-технічна бібліотека Тернопільського національного технічного університету імені Івана Пулюя.* URL: https://library.tntu.edu.ua/resources/peredplachenyj-dostup/scopus/korekcija-profiliv/.

17. Web of Science Core Collection. – URL: https://clarivate.com/cis/solutions/web-of-science-core-collection/.

18. Caroline Birkle, David Pendlebury, Joshua Schnell. Web of Science as a data source for research on scientific and scholarly activity : Institute for Scientific Information, London, 2019. P. 365-367.

19. About Google Scholar. URL: https://scholar.google.com/init/en/scholar/about.html.

20. Science Direct-ly into Google. URL: http://radar.oreilly.com/archives/2007/07/science-directly-into-google.html.

21. Michael Meng, Stephanie Steinhardt, Andreas Schubert. Application Programming Interface Documentation: What Do Software Developers Want? : *Journal of Technical Writing and Communication*, 2018. P. 296.

22. Web Framework Benchmark. – URL: https://www.techempower.com/benchmarks/#section=data-r21&test=fortune.

23. Java is the most used language in South Korea, China, and Germany. The Java share in South Korea is 53%, in China 47%, and in Germany 33%. URL: https://www.jetbrains.com/lp/devecosystem-2021/java/.

24. Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho. Pro Spring 5. An In-Depth Guide to the Spring Framework and Its Tools. Fifth edition : Apress, 2017. P. 1.

25. Introduction to Spring Framework. URL: https://www.geeksforgeeks.org/introduction-to-spring-framework/.

26. Introduction to Spring Framework. URL: https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/overview.html.

27. Java Spring Boot. *IBM Cloud Education*. URL: https://www.ibm.com/cloud/learn/java-spring-boot.

28. 2021 Developer Survey. URL: https://insights.stackoverflow.com/survey/2021.

29. MS SQL Server. – Tutorials Point, 2016. – P. 1.

30. Editions and supported features of SQL Server 2017. URL: https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2017?view=sql-server-2017.

31. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others. URL: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/.

32. What is Application Deployment? URL: https://www.vmware.com/topics/glossary/content/application-deployment.html.

33. Taking Snapshots of Virtual Machines. URL: https://docs.vmware.com/en/VMware-Workstation-Pro/16.0/com.vmware.ws.using.doc/GUID-81701CA5-F1D4-47F2-8CC2-B47388AFF6C1.html.

34. 1.10. Snapshots [Electronic resource]: Oracle. – Access: https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/snapshots.html.

35. Containerization. URL: https://www.ibm.com/cloud/learn/containerization.

36. Containers in the enterprise. Rapid enterprise adoption continues. – IBM, 2020. – P. 8.

37. Amazon Elastic Kubernetes Service (EKS). URL: https://aws.amazon.com/eks/.

38. Azure Kubernetes Service (AKS). URL: https://azure.microsoft.com/en-us/products/kubernetes-service/.

39. What is Docker? How Does it Work? URL: https://devopscube.com/what-is-docker/.

40. How Kubernetes works. URL: https://sensu.io/blog/how-kubernetes-works.

41. About AWS. URL: https://aws.amazon.com/about-aws/.

42. Glenn E. Krasner, Stephen T. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System : ParcPlace Systems, 1988. P. 2.

43. 18. Using the @SpringBootApplication Annotation. URL: https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-using-springbootapplication-annotation.html.

44. 5.1.3. Query Methods. URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.

45. Spring Data JPA @Query. URL: https://www.baeldung.com/spring-data-jpa-query.

46. Why Interfaces are recommend by Spring? URL: https://techsach.com/why-interfaces-are-recommend-by-spring-framework/.

47. Будівельник. URL: https://refactoring.guru/uk/design-patterns/builder.

48. Package java.util.stream. URL: https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html.

49. 4.2. Mono, an Asynchronous 0-1 Result. URL: https://projectreactor.io/docs/core/release/reference/index.html#mono.

50. Set up AWS Credentials and Region for Development. URL: https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-credentials.html.

51. Why can’t I connect to my Amazon EKS cluster? URL: https://aws.amazon.com/premiumsupport/knowledge-center/eks-cluster-connection/.

52. Доменні імена. URL: http://www.godaddy.com/uk-ua/%D0%B4%D0%BE%D0%BC%D0%B5%D0%BD.

53. Subscription options & Pricing. URL: https://www.jetbrains.com/idea/buy/#commercial.

54. How to license SQL Server. URL: https://www.microsoft.com/en-us/sql-server/sql-server-2019-pricing.

55. Pricing & Subscriptions. URL: https://www.docker.com/pricing/.

56. AWS Pricing. URL: https://aws.amazon.com/pricing/?aws-products-pricing.sort-by=item.additionalFields.productNameLowercase&aws-products-pricing.sort-order=asc&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all.

ДОДАТОК А

Програмний код інтерфейсів WorkerRepository, OrganizationRepository, CollectedDataRepository, AchievementRepository

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ТВ-12мп

Аркушів 2

2022

Інтерфейс WorkerRepository:

```
package kpi.ipze.tv.makarenko.repository;

import kpi.ipze.tv.makarenko.entity.Worker;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface WorkerRepository extends JpaRepository<Worker, String> {
    Optional<Worker> findByNameAndSurname(String name, String surname);
}
```

Інтерфейс OrganizationRepository:

```
package kpi.ipze.tv.makarenko.repository;

import kpi.ipze.tv.makarenko.entity.Organization;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface OrganizationRepository extends JpaRepository<Organization,
String> {
    Optional<Organization> findByName(String name);
}
```

Інтерфейс CollectedDataRepository:

```
package kpi.ipze.tv.makarenko.repository;

import kpi.ipze.tv.makarenko.entity.CollectedData;
import kpi.ipze.tv.makarenko.entity.CollectedDataId;
import kpi.ipze.tv.makarenko.entity.Worker;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Collection;

@Repository
public interface CollectedDataRepository extends JpaRepository<CollectedData,
CollectedDataId> {
    int countByWorkerIn(Collection<Worker> workers);
}
```

Інтерфейс AchievementRepository:

```
package kpi.ipze.tv.makarenko.repository;

import kpi.ipze.tv.makarenko.entity.Achievement;
import kpi.ipze.tv.makarenko.entity.Worker;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
```

```
import java.util.List;
import java.util.Set;

@Repository
public interface AchievementRepository extends JpaRepository<Achievement, String>
{
    @Query("SELECT COUNT(achievement) FROM Achievement achievement JOIN
achievement.workers worker WHERE worker IN :workers AND
UPPER(achievement.resultType.name) = UPPER(:resultTypeName) GROUP BY worker.id")
    List<Integer> countByWorkersAndResultTypeNameIgnoreCase(Set<Worker> workers,
String resultTypeName);

    @Query("SELECT COUNT(achievement) FROM Achievement achievement JOIN
achievement.workers worker WHERE worker IN :workers AND
achievement.publicationEvent LIKE CONCAT('%', UPPER(:publicationEvent), '%') GROUP
BY worker.id")
    List<Integer>
countByWorkersAndPublicationEventContainingIgnoreCase(Set<Worker> workers, String
publicationEvent);

    @Query("SELECT COUNT(achievement) FROM Achievement achievement JOIN
achievement.workers worker WHERE worker IN :workers AND
achievement.publicationEvent LIKE CONCAT('%', UPPER(:publicationEvent), '%') AND
UPPER(achievement.country.name) <> UPPER(:countryName) GROUP BY worker.id")
    List<Integer>
countByWorkersAndPublicationEventContainingIgnoreCaseAndNotCountryNameIgnoreCase(S
et<Worker> workers, String publicationEvent, String countryName);
}
```

