НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Наталія АУШЕВА
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Використання мікросервісів для збору відкритої інформації**»**

Виконав (-ла):
студент (-ка) IV курсу, групи ТМ-82
Грицюк Артем Олександрович _____

Керівник:
Доцент, к.т.н.
Кузьміних Валерій Олександрович _____

Рецензент:
Директор,НВП «Символ», к.т.н.
Сенченко В'ячеслав Родіонович _____

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

Київ – 2022

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

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

спеціальність 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ
на дипломну роботу студенту
Грицюк Артем Олександрович
(прізвище, ім’я, по батькові)

1. Тема роботи «Використання мікросервісів для збору відкритої інформації»

керівник роботи доцент Кузьміних Валерій Олександрович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2022р. № _____

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи операційна система Windows 10 pro, мова програмування Python, середовище розробки PyCharm, Microsoft Visual Studio,забезпечення бази даних Core, середовище Microsoft Visio

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розробити сервіс для обробки даних з використанням мікросервісної архітектури, проаналізувати діючі сервіси

5.Перелік ілюстративного матеріалу зображення аналогічних сервісів, прицидентна діаграма класів, структурована система схем відношень до бази даних, показ роботи системи

6. Консультанти розділів роботи

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

7. Дата видачі завдання ”15” вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	15.09.21	
2.	Вивчення та аналіз задачі	10.03.22	
3.	Розробка архітектури та загальної структури системи	20.03.22	
4.	Розробка структур окремих підсистем	25.03 - 1.04.22	
5.	Програмна реалізація системи	20.04.22	
6.	Оформлення пояснювальної записки	1.05.22	
7.	Захист програмного продукту	15.05.22	
8.	Передзахист	06.06 – 09.06.22	
9.	Захист	09.06-10.06.22	

Студент _____ Грицюк А.О
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Кузьміних В.О
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Ціль дипломної роботи розробк амікросервіс для збору відкритої інформації з бібліографічних та наукових баз.

Для успішнішого прогресу серед науково-освітнього процесу та збільшення інтересу серед науковців, викладачів, студентів іноземних університетів. Це є великою часткою перед інвестуванням в освіту та науку зі сторони держави, коли існує необхідність вирахувати явну якість робіт, які можуть претендувати до міжнародного представлення.

Застосунок розроблено на мові програмування Python та залучено CSS для розробки програмного інтерфейсу, також базу даних CORE.

Ключові слова: мікросервіс, обробка інформації, Core, запит, мікросервісна архітектура.

ABSTRACT

` The purpose of the thesis is to develop an amicroservice for collecting open information from bibliographic and scientific databases.

For more successful progress in the scientific and educational process and increase interest among scientists, teachers, students of foreign universities. This is a big part of investing in education and science on the part of the state, when there is a need to calculate the explicit quality of work that can claim international representation.

The application is developed in the Python programming language and uses CSS to develop the programming interface, as well as the CORE database.

Keywords: microservice, information processing, Core, query, microservice architecture.

ЗМІСТ

ВСТУП

На сьогодні, накопичується велике різноманіття інформації. Одним із основних пунктів є достовірність інформації, що дуже тяжко визначити через велику кількість результатів, по вашому запиту. Написання курсових, дипломних, докторських робіт та будь яких інших робіть інформаційного або наукового характеру полягають у точності інформації, та знайти серед десятки тисяч варіантів, потрібний, стає справжнім випробуванням, тому на допомогу приходять мікро сервіси для збору відкритої інформації.

Для успішнішого прогресу серед науково-освітнього процесу та збільшення інтересу серед науковців, викладачів, студентів іноземних університетів. Це є великою часткою перед інвестуванням в освіту та науку зі сторони держави, коли існує необхідність вирахувати явну якість робіт, які можуть претендувати до міжнародного представлення.

Під поняттям «міжнародного представлення», йдеться про експертну оцінку, яка залежить від того, яку мету ставлять перед собою наукові інститути, від цього і виникає кількість цих параметрів перевірки та аналізу, які надалі впливають на загальну оцінку показника якості роботи.

Існує безліч таких показників міжнародного рівня, наприклад: наявність та кількість іноземних студентів та інших науковців задіяних у роботі, тощо. Але один з критично важливих є якісь робіт наявних у бібліографічних базах.

Інформація про наукові видання міститься у Всесвітньому списку бібліографічних джерел, який щодня поповнюється новими працями. Інформацію про іншу діяльність агентства отримати важче і не об’єднано в єдиний ресурс. Вирішення проблеми розроблених програмних продуктів – це питання інтеграції показників міжнародної діяльності в одному місці.

Новизна продукту – створення адаптивної програмної системи для збору та обробки вичерпної інформації, яка накопичує дані про міжнародну діяльність окремих осіб та установ, з гнучкістю розширення джерел збору інформації відповідно до потреб користувачів.

Користувачами такої системи можуть бути установи у вигляді інститутів, університетів, тощо, а також окремі посадові особи, державні установи та інші організації, що інвестують у дослідження. Цей аналіз допоможе дослідницьким установам стежити за своєю позицією у світових рейтингах, державним установам – спрямувати увагу громадськості на екстремальні рівні досліджень, інвесторам – створити загальне інвестиційне поняття для окремих установ, окремим дослідникам – переглянути популярність та цитування своїх публікацій. глобальний ресурс.

Основна мета системи – надати користувачам можливість отримувати актуальну аналітичну інформацію.

1 НАДАННЯ МОЖЛИВОСТІ КОРИСТУВАЧАМ ОТРИМУВАТИ ДІЙСНУ ІНФОРМАЦІЮ

Постановка задачі практики – основною метою є надання можливості користувачам отримувати дійсну інформацію яка аналізується на рівні міжнародної діяльності з певних джерел.

Основні задачі, які буде вирішувати система:

- збір даних з різних за структурою та складом джерел, передусім бібліографічних баз,
- агрегація отриманих даних відповідно до запиту користувача,
- вилучення помилкових даних та дублів вже отриманих.

Оскільки система орієнтована на змінну кількість джерел, які будуть відрізнятись один від одного як структурою даних, так і інтерфейсами взаємодії, головною вимогою є створення гнучкої архітектури з можливістю постійного модифікації з мінімальними затратами на включеннях нових джерел до системи, виключення старих і вже непотрібних та модифікацію ще актуальних існуючих.

2 МІКРОСЕРВІСНА АРХІТЕКТУРА ДЛЯ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Мікросервісна архітектура для побудови інформаційної системи являє собою набір незалежно розгорнутих сервісів. Системи, побудовані таким чином, мають здатність частково розвиватися, головним чином тому, що кожен мікросервіс є значною мірою автономним. Крім того, архітектура дозволяє гнучкою розширювати компоненти інформаційної системи, забезпечуючи таким чином оптимальне використання існуючого серверного обладнання. Масштабування кожного мікросервісу виконується незалежно від інших мікросервісів. Стає можливим розподіл обчислювальних ресурсів з урахуванням фактичної діяльності користувача та вимог до конкретних функцій системи.

Мікросервісна архітектура для побудови інформаційних систем являє собою набір незалежно розгорнутих служб. Системи, побудовані таким чином, мають здатність частково розвиватися, головним чином тому, що кожен мікросервіс є значною мірою автономним. Крім того, архітектура дозволяє гнучко розширювати компоненти вашої інформаційної системи, забезпечуючи таким чином оптимальне використання наявного серверного обладнання. Масштабування кожного мікросервісу виконується незалежно від інших мікросервісів. З'являється можливість розподілити обчислювальні ресурси з урахуванням фактичної діяльності користувачів і вимог конкретних функцій системи.

Оскільки складові послуги невеликі, вони можуть бути створені однією або декількома невеликими командами із самого початку, розділеними межами сервісів, що полегшує масштабування зусиль з розробки у разі потреби.

Після розробки ці послуги можуть бути розгорнуті незалежно один від одного, що дозволяє легко виявляти "гарячі" послуги і масштабувати їх незалежно від всього додатка. Мікросервіси також забезпечують покращену ізоляцію від збоїв, тому у разі помилки в одному сервісі вся програма не обов'язково перестає працювати. Коли

помилка виправлена, її можна розгорнути лише для відповідного сервісу замість розгортати всю програму.

Ще одна перевага, яку дає архітектура мікросервісів, - це можливість вибору технологічного стеку (мови програмування, бази даних і т.д.), який найкраще підходить для необхідної функціональності (сервісу), замість використання більш стандартного, універсального підходу.

Ключовою особливістю цієї системи є налаштування параметрів системи відповідно до вимог користувача. Тобто відповідно до запиту, відповідно до конкретної ситуації запиту та можливостей системи обробити запит формується можливий спосіб вилучення та обробки даних. Це досягається за допомогою загальної гнучкої моделі даних, архітектури мікросервісів, орієнтованої на події, та оркестровки програмного забезпечення.

2.2 Мікросервісна архітектура для розподілення сервісів.

Основою для реалізації проекту є мікросервісна архітектура, яка розподіляє сервіси за функціональним призначенням. Це пов’язано з необхідністю розширити функціональність системи, не порушуючи існуючі компоненти. Крім того, під час масштабування системи для збільшення пропускної здатності потрібна більша гнучкість.

Є організації, які досягли успіху з мікросервісами і дотримувались моделі, коли команди, які створюють сервіси, піклуються про все, що стосується цієї служби. Саме вони розробляють, розгортають, підтримують і підтримують його. Немає окремих команд підтримки чи обслуговування.

Інший спосіб досягти того ж – мати внутрішню модель з відкритим кодом. Використовуючи цей підхід, розробник, якому потрібні зміни в службі, може перевірити код, попрацювати над функцією та подати PR самостійно, замість того, щоб чекати, поки власник служби забере і попрацює над необхідними змінами.

Щоб ця модель працювала належним чином, необхідна відповідна технічна документація, а також інструкції з налаштування та вказівки для кожної служби, щоб будь-кому було легко отримати послугу та працювати з нею.

Ще одна прихована перевага цього підходу полягає в тому, що він зосереджує розробників на написанні високоякісного коду, оскільки вони знають, що інші будуть дивитися на нього.

Є також деякі архітектурні шаблони, які можуть допомогти в децентралізації речей. Наприклад, у вас може бути архітектура, де набір служб обмінюється інформацією через центральну шину повідомлень.

Цей підхід надає можливості для реалізації важливих переваг мікросервісної архітектури:

- надійність роботи під час відмов: при відключенні одного з серверів, всі інші залишаються активними;
- реалізація системи може відбуватись на різних мовах програмування;
- хороша безпека даних;

Взаємодія програмних середовищ в межах системи та створення інформаційних потоків, розташовується на архітектурі. Це вимушено тим, що необхідно мати велику пропускну здатність системи обробки великих запитів до безпосередньо системи.

Цей підхід до розробки архітектури дозволяє отримувати можливість, домовленість у підсумках висновків. Цими діями, не залежно від об’єму та навантажень на систему, у результаті користувач отримує конкретизовані дані, по яким був сам запит.

