

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
“ ” _____ 2024 р.

Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»
спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для агрегації та автоматичного перекладу новин

Виконав студент IV курсу, групи ІТ-01
(шифр групи)
Чабан Антон Євгенович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Сирота О. П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н. Крилов Є. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Чабану Антону Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для агрегації та автоматичного перекладу новин

керівник проєкту Сирота Олена Петрівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» травня 2024 р. № 2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Передпроєктне обстеження предметної області: аналіз предметної області, аналіз існуючих рішень, опис бізнес-процесів постановка задачі.

2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, розроблення функціональних та нефункціональних вимог.

3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, обґрунтування вибору засобів розробки, конструювання програмного забезпечення, аналіз безпеки даних.

4) Аналіз якості та тестування: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.

5) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема бази даних _____

3) Креслення вигляду екранних форм _____

4) Діаграма контейнерів _____

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «11» березня 2024 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв'язання задачі	18.03.2024	
3	Постановка та формалізація задачі	25.03.2024	
4	Розробка інформаційного забезпечення	07.04.2024	
5	Алгоритмізація задачі	15.04.2024	
6	Обґрунтування вибору використаних технічних засобів	22.04.2024	
7	Розробка програмного забезпечення	06.05.2024	
8	Налагодження програми	13.05.2024	
9	Виконання графічних документів	20.05.2024	
10	Оформлення пояснювальної записки	02.06.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент

(підпис)

Антон ЧАБАН

(ініціали, прізвище)

Керівник

(підпис)

Олена СИРОТА

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п'яти розділів, містить 53 таблиці, 25 рисунків та 40 джерел – загалом 77 сторінок.

Дипломний проєкт присвячений вебзастосунку для агрегації та автоматичного перекладу новин.

Мета розробки це забезпечення зручного перегляду та написання статей для користувачів що володіють різними мовами, з функціоналом для пошуку за категоріями, датою, ключовими словами та джерелами, а також автоматичною агрегацією та перекладом новин зі сторонніх джерел.

Об'єкт дослідження: сфера медіа та новин, а також їх редактори та користувачі.

Предмет дослідження: розробка архітектури та програмної реалізації вебзастосунку для агрегації та автоматичного перекладу новин, що складається з аналізу архітектурних рішень та підходів і їх застосуванні.

У першому розділі було здійснено аналіз предметної області та існуючих рішень, описано бізнес процеси та сформульовано задачу.

Другий розділ присвячений аналізу вимог до програмного забезпечення, формулюванню функціональних та нефункціональних вимог до системи.

В третьому розділі розглянуто підходи до конструювання програмного забезпечення, проаналізовано різні підходи, та описано деталі вибору архітектури та реалізації системи.

Четвертий розділ присвячений аналізу якості розробленого програмного продукту та здійсненню його тестування.

В останньому, п'ятому розділі, здійснюється опис процесу розгортання та супроводу програмного забезпечення.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, НОВИНИ, ПЕРЕКЛАД, АГРЕГАЦІЯ, КАТЕГОРІЇ, МІКРОСЕРВІСИ, ПОШУК, API, БАЗА ДАНИХ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 53 tables, 25 figures and 40 sources – in total 77 pages.

The purpose of the diploma project is a Web-application for news aggregation and automatic translation.

The goal of the development is to provide convenient viewing and writing of articles for users who speak different languages, with functionality for searching by category, date, keywords, and sources, as well as automatic aggregation and translation of news from third-party sources.

Research object: the media and news industry, as well as its editors and users.

Research subject: development of the architecture and software implementation of a web application for news aggregation and automatic translation, which includes the analysis of architectural solutions and approaches and their application.

The first chapter analyzes the subject area and existing solutions, describes business processes, and formulates the task.

The second chapter is dedicated to analyzing software requirements, formulating functional and non-functional requirements for the system.

Part three discusses approaches to software design, analyzes different approaches, and describes the details of choosing an architecture and implementing the system.

The fourth chapter is devoted to analyzing the quality of the developed software product and testing it.

The last, fifth chapter describes the process of software deployment and maintenance.

KEYWORDS: WEB APPLICATION, NEWS, TRANSLATION, AGGREGATION, CATEGORIES, MICROSERVICES, SEARCH, API, DATABASE.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2024 р.

Вебзастосунок для агрегації та автоматичного перекладу новин

Технічне завдання

КПІ.IT-0325.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена СИРОТА

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Антон ЧАБАН

Київ – 2024

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	5
4.1	Вимоги до функціональних характеристик	6
4.1.1	Функції користувачького інтерфейсу.....	6
4.1.2	Для користувача:.....	8
4.1.3	Для адміністратора системи (якщо він передбачений):	9
4.1.4	Додаткові вимоги:.....	9
4.2	Вимоги до надійності	9
4.3	Умови експлуатації.....	9
4.3.1	Вид обслуговування	10
4.3.2	Обслуговуючий персонал	10
4.4	Вимоги до складу і параметрів технічних засобів	10
4.5	Вимоги до інформаційної та програмної сумісності	10
4.5.1	Вимоги до вхідних даних.....	10
4.5.2	Вимоги до вихідних даних	11
4.5.3	Вимоги до мови розробки.....	11
4.5.4	Вимоги до середовища розробки	11
4.5.5	Вимоги до представленню вихідних кодів	11
4.6	Вимоги до маркування та пакування.....	11
4.7	Вимоги до транспортування та зберігання	11
4.8	Спеціальні вимоги	11
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	12
5.1	Попередній склад програмної документації	12
5.2	Спеціальні вимоги до програмної документації	12
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	13
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	14

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: вебзастосунок для агрегації та автоматичного перекладу новин.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення агрегації та автоматичного перекладу новин, котре використовується для забезпечення поділу статей за категоріями та іншими параметрами, а також здійснення їх автоматичного перекладу та призначена для користувачів та редакторів, які є носіями різних мов.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунку для агрегації та автоматичного перекладу новин є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання можливості автоматизації та полегшенню роботи з великою кількістю новин в мережі, що представлені різними мовами.

Метою розробки є забезпечення зручного перегляду та написання статей для користувачів що володіють різними мовами, з функціоналом для пошуку за категоріями, датою, ключовими словами та джерелами, а також автоматичною агрегацією та перекладом новин зі сторонніх джерел.

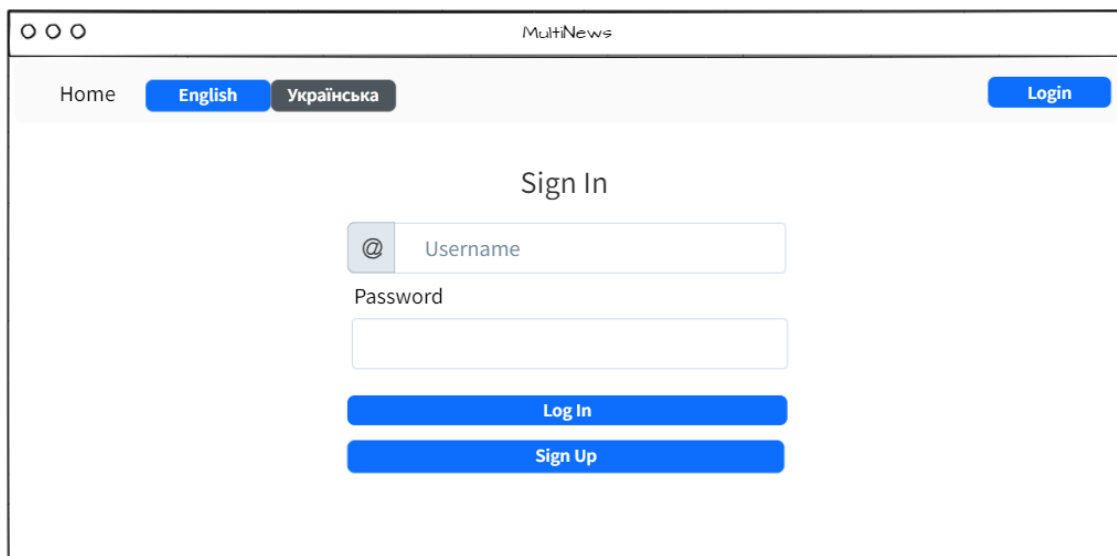
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

- сторінка авторизації редакторів та адміністрації сайту (рисунок 4.1);



The screenshot shows a web browser window with the title 'MultiNews'. The browser's address bar is empty. The website's header includes a 'Home' link, two language selection buttons ('English' and 'Українська'), and a 'Login' button. The main content area is titled 'Sign In' and contains two input fields: 'Username' (with an '@' icon) and 'Password'. Below these fields are two blue buttons: 'Log In' and 'Sign Up'.

Рисунок 4.1 – Сторінка авторизації

- сторінка створення статті, де редактори можуть писати власний матеріал (рисунок 4.2);

MultiNews

Home English Українська Welcome, user Logout

Create Article

Article Link

Categories

Sources

Article Title

Article Description

Create

Рисунок 4.2 – Сторінка створення статті

– сторінка адміністратора де він може переглянути список редакторів (рисунок 4.3);

MultiNews

Home English Українська Welcome, user Logout

Admin Panel

Username	Role	Delete	Edit	More Information
Jeff Bezos	Editor	Delete	Role Edit	View More
Clarence Ortiz	Editor	Delete	Role Edit	View More
Edward Manning	Editor	Delete	Role Edit	View More
Jeff Hall	Admin	Delete	Role Edit	View More
Jeff Hall Jr.	Editor	Delete	Role Edit	View More

Рисунок 4.3 – Панель адміністратора

– головна сторінка з останніми новинами (рисунок 4.4);

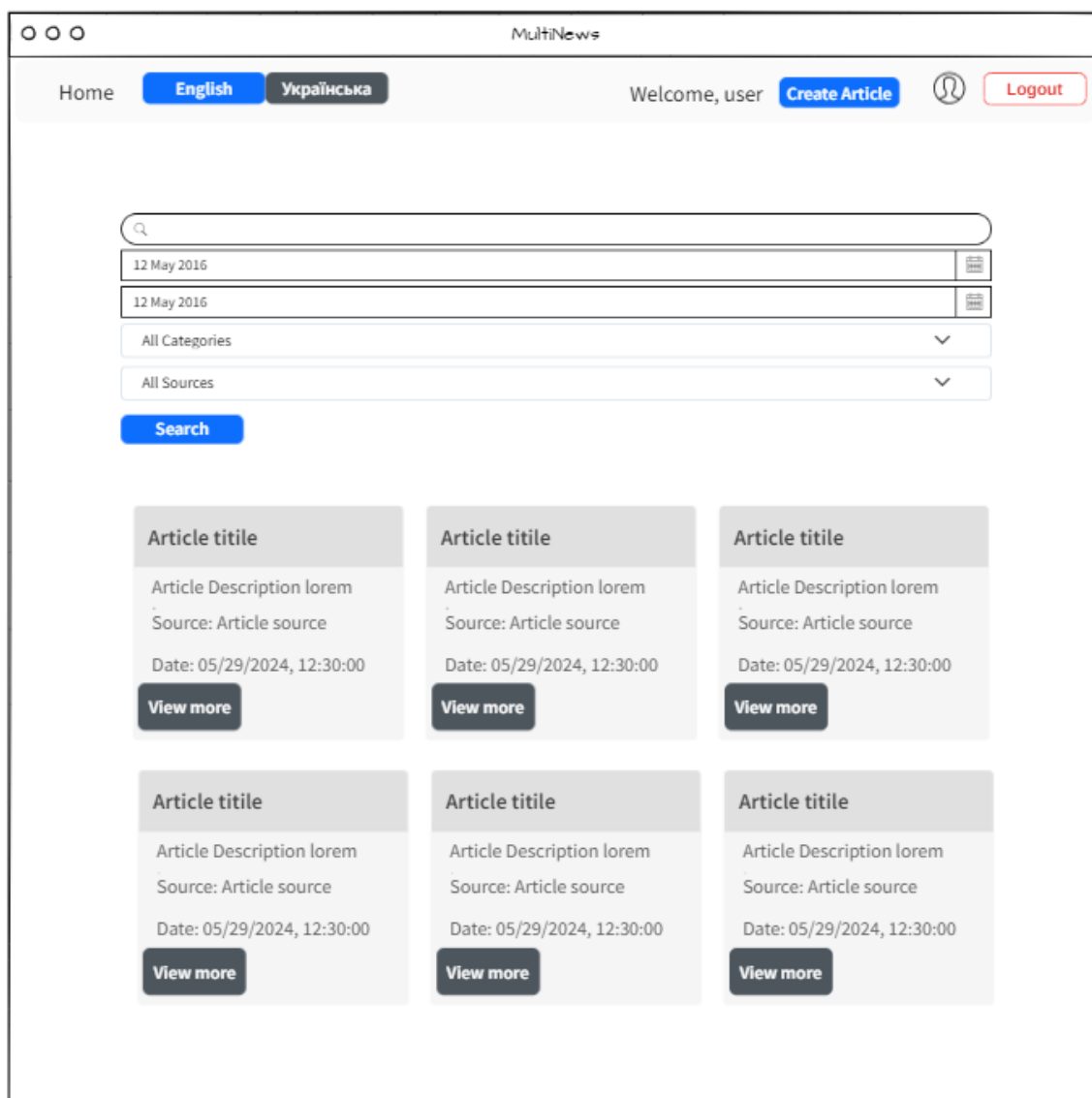


Рисунок 4.4 – Список останніх новин

4.1.2 Для користувача:

- перегляд усіх новин обраною мовою;
- перегляд новин за встановленими параметрами пошуку (ключове слово/ресурс/дата/тематика);
- перегляд детальної інформації обраної новини (детальний опис, ресурси на посилання на оригінальні джерела);
- зміна мови ресурсу;
- перегляд всіх статей обраного користувача;
- авторизація та реєстрація.

4.1.3 Для адміністратора системи:

- редагування/видалення користувачів системи (можливість зміни ролей);
- редагування/видалення статей редакторів на відповідній сторінці новини (заголовок, опис, посилання, джерела, категорії);
- створення/редагування/видалення власних статей.

4.1.4 Для редактора сайту:

- створення/редагування/видалення власних статей (заголовок, опис, посилання, джерела, категорії);
- перегляд власних статей та можливість редагування/видалення.

4.1.5 Додаткові вимоги:

- відображати статті за обрану дату;
- відображати статті по обраній тематиці;
- відображати статті по обраному ресурсу;
- обробляти новини зі сторонніх новинних ресурсів (Українська Правда, CNN, FoxNews, УНІАН);
- перекладати оброблені та створені новини на українську/англійську мови;
- оновлювати базу доступних новин кожні 30 хвилин.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Спеціальні умови до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп'ютерах.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i3;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 50 мегабіт;
- браузер з підтримкою JavaScript.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 16 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
- браузер з підтримкою JavaScript.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows або Unix.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: інформація яку отримує сайт з таких ресурсів як: Pravda, CNN, FoxNews та УНІАН повинні бути отримані в форматі RSS. Контент який надають редактори сайту, має надходити в текстовому форматі.

4.5.2 Вимоги до вихідних даних

Вимог до вихідних даних не висувається, оскільки програмний продукт не генерує жодних вихідних файлів.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Java та Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Java Platform з використанням Spring Framework та Python, з використанням FastAPI Framework за допомогою середовища IntelliJ IDEA Ultimate Edition.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді завантаженого готового проекту в репозиторій на сайті GitHub. Код що виконує функціональні задачі має бути представлений у документі Текст Програми.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги до програмного продукту не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема бази даних;
- креслення вигляду екранних форм;
- діаграма контейнерів.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	11.03	
2.	Розробка технічного завдання	25.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	07.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	22.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	06.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	13.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	20.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	02.06	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

**Пояснювальна записка
до дипломного проєкту**

на тему: **Вебзастосунок для агрегації та автоматичного перекладу новин**

КПІ.IT-0325.045440.02.81

ЗМІСТ

ВСТУП	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	6
1.1 Аналіз предметної області	6
1.2 Аналіз існуючих рішень.....	6
1.2.1 Аналіз відомих програмних продуктів.....	7
1.3 Опис бізнес-процесів	8
1.4 Постановка задачі	10
Висновки до розділу	11
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	12
2.1 Варіанти використання програмного забезпечення.....	12
2.2 Розроблення функціональних вимог	23
2.3 Розроблення нефункціональних вимог	28
Висновки до розділу	29
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	30
3.1 Архітектура програмного забезпечення.....	30
3.2 Обґрунтування вибору засобів розробки	34
3.3 Конструювання програмного забезпечення.....	38
3.4 Аналіз безпеки даних	42
Висновки до розділу	44
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	45
4.1 Аналіз якості ПЗ.....	45
4.2 Опис процесів тестування.....	47
4.3 Опис контрольного прикладу	61
Висновки до розділу	63
5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	65
5.1 Розгортання програмного забезпечення.....	65
5.2 Супровід програмного забезпечення	70

Висновки до розділу	70
ВИСНОВКИ.....	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	73
ДОДАТКИ.....	77

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс
SDK	– Software development kit
IT	– Інформаційні технології
ER	– Entity-Relation diagram
OC	– Операційна система.
БД	– База даних
K8s	– Kubernetes
HTTP	– Hypertext Transfer Protocol
MQ	– Message Query

ВСТУП

У нинішніх реаліях, завдяки доступності Інтернету, кількість доступних новин є дуже великою. Все це доступно різними мовами, на різних ресурсах та в абсолютно різних категоріях. Через це слідкувати за всім потоком інформації стає складно та не зручно, що може призвести до втрати актуальності новин та розуміння загальної ситуації.

Ще однією проблемою, з якою досить часто зіштовхуються люди при перегляді новин – це різні мови новинних ресурсів. Це питання частково вирішується за використання перекладачів, але це все ще не є ідеальним варіантом:

- для перекладу новин через перекладач – користувачу необхідно використовувати сторонні застосунки, що не є ефективною тратою часу і не є зручним підходом;
- обсяг тексту для перекладу може бути обмеженим для безкоштовного використання.

Дивлячись на статистику найпопулярніших ЗМІ в Україні[1] – можна побачити значно більшою популярністю користуються україномовні ресурси, де все подано рідною мовою. Так само за статистикою в США[2] більшою популярністю користуються англomовні видання. Отже, можна побачити що більш популярними є новини рідною мовою, що може призвести до втрати частки інформації та її актуальності.

Отже, для розв’язання цих проблем було вирішено розробити застосунок який дозволить поєднати в собі новини з різноманітних джерел та надати функціонал для перегляду обраною мовою, а також надати можливість для написання власного контенту.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Наразі в умовах війни, пандемії та інших обставин, онлайн-системи для моніторингу новин є як ніколи актуальними, оскільки дозволяються не «випасти» з інформаційного простору та бути в курсі світової ситуації. Такі веб-застосунки використовуються по всьому світу, адже всі люди стараються сліdkувати за ситуацією в своїй країні та світі.