ДОДАТОК Б

Програмний код класів-імплементацій для інтерфейсів OrganizationService та WorkerService

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ТВ-12мп

Аркушів 5

2022

Клас-імплементація для інтерфейсу OrganizationService:

```
package kpi.ipze.tv.makarenko.service.impl;

import kpi.ipze.tv.makarenko.dto.OrganizationParamsDto;
import kpi.ipze.tv.makarenko.entity.*;
import kpi.ipze.tv.makarenko.repository.AchievementRepository;
import kpi.ipze.tv.makarenko.repository.CollectedDataRepository;
import kpi.ipze.tv.makarenko.repository.OrganizationRepository;
import kpi.ipze.tv.makarenko.service.OrganizationService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
@RequiredArgsConstructor
public class OrganizationServiceImpl implements OrganizationService {
    private final OrganizationRepository organizationRepository;

    private final AchievementRepository achievementRepository;

    private final CollectedDataRepository collectedDataRepository;

    @Override
    public int amountOfWorkersInInternationalConferences(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return 0;
        Organization organization = optionalOrganization.get();
        int count = 0;
        for (Subhead subhead : organization.getSubheads()) {
            for (Team team : subhead.getTeams()) {
                count += achievementRepository

.countByWorkersAndPublicationEventContainingIgnoreCase(team.getWorkers(),
"international conference")
                        .stream()
                        .reduce(0, Integer::sum);
            }
        }
        return count;
    }

    @Override
    public int amountOfDefendedDissertations(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return 0;
        Organization organization = optionalOrganization.get();
        int count = 0;
        for (Subhead subhead : organization.getSubheads()) {
            for (Team team : subhead.getTeams()) {
                count += achievementRepository

.countByWorkersAndResultTypeNameIgnoreCase(team.getWorkers(), "dissertation")
                        .stream()
                        .reduce(0, Integer::sum);
            }
```

```
        }
        return count;
    }

    @Override
    public int amountOfWorkersInInternationalEvents(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return 0;
        Organization organization = optionalOrganization.get();
        int count = 0;
        for (Subhead subhead : organization.getSubheads()) {
            for (Team team : subhead.getTeams()) {
                count += achievementRepository

.countByWorkersAndPublicationEventContainingIgnoreCase(team.getWorkers(),
"international")
                        .stream()
                        .reduce(0, Integer::sum);
            }
        }
        return count;
    }

    @Override
    public int amountOfPublishedThesesOrPresentations(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return 0;
        Organization organization = optionalOrganization.get();
        int count = 0;
        for (Subhead subhead : organization.getSubheads()) {
            for (Team team : subhead.getTeams()) {
                count += achievementRepository

.countByWorkersAndResultTypeNameIgnoreCase(team.getWorkers(), "theses")
                        .stream()
                        .reduce(0, Integer::sum);
            }
        }
        for (Subhead subhead: organization.getSubheads()) {
            for (Team team: subhead.getTeams()) {
                count += achievementRepository

.countByWorkersAndResultTypeNameIgnoreCase(team.getWorkers(), "presentation")
                        .stream()
                        .reduce(0, Integer::sum);
            }
        }
        return count;
    }

    @Override
    public int amountInScienceDb(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return 0;
        Organization organization = optionalOrganization.get();
```

```
        int count = 0;
        for (Subhead subhead : organization.getSubheads()) {
            for (Team team : subhead.getTeams()) {
                count +=
collectedDataRepository.countByWorkerIn(team.getWorkers());
            }
        }
        return count;
    }

    @Override
    public int amountOfForeignPublications(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return 0;
        Organization organization = optionalOrganization.get();
        int count = 0;
        for (Subhead subhead : organization.getSubheads()) {
            for (Team team : subhead.getTeams()) {
                count += achievementRepository

.countByWorkersAndPublicationEventContainingIgnoreCaseAndNotCountryNameIgnoreCase(
team.getWorkers(), "publication", "ukraine")
                        .stream()
                        .reduce(0, Integer::sum);
            }
        }
        return count;
    }

    @Override
    public OrganizationParamsDto allParams(String name) {
        Optional<Organization> optionalOrganization =
organizationRepository.findByName(name);
        if (optionalOrganization.isEmpty())
            return OrganizationParamsDto.builder().build();
        return OrganizationParamsDto.builder()
                .name(name)
                .amountOfForeignPublications(amountOfForeignPublications(name))

.amountOfPublishedThesesOrPresentations(amountOfPublishedThesesOrPresentations(nam
e))
                .amountInScienceDb(amountInScienceDb(name))

.amountOfDefendedDissertations(amountOfDefendedDissertations(name))

.amountOfWorkersInInternationalEvents(amountOfWorkersInInternationalEvents(name))

.amountOfWorkersInInternationalConferences(amountOfWorkersInInternationalConferenc
es(name))
                .build();
    }
}
```

Клас-імплементація для інтерфейсу WorkerService:

```
package kpi.ipze.tv.makarenko.service.impl;

import kpi.ipze.tv.makarenko.dto.WorkerParamsDto;
```

```
import kpi.ipze.tv.makarenko.entity.Worker;
import kpi.ipze.tv.makarenko.repository.AchievementRepository;
import kpi.ipze.tv.makarenko.repository.CollectedDataRepository;
import kpi.ipze.tv.makarenko.repository.WorkerRepository;
import kpi.ipze.tv.makarenko.service.WorkerService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.Set;

@Service
@RequiredArgsConstructor
public class WorkerServiceImpl implements WorkerService {
    private final WorkerRepository workerRepository;

    private final CollectedDataRepository collectedDataRepository;

    private final AchievementRepository achievementRepository;

    @Override
    public int amountInScienceDb(String name, String surname) {
        Optional<Worker> optionalWorker =
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return 0;
        Worker worker = optionalWorker.get();
        return collectedDataRepository.countByWorkerIn(Set.of(worker));
    }

    @Override
    public float hIndex(String name, String surname) {
        Optional<Worker> optionalWorker =
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return 0;
        return optionalWorker.get().getHIndex();
    }

    @Override
    public float citationLevel(String name, String surname) {
        Optional<Worker> optionalWorker =
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return 0;
        return optionalWorker.get().getCitationLevel();
    }

    @Override
    public float index10(String name, String surname) {
        Optional<Worker> optionalWorker =
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return 0;
        return optionalWorker.get().getIndex10();
    }

    @Override
    public int amountOfPublishedThesesOrPresentations(String name, String surname)
{
        Optional<Worker> optionalWorker =
```

```
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return 0;
        Worker worker = optionalWorker.get();
        //maybe stream?
        return
achievementRepository.countByWorkersAndResultTypeNameIgnoreCase(Set.of(worker),
"theses").get(0)
                +
achievementRepository.countByWorkersAndResultTypeNameIgnoreCase(Set.of(worker),
"presentation").get(0);
    }

    @Override
    public int amountOfForeignPublications(String name, String surname) {
        Optional<Worker> optionalWorker =
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return 0;
        Worker worker = optionalWorker.get();
        //maybe stream?
        return achievementRepository

.countByWorkersAndPublicationEventContainingIgnoreCaseAndNotCountryNameIgnoreCase(
Set.of(worker), "publication", "ukraine").get(0);
    }

    @Override
    public WorkerParamsDto allParams(String name, String surname) {
        Optional<Worker> optionalWorker =
workerRepository.findByNameAndSurname(name, surname);
        if (optionalWorker.isEmpty())
            return WorkerParamsDto.builder().build();
        return WorkerParamsDto.builder()
                .name(name)
                .surname(surname)

.amountOfPublishedThesesOrPresentations(amountOfPublishedThesesOrPresentations(nam
e, surname))
                .amountOfForeignPublications(amountOfForeignPublications(name,
surname))
                .hIndex(hIndex(name, surname))
                .index10(index10(name, surname))
                .citationLevel(citationLevel(name, surname))
                .amountInScienceDb(amountInScienceDb(name, surname))
                .build();
    }
}
```