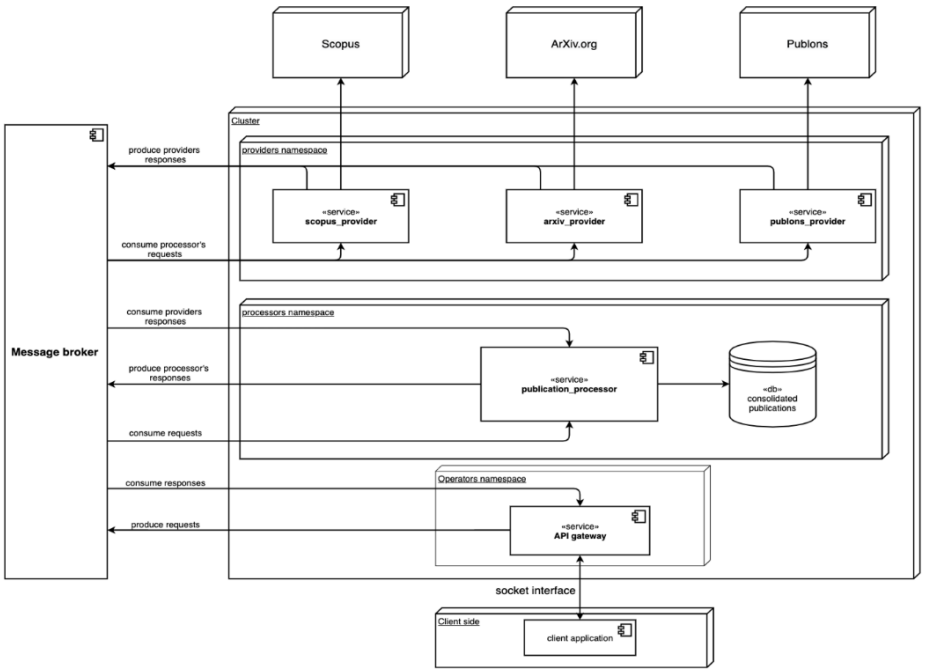


Рисунок 2.1 – Архітектура системи

Діаграма послідовностей, яка показує процес реалізації запиту користувача для отримання результату аналізу по публікаціям, зображується на рисуноку 2.2.

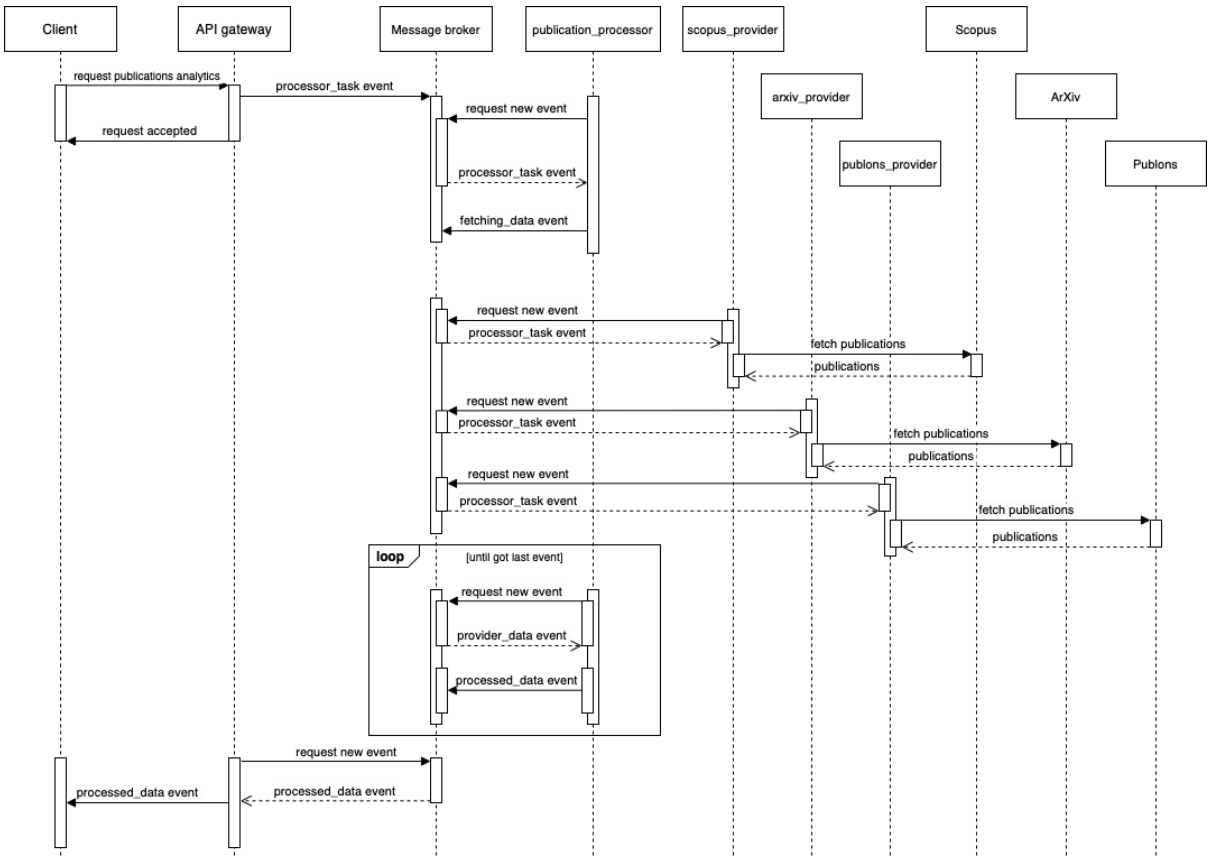


Рисунок 2.2 – Діаграма послідовностей

Користувач відправляє запит для отримання даних, наприклад по назві твору, на програмний інтерфейс системи по каналам сокетів. Приймаючий слюз, передає на тіло запиту. Повідомлення формується в подію, яку кладе у чергу в брокер повідомлень, переходячи до клієнтського запиту.

Якщо дані доступні, він негайно генерує та надсилає подію для повернення результату запиту. Якщо даних немає або відповідь недостатньо повна, формується подія, яка надсилається в систему, про те, що певні дані необхідно зібрати за певними критеріями (наприклад, публікація автора).

Служба постачальника (послуга, що закінчується на _provider на малюнку 1) підписується на подію збору даних і може обробляти її. Їх завдання — зібрати запитувані дані з конкретних зовнішніх джерел. Кожен окремий провайдер може використовувати власне джерело даних - віддалений ресурс, котрий являє собою специфічну наукометричну основу. Для кожного окремого ресурсу та інтерфейсу, який він надає (або ні), дані, зібрані з такого ресурсу, є окремими. Це може бути збір

даних через відкриті інтерфейси програмування або через платні інтерфейси програмування. Без програмного інтерфейсу такі веб-ресурси можна проаналізувати, реалізувавши службу синтаксичного аналізу, наприклад рівень між постачальником і джерелом даних, що цікавить.

Результати збору даних також надсилаються в систему як події (або послідовності подій). Служби процесора, які підписують ці події (сервіси, що закінчуються на _processor на малюнку 1), обробляють результати відповідно до своєї бізнес-логіки. Це може бути дедуплікація, агрегація, додатковий пошук за ключовими словами тощо.

Результати цієї обробки надсилаються назад до шлюзу, де вони потрапляють на клієнта у вигляді подій сокета.

Таким чином, система має повністю «зрештою узгоджену» модель. Клієнт отримує дані в міру їх отримання, повторна відправка того ж запиту не вимагає повторної обробки і повертає результат негайно. Такий підхід забезпечує незалежність кожного окремого компонента системи та дає можливість гнучко змінювати склад сервісу, масштабувати та примусово залишати бездіяльність без присутності всієї системи. Ключовою особливістю тут є можливість додавати нові джерела даних і процесори, не порушуючи інших компонентів системи. Крім того, кожен сервіс може по-різному обробляти запити залежно від стадії розробки або повноти доступних джерел даних, що дозволяє системі адаптуватися до кожного окремого запиту користувача.

Дані, отримані з різних зовнішніх джерел, об’єднуються в єдину загальну модель в системі. Кожне окреме зовнішнє джерело даних надає інформацію у своєму власному форматі. Проте існує чіткий набір понять і сутностей, заснованих на предметній області, як-от: видання, видавництва, науково-дослідні установи, відділи наукових установ тощо.

Завдяки цілісності даних у кожній події, дані передаються через систему в денормалізованому вигляді, щоб зменшити кількість подій, що обробляються. Всі дані передаються в систему в чітко визначеному форматі для злагодженої роботи всіх компонентів системи.

Нормована модель даних встановлених зв'язків зберігається в базі даних. Тому в якості бази даних використовується реляційна база даних. Це необхідно для зменшення обсягу енергонезалежної пам'яті, необхідної для зберігання великих обсягів оброблених даних.

Варто відзначити структуру запиту користувача. Найзручнішим форматом, який забезпечує гнучкість запитів, є мова запитів, яка добре визначена для системи.

Варто звернути увагу на структуру запиту користувача. Найзручнішим форматом, який забезпечує гнучкість запитів, є мова запитів, яка добре визначена для системи. Структура цих запитів представлена у вигляді пар ключ-значення у форматі JSON, де ключі можна використовувати для відображення результатів, укрупнення, фільтрації за полями або зведених результатів, сортування та обмеження кількості результатів.

Впроваджуйте інфраструктуру системи, моніторинг та управління ресурсами за допомогою програмного забезпечення для контейнеризації додатків і оркестраторів контейнерних додатків3.

Оскільки між набором програмних служб необхідно встановити зв’язок, ці інструменти використовуються для організації взаємодії програм, забезпечення незалежності системи від того, як реалізовано додаток, створення безпечної комунікаційної мережі в системі та можливості моніторингу життєвий цикл програмних послуг.

Дійсно вартує уваги, те, що мікросервіси у системі мусять мати вожливості масштабування у багато реплік при цьому не допускаючи можливості порушення узгодженості даних. Це є необхідністю для того, щоб зберігати можливості збільшувати пропусну здатність мікросервісної системи.

Варто зауважити певні переваги використання контейнерів та засобів контейрезованих додатків:

- портативність додатків;
- масштабованість системи;
- швидше розгортання програм;
- висока продуктивність та гнучкість розробки додатків;

– підвищена безпека комунікації додатків у рамках системи;

– безперервність роботи системи;

Нижче наведено узагальнену схему роботи програмних мікросервісів у контейнеризації системи на рисунку 3.

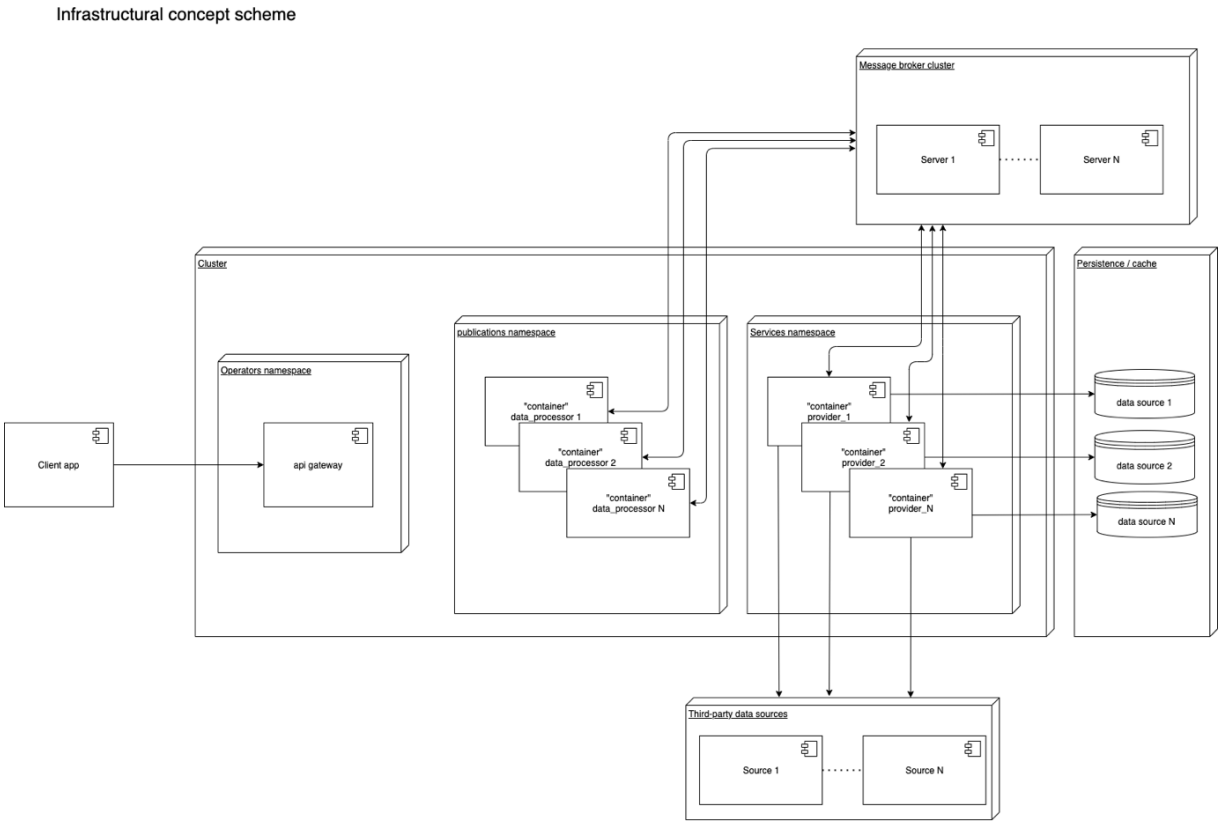


Рисунок 2.2.2 – Інфраструктура системи

3. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Сьогодні аналогів із залученням мікросервісної архітектури для використання у потрібних функціях не було знайдено, тоді будемо аналізувати схожі певним фунціоналом або образом розробки.