На просторах Інтернету зараз існує багато онлайн сервісів для перегляду новин, що дозволяють переглядати та створювати власні статті для редакторів. Зазвичай, сайти з новинами мають вигляд стрічки з новинами з вибором категорій (наприклад УНІАН[3], CNN[4]). В подібних системах редактори пишуть власні статті та задають відповідні категорії.

Однією з проблем подібних застосунків є те, що статті належать одному ресурсу та відповідають певній позиції, яка може бути суб'єктивною. Також за потреби в подібних застосунках не можна обрати мову статті та ресурсу, або знайти потрібну новину за ключовими словами.

Також існують агрегатори новин – це онлайн сервіси, які збирають новини з різних джерел та демонструють їх користувачам на одній платформі.

Всі ці системи мають спільну цільову аудиторію людей, які зацікавлені в перегляді новин. І зручним рішенням для охоплення аудиторії послугувало б об'єднання агрегатору новин та звичайних порталів в один спільний сервіс.

1.2 Аналіз існуючих рішень

Проаналізуємо відомі на сьогодні технічні рішення, що допоможуть у реалізації веб-застосунку для агрегації та автоматичного перекладу новин.

Далі будуть розглянуті відомі програмні продукти, та буде знійснено порівняльну характеристику між аналогами.

1.2.1 Аналіз відомих програмних продуктів

Вукву[5] – це українське інформаційне агентство, яке було засновано в 2014 році. Сайт публікує новини, статті та аналітику з України щодо останніх актуальних новин. Букви один із небагатьох українських новинних сайтів, який не показує банерної реклами. Це дозволяє читачам зосередитися на контенті та уникнути відволікаючих факторів. Вукву є одним із найпопулярніших новинних сайтів в Україні. У 2023 році сайт відвідало понад 10 мільйонів користувачів. Вукву має значний вплив на громадську думку в Україні. Сайт часто цитується в інших ЗМІ та використовується як джерело інформації для політичних лідерів та громадських діячів.

Google News[6] – це агрегатор новин, який збирає новини з різних джерел по всьому світу. Він був запущений у 2002 році і є одним із найпопулярніших джерел новин у світі. Google News збирає новини з понад 25 000 видань, включаючи газети, журнали, веб-сайти та телевізійні станції. Він використовує алгоритми машинного навчання для сортування новин за релевантністю та актуальністю.

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	MultiNews	Вукву	Google News	Пояснення
Підтримка кількох мов	+	+-	+-	Google News підлаштовує мову відповідно до регіону, тобто новини будуть відрізнятись. А Вукву здійснюють не миттєвий переклад новин.
Новини з багатьох ресурсів	+	-	+	Вукву самі пишуть власні статті.
Підтримка написання власних статей	+	+	-	Google News є виключно агрегатором новин.

1.3 Опис бізнес-процесів

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 1.1 та 1.2).

Опис послідовності реєстрації користувача:

- а) користувач переходить на сторінку реєстрації користувача;
- б) користувач заповнює поля необхідні для реєстрації. Відбувається перевірка введеної інформації на стороні користувача. Можливі 2 варіанти:
 - 1) введена інформація некоректна. В такому випадку, під некоректним полем виводиться повідомлення з детальною інформацією про помилку;

2) введена інформація коректна. В такому випадку, вся інформація відправляється на сервер на опрацювання;

в) відбувається перевірка чи вільне ім'я користувача. Можливі 2 варіанти:

1) ім'я зайняте. В такому випадку видається помилка про те, що ім'я недоступне;

2) ім'я вільне. В такому випадку користувач зберігається з таким іменем.

г) користувач реєструється та зберігається в системі;

д) відбувається перенаправлення на сторінку входу в систему.

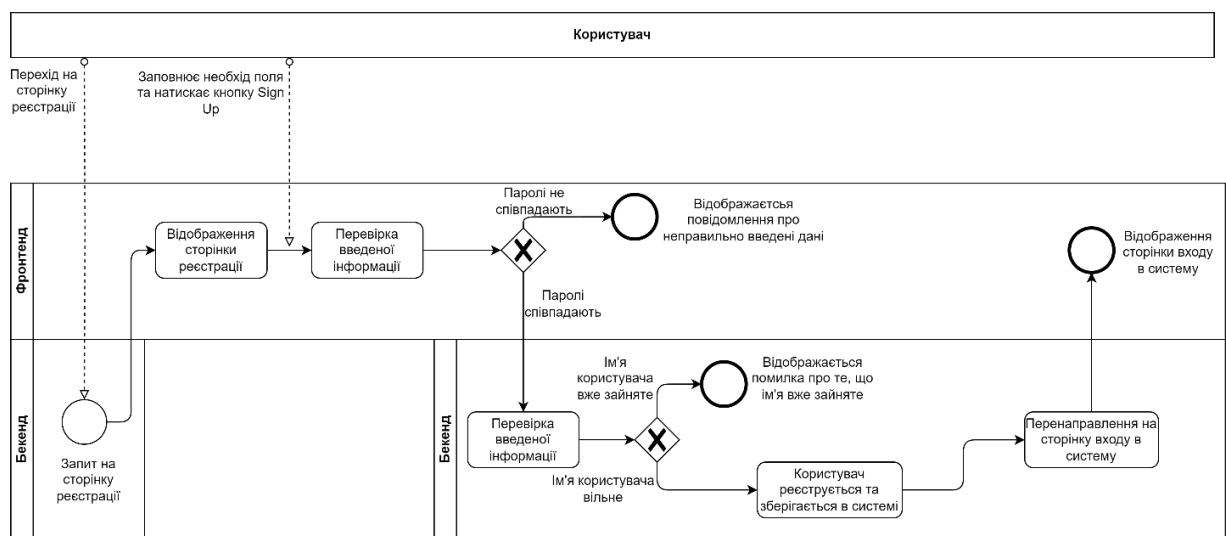


Рисунок 1.1 – BPMN модель реєстрації користувача

Опис послідовності створення статті:

а) користувач переходить на сторінку створення статті;

б) користувач заповнює поля необхідні для створення статті.

Відбувається перевірка введеної інформації на стороні користувача. Можливі 2 варіанти:

1) введена інформація некоректна. В такому випадку, під некоректним полем виводиться повідомлення з детальною інформацією про помилку;

2) введена інформація коректна. В такому випадку, вся

інформація відправляється на сервер на опрацювання;

в) відбувається перевірка чи встановлене посилання на джерело новини. Можливі 2 варіанти:

1) посилання не надано. В такому випадку встановлюється посилання за замовчуванням на дану статтю;

2) посилання надано. В такому випадку встановлюється необхідне посилання на оригінал статті.

г) стаття генерується та зберігається в системі;

д) відбувається перенаправлення на сторінку з усіма статтями.

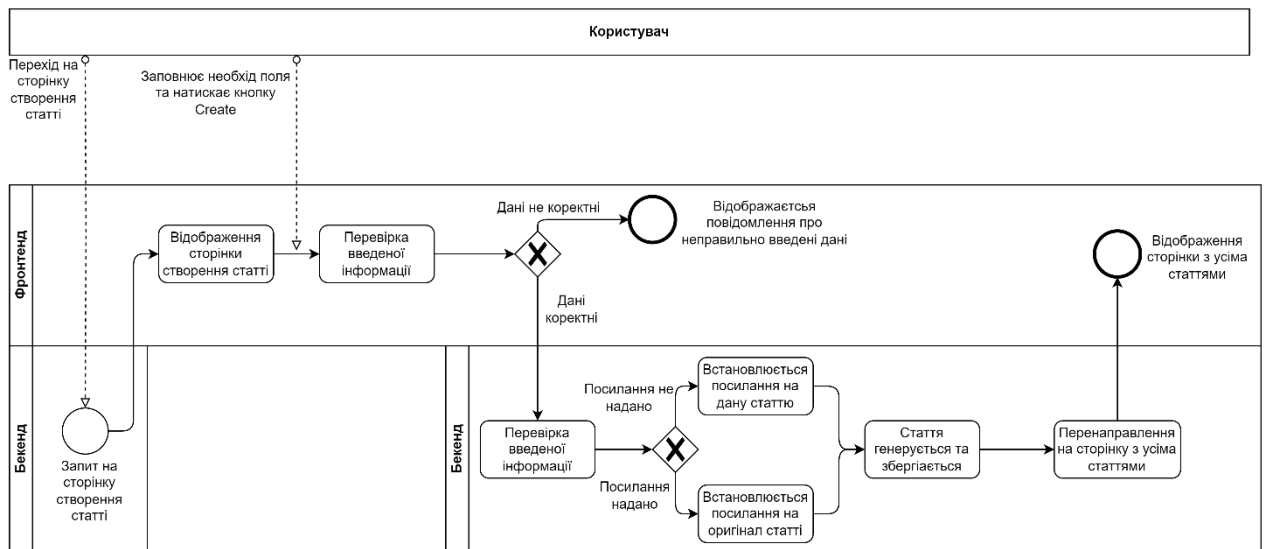


Рисунок 1.2 – BPMN модель створення статті

1.4 Постановка задачі

Ця розробка має на меті спростити доступ до новин незалежно від мови та поєднати в собі агрегацію новин та можливості звичайних новинних порталів, де редактори створюють контент, а також підвищити зручність роботи з новинами, реалізувати зручний пошук за категоріями, ключовими словами, датою. Проект передбачає створення інтуїтивно зрозумілого та ефективного інтерфейсу, що враховує потреби користувачів у зручному пошуку та фільтрації інформації. Цієї мети можна досягнути за допомогою

реалізації функціональних вимог. Таким чином, ми дамо кінцевому користувачу вже звичні інструменти для використання таких систем, а також забезпечимо їх новими можливостями, що відсутні в подібних системах. А дотримання нефункціональних вимог забезпечить довге і комфортне використання системи.

Висновки до розділу

В цьому розділі було здійснено передпроектне обстеження предметної області.

В першому пункті розділу «Аналіз предметної області» - була розглянута актуальність таких систем. Також була описана їх реалізація в межах Інтернету, та сформована проблематика подібних рішень.

В другому розділі – «Аналіз існуючих рішень» – були описані відомі, на даний момент, програмні рішення та здійснено порівняльну характеристику з проектом «MultiNews».

В третьому розділі – «Опис бізнес-процесів» - було описано ключові бізнес процеси програмного забезпечення, що буде розроблятися. Також були розглянуті найголовніші бізнес-процеси нашого застосунку – реєстрація користувача та створення статті. Вони були детально прописані та проілюстровані за допомогою BPMN-моделей.

В четвертому розділі – «Постановка задачі» – було описано узагальнену мету та цілі з якими виконується розробка даного програмного забезпечення.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є перегляд, створення та редагування новин, більше функцій можна побачити на графічному матеріалі, креслення 1.

Також програмне забезпечення автоматично опрацьовує RSS-feed доступних новинних ресурсів, здійснює їх переклад та визначає категорію. Article Parser є компонентом системи, що відповідає за обробку статей зі сторонніх джерел. Детальна діаграма представлена в графічному матеріалі, креслення 1.

В таблицях 2.1 - 2.14 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (неzareєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-

Продовження таблиці 2.1

Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім'я користувача (username) та пароль в системі і його підтвердження. Після заповнення даних, користувач натискає кнопку реєстрації. Після цього користувач перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних, буде показано сторінку з повідомленням про помилку з описом, що пішло не так.
Post-Condition	Створення користувача, перехід на сторінку входу.

Таблиця 2.2 - Варіант використання UC-2

Use case name	Авторизація користувача
Use case ID	UC-02
Goals	Авторизація користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає авторизуватися в системі
Pre-conditions	Користувач зареєстрований в системі
Flow of Events	Користувач переходить на сторінку авторизації. В поля для авторизації вводяться відповідна інформація: ім'я користувача (username) та пароль. Після заповнення даних, користувач натискає кнопку авторизації. Після цього користувач перенаправляється на головну сторінку.
Extension	В випадку введення не коректних даних, буде показано сторінку з повідомленням про помилку з описом, що пішло не так.

Продовження таблиці 2.2

Post-Condition	Авторизація користувача, перехід на головну сторінку.
----------------	---

Таблиця 2.3 - Варіант використання UC-3

Use case name	Вихід користувача
Use case ID	UC-03
Goals	Вихід користувача з системи
Actors	Користувач, адміністратор
Trigger	Користувач бажає вийти з системи
Pre-conditions	Користувач авторизований
Flow of Events	Користувач натискає кнопку виходу з системи. Після цього користувач перенаправляється на головну сторінку
Extension	-
Post-Condition	Користувач вийшов з системи, перехід на головну сторінку

Таблиця 2.4 - Варіант використання UC-4

Use case name	Перегляд власних статей
Use case ID	UC-04
Goals	Перегляд статей створених певним користувачем
Actors	Користувач, адміністратор
Trigger	Користувач хоче побачити статті, які він створив
Pre-conditions	Користувач авторизований
Flow of Events	Користувач відкриває власний профіль та бачить перелік своїх статей.
Extension	Якщо власних статей не знайдено – відображається пустий профіль користувача.

Продовження таблиці 2.4

Post-Condition	Перехід на профіль користувача.
----------------	---------------------------------

Таблиця 2.5 - Варіант використання UC-5

Use case name	Створення статті
Use case ID	UC-05
Goals	Створення власної статті
Actors	Користувач, адміністратор
Trigger	Користувач хоче створити власну статтю
Pre-conditions	Користувач авторизований та знаходиться на сторінці з власними темами
Flow of Events	Користувач натискає на кнопку «Add New Article». Користувачу відкриваються поля, які необхідно заповнити. Після заповнення даних, користувач натискає кнопку створення. Після цього користувач перенаправляється на сторінку з усіма статтями.
Extension	-
Post-Condition	Створення статті, перехід на сторінку з усіма статтями

Таблиця 2.6 - Варіант використання UC-6

Use case name	Редагування статті
Use case ID	UC-06
Goals	Редагування статті
Actors	Користувач, адміністратор
Trigger	Користувач хоче відредагувати статтю
Pre-conditions	Користувач авторизований та знаходиться на сторінці з власними статтями, або відкрив статтю і є її власником. Адміністратор може редагувати будь-яку статтю відкривши її та натиснувши відповідну кнопку
Flow of Events	Користувач натискає на «Edit» біля статті. Користувач переходить на сторінку редагування статті. Відповідні поля заповнені поточною інформацією, проте можуть бути відредагованими. Після заповнення даних, користувач натискає кнопку оновлення. Після цього користувач перенаправляється на оновленої статті.
Extension	-
Post-Condition	Зміна статті, перехід на сторінку з оновленою статтею

Таблиця 2.7 - Варіант використання UC-7

Use case name	Видалення статті
Use case ID	UC-07
Goals	Видалення статті
Actors	Користувач, адміністратор
Trigger	Користувач хоче видалити статтю

Продовження таблиці 2.7

Pre-conditions	Користувач авторизований та знаходиться на сторінці з власними статтями, або відкрив статтю і є її власником. Адміністратор може видаляти будь-яку статтю відкривши її та натиснувши відповідну кнопку
Flow of Events	Користувач натискає на «Delete» біля статті. Після цього користувач перенаправляється на сторінку зі статтями.
Extension	-
Post-Condition	Видалення статті, перехід на сторінку зі статтями

Таблиця 2.8 - Варіант використання UC-8

Use case name	Перегляд статей присвоєних користувачу
Use case ID	UC-08
Goals	Перегляд статей
Actors	Гість, користувач, адміністратор
Trigger	Користувач/гість хоче побачити статті, які належать певному користувачу
Pre-conditions	Користувач перейшов на профіль іншого користувача
Flow of Events	Користувач бачить всі статті що належать користувачу
Extension	У випадку, якщо статей присвоєних користувачу не знайдено, виводиться відповідне повідомлення.
Post-Condition	Показ профіля користувача з його статтями

Таблиця 2.9 - Варіант використання UC-9

Use case name	Пошук статті по ключовому слову
Use case ID	UC-09
Goals	Відобразити статті які відповідають ключовим словам
Actors	Гість, користувач, адміністратор
Trigger	Користувач хоче знайти статті за ключовим словом
Pre-conditions	Користувач/гість знаходиться на головній сторінці
Flow of Events	Користувач/гість вводить ключове слово та натискає кнопку пошуку. Після цього користувачу відображаються статті, що відповідають заданому ключовому слову.
Extension	У випадку, якщо статей за заданими словами не знайдено, виводиться відповідне повідомлення.
Post-Condition	Відображення статей що відповідають ключовим словам

Таблиця 2.10 - Варіант використання UC-10

Use case name	Пошук статті по встановленим параметрам
Use case ID	UC-10
Goals	Відобразити статті які відповідають встановленим параметрам
Actors	Гість, користувач, адміністратор
Trigger	Користувач хоче знайти статті за встановленими параметрами
Pre-conditions	Користувач/гість знаходиться на головній сторінці та встановив параметри пошуку.