ДОДАТОК В

Програмний код класів OrganizationController та WorkerController

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ТВ-12мп

Аркушів 2

2022

Клас OrganizationController:

```
package kpi.ipze.tv.makarenko.controller;

import kpi.ipze.tv.makarenko.dto.OrganizationParamsDto;
import kpi.ipze.tv.makarenko.service.OrganizationService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

@RestController
@RequestMapping("/organization")
@RequiredArgsConstructor
public class OrganizationController {
    private final OrganizationService organizationService;

    @GetMapping("/{name}/amount-of-workers-in-international-conferences")
    public Mono<Integer> amountOfWorkersInInternationalConferences(@PathVariable
String name) {
        return
Mono.just(organizationService.amountOfWorkersInInternationalConferences(name));
    }

    @GetMapping("/{name}/amount-of-defended-dissertations")
    public Mono<Integer> amountOfDefendedDissertations(@PathVariable String name)
{
        return Mono.just(organizationService.amountOfDefendedDissertations(name));
    }

    @GetMapping("/{name}/amount-of-workers-in-international-events")
    public Mono<Integer> amountOfWorkersInInternationalEvents(@PathVariable String
name) {
        return
Mono.just(organizationService.amountOfWorkersInInternationalEvents(name));
    }

    @GetMapping("/{name}/amount-of-published-theses-or-presentations")
    public Mono<Integer> amountOfPublishedThesesOrPresentations(@PathVariable
String name) {
        return
Mono.just(organizationService.amountOfPublishedThesesOrPresentations(name));
    }

    @GetMapping("/{name}/amount-in-science-db")
    public Mono<Integer> amountInScienceDb(@PathVariable String name) {
        return Mono.just(organizationService.amountInScienceDb(name));
    }

    @GetMapping("/{name}/amount-of-foreign-publications")
    public Mono<Integer> amountOfForeignPublications(@PathVariable String name) {
        return Mono.just(organizationService.amountOfForeignPublications(name));
    }

    @GetMapping("/{name}")
    public Mono<OrganizationParamsDto> allParams(@PathVariable String name) {
        return Mono.just(organizationService.allParams(name));
    }
}
```

Клас WorkerController:

```
package kpi.ipze.tv.makarenko.controller;

import kpi.ipze.tv.makarenko.dto.WorkerParamsDto;
import kpi.ipze.tv.makarenko.service.WorkerService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

@RestController
@RequestMapping("/worker")
@RequiredArgsConstructor
public class WorkerController {
    private final WorkerService workerService;

    @GetMapping("/{name}-{surname}/amount-in-scientific-db")
    public Mono<Integer> amountInScienceDb(@PathVariable String name,
@PathVariable String surname) {
        return Mono.just(workerService.amountInScienceDb(name, surname));
    }

    @GetMapping("/{name}-{surname}/h-index")
    public Mono<Float> hIndex(@PathVariable String name, @PathVariable String
surname) {
        return Mono.just(workerService.hIndex(name, surname));
    }

    @GetMapping("/{name}-{surname}/citation-level")
    public Mono<Float> citationLevel(@PathVariable String name, @PathVariable
String surname) {
        return Mono.just(workerService.citationLevel(name, surname));
    }

    @GetMapping("/{name}-{surname}/index-10")
    public Mono<Float> index10(@PathVariable String name, @PathVariable String
surname) {
        return Mono.just(workerService.index10(name, surname));
    }
    @GetMapping("/{name}-{surname}/amount-of-published-theses-or-presentations")
    public Mono<Integer> amountOfPublishedThesesOrPresentations(@PathVariable
String name, @PathVariable String surname) {
        return
Mono.just(workerService.amountOfPublishedThesesOrPresentations(name, surname));
    }
    @GetMapping("/{name}-{surname}/amount-of-foreign-publications")
    public Mono<Integer> amountOfForeignPublications(@PathVariable String name,
@PathVariable String surname) {
        return Mono.just(workerService.amountOfForeignPublications(name,
surname));
    }

    @GetMapping("/{name}-{surname}")
    public Mono<WorkerParamsDto> allParams(@PathVariable String name,
@PathVariable String surname) {
        return Mono.just(workerService.allParams(name, surname));
    }
}
```

ДОДАТОК Г

Вміст файлів-конфігурацій service.yml та deployment.yml для розгортання ресурсів бази даних у середовищі Kubernetes

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ТВ-12мп

Аркушів 1

2022

Файл service.yml:

```
apiVersion: v1
kind: Service
metadata:
  name: kpi-database-service
  labels:
    app: kpi-database
spec:
  clusterIP: 10.100.250.1
  selector:
    app: kpi-database
  ports:
    - protocol: TCP
      port: 1433
      targetPort: 1433
```

Файл deployment.yml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kpi-database
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kpi-database
  template:
    metadata:
      labels:
        app: kpi-database
    spec:
      containers:
        - name: kpi-database
          image: antonmakarenko/kpi-database:2019-CU17-ubuntu-20.04
          imagePullPolicy: Always
          ports:
            - containerPort: 1433
```

ДОДАТОК Ґ

Вміст файлів-конфігурацій service.yml та deployment.yml для розгортання ресурсів веб-додатку у середовищі Kubernetes

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ТВ-12мп

Аркушів 1

2022

Файл service.yml:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kpi-web-service-service
  labels:
    app: kpi-web-service
spec:
  type: LoadBalancer
  selector:
    app: kpi-web-service
  ports:
    - protocol: TCP
      name: http
      port: 80
      targetPort: 8080
```