3.1 Контейнеризація та мікросервіси

` Мікросервісна архітектура, вона ж мікросервіси, - це особливий метод проектування програмних систем, що дозволяє структурувати одну програму як набір слабко пов'язаних сервісів. Програми зазвичай починаються як монолітна архітектура (докладніше про це нижче), а згодом перетворюються на набір взаємопов'язаних мікросервісів.

Основна ідея мікросервісів полягає в тому, що деякі типи програм легше створювати та підтримувати, коли вони розбиті на безліч невеликих частин, що працюють разом. Хоча складність архітектури зросла, мікросервіси, як і раніше, пропонують безліч переваг у порівнянні з монолітною структурою.

Концепція мікро випливає з існуючої монолітної інфраструктури, яку використовувало більшість компаній, особливо якщо компанія існує вже десять років чи довше. Замість монолітної архітектури кожен компонент мікросервісної архітектури має:

- Власний процесор
- Власне середовище реалізаціїї
- Частіше всього цим займається команда людей, забезпечуючи один одного відмінностями своїх сервісів

Така архітектура означає, що:

- Виконувати свій власний унікальний процес
- Автономно працювати без необхідної взаємодії з другими мікросервісами або з додатком в цілому.

Такі можливості до розділуабо об’єднанню, захищають всю програму від багів та глюків, що робить її(програму) використання ще більш приємне, особливо для тих хто до цих пір використовую монолітну інфраструктуру.

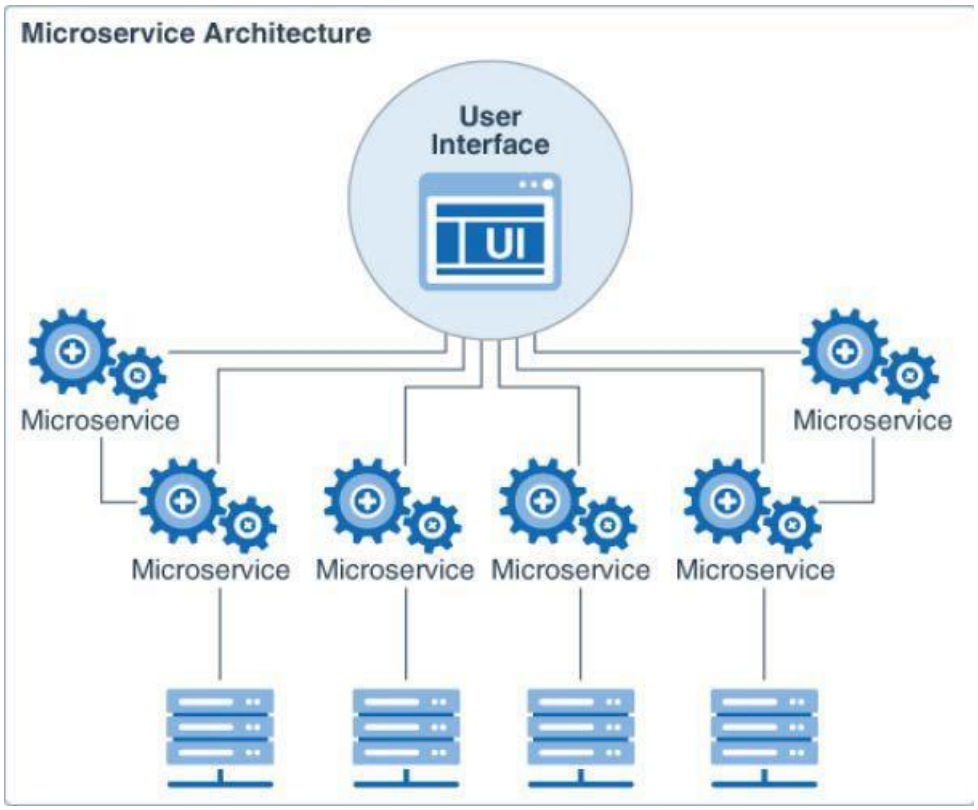


Рисунок 3.1 — Мікросервісна архітектура

3.2 Структура мікросервісної архітектури

Мікросервісна архітектура застосовує бібліотеки, але головна ціль розбиття програми – способом розділення на частини мікросервісів. Визначаються бібліотеки, як контейнери, що піключаються до додатку і виконуються програмою в тому самому процесі, паралельно цьому, мікросервіси здійснюють комінікацію серед своїх через запити.

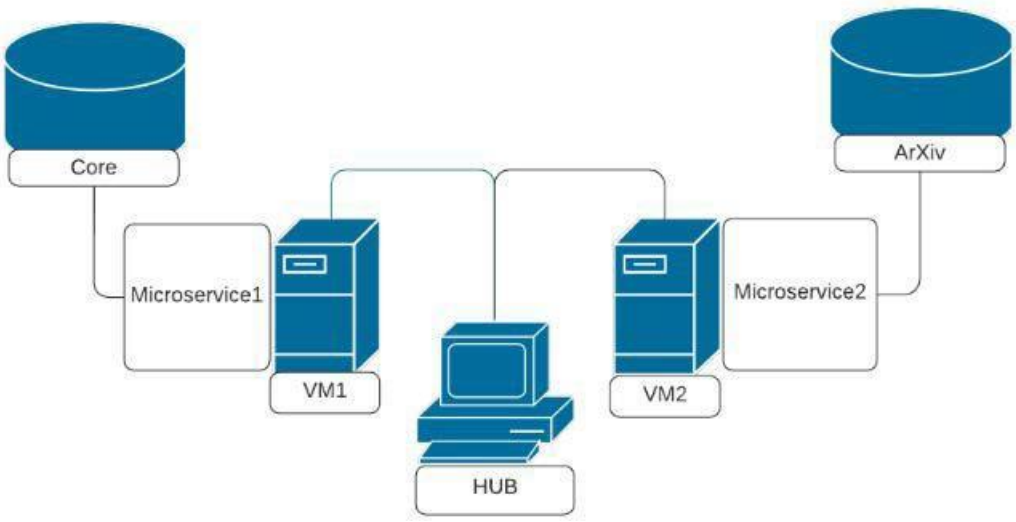


Рисунок 3.2 — Мікросервісна архітектура

Контейнери це основне та незамінні сервіси, після їх застосунку, не слід змінювати. Якщо появляється свіжа версія програмного кода появляється в наявності, старий контейнер знищується, на його місці створюється новий із новим кодом. Запуск контейнерів здійснюється за лічені секунди або навіть мілісекунди.

Додактки розроблені як набір незалежних модульних компонентів, які краще так легше тестувати, підтримувати. Основні переваги:

- Підвищення гнучкість
- Покращує робочі процеси
- Зберегти час, необхідне для покращення продуктивності

Але кожний незалежний компонент збільшує складність, тоді для боротьби з цією проблемою, можуть буду створенні додаткові компоненти моніторингу.

Ось це і є найпопулярніші плюси мікросервісів, та чому вони користуються популярністю серед розробників.

4 СПОСОБИ РОЗРОБКИ

У результаті незалежності мікросервісів та підвищеної вразливості, створюється велика перевага у вигляді покращення засобів програмування, професійних середовищ програмування та розробки.

Далі буде відображено засоби та мови програмування та способи створення чинного додатку.

4.1 Docker Compose

Docker Compose — це інструмент, який входить до складу Docker. Він призначений для вирішення проблем, пов'язаних з реалізацією проектів.

Вивчаючи основи Docker, ви, можливо, стикалися зі створенням простих програм, які працюють автономно, незалежно від, наприклад, зовнішніх джерел даних або деяких служб. На практиці такі додатки зустрічаються рідко. Реальні проекти зазвичай охоплюють весь набір додатків для спільної роботи.

Як дізнатися, чи потрібно використовувати Docker Compose під час розгортання проекту? Насправді це дуже просто. Якщо ви використовуєте кілька служб для виконання цього проекту, Docker Compose може стати в нагоді. Наприклад, коли ви створюєте веб-сайт, якому потрібно підключитися до бази даних для автентифікації користувача. Такий проект може складатися з двох служб - одного, що забезпечує роботу веб-сайту, а іншого відповідає за ведення бази даних.

Технологія Docker Compose, якщо описати її спрощено, дозволяє запускати кілька служб за допомогою однієї команди.

4.2 Architecture Docker

У цьому розділі ми розглянемо архітектуру Docker та пов’язані з нею компоненти. Ми також побачимо, як кожен компонент працює разом, щоб Docker працював.

Архітектура Docker змінювалася кілька разів з моменту її створення.

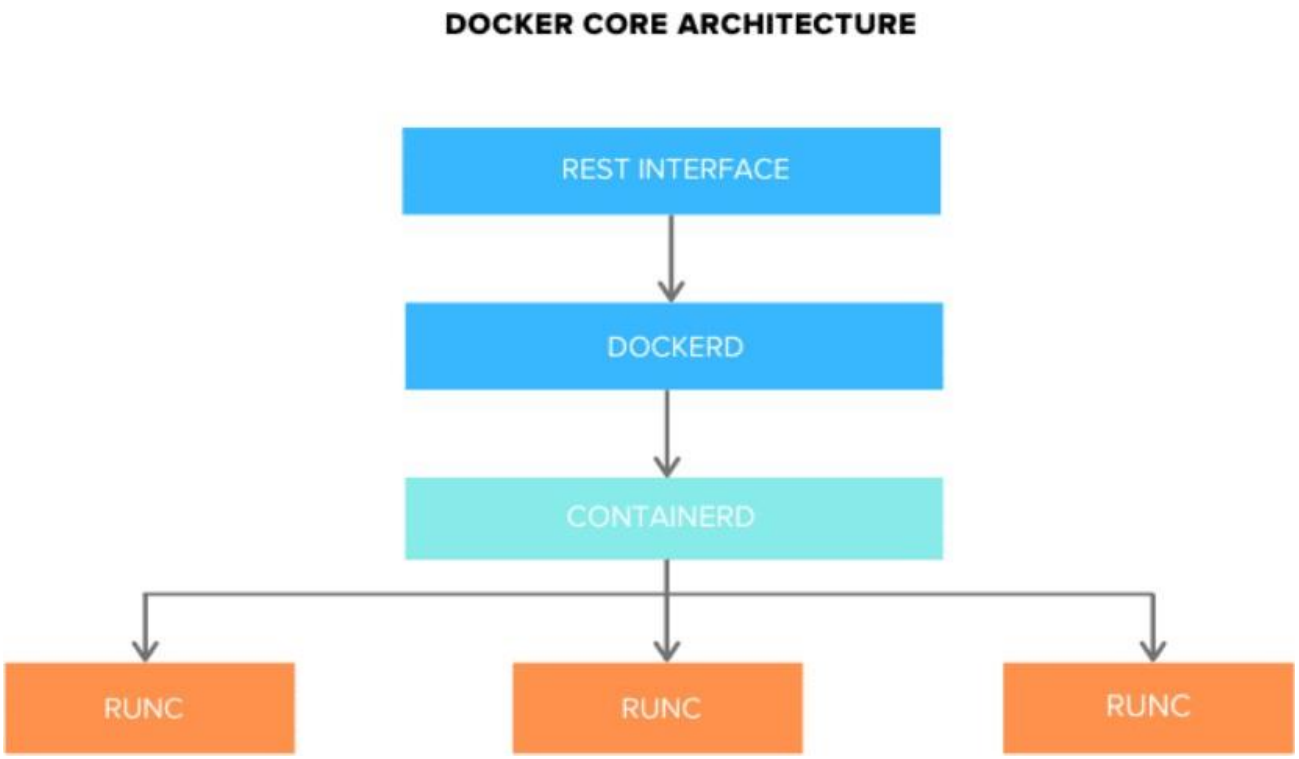


Рисунок 4.1 — Architecture Docker

Контейнер – це поточне зображення, яке відображається. Ви можете створювати, запускати, зупиняти, переміщувати або видаляти контейнери за допомогою Docker API або CLI. Ви можете підключити контейнер до однієї або кількох мереж, приєднати до нього репозиторії і навіть створити нові образи на основі його поточного стану.