Продовження таблиці 2.10

Flow of Events	Користувач/гість встановлює необхідні йому параметри, такі як: категорія, джерело, дата, ключове слово та натискає кнопку пошуку. Після цього користувачу відображаються статті, що відповідають заданим параметрам.
Extension	У випадку, якщо статей за встановленими параметрами не знайдено, виводиться відповідне повідомлення.
Post-Condition	Відображення статей що відповідають встановленим параметрам пошуку

Таблиця 2.11 - Варіант використання UC-11

Use case name	Перегляд статті
Use case ID	UC-11
Goals	Переглянути повну інформацію про статтю
Actors	Гість, користувач, адміністратор
Trigger	Користувач бажає переглянути повну інформацію про статтю
Pre-conditions	Користувач обрав певну статтю та натиснув «View More»
Flow of Events	Користувачу відображається сторінка з детальною інформацією про статтю
Extension	У випадку, якщо користувач є адміністратором, або власником статті він побачить кнопки для редагування/видалення статті
Post-Condition	Відображення сторінки з повною інформацією про статтю

Таблиця 2.12 - Варіант використання UC-12

Use case name	Зміна мови
Use case ID	UC-12
Goals	Змінити мову ресурсу
Actors	Гість, користувач, адміністратор
Trigger	Користувач бажає змінити мову ресурсу
Pre-conditions	Користувач натискає на кнопку обраної мови
Flow of Events	Загальна мова інтерфейсу та статей змінюється на обрану
Extension	-
Post-Condition	Інтерфейс змінює мову на обрану

Таблиця 2.13 - Варіант використання UC-13

Use case name	Редагування користувача
Use case ID	UC-13
Goals	Змінити ролі користувача
Actors	Адміністратор
Trigger	Адміністратор бажає змінити ролі користувача
Pre-conditions	Адміністратор натискає потрібну роль для користувача в панелі адміністратора
Flow of Events	Адміністратор натискає на кнопку «Edit».
Extension	-
Post-Condition	Відбувається зміна ролей користувача та перехід назад до панелі адміністратора

Таблиця 2.14 - Варіант використання UC-14

Use case name	Видалення користувача
Use case ID	UC-14
Goals	Видалення користувача
Actors	Адміністратор
Trigger	Адміністратор бажає видалити користувача
Pre-conditions	Адміністратор натискає кнопку Delete в панелі адміністратора
Flow of Events	Адміністратор натискає на кнопку «Delete». Після цього обраний користувач видаляється
Extension	-
Post-Condition	Відбувається видалення користувача та перехід назад до панелі адміністратора

Таблиця 2.15 - Варіант використання UC-15

Use case name	Визначення категорії статті
Use case ID	UC-15
Goals	Визначення категорії статті
Actors	Article Parser
Trigger	Стаття зі стороннього ресурсу прибула на обробку
Pre-conditions	Текст статті доступний англійською мовою
Flow of Events	Заголовок та опис статті передається на обробку для визначення категорії
Extension	У випадку якщо категорію визначити не вдалося – встановлюється категорія UNKNOWN
Post-Condition	Статті присвоюється категорія, відповідно до її вмісту

Таблиця 2.16 - Варіант використання UC-16

Use case name	Визначення мови статті
Use case ID	UC-16
Goals	Визначення мови статті
Actors	Article Parser
Trigger	Стаття зі стороннього ресурсу прибула на обробку
Pre-conditions	Було надано текст статті на обробку
Flow of Events	Заголовок та опис статті передається на обробку перекладача
Extension	-
Post-Condition	Залежно від мови, текст статті записується у відповідне поле бази даних, а інші поля заповнюються перекладеним текстом

Таблиця 2.17 - Варіант використання UC-17

Use case name	Завантаження статей з RSS-feed
Use case ID	UC-17
Goals	Наповнення бази даних новими статтями зі сторонніх джерел
Actors	Article Parser
Trigger	Пройшло 30 хвилин з моменту останнього оновлення
Pre-conditions	-
Flow of Events	Статті надходять на обробку зі встановлених ресурсів
Extension	-
Post-Condition	Статті передаються на подальше визначення мови та категорії

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.18 наведено загальну модель вимог, а в таблиці 2.19 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.3.

Таблиця 2.18 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1.	Реєстрація користувача	REQ-1	Високий	Високий
1.1.	Ввід імені користувача (username)	REQ-2	-	-
1.1.1.	Перевірка на унікальність	REQ-3	-	-
1.2.	Ввід паролю	REQ-4	-	-
1.2.1.	Перевірка на довжину вводу (мін. 3)	REQ-5	-	-
1.3.	Підтвердження паролю	REQ-6	-	-
2.	Авторизація користувача	REQ-7	Високий	Високий
2.1.	Ввід імені користувача	REQ-8	-	-
2.2.	Ввід паролю	REQ-9	-	-
2.2.1.	Перевірка на довжину вводу (мін. 3)	REQ-10	-	-
2.3	Аутентифікація	REQ-11	-	-
3.	Вихід	REQ-12	Низький	Низький
4.	Перегляд власних статей	REQ-13	Середній	Середній
5.	Створення статті	REQ-14	Високий	Високий
5.1.	Ввід заголовку статті	REQ-15	-	-
5.1.1	Перевірка чи заповнено поле заголовку	REQ-16	-	-
5.2.	Ввід посилання на статтю	REQ-17	-	-

Продовження таблиці 2.18

№	Назва	ID вимоги	Пріоритети	Ризики
5.3.	Ввід опису статті	REQ-18	-	-
5.3.1	Перевірка чи заповнено поле опису	REQ-19	-	-
5.4.	Ввід джерела статті	REQ-20	-	-
5.4.1	Перевірка чи заповнено поле джерела	REQ-21	-	-
5.5.	Вибір категорії статті	REQ-22	-	-
6.	Редагування статті	REQ-23	Середній	Високий
6.1.	Ввід заголовку статті	REQ-24	-	-
6.1.1	Перевірка чи заповнено поле заголовку	REQ-25	-	-
6.2.	Ввід посилання на статтю	REQ-26	-	-
6.3.	Ввід опису статті	REQ-27	-	-
6.3.1	Перевірка чи заповнено поле опису	REQ-28	-	-
6.4.	Ввід джерела статті	REQ-29	-	-
6.4.1	Перевірка чи заповнено поле джерела	REQ-30	-	-
6.5.	Вибір категорії статті	REQ-31	-	-
7.	Видалення статті	REQ-32	Середній	Низький
8.	Перегляд статей присвоєних користувачу	REQ-33	-	-
9.	Пошук статті по ключовому слову	REQ-34	Високий	Низький
9.1.	Ввід ключового слова	REQ-35	-	-
10.	Пошук статті по встановленим параметрам	REQ-36	Високий	Низький

Продовження таблиці 2.18

№	Назва	ID вимоги	Пріоритети	Ризики
10.1.	Вибір категорії для пошуку	REQ-37	-	-
10.2.	Вибір джерела для пошуку	REQ-38	-	-
10.3.	Дата для пошуку	REQ-39	-	-
10.4.	Ключове слово для пошуку	REQ-40	-	-
11.	Перегляд статті	REQ-41	Високий	Низький
12.	Зміна мови	REQ-42	Середній	Низький
12.1.	Вибір мови	REQ-43	-	-
13.	Редагування користувача	REQ-44	Середній	Високий
13.1.	Вибір нової ролі користувача	REQ-45	-	-
14.	Видалення користувача	REQ-46	Низький	Середній
15.	Автоматичне завантаження статей з RSS-feed сторонніх ресурсів	REQ-47	Високий	Середній
16.	Автоматичне оновлення списку новин зі сторонніх ресурсів кожні 30 хвилин	REQ-48	Високий	Низький
17.	Автоматичний переклад статті	REQ-49	Високий	Середній
17.1	Автоматичне визначення мови статті	REQ-50	Середній	Середній
18.	Автоматичне визначення категорій статей зі сторонніх ресурсів	REQ-51	Високий	Високий

Таблиця 2.19 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-1	Реєстрація користувача Система повинна надавати можливість реєстрації користувачеві шляхом введення імені користувача (username), паролю, підтвердження паролю.
FR-2	Авторизація користувача Система повинна надавати можливість авторизації користувачеві шляхом введення імені користувача та паролю.
FR-3	Вихід користувача Система повинна надавати можливість виходу із системи користувачеві.
FR-4	Перегляд власних статей Система повинна надавати можливість перегляду статей, які належать користувачу.
FR-5	Створення статті Система повинна надавати можливість створення статті шляхом введення заголовку, опису, джерела, категорії.
FR-6	Редагування статті Система повинна надавати можливість редагувати інформацію про статтю.
FR-7	Видалення статті Система повинна надавати можливість видалення статті.
FR-8	Перегляд статей присвоєних користувачу Система повинна надавати можливість перегляду статей, створених певним користувачем.
FR-9	Пошук статті по ключовому слову Система повинна надавати можливість пошуку статей по ключовим словам з їх тексту.

Продовження таблиці 2.19

ID вимоги	Назва та опис
FR-10	Пошук статті по встановленим параметрам Система повинна надавати можливість пошуку статей по таким параметрам як: категорія, джерело, дата, ключове слово.
FR-11	Перегляд статті Система повинна надавати можливість переглядати статтю в її повному обсязі.
FR-12	Зміна мови Система повинна надавати можливість змінити мову інтерфейсу та вмісту статей на українську, або англійську мови.
FR-13	Редагування користувача Система повинна надавати можливість редагувати ролі користувача.
FR-14	Видалення користувача Система повинна надавати можливість видалити користувача.
FR-15	Автоматичне завантаження статей з RSS-feed сторонніх ресурсів Система повинна автоматично завантажувати статті з встановлених новинних ресурсів (Українська Правда, CNN, Fox News, УНІАН).
FR-16	Автоматичне оновлення списку новин зі сторонніх ресурсів кожні 30 хвилин Система повинна автоматично оновлювати список доступних новин кожні 30 хвилин, використовуючи планувальник.
FR-17	Автоматичний переклад статті Система повинна автоматично здійснювати переклад статті на іншу мову.

Продовження таблиці 2.19

ID вимоги	Назва та опис
FR-18	Автоматичне визначення категорій статей зі сторонніх ресурсів Система повинна автоматично здійснювати визначення категорії статті за наданим текстом.

На рисунку 2.1 представлена матриця трасування функціональних вимог між варіантами використання.

	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07	FR-08	FR-09	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18
UC-01	+																	
UC-02		+																
UC-03			+															
UC-04				+														
UC-05					+													
UC-06						+												
UC-07							+											
UC-08								+										
UC-09									+									
UC-10									+	+								
UC-11											+							
UC-12												+						
UC-13													+					
UC-14														+				
UC-15																		+
UC-16																	+	
UC-17															+	+		

Рисунок 2.1 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

В нашому веб-застосунку повинні бути дотримані такі нефункціональні вимоги:

- безпека (security) – система має бути захищена від нелегального

доступу. Система повинна бути захищена від таких вразливостей як SQL Injection та XSS-атаки;

- доступність (accessibility) - система повинна бути доступна більшу частину часу;
- сумісність (compatibility) - система повинна підтримувати роботу в різних браузерах без порушення функціональних;
- тестованість (testability) - функціональність системи повинна бути реалізована таким чином, щоб її можна було протестувати на відповідність функціональним вимогам;
- зручність у використанні (usability) - система повинна мати недвозначний, інтуїтивно зрозумілий інтерфейс, тобто користувач повинен розуміти, що відбудеться при кожній взаємодії із системою.

Висновки до розділу

В цьому розділі було здійснено формулювання та аналіз вимог до програмного забезпечення.

Були розписані основні варіанти використання застосунку і наведена відповідна Use-case діаграма. Далі, були проаналізовані та розроблені функціональні та нефункціональні вимоги, які повинні бути реалізовані. За допомогою матриці трасування вимог було побудовано і доведено відповідність між варіантами використання та встановленими функціональними вимогами.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Існує два основних архітектурних підходи для побудови застосунків – монолітний та мікросервісний. Монолітна архітектура – являє собою звичний традиційний підхід до розробки програмного забезпечення, при якому застосунок будується як один єдиний блок [7]. При використанні мікросервісної архітектури здійснюється розбиття застосунку на невеликі, незалежні сервіси які спілкуються між собою через чітко визначені API[8]. Обидва підходи мають як свої переваги, так і недоліки. Монолітний застосунок легше розгортати, так як він має один файл для виконання, а також вимагає менше системних ресурсів, але при цьому він менш масштабований та не такий гнучкий, так як компоненти пов'язані між собою, що ускладнює подальший розвиток. В свою чергу мікросервісний додаток легше підлягає горизонтальному масштабуванню, має вищу надійність, за рахунок розділення компонентів. Детальніше переваги та недоліки обох архітектурних підходів представлено в таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика архітектурних підходів

Характеристика	Мікросервісна архітектура	Монолітна архітектура
Масштабованість	Легко масштабовується горизонтально	Складно масштабується
Гнучкість	Компоненти можна розвивати і впроваджувати окремо	Всі компоненти тісно зв'язані між собою

Продовження таблиці 3.1

Характеристика	Мікросервісна архітектура	Монолітна архітектура
Швидкодія	Швидше завдяки розділенню функціональності на компоненти	Сильно залежить від реалізації та архітектури
Ресурсоемність	Більш ресурсоемний через велику кількість мікросервісів	Менш ресурсоемний
Складність управління	Вимагає більше управління та контролю, важче розгортати	Простіше управління та розгортання

Зважаючи на вищезазначені порівняння, для реалізації застосунку було обрано мікросервісну архітектуру. При використанні цього підходу здійснюється розділення великої системи на маленькі незалежні модулі – мікросервіси, що працюватимуть незалежно один від одного. Кожен такий сервіс відповідатиме за свою встановлену функціональність в системі.

Цей архітектурний підхід було обрано тому що він дає змогу розділити застосунок агрегації та автоматичного перекладу на сервіси, що відповідатимуть за свій логічний розділ, що сприяє масштабованості, за рахунок можливості розвивати кожну частину незалежно від іншої. Також цей підхід дає змогу використовувати різні технології в межах одного проекту, що дає змогу обрати зручний інструментарій для певної задачі (наприклад для створення моделі для класифікації тексту).

Для того щоб якісно побудувати архітектуру застосунку, було використано C4 модель діаграм[9]. Цей підхід дозволяє гарно візуалізувати архітектуру програмного забезпечення на різних рівнях абстракції.

Перший рівень – контекстна діаграма, являє собою стартову точку для побудови архітектури системи, ця діаграма в загальному відображає взаємодію користувачів з системою, та системи з зовнішнім програмним забезпеченням. При використанні цього рівня абстракції – деталі не є важливими, і відображається лише загальна картина архітектури системи, без акценту уваги на технологіях, протоколах на інших низькорівневих деталях.

На рисунку 3.1 представлено контекстну діаграму веб-застосунку агрегації та автоматичного перекладу новин в нотації моделювання С4.

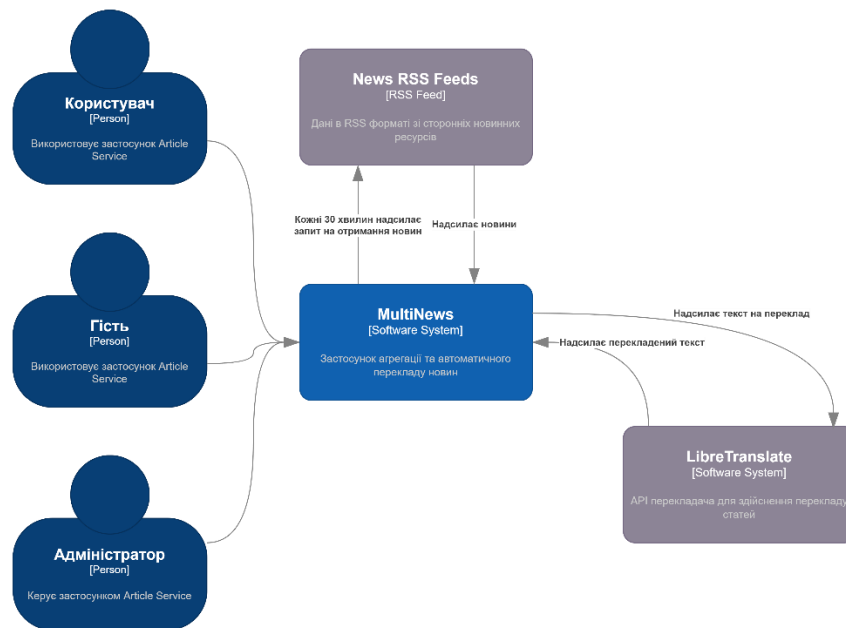


Рисунок 3.1 – Контекстна діаграма застосунку MultiNews

Другим рівнем виступає діаграма контейнерів. На цьому рівні показується внутрішня структура системи та взаємодія між її компонентами. У випадку мікросервісної архітектури, такими контейнерами виступають окремі мікросервіси, бази даних та брокери повідомлень. Ця діаграма відображає розподіл обов’язків в системі, основні технології що будуть використані та те, яким чином контейнери взаємодіють між собою.

Детальну діаграму контейнерів представлено в графічному матеріалі, креслення 4.

Розглянемо елементи діаграми детальніше:

- Frontend – клієнтська частина застосунку, яка відображає користувацький інтерфейс та взаємодіє з серверною частиною застосунку надсилаючи HTTP запити;
- User Service – цей сервіс відповідальний за роботу з користувачами, їх аутентифікацію, реєстрацію та зміни даних;
- Parsing Service – відповідає за обробку статей, їх отримання зі сторонніх ресурсів, а також автоматичне додавання перекладу до створених/відредагованих статей;
- Article Service – відповідає за роботу зі статтями, а саме їх показ, створення, редагування та видалення;
- Category Service – відповідає за визначення категорії переданого тексту на основі вже навченої моделі[10];
- Users Database – база даних користувачів системи;
- Articles Database – база даних доступних статей;
- Message Broker – брокер повідомлень для зв'язку між сервісами;
- LibreTranslate – API перекладача;
- RSS Feeds – контент зі сторонніх ресурсів.

Застосунок MultiNews взаємодіє з таким зовнішнім сервісом як LibreTranslate. Ця взаємодія відбувається шляхом надсилання HTTP запитів, а API перекладача надсилає певну відповідь.

Також для отримання новин зі сторонніх ресурсів використовуються їх RSS-feed. Інформація отримується шляхом надсилання HTTP запиту, після чого приходить відповідь у форматі RSS.

В якості бази даних було обрано реляційні БД[11]. Саме цей тип баз даних добре підходить для розробки застосунку агрегації та автоматичного перекладу новин, адже він забезпечує гарну структурованість даних, та зручний взаємозв'язок між ними.

Для розробки мікросервісної архітектури використовують Docker[12] та Kubernetes[13]. Docker використовується для того щоб запакувати кожен

сервіс до окремого контейнеру в якому той буде виконуватися, а Kubernetes це система, що використовується для оркестрації контейнеризованих додатків.

K8s має такі основні компоненти[14]:

- Pod – мінімальна одиниця в K8s, яка містить контейнер, або групу контейнерів що запускаються разом;
- Deployment – керують життєвим циклом Pod;
- Service – засіб для доступу одного, або декількох Pod, які надають однакову функціональність, Service створює єдину, узгоджену точку входу для клієнтів для доступу до сервісу;
- ConfigMaps і Secrets – використовуються для зберігання даних конфігурації та конфіденційної інформації;
- Persistent Volumes – використовуються для надання сховища для додатків, що працюють в рамках Pod;
- Ingress – ресурс K8s, який використовується для розкриття маршрутів HTTP і HTTPS ззовні кластера до сервісів всередині кластера.

3.2 Обґрунтування вибору засобів розробки

Програмні засоби, що будуть використовуватись під час розробки застосунку – це мова програмування, фреймворки, бібліотеки, які будуть використовуватись при написанні серверної частини; сховище для даних (СУБД); мови, фреймворки, бібліотеки, які будуть використовуватись при розробці клієнтського інтерфейсу; середовища розробки (IDE).