Файл deployment.yml:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kpi-web-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kpi-web-service
  template:
    metadata:
      labels:
        app: kpi-web-service
    spec:
      containers:
        - name: kpi-web-service
          image: antonmakarenko/kpi-web-service:1.0-SNAPSHOT
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
```

ДОДАТОК Д

Презентація

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ТВ-12мп

Аркушів 28

2022

Національний технічний університет україни
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра Інженерії програмного забезпечення в енергетиці

Веб-сервіс обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері

виконав:
студент 6 курсу
групи ТВ-12мп
Макаренко Антон Олександрович

керівник:
доцент,
доктор технічних наук
Коваль Олександр Васильович

Київ-2022

Актуальність теми дослідження

Для максимальної ефективної діяльності у науковій сфері науковим установам необхідно взаємодіяти із іншими науковими установами по всьому світу. Обчислення показників рівня міжнародного співробітництва дозволяє оцінити вклад організації у світову науку.

Рівень ефективності наукової діяльності міжнародного співробітництва наукової організації в науково-технічній сфері визначається відповідними визначеними показниками.

Завдання

- Мета: дослідження показників рівня міжнародного співробітництва та реалізація веб-сервісу для їхнього обчислення.
- Об’єкт дослідження: інформаційні технології для побудови, розгортання та підтримки веб-сервісів.
- Предмет дослідження: веб-сервіс обчислення показників рівня міжнародної взаємодії наукової організації в науково-технічній сфері.

Наукове завдання

Дослідження, розробка та розгортання програмного продукту для обчислення показників рівня міжнародного співробітництва наукової організації в науково-технічній сфері.

