Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

Студента Кукли Надії Денисівни
академічної групи 124–20–1
спеціальності 124 Системний аналіз
на тему: «Системний аналіз в проектуванні та оптимізації веб-сайтів з використанням Angular»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**Інституційною**	
кваліфікаційної роботи	*доцент Коряшкіна Л.С.*			
розділів:				
Інформаційно–аналітичний	*доцент Коряшкіна Л.С.*			
Спеціальний розділ	*доцент Коряшкіна Л.С.*			
Рецензент				
Нормоконтролер	*к.ф.–м.н., доц. Хом'як Т.В.*			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
Системного аналізу та управління
(повна назва)

_____ *к.т.н., доц. Желдак Т.А.*
(підпис) (прізвище,ініціали)

«____» _____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *бакалавра*

студенту *Куклі Н.Д.* **академічної групи** *124–20–1*
спеціальності: *124 Системний аналіз*
на тему «*Системний аналіз в проектуванні та оптимізації веб-сайтів з використанням Angular*»
затверджену наказом ректора НТУ «Дніпровська політехніка»
від *16.05.2023 р. №350–с*

Розділ	Зміст	Терміни виконання
1. Інформаційно–аналітичний розділ		
2. Спеціальний розділ		

Завдання видано _____ *Коряшкіна Л.С.*
(підпис) (прізвище, ініціали)

Дата видачі: *02.01.2024 р.*
Дата подання до екзаменаційної комісії: _____
Прийнято до виконання _____ Кукла Н.Д.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 64 с., 13 рис., 1 табл., 1 додаток, 8 джерел.

Об'єктом дослідження в роботі є процес проектування, розробки та оптимізації веб- каталогу науково-освітніх публікацій.

Предметом дослідження є методи та засоби проектування та оптимізації веб-каталогу.

Метою даної кваліфікаційної роботи є створення веб-каталогу для забезпечення швидкого доступу до якісної та актуальної інформації науково-освітньої тематики для науковців, викладачів, практикуючих спеціалістів та студентів.

Методи дослідження: аналіз потреб користувачів та функціональний аналіз для визначення ключових функцій сайту, методи системного аналізу при проектуванні веб-сайту, методи обробки природної мови для генерації наборів ключових слів.

В інформаційно–аналітичному розділі наведено огляд основних інструментів та технологій, використаних при проектуванні та розробці веб-каталогу. Розглянуто стратегію пошукової оптимізації з використанням ключових слів.

У спеціальному розділі здійснено проектування та розробка веб-каталогу. Розроблено програмне забезпечення, яке здійснює генерацію наборів ключових слів за описом публікацій.

Практична цінність отриманих результатів полягає в тому, розроблений веб-каталог допоможе задовольнити потреби учасників освітнього процесу в швидкому доступі до актуальної та достовірної інформації.

Ключові слова: Angular, системний аналіз, веб-каталог, пошукова оптимізація, NLP.

ABSTRACT

Explanatory note: 64 p., 13 figures, 1 table, 1 appendix, 8 sources.

The object of research is the process of designing, developing and optimizing a web catalog of scientific and educational publications.

The subject of research is methods and tools for designing and optimizing a web catalog.

The purpose of this qualification work is to create a web catalog to provide quick access to quality and up-to-date information on scientific and educational topics for scientists, teachers, practitioners and students.

Research methods: user needs analysis and functional analysis to determine the key functions of the site, methods of system analysis in website design, methods of natural language processing to generate keyword sets.

The *information and analytical section* provides an overview of the main tools and technologies used in the design and development of the web catalog. The strategy of search engine optimization using keywords is discussed.

The *special section* describes the design and development of the web catalog. Software has been developed that generates sets of keywords based on the description of publications.

The practical value of the results obtained is that the developed web catalog will help to meet the needs of participants in the educational process in quick access to relevant and reliable information.

Keywords: Angular, system analysis, web catalog, search engine optimization, NLP.

ЗМІСТ

ВСТУП

У сучасному цифровому світі інтернет-ресурси виступають як ключовий інструмент для комунікації та обміну інформацією, тож швидкий доступ до якісної та актуальної інформації є вкрай важливим для всіх людей. Проте, навіть у епоху широкого доступу до інформаційних ресурсів, існуючі платформи не завжди забезпечують ефективний пошук та високу якість необхідних матеріалів, що часто ускладнює процес навчання, здійснення наукових досліджень або практичної діяльності.

Метою даної кваліфікаційної роботи є створення веб-каталогу для забезпечення швидкого доступу до якісної та актуальної інформації освітньої тематики для науковців, викладачів, практикуючих спеціалістів та студентів. Застосування методів системного аналізу забезпечує комплексний підхід до проектування та розробки, охоплюючи всі етапи починаючи від визначення потреб користувачів і закінчуючи кінцевою реалізацією проекту. Такий підхід гарантує ефективність та якість проектування, забезпечуючи врахування всіх необхідних аспектів і вимог до проекту.

Для досягнення даної мети необхідно провести аналіз цільової аудиторії, визначити список функціональних вимог, розробити архітектуру веб-каталогу, включаючи модулі й компоненти клієнтської частини та систему навігації, спроектувати базу даних, а також виконати пошукову оптимізацію створеного веб-сайту.

Забезпечення студентів, викладачів та інших зацікавлених осіб інструментом, який спрощує процес пошуку та надає доступ до різноманітних освітніх матеріалів не лише підвищує ефективність їхньої роботи, але й сприяє поширенню та обміну новітніми дослідженнями, методиками та публікаціями.

Такий веб-каталог може стати невід'ємною складовою сучасної освітньої інфраструктури, сприяючи розвитку та поширенню знань у різних галузях.

ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Системний аналіз як метод проектування та розробки веб-сайтів

Системний аналіз є методологічним підходом до вивчення та проектування складних систем, який включає в себе процес дослідження компонентів системи та їх взаємозв’язків. Він дозволяє комплексно розглядати поставлену задачу, аналізувати різні аспекти на фактори, що можуть вплинути її розв’язання.

У загальному вигляді системний аналіз можна охарактеризувати як методологію вирішення великих комплексних проблем. Він досліджує об'єкти системи з використанням системних принципів і покликаний надавати достовірну картину розвитку і діяльності об'єкта. Важливе значення системний аналіз має при розробці та побудові складних інформаційних систем, зокрема веб-орієнтованих. Застосування підходів системного аналізу та теорії систем дає змогу забезпечити подання процесів розробки веб-сайту із необхідним ступенем деталізації, що дозволить враховувати ряд додаткових (якісних) характеристик і в умовах неповної та неточної вихідної інформації формувати раціональні рішення [1].

В якості методу проектування та розробки веб-сайтів, системний аналіз дозволяє розробникам глибше зрозуміти вимоги користувачів до проектованої системи, взаємозв'язки між її компонентами та архітектурні особливості. Для створення якісних веб-сайтів необхідно застосовувати системний підхід, який дозволяє організувати процес проектування та розробки найбільш ефективним чином. Такий підхід дозволяє структуровано підійти до процесу створення інформаційної системи, мінімізувати ризики та досягти проставлених цілей при подальшій розробці.

Серед основних аспектів системного аналізу при проектуванні веб-додатку було визначено наступні:

- Постановка цілей та визначення цільової аудиторі веб-сайту.
- Визначення функціональних та нефункціональних вимог користувачів та інших зацікавлених осіб.
- Аналіз існуючих систем та технологій, використання або інтеграція яких може позитивно вплинути на процес розробки веб-сайту.
- SWOT-аналіз аналогів серед веб-додатків,
- Моделювання системи за допомогою створення візуальних моделей, які описують структуру, поведінку та взаємодію компонентів системи. Для цього можуть бути використані UML та ER діаграми, блок-схеми тощо.
- Проектування архітектури системи. Цей процес включає в себе визначення компонентів системи та їхню взаємодію, а також вибір інструментів та технологій для реалізації проекту.
- Забезпечення ефективної взаємодії всіх компонентів системи, швидка навігація та пошукова оптимізація веб-сайту.

Створення веб-сайту – це складний та багатоетапний процес, який потребує врахування різних факторів та вимог. У контексті проектування та розробки веб-сайтів, системний аналіз використовується для того, щоб розібратися з різними аспектами веб-проекту та забезпечити його ефективне втілення.

Починаючи зі збору вимог, системний аналіз допомагає ідентифікувати потреби користувачів та бізнесу, а також визначити основні цілі проекту. Цей етап включає в себе вивчення цільової аудиторії, її поведінки та очікувань від веб-сайту. Визначення вимог є критично важливим для розуміння того, які функції та сервіси повинні бути реалізовані, і які ресурси для цього необхідні. Наступним кроком є функціональний аналіз, під час якого визначається необхідний функціонал веб-сайту та його можливості. Це включає в себе

вирішення питань щодо пошуку та сортування контенту, реєстрації користувачів, створення та редагування матеріалів тощо.

Після визначення вимог та функціональності, системний аналіз переходить до архітектурного етапу, де розробляється структура веб-сайту, його навігація та загальний вигляд. Цей етап також включає в себе аналіз технічних аспектів, таких як вибір технологій, безпека та інтеграція з іншими системами. Нарешті, системний аналіз допомагає забезпечити тестування та валідацію веб-сайту перед випуском, щоб переконатися, що він працює належним чином та відповідає всім вимогам.

Системний аналіз допомагає забезпечити комплексний підхід до розробки веб-сайту, що враховує як бізнес-потреби, так і потреби користувачів, а також технічні обмеження та можливості. Цей метод дозволяє мінімізувати ризики та помилки під час реалізації проекту, а також забезпечує більш ефективне управління процесом розробки веб-сайту.

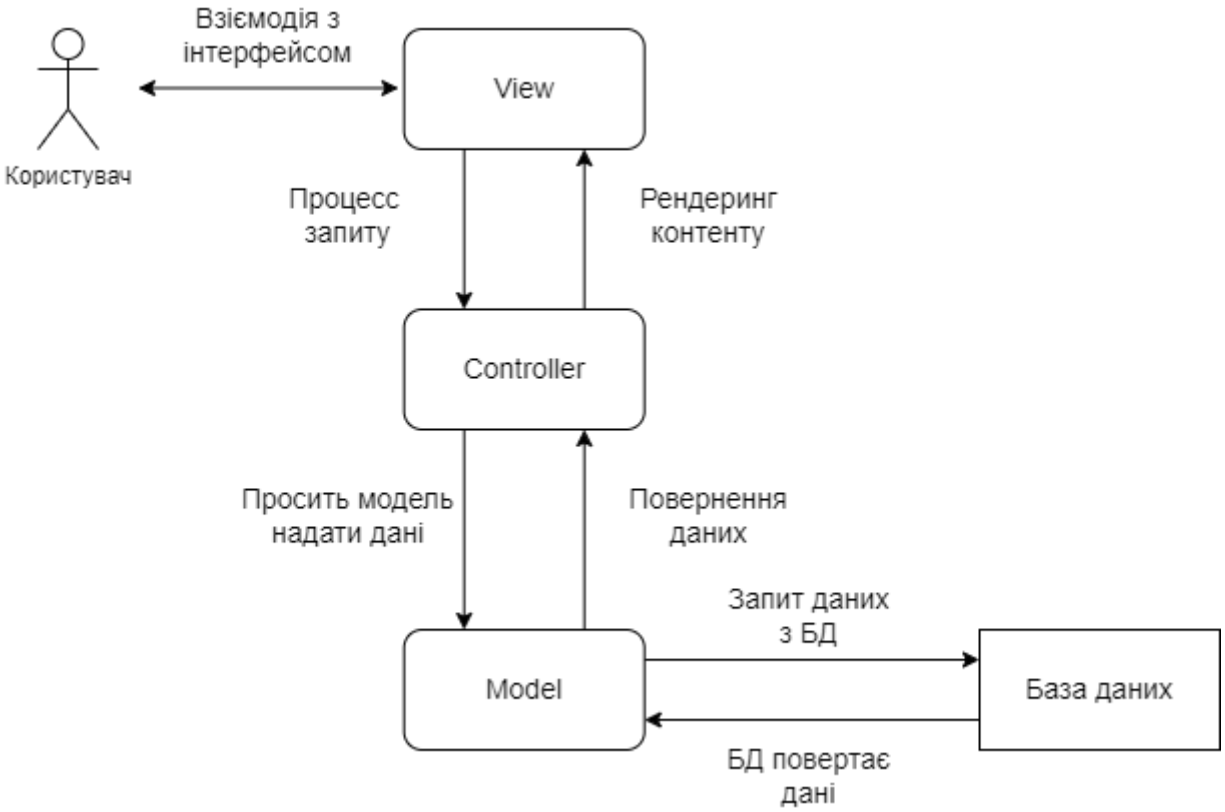
Після завершення розробки та тестування веб-сайт впроваджується в експлуатацію. Системний аналіз не завершується на цьому етапі, оскільки важливо забезпечити постійну підтримку та оновлення веб-сайту відповідно до змін у вимогах або умовах експлуатації. Регулярний моніторинг продуктивності та безпеки, а також збір зворотного зв’язку від користувачів допомагають підтримувати веб-сайт в актуальному стані і відповідати потребам користувачів.