Для реалізації серверної частини можна обрати мову з кількох варіантів. Найпопулярнішими є JS (Node.js), Java, PHP. Всі ці мови є популярними, проте для великого мікросервісного додатку краще підійде Java, адже вона має багату екосистему бібліотек та фреймворків для розробки серверної частини і є продуктивною мовою, а строга типізація та об'єктно орієнтований підхід допомагає розробляти якісне програмне забезпечення. А для розробки сервісу

для класифікацію тексту, варто використати Python, він поєднує в собі простоту та швидкодію, яка чудово задовольнить вимоги цього сервісу.

Після того, як ми обрали Java, треба обрати фреймворк. Тут є 2 варіанти – Java EE [15] та Spring [16]. Наш вибір був зроблений на користь Spring, бо він дозволяє зменшити складність написаного коду (що в свою чергу дозволяє зменшити час розробки), збільшуючи кількість коду, який буде згенерований автоматично, використовуючи анотації.

Для доступу до БД можна використовувати або ORM Framework (який працює на основі JDBC) або чистий JDBC. ORM Framework дозволяє уникнути написання SQL-запитів вручну, дозволяючи створювати запити, керувати таблицями та зв'язками між ними за допомогою класів та методів. Хоча JDBC працює трішки швидше за рахунок того, що SQL-запити для отримання чи збереження даних треба писати вручну і виконуються вони напряму. Але було прийнято рішення використовувати ORM Framework, для економії часу та читабельності коду. А різниця ж у швидкості не є настільки великою, щоб здійснити вибір на користь JDBC. В якості ORM Framework був використаний Hibernate. Він є стандартною реалізацією JPA (стандартизований інтерфейс для Java ORM фреймворків) для Spring-застосунків, а тому містить зручний і пристосований до мови Java інструментарій для взаємодії з БД.

Стандартом забезпечення безпеки в Spring-застосунках є Spring Security – це фреймворк, який зосереджений на забезпеченні як аутентифікації, так і авторизації програм Java [17]. Також він дозволить вирішити деякі можливі вразливості веб-застосунків.

Для розробки клієнтської частини додатку, вибір було зроблено на користь React.js. Ця бібліотека має ряд переваг[18]:

- компонентний підхід, який дозволяє розділяти інтерфейс на логічні компоненти, які можна в подальшому повторно використовувати в додатку;
- висока продуктивність - React використовує віртуальний DOM та механізми оптимізації для ефективного оновлення UI;

– легка інтеграція з іншими технологіями - React.js легко інтегрується з іншими технологіями та фреймворками, що робить його універсальним вибором для веб-розробки.

В якості основної СУБД було обрано PostgreSQL[19]. Вибір стояв між MS SQL Server та PostgreSQL. Так як в нас не складний, в плані структури бази даних, застосунок - вибір було зроблено на користь PostgreSQL так як він є не комерційним проектом з відкритим кодом.

Також для реалізації зв'язку між сервісами проекту необхідний брокер повідомлень є два найпопулярніші рішення RabbitMQ[20] та Apache Kafka[21]. RabbitMQ - це брокер розподілених повідомлень, який збирає поточкові дані з декількох джерел і маршрутизує їх у різні пункти призначення для обробки. Apache Kafka - це платформа потокової трансляції для створення конвеєрів даних і додатків потокової трансляції в реальному часі[22]. Простіше кажучи – Kafka має потужніший інструментарій, який для реалізації простого обміну повідомленнями між сервісами не потрібен, відповідно вибір було зроблено на користь RabbitMQ.

Є багато IDE, які можна використовувати для написання Spring застосунків. По факту, вони відрізняються лиш графічним інтерфейсом, розміром та кількістю підтримуваних сторонніх модулів. Вирішальним фактором для того, щоб обрати IntelliJ IDEA [23] в якості основного середовища написання коду, стала велика кількість сторонніх модулів, плагінів, що забезпечують додаткову функціональність.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.1.

Таблиця 3.2 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Головне середовище розробки програмного забезпечення курсової роботи.
2	DataGrip	Програмне забезпечення, яке надає легкий графічний інтерфейс для доступу до будь-якого серверу бази даних, в тому числі PostgreSQL.
3	WebStorm	Середовище розробки клієнтської частини застосунку.
4	Spring Boot	Фреймворк, що використовується для розробки веб-застосунків.
5	Spring Security	Фреймворк, що дозволяє реалізувати аутентифікацію та авторизацію.
6	Hibernate ORM	Програмне забезпечення, що дозволяє взаємодіяти з БД за допомогою класів Java.
7	LibreTranslate	API перекладача яке використовується для автоматичного перекладу.
8	Docker	Інструментарій для управління ізольованими Linux-контейнерами.

Продовження таблиці 3.2

№ п/п	Назва утиліти	Опис застосування
9	Kubernetes	Система автоматичного розгортання, масштабування та управління застосунками у контейнерах.
10	Minikube	Інструмент для запуску Kubernetes на локальній машині.
11	RabbitMQ	Платформа, що реалізує систему обміну повідомленнями між компонентами програмної системи на основі стандарту AMQP.
12	React.js	Бібліотека JavaScript, яка використовується для розробки інтерфейсу користувача.
13	Postman	Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.
14	SonarCloud	Програмне забезпечення, яке дозволяє проводити статичний аналіз коду

3.3 Конструювання програмного забезпечення

Розглянемо те, як має працювати застосунок агрегації та автоматичного перекладу новин. Існує два способи як статті з'являються в застосунку: зареєстрований користувач створює статтю власноруч, або стаття надходить зі стороннього джерела.

Так як застосунок використовує мікросервісну архітектуру – правильним підходом буде використання архітектурного патерну Database per service [24]. Це дозволяє ізолювати дані для кожного сервісу, та не навантажувати його зайвою бізнес-логікою. Отже у випадку застосунку MultiNews можна виділити 2 бази даних які можна розділити, Users Database та Articles Database. Їх структура відображена в графічному матеріалі, креслення 2.

База даних Users Database складається з двох таблиць, що відповідають за зберігання даних про користувача та його роль у системі. Цю БД використовують User Service, адже їм не потрібно мати інформацію про статті. Опис таблиць бази даних Users Database наведено в таблицях 3.3-3.4:

Таблиця 3.3 – Опис таблиці Users

Таблиця	Назва поля	Тип даних	Опис
Users	id	bigserial	Ідентифікатор користувача
	username	varchar(255)	Ім'я користувача
	password	varchar(2048)	Пароль в зашифрованому вигляді

Таблиця 3.4 – Опис таблиці role

Таблиця	Назва поля	Тип даних	Опис
role	id	bigint	Ідентифікатор користувача
	roles	varchar(255)	Роль користувача

База даних Articles Database складається з 4 таблиць, що відповідають за зберігання даних про статті, категорії, користувачів та їх ролі. Цю базу даних використовує Article Service, так як для його роботи необхідно знати інформацію як про користувачів, так і про статті. Для забезпечення узгодженості між базами даних, зміни над користувачами в Users Database передаються через брокер повідомлень до Article Database. Опис таблиць бази даних Articles Database наведено в таблицях 3.5-3.8:

Таблиця 3.5 – Опис таблиці articles

Таблиця	Назва поля	Тип даних	Опис
articles	id	bigserial	Ідентифікатор статті
	date	timestamp	Дата статті
	description_en	varchar(16384)	Опис статті англійською
	description_ua	varchar(16384)	Опис статті українською
	link	varchar(2048)	Посилання на оригінал статті
	source	varchar(2048)	Джерело статті
	title_en	varchar(2048)	Заголовок статті англійською
	title_ua	varchar(2048)	Заголовок статті українською
	user_id	bigint	Ідентифікатор автора

Таблиця 3.6 – Опис таблиці category

Таблиця	Назва поля	Тип даних	Опис
category	id	bigint	Ідентифікатор статті
	categories	varchar(255)	Назва категорії

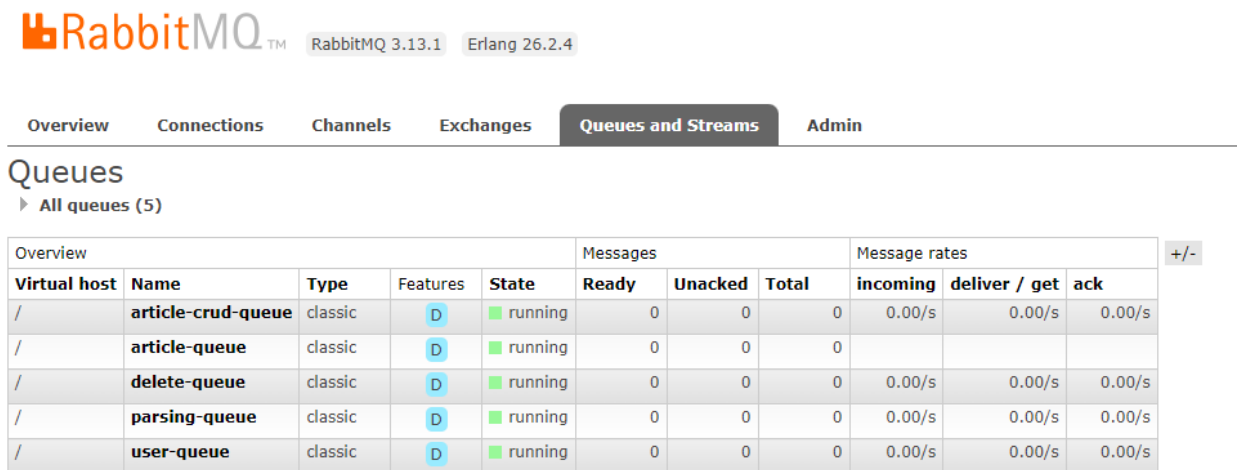
Таблиця 3.7 – Опис таблиці Users

Таблиця	Назва поля	Тип даних	Опис
Users	id	bigserial	Ідентифікатор користувача
	username	varchar(255)	Ім'я користувача
	password	varchar(2048)	Пароль в зашифрованому вигляді

Таблиця 3.8 – Опис таблиці role

Таблиця	Назва поля	Тип даних	Опис
role	id	bigint	Ідентифікатор користувача
	roles	varchar(255)	Роль користувача

Для зв'язку між сервісами використовується RabbitMQ, на рисунку 3.2 показано які черги створено для роботи системи.



Overview					Messages			Message rates			+/-
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/	article-crud-queue	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s	
/	article-queue	classic	D	running	0	0	0				
/	delete-queue	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s	
/	parsing-queue	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s	
/	user-queue	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s	

Рисунок 3.2 – Черги для повідомлень в RabbitMQ

Розглянемо те, для чого використовується кожна черга:

- article-crud-queue – використовується для передачі перекладених статей з Parsing Service;
- article-queue – використовується для передачі статей до Article Service що надійшли зі сторонніх джерел та були оброблені в Parsing Service;
- delete-queue – використовується для надсилання користувачів, яких необхідно видалити з БД іншого сервісу;
- parsing-queue – використовується для надсилання статті створеної/відредагованої користувачем на обробку в Parsing Service для автоматичного перекладу;
- user-queue – використовується для передачі змін над користувачами виконаних в User Service до Article Service, щоб зберегти узгодженість між базами даних.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Нашому веб-застосунку можуть зашкодити 2 типи вразливостей:

- доступ до сторінок, до яких користувач не має відповідних прав;
- безпека даних.

Для уникнення цих проблем в нашому застосунку використовується:

- Spring Security;
- Hibernate ORM;
- React.js.

Проблеми доступу до сторінок, на які користувача немає права заходити повністю вирішуються за допомогою Spring Security та системи ролей. Використовуючи цей фреймворк та присвоюючи користувачу необхідні ролі ми можемо явно оголосити сторінки, на які у користувача з певною роллю є доступ.

Проблеми пов'язані з безпекою даних вирішуються за допомогою Hibernate та React.js. Види подібних вразливостей:

- XSS (Cross site scripting) – вразливість, суть якої досить проста – зломиснику вдається впровадити на сторінку JavaScript-код, який не було передбачено розробниками. Цей код буде виконуватися щоразу, коли користувачі заходитимуть на сторінку програми, куди цей код було додано [25]. Ця вразливість вирішується за допомогою механізму бібліотеки React.js, зокрема її конструкції {}, яка буде екранувати текст. Тобто якщо введена користувачем інформація містить теги JavaScript, вони будуть відображатися як звичайний текст, а не як конструкція JavaScript;

- SQL ін'єкція – це розміщення зловмисного коду в операторах SQL за допомогою поля введення веб-сторінки. Таке відбувається, коли ви просите користувача ввести дані, але замість них користувач дає вам оператор SQL, який ви несвідомо запустите у своїй базі даних [26]. Це вирішується за допомогою Hibernate ORM. При використанні методів, створених за шаблоном, у репозиторії, цей фреймворк аналізує SQL-запит, визначає його структуру, і вставляє можливу SQL-конструкцію в запит екрануючи його, тобто вставляючи як звичайний текст;

- CSRF (Cross-site request forgery) - це атака, яка змушує кінцевого

користувача виконувати небажані дії у програмі, у якій він наразі пройшов аутентифікацію [27]. Реалізацію захисту від цієї вразливості бере на себе фреймворк Spring Security. Щоб уникнути цієї уразливості потрібно при виконанні запиту, який може змінювати стани (POST, PUT, PATCH, DELETE) відправляти токен, якого не може бути в злоумисника, [28].

Висновки до розділу

В цьому розділі було описано архітектурний підхід до розробки програмного забезпечення веб-застосунку агрегації та автоматичного перекладу, побудовано C4 діаграми для візуалізації необхідних архітектурних рішень.

Розглянуто різні варіанти засобів розробки програмного забезпечення, проведено їх порівняльну характеристику та обрано найбільш підходящі рішення.

Спроектовано та сконструйовано програмне забезпечення, та зв'язок між його компонентами, також були описані таблиці в БД, надано визначення їх полям. Зв'язки між таблицями було продемонстровано у вигляді ER-діаграми.

Були визначені можливі вразливості нашого програмного забезпечення. Також були розглянуті інструменти та їхній функціонал за допомогою якого ці вразливості були вирішені.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Показник якості коду визначає те чи написано хороший високоякісний код, чи поганий (низькоякісний).

На перший погляд, оцінка якості є суб'єктивною для кожного, хто його переглядає. Але все ж таки можна виділити основні характеристики високоякісного коду, яких варто дотримуватись реалізуючи застосунок:

- відсутність помилок та багів – є важливим критерієм, так як в ідеалі застосунок має працювати стабільно і без збоїв;
- функціональність – код має в точності виконувати те, для чого був написаний;
- читабельність – код та те, що він має робити – повинно бути зрозуміло при читанні.

Ці та інші характеристики можна перевірити за допомогою статичних та динамічних аналізаторів коду.

Для нашого застосунку використаємо статичний аналізатор SonarCloud. Даний аналізатор дозволяє перевірити код на помилки при написанні. Результат роботи SonarCloud можна побачити на рисунку 4.1:

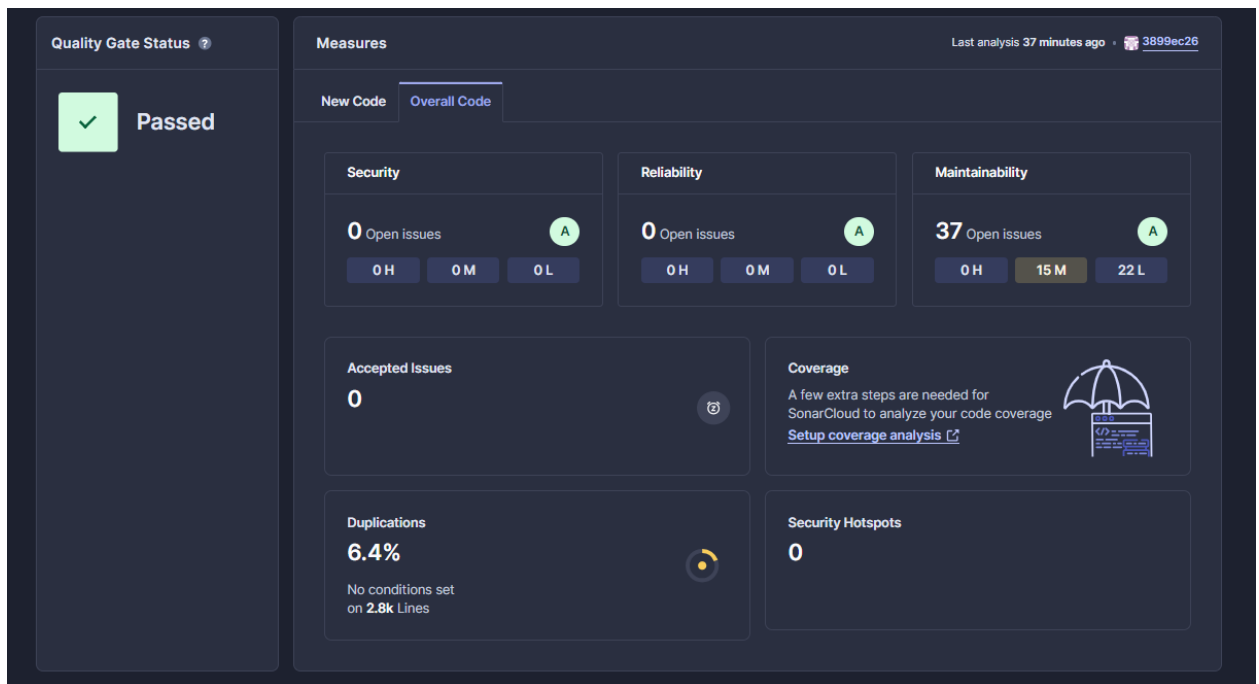


Рисунок 4.1 - Результат аналізу SonarCloud

Переглянувши аналіз, можна побачити що помилок та вразливостей в коді не виявлено.

В коді зустрічаються повтори, через особливість архітектури, і того як зберігаються файли в проекті, відповідно кожен сервіс має схожі файли залежностей а також деякі моделі та конфігураційні файли.

Серед недоліків зазначених аналізатором є незначні недоліки, із серії використання System.out, замість бібліотеки для логування, а також випадки, де можна використовувати if-вирази, замість switch-case. Загалом аналізатор визначив незначні помилки, а всі що були значними – було виправлено.

Другим засобом для аналізу послугує Minikube Dashboard[29], це веб-інтерфейс для керування та моніторингу ресурсів що використовуються K8s кластером. Використовуючи цю утиліту, ми зможемо проаналізувати використання процесору та пам'яті нашими сервісами, це допоможе побачити які частини застосунку використовують найбільше ресурсів. На рисунку 4.2 наведено графік споживання ресурсів додатком.