Часткові наукові завдання:

1. Визначити яка інформація із бази даних може бути використана для обчислення показників.
2. Проаналізувати існуючі системи обчислення наукометричних показників.
3. Розробити відкритий API для
4. Створити веб-сервіс для обчислення визначених показників.

Параметри оцінки рівня

Визначені параметри оцінки рівня

- Кількість міжнародних конференцій, у яких установа прийняла участь.
- Кількість публікацій співробітників установи у зарубіжних виданнях.
- Кількість публікацій у наукометричних базах даних.
- Цитування у міжнародних бібліографічних базах.
- Кількість співавторських з іноземцями публікацій.
- Кількість міжнародних подій, у яких приймали участь співробітники організації.

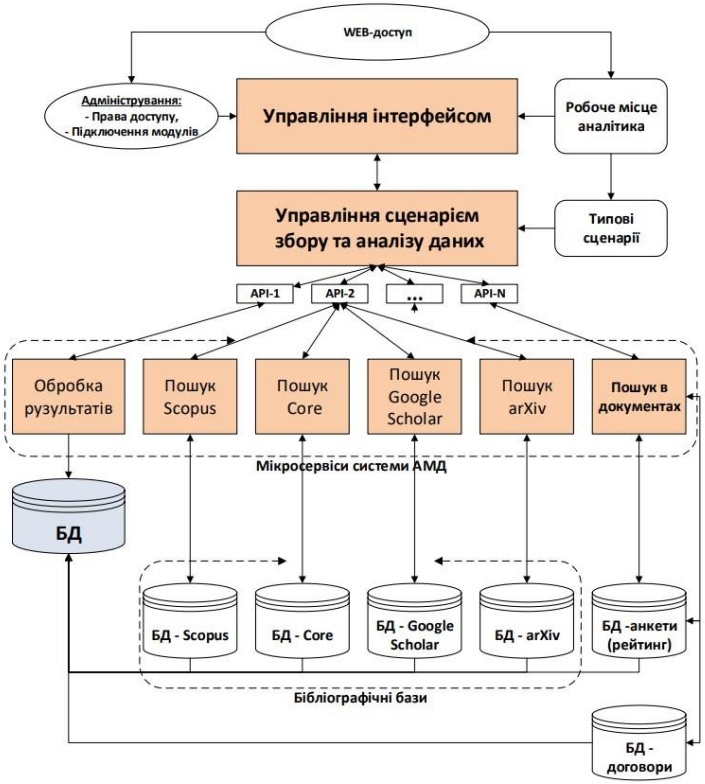
Існуючі системи обчислення показників

Scopus®

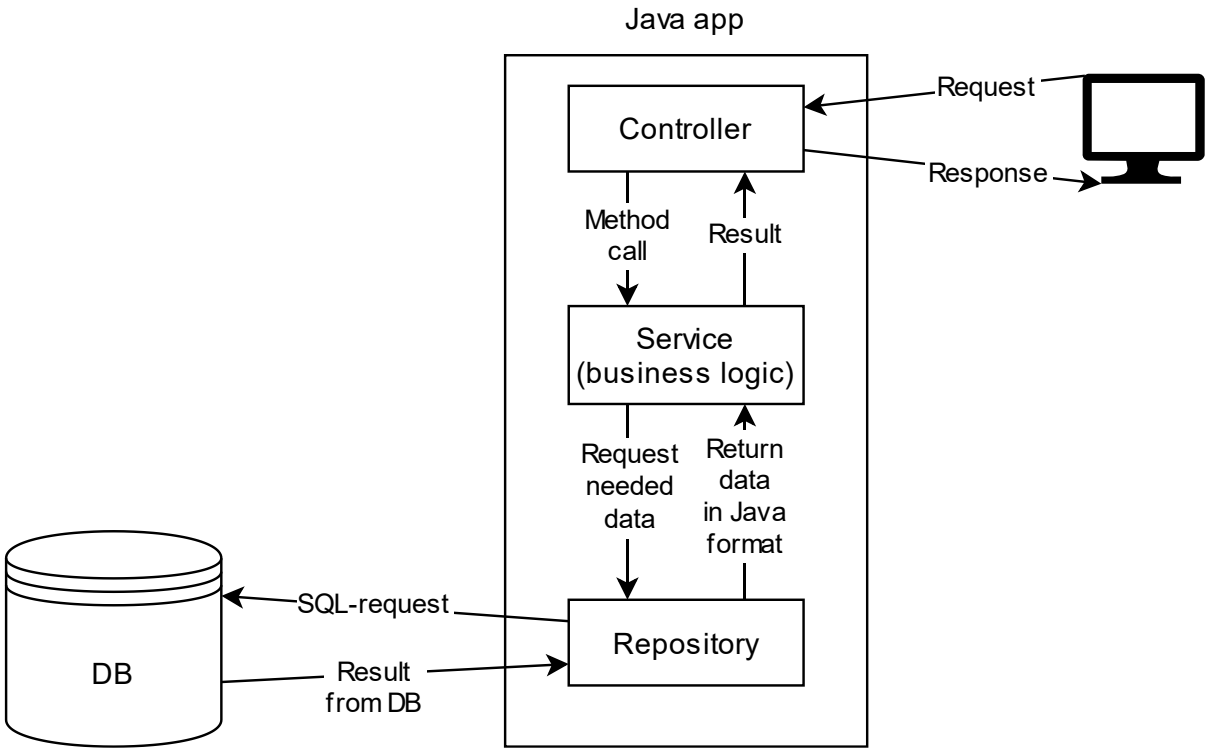