1.2 Особливості розробки веб-сайту з використанням Angular

Angular – це потужний фреймворк для створення односторінкових застосунків (SPA), який забезпечує структурований підхід до розробки веб-додатків. Він використовує мову TypeScript, яка настійно рекомендується для

використання при написанні веб-додатків через її синтаксичну структуру, яка робить кодову базу зрозумілою та легкою для обслуговування.

Цей фреймворк дотримується архітектури Model-View-Controller (MVC), яка є шаблоном проектування, який використовується для розділення проблем у програмі, полегшуючи керування та масштабування. MVC являє собою схему розподілу даних додатків і керуючої логіки на трьох окремих компонентах – модель, представлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно.



x1.1 – Принцип роботи архітектурного патерну MVC

Розглянемо детально, яким чином Angular вписується в шаблон MVC та як відбувається розподілення на компоненти.

1. Модель (Model)
    - Модель представляє дані та бізнес-логіку програми. В Angular це зазвичай обробляється службами та моделями.

- Сервіси в Angular використовуються для інкапсуляції бізнес-логіки. Вони можуть взаємодіяти з API та іншими джерелами даних для їх отримання, обробки та керування.
- Моделі в Angular – це переважно прості класи або інтерфейси, які визначають структуру даних.

2. Вид (View)
    - Вид представляє інтерфейс користувача програми. В Angular такі інтерфейси створюються за допомогою шаблонів HTML у поєднанні з синтаксисом шаблонів Angular.
    - Angular компоненти є основним способом визначення інтерфейсів. Кожен компонент має шаблон HTML, який являє собою окремий візуальний інтерфейс, і клас TypeScript, який визначає поведінку та стан інтерфейсу.
    - Прив’язка даних Angular (як одностороння, так і двостороння) і директиви (наприклад, *ngFor, *ngIf) допомагають динамічно оновлювати вид на основі стану моделі.
3. Контролер (Controller)
    - Контролер обробляє вхідні дані від користувача, обробляє їх і відповідно оновлює модель. В Angular компоненти діють як контролери.
    - Компоненти містять логіку для обробки взаємодії користувачів (клік, перехід по посиланню, надсилання форм тощо) і виклику служб для оновлення моделі.
    - Компоненти також керують станом і можуть обмінюватися даними між моделлю та представленням, щоб забезпечити відповідне оновлення інтерфейсу користувача.

Angular сприяє розробці як прогресивних веб-додатків (PWA), так і односторінкових додатків (SPA), які можуть похвалитися своїми унікальними

функціями та перевагами як для розробників, так і для кінцевих користувачів. Перш за все, SPA має функції, які допомагають швидко завантажувати першу сторінку та підвищують продуктивність веб-сайту на малопотужних пристроях. Він також надає можливості візуалізації на стороні сервера, які покращують рейтинг SEO.

Окрім цього, в Angular є концепція лінивого завантаженя, що дозволяє завантажувати модулі або компоненти тільки тоді, коли вони потрібні, замість завантаження всього коду відразу. Це значно зменшує початковий час завантаження веб-додатка, покращуючи продуктивність та зменшуючи навантаження на мережу.

Фреймворк має модульну структуру, яка дає змогу організовувати код у різні сегменти або модулі, що робить організацію функцій програми бездоганною. Це означає, що всі його компоненти, директиви, служби та канали організовані в окремі сегменти.

Компоненти Angular є інкапсульованими, тому їх можна повторно використовувати в будь-якій іншій частині програми. Такими компонентами можуть бути форми, поля пошуку, календарі, модальні вікна та багато іншого. Така особливість фреймворку також дозволяє розробникам забезпечити обслуговування веб-додатку без зайвих зусиль, оскільки добре відокремлені компоненти легко оновлювати, покращувати чи замінювати.

Вбудований функціонал ін’єкції залежностей Angular полегшує безперебійне керування та впровадження залежностей між різними компонентами. Ця практика покращує можливість багаторазового використання та тестування коду, уможливлюючи заміну залежностей за потреби.

Двостороння архітектура прив’язки даних Angular сприяє автоматизованій синхронізації між моделлю та видображенням. Це означає, що

зміни, внесені до моделі, негайно відображаються на екрані, і, навпаки, мінімізується потреба в маніпуляціях з об’єктною моделлю документа.

Механізм маршрутизації в Angular дозволяє створювати такі веб-додатки, в яких навігація між різними візуальними представленнями здійснюється без перезавантаження сторінки. Визначення маршрутів здійснюється за допомогою масиву об'єктів, де кожен об'єкт описує певний маршрут та відповідний йому компонент. Додатково можна визначити параметри маршруту, які дозволяють створювати динамічні шляхи, що можуть змінюватися залежно від даних (наприклад, id користувача). Окрім параметрів, можна вказати резолвери для шляху. Вони дозволяють завантажувати дані перед тим, як маршрут буде активовано, що забезпечує наявність необхідних даних до відображення компонента.

1.3 Огляд основних інструментів Firebase

Firebase – це платформа, розроблена Google, яка пропонує рішення, що допомагають скоротити час розробки, підвищити продуктивність та забезпечити високу якість додатків. Платформа пропонує розробникам інструменти та сервіси, необхідні для створення та підтримки бекенду їхніх застосунків. Послуги включають хостинг, автентифікацію користувачів, базу даних в режимі реального часу та інші функції.

Firebase також можна використовувати для обробки авторизації, сповіщень, зберігання файлів та інших речей, які можуть знадобитися вашим застосункам. А оскільки Google часто оновлює Firebase, нові функції завжди доступні. Крім того, консоль Firebase має гарний дизайн і робить взаємодію між Firebase Console та застосунком дуже простою [2].

1.3.1 Хмарна база даних Cloud Firestore

Кожному застосунку потрібна база даних. Для цього Firebase пропонує Cloud Firestore та Realtime Database. Обидві є базами даних NoSQL, але Cloud Firestore є переробленою та покращеною версією Realtime Database.

База даних Firebase (Firestore або Realtime Database) — це база даних NoSQL, завдяки чому вона дуже зручна для користувача. На відміну від SQL, де таблиці і рядки є нормою, бази даних NoSQL на кшталт Firebase використовують колекції зі списком документів без таблиць [3].

На рис.1.2. наведено основні відмінності між базами даних Cloud Firestore та Realtime Database, які переважно полягають у різних структурах даних та можливостях щодо масштабованості:

Рис.1.2 – Відмінності між базами даних Cloud Firestore та Realtime Database

В якості бази даних для веб-сайту каталогу було обрано Cloud Firestore – хмарну NoSQL базу даних, яка розроблена компанією Google для реалізації зберігання та синхронізації даних для веб- та мобільних додатків. Cloud Firestore дозволяє зберігати дані на віддаленому сервері, легко отримувати доступ до них і стежити за змінами в режимі реального часу. База даних

масштабується автоматично, що дозволяє працювати з великими обсягами даних та високими навантаженнями без необхідності у власному адмініструванні масштабування бази даних.

Дані у Cloud Firestore зберігаються у вигляді колекцій та документів. Документ представляє собою запис, що містить у собі будь-які поля; колекція – це поєднання декількох документів, причому кожен документ може містити вкладені колекції. Якщо порівнювати з реляційною базою даних, то колекція аналогічна таблиці, а документ відповідає запису в цій таблиці. У межах однієї колекції можуть існувати документи з різними наборами полів. Загальна структура зберігання даних в Cloud Firestore може бути подібною до дерева, де коренем є база даних, а вузлами - колекції документів, які, в свою чергу, можуть містити інші колекції або окремі документи.

Однією з важливих переваг Cloud Firestore є те, що вона може бути легко інтегрована з іншими сервісами Google, серед яких найбільш актуальними для веб-сайту каталогу можуть бути Google Analytics та Google Search Console. Це дозволяє легко використовувати дані для проведення аналізу та подальшої оптимізації веб-сайту.

Cloud Firestore ідеально підходить зберігання даних для веб-сайту каталогу, оскільки забезпечує швидкий доступ до даних та зручність в роботі з ними. Більш того, дані, отримані під час аналізу контенту сайту для пошукової оптимізації, можуть бути збережені у Firestore у вигляді документів колекцій, забезпечуючи зручний доступ до них для подальшої обробки та аналізу.

1.3.2 Хмарне сховище Cloud Storage for Firebase

Хмарне сховище Cloud Storage for Firebase є частиною платформи Firebase, розробленої компанією Google, і призначене для зберігання та управління

файлами різних типів у хмарі. Воно забезпечує надійне зберігання даних завдяки розподіленій інфраструктурі Google Cloud. Google Cloud Platform, на якій ґрунтується Firebase, забезпечує високу масштабованість та надійність інфраструктури, що гарантує доступність файлів у хмарі в будь-який час.

Cloud Storage має інтеграцію з іншими сервісами Firebase, такими як вищеописана база даних Cloud Firestore. Це дозволяє легко інтегрувати збережені файли у додатку Firebase та забезпечити доступ до них з інших компонентів. Firebase також надає ряд заходів безпеки, таких як доступ до файлів на рівні прав доступу, можливість обмеження доступу за допомогою підпису URL-адреси тощо, що дозволяє забезпечити конфіденційність та цілісність даних.

Дані зберігаються у відрі Google Cloud Storage — об’єктному сховищі ексабайтного масштабу з високою доступністю та глобальною надмірністю. Хмарне сховище для Firebase дозволяє безпечно завантажувати ці файли безпосередньо з мобільних пристроїв і веб-браузерів, з легкістю керуючи нестабільними мережами.

1.3.3 Firebase Authentication для автентифікації користувачів

Firebase Authentication – це сервіс від Google, який входить до складу платформи Firebase і надає можливості для аутентифікації користувачів у мобільних та веб-додатках. Він забезпечує розробників простими інструментами для управління автентифікацією користувачів, а також зменшує складність та час, необхіднідля реалізації безпечних систем входу.

Сервіс пропонує різні методи автентифікації, а саме:

- Автентифікація за допомогою електронної пошти та пароля. Це також включає надсилання електронних листів для скидання пароля.

- Автентифікація користувачів шляхом інтеграції з федеральними постачальниками ідентифікаційної інформації.
- Вхід за допомогою номера телефону з підтвердженням через SMS.
- Анонімна автентифікація, яка дозволяє користувачам використовувати веб-додаток без необхідності реєстрації. Якщо пізніше користувач вирішить зареєструватися, існує можливість оновити анонімний обліковий запис до звичайного.

Firebase забезпечує високий рівень безпеки для збереження даних користувачів, а також підтримує такі стандарти безпеки, як OAuth 2.0 і OpenID Connect.

1.4 Використання UML-діаграм при проектуванні веб-сайтів

UML розшифровується як Unified Modeling Language, тобто уніфікована мова моделювання, і відноситься до різноманітних методів, за допомогою яких програмне забезпечення можна візуалізувати за допомогою діаграм. UML — це складна мовна система, яка використовується для створення бізнес-процесів, програмних рішень, архітектури програм і поведінки системи.

Зрозумілість і прозорість проектів, які описані мовою UML забезпечується використанням графічних засобів – так званих UML-діаграм. Процес проектування з використанням UML-діаграм ґрунтується на двох базових принципах: орієнтації на об’єктне подання системи (тобто уявленні про систему як сукупність окремих об’єктів, які взаємодіють один з одним) і ітераційності проектування (тобто відмові від намагання з першого кроку передбачити усі функції, властивості і характеристики системи, а уточнення і розвиток системи в процесі розробки) [4].

UML визнано стандартним рішенням для моделювання розробки програмного забезпечення, оскільки вона охоплює широкий спектр областей інженерії програмного забезпечення. Такі діаграми є корисним інструментом для проектування і документування складних веб-сайтів; вони допомагають візуалізувати архітектуру, поведінку і взаємодію компонентів веб-сайту. Крім того, візуалізація допомагає виявити потенційні проблеми і неузгодженості на ранніх етапах проектування.