За замовчуванням контейнери відносно добре ізольовані від інших контейнерів та їх хостів. Ви можете контролювати, наскільки контейнерна мережа, сховище чи інші важливі підсистеми ізольовані від інших контейнерів або хостів.

Контейнер визначається його зображенням і параметрами конфігурації, які ви надаєте під час його створення або запуску. Будь-які зміни стану, які не зберігаються в постійному сховищі, зникають після видалення контейнера.

4.4 Програмне середовище розробки PyCharm

Як програміст, ви повинні бути зосереджені на бізнес-логіці та створення корисних додатків для ваших користувачів. При цьому PyCharm від JetBrains заощадить вам багато часу, взявши на себе всю рутину та полегшивши ряд інших завдань, таких як налагодження та візуалізація.

PyCharm - це інтегроване середовище розробки (IDE), яке використовується в комп'ютерному програмуванні, створене для мови програмування Python. При використанні PyCharm на Databricks за замовчуванням PyCharm створює віртуальне середовище Python, але ви можете настроїти створення середовища Conda або використовувати існуюче середовище.

4.5 Core

CORE збирає дослідницькі роботи з таких джерел, як інституційні та предметні репозиторії, журнали відкритого доступу та гібридні журнали.

Принципи відкритої наукової інфраструктури (POSI) пропонують набір керівних принципів, відповідно до яких організації та ініціативи відкритої наукової інфраструктури, що підтримують дослідницьку спільноту, можуть працювати та підтримуватися. Принципи розділені на три основні категорії: управління, стійкість та страхування.

4.6 Хостинг Google

Екземпляри Compute Engine можуть запускати загальнодоступні образи, надані Google для Linux і Windows Server, а також приватні користувацькі зображення, які можна створювати або імпортувати з існуючих систем. Ви також можете розгорнути контейнери Docker, які автоматично запускаються на екземплярах із загальнодоступними оптимізованими для контейнерів образами ОС.

Ви можете вибрати властивості машини екземпляра, такі як кількість віртуальних процесорів і обсяг пам’яті, використовуючи попередньо визначений набір типів машин або створюючи власні. Копії та проекти

Кожен екземпляр належить до проекту Google Cloud Console, і цей проект може мати один або кілька екземплярів. Коли ви створюєте екземпляр у проекті, ви вказуєте регіон екземпляра, операційну систему та тип машини. Коли ви видаляєте екземпляр, він видаляється з проекту.

За замовчуванням кожен екземпляр Compute Engine має невеликий постійний диск, який містить операційну систему. Ви можете додати додаткові параметри зберігання до свого екземпляра, коли програми, що працюють на вашому екземплярі, потребують більше місця для зберігання.

Кожен мережевий інтерфейс екземпляра Compute Engine підключений до підмережі унікальної мережі VPC. Щоб отримати докладнішу інформацію про VPC, див. Огляд мереж VPC та квот VPC.

Екземпляри Compute Engine підтримують декларативний підхід до використання контейнерів. Створюючи шаблон віртуальної машини або екземпляра, ви можете вказати ім’я образу Docker та конфігурацію запуску. Compute Engine подбає про все інше, включаючи обслуговування сучасних оптимізованих для контейнерів зображень із встановленим Docker та запуск контейнерів під час запуску віртуальних машин.

До деяких ресурсів може отримати доступ будь-який інший ресурс в іншому регіоні та регіоні. Ці глобальні ресурси включають попередньо налаштовані образи дисків, диски та знімки мережі. До деяких ресурсів можуть отримати доступ лише ресурси в одному регіоні. Ці регіональні ресурси включають статичні зовнішні IP-адреси. Тільки ресурси в одному регіоні можуть отримати доступ до інших ресурсів. Ці зональні ресурси включають екземпляри ВМ, їх типи та диски.

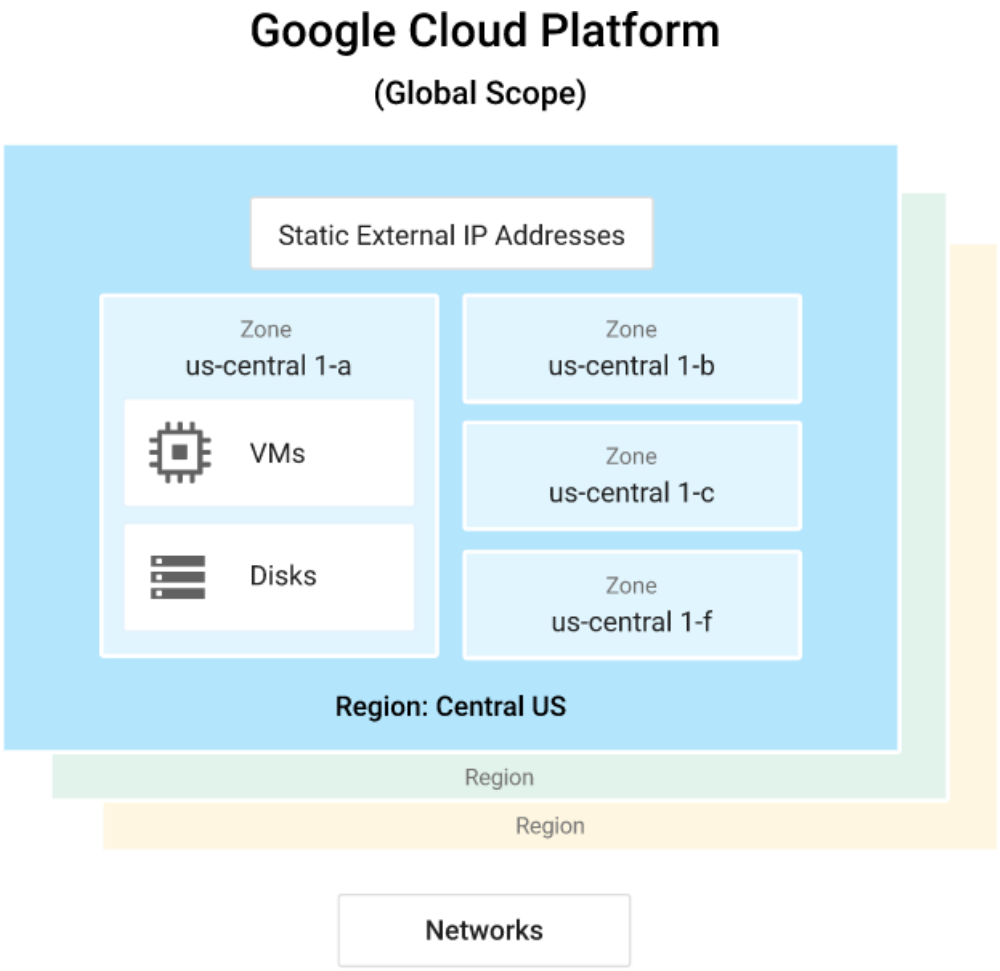


Рисунок 4.2 – Елементи хостингу

На наступній діаграмі показано взаємозв’язок між глобальним масштабом, регіонами та регіонами та деякими їх ресурсами:

Усі ресурси GCP, які ви поширюєте та використовуєте, мають належати цьому проекту. Ви можете розглядати проекти як організатори того, що ви будуєте. Цей елемент складається з налаштувань, дозволів та інших метаданих, які описують вашу програму. Ресурси в рамках проекту можуть легко працювати разом, наприклад, спілкуватися через внутрішню мережу відповідно до правил регіону та регіону

5 СТРУКТУРА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ЗБОРУ ДАНИХ ДЛЯ АНАЛІЗУ МІЖНАРОДНОЇ ДІЯЛЬНОСТІ

Для розробки рішення для такого складного питання, як створення та розробка мікросервісної системи для збору даних для аналізу міжнародної діяльності, задіяна архітектура, яка базується на принципах сценаріїв із залученням підсистем архітектури з частинним залученням машинного навчання. Цей мікросервіс призначається для одночасного збору даних з різними аспектами міжнародної діяльності з різноманітних за будовою і формулою збереження та розміщення інформаційних баз у реальному часі. Мікросервіс надає можливості збирати різну інформацію, з окремих науковців, так і з груп. Схема реалізації буде представлена на рисунку 4.

Узагальнений принцип системи для інформатично-аналітичного рівня міжнародної діяльності можна розділити на основні кроки:

1. Оброблення запитів та форматування сценарію;
2. Форматування на базі основного плану структури міросервісної архітектури за для виконання задач по запиту;
3. Обробка запиту та оброблення інформаціїї за заданим запитом;
4. Збереження там консолідація розультатів для дальшого аналізу
5. Обробка аналізу роботи мікросервісів та адаптування їх параметрів;
6. Оцінювання роботи сценарію та адаптування під конкретні задачі;
7. Аналізування отриманої інформації та отримання результатів запиту.

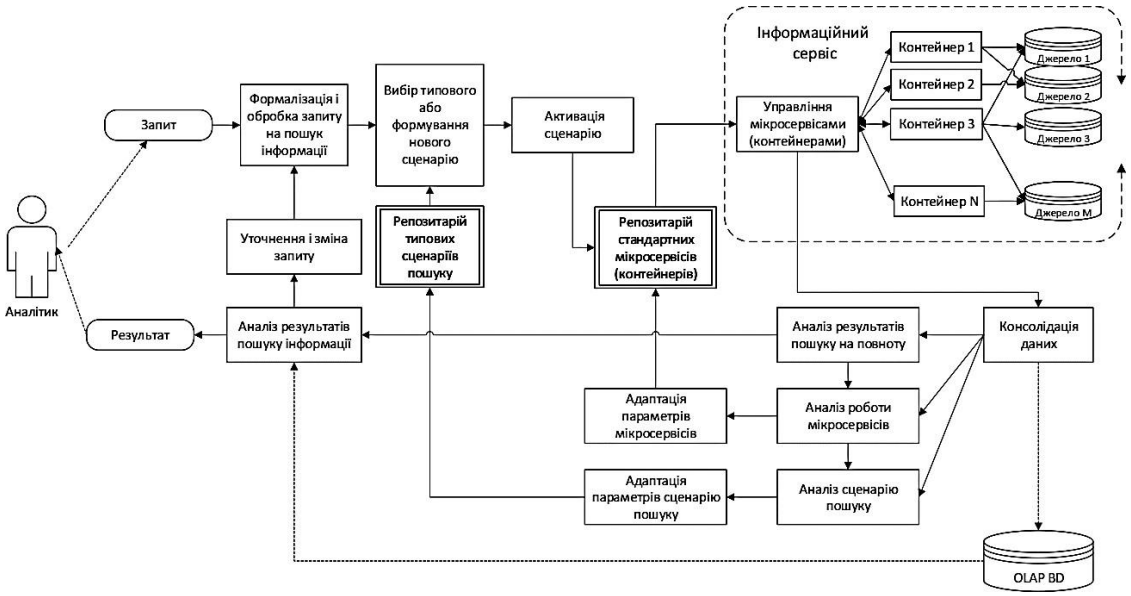


Рисунок 5.1 – Зображення мікросервісної підсистеми для збору відкритої інформації

Перший пункт проводиться аналізуючи запит, якщо необхідне уточнення для формування запиту з сторони можливості його виконання в межах інформаційно-аналітичних можливостей функціональності системи. Для точного та форматного запиту залучається стандартний сценарій, через необхідність формуються нові сценарії, стандартно, на основі типового плану.