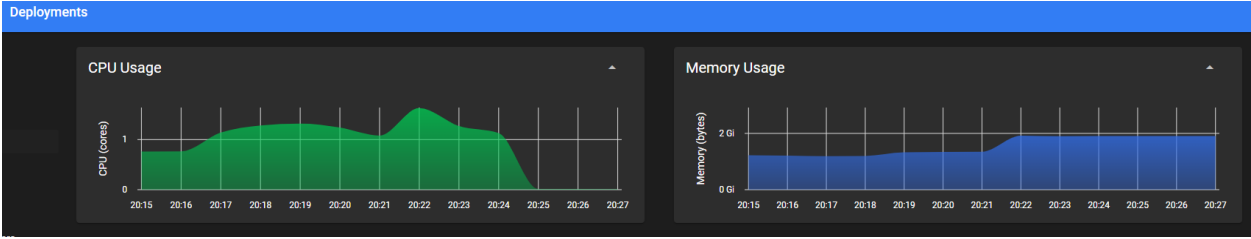


Рисунок 4.2 – Графік споживання ресурсів

Дивлячись на цей графік, можна побачити що найбільше навантаження на процесор відбувається в момент обробки статей зі сторонніх ресурсів, далі споживання процесору падає, а обсяг використаної пам’яті зростає.

4.2 Опис процесів тестування

Для забезпечення високої функціональності коду, було здійснено тестування для різноманітних сценаріїв роботи програми.

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.25.

Таблиця 4.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Ім’я користувача, пароль, підтвердження паролю

Продовження таблиці 4.1

Опис проведення тесту	У відповідні поля вводяться: ім'я користувача, яке до цього не було зареєстроване в системі, пароль від 3 символів (включно), підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка «Sign Up».
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.

Таблиця 4.2 – Тест 2.1

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	2.1
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні данні	Ім'я користувача, пароль
Опис проведення тесту	У відповідні поля вводяться: коректне ім'я користувача, яке до цього було зареєстроване в системі, пароль, що відповідає цьому користувачу. Після цього натискається кнопка «Login».
Очікуваний результат	Авторизація проходить успішно, користувачу надаються відповідні права і відбувається перенаправлення на головну сторінку.
Фактичний результат	Авторизація проходить успішно, користувачу надаються відповідні права і відбувається перенаправлення на головну сторінку.

Таблиця 4.3 – Тест 3.1

Тест	Вихід із системи користувача
Модуль	Вихід із системи користувача
Номер тесту	3.1
Початковий стан системи	Користувач авторизований
Вхідні данні	-
Опис проведення тесту	Натискається кнопка «Logout».
Очікуваний результат	Вихід із системи проходить успішно, користувач позбувається прав і відбувається перенаправлення на головну сторінку.
Фактичний результат	Вихід із системи проходить успішно, користувач позбувається прав і відбувається перенаправлення на головну сторінку.

Таблиця 4.4 – Тест 4.1

Тест	Створення статті
Модуль	Створення статті
Номер тесту	4.1
Початковий стан системи	Користувач знаходиться на сторінці для створення статті
Вхідні данні	Заголовок, опис, джерело, категорія.
Опис проведення тесту	У відповідне поле вводиться заголовок, опис, джерело, категорія. Після цього натискається кнопка «Create».
Очікуваний результат	Стаття успішно додана в систему, користувач перенаправляється на головну сторінку.

Продовження таблиці 4.4

Фактичний результат	Стаття успішно додана в систему, користувач перенаправляється на головну сторінку.
---------------------	--

Таблиця 4.5 – Тест 5.1

Тест	Редагування статті
Модуль	Редагування статті
Номер тесту	5.1
Початковий стан системи	Користувач знаходиться на сторінці для редагування статті, і є її власником, або адміністратором
Вхідні данні	Новий заголовок, опис, джерело, категорія
Опис проведення тесту	У відповідне поле вводиться новий заголовок, опис, джерело, категорія. Після цього натискається кнопка «Edit».
Очікуваний результат	Стаття успішно змінена, користувач перенаправляється на головну сторінку.
Фактичний результат	Стаття успішно змінена, користувач перенаправляється на головну сторінку.

Таблиця 4.6 – Тест 6.1

Тест	Видалення статті
Модуль	Видалення статті
Номер тесту	6.1
Початковий стан системи	Користувач знаходиться на сторінці для статті, і є її власником, або адміністратором
Вхідні данні	-
Опис проведення тесту	Користувач натискає кнопку Delete, після чого відбувається видалення статті

Продовження таблиці 4.6

Очікуваний результат	Стаття успішно видалена з системи, користувач перенаправляється на головну сторінку.
Фактичний результат	Стаття успішно видалена з системи, користувач перенаправляється на головну сторінку.

Таблиця 4.7 – Тест 7.1

Тест	Редагування ролей користувача
Модуль	Редагування користувача
Номер тесту	7.1
Початковий стан системи	Користувач є адміністратором, та знаходиться на сторінці панелі адміністратора
Вхідні данні	Нова роль
Опис проведення тесту	Адміністратор обирає нову роль користувача та натискає кнопку Edit.
Очікуваний результат	Ролі користувача успішно змінені.
Фактичний результат	Ролі користувача успішно змінені.

Таблиця 4.8 – Тест 8.1

Тест	Видалення користувача
Модуль	Видалення користувача
Номер тесту	8.1
Початковий стан системи	Користувач є адміністратором та знаходиться на сторінці панелі адміністратора.
Вхідні данні	-

Продовження таблиці 4.8

Опис проведення тесту	Адміністратор натискає на кнопку «Delete». Після цього обраний користувач видаляється
Очікуваний результат	Відбувається видалення користувача та перехід назад до панелі адміністратора.
Фактичний результат	Відбувається видалення користувача та перехід назад до панелі адміністратора.

Таблиця 4.9 – Тест 9.1

Тест	Автоматичний переклад статті
Модуль	Автоматичний переклад статті
Номер тесту	9.1
Початковий стан системи	Користувач створює статтю та заповнив встановлені поля мовою системи
Вхідні данні	Заголовок, опис.
Опис проведення тесту	Користувач створює статтю та заповнює поля заданою мовою.
Очікуваний результат	Буде створено переклад даної статті на іншу мову, який можна побачити при зміні мови.
Фактичний результат	Буде створено переклад даної статті на іншу мову, який можна побачити при зміні мови.

Таблиця 4.10 – Тест 10.1

Тест	Пошук статті по ключовому слову
Модуль	Пошук статті
Номер тесту	10.1

Продовження таблиці 4.10

Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями
Вхідні данні	Ключове слово
Опис проведення тесту	Користувач заповнює поле пошуку рядком, який можливо знайти серед статей. Та натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають введеному рядку.
Фактичний результат	Користувачу було відображено статті що відповідають введеному рядку.

Таблиця 4.11 – Тест 10.2

Тест	Пошук статті по ключовому слову
Модуль	Пошук статті
Номер тесту	10.2
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями
Вхідні данні	Ключове слово
Опис проведення тесту	Користувач заповнює поле пошуку рядком, який не можливо знайти серед статей. Та натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено повідомлення про те, що нічого не було знайдено за його запитом.
Фактичний результат	Користувачу було відображено повідомлення про те, що нічого не було знайдено за його запитом.

Таблиця 4.12 – Тест 11.1

Тест	Пошук статті по встановленим параметрам
------	---

Продовження таблиці 4.12

Модуль	Пошук статті
Номер тесту	11.1
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, категорія, джерело, дата
Опис проведення тесту	Користувач заповнює поле пошуку рядком, вказує категорію, джерело та дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.13 – Тест 11.2

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.2
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, категорія, джерело
Опис проведення тесту	Користувач заповнює поле ключового слова, категорію, джерело. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.14 – Тест 11.3

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.3
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, категорія, дата
Опис проведення тесту	Користувач заповнює ключове слово, категорію, дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.15 – Тест 11.4

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.4
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, категорія
Опис проведення тесту	Користувач вказує ключове слово, категорію. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.16 – Тест 11.5

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.5
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, джерело, дата.
Опис проведення тесту	Користувач вказує ключове слово, джерело, дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.17 – Тест 11.6

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.6
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, джерело.
Опис проведення тесту	Користувач вказує ключове слово, джерело. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.18 – Тест 11.7

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.7
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Ключове слово, дата
Опис проведення тесту	Користувач вказує ключове слово, дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.19 – Тест 11.8

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.8
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Категорія, джерело, дата.
Опис проведення тесту	Користувач вказує категорію, джерело, дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.20 – Тест 11.9

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.9
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Категорія, джерело.
Опис проведення тесту	Користувач вказує категорію, джерело. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.21 – Тест 11.10

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.10
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Категорія, дата.
Опис проведення тесту	Користувач вказує категорію, дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.22 – Тест 11.11

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.11
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Категорія.
Опис проведення тесту	Користувач вказує категорію. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.23 – Тест 11.12

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.12
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Джерело, дата.
Опис проведення тесту	Користувач вказує джерело, дату. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.24 – Тест 11.13

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.13
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Джерело.
Опис проведення тесту	Користувач вказує джерело. Після чого натискає кнопку Search.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

Таблиця 4.25 – Тест 11.14

Тест	Пошук статті по встановленим параметрам
Модуль	Пошук статті
Номер тесту	11.14
Початковий стан системи	Користувач знаходиться на головній сторінці зі статтями, та встановив параметри пошуку.
Вхідні данні	Дата.
Опис проведення тесту	Користувач вказує дату.
Очікуваний результат	Користувачу буде відображено статті що відповідають встановленим параметрам пошуку.
Фактичний результат	Користувачу було відображено статті що відповідають встановленим параметрам пошуку.

4.3 Опис контрольного прикладу

В якості контрольного прикладу використаємо створення статті та її перегляд обома доступними мовами.

Ми авторизовані в системі та натискає кнопку Create Article (рисунок 4.3).

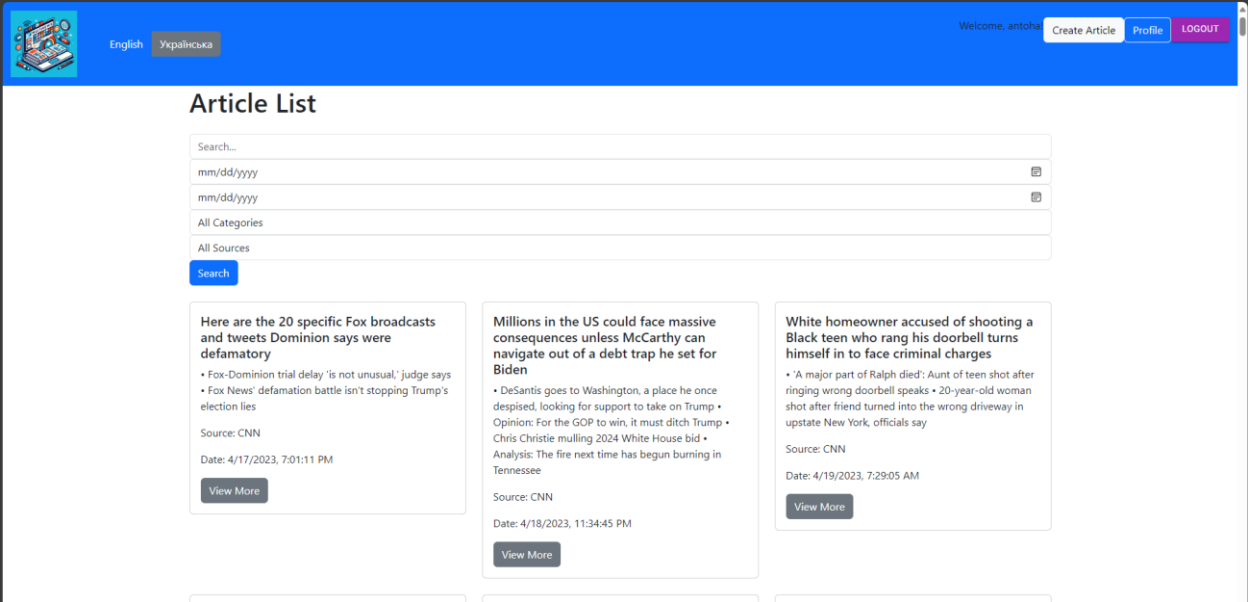


Рисунок 4.3 – Користувач авторизований та може створити власну статтю

Після цього користувачу потрібно заповнити всі необхідні поля (рисунок 4.4).

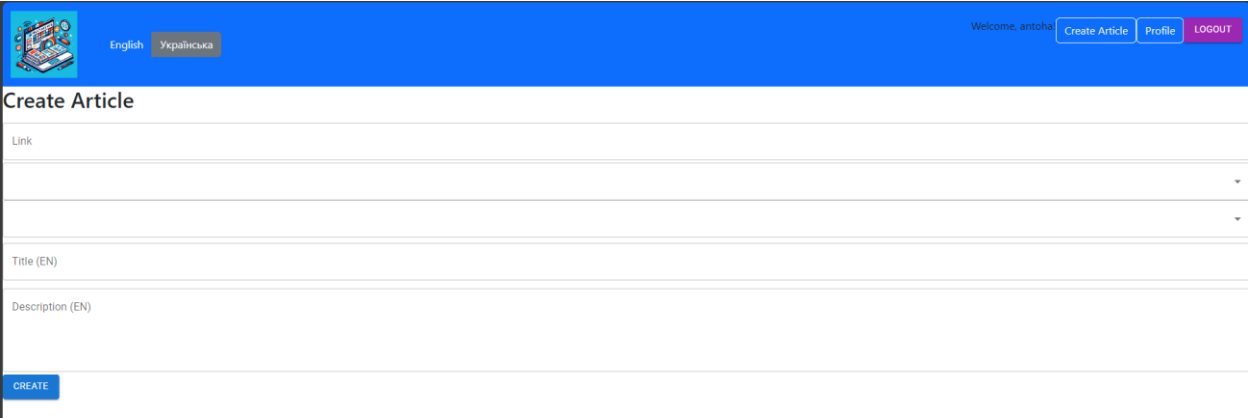


Рисунок 4.4 – Поля для створення статті

При спробі лишити необхідні поля не заповненими – користувачу буде відображене відповідне повідомлення (рисунок 4.5).

Create Article

Link

chess.com

Tech

Select Source

Source is required

Title (EN)

Title is required

Description (EN)

Description is required

CREATE

Рисунок 4.5. – Сповіщення про пусті поля

Заповнимо всі необхідні поля та натиснемо кнопку Create (рисунок 4.6).



English Українська

Welcome, antoha

Create Article Profile LOGOUT

Create Article

Link

chess.com

Sports

Other users

Title (EN)

Here we can see simple article title

Description (EN)

Here you can see simple article description which will be translated

CREATE

Рисунок 4.6 – Заповнені поля для створення статті

Після створення переглянемо нашу статтю в оригіналі (англійською мовою), на сторінці відображена повна інформація про статтю: заголовок, опис, джерело, категорія та автор (рисунок 4.7).



English Українська

Welcome, antoha

Create Article Profile LOGOUT

Here we can see simple article title

Here you can see simple article description which will be translated

Source: Other users

Date: 5/29/2024, 12:29:18 PM

Categories: SPORTS

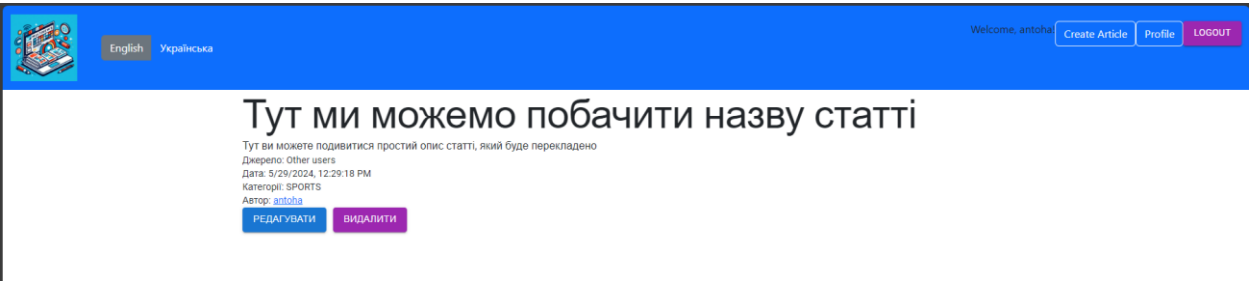
Author: antoha

EDIT

DELETE

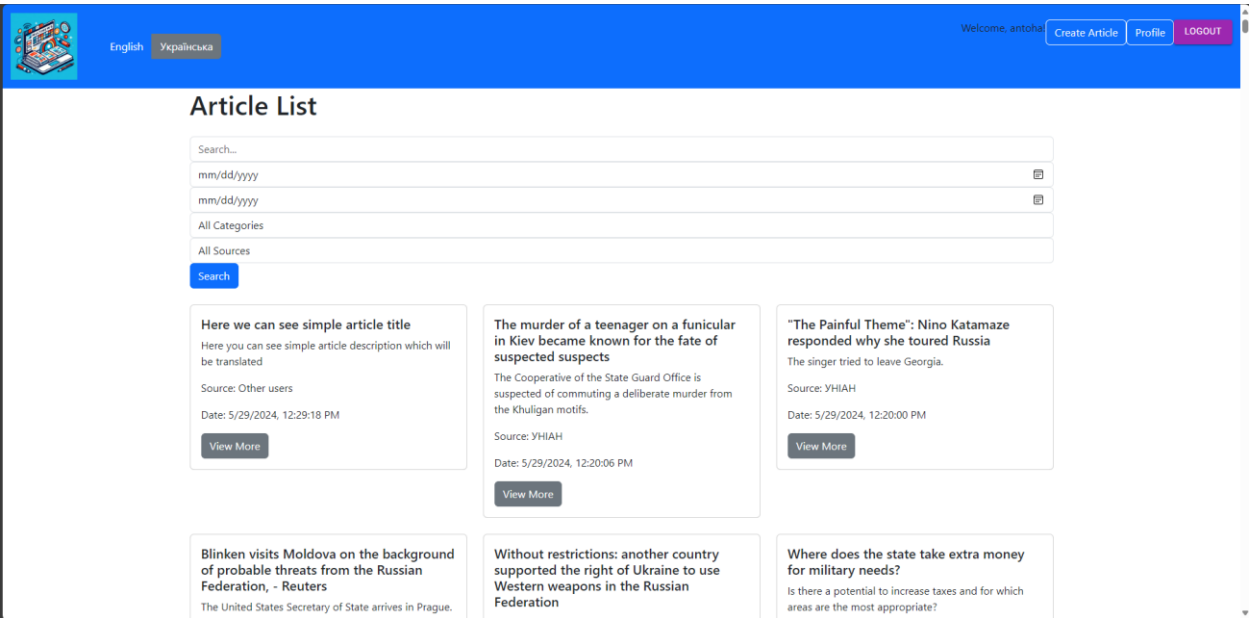
Рисунок 4.7 – Створена стаття англійською мовою

Далі змінимо мову ресурсу на українську і переглянемо цю ж статтю в перекладеному вигляді (рисуюнок 4.8).