При проектування та створенні документації до веб-сайту найбільш вживаними є такі види канонічних діаграм:

1. Діаграма класів – використовуються для моделювання структури системи та відображає різні архітектурні аспекти. Вона показує класи, їх атрибути (властивості), методи, а також різні типи зв'язків між класами, такі як асоціації, агрегації, композиції та успадкування. Для веб-сайту каталогу діаграма класів може бути використана при моделюванні бази даних, а також для визначення сутностей (таких як користувачі, публікації тощо) та їх взаємодії, що дозволяє детально спланувати структуру даних.
2. Діаграма послідовностей – використовується для моделювання динамічних аспектів системи, демонструючи, як об'єкти взаємодіють один з одним в певному сценарії використання. Такі діаграми є корисними для візуалізації порядку виконання операцій та обміну інформацією між різними компонентами системи.
3. Діаграма станів – описує можливі стани об'єкта та переходи від одного стану до іншого. Такі діаграми є корисними при моделюванні поведінки об'єктів у різних сценаріях використання.
4. Діаграма прецендентів, або діаграма варіантів використання (use cases diagram) – використовується для візуалізації взаємодії користувачів (акторів) з системою і відображає функціональні можливості, які доступні кожній з груп користувачів. Для веб-сайту каталогу діаграми прецедентів

можуть бути використані для моделювання сценаріїв використання, таких як реєстрація користувачів, створення та редагування публікацій, пошук конкретних матеріалів тощо. Це дозволяє зрозуміти, як саме користувачі будуть взаємодіяти з веб-сайтом і які функції їм необхідні для комфортної і ефективної роботи.

Кожна з цих діаграм подробно описує та конкретизує різні аспекти моделі складної системи в рамках мови UML. Їхнє аикористання допомагає зрозуміти взаємодію та функціонал системи на ранніх етапах проектування, що сприяє виявленню потенційних проблем і неузгодженостей та забезпечує подальшу ефективну розробку веб-проектів.

1.5 Інструменти обробки природної мови

Обробка природної мови (NLP) – це галузь комп'ютерної науки та штучного інтелекту, яка займається розумінням, аналізом та генерацією людської мови з використанням комп'ютерних технологій. NLP дозволяє застосовувати алгоритми машинного навчання для створення моделей, які можуть навчатися на основі великих обсягів текстових даних. Основним завданням NLP є розробка систем, які можуть ефективно розуміти, аналізувати та генерувати природну мову.

NLP застосовується в різних областях та відкриває безліч можливостей для автоматизації та поліпшення різних аспектів людської комунікації та обробки інформації. Наприклад, можна використовувати NLP для автоматичного перекладу тексту, створення чат-ботів або віртуальних асистентів, розпізнавання та синтезу мовлення тощо.

Існує багато інструментів та бібліотек для NLP, що дозволяють розробникам створювати програми, які здатні працювати з текстом та мовою.

Вибір конкретного інструменту залежить від потреб проекту, обсягу даних для обробки та необхідного рівню точності. Для вирішення задачі генерації ключових слів з тексту для подальшої пошукової оптимізації веб-сайту було обрано бібліотеку SpaCy.

1.5.1 SpaCy – Python-бібліотека для обробки природної мови

SpaCy є потужною бібліотекою для обробки природної мови, яка особливо відома своєю швидкістю і ефективністю в порівнянні з іншими бібліотеками NLP. Бібліотека позиціонується як така, що була розроблена спеціально для використання у виробництві, і надає можливості для написання програм, які можуть обробляти і розуміти великі обсяги текстових даних. Модульний підхід бібліотеки, підтримка високопродуктивних обчислень та зручність інтеграції з іншими інструментами забезпечують широкий спектр можливостей для розробників та дослідників у сфері обробки природної мови.

Наведемо основні концепції spaCy:

- Токенізація та сегментація тексту
- Розділення тексту на слова та пунктуацію
- Розбиття тексту на речення
- Визначення частин мовлення (POS-тегування)
- Лематизація та нормалізація
- Виявлення та класифікація іменованих сутностей (NER)
- Використання вбудованих та користувацьких векторів слів
- Синтаксичний аналіз
- Поняття синтаксичних відносин
- Вилучення ключових слів

SpaCy надає попередньо навчені моделі для різних мов, які можуть бути завантажені і використовувані для різних завдань NLP. Вони включають в себе компоненти для токенізації, POS тегування, розпізнавання іменованих сутностей (NER), парсинг залежностей та векторні представлення слів. Також важливо зазначити, що SpaCy написана з використанням Cython для забезпечення високої продуктивності. Тобто, Cython дозволяє компілювати код Python до C, що дозволяє значно підвищити швидкість роботи.

Розглянемо архітектурні особливості цієї бібліотеки. Центральними структурами даних у spaCy є Doc та Vocab. Об'єкт Doc зберігає послідовності токенів і їх анотації. Об'єкт Vocab зберігає набір довідкових таблиць, що робить загальну інформацію доступною всім документів. При централізованому зберіганні рядків, векторів слів та лексичних атрибутів відсутня необхідність зберігання кількох копій цих даних. Це заощаджує пам'ять і забезпечує єдине джерело правди.

Об'єкт Doc володіє даними, а Span і Token є уявленнями, що дозволяє spaCy працювати швидко, без зайвих копій, що вказують на них. Об'єкт Doc створюється об'єктом Tokenizer, а потім модифікується in-place компонентами pipeline. Об'єкт Language координує ці компоненти. Він бере необроблений текст і відправляє його pipeline, повертаючи анотований документ [5].

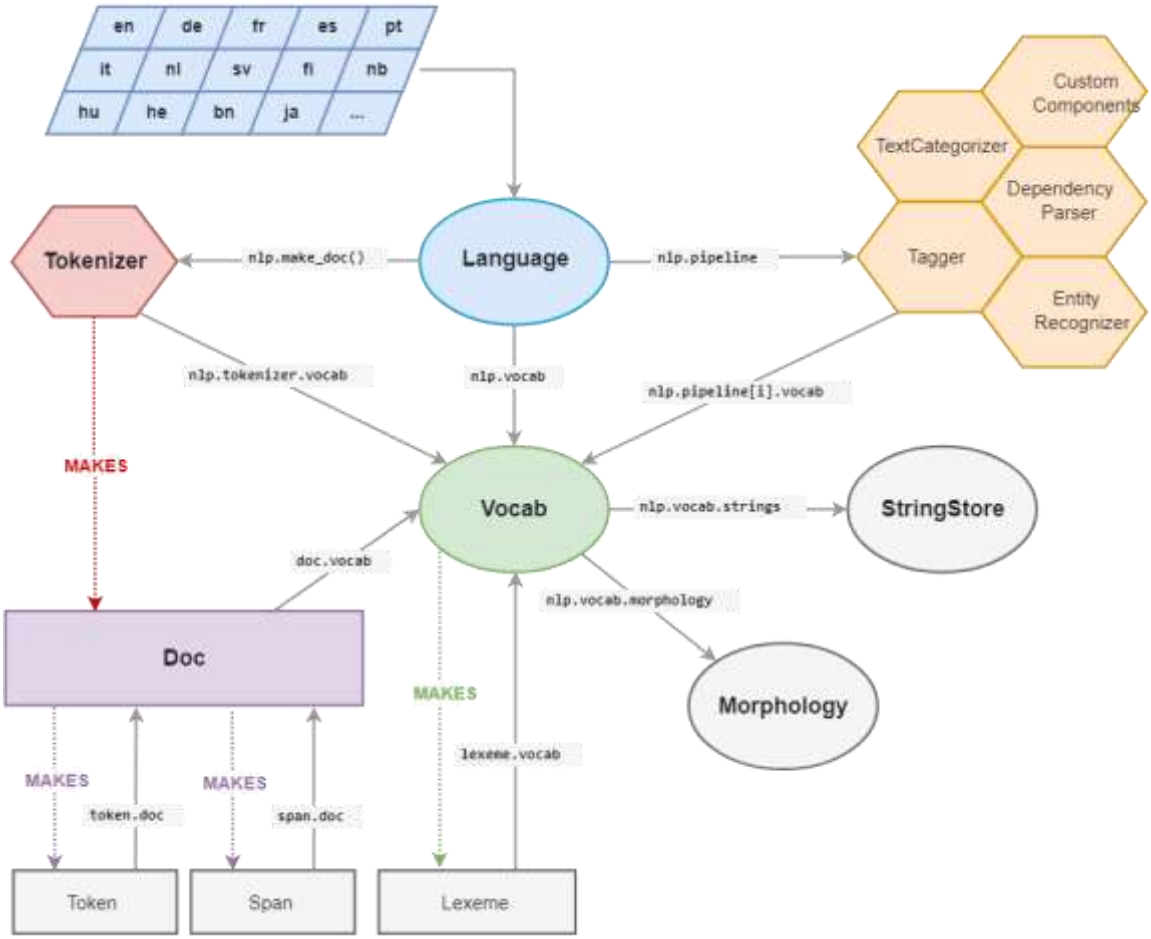


Рис. 1.3 – Архітектура бібліотеки spaCy

1.6 Стратегія пошукової оптимізації за допомогою ключових слів

Пошукова оптимізація сайту або ж SEO (search engine optimization) — процес коригування HTML-коду, текстового наповнення (контенту), структури сайту, контроль зовнішніх чинників для відповідності вимогам алгоритму пошукових систем, з метою підняття позиції сайту в результатах пошуку в цих системах за певними запитами користувачів. Чим вище позиція сайту в результатах пошуку, тим більша ймовірність, що відвідувач перейде на нього з пошукових систем, оскільки люди зазвичай йдуть за першими посиланнями [6].

Розрізняють внутрішню та зовнішню пошукову оптимізацію. У даній кваліфікаційної роботі увага була зосереджена на білій внутрішній оптимізації.

Біла оптимізація – це легальні професійні методи, що доповнюють один одного і дають стабільний результат. До внутрішньої оптимізації, перш за все, відноситься робота з мета-тегами (мета-тег Desciption, мета-тег Keywords), чітке структурування сайту, правильне оформлення головної сторінки сайту, карти сайту, навігаційного меню. Особлива увага приділяється роботі з ключовими словами. Вона починається з створення семантичного ядра сайту, формування ключових слів, розподілу їх по сторінках сайту. Оформлення контенту також повинно виконуватися з урахуванням певних вимог, які впливають на кінцеве ранжування сайту. Ще один важливий момент – це внутрішнє перелінкування сторінок. Вона має свої особливості, які обов'язково повинен враховувати оптимізатор [7].

Дослідження ключових слів є однією з основних складових пошукової оптимізації веб-ресурсів. Цей процес включає систематичний аналіз та вибір конкретних «ключових» слів та фраз, які користувачі найчастіше використовують під час веб-пошуку в Google або інших пошукових системах.

Пошукові системи користуються власними алгоритмами для аналізу та оцінки контенту веб-сторінок, їхньої індексації та подальшого ранжування. Окрім актуальності та унікальності вмісту сторінок, до уваги також береться наявність релевантних ключових слів. Такі слова допомагають пошуковим системам зрозуміти тематику контенту і надати користувачам результати згідно з їхніми запитами.

Розміщення ключових слів у заголовках, метатегах, альтернативних текстах зображень і основному контенті сторінки покращує видимість сайту у пошукових системах. Правильна кількість та оптимальне розташування цих слів впливає на позиції сайту у пошукових результатах, що забезпечує збільшення трафіку на веб-ресурс.

Після застосування ключових слів на сайті важливо регулярно відстежувати та аналізувати їх ефективність. Існують інструменти для відстеження позицій, такі як Google Search Console, щоб контролювати зміни у ранжуванні та видимості вашого сайту в результаті пошуку. Використання таких інструментів дає можливість оцінити, які ключові слова приносять найбільше трафіку, а які показують слабкі результати або втрачають свою актуальність [8].

1.7 Висновки до розділу

У даному розділі кваліфікаційної роботи розглянуто застосування системного аналізу в якості підходу для проектування та розробки інформаційних систем, зокрема веб-орієнтованих. Розглянуто види UML-даіграм, які були використані при проектуванні й візуалізації складових веб-сайту та бази даних.

Розглянуто технології та сервіси, які були використані при розробці клієнтської частини веб-сайту, зокрема функціональні можливості Angular; в якості серверної частини веб-сайту – для автентифікації користувачів, зберігання даних та завантажених матеріалів – інструменти Firebase.

Розглянуто інструменти обробки природної мови. Описано архітектурні особливості бібліотеки для обробки природної мови spaCy, яку було обрано для вирішення задачі генерації ключових слів за наданим описом для подальшої пошукової оптимізації веб-сайту. Візуалізовано архітектуру цієї бібліотеки, наведено основні концепції. Описано стратегію пошукової оптимізації за допомогою ключових слів.

СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Постановка задачі

Системний аналіз є ключовим етапом у процесі проектування та розробки веб-сайтів. Він передбачає детальне вивчення та аналіз усіх аспектів проекту з метою визначення його функціональних, технічних та бізнес-вимог. В цьому розділі наведено різні методи та підходи системного аналізу для проектування веб-сайту каталогу з науковими публікаціями, навчальними посібниками та іншими матеріалами – для забезпечення його швидкодії, зручності та відповідності потребам користувачів.