Clarivate Web of Science™

Google Академія

Архітектура системи



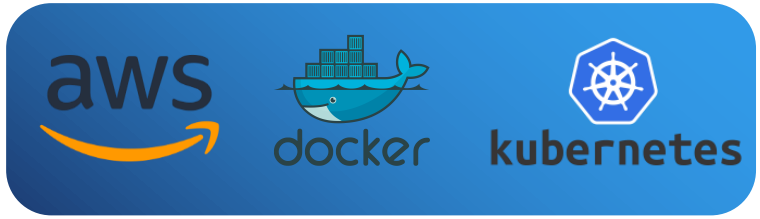
Архітектура системи



Засоби розробки

База даних

Прикладна програма



Розгортання

Kubernetes

Необхідні для розгортання веб-сервісу ресурси Kubernetes.

- Deployment
- Service

Deployment контролює життєвий цикл контейнеризованих програм.

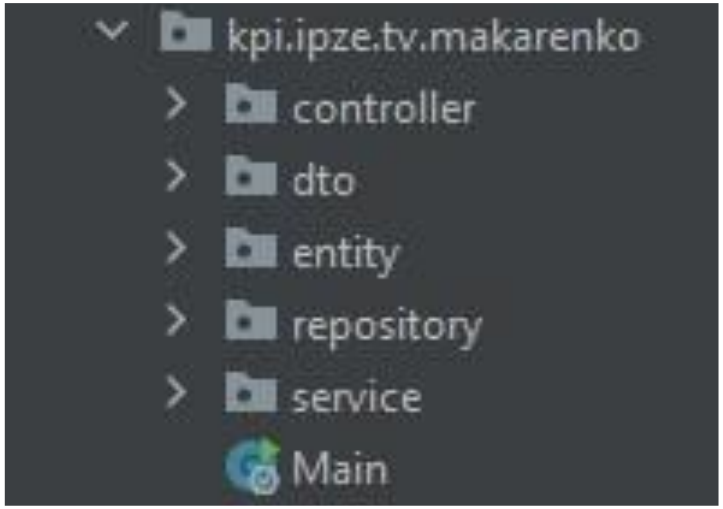
Service контролює мережевий доступ до програм.

Amazon Web Services

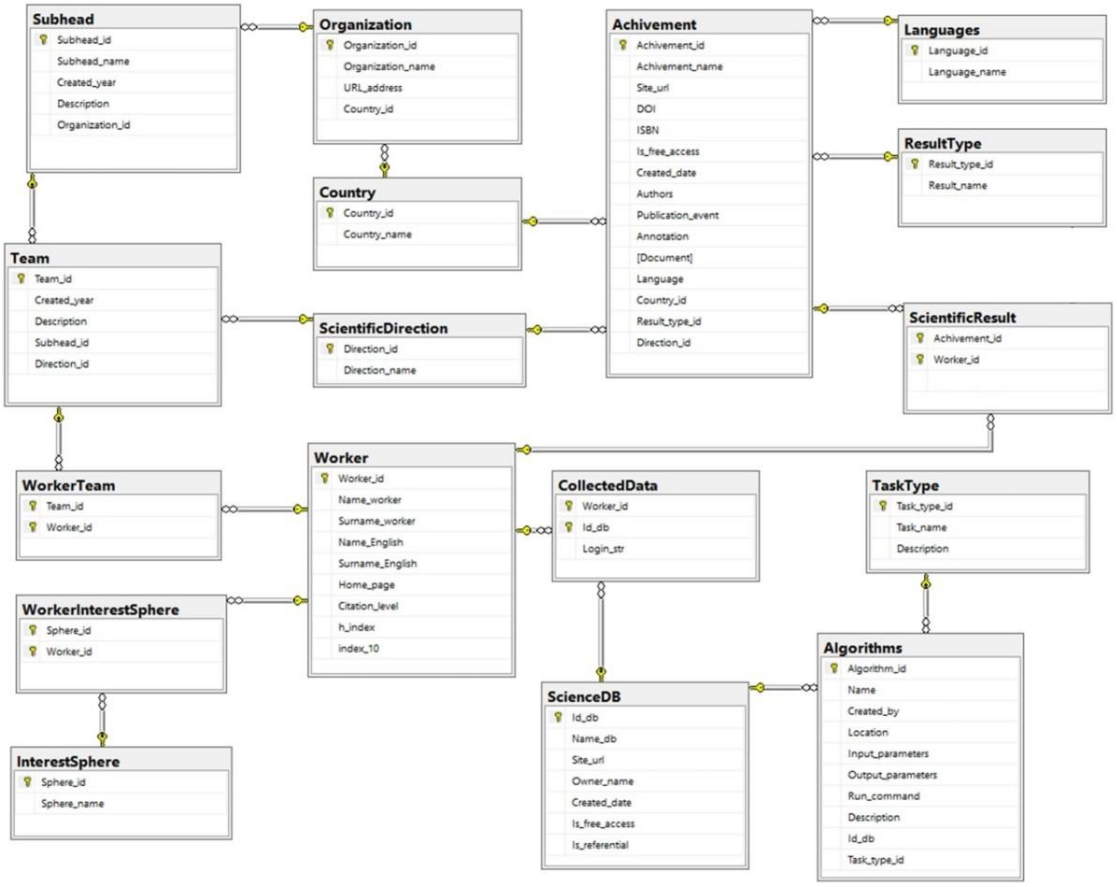
Необхідні для розгортання веб-сервісу ресурси AWS.

- Роль для EKS - Cluster.
- Роль для робочого вузла EC2.
- Кластер для EKS.
- Робочу групу (групу робочих вузлів).

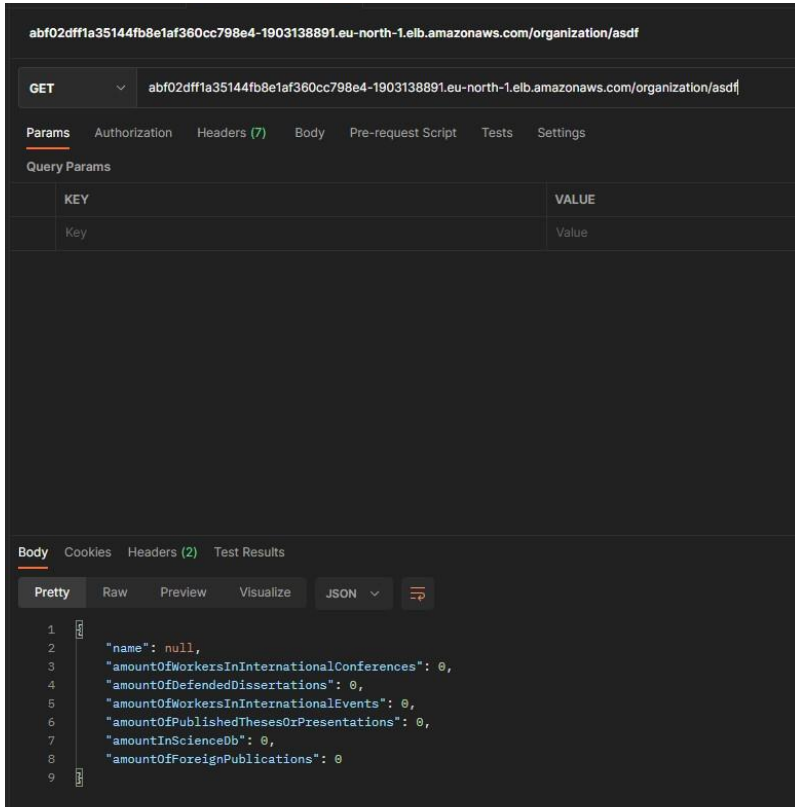
Структура веб-додатку



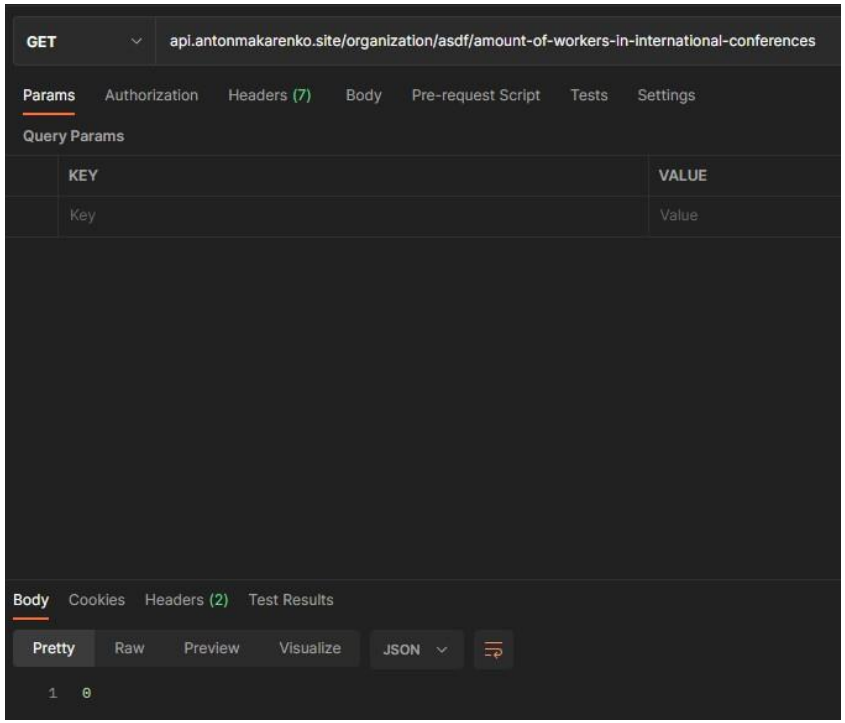
Структура бази даних



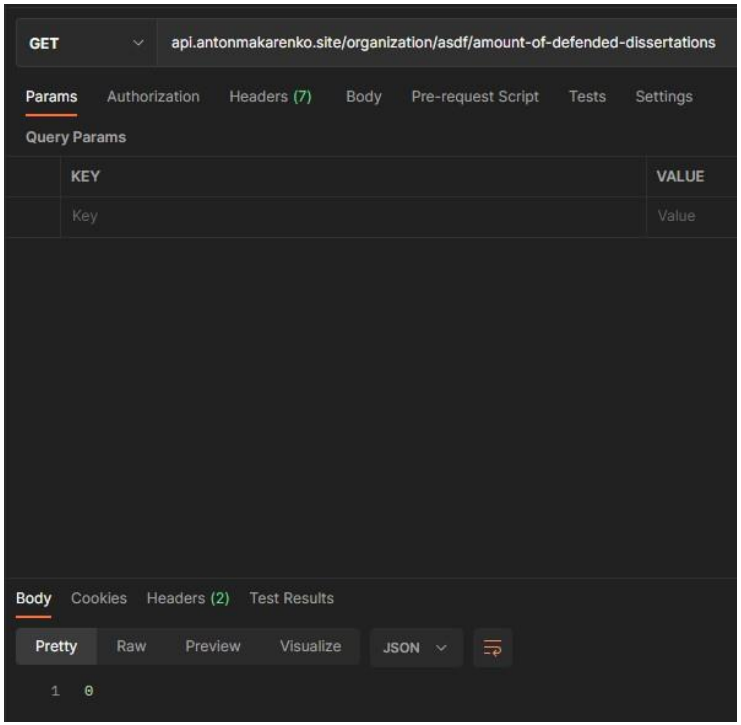
Демонстрація роботи



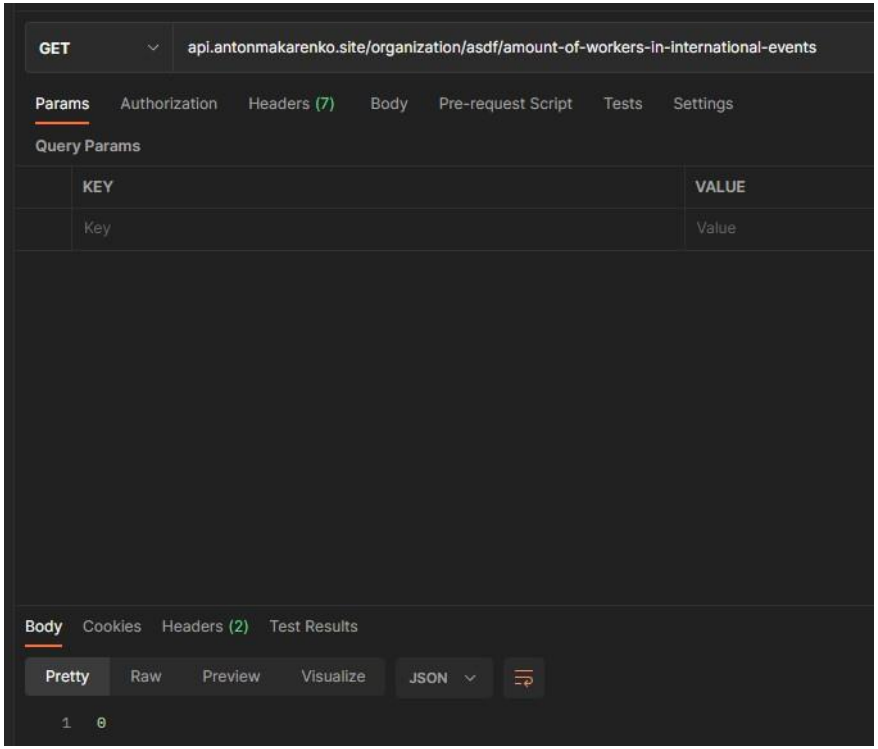