Рисуюнок 4.8 – Створена стаття автоматично перекладено на українську мову

Можемо побачити щойно створену статтю серед загального списку новин (рисуюнок 4.9).



Рисуюнок 4.9 – Стаття в загальній стрічці новин

Висновки до розділу

В цьому розділі було викладено інформацію про такий аспект розробки програмного забезпечення як тестування.

В першому пункті – «Аналіз якості ПЗ» – було описано основні характеристики якісного коду. Також було описано досягнення цих

характеристик за допомогою аналізаторів коду, яке було підкріплене ілюстраціями та прикладами.

В другому пункті – «Опис процесів тестування» – було визначено спосіб тестування. За допомогою спеціальних таблиць – тест-кейсів – було описано тестування головних функцій нашого веб-застосунку. В тест-кейсах була описана основна інформація щодо процесу: послідовність кроків, необхідних для відтворення тесту, очікуваний та фактичний результати, стан системи необхідний для виконання тесту, вхідні дані та інше.

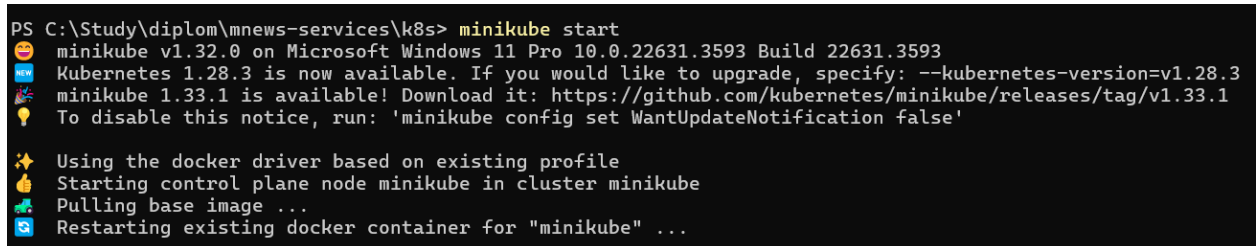
В третьому пункті – «Опис контрольного прикладу» – було продемонстровано сценарій створення статті, можливі причини відмов та результати перекладу, процес супроводжено ілюстраціями. Таким чином ми змогли впевнитися в ефективності створення та перекладі статей.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання мікросервісного застосунку використовується Kubernetes. Розглянемо розгортання локального K8s кластеру, для цього буде використовуватись Minikube[30], цей інструмент дає можливість запустити кластер з одним вузлом у віртуальній машині. Також необхідний інтерфейс командного рядка – kubectl[31], для роботи з Kubernetes. Кожен мікросервіс складається з Docker образу.

Отже, після встановлення всіх необхідних утиліт – існує наступний алгоритм дій. Першим кроком – необхідно запустити Minikube, за допомогою команди `minikube start` (рисунок 5.1).

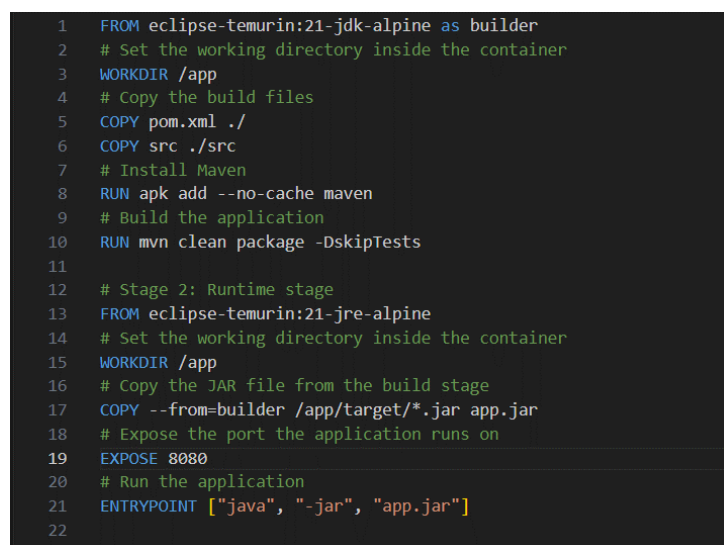


```
PS C:\Study\diplom\mnews-services\k8s> minikube start
minikube v1.32.0 on Microsoft Windows 11 Pro 10.0.22631.3593 Build 22631.3593
Kubernetes 1.28.3 is now available. If you would like to upgrade, specify: --kubernetes-version=v1.28.3
minikube 1.33.1 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.33.1
To disable this notice, run: 'minikube config set WantUpdateNotification false'

✨ Using the docker driver based on existing profile
👉 Starting control plane node minikube in cluster minikube
🔄 Pulling base image ...
🔄 Restarting existing docker container for "minikube" ...
```

Рисунок 5.1 – Запуск локального K8s кластеру

Далі з кожного сервісу системи необхідно створити Docker образ. Для прикладу буде розглянуто структуру Dockerfile, що використовується для створення образу Java Spring застосунку (рисунок 5.2).



```
1 FROM eclipse-temurin:21-jdk-alpine as builder
2 # Set the working directory inside the container
3 WORKDIR /app
4 # Copy the build files
5 COPY pom.xml ./
6 COPY src ./src
7 # Install Maven
8 RUN apk add --no-cache maven
9 # Build the application
10 RUN mvn clean package -DskipTests
11
12 # Stage 2: Runtime stage
13 FROM eclipse-temurin:21-jre-alpine
14 # Set the working directory inside the container
15 WORKDIR /app
16 # Copy the JAR file from the build stage
17 COPY --from=builder /app/target/*.jar app.jar
18 # Expose the port the application runs on
19 EXPOSE 8080
20 # Run the application
21 ENTRYPOINT ["java", "-jar", "app.jar"]
22
```

Рисунок 5.2 – Dockerfile для створення образу Java Spring application

Цей Dockerfile використовує Multi-stage build[32], даний підхід до побудови образу допомагає зменшити фінальний розмір, що є важливою складовою для мікросервісної архітектури. На першому етапі ми виконуємо побудову образу та завантаження всіх залежностей, а на другому етапі всі інструменти збірки, та інші непотрібні файли відкидаються, і лишається лише кінцевий застосунок.

В Kubernetes є ключові об'єкти що використовуються для керування конфігураціями та процесами розгортання застосунку. Вебзастосунок агрегації та автоматичного перекладу новин має наступну структуру конфігураційних файлів для кожного з сервісів (рисунок 5.3).

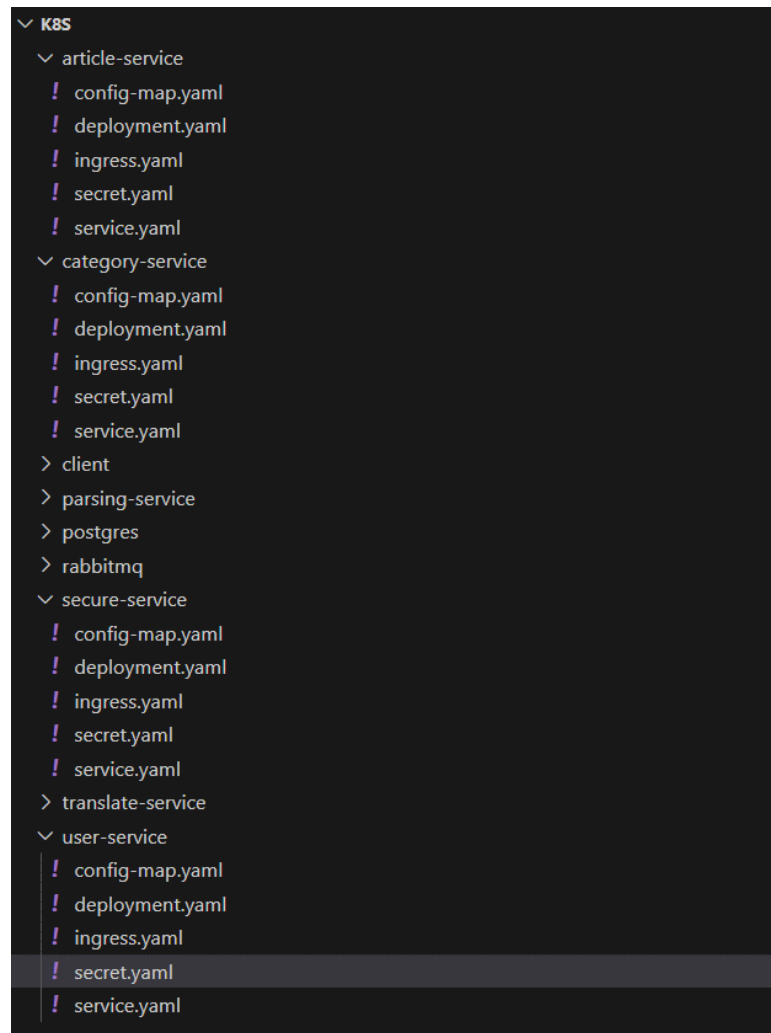


Рисунок 5.3 – Структура файлів конфігурації мікросервісів

ConfigMap[33] – K8s об'єкт, що дозволяє зберігати дані конфігурації (змінні оточення, команди, аргументи) у форматі ключ-значення (рисунок 5.4).

```

article-service > ! config-map.yaml
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: article-service-config
5  data:
6    POSTGRES_DB: "mnews_articles"
7    POSTGRES_HOST: "postgres:5432"
8    RABBITMQ_HOST: "rabbitmq-cluster"

```

Рисунок 5.4 – Файл config-map.yaml, що використовується для конфігурації

Deployment[34] – K8s об'єкт що відповідає за управління подами та репліками, допомагаючи забезпечити необхідну кількість подів в потрібний момент часу. В Deployment описується бажаний стан при розгортанні, а Deployment Controller[35] змінює поточний стан, на необхідний. Приклад deployment.yaml представлено на рисунку 5.5.

```

article-service > ! deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: article-service-deployment
5  spec:
6    replicas: 1                                # кількість екземплярів Pod, що буде створено цим Deployment
7    selector:
8      matchLabels:
9        app: article-service                  # селектор по якому Deployment зв'язаний з Pod
10   template:
11     metadata:
12       labels:
13         app: article-service                # мітка Pod по якому вона буде зв'язана з Deployment та Service
14     spec:
15       initContainers:
16         - name: run-migrations-news
17           image: antohachaban/article-service-migrations:latest
18           command: ["migrate", "-path", "/migrations", "-database", "${POSTGRES_URL}", "goto", "${VERSION}"]
19           env:
20             - name: POSTGRES_URL
21               value: "postgres://admin:admin@postgres:5432/mnews_articles?sslmode=disable"
22             - name: VERSION
23               value: "000001"
24       containers:
25         - image: antohachaban/article-service:latest      # образ Docker
26           imagePullPolicy: IfNotPresent
27           name: article-service                            # ім'я контейнеру
28           ports:
29             - containerPort: 8080                        # порт по якому можна звертатись до поди
30           envFrom:
31             - configMapRef:
32               name: article-service-config
33             - secretRef:
34               name: article-service-secret

```

Рисунок 5.5 – Файл deployment.yaml, що використовується для конфігурації

Ingress[36] – об'єкт K8s що надає доступ до сервісів ззовні кластера, використовуючи HTTP/HTTPS. Також надає можливість балансування

навантаження. При використанні з Minikube – необхідно увімкнути відповідне розширення, використовуючи команду `minikube addons enable ingress`. Приклад файлу `ingress.yaml` представлено на рисунку 5.6.

```

article-service > ! ingress.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: article-service-ingress
5    annotations:
6      kubernetes.io/ingress.class: "nginx"
7  spec:
8    rules:
9      - http:
10        paths:
11          - path: /api/articles
12            pathType: "Prefix"
13            backend:
14              service:
15                name: article-service-service
16                port:
17                  number: 80

```

Рисунок 5.6 – Файл `ingress.yaml`, що використовується для контролю зовнішнього доступу до сервісів

`Secret`[37] – об’єкт що використовується для зберігання конфіденційної інформації, наприклад паролів, токенів та інших ключів (рисунок 5.7).

```

article-service > ! secret.yaml
1  apiVersion: v1
2  kind: Secret
3  type: Opaque
4  metadata:
5    name: article-service-secret
6
7  data:
8    POSTGRES_USER: "YwRtaW4=" # admin
9    POSTGRES_PASSWORD: "YwRtaW4=" # admin
10   RABBITMQ_USER: "bWRTYmQ3NzFpM3pmdEJ0NDRFZDk3bEJXVmN5cW1ubEw=" # mdmbd771i3zftBt44Ed97lBwVcyqmnLL
11   RABBITMQ_PASSWORD: "VmMTZDhMV0lfeVFKRk9LMjNmMFBAQWJwSXRyRmNlbiDY=" # Vc-d8LWI_yQJF0K23p0PZAbpItXFcb16

```

Рисунок 5.7 – Файл `secret.yaml`, використовується для збереження конфіденційних даних конфігурації

Service[38] – визначає метод представлення застосунку, дозволяють визначати те, як поди спілкуються між собою, та зовнішніми клієнтами (рисунок 5.8).

```

article-service > ! service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: article-service-service
5  spec:
6    type: ClusterIP
7    ports:
8      - port: 80          # порт по якому можна звертись до сервісу
9        targetPort: 8080 # порт контейнера
10   selector:
11     app: article-service

```

Рисунок 5.8 – Файл service.yaml, що використовується для конфігурації

Після того як всі файли конфігурацій готові – необхідно застосувати їх, використовуючи команду `kubectl apply -f <file/folder>` (рисунок 5.9).

```

Windows PowerShell
PS C:\Study\diplom\mnews-services> kubectl apply -f .\k8s\article-service\
configmap/article-service-config created
deployment.apps/article-service-deployment created
ingress.networking.k8s.io/article-service-ingress created
secret/article-service-secret created
service/article-service-service created
PS C:\Study\diplom\mnews-services>

```

Рисунок 5.9 – Виконання команди `kubectl apply` для застосування конфігурації

Далі виконаємо такі ж кроки для інших сервісів, та перевіримо успішність запуску за допомогою команди `kubectl get pods` (рисунок 5.10).

```

PS C:\Study\diplom\mnews-services> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
article-service-deployment-576956857f-rdwwc  1/1     Running   0           2m58s
category-service-deployment-6c5757db4c-5v9p6  1/1     Running   2 (15h ago)  2d14h
client-service-deployment-5bf8d469df-78gbm    1/1     Running   4 (15h ago)  3d1h
parsing-service-deployment-79dc7cb987-kmvr    1/1     Running   2 (15h ago)  2d14h
postgres-6b875d7596-ckqbm                  1/1     Running   6 (15h ago)  3d18h
rabbitmq-cluster-server-0                   1/1     Running   14 (15h ago)  13d
secure-service-deployment-64cddf9b5-rp47m     1/1     Running   6 (15h ago)  3d18h
user-service-deployment-6c8dcd896-b5t8h       1/1     Running   6 (15h ago)  3d18h
PS C:\Study\diplom\mnews-services> |

```

Рисунок 5.10 – Виконання команди `kubectl get pods`

Після того як всі поди запущені та мають статус Running – необхідно виконати команду `minikube tunnel`[39]. Після виконання цієї команди, створюється новий мережевий інтерфейс та призначається відповідна IP-адреса для доступу. Результат виконання команди представлено на рисунку 5.11.

```
PS C:\Study\diplom\mnews-services> minikube tunnel
Tunnel successfully started

NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

! Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: http
s://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
! Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: http
s://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
* Starting tunnel for service article-service-ingress.
! Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: http
s://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
* Starting tunnel for service category-service-ingress.
! Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: http
s://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
* Starting tunnel for service client-service-ingress.
! Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: http
s://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
* Starting tunnel for service secure-service-ingress.
* Starting tunnel for service user-service-ingress.
```

Рисунок 5.11 – Виконання команди minikube tunnel

Після виконання цих дій, наш застосунок доступний локально за адресою <http://localhost/>.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Користувачі повинні мати можливість отримати оновлення веб застосунку з кожною версією.

Так як застосунок працює на основі Docker образів, то для підтримки програмного забезпечення використовується їх оновлення. Для цього нову версію Docker Image необхідно зібрати та завантажити на Docker Hub[40] за допомогою команди `docker image push`. Так як Kubernetes Deployment використовує тег `:latest`, то програмне забезпечення завжди буде використовувати останню версію вказаного образу.

Висновки до розділу

В розділі «Розгортання та супровід програмного забезпечення» було розглянуто підхід до розгортання застосунку за допомогою Docker та

Kubernetes з використанням утиліти Minikube. Алгоритм розгортання було описано покроково, та супроводжено відповідними ілюстраціями.

Було описано процес супроводу програмного забезпечення, та описано алгоритм підтримки нових версій застосунку.

ВИСНОВКИ

У даній роботі було проаналізовано та надано рішення проблеми доступу до новин різними мовами та з різних джерел. В ході аналізу предметної області та вже існуючих рішень було сформульована задача, в ході виконання якої необхідно створити продукт, що поєднав би в собі агрегатор новин, та звичайний новинний портал, з підтримкою декількох мов, та зручним пошуком за встановленими параметрами.

В ході виконання дипломного проекту було розглянуто різні підходи до побудови архітектури та обрано найбільш підходящу – мікросервісну. Цей підхід забезпечив можливість зручного поділу застосунку на логічні модулі, та його подальшої модифікації.

В результаті розробки застосунку «MultiNews», вдалося виконати всі зі сформульованих задач і отримати готовий продукт, а саме:

- створено підтримку декількох мов;
- реалізовано агрегацію новин з різних новинних джерел, а також автоматичне визначення категорій для них;
- поєднано в собі функціонал агрегаторів новин та звичайних новинних порталів, де редактори можуть писати власні статті;
- реалізовано пошук за ключовими словами та параметрами (категорії, дати, джерела);
- автоматичне оновлення списку доступних новин з різних джерел кожні 30 хвилин.

Після реалізації веб-застосунку він був протестований в різних браузерах та версіях, з різними розмірами сторінки щоб переконатися, що додаток коректно працює в різних форматах перегляду.