Основна задача практичного розділу кваліфікаційної роботи – проектування та розробка веб-каталогу для забезпечення швидкого доступу до якісної та актуальної інформації (наукових досліджень, публікацій, методичних посібників тощо), що стосується системного аналізу та суміжних галузей, для науковців, студентів та практикуючих спеціалістів. Каталог повинен мати зручну навігацію, простий та інтуїтивно зрозумілий інтерфейс, а також швидко обробляти запити користувачів, такі як пошук по сайту чи фільтрація контенту.

Наведемо основні принципи, які були використані при проектування веб-сайту:

- Спрямованість на користувача. Веб-сайт орієнтований на потреби користувачів, забезпечуючи швидкий та зручний доступ до актуальних наукових досліджень та публікацій.
- Доступність та безпека. Веб-сайт повинен бути доступний для користувачів з різних регіонів та забезпечувати безпеку конфіденційної інформації, такої як особисті дані користувачів та ліцензійні умови публікацій.

- Архітектура та технології. Система повинна бути побудована на надійних та сучасних технологіях, забезпечуючи ефективну роботу та швидкий доступ до інформації.
- Аналітика та звітність. Система може включати засоби аналітики для відстеження використання користувачами та забезпечення звітності щодо популярних публікацій та паттернів пошуку.
- Мобільна сумісність. Веб-сайт оптимізований для використання на мобільних пристроях, забезпечуючи зручний та приємний досвід перегляду для користувачів з різних пристроїв.

В цьому розділі також розглянуті вимоги користувачів, їхні потреби та очікування щодо функціональності веб-сайту. Це включає аналіз цільової аудиторії, її поведінки та попереднього досвіду використання схожих веб-ресурсів. На основі цього аналізу розробляються вимоги до інтерфейсу користувача, функціональних можливостей та взаємодії з веб-сайтом.

Крім того, системний аналіз включає аналіз технічних аспектів проекту, таких як вибір технологій, архітектура системи, безпека даних та інтеграція з іншими системами. Правильно спроектована архітектура дозволяє оптимізувати навігацію, полегшити доступ до інформації та забезпечити зручний користувацький досвід.

2.2 Аналіз потреб користувачів. Визначення ролей користувачів

Детальний аналіз потреб та вимог користувачів є одним із перших етапів проектування будь-якої інформаційної системи. Цей аналіз має на меті зрозуміти їхні потреби, очікування та завдання, які вони сподіваються вирішити за допомогою цього веб-сайту.

Цільова аудиторія веб-сайту – каталогу з науковими публікаціями – спеціалісти у галузі системного аналізу та суміжних областей (дослідники, викладачі, студенти, практикуючі фахівці).

Визначимо основні ролі користувачів веб-сайту. Всі користувачі будуть поділятися на дві основні групи – студенти та викладачі. До категорії студентів будуть належати користувачі, які зацікавлені в пошуку інформації для навчання, досліджень або виконання практичних робіт. Вони відвідують веб-сайт, щоб знайти необхідні навчальні матеріали, методичні рекомендації, підручники або конспекти лекцій. До категорії викладачів будуть переважно належати користувачі, які хочуть опублікувати нові матеріали або продивитися архів власних публікацій. Такі користувачі необов’язково мають бути авторами наукових публікацій – вони можуть публікувати корисні та цікаві матеріали, щоб знаходяться у вільному доступі, ділитися ними та рекомендувати студентам для ознайомлення або поглибленого вивчення теми.

Таким чином, сформуємо потреби кожної з груп користувачів:

- Студенти
    - Потреби:
        - Доступ до навчальних матеріалів для підготовки до занять, поглиблення власних знань або цитування при написанні практичних робіт
        - Пошук та фільтрація публікацій за темою, автором, ключовими словами тощо
        - Можливість завантажити матеріали публікації на телефон/планшет/комп’ютер
- Викладачі
    - Потреби:
        - Пошук актуальних матеріалів для підготовки лекцій або практичних занять

- Публікація нових навчальних матеріалів, методичних рекомендацій, конспектів лекцій
- Редагування опублікованих матеріалів
- Перегляд списку власних публікацій
- Можливість видалення публікацій

При подальшому проектуванні веб-сайту обов’язково будуть враховані вищезазначені потреби користувачів, оскільки на їхній основі розробляється основний функціонал та варіанти взаємодії з веб-сайтом. Розуміння цих потреб дозволяє ефективно створити веб-сайт, який відповідає очікуванням користувачів і забезпечує їм зручний та ефективний досвід використання.

2.3 Використання підходів функціонального аналізу при проектуванні веб-сайту

Функціональний аналіз дозволяє детально визначити, як саме повинен сайт функціонувати веб-ресурс, які конкретні завдання він повинен виконувати та які функціональні можливості мають бути доступні для користувачів. Проведений аналіз потреб цільової аудиторії веб-сайту дозволив точно визначити, які основні функції та можливості необхідно передбачити та реалізувати на веб-сайті. Це дозволяє розробити інтерфейс, який буде зручним та інтуїтивно зрозумілим для цільової аудиторії, а також забезпечить їм увесь необхідний функціонал для взаємодії.

Було розроблено список функціональних вимог:

1. Реєстрація та авторизація користувачів:

- Веб-сайт має дозволяти користувачам створювати облікові записи та входити за допомогою облікових даних, таких як електронна пошта та пароль.
- Система демонструє інформаційне повідомлення при введенні некоректних облікових даних.
- Неавторизовані користувачі мають доступу тільки до сторінки «Каталог» веб-сайту.
- Авторизовані користувачі мають доступ до всіх функцій веб-сайту.

2. Публікації – пошук, фільтрація, CRUD-операції:

- Кожна публікація має власну сторінку, на якій знаходиться інформація про публікацію та авторів, ключові слова та посилання для завантаження матеріалів публікації.
- Кожна публікація містить заголовок, опис, список авторів, перелік ключових слів, дату публікації та посилання на матеріали для завантаження.
- Авторизований користувач має можливість створити власну публікацію.
- Користувач може переглянути список публікацій, опублікованих іншими користувачами.
- Функція сортування дозволяє користувачам сортувати публікації за заголовком, автором, описом чи датою публікації.
- Функція пошуку дозволяє користувачам шукати публікації, ввівши запит у рядок пошуку.
- Авторизований користувач може переглянути список власних публікацій.
- Авторизований користувач має можливість редагувати дані опублікованих ним публікацій.

- Авторизований користувач має можливість видаляти опубліковані ним публікації.

З метою моделювання основного потоку подій, а також візуалізації різних операцій, які користувач повинен мати можливість виконувати у системі, було створено діаграму варіантів використання:

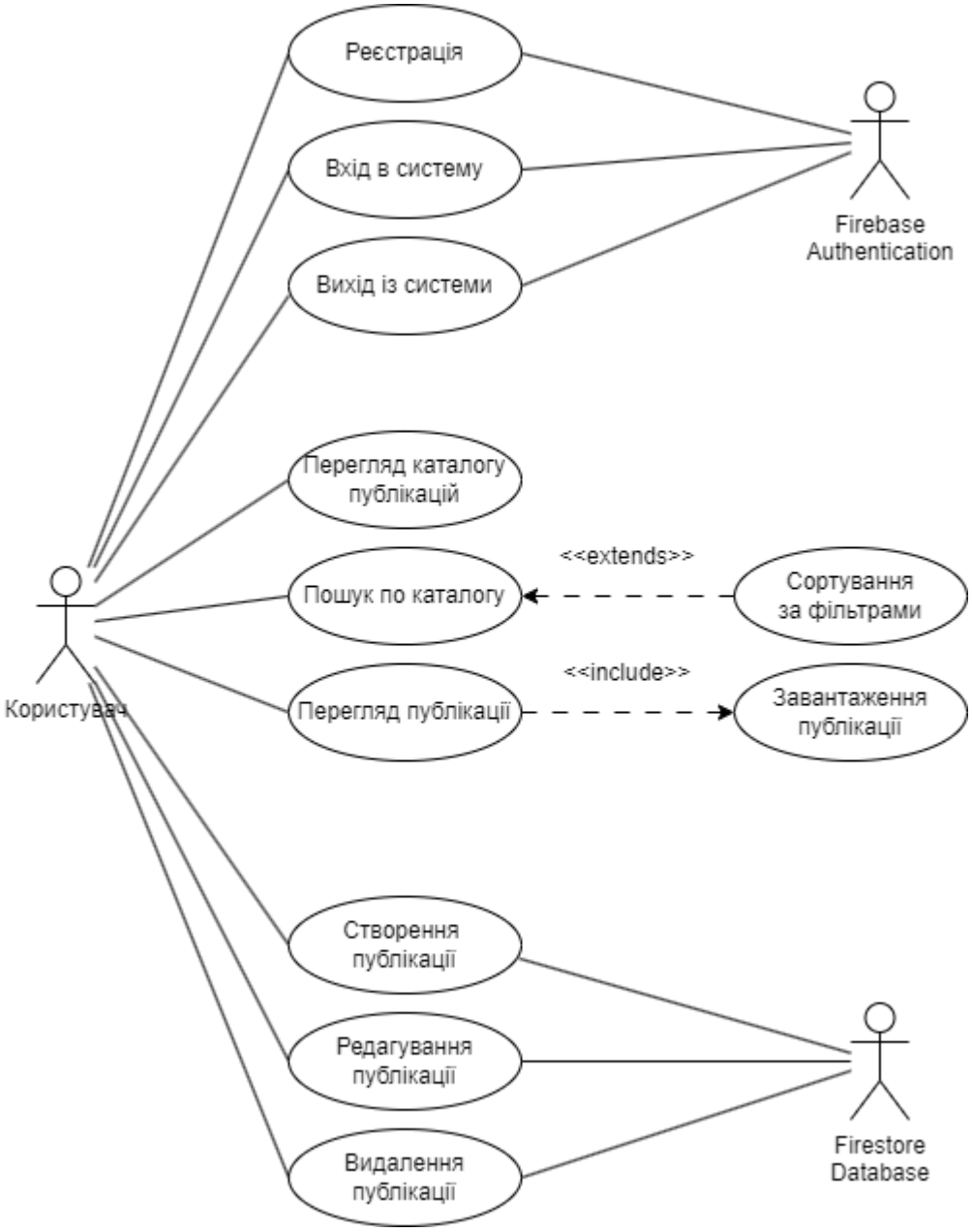


Рис. 2.1 – Use-cases діаграма веб-сайту–каталогу

Ця діаграма допоможе зрозуміти основний потік взаємодії користувачів з нашим веб-сайтом. Вона надає чітку візуалізацію того, як користувачі взаємодіють з системою та які функції вони можуть використовувати. Інформація, що надається діаграмою, може бути використана для оцінки та аналізу того, наскільки ефективно функції сайту задовольняють потреби користувачів. Було зроблено висновок, що вищеописані функціональні можливості повністю покривають вимоги цільової аудиторії. Після релізу розробленої системи необхідно провести збір інформації та відгуків від користувачів веб-сайту, щоб мати можливість виявити недоліки спроектованої системи або додаткові функції, які користувачі бажають бачити.

Важливо розуміти, що створення веб-сайту не закінчується його проектуванням та розробкою, оскільки це постійний процес, який потребує вдосконалення та врахування технологічного прогресу. При подальшій розробці діаграма може бути ускладнена новими функціональними можливостями, функціями або додатковими відомостями, які забезпечують більш детальний та повний огляд функціональності системи для розробників, дизайнерів та інших спеціалістів.

2.4 Обгрунтування вибору технологій при проектуванні веб-сайту

Для проектування клієнської частини веб-сайту–каталогу було обрано Angular, потужний міжплатформний JavaScript фреймворк, який може задовольнити потреби будь-якого бізнесу. Головною перевагою Angular є можливість створювати дивовижні високоефективні користувацькі інтерфейси, які гарантують позитивний досвід під час його використання. Окрім швидкого завантаження програм, він також забезпечує чудову продуктивність. Angular може забезпечити користувачам «майже миттєвий рендеринг» завдяки

використанню таких функцій, як Component Router, який завантажує лише код, необхідний для рендерингу.

З точки зору розробників, цей фреймворк надає безліч зручних функцій та можливостей, серед яких виділимо лише основні з тих, що були використані при проектуванні та розробці веб-сайту:

- Angular CLI (інструмент командного рядка)
- Ін’єкція залежностей
- Двустороння прив’язка даних
- Сервіси, директиви та «пайпи»
- Відкладене завантаження компонентів
- Вбудований механізм маршрутизації

Веб-сайт розроблений у технології SPA, тобто single page application - односторінковий застосунок. Такий підхід значно покращує користувацький досвід з веб-сайтом, робить його більш швидким та зручним у використанні, оскільки SPA завантажується тільки один раз. Використання SPA дозволило нам вбудувати різноманітні функції до веб-сайту з найкращою оптимізацією, такі як: пошук, фільтрація, сортування контенту. Також, новий контент може підгружатися з серверу, і при отриманні нових даних необхідні частини сторінки будуть оновлені або відмальовані знов динамічно, з використанням нових отриманих даних, без потреби завантаження усієї сторінки знов.