Другий пункт проводиться підключення відповідно до сценарію необхідних мікросервісів, як елементів виконання функцій збору та обробки інформації за типовим сценарієм чи його версією. Ця процедура активації сценарію виконується з використанням репозитарію стандартних мікросервісів. При цьому мікросервіси реалізуються як окремі контейнери. Для цього використовують засоби програмної платформи Core для швидкої розробки, тестування та розгортання мікросервісів. Core упаковує програмні засоби в стандартизовані блоки, які називаються контейнерами. Кожен контейнер включає все необхідне для роботи програми: бібліотеки, системні інструменти, код і середу виконання. Завдяки Core можна швидко розгортати та масштабувати мікросервіси в будь-якому середовищі.

Третій пункт проводиться виконання збору та обробки інформації за запитом. Для цього використовується платформа CORE, яка дозволяє легко впроваджувати і

використовувати додатки в мікросервісної архітектурі, створюючи рівень абстракції поверх групи хостів. При цьому платформа CORE дає можливість:

1) розгортання контейнерів і всі операції для запуску необхідної конфігурації. До їхнього числа входять перезапуск зупинених контейнерів, їх переміщення для виділення ресурсів на нові контейнери та інші операції;

2) масштабування і запуск декількох контейнерів одночасно на великій кількості хостів;

3) балансування безлічі контейнерів в процесі запуску. Для цього CORE використовує API, завдання якого полягає в логічному групуванні контейнерів. Це дає можливість визначити їх пули, задати їм розміщення, а також рівномірно розподілити навантаження.

Четвертий пункт передбачається консолідація отриманої з різноманітних джерел інформації. Консолідація розглядається як комплекс методів і процедур, спрямованих на об’єднання даних з різних джерел, забезпечення необхідного рівня їх релевантності, актуальності та повноти, перетворення в єдиний формат, в якому вони можуть бути завантажені до бази даних.

Для збереження отриманої інформації після її консолідації передбачено використання OLAP бази даних, що дає ряд переваг для подальшої обробки в залежності від необхідності аналітика, що надав запит .

П’ятий пункт визначає процедури аналізу роботи мікросервісів та адаптація їх параметрів. Адаптація параметрів мікросервісів розглядається як корекція значень параметрів стандартних мікросервісів з репозитарію мікросервісів, що приводить до появи нових версій цих мікросервісів, які будуть закріплені за типовими сценаріями, в результаті виконання яких і з’явилися ті версії.

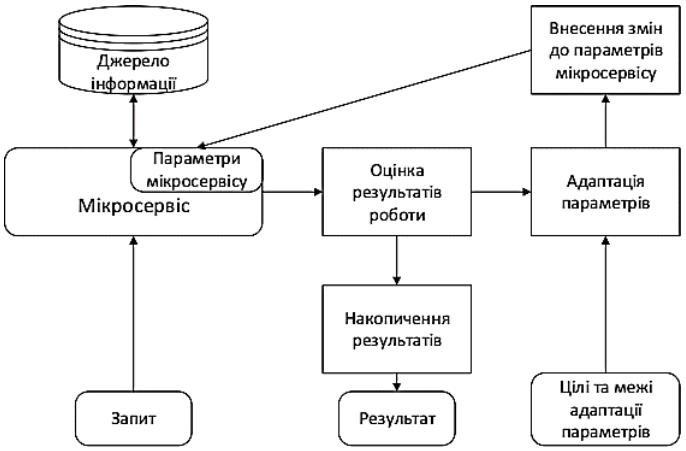


Рисунок 5.2 – Адаптація параметрів

Експериментальні дані показали, що досить ефективним є зменшення чи збільшення значень k_{zj} залежно від значень $Q_j(A_j)$ та $Q^*_j(A_j)$ на 15-20% з подальшим нормуванням до сумарного значення M.

Шостим пунктуом є оцінка роботи сценарію та його адаптація для конкретних задач аналізу рівня міжнародної діяльності. Для побудови нових версій і примірників типових сценаріїв пропонується використання алгоритму, що побудований на основі лінійного стохастичного автомату [27]. Структурна адаптація на основі використання лінійних стохастичних автоматів базується на оцінці найбільш ефективних джерел інформації для конкретного запиту та методів обробки інформації можливих для цих джерел [28]. Це дозволить правильно, по відношенню до найбільшої відповідності, вибрати джерела інформації відповідно до запиту та методи обробки цієї інформації.

6 СИСТЕМА ДЛЯ ОЦІНКИ РІВНЯ МІЖНАРОДНОЇ ДІЯЛЬНОСТІ

Для системи оцінки рівня міжнародної діяльності у структурі сценаріїв найбільш критичним є альтернативні методи вилучення інформації, що пов'язані напряму відповідними джерелами інформації.

Для першої групи важливим параметром якості роботи, що може впливати на адаптацію сценарію, є кількість отриманих релевантних результатів за певний час.

С точки зору пошуку інформації можна визначити:

$u(t)$ — стан системи на поточний момент часу t, що визначає ймовірності вибору альтернативних мікросервісів вилучення інформації чи альтернативних мікросервісів обробки інформації (M_1, …, M_n) на поточній ітерації;

$u(t+1)$ – стан системи на наступний момент часу (на наступній ітерації), що визначає ймовірності вибору альтернативних мікросервісів (M_1, …, M_n) на наступній ітерації;

$x(t)$ – вхідні дані системи на поточній ітерації, що визначають результати оцінки вибірки розміром V з обраного на поточній ітерації альтернативного мікросервісу;

$y(t)$ – вихідні дані системи на поточній ітерації, що визначають обраний альтернативний мікросервіс (M_1, …, M_n).

Реалізація використання такого автомату для вибору альтернативного мікросервісу може бути побудовано таким чином, що зміна стану автомату $u(t+1)$ визначається у вигляді регулярної залежності, а його вихід $y(t)$ визначається у вигляді стохастичного процесу.

Алгоритм, що побудовано на основі використання стохастичного автомату складається з наступних кроків.

Наприклад, коефіцієнт перерахунку ймовірностей стохастичного автомату можна зробити, як показано у таблиці. Значення R_i та k_R, що використані у таблиці, отримані і добре себе показали при тестуванні алгоритму, що зображено у таблиці 1.

Таблиця 1

R_i	0-0.2<	0.2-0.4<	0.4-0.6<	0.6-0.8<	0.8-1.0
k_R	0.5	1.0	1.5	1.7	2

1. Переходимо до кроку 2 для подальшої роботи сценарію з обраним альтернативним мікросервісом, який пов'язаний з певним джерелом інформації чи методом обробки вилученої інформації.

Сьомий останній пункт відноситься до аналіз отриманої інформації відповідно до запиту. Залежно від оцінки якості отриманої інформації та її відповідності запиту аналітиком, у випадку його невдоволення, проводиться уточнення чи зміна запиту, а в протилежному випадку процес обробки запиту вважається закінченим.

7 ВАРІАНТИ ВИКОНАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Опираючись на постановку задачі, важливими досягненнями при розробці програми можна вилучити та зберегти інформацію, також в подальшому впорядкувати у певну послідовність для аналітики. Узагальнена думка на рахунок раціонального використання ресурсів при осягнені цих цілей, є вимушеним скласти певні під задачі. Вони дають можливості зробити краще у ході розробки програмний продукт та цим самим оптимізувати роботу при цьому. Для формування під задач потрібно мати на увазі власні ресурси для цієї розробки та провести аналіз для викриті слабких місць.

З огляду на основне завдання, найбільший технологічний потенціал вимагає створення рішень для пошуку, пошуку та зберігання інформації. Необхідна інформація може бути збережена на носії, що використовується системою. Для цього часто потрібні терабайти місця, оскільки постійно оновлювані бази даних, в яких зберігаються науково-технічні публікації, можуть містити сотні тисяч публікацій. У рамках розробки програмного продукту за темою дисертації було вирішено використовувати зовнішній ресурс, який має власну базу даних та дозволяє використовувати її віддалено. І, що важливо, він безкоштовний у використанні. Безкоштовна база даних пропонує всім учасникам можливість будь-якої взаємодії, незалежно від виду діяльності, цілей, залежностей чи статків. Таким чином, підзавдання полягає в тому, щоб створити схему доступу до таких (віддалених і безкоштовних) ресурсів для збереження та раціоналізації пам’яті, що лежить в основі методів розробки програмного забезпечення.

Враховуючи, що ці два завдання є різними за своєю природою, їх виконання може виконуватися таким чином, щоб воно могло бути незалежним від виконання іншого завдання. Об’єднавши наведені вище завдання та припущення, ми робимо висновок, що платформи пошуку, пошуку та зберігання інформації можуть бути сконструйовані так, що при зміні мети використання інформації переконфігурація не потрібна. Тому реалізація інформаційного потоку, Будуть створені кореляції та

презентації, щоб конкретно взаємодіяти з платформою для пошуку, отримання та зберігання інформації. Хоча платформа для пошуку, отримання та зберігання інформації може бути використана пізніше в інших областях розробки для взаємодії з потрібною інформацією. Платформа для пошуку, пошуку та зберігання інформації ефективно позиціонується як платформа, на якій можна використовувати окремо реалізовані засоби обробки інформації.

7.1 Розробка платформи для збору, збереження інформації

Здатність охоплювати великі обсяги наукових даних вимагає використання бази даних публікацій з усіх куточків наукового середовища в Інтернеті та об’єднання їх у форму, яка полегшує їй читання та відображення обраної інформації. Також слід пам’ятати, що база даних має бути віддаленою, забезпечуючи вільний доступ до неї. Під використанням бази даних слід розуміти можливість виконувати над нею певні операції. Це надсилання запиту до бази даних з необхідною інформацією. Важливо мати можливість робити більш конкретні запити, ніж звичайні, під час пошуку в усіх полях бази даних. Це означає, що, якщо це потрібно для цілей дослідження, аналітик повинен мати можливість надіслати вузький запит, наприклад, у певні поля бази даних, щоб уточнити вибірку необхідної інформації на запит. Після обробки запиту має бути повернута відповідь з результатом запиту.