Демонстрація роботи



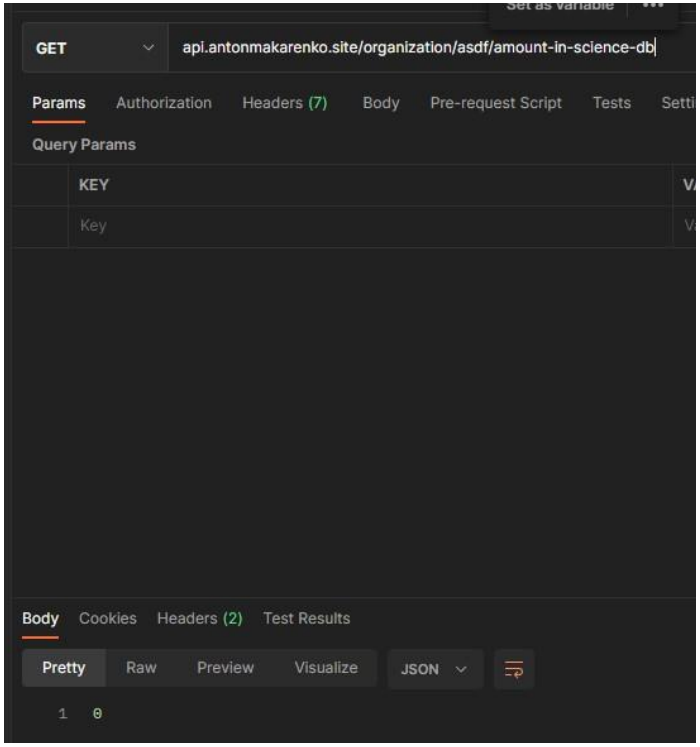
Демонстрація роботи



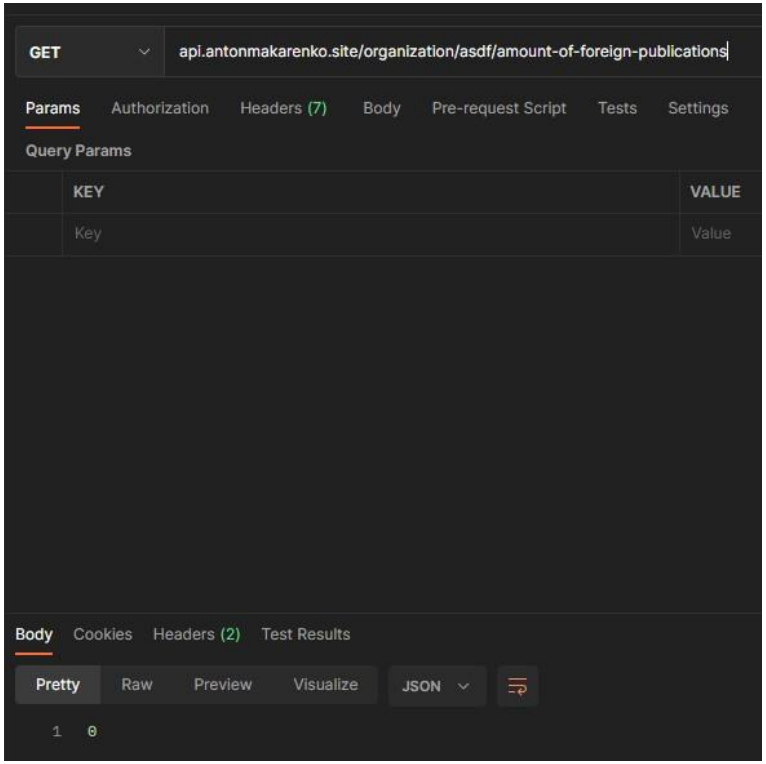
Демонстрація роботи



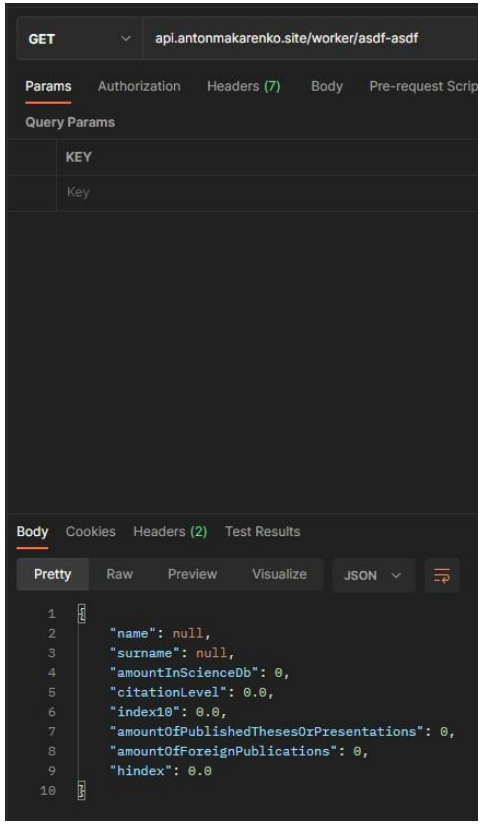
Демонстрація роботи



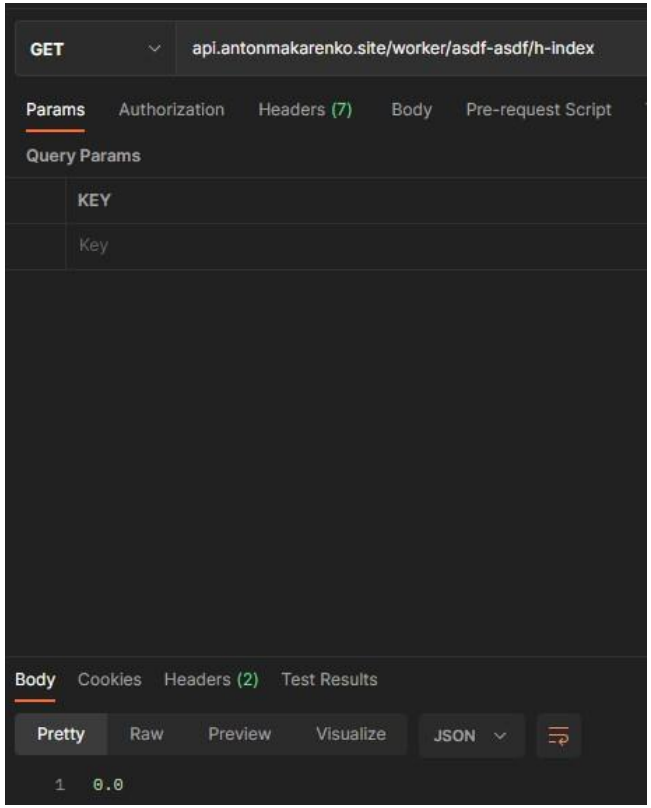
Демонстрація роботи



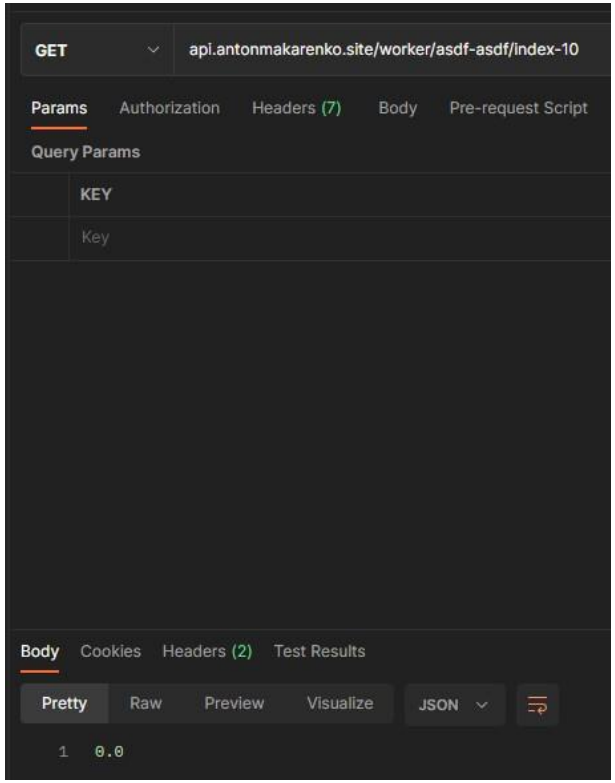
Демонстрація роботи



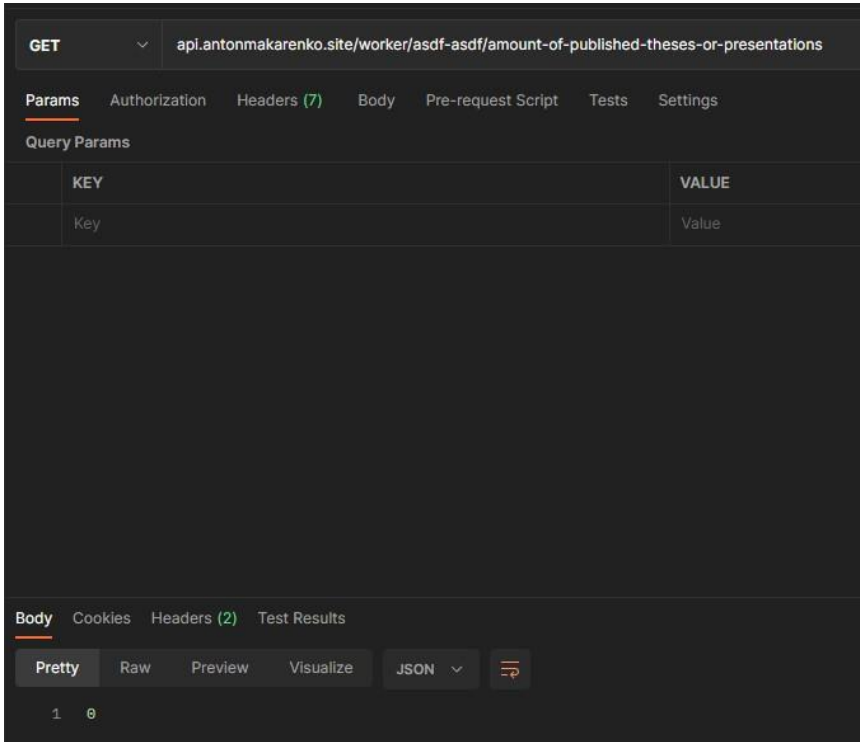
Демонстрація роботи



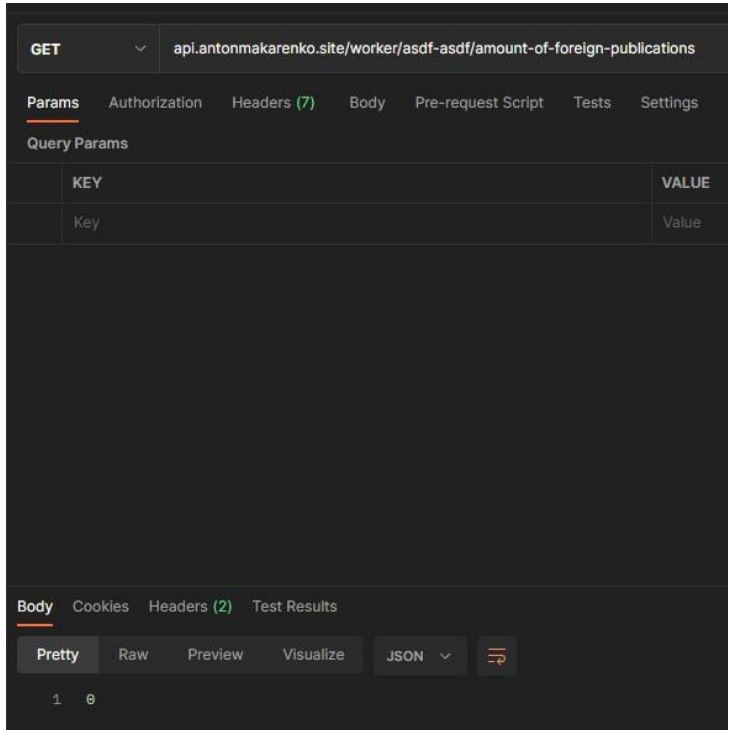
Демонстрація роботи



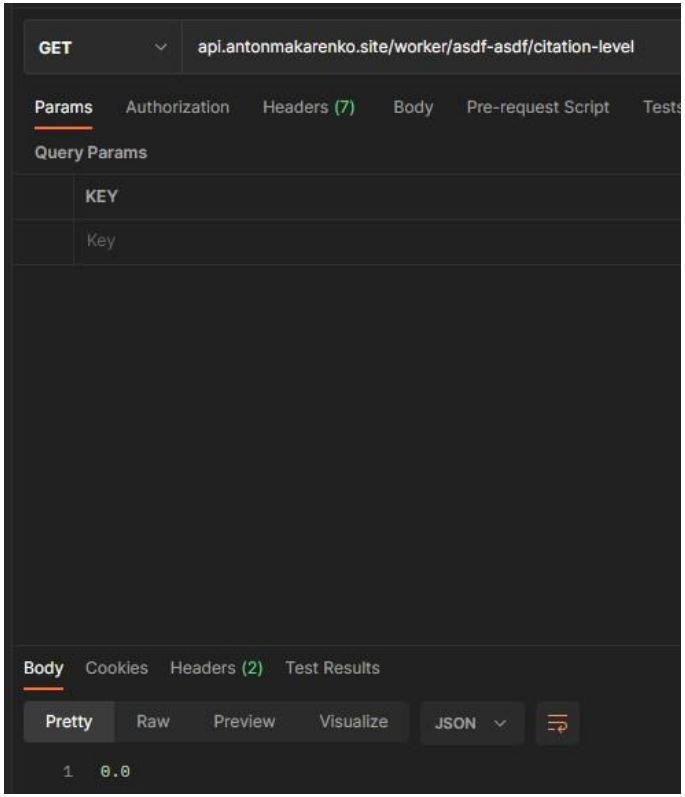
Демонстрація роботи



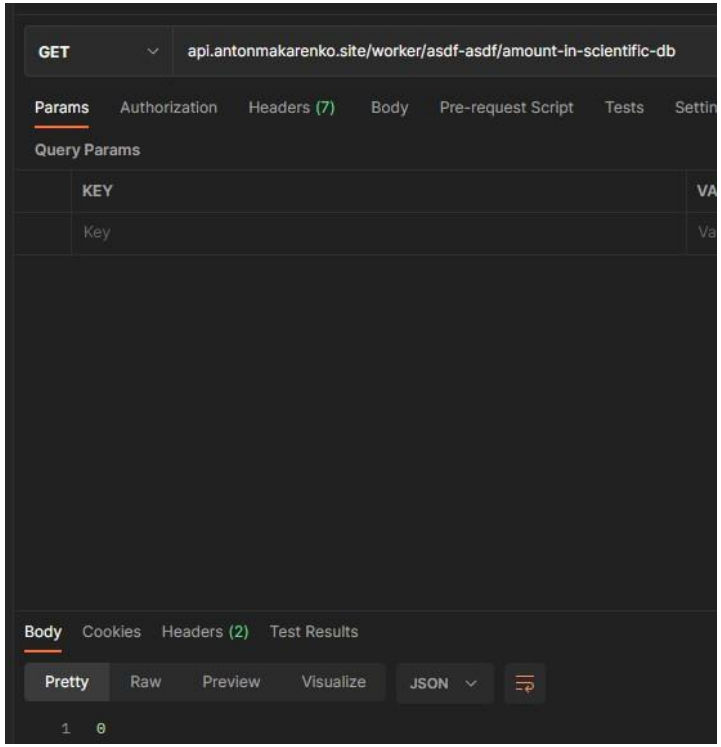
Демонстрація роботи



Демонстрація роботи



Демонстрація роботи



Висновки

- Досліджено та визначено набір показників, що обчислюються.
- Обрано технології для реалізації веб-сервісу.
- Запрограмовано отримання даних із БД та обчислення показників.
- Розгорнуто веб-сервіс за допомогою хмарних ресурсів.

Дякую за увагу!