В ролі серверної частини для веб-сайту виступив Firebase – платформа розробки додатків типу Backend-as-a-Service. Ця платформа надає широкий спектр сервісів та інструментів, серед яких було використано наступні:

- Firebase Authentication – серверні служби та SDK для автентифікації користувачів у веб-додатку. Був обраний варіант автентифікації на основі електронної пошти та пароля.

- Cloud Firestore – хмарна база даних, що дозволяє зберігати дані на віддаленому сервері (в хмарі), забезпечує легкий доступ до них через SDK для різних платформ та надає можливість стежити за змінами в режимі реального часу.
- Cloud Storage for Firebase – хмарне сховище, побудоване на швидкій і безпечній інфраструктурі Google Cloud, являє собою ефективну службу зберігання об’єктів – наприклад, завантажених користувацьких файлів.
- Firebase Hosting – хостинг для швидкої та безпечної розгортки веб-сайту у глобальній мережі доставки вмісту (CDN).

Усі служби Firebase легко інтегруються між собою, що дозволяє створити повноцінний стек рішень для розробки гнучкого та легко масштабованого веб-додатку. Cloud Firestore також пропонує ряд інтеграцій з бібліотеками з відкритим вихідним кодом. Для розробки веб-сайту на базі Angular було використано AngularFire – бібліотеку Angular для Firebase. За допомогою AngularFire і Cloud Firestore можна додавати свої дані до документів і колекцій, надсилати запити на дані та вмикати офлайн-доступ до даних.

Дизайн сайту виконано за допомогою вже готових компонентів з Angular Material – бібліотеки компонентів користувацького інтерфейсу. Ця бібліотека пропонує красиві стилізовані компоненти для багаторазового використання, які легко налаштовуються під дизайн веб-сайту. Для відображення списку публікацій було використано компонент MatTable - таблицю з бібліотеки Angular Material. Така таблиця дозволяє найбільш ефективно та продуктивно відображати масив даних, та може бути включена до композиції компонентів, що розширять її функціонал. Серед найбільш актуальних функціональних можливостей можна виділити пошук, фільтрацію даних та пагинацію, тобто розбивку даних на сторінки.

Вибір кольорів для веб-сайту–каталогу повинен відповідати не лише естетичним вимогам, а й функціональним потребам користувачів та створювати

комфортне візуальне сприйняття контенту. Кольорова тема сайту є мінімалістичною та включає в себе базові чорний та білий кольори, а також акцентний колір морської хвилі та приглушений помаранчевий колір для обробки помилок (наприклад, при введенні неправильного паролю).

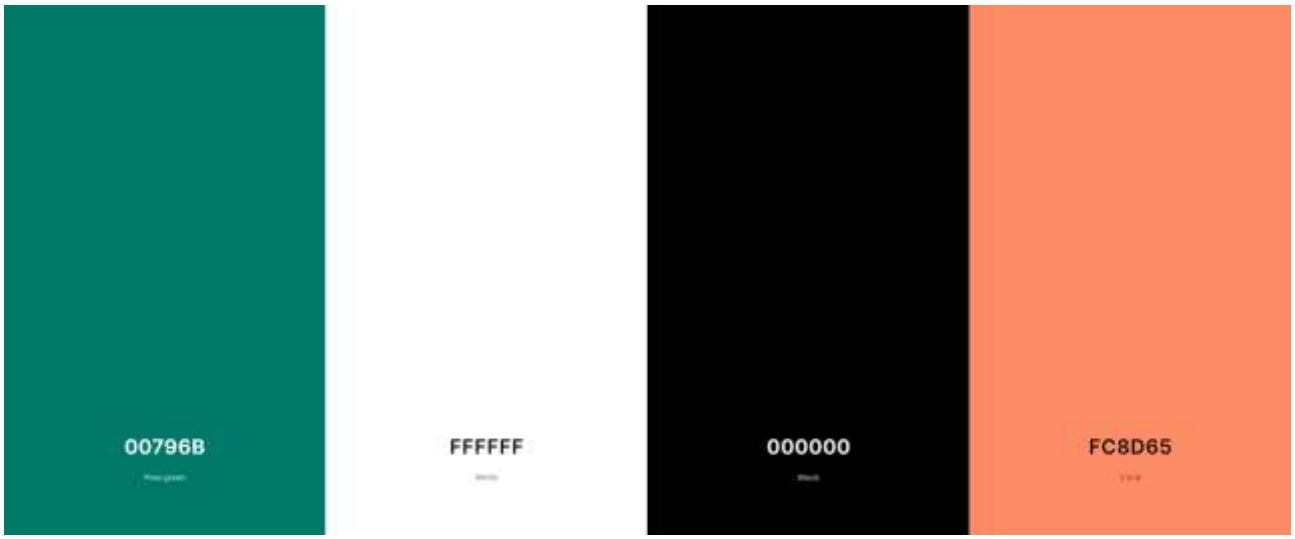


Рис.2.2 – Кольорова тема веб-сайту

Чорний та білий є базовими кольорами, які створюють контраст та забезпечують чіткість візуального сприйняття. Чорний забезпечує контур та визначеність, тоді як білий надає простору світлоту та відкритості. Ці кольори допомагають підкреслити серйозність та професійність каталогу. Морська хвиля, як акцентний колір, може додати елемент живої активності та свіжості до мінімалістичного дизайну. Крім того, він асоціюється зі спокоєм, глибиною та стабільністю, що є важливими аспектами для веб-сайту, що презентує публікації наукової тематики. Помаранчевий колір часто використовується для позначення помилок або попереджень, оскільки він привертає увагу та викликає асоціації з небезпекою або попередженням. Приглушений варіант допомагає зберегти загальний мінімалістичний стиль сайту, та при цьому чітко вказує на проблемні місця.

Таким чином, зазначена комбінація кольорів створює приємний візуальний досвід для користувачів, допомагаючи їм швидше зорієнтуватися на веб-сайті

та зосередитися на його змісті. Кольори теми гарно взаємодіють між собою, створюючи гармонійний та збалансований дизайн, а також надають сайту професійний вигляд, що відповідає його академічному спрямуванню.

2.5 Верхньорівневе проектування архітектури. Структура сайту

Верхньорівневе проектування архітектури являє собою процес створення загальної структури та організації веб-сайту на високому рівні абстракції та без деталей реалізації. Основна мета полягає в тому, щоб зрозуміти, як різні частини сайту взаємодіють між собою та з користувачем.

На рис.2.3 наведено перелік основних компонентів системи:

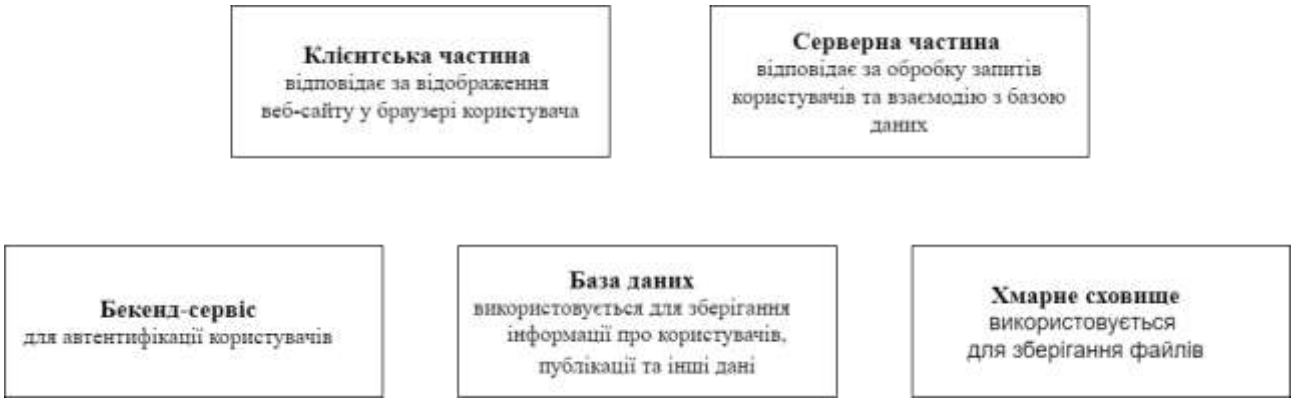


Рис.2.3 – Основні компоненти архітектури веб-сайту

Верхньорівневе проектування архітектури для веб-сайту каталогу з використанням Angular та Firebase передбачає розподіл функціональності на клієнтську та серверну частини. На клієнтській стороні Angular використовується для реалізації інтерфейсу користувача та взаємодії з користувачем. На серверній стороні Firebase виступає як хмарна платформа для зберігання даних та обробки запитів від клієнтів, а також для реалізації системи автентифікації користувачів.

Структура веб-сайту включає компоненти для різних елементів інтерфейсу, таких як бічна панель навігації, профіль користувача, сторінки каталогу та архіву публікацій, сторінки детального перегляду окремих публікацій, а також окремі компоненти для виконання різних маніпуляцій над публікаціями – завантаження матеріалів, редагування та видалення публікацій. Кожен компонент має відповідний функціонал та логіку, що реалізується за допомогою Angular.

Загальний підхід полягає в створенні ефективного та масштабованого веб-додатку, який забезпечує зручний доступ до інформації та дозволяє ефективно керувати контентом сайту. Для визуалізації архітектури було використано контекстну діаграму з нотації моделювання архітектури C4. Це найвищий рівень схеми архітектури C4, «перший погляд на систему». В ній продемонстровано тільки систему та її оточення – користувачів та зовнішні системи, з якими потрібно зробити інтеграцію.

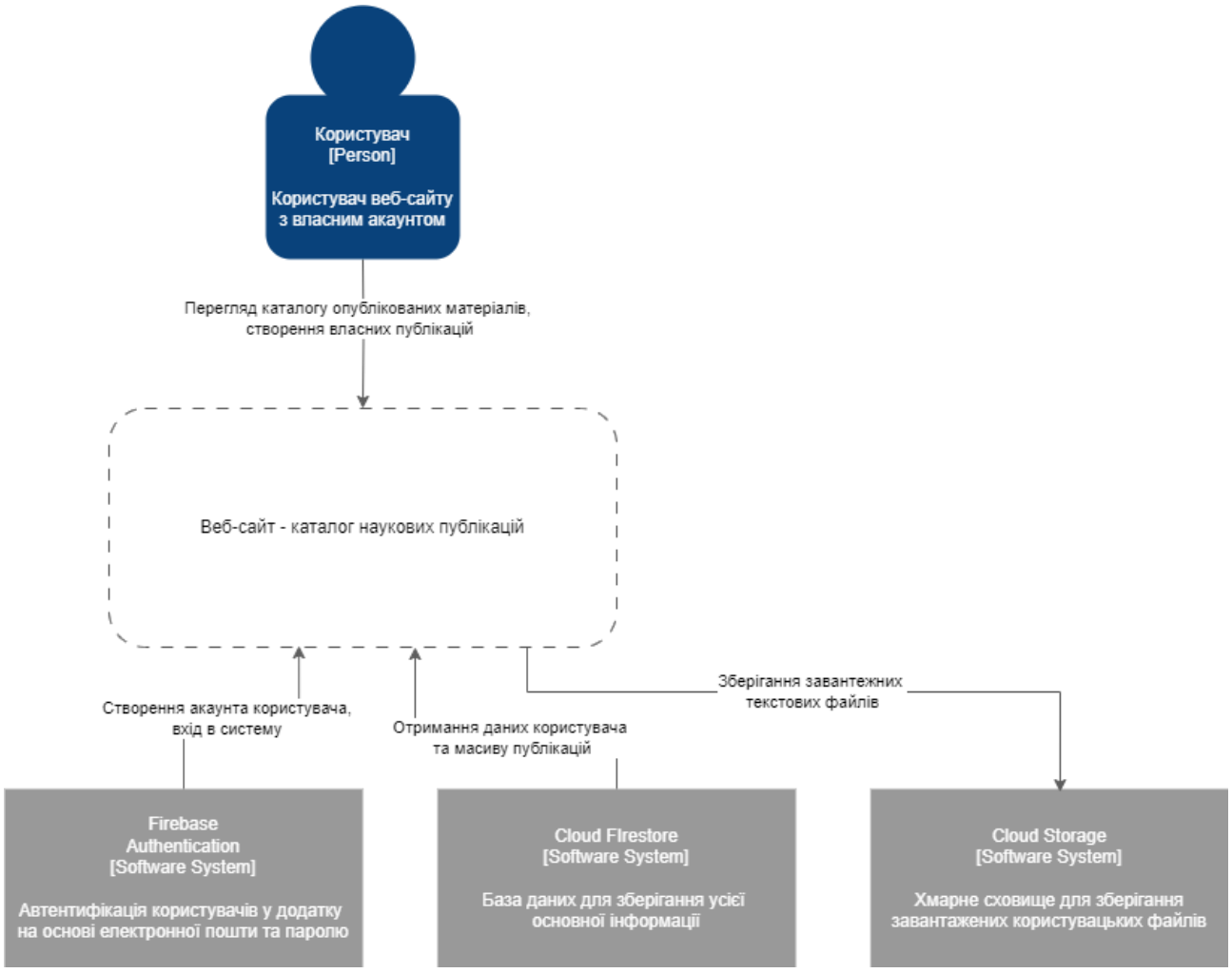


Рис.2.4 – C4 модель для візуалізації архітектури веб-сайту

Детальне пояснення щодо використаних технологій наведено у розділі 2.5. Для взаємодії між Angular та Firebase було використано офіційно підтримувану прив’язку AngularFire. Обмін даними між клієнтом та сервером відбувається у форматі файлів JSON.