7.2 Упорядкування та сортування інформації

Визначаючи термін релевантність, можна сказати, що це ступінь відповідності одиниці інформації певним потребам користувача (при використанні платформи для підтримки прийняття рішень у сфері міжнародного науково-технічного співробітництва - аналіз) Аналітикам потрібна одиниця інформації, яка матиме перевагу перед тими, що стосуються Іншої подібної інформації, пов’язаної з індикатором. Показниками можуть бути тема публікації, автор, дата створення

публікації, ресурс для пошуку публікації тощо. Щоб отримати відповідний порядок у будь-якій метриці, кожна одиниця інформації повинна бути оцінена таким чином, щоб вона якимось чином відрізнялася і могла зайняти своє власне місце в порядку. У джерелі науково-технічної інформації одиницею інформації є видання. Існує кілька критеріїв оцінки публікацій:

1. Кількість робіт, яку автор повинен запитати
2. Кількість посилань на авторів у базі даних
3. Кількість учасників (дописувачів)

Залежно від кількості робіт, запитаних автором – кожну тему дослідження має розробляти багато людей, щоб отримати найбільш чіткі результати. Зрештою, поєднання багатьох досліджень, проведених різними дослідниками, збільшує ймовірність того, що ви знайдете правильний підхід до проблеми дослідження. Однак якість кінцевих результатів, отриманих під час роботи над темою, покращиться, якщо одні й ті самі дослідники будуть знову і знову обробляти результати попередніх досліджень за темою. Тому такі критерії існують і використовуються для оцінки публікацій, наприклад, кількість праць авторів з певної тематики.

Кількість згадок авторів у базі даних – діяльність наукової діяльності дуже важлива для сприяння прогресу. Щодня проводяться дослідження, і вчені роблять нові відкриття, які розширюють горизонти тієї чи іншої галузі науки. Людина, яка постійно цікавиться та залучається до відповідних робіт, не тільки підвищує свою цінність як експерта, але й більше з’являється в тих чи дослідженнях.

Кількість людей, залучених до публікацій (дописувачів) – неможливо не оцінити вплив колективної роботи на вирішення людських проблем. Звісно, є й вчені, які використовують свою особисту винахідливість для просування науки, але саме поєднання професіоналів дозволяє й досі дозволяє знаходити рішення глобальних проблем та шукати шляхи покращення життя людини. Звідси можна зробити висновок про критерій якості публікації, що кількість науковців, залучених до їхньої роботи, свідчить про комплексний та виважений підхід до отримання результатів.

Застарілість цієї публікації - З роками змінювалися методи дослідження, інтерпретації попередніх результатів іншими вченими. Те, що було визначено давно,

можна перевизначити шляхом повторного аналізу або дослідження, використовуючи нові інструменти часу та нові результати відповідних досліджень. Як наслідок, опублікована робота з часом може втратити актуальність, а отже впевненість у її результатах може знизитися.

Застарілість цієї публікації - з роками змінювалися методи дослідження, інтерпретації попередніх результатів іншими вченими. Те, що було визначено давно, можна перевизначити шляхом повторного аналізу або дослідження, використовуючи нові інструменти часу та нові результати відповідних досліджень. Як наслідок, раніше опублікована робота з часом може втратити актуальність, тому її результати можуть бути менш достовірними, ніж нові дослідження. Кожен критерій має свої детермінанти, чи то кількість авторів, чи скільки часу минуло з моменту публікації. Тому кожен критерій вимагає нормалізації даних (зведення показників для кожного критерію до одного типу), а потім об’єднання нормованих значень критерію публікації, щоб отримати власне значення – оцінку.

Усі винайдені інформаційні джерела повинні бути впорядковані за рейтингом та таким способом створюється послідовність. На інтерфейсі позначеннями у відсотках показується кожна публікація, стаття, тощо, що покращує та пришвидшує роботу аналітика, опираючись приймати рішення в заходах досліджень.

8 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для відмінної роботи програми потрібно дотримуватись простих умов, які забезпечують безперебійну роботу мікросервісів для збору відкритої інформації, у разі правильної інсталяції та слідкуванню усіх рекомендацій, програма буде працювати відмінно.

8.1 Системні вимоги

Програма добре оптимізована для будь яких технічних характеристик, але чим кращі показники комп’ютеру та кращого покриття інтернету, тим швидша робота програми, відповідно. Мінімальні рекомандації можуть бути наступні: процесор не нижче Intel ® Core ™ 2, операційна система не раніше Windows XP, такожне менше 30 МБ вільного місця на жорсткому диску.

8.2 Отримання доступу до бази даних CORE

Доступ до бібліографічної бази даних CORE відбувається за допомогою API-key, якого потрібно отримати попередньо зареєструвавшись на сайті платформи CORE, що зображено на рисунку 5,6,7, та зареєструвавшись через поштову скриньту Gmail. Після реєстрації, на пошту прийде API ключ, який безпосередньо важливий для з’єднання з платформою CORE.

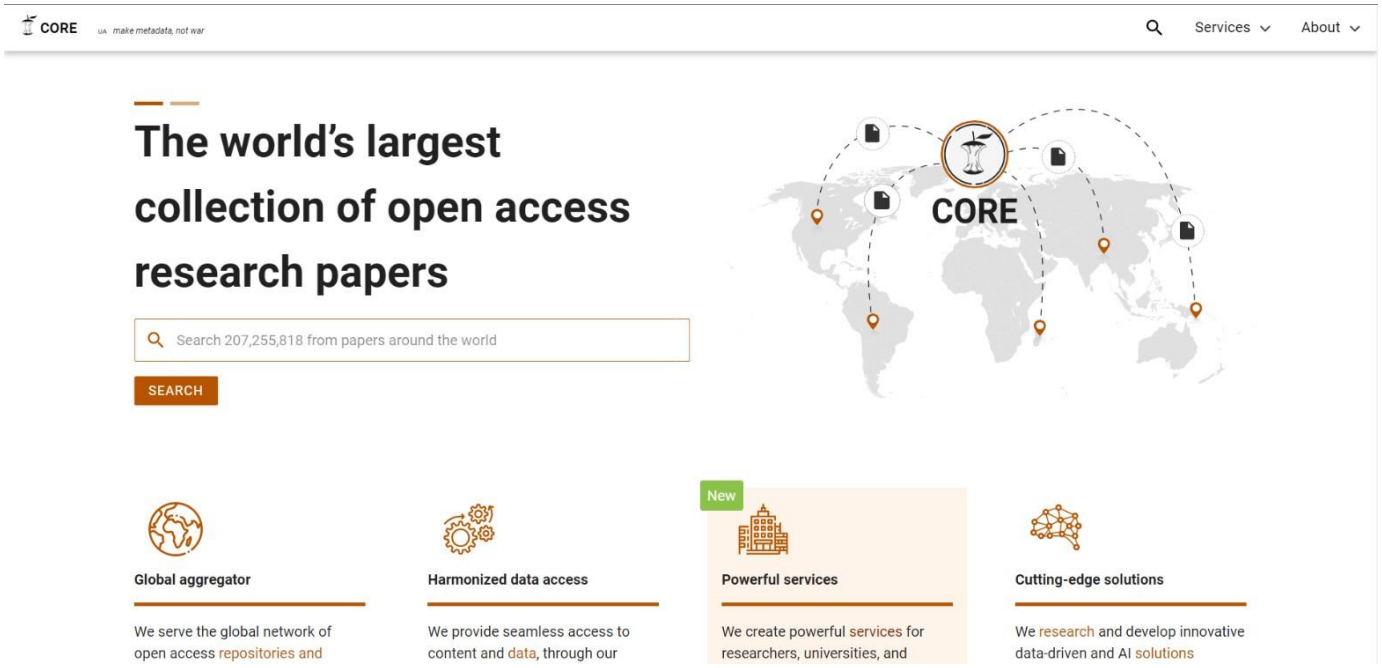


Рисунок 8.1 – Інтерфейс сайту CORE

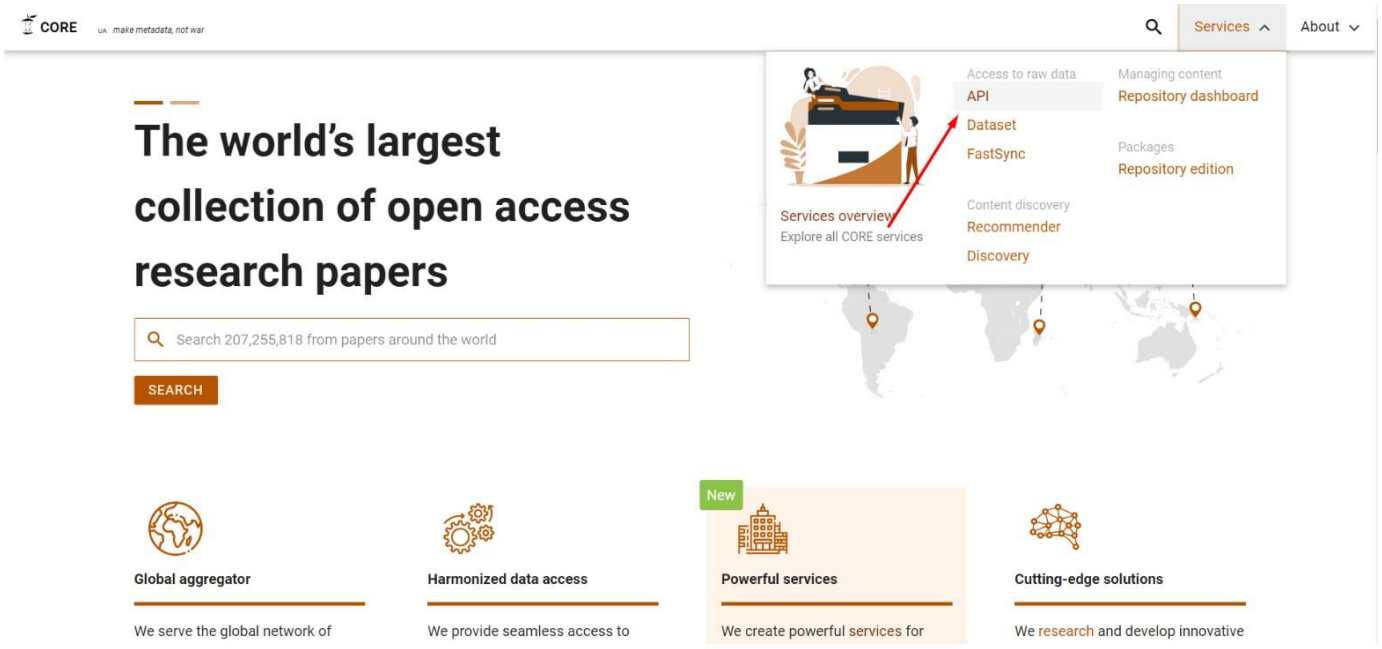
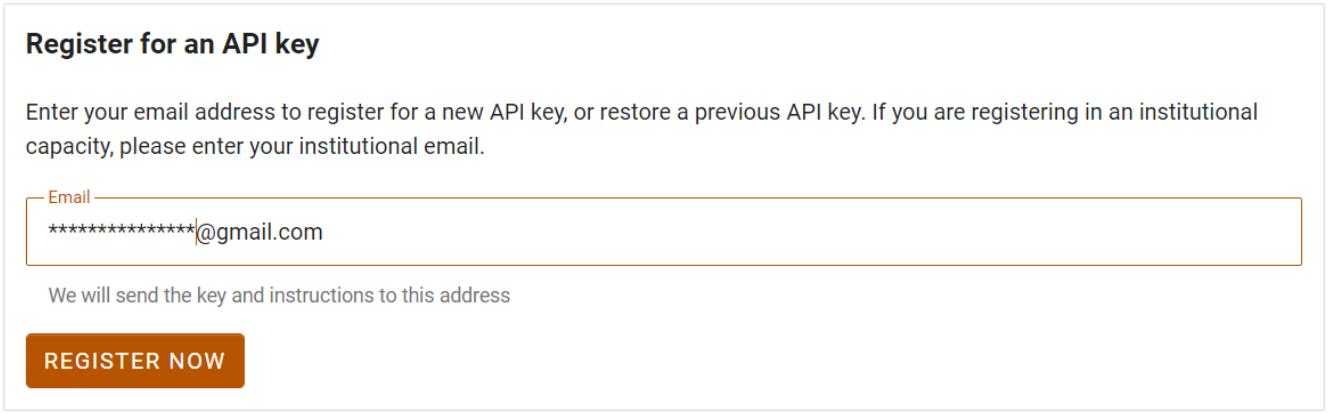


Рисунок 8.2 – Процес отримання API ключа сайту CORE



` Рисунок 8.3 – Процес отримання API ключа сайту CORE

Після того, як ми ввели пошту, очкікуємо на повідомлення. Повідомлення має наспуний вигляд, як на рисунку 8.