Головними інструментами, що використовувались при розробці були мова Java та середовище розробки IntelliJ IDEA. В якості бази даних було використано PostgreSQL, а для оркестрації мікросервісів була використана платформа Kubernetes.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Список найпопулярніших інтернет-ЗМІ в Україні – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Список_найпопулярніших_інтернет-ЗМІ_в_Україні (дата звернення: 20.03.2024).
- 2) Top News & Media Publishers Websites Ranking in United States in April 2024. URL: <https://www.similarweb.com/top-websites/united-states/news-and-media/> (дата звернення: 20.03.2024)
- 3) Новини України - останні новини України сьогодні - УНІАН. URL: <https://www.unian.ua/> (дата звернення: 25.03.2024)
- 4) Breaking News – CNN. URL: <https://edition.cnn.com/> (дата звернення: 25.03.2024).
- 5) Букви новини. URL: <https://bukvy.org/> (дата звернення: 25.03.2024).
- 6) Google News. URL: <https://news.google.com/> (дата звернення: 26.03.2024).
- 7) Pattern: Monolithic Architecture. URL: <https://microservices.io/patterns/monolithic.html> (дата звернення: 17.04.2024).
- 8) Microservice Architecture pattern. URL: <https://microservices.io/patterns/microservices.html> (дата звернення: 17.04.2024).
- 9) The C4 model for visualising software architecture. URL: <https://c4model.com/> (дата звернення: 20.04.2024).
- 10) News Category Dataset. URL: <https://www.kaggle.com/datasets/rmisra/news-category-dataset> (дата звернення: 07.05.2024).
- 11) What is a Relational Database (RDBMS)? URL: <https://www.oracle.com/database/what-is-a-relational-database/> (дата звернення: 07.05.2024).
- 12) Docker Overview URL: <https://docs.docker.com/get-started/overview/> (дата звернення: 07.05.2024).

- 13) Kubernetes Overview. URL: <https://kubernetes.io/docs/concepts/overview/> (дата звернення: 08.05.2024).
- 14) Objects Used in Kubernetes. URL: <https://kodekloud.com/blog/kubernetes-objects/> (дата звернення: 08.05.2024).
- 15) Java Platform, Enterprise Edition. URL: <https://www.oracle.com/cis/java/technologies/java-ee-glance.html> (дата звернення: 08.05.2024).
- 16) Java Spring. URL: <https://spring.io/> (дата звернення: 08.05.2024).
- 17) Spring Security. URL: <https://spring.io/projects/spring-security> (дата звернення: 08.05.2024).
- 18) Advantages of React JS. URL: <https://www.linkedin.com/pulse/advantages-react-js-thescholarhat-v8c3f/> (дата звернення: 08.05.2024).
- 19) About PostgreSQL. URL: <https://www.postgresql.org/about/> (дата звернення: 08.05.2024).
- 20) RabbitMQ – broker overview. URL: <https://www.rabbitmq.com/> (дата звернення: 08.05.2024).
- 21) Apache Kafka. URL: <https://kafka.apache.org/> (дата звернення: 08.05.2024).
- 22) What's the Difference Between Kafka and RabbitMQ? URL: https://aws.amazon.com/compare/the-difference-between-rabbitmq-and-kafka/?nc1=h_ls (дата звернення: 08.05.2024).
- 23) IntelliJ IDEA The Leading Java and Kotlin IDE. URL: <https://www.jetbrains.com/idea/> (дата звернення: 09.05.2024).
- 24) Pattern: Database per service. URL: <https://microservices.io/patterns/data/database-per-service.html> (дата звернення: 09.05.2024).
- 25) Cross Site Scripting (XSS) Attack. URL: <https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/> (дата звернення: 10.05.2024).

- 26) SQL Injection. URL: https://www.w3schools.com/sql/sql_injection.asp (дата звернення: 10.05.2024).
- 27) Cross Site Request Forgery (CSRF). URL: <https://owasp.org/www-community/attacks/csrf> (дата звернення: 10.05.2024).
- 28) A Guide to CSRF Protection in Spring Security. URL: <https://www.baeldung.com/spring-security-csrf> (дата звернення: 10.05.2024).
- 29) Minikube Dashboard. URL: <https://minikube.sigs.k8s.io/docs/handbook/dashboard/> (дата звернення: 13.05.2024).
- 30) Minikube Overview. URL: <https://kubernetes.io/uk/docs/tasks/tools/install-minikube/> (дата звернення: 14.05.2024).
- 31) K8s tools – kubectl URL: <https://kubernetes.io/docs/tasks/tools/> (дата звернення: 15.05.2024).
- 32) Docker Multi-stage builds. URL: <https://docs.docker.com/build/building/multi-stage/> (дата звернення: 15.05.2024).
- 33) ConfigMaps – Kubernetes. URL: <https://kubernetes.io/docs/concepts/configuration/configmap/> (дата звернення: 17.05.2024).
- 34) Deployments – Kubernetes. URL: <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/> (дата звернення: 17.05.2024).
- 35) Deployment Controllers – Kubernetes. URL: <https://kubernetes.io/docs/concepts/architecture/controller/> (дата звернення: 17.05.2024).
- 36) Ingress – Kubernetes. URL: <https://kubernetes.io/docs/concepts/services-networking/ingress/> (дата звернення: 17.05.2024).

- 37) Secrets — Kubernetes. URL:
<https://kubernetes.io/docs/concepts/configuration/secret/> (дата
звернення: 17.05.2024).
- 38) Service — Kubernetes. URL:
<https://kubernetes.io/docs/concepts/services-networking/service/> (дата
звернення: 17.05.2024).
- 39) Minikube Tunnel. URL:
<https://minikube.sigs.k8s.io/docs/commands/tunnel/> (дата звернення: 18.05.2024).
- 40) Docker Image Push. URL:
<https://docs.docker.com/reference/cli/docker/image/push/> (дата
звернення: 18.05.2024).

ДОДАТКИ

ДОДАТОК А Звіт подібності



Ім'я користувача: Лісовиченко Олег Іванович	ID перевірки: 1016311086
Дата перевірки: 02.06.2024 17:01:06 EEST	Тип перевірки: Doc vs Internet + Library
Дата звіту: 02.06.2024 18:55:44 EEST	ID користувача: 76913

Назва документа: IT-01_Чабан_ПЗ
Кількість сторінок: 71 Кількість слів: 9451 Кількість символів: 75536 Розмір файлу: 1.81 MB ID файлу: 1016107869

12.5%
Схожість

Найбільша схожість: 4.57% з джерелом з Бібліотеки (ID файлу: 1016100512)

3.06% Джерела з Інтернету	144		Сторінка 73
12.4% Джерела з Бібліотеки	298		Сторінка 74

0% Цитат

- Вилучення цитат вимкнене
- Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

- Немає вилучених джерел

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2024 р.

Вебзастосунок для агрегації та автоматичного перекладу новин

Текст програми

КПІ.IT-0325.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена СИРОТА

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Антон ЧАБАН

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
<https://github.com/antonchaban/mnews-services/>

Файл UserController.java

Реалізація функціональних вимог: реєстрація користувача, авторизація користувача, вихід користувача, редагування користувача, видалення користувача.

```
@RestController
@RequestMapping("/api")
@AllArgsConstructor
public class UserController {
    private final UserServiceImpl userService;
    private final UserRepo userRepo;

    @GetMapping("users")
    public ResponseEntity<List<UserDto>> getAllUsers() {
        return ResponseEntity.ok(userService.findAll());
    }

    @PostMapping("users")
    public ResponseEntity<User> createUser(@RequestBody CreateUserDto user)
    throws NameAlreadyBoundException {
        return ResponseEntity.ok(userService.createUser(user));
    }

    @GetMapping("users/{id}")
    public ResponseEntity<UserDto> getUserById(@PathVariable Long id) {
        return ResponseEntity.ok(userService.findUserById(id));
    }

    @PatchMapping("users/{id}")
    public ResponseEntity<UserDto> updateUser(@PathVariable Long id,
    @RequestBody UpdateUserDto user,
    @CookieValue(name = "USER_ID")
    Long userId) {
        User curUser = userRepo.findById(userId).orElse(null);
        assert curUser != null;
        if (curUser.getRoles().stream().anyMatch(role ->
        role.equals(Role.ROLE_ADMIN))) {
            return ResponseEntity.ok(userService.updateUser(id, user));
        }
        return ResponseEntity.status(403).build();
    }

    @DeleteMapping("users/{id}")
    public ResponseEntity<Void> deleteUser(@PathVariable Long id,
    @CookieValue(name = "USER_ID") Long userId) {
        User curUser = userRepo.findById(userId).orElse(null);
        assert curUser != null;
        if (curUser.getRoles().stream().anyMatch(role ->
        role.equals(Role.ROLE_ADMIN))) {
            userService.deleteUser(id);
            return ResponseEntity.ok().build();
        }
    }
}
```

```

    }
    return ResponseEntity.status(403).build();
}
}

```

Файл UserServiceImpl.java

Реалізація функціональних вимог: реєстрація користувача, редагування користувача, видалення користувача.

```

@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {
    private final UserRepo userRepo;
    private final PasswordEncoder passwordEncoder;
    private final MappingUtils mappingUtils;
    private final RabbitMQUserProducer producer;

    @Override
    public User findByUsername(String username) {
        return userRepo.findByUsername(username);
    }

    @Override
    public UserDto findUserById(Long id) {
        return
userRepo.findById(id).map(mappingUtils::mapToUserDto).orElse(null);
    }

    @Override
    public List<UserDto> findAll() {
        return
userRepo.findAll().stream().map(mappingUtils::mapToUserDto).collect(Collectors.toList());
    }

    @Override
    public User createUser(CreateUserDto user) throws
NameAlreadyBoundException {
        if (userRepo.findByUsername(user.getUsername()) != null) {
            throw new NameAlreadyBoundException("Username already taken");
        }
        User userEntity = new User();
        userEntity.setUsername(user.getUsername());
        userEntity.setPassword(passwordEncoder.encode(user.getPassword()));
        userEntity.getRoles().add(Role.ROLE_EDITOR);
        userRepo.save(userEntity);
        producer.sendUserEntity(userEntity);
        return userEntity;
    }

    @Override
    public UserDto updateUser(Long id, UpdateUserDto user) {
        User userEntity = userRepo.findById(id).orElse(null);
        if (userEntity == null) {
            return null;
        }
        userEntity.getRoles().clear();
        userEntity.getRoles().addAll(user.getRole());
        userRepo.save(userEntity);
    }
}

```

```

        producer.sendUserEntity(userEntity);
        return mappingUtils.mapToUserDto(userEntity);
    }

    @Override
    public void deleteUser(Long id) {
        userRepo.deleteById(id);
        producer.sendDeleteUser(id);
    }
}

```

Файл UserRepo.java

Реалізація функціональних вимог: реєстрація користувача, авторизація користувача, вихід користувача, редагування користувача, видалення користувача.

```

@Repository
public interface UserRepo extends JpaRepository<User, Long> {

    User findByUsername(String username);

    User findUserById(Long id);
}

```

Файл User.java

Реалізація функціональних вимог: реєстрація користувача, авторизація користувача, вихід користувача, редагування користувача, видалення користувача.

```

@Entity
@Table(name = "users")
@Getter
@Setter
@NoArgsConstructor
@ToString
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    @Column(length = 1000)
    @JsonIgnore
    private String password;

    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "role"
        , joinColumns = @JoinColumn(name = "id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles = new HashSet<>();

    // Spring Security

    public boolean isAdmin() {
        return roles.contains(Role.ROLE_ADMIN);
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {

```

```

        return roles;
    }

    @Override
    public String getUsername() {
        return username;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}

```

Файл Role.java

Реалізація функціональних вимог: реєстрація користувача, авторизація користувача, редагування користувача, видалення користувача.

```

public enum Role implements GrantedAuthority {
    ROLE_EDITOR, ROLE_ADMIN;

    @Override
    public String getAuthority() {
        return name();
    }
}

```

Файл AuthFilter.java

Реалізація функціональних вимог: авторизація користувача.

```

@AllArgsConstructor
public class AuthFilter extends UsernamePasswordAuthenticationFilter {

    private final AuthenticationManager authManager;
    private final ObjectMapper objectMapper = new ObjectMapper();

    @Override
    public Authentication attemptAuthentication(HttpServletRequest request,
        HttpServletResponse response) throws AuthenticationException {
        try {
            // Read the credentials from the request body
            Credentials credentials =
                objectMapper.readValue(request.getInputStream(), Credentials.class);

            UsernamePasswordAuthenticationToken token = new
                UsernamePasswordAuthenticationToken(credentials.getUsername(),
                    credentials.getPassword());

```

```

        response.setHeader("Access-Control-Allow-Origin", "*");
        return authManager.authenticate(token);
    } catch (IOException e) {
        throw new AuthenticationServiceException("Error reading
credentials from the request body", e);
    }
}

@Override
protected void successfulAuthentication(HttpServletRequest request,
HttpServletResponse response, FilterChain chain, Authentication authResult)
throws IOException, ServletException {
    User user = (User) authResult.getPrincipal();
    Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());

    String accessToken = JWT.create()
        .withSubject(user.getUsername())
        .withExpiresAt(new Date(System.currentTimeMillis() + 10 * 60
* 1000))
        .withIssuer(request.getRequestURL().toString())
        .withClaim("roles",
user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()))
        .sign(algorithm);

    String refreshToken = JWT.create()
        .withSubject(user.getUsername())
        .withExpiresAt(new Date(System.currentTimeMillis() + 60 * 60
* 1000))
        .withIssuer(request.getRequestURL().toString())
        .sign(algorithm);

    Map<String, String> tokens = new HashMap<>();
    tokens.put("accessToken", accessToken);
    tokens.put("refreshToken", refreshToken);
    tokens.put("id", user.getId().toString());

    response.setContentType(APPLICATION_JSON_VALUE);
    new ObjectMapper().writeValue(response.getOutputStream(), tokens);
}

@Getter
@Setter
static
class Credentials {
    private String username;
    private String password;
}
}

```

Файл AuthComponent.jsx

Реалізація функціональних вимог: авторизація користувача.

```

const AuthComponent = () => {
    const [cookies] = useCookies(['ACCESS_TOKEN']);
    const token = cookies['ACCESS_TOKEN'];
    let isLoggedIn = false;
    let username = '';
    const userId = cookies['USER_ID'];

    if (token) {
        const decodedToken = jwtDecode(token);
    }
}

```

```

        username = decodedToken.sub;
        isLoggedIn = true;
    }

    if (isLoggedIn) {
        return (
            <div>
                <p>Welcome, {username}!</p>
                <p><Link to={`/profile/${userId}`}>Profile</Link></p>
            </div>
        );
    } else {
        return (
            <div>
                <p>Please log in to view your profile.</p>
                <p><Link to="/auth">Login</Link></p>
            </div>
        );
    }
};

export default AuthComponent;

```

Файл SignUp.jsx

Реалізація функціональних вимог: реєстрація користувача.

```

const SignUp = () => {
    const [formData, setFormData] = useState({
        username: '',
        password: '',
        confirmPassword: ''
    });
    const [error, setError] = useState('');
    const navigate = useNavigate();

    const handleChange = (e) => {
        setFormData({...formData, [e.target.name]: e.target.value});
    };

    const handleSubmit = async (event) => {
        event.preventDefault();

        const {username, password, confirmPassword} = formData;

        if (password !== confirmPassword) {
            setError("Passwords don't match");
            return;
        }

        try {
            const newUser = {username, password};

            // Send sign-up request
            const res = await axios.post('http://localhost/api/users',
newUser);

            if (res.status === 200) {
                // Redirect to login page after successful sign-up
                navigate('/auth');
            }
        } catch (error) {
            console.error('Error processing sign-up request', error);
            setError('An error occurred. Please try again.');
```

```

    }
  };

  return (
    <Container component="main" maxWidth="xs">
      <Box
        sx={{
          marginTop: 8,
          display: 'flex',
          flexDirection: 'column',
          alignItems: 'center'
        }}
      >
        <Avatar sx={{m: 1, bgcolor: 'secondary.main'}}>
          <LockOutlinedIcon color="primary"/>
        </Avatar>
        <Typography component="h1" variant="h5">
          Sign up
        </Typography>
        <Box component="form" onSubmit={handleSubmit} noValidate
          sx={{mt: 3}}>
          <Grid container spacing={2}>
            <Grid item xs={12}>
              <TextField
                fullWidth
                id="username"
                label="Username"
                name="username"
                autoComplete="username"
                value={formData.username}
                onChange={handleChange}
              />
            </Grid>
            <Grid item xs={12}>
              <TextField
                fullWidth
                name="password"
                label="Password"
                type="password"
                id="password"
                autoComplete="new-password"
                value={formData.password}
                onChange={handleChange}
              />
            </Grid>
            <Grid item xs={12}>
              <TextField
                fullWidth
                name="confirmPassword"
                label="Confirm Password"
                type="password"
                id="confirmPassword"
                autoComplete="new-password"
                value={formData.confirmPassword}
                onChange={handleChange}
              />
            </Grid>
            <Grid>
              <Button type="submit" fullWidth variant="contained"
                sx={{mt: 3, mb: 2}}>
                Sign Up
              </Button>
              {error && (
                <Typography variant="body2" color="error"

```

```

align="center">
    {error}
  </Typography>
)}
<Grid container justifyContent="flex-end">
  <Grid item>
    <Link href="/auth" variant="body2">
      Already have an account? Sign in
    </Link>
  </Grid>
</Grid>
</Box>
</Box>
</Container>
);
};

export default SignUp;

```

Файл AdminComponent.jsx

Реалізація функціональних вимог: редагування користувача, видалення користувача.

```

const AdminComponent = () => {
  const [cookies] = useCookies(['USER_ID']);
  const [users, setUsers] = useState([]);
  const [roleEdits, setRoleEdits] = useState({});
  const [error, setError] = useState('');
  const navigate = useNavigate();

  useEffect(() => {
    if (!cookies.USER_ID) {
      navigate('/auth');
    } else {
      axios.get(`http://localhost/api/users/${cookies.USER_ID}`)
        .then(response => {
          if (!response.data.roles.includes('ROLE_ADMIN')) {
            setError('This page is not allowed for you');
          } else {
            axios.get('http://localhost/api/users')
              .then(response => {
                setUsers(response.data);
              })
              .catch(error => {
                console.error('Error fetching users:',
error);
              });
          }
        })
        .catch(error => {
          console.error('Error fetching user profile:', error);
        });
    }
  }, [cookies.USER_ID, navigate]);

  const handleDelete = (id) => {
    axios.delete(`http://localhost/api/users/${id}`)
      .then(response => {
        setUsers(users.filter(user => user.id !== id));
      })
      .catch(error => {
        console.error('Error deleting user:', error);
      });
  };