Одним із ключових кроків проектування архітектури є розробка основних розділів та підпунктів сайту, а також планування маршрутизації для ефективної та швидкої навігації по сторінках веб-сайту. Було розроблено наступну структуру веб-сайту із врахуванням вимог для подальшої пошукової оптимізації:

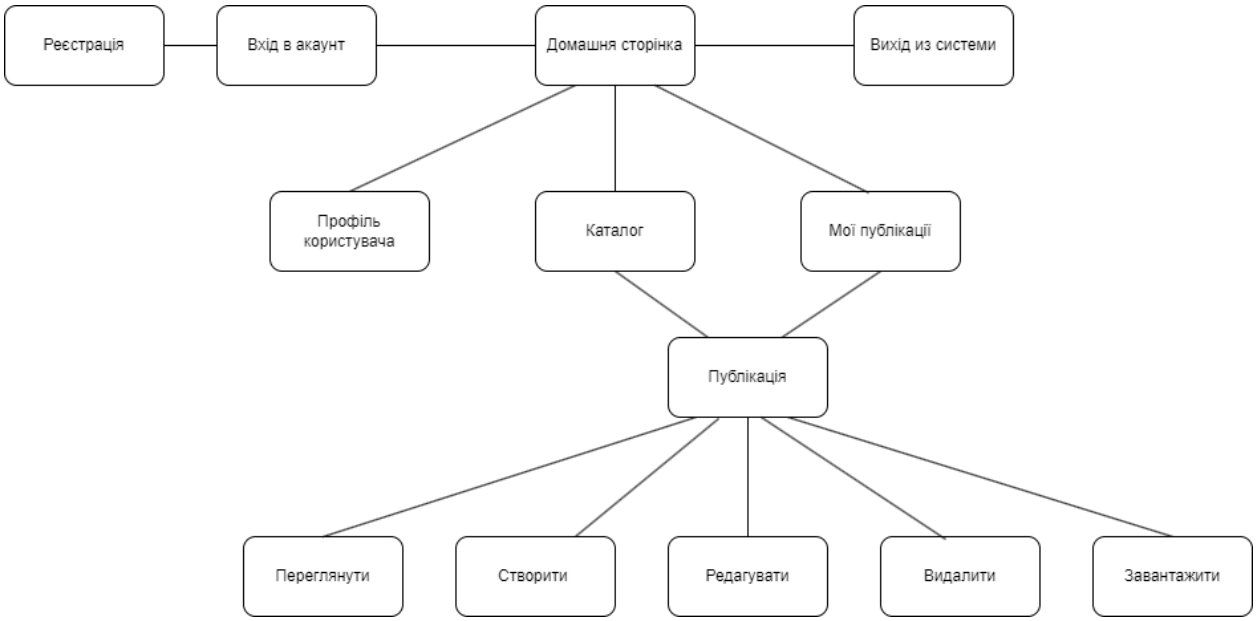


Рис.2.5 – Компонентна структура веб-сайту

При завантажені сайту користувач одразу попадає на домашню сторінку, яка пропонує можливості реєстрації для нових користувачів або автентифікації, якщо акаунт вже було створено. Після успішного входу в систему, відбувається перенаправлення користувача на сторінку каталогу публікацій, а з лівого боку екрану розроблено навігаційне меню, яке дозволяє вже автентифікованим користувачам здійснювати навігацію по сторінках веб-сайту.

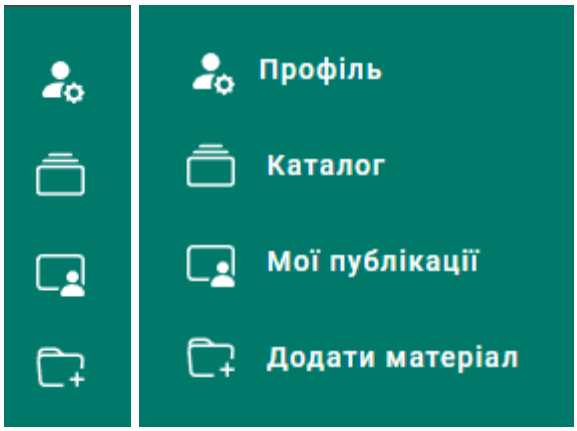


Рис.2.6 – Навігаційне меню веб-сайту (у згорнотому та розгорнотому варіантах)

Масив маршрутів та відповідних компонентів, які будуть ліниво завантажені при навігації на відповідну сторінку, наведено на рис.2.7:

```
export const routes: Routes = [
    {
        path: '',
        pathMatch: 'full',
        redirectTo: 'sign-in',
    },
    {
        path: 'sign-in',
        loadComponent: () => SignInComponent,
    },
    {
        path: 'sign-up',
        loadComponent: () => SignUpComponent,
    },
    {
        path: '',
        resolve: {
            user: CurrentUserResolver,
        },
        component: HomeComponent,
        children: [
            {
                path: 'profile',
                loadComponent: () => ProfileComponent,
            },
            {
                path: 'catalog',
                loadComponent: () => CatalogComponent,
            },
            {
                path: 'publications',
                children: [
                    {
                        path: 'list',
                        loadComponent: () => UserPublicationsComponent,
                    },
                    {
                        path: 'create',
                        loadComponent: () => CreatePublicationComponent,
                    },
                    {
                        path: ':id/edit',
                        loadComponent: () => EditPublicationComponent,
                    },
                    {
                        path: ':id/view',
                        loadComponent: () => PublicationPreviewComponent,
                    }
                ]
            },
        ]
    }
];
```

Рис.2.7 – Масив маршрутів веб-сайту

2.6 Проектування бази даних

При проектуванні бази даних перш за все необхідно ідентифікувати сутності, які будуть фігурувати у системі, та створити супровідну документацію, яка міститиме детальний опис розробленої моделі даних. Проаналізувавши вимоги до розробки веб-сайту каталогу, було визначено дві ключові сутності – «Публікація» та «Користувач». Нижче наведено атрибути (поля) кожної з сутностей, їхній опис та тип даних.

- Публікація (Publication)
    - Id – первинний ключ типу string.
    - Title – атрибут типу string. Визначає заголовок публікації довжиною до 256 символів.
    - Description – атрибут типу string. Визначає опис публікації довжиною до 2000 символів.
    - Authors – атрибут типу string. Визначає автора або перелік авторів матеріалу публікації.
    - Format – атрибут типу string. Визначає тип файлу завантаженого матеріалу.
    - Src – атрибут типу string. Визначає посилання на завантажений матеріал, збережений у сховищі Firebase.
    - PublicationDate – атрибут типу Date. Визначає дату створення публікації.
    - PublisherId – унікальний ідентифікатор типу string. Посилається на Id користувача, що створив публікацію.
    - Keywords – масив типу string. Визначає масив згенерованих ключових слів на основі заголовку та опису публікації для подальшої пошукової оптимізації веб-сайту.

- Користувач (User)
    - Id – первинний ключ типу string.
    - Email – атрибут типу string. Визначає адрес електронної пошти користувача, введений при реєстрації акаунта.
    - Name – атрибут типу string. Визначає ім’я користувача.

Наступним важливим пунктом при проектуванні та реалізації бази даних веб-сайту є створення переліку правил і ряду обмежень, які допоможуть забезпечити надійність, цілісність і масштабованість системи, унікальність екземплярів сутностей, а також гарантують, що всі дані будуть коректно зберігатися та оброблятися. Така система також має яка задовольняти усі функціональні та нефункціональні вимоги користувачів веб-сайту. Було сформовано основні правила, яких важливо дотримуватись при проектуванні бази даних веб-сайту каталогу:

1. Кожен з вищезазначених атрибутів є обов’язковим, тобто вони не можуть мати значення NULL. Це гарантує, що всі сутності бази даних будуть мати усю необхідну інформацію, яка потрібна для коректного функціонування системи.
2. Атрибути «Id», «Email» та «Src» є унікальними і не можуть мати повторів у базі даних. Це забезпечить унікальну ідентифікацію кожної сутності та допоможе уникнути повторів.
3. Веб-сайт не має обмежень на кількість створених користувачів чи публікацій.
4. Кожен користувач може створити будь-яку кількість публікацій.
5. Кожна публікація може бути створена лише одним користувачем.
6. Автор публікації може не бути одним з авторів публікації, тобто він має можливість опублікувати матеріал зі згоди його авторів.

Після визначення основних сутностей, їх атрибутів та взаємозв’язків, було створено ER-модель бази даних веб-сайту каталогу (рис.2.8):

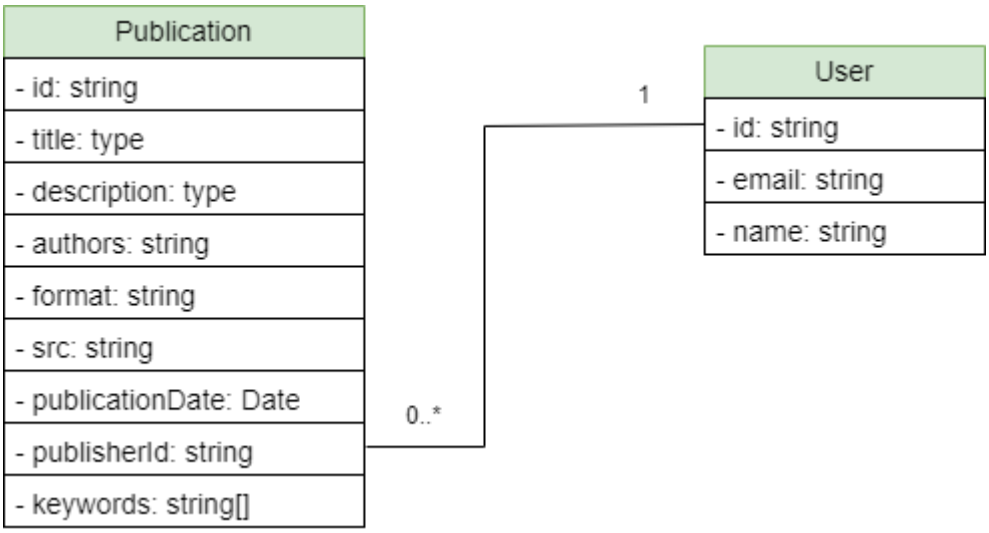


Рис.2.8 – ER-модель бази даних веб-сайту

2.7 Пошукова оптимізація веб-сайту

2.7.1 Постановка задачі

Основна задача полягає у створенні програмного забезпечення, яке здатне аналізувати дані, що містяться на веб-сайті (контент), та генерувати оптимальний набір ключових слів, що допоможе підвищити його видимість та рейтинг у пошукових системах. Таким чином, буде розроблена система визначення ключових слів, що показали найкращі результати при пошуку, для публікацій веб-сайту на основі їхнього змісту.

Для подальшої роботи використано вже спроектований та розроблений у практичному розділі сайт-каталог, який містить список публікацій, тематично пов’язаних з сферою системного аналізу. Важливо зазначити, що дані про публікації зберігаються у базі даних Cloud Firestore.