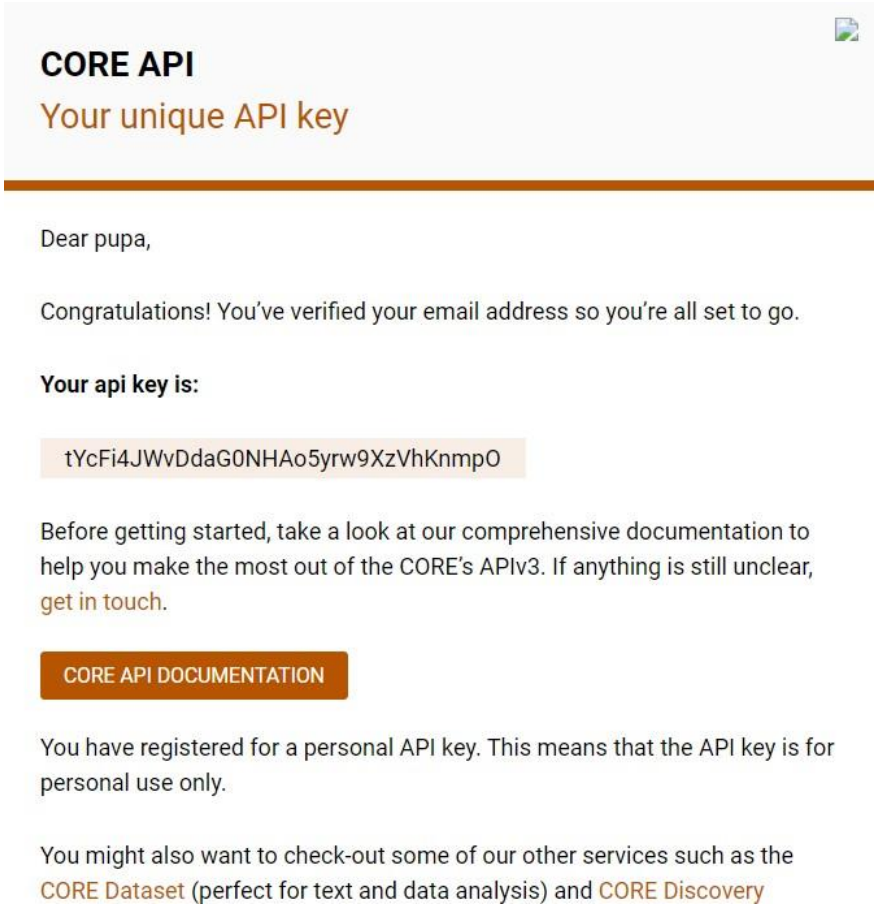


Рисунок 8.4 – Вигляд повідомлення для отримання API ключа

Вносимо ключ у файл config.json, рисунок 9.

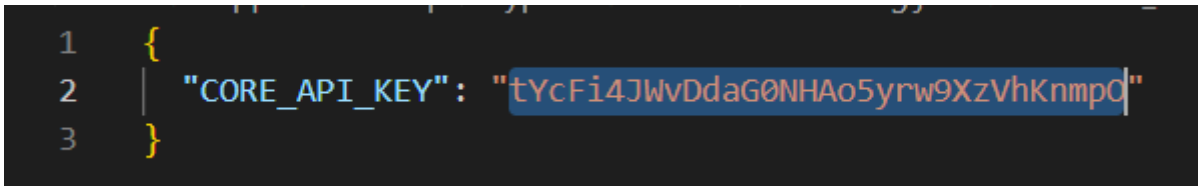


Рисунок 8.5 – Внесення API ключа у систему.

Користування програмним продуктом для збору відкритої інформації

Для пошуку, збору, аналізу інформації, тощо, в пошукове поле вводимо ключові слова, як зображено на рисунку 10.

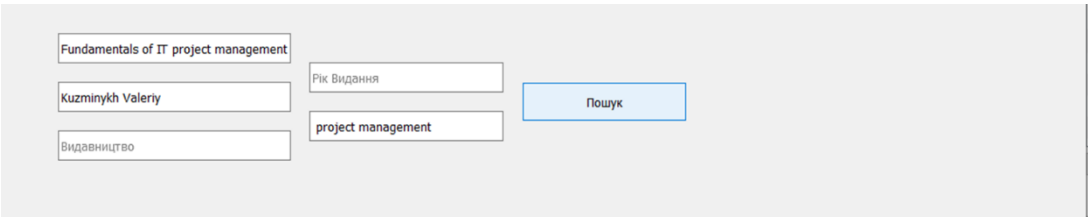


Рисунок 8.6 – створення запиту для збору інформаціїї

Після проведеного збору, отримуємо наступне, рисунок 11.

	Rate	Title	Authors
1	100%	Fundamentals of IT project management	Kuzminykh Valeriy Oleksandrovich, Otrokh Sergey Ivanovich, Taranenko Ruslan Anatolyevich, Voronko ...
2	97%	Fundamentals of IT ...	Kuzminykh Valeriy Oleksandrovich, Otrokh Sergey ...
3	91%	Audio interval ...	Kuzminykh Ievgeniia, Shi...Stavros, Shevchuk Dan, ...
4	78%	Investigation of the ...	Kuzminykh Ievgeniia, Sokolov V. Y., Yevdokymenko ...
5	68%	Impact of Network ...	Kuzminykh Ievgeniia, Silonosov Alexandr, Ghita ...
6	61%	Economic ...iciency ...	Kuzminykh Roman
7	57%	Possibilities for a ...	Kuzminykh L, Gutko A, Suvorova O
8	55%	The Chall...es with ...	Ghita B, Such JM, Kuzminykh I
9	55%	Erzählen im ...	KUZMINYKH KSENIA

Рисунок 8.7 – результат запиту для збору інформаціїї через ключові слова

8.4 Вилучення та пошук інформації за допомогою core_api

Надсилання запитів до бази даних CORE відбувається за допомогою пакетів через CORE API-key, та називається api який складається з чотирьох класів

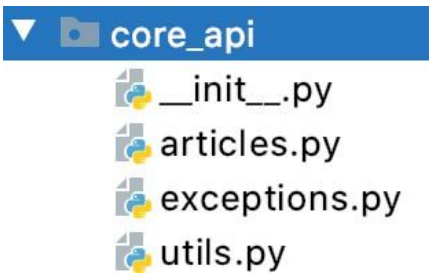


Рисунок 8.8 – пакет core_api, що складається з чотирьох модулів

init.py - Ініціалізує модуль. Python повинен ініціалізувати файл .py, щоб розглядати каталог як пакет; це робиться для того, щоб каталог із загальним іменем не приховував дійсний модуль від появи пізніше в шляху пошуку модуля.

Якщо ви використовуєте CORE API, init.py зчитує спеціальний ключ доступу, наданий CORE, і перевіряє, що це дозволений ключ CORE за допомогою функції update_api_key() у модулі utils.py. Щоб отримати дозвіл на використання бази даних через CORE API, потрібно зареєструватися та отримати приватний ключ доступу, який потрібно зберегти у файлі JSON, наприклад config.json.

Імпортуйте функцію update_api_key() init.py з модуля utils.py

```
def update_api_key(api_key):
    global g_api_key
    g_api_key = api_key
```

Рисунок 8.9 – приклад коду

Аутентифікація унікальності ключа у init.py

```
with open('config.json') as config_file:
    json_data = json.load(config_file)

    if 'CORE_API_KEY' in json_data:
        update_api_key(json_data['CORE_API_KEY'])
```

Рисунок 8.10 – приклад коду

exceptions.py – визначання класів вилучень, які мусять спрацьовувати при певних помилках.

utils.py – клас який безпосередньо відповідає за сполучення з базою даних CORE через CORE API-key.

```
import requests
from .exceptions import GeneralHTTPException, MalformedJSONRequest, InvalidAPIKey, \
                        TooManyQueriesInRequest, TooManyRequests, ProxyError, ResponseTimeout, \
                        PageNotFound, InternalServerError

import logging
import time
```

Рисунок 8.11 – приклад коду

Бібліотека запитів Python – запити – дозволяє надсилати HTTP-запити до Python. Оскільки HTTP-запити надсилаються, а відповіді отримуються під час використання API, бібліотека запитів відкриває можливість використання API в Python.

HTTP-запити є основою всесвітньої павутини. Браузер надсилає запит на веб-сервер. Сервер відповідає, надсилаючи всі дані, необхідні для відображення сторінки, а браузер відображає сторінку, щоб її можна було побачити. Процес виглядає так: клієнт (наприклад, браузер або скрипт Python, який використовує бібліотеку запитів) надсилає дані на URL-адресу, сервер зчитує дані з цієї URL-адреси, вирішує, що з ними робити, і надсилає відповідь клієнту. . Потім клієнт може вирішити, що робити з отриманими даними відповіді.

У рамках запиту клієнт надсилає дані через метод запиту. Найпоширенішими методами запиту є GET, POST і PUT. Запити GET зазвичай використовуються лише для читання даних без їх зміни, тоді як запити POST і PUT зазвичай призначаються

ВИСНОВОК

Отже, в процесі опанування мети дипломної роботи було проведено аналіз над інструментами та засобами, які надають можливість отримувати доступ до потрібної інформації.

Друге, ефективний шлях вирішення проблеми побудови платформи пошуку, пошуку та зберігання інформації для оцінки міжнародної науково-технічної діяльності.

Третє, алгоритми та програмне забезпечення розробляються, щоб надати аналітиків необхідну інформацію та оцінити їх за релевантністю відповідно до вимог.

Також, платформа, розроблена для підтримки прийняття рішень у сфері міжнародного науково-технічного співробітництва, дає чудову можливість допомогти фахівцям у галузі науково-технічного прогресу проводити дослідження для вирішення глобальних проблем та проблем комфортного проживання. Розробки та подібні інструменти для підтримки прийняття рішень у сфері міжнародного науково-технічного співробітництва можуть допомогти вченим працювати над покращенням життя людей за умови їх правильного використання, розширеної функціональності та швидкості обробки запитів у майбутньому.

Розроблена система на базі мікросервісів, яка в свою чергу дозволяє збирати інформацію з відкритих джерел про міжнародну діяльність. Змість цих самих джерел, у яких відбувається збір інформації без втручання до інших складових системи. Кожний окремий мікросервіс для обробки інформації має змогу надавати тільки часткові дані. Система є адаптивною до запитів користувачів тим чином, що інформація береться з тих джерел, які будуть відповідати вимогам запитів користувачів та зможуть задовольнити їх.

Системний продукт також вирізняється серед інших своєю адаптивністю до навантаження система та її високошвидкісною пропускною здатністю, до якої

призвело використання новітніх рішень в програмуванні так організації архітектури програмного середовища.