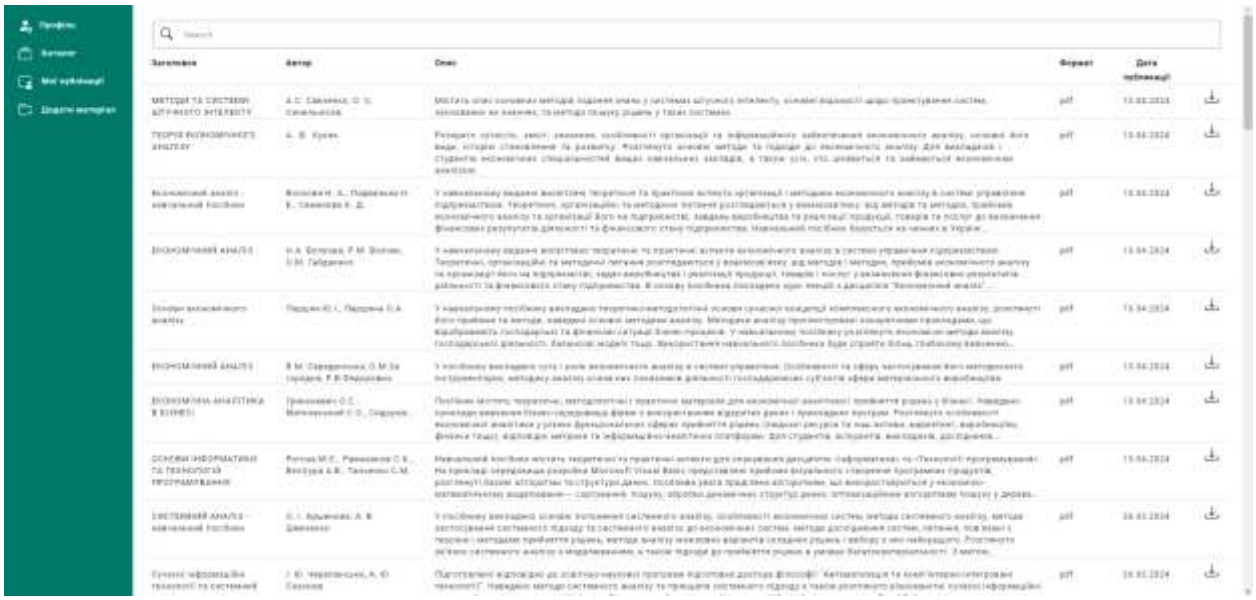


Рис.2.9 – Сайт-каталог, який містить список публікацій

Задача формування ключових слів для пошукової оптимізації є критично важливою для успішного розвитку веб-сайтів у сучасному конкурентному цифровому середовищі. Використання інтелектуальних систем та алгоритмів штучного інтелекту дозволяє ефективно оптимізувати цей процес.

Для розробки вищеописаної системи буде використано наступне програмне середовище:

- Python - мова програмування для реалізації всієї системи.
- PyCharm - IDE для Data Science та веб-розробки на мові Python
- Spacy - Python бібліотека для обробки тексту та лематизації.
- Scikit-learn - Python бібліотека для машинного навчання, яка використовується для побудови моделі класифікації.
- Firebase - сервіс для зберігання та доступу до даних.
- Google - використовується для отримання кількості результатів пошуку.

2.7.2 Аналіз вхідних даних

Генерація ключових слів, а також подальше навчання моделі будуть здійснені на основі вхідних даних, а саме: контенту публікацій веб-сайта. Для даної кваліфікаційної роботи був сформований набір даних – публікацій наукової тематики українською мовою, та збережений у Cloud Firestore. Кожна публікація містить наступні поля:

- title – заголовок публікації
- description – опис публікації
- authors – список авторів
- src – посилання на файл (посібник, видання тощо)
- format – формат файлу
- publicationDate – дата створення публікації
- publisherId – айді автора публікації
- keywords – масив ключових слів, що будуть згенеровані

На рис.2.10 продемонстровано, яким чином виглядає створений об’єкт публікації у базі даних Cloud Firestore:

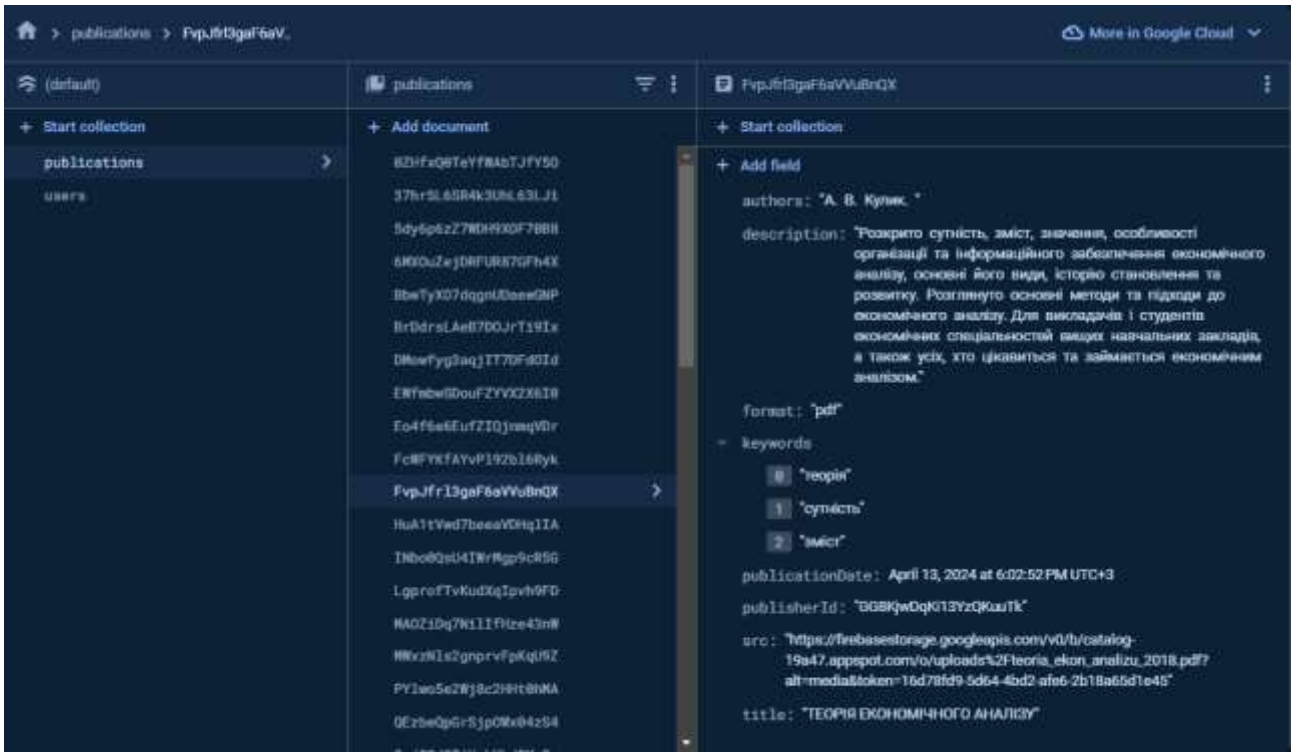


Рис.2.10 – Приклад об’єкту – публікації з усіма полями

2.7.3 Генерація ключових слів

У цьому пункті розглянуто основні кроки роботи системи для автоматичної генерації ключових слів з використанням технологій обробки природної мови (NLP) та даних пошукових запитів Google. При генерації ключових слів кожним з методів було обрано топ-3 найбільш відповідних слів. Кількість обраних слів може бути змінена при необхідності. Нижче наведено основні кроки роботи нашої системи для генерації ключових слів:

1. Збір даних. Спочатку ми здійснюємо з'єднання з базою даних Cloud Firestore, з якої отримуються дані про публікації веб-сайту.

```
cred = credentials.Certificate('credentials.json')
app = firebase_admin.initialize_app(cred)
db = firestore.client()
publications_ref = db.collection(u'publications').stream()
publications = []

descriptions_array = []
keywords_array = []
```

2. Обробка та аналіз тексту. Попередньо описи публікацій обробляються, використовуючи NLP бібліотеку SpaCy для лематизації та визначення частин мови. Використання природної обробки мови дозволяє автоматизувати процес визначення ключових слів, враховуючи граматичні та семантичні особливості мови.

```
def extract_keywords_nlp(text):
    doc = nlp(text)
    keywords = [token.lemma_ for token in doc if token.pos_ in ['NOUN']
                and not token.is_stop and not token.is_punct]
    keyword_freq = Counter(keywords)
    top_keywords = keyword_freq.most_common(8)
    return [keyword for keyword, _ in top_keywords]
```

3. Створення наборів ключових слів. Після обробки описів публікацій, для кожної публікації створюється набір потенційних ключових слів на основі їхньої популярності при пошуку, яка визначається за допомогою запитів до Google через Google Search.

```
def extract_keywords(text):
    doc = nlp(text)
    noun_phrases = [token.lemma_.lower() for token in doc if token.pos_ in ['NOUN', 'PROPN']]
    noun_phrase_counts = Counter(noun_phrases)
    keyword_phrases = []
    for phrase, count in noun_phrase_counts.most_common():
        if count >= 2:
            search_count = get_google_search_results_count(phrase)
            if search_count > 0:
                keyword_phrases.append((phrase, search_count))
    keyword_phrases.sort(key=lambda x: x[1], reverse=True)
    return [phrase for phrase, _ in keyword_phrases[:8]]
```

4. Оцінка популярності ключових слів за допомогою Google Search. Для кожного визначеного ключового слова виконується пошук у Google для визначення кількості результатів пошуку, що вказує на популярність цього слова.

```
def get_google_search_results_count(query):
    url = f"https://www.google.com/search?q={query.replace(' ', '+')}"
    headers = {"User-Agent":
                   "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 "
                   "(KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36"}
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, features: "html.parser")
    if soup.select_one('#result-stats nobr') is not None:
        result_stats_text = soup.select_one('#result-stats nobr').previous_sibling
        results_num = ''.join([num for num in result_stats_text if num.isdigit()])
        return int(results_num)
    else:
        return 0
```

5. Вибір найкращих ключових слів для публікації на основі їхньої популярності у пошукових запитах.

```
def select_best_keywords(nlp_keywords, publication_keywords):
    selected_keywords = []
    combined_keywords = set(nlp_keywords) | set(publication_keywords)
    for keyword in combined_keywords:
        popularity_score = get_google_search_results_count(keyword)
        if (keyword not in selected_keywords and
            all(token.lemma_ not in selected_keywords for token in nlp(keyword)) and
            all(token.pos_ != 'VERB' for token in nlp(keyword))):
            if popularity_score > MIN_POPULARITY_THRESHOLD:
                selected_keywords.append(keyword)
        if len(selected_keywords) == 3:
            break
    return selected_keywords
```

6. Збереження ключових слів. Після генерації набору ключових слів необхідно оновити поле 'keywords' кожної публікації, записавши туда нові дані.

```
db.collection('publications').document(publication_id).update({'keywords': best_keywords})
```

Таким чином, розроблена система дозволяє автоматизувати процес визначення та вибору оптимальних ключових слів для пошукової оптимізації веб-сайту, що сприяє підвищенню його видимості та рейтингу у пошукових системах. Повний лістинг коду генерації ключових слів для пошукової оптимізації наведено у ДОДАТКУ А.

2.7.4 Результати вирішення задачі

Набори ключових слів, які були згенеровані на основні аналізу вмісту публікацій, зокрема їхнього заголовку та опису, та додані до мета-тегів сторінок з метою підвищення видимості та рейтингу матеріалів каталогу у пошукових системах являють собою результат вирішення задачі з пошукової оптимізації веб-сайту. Для наочної демонстрації було обрано десять статей з рецензованого видання «Information Technology: Computer Science, Software

Engineering and Cyber Security». Журнал містить оригінальні статті, які висвітлюють результати наукових, практичних та навчально-методичних досліджень у сфері інформаційних технологій. У табл.2.1 занесено результати згенерованих наборів ключових слів для десяти статей:

Таблиця 2.1

Згенеровані набори ключових слів до публікацій

Бібліографічний опис	Ключові слова автора	Ключові слова, визначені програмно	Відсоток співпадінь
Зіборов, І., Желдак, Т. (2023). Еволюційний метод пошукової оптимізації на основі рою часток та моделювання штучних імунних систем	еволюційний метод, оптимізація, штучні імунні системи, ройовий інтелект, популяція, стиснення, технологія	оптимізація, метод, підходи методів рою часток, моделювання штучної імунної системи, металургійне виробництво, розв’язання задач оптимізації	54,5
Корнієнко, Д., Голян, Н. (2023). Методи прогнозування для вивчення та попередження надзвичайних природних явищ	природне явище, попередження надзвичайних явищ, архітектура персептронів, нейронна мережа, метод прогнозування, моніторинг надзвичайних явищ	виявлення природних явищ, способи дослідження, програмне забезпечення, явище, метод, прогноз	42,9
Коряшкіна, Л., Лубенець, Д. (2023). Математичні моделі та методи мультиплексного розбиття і багатократного покриття множин для задач розміщення-розподілу	логістика, зонування території, багатократне покриття множини, математичне моделювання, мультиплексне розбиття множин, потужність центру	розбиття, зонування територій, покриття, розподіл транспортних потоків, модель, математична модель, центр	53,8
Лактіонов, І., Жабко, О., Дяченко, Г.,	IoT-технологія, моніторинг,	продовольча безпека, моніторинг,	44,4

Прокопенко, М. (2023). Обґрунтування вимог до структурно-алгоритмічного забезпечення IoT системи моніторингу ґрунтокліматичних параметрів сільськогосподарських підприємств рослинництва	архітектура, бездротова сенсорна мережа, сільське господарство	економічне відновлення, технологія, сільськогосподарська продукція	
Литвинов, О., Грузін, Д. (2023). Методи оптимізації завантаження та оновлення вебсторінок за допомогою хмарних технологій	хмарні технології, рендеринг на боці сервера, статичні вебсторінки, едж кешування, кеш проксі-сервер, доменно-орієнтований дизайн	сервер, кешування, хмарні технології, проксі, кеш, система	50
Любченко, В., Чумаченко, Д. (2023). Модель якості програмного забезпечення для біонічних протезів	біонічні протези, розробка програмного забезпечення, управління протезами, тактильний зворотній зв'язок, математичне моделювання, забезпечення якості	біонічні протези, програмне забезпечення, розробка, протез, характеристика, зворотній зв’язок	64,3
Селівьорстова, Т., Красношапка, Н. (2023). Особливості проєктування масштабованої мікросервісної архітектури для вебсервісів	архітектура мікросервісів, масштабованість, вебсервіси, оркестровка, продуктивність	хмарні сервіси, мікросервіс, продуктивність, масштабування, доступність	66,6
Ус, С., Тимошенко, Л., Юдина, А. (2023). Системний аналіз регіонів України для започаткування бізнесу у сфері надання ветеринарних послуг	підприємництво, системний аналіз, прогнозування ветеринарних послуг	медицина, лікування тварин, ветеринарні послуги, бізнес, дослідження	50

Бойко, Н., Левицький, Б. (2023). Алгоритми тренування та оцінки моделей машинного навчання для структурованого набору даних	машинне навчання, дані, алгоритм, обробка даних, регресійні моделі машинного навчання, лінійна регресія, дерево прийняття рішень, випадковий ліс	машинне навчання, модель, лінійна регресія, обробка структурованих даних, аналіз	64,7

Опис та заголовок є критично важливими елементами для формування ключових слів, оскільки вони визначають основну тематику та контекст публікації. Згенеровані ключові слова мають відображати головні теми та ідеї, що містяться в тексті. Вони повинні відповідати змісту тексту, щоб забезпечити коректне співвідношення між контентом і запитами користувачів у пошукових системах.

Враховуючи це, можна побачити, що доцільність та актуальність ключових слів напряму залежить від того, наскільки якісний зміст публікації. Чим більш точно опис відображає основну тематику та ідеї публікації, тим ефективніше можуть бути вибрані ключові слова. Інформативний, якісний та релевантний опис гарантує генерацію найбільш актуальних та ефективних ключових слів, які сприятимуть підвищенню видимості та рейтингу публікації у пошукових системах. Такий опис дозволить системі аналізувати контент і визначати головні теми та ключові слова, які найкраще відображають зміст публікації. Як результат, вибрані ключові слова будуть максимально релевантними та сприятимуть залученню цільової аудиторії до контенту. Таким чином, створення інформативного, якісного та релевантного опису є важливим етапом у процесі оптимізації контенту для пошукових систем.

Отже, результати роботи програми вказують на успішність системи у виконанні аналізу та генерації ключових слів для кожної публікації на веб-сайті, враховуючи їх зміст та специфіку. Ці ключові слова є результатом

обробки тексту публікації з використанням методів обробки природної мови (NLP) та аналізу їхньої популярності у пошукових системах. Вони можуть бути використані для пошукової оптимізації контенту та підвищення його видимості у веб-пошуку, що відповідає основній меті системи. Такий підхід дозволяє автоматизувати процес вибору ключових слів і забезпечує ефективну стратегію SEO для веб-сайту.

2.8 Висновки до розділу

У другому розділі кваліфікаційної роботи було здійснено аналіз потреб користувачів та визначення їхніх ролей, що є одним із перших етапів проектування будь-якої інформаційної системи. Проведений аналіз дозволив розробити список функціональних вимог та описати, як саме відбувається взаємодія користувачів із веб-сайтом. Для візуалізації операцій, які користувач повинен мати можливість виконувати у системі, було створено діаграму варіантів використання.

Наступним кроком було здійснено аналіз технічних аспектів проекту. Обґрунтовано вибір технологій та інструментів, використаних при розробці веб-каталогу. Верхньорівневе проектування архітектури для веб-каталогу передбачає розподіл функціональності на клієнтську та серверну частини. Для клієнської частини веб-каталогу обрано фреймворк Angular та шаблон проектування MVC; в якості серверної частини обрано інструменти платформи Firebase, серед яких використано наступні: сервіс для автентифікації користувачів, хмарна база даних та хмарне сховище для зберігання завантажених файлів. Основні компоненти архітектури веб-сайту було візуалізовано.

Після проектування архітектури веб-сайту було розроблено його структуру на основі аналізу визначених функціональних вимог. Цей процес включав в себе розробку структури основних розділів й підпунктів сайту та створення масиву навігаційних маршрутів на основі цієї структури. Також було визначено ключові сутності системи (публікації та користувачі) та створено реляційну модель бази даних веб-сайту.

Останнім етапом практичної частини кваліфікаційної роботи було створення програмного забезпечення для генерації ключових слів на основі наданого опису з використанням бібліотеки для обробки природної мови spaCy та даних пошукових запитів Google.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було спроектовано та розроблено веб-каталог науково-освітніх матеріалів, який забезпечує швидкий та зручний доступ до актуальної інформації. Для пошукової оптимізації системи було розроблено програмне забезпечення, яке генерує набори ключових слів за описом публікацій, використовуючи для цього інструменти обробки природної мови. Для досягнення поставленої мети було виконано наступні завдання:

1. Проведено аналіз цільової аудиторії, визначено ключові ролі користувачів та їхні потреби. Результати цього аналізу допомогли визначити основні функції та можливості, які необхідно реалізувати для веб-каталогу.

2. На основі проведеного аналізу потреб цільової аудиторії розроблено список функціональних вимог до системи, що дозволило чітко визначити, як має функціонувати веб-сайт та як користувачі можуть взаємодіяти з ним.

3. Визначено інструменти та технології, необхідні для розробки веб-сайту. Для розробки клієнтської частини було обрано фреймворк Angular, а в якості серверної частини виступили інструменти Firebase для автентифікації користувачів, збереження даних та завантаження матеріалів.

4. Спроектувано верхньорівневу архітектуру каталогу, визначено основні компоненти системи та способи інтеграції між ними.

5. Визначено оптимальну структуру, основні модулі та компоненти веб-сайту. Розроблений каркас дозволив розпочати процес написання веб-сайту.

6. Розроблено реляційну модель даних згідно з визначеними сутностями, на основі якої було спроектовано базу даних.

7. Розроблено програмне забезпечення, яке здійснює генерацію наборів ключових слів за описом публікацій для пошукової оптимізації веб-сайту.

В контексті сучасних вимог інформаційного суспільства, реалізація цього проекту допоможе задовольнити потреби учасників освітнього процесу в швидкому доступі до актуальної та достовірної інформації. Такий веб-каталог дозволяє зручно створювати та редагувати інформаційний контент, а також здійснювати швидкий пошук та отримувати релевантні результати, що забезпечує його актуальність і відповідність потребам користувачів у сферах науки та освіти.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Прокопенко Т. О. Теорія систем і системний аналіз: навч. посіб. [Електронний ресурс] / Т. О. Прокопенко; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси: ЧДТУ, 2019. – 139 с.
2. Firebase як бекенд для будь-яких застосунків, та як використовувати Firebase-сервісию. URL: https://dou.ua/forums/topic/44058/ (дата звернення: .05.2024).
3. Firebase як бекенд для будь-яких застосунків, та як використовувати Firebase-сервісию. URL: https://dou.ua/forums/topic/44058/ (дата звернення: .05.2024).
4. Бойко Ю. П. Опорний конспект лекцій з дисципліни «Моделювання в цифровій економіці» [Електронний ресурс] – Київ: НАУ, 2019. – 45с.
5. Формування ефективних механізмів державного управління та менеджменту в умовах сучасної економіки: теорія і практика: матеріали XI Міжнародної заочної науково-практичної конференції 24 листопада 2023 р. / за ред. В. М. Огаренка, О. В. Мащенко та ін. Запоріжжя : КПУ, 2023. 332 с.
6. Павленко Ю.С. Пошукова оптимізація, технології та сервіси веб-аналітики : конспект лекцій [Електронний ресурс] / Ю.С. Павленко; ВНУ імені Лесі Українки. – Електронні текстові дані (1 файл: 968 КБ). – Луцьк: ВНУ імені Лесі Українки, 2022. – 51 с.
7. Павленко Ю.С. Пошукова оптимізація, технології та сервіси веб-аналітики : конспект лекцій [Електронний ресурс] / Ю.С. Павленко; ВНУ імені Лесі Українки. – Електронні текстові дані (1 файл: 968 КБ). – Луцьк: ВНУ імені Лесі Українки, 2022. – 51 с.
8. Ефективність ключових слів. Як вони впливають на SEO та онлайн-видимість. URL: https://guildofmarketing.com/efektyvnist-kliuchovykh-sliv-yak-vony-vplyvaiut-na-seo-ta-onlain-vydymist/ (дата звернення: 10.05.2024).
9. Шевченко, Ю. О. (2022). Обробка і аналіз даних з використанням електронних таблиць. Частина II «Аналіз даних та макроси». https://ir.nmu.org.ua/handle/123456789/162624

ДОДАТОК А

```
import spacy

import firebase_admin

from firebase_admin import credentials

from firebase_admin import firestore

import requests

from bs4 import BeautifulSoup

from collections import Counter


MIN_POPULARITY_THRESHOLD = 10000

nlp = spacy.load("uk_core_news_sm")


cred = credentials.Certificate('credentials.json')

app = firebase_admin.initialize_app(cred)

db = firestore.client()

publications_ref = db.collection(u'publications').stream()


descriptions_array = []

keywords_array = []

publications = []
```

```
def select_best_keywords(nlp_keywords, publication_keywords):
    selected_keywords = []
    combined_keywords = set(nlp_keywords) | set(publication_keywords)
    for keyword in combined_keywords:
        popularity_score = get_google_search_results_count(keyword)
        if (keyword not in selected_keywords and
            all(token.lemma_ not in selected_keywords for token in nlp(keyword)) and
            all(token.pos_ != 'VERB' for token in nlp(keyword))):
            if popularity_score > MIN_POPULARITY_THRESHOLD:
                selected_keywords.append(keyword)
        if len(selected_keywords) == 3:
            break
    return selected_keywords



def process_keywords(keywords):
    processed_keywords = []
    for word in keywords:
        for token in nlp(word):
```

```
            processed_keywords.append(token.lemma_)
    return processed_keywords



def extract_keywords_nlp(text):
    doc = nlp(text)
    keywords = [token.lemma_ for token in doc if token.pos_ in ['NOUN'] and not token.is_stop and not token.is_punct]
    keyword_freq = Counter(keywords)
    top_keywords = keyword_freq.most_common(3)
    return [keyword for keyword, _ in top_keywords]



def get_google_search_results_count(query):
    url = f"https://www.google.com/search?q={query.replace(' ', '+')}"
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36"}
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, "html.parser")
    if soup.select_one('#result-stats nobr') is not None:
        result_stats_text = soup.select_one('#result-stats nobr').previous_sibling
```

```
        results_num = ''.join([num for num in result_stats_text if num.isdigit()])
        return int(results_num)
    else:
        return 0



def extract_keywords(text):
    doc = nlp(text)
    noun_phrases = [token.lemma_.lower() for token in doc if token.pos_ in ['NOUN', 'PROPN']]
    noun_phrase_counts = Counter(noun_phrases)
    keyword_phrases = []
    for phrase, count in noun_phrase_counts.most_common():
        if count >= 2:
            search_count = get_google_search_results_count(phrase)
            if search_count > 0:
                keyword_phrases.append((phrase, search_count))
    keyword_phrases.sort(key=lambda x: x[1], reverse=True)
    return [phrase for phrase, _ in keyword_phrases[:3]]
```

```
for doc in list(publications_ref):
    publications.append(doc.to_dict())
    publication_id = doc.id
    publication_data = doc.to_dict()
    title = publication_data['title']
    description = publication_data['description']
    keywords = publication_data['keywords']
    nlp_keywords = extract_keywords_nlp(title+' '+description)
    best_keywords = select_best_keywords(nlp_keywords, keywords)
    print(f"Публікація: {title}")
    print(f"Популярні при пошуку ключові слова: {keywords}")
    print(f"Визначені NLP ключові слова: {nlp_keywords}")
    print(f"Найкращі ключові слова для SEO: {best_keywords}\n")
    db.collection('publications').document(publication_id).update({'keywords':
best_keywords})
    descriptions_array.append(description)
    keywords_array.append(keywords)
```