Програмний продукт надає користувачеві одну точку входу для запиту зі збору інформації через запит, щоб отримати аналітичну інформацію щодо міжнародної діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ford Neal, Parsons Rebecca, Kua Patrick. Building Evolutionary Architectures: Support Constant Change: O'Reilly Media, 2017. 332 p
2. Chris Richardson: Microservices Patterns: Manning Publications, 2019. 544 с.
3. S. Newman, Building Microservices – Designing Fine-Grained Systems: O'Reilly: 2015. 278 p
4. Christudas Binildas. Practical Microservices Architectural Patterns: Event-Based Java Microservices with Spring Boot and Spring Cloud: Apress: 2019. 456 p
5. 1. Qt Framework - Qt Features / – Режим доступу до ресурсу: https://www.qt.io/product/features
6. 2. Python 3 и PyQt 5. Разработка приложений. 2-е издание / Прохоренок, Дронов
7. 3. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces 1st Edition / Mark Masse
8. 4. Обзор HTTP / С.М. Тимачев
9. Python for Data Analysis. Data Wrangling with Pandas, NumPy, and Ipython / Wes McKinney
10. Learning pandas / Michael Heydt
11. Continuous API Management / Mehdi Medjaoui, Erik Wilde, Ronnie Mitra, Mike Amundsen
12. Python API Development Fundamentals / Jack Chan, Ray Chung, Jack Huang
13. Building RESTful Python Web Service / Gaston C. Hillar
14. Kreibich J. Using SQLite / Jay A. Kreibich., 2010. – (O'Reilly Media, Inc.).
15. McKinney W. Python for Data Analysis / Wes McKinney., 2012. – (O'Reilly Media, Inc.).
16. Modernanalyst. Use Cases Diagrams [Електронний ресурс] – Режим доступу: https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/2788/V erifying-Use-Cases-Data-Flow-Diagrams-Entity-Relationship-Diagramsand-State-Diagrams-via-State-Linkages.aspx 4. Pilgrim M. Dive into Python / Mark Pilgrim., 2012. – (CreateSpace).
17. PyQt [Електронний ресурс] – Режим доступу: https://doc.qt.io/qtforpython/#documentation
18. Sidi A. Practical Extrapolation Methods / Avram Sidi., 2010. – (Cambridge University Press).
19. SciPy docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.ht ml.
20. Summerfield M. - Programming in Python 3: A Complete Introduction to the Python Language / Mark Summerfield., 2008. – (Addison-Wesley Professional).
21. SQLite [Електронний ресурс] – Режим доступу: https://www.sqlite.org/docs.html 10.Brezinski C. Extrapolation and Rational Approximation / C. Brezinski, M. Redivo-Zaglia., 2020. – 406 с. – (Springer, Cham).
22. Microservices Architecture [Електронний ресурс] — Доступ до джерела: https://www.bmc.com/blogs/microservices-architecture/

23. Docker Compose [Електронний ресурс] — Доступ до джерела: https://habr.com/ru/company/ruvds/blog/450312/
24. Docker Architecture [Електронний ресурс] — Доступ до джерела: https://k21academy.com/docker-kubernetes/docker-architecture-docker-engine-components-container-lifecycle/
25. PyCharm guide [Електронний ресурс] — Доступ до джерела: https://realpython.com/pycharm-guide/

Додаток А

Використання мікросервісів для збору відкритої інформації

ТЕКСТ ПРОГРАМНОГО МОДУЛЮ

Аркушів 9

Київ — 2022

Текст програми:

```
import core_api.articles
import logging
import pandas as pd
from statistics import mean


def authors_hits(authors_series, progressSignal, statusSignal):
    # explode series of 1d arrays to list
    # and filter out empty authors
    authors = [author for authors_arr in authors_series for author in authors_arr if author]

    logging.info('Authors n: \'{n}\".format(n=len(authors)))
    # remove duplicates
    authors = list(dict.fromkeys(authors))
    logging.info('Unique authors found: \'{n}\".format(n=len(authors)))

    progressSignal.emit(0, len(authors))

    # find hits in core for every author
    core_hits = []
    for index, author in enumerate(authors):
        statusSignal.emit('Retrieving rating for author \'{author}\".format(author = author))
        core_hits.append(core_api.articles.get_author_hits_n(author))
        progressSignal.emit(index, len(authors))

    self_hits = []
    for author in authors:
        hit = 0
```

```
        for authors_arr in authors_series:
            for raw_author in authors_arr:
                if author == raw_author: hit += 1

        self_hits.append(hit)

    return pd.DataFrame({'self': self_hits, 'core': core_hits}, index = authors)


def rate(articles, progressSignal, statusSignal):
    authors_rates = authors_hits(articles['authors'], progressSignal, statusSignal)[['core',
'self']]\
                .apply(lambda row: row['core'] * row['self'], axis=1)

    authors_rates = authors_rates.apply(lambda r: r / authors_rates.max())

    rates = pd.DataFrame(index = articles.index)
    rates['authors'] = articles['authors'].map(lambda arr:
                                        0 if not arr
                                        else mean(map(lambda a: authors_rates[a], arr)))

    rates['year'] = articles['year'].map(lambda y: 0 if pd.isna(y) else y / articles['year'].max())

    rates['contributors'] = articles['contributors'].map(lambda arr: len(arr)) / \
                        (max(articles['contributors'].map(lambda arr: len(arr))) + 1)

    reduced_rates = rates.apply(lambda r: r['year']*2 + r['contributors'] + r['authors']*3, axis
= 1)
```

```
    return reduced_rates.apply(lambda r: r /
reduced_rates.max()).sort_values(ascending=False)
import logging
import core_api.articles
import json
import pandas as pd
import re
import os.path
from core_api.exceptions import GeneralHTTPException

json_ids_dump_path = 'dumps/{query}_found_articles_ids.json'
json_data_dump_path = 'dumps/{query}_fetched_articles_data.json'
pkl_dump_path = 'dumps/{query}_fetched_articles_data.pkl'



def find_and_fetch_articles(query, progressSignal, statusSignal):
    logging.info('[algo] Searching for query: \'{query}\".format(query=query))

    if os.path.isfile(pkl_dump_path.format(query=query)):
        logging.info('[algo] Found cached results for this query, using them')
        statusSignal.emit('Found cached results for this query')
        progressSignal.emit(1, 1)

        return pd.read_pickle(pkl_dump_path.format(query=query))

    ids = core_api.articles.search(query, progressSignal, statusSignal)
    logging.info('[algo] Found \'{n}\' matching articles'.format(n=len(ids)))

    with open(json_ids_dump_path.format(query=query), 'w') as dump_file:
        dump_file.write(json.dumps(ids))
```

```
progressSignal.emit(0, len(ids))

articles = []
for index, id in enumerate(ids[:20]):
    logging.info('[algo] Fetching article id: \'{id}\".format(id=id))
    statusSignal.emit('Fetching article id: \'{id}\".format(id=id))

    try:
        data = core_api.articles.get(id)
    except GeneralHTTPException:
        logging.error('[algo] Cannot fetch article id: \'{id}\".format(id=id))
        continue

    # every author has coma separated firstname and lastname.
    # this is bad for pandas parser
    data['authors'] = [author.replace(',', '') for author in data['authors']]
    articles.append(data)

    progressSignal.emit(index, len(ids))

with open(json_data_dump_path.format(query=query), 'w') as dump_file:
    dump_file.write(json.dumps(articles))

articles_df = pd.DataFrame(articles)
articles_df.set_index('id', inplace=True)

articles_df.to_pickle(pkl_dump_path.format(query=query))

return articles_df
```

```
def filter_gargbage(articles):
    replace_breaks = lambda t: t.replace('\n', ' ').replace('\r', ' ').replace('\t', ' ')

    # remove raw unicode codes from text
    articles['authors'] = articles['authors'].apply(lambda arr: [re.sub(r'\\u\d.', ' ', a) for a in
arr])
    # remove trailing spaces
    articles['authors'] = articles['authors'].apply(lambda arr: [a.strip() for a in arr])
    # replace \n \t \r with spaces
    articles['authors'] = articles['authors'].apply(lambda arr: [replace_breaks(a) for a in arr])
    # replace whitespace duplicates with single space
    articles['authors'] = articles['authors'].apply(lambda arr: [' '.join(a.split()) for a in arr])
    # some authors have 'null' values, dismiss them
    articles['authors'] = articles['authors'].apply(lambda arr: [a for a in arr if 'null' not in
a.split()])
    # filter empty authors cells
    articles['authors'] = articles['authors'].apply(lambda arr: [a for a in arr if a])
    # need to interpret list as a set cause later pandas may hash that column
    articles['authors'] = articles['authors'].apply(frozenset)

    articles['title'] = articles['title'].apply(lambda t: t if 'null' not in t.split() else pd.NA)
    articles['title'] = articles['title'].apply(replace_breaks)
    articles['title'] = articles['title'].apply(lambda t: ' '.join(t.split()))

    articles = articles[articles['authors'].map(len) > 0]
    articles = articles[articles['title'].map(pd.isna) == False]

    articles = articles.drop_duplicates(['authors', 'title'])
```

```
    return articles

from algo import routines, rating
from PyQt5.QtWidgets import QTableWidgetItem
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import *
import webbrowser
import logging
from threading import Lock

redrawLock = Lock()
threadpool = QThreadPool()


class SignalsWrapper(QObject):
    progress_signal = pyqtSignal(int, int)
    status_signal = pyqtSignal(str)
    results_signal = pyqtSignal(object, object)


class SearchRunnable(QRunnable):

    def __init__(self, query, signals_wrapper, *args, **kwargs):
        super(SearchRunnable, self).__init__()
        self.query = query
        self.signals_wrapper = signals_wrapper

    @pyqtSlot()
    def run(self):
```

```
        articles = routines.find_and_fetch_articles(self.query,
self.signals_wrapper.progress_signal,
                                            self.signals_wrapper.status_signal)
        articles = routines.filter_gargbage(articles)
        ratings = rating.rate(articles, self.signals_wrapper.progress_signal,
self.signals_wrapper.status_signal)
        self.signals_wrapper.results_signal.emit(articles,  ratings)


def tbl_cell_clicked(item):
    if item.column() == 3:
        webbrowser.open(item.text())


def progress_bar_handler(w, val, max):
    with redrawLock:
        w.prgbrStatus.setValue(val)
        w.prgbrStatus.setMaximum(max)


def status_label_handler(w, status):
    with redrawLock:
        w.lblStatus.setText(status)


def results_handler(w, articles, ratings):
    with redrawLock:
        w.btSearch.setEnabled(True)
        w.prgbrStatus.setValue(0)
        w.lblStatus.setText('Done')
```

```
    w.tblResults.clear()
    w.tblResults.setRowCount(len(ratings.index))
    w.tblResults.setColumnCount(4)
    w.tblResults.setHorizontalHeaderLabels(['Rate', 'Title', 'Authors', 'URL'])
    w.tblResults.setCurrentCell(0, 0)

    download_icon = QIcon('gui/res/download.png')

    logging.info('Rates: ')
    for id, rate in ratings.items():
        row_index = w.tblResults.currentRow()


        w.tblResults.setItem(row_index, 0,
QTableWidgetItem('{rate}%'.format(rate=int(rate * 100))))
        w.tblResults.setItem(row_index, 1, QTableWidgetItem(articles.at[id, 'title']))
        w.tblResults.setItem(row_index, 2, QTableWidgetItem(', '.join(articles.at[id,
'authors'])))
        w.tblResults.setItem(row_index, 3, QTableWidgetItem(download_icon,
articles.at[id, 'downloadUrl']))

        w.tblResults.setCurrentCell(row_index + 1, 0)

        logging.info('rate: \'{rate:.3}\' id: \'{id}\' authors: \'{authors}\' title: \'{title}\' url:
\'{url}\'"
                     .format(rate=rate, id=id, authors=list(articles.at[id, 'authors']),
                             title=articles.at[id, 'title'], url=articles.at[id, 'downloadUrl']))

def bt_search_handler(w):
```

```
    w.btSearch.setEnabled(False)
    w.tblResults.itemClicked.connect(tbl_cell_clicked)


    signals_wrapper = SignalsWrapper()
    signals_wrapper.progress_signal.connect(lambda val, max: progress_bar_handler(w,
val, max))
    signals_wrapper.status_signal.connect(lambda status: status_label_handler(w, status))
    signals_wrapper.results_signal.connect(lambda articles, ratings: results_handler(w,
articles, ratings))
    threadpool.start(SearchRunnable(w.txtSearch.text(),  signals_wrapper))
```