```

```

};

const handleRoleChange = (id, role) => {
  setRoleEdits(prevState => ({...prevState, [id]: role}));
};

const handleRoleEdit = (id) => {
  const role = roleEdits[id];
  if (role) {
    axios.patch(`http://localhost/api/users/${id}`, {id, role:
[role]})
      .then(response => {
        setUsers(users.map(user => user.id === id ? {...user,
role: [role]} : user));
      })
      .catch(error => {
        console.error('Error updating user role:', error);
      });
  }
};

if (error) {
  return <Typography variant="h4" color="error">{error}</Typography>;
}

return (
  <Container>
    <Typography variant="h2">Admin Panel</Typography>
    <TableContainer>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell>Username</TableCell>
            <TableCell>Role</TableCell>
            <TableCell>Delete</TableCell>
            <TableCell>Edit</TableCell>
            <TableCell>More information</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {users.map(user => (
            <TableRow key={user.id}>
              <TableCell>{user.username}</TableCell>
              <TableCell>{user.roles.join(',
' )}</TableCell>
              <TableCell>
                <Button
                  variant="contained"
                  color="secondary"
                  onClick={() => handleDelete(user.id)}
                >
                  Delete
                </Button>
              </TableCell>
              <TableCell>
                <Select
                  value={roleEdits[user.id] ||
user.roles[0]}
                  onChange={(e) =>
handleRoleChange(user.id, e.target.value)}
                >
                  <MenuItem
value="ROLE_ADMIN">Admin</MenuItem>
                  <MenuItem

```

```

value="ROLE_EDITOR">Editor</MenuItem>
    </Select>
    <Button
      variant="contained"
      color="primary"
      onClick={ () =>
        handleRoleEdit(user.id) }
    >
      Edit
    </Button>
  </TableCell>
  <TableCell>
    <Button
      variant="contained"
      component={Link}
      to={` /profile/${user.id}`}
    >
      View more
    </Button>
  </TableCell>
</TableRow>
  )}
</TableBody>
</Table>
</TableContainer>
</Container>
);
};

export default AdminComponent;

```

Файл Header.jsx

Реалізація функціональних вимог: вихід користувача.

```

const Header = ({isLoggedIn}) => {
  const location = useLocation();
  const [cookies, setCookie, removeCookie] = useCookies();
  const userId = cookies['USER_ID'];

  const handleLogout = () => {
    removeCookie(Cookies.USER_ID);
    removeCookie(Cookies.ACCESS_TOKEN);
    removeCookie(Cookies.REFRESH_TOKEN);
    window.location.href = '/articles';
  };

  // Check if the current location is the login page
  const isLoginPage = location.pathname === '/auth';

  return (
    <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
      <div className="container-fluid">
        <Link className="navbar-brand" to="/articles">
          
        </Link>
        <LanguageSwitcher />
        <div className="collapse navbar-collapse"
          id="navbarSupportedContent">
          <ul className="navbar-nav ms-auto">
            {!isLoginPage && (

```

```

        <li className="nav-item">
            <AuthComponent isLoggedIn={isLoggedIn}/>
        </li>
    )}
    {isLoggedIn && (
        <li className="nav-item">
            <Link to="/articles/create" className="btn
btn-outline-light">
                Create Article
            </Link>
        </li>
    )}
    {isLoggedIn && (
        <li className="nav-item">
            <Link to={` /profile/${userId}`}
className="btn btn-outline-light">
                Profile
            </Link>
        </li>
    )}
    {isLoggedIn && (
        <li className="nav-item">
            <Button onClick={handleLogout}
variant="contained" color="secondary">
                Logout
            </Button>
        </li>
    )}
    {!isLoggedIn && (
        <li className="nav-item">
            <Link to="/auth" className="btn btn-outline-
light">
                Login
            </Link>
        </li>
    )}
    </ul>
</div>
</div>
</nav>
);
};

export default Header;

```

Файл Article.java

Реалізація функціональних вимог: створення статті, редагування статті, видалення статті, перегляд статей присвоєних користувачу, пошук статті по ключовому слову, пошук статті по встановленим параметрам, перегляд статті.

```

@Entity
@Table(name = "articles")
@Getter
@Setter
@NoArgsConstructor
@ToString
@JsonInclude(JsonInclude.Include.NON_NULL)
public class Article {

```

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

private String link;

private Date articleDate;

private String source;

@Column(length = 2048)
private String title_ua;

@Column(length = 16384)
private String description_ua;

@Column(length = 2048)
private String title_en;

@Column(length = 16384)
private String description_en;

@ToString.Exclude
@ElementCollection(targetClass = Category.class, fetch = FetchType.EAGER)
@CollectionTable(name = "category"
    , joinColumns = @JoinColumn(name = "id"))
@Enumerated(EnumType.STRING)
private Set<Category> categories = new HashSet<>();

@ManyToOne(cascade = CascadeType.REFRESH)
@JoinColumn
@JsonIgnoreProperties("articles")
@ToString.Exclude
private User user;

@Transient
private Long userId;
}

```

Файл Category.java

Реалізація функціональних вимог: пошук статті по встановленим параметрам.

```

public enum Category {
    U_S_NEWS,
    COMEDY,
    PARENTING,
    WORLD_NEWS,
    CULTURE_AND_ARTS,
    TECH,
    SPORTS,
    ENTERTAINMENT,
    POLITICS,
    WEIRD_NEWS,
    ENVIRONMENT,
    EDUCATION,
    CRIME,
    SCIENCE,
    WELLNESS,
}

```

```

    BUSINESS,
    STYLE_AND_BEAUTY,
    FOOD_AND_DRINK,
    MEDIA,
    QUEER_VOICES,
    HOME_AND_LIVING,
    WOMEN,
    BLACK_VOICES,
    GEOPOLITICS,
    MONEY,
    RELIGION,
    LATINO_VOICES,
    IMPACT,
    WEDDINGS,
    COLLEGE,
    PARENTS,
    ARTS_AND_CULTURE,
    STYLE,
    GREEN,
    TASTE,
    HEALTHY_LIVING,
    THE_WORLDPOST,
    GOOD_NEWS,
    WORLDPOST,
    FIFTY,
    ARTS,
    DIVORCE,
    UNKNOWN
}

```

Файл ArticleController.java

Реалізація функціональних вимог: перегляд власних статей, створення статті, редагування статті, видалення статті, перегляд статей присвоєних користувачу, пошук статті по ключовому слову, пошук статті по встановленим параметрам, перегляд статті.

```

@RestController
@RequestMapping("/api")
@AllArgsConstructor
public class ArticleController {
    private final ArticleServiceImpl articleService;
    private final UserRepo userRepo;

    @GetMapping("/articles")
    public ResponseEntity<List<Article>> getArticles(
        @RequestParam(name = "userId", required = false) Long userId) {
        List<Article> articles;
        if (userId != null && userId > 0) {
            articles = articleService.findAllByUserId(userId);
        } else {
            articles = articleService.findAll();
        }
        return ResponseEntity.ok(articles);
    }

    @GetMapping("articles/{id}")
    public ResponseEntity<Article> getArticleById(@PathVariable long id) {
        return ResponseEntity.ok(articleService.findById(id));
    }
}

```

```

    @DeleteMapping("articles/{id}")
    public ResponseEntity<Void> deleteArticleById(@PathVariable long id,
    @CookieValue(name = "USER_ID") Long userId) {
        var article = articleService.findById(id);
        if (Objects.equals(article.getUser().getId(), userId) ||
isUserAdmin(userId)) {
            articleService.deleteById(id);
            return ResponseEntity.ok().build();
        } else {
            return ResponseEntity.status(403).build();
        }
    }

    @PostMapping("articles")
    public ResponseEntity<Article> createArticle(@RequestBody
ArticleCreateDTO article,
                                                @CookieValue(name =
"USER_ID") Long userId) {
        if (userId == null || userId <= 0) {
            return ResponseEntity.badRequest().build();
        }
        return ResponseEntity.ok(articleService.createArticle(article,
userId));
    }

    @PutMapping("articles/{id}")
    public ResponseEntity<Article> updateArticle(@PathVariable long id,
    @RequestBody ArticleEditDTO article,
                                                @CookieValue(name =
"USER_ID") Long userId) {
        var articleFromDb = articleService.findById(id);
        if (Objects.equals(articleFromDb.getUser().getId(), userId) ||
isUserAdmin(userId)) {
            return ResponseEntity.ok(articleService.updateArticle(article));
        } else {
            System.out.println("User id is not equal to article user id");
            System.out.println("User id: " + userId);
            System.out.println("Article user id: " +
articleFromDb.getUserId());
            return ResponseEntity.status(403).build();
        }
    }

    private boolean isUserAdmin(Long userId) {
        var user = userRepo.findById(userId).orElse(null);
        assert user != null;
        return user.getRoles().stream().anyMatch(role ->
role.equals(Role.ROLE_ADMIN));
    }
}

```

Файл ArticleServiceImpl.java

Реалізація функціональних вимог: перегляд власних статей, створення статті, редагування статті, видалення статті, перегляд статей присвоєних користувачу, пошук статті по ключовому слову, пошук статті по встановленим параметрам, перегляд статті.

```

@Service
@RequiredArgsConstructor
public class ArticleServiceImpl implements ArticleService {

```

```

private final ArticleRepo articleRepo;
private final UserRepo userRepo;

private final RabbitMQArticleProducer rabbitMQArticleProducer;
private static final Logger LOGGER =
LoggerFactory.getLogger(RabbitMQArticleProducer.class);

@Override
public Article findById(long artId) {
    return articleRepo.findById(artId).orElse(null);
}

@Override
public Article createArticle(ArticleCreatedDTO article, Long userId) {
    var newArticle = new Article();
    convertFromDTO(newArticle, article.getLanguage(), article.getTitle(),
article.getDescription(),
        article.getLink(), article.getSource(),
article.getCategory());
    newArticle.setArticleDate(Date.from(Instant.now()));
    newArticle.setUser(userRepo.findById(userId).orElse(null));
    newArticle = articleRepo.save(newArticle);
    LOGGER.info(String.format("New article in createArticle() -> %s",
newArticle.toString()));
    sendToTranslate(newArticle, article.getLanguage());
    return newArticle;
}

private void sendToTranslate(Article article, String language) {
    var articleToSend = new ArticleSendDTO();
    articleToSend.setArticle(article);
    articleToSend.setLanguage(language);
    rabbitMQArticleProducer.sendArticleEntityToTranslate(articleToSend);
}

private void convertFromDTO(Article newArticle, String language, String
title, String description,
                            String link, String source, String category)
{
    switch (language) {
        case "en":
            newArticle.setTitle_en(title);
            newArticle.setDescription_en(description);
            break;
        case "ua":
            newArticle.setTitle_ua(title);
            newArticle.setDescription_ua(description);
            break;
    }
    newArticle.setLink(link);
    newArticle.setSource(source);
    try {
        newArticle.getCategories().clear();
        newArticle.getCategories().add(Category.valueOf(category));
    } catch (NullPointerException e) {
        System.out.println("Category is null, setting other value");
        newArticle.getCategories().clear();
        newArticle.getCategories().add(Category.UNKNOWN);
    }
}

@Override

```

```

public void deleteById(long artId) {
    articleRepo.deleteById(artId);
}

@Override
public List<Article> findAll() {
    return articleRepo.findAll();
}

@Override
public List<Article> findAllByUserId(long userId) {
    return
articleRepo.findAllByUser(userRepo.findById(userId).orElse(null));
}

@Override
public Article updateArticle(ArticleEditDTO articleEditDTO) {
    var article =
articleRepo.findById(articleEditDTO.getId()).orElse(null);
    if (article == null) return null;
    convertFromDTO(article, articleEditDTO.getLanguage(),
articleEditDTO.getTitle(),
                    articleEditDTO.getDescription(), articleEditDTO.getLink(),
articleEditDTO.getSource(),
                    articleEditDTO.getCategory());
    article = articleRepo.save(article);
    sendToTranslate(article, articleEditDTO.getLanguage());
    return article;
}
}

```

Файл ArticleRepo.java

Реалізація функціональних вимог: перегляд власних статей, створення статті, редагування статті, видалення статті, перегляд статей присвоєних користувачу, пошук статті по ключовому слову, пошук статті по встановленим параметрам, перегляд статті.

```

@Repository
public interface ArticleRepo extends JpaRepository<Article, Long> {
    List<Article> findAllByUser(User user);

    @Query("SELECT a FROM Article a WHERE a.title_en = :titleEn")
    Article findByTitle_en(String titleEn);
}

```

Файл ArticleCreate.jsx

Реалізація функціональних вимог: створення статті.

```

const ArticleCreate = () => {
    const navigate = useNavigate();
    const { t, i18n } = useTranslation();

    const [article, setArticle] = useState({
        link: '',
        source: '',
        category: '',
        title: '',

```

```

        description: '',
        language: i18n.language
    });
    const [selectedCategory, setSelectedCategory] = useState('');
    const [selectedSource, setSelectedSource] = useState('');
    const [errors, setErrors] = useState({});

    const categories = t('articleList.allCategories', { returnObjects: true
});
    const sources = t('articleList.allSources', { returnObjects: true });

    const handleCategoryChange = (event) => {
        setSelectedCategory(event.target.value);
    };

    const handleSourceChange = (event) => {
        setSelectedSource(event.target.value);
    };

    const handleChange = (e) => {
        const { name, value } = e.target;
        setArticle(prevState => ({ ...prevState, [name]: value }));
    };

    const validate = () => {
        let tempErrors = {};
        tempErrors.link = article.link ? "" : t('articleCreate.error.link');
        tempErrors.category = selectedCategory ? "" :
t('articleCreate.error.category');
        tempErrors.source = selectedSource ? "" :
t('articleCreate.error.source');
        tempErrors.title = article.title ? "" :
t('articleCreate.error.title');
        tempErrors.description = article.description ? "" :
t('articleCreate.error.description');
        setErrors(tempErrors);
        return Object.values(tempErrors).every(x => x === "");
    };

    const handleCreateArticle = () => {
        if (validate()) {
            const newArticle = {
                link: article.link,
                title: article.title,
                description: article.description,
                category: selectedCategory,
                source: selectedSource,
                language: i18n.language
            };

            axios.post('http://localhost/api/articles', newArticle)
                .then(response => {
                    console.log('Article created successfully:',
response.data);
                    navigate('/articles');
                })
                .catch(error => {
                    console.error('Error creating article:', error);
                });
        }
    };

    return (
        <div>

```

```

<h2>{t('articleCreate.createArticle')}</h2>
<TextField
  label={t('articleCreate.link')}
  name="link"
  value={article.link}
  onChange={handleChange}
  fullWidth
  margin="dense"
  variant="outlined"
  error={!!errors.link}
  helperText={errors.link}
/>
<Select
  value={selectedCategory}
  onChange={handleCategoryChange}
  fullWidth
  margin="dense"
  variant="outlined"
  error={!!errors.category}
  displayEmpty
>
  <MenuItem
value="">{t('articleCreate.selectCategory')}</MenuItem>
    {Object.keys(categories).filter(categoryKey => categoryKey
!== 'ALL_CATEGORIES').map(categoryKey => (
      <MenuItem key={categoryKey}
value={categoryKey}>{categories[categoryKey]}</MenuItem>
    ))}
  </Select>
  <p style={{ color: 'red', margin: '4px 0'
}}>{errors.category}</p>
  <Select
    value={selectedSource}
    onChange={handleSourceChange}
    fullWidth
    margin="dense"
    variant="outlined"
    error={!!errors.source}
    displayEmpty
  >
    <MenuItem
value="">{t('articleCreate.selectSource')}</MenuItem>
      {Object.keys(sources).filter(sourceKey => sourceKey !==
'ALL_SOURCES').map(sourceKey => (
        <MenuItem key={sourceKey}
value={sourceKey}>{sources[sourceKey]}</MenuItem>
      ))}
    </Select>
    <p style={{ color: 'red', margin: '4px 0' }}>{errors.source}</p>
  <TextField
    label={t(`articleCreate.title.title_${i18n.language}`)}
    name="title"
    value={article.title}
    onChange={handleChange}
    fullWidth
    margin="dense"
    variant="outlined"
    error={!!errors.title}
    helperText={errors.title}
  />
  <TextField

label={t(`articleCreate.description.description_${i18n.language}`)}
  name="description"

```

```

        value={article.description}
        onChange={handleChange}
        fullWidth
        margin="dense"
        variant="outlined"
        multiline
        rows={4}
        error={!!errors.description}
        helperText={errors.description}
      />

      <Button onClick={handleCreateArticle} variant="contained"
color="primary">
        {t('articleCreate.create')}
      </Button>
    </div>
  );
};

export default ArticleCreate;

```

Файл ArticleEdit.jsx

Реалізація функціональних вимог: редагування статті.

```

const ArticleEdit = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const { t, i18n } = useTranslation();

  const [article, setArticle] = useState({
    id: '',
    link: '',
    source: '',
    category: '',
    title: '',
    description: '',
    language: ''
  });

  const [selectedCategory, setSelectedCategory] = useState('');
  const [selectedSource, setSelectedSource] = useState('');
  const [errors, setErrors] = useState({});

  const categories = t('articleList.allCategories', { returnObjects: true });
  const sources = t('articleList.allSources', { returnObjects: true });

  useEffect(() => {
    axios.get(`http://localhost/api/articles/${id}`)
      .then(response => {
        const fetchedArticle = response.data;
        setArticle({
          ...fetchedArticle,
          title: i18n.language === 'en' ? fetchedArticle.title_en :
fetchedArticle.title_ua,
          description: i18n.language === 'en' ?
fetchedArticle.description_en : fetchedArticle.description_ua
        });
        setSelectedCategory(fetchedArticle.category);
        setSelectedSource(fetchedArticle.source);
      })
      .catch(error => {
        console.error('Error fetching article details:', error);
      });
  });

```

```

    });
  }, [id, i18n.language]);

  const handleCategoryChange = (event) => {
    setSelectedCategory(event.target.value);
  };

  const handleSourceChange = (event) => {
    setSelectedSource(event.target.value);
  };

  const handleChange = (e) => {
    const { name, value } = e.target;
    setArticle(prevState => ({ ...prevState, [name]: value }));
  };

  const validate = () => {
    let tempErrors = {};
    tempErrors.link = article.link ? "" : t('articleEdit.error.link');
    tempErrors.category = selectedCategory ? "" :
t('articleEdit.error.category');
    tempErrors.source = selectedSource ? "" :
t('articleEdit.error.source');
    tempErrors.title = article.title ? "" : t('articleEdit.error.title');
    tempErrors.description = article.description ? "" :
t('articleEdit.error.description');
    setErrors(tempErrors);
    return Object.values(tempErrors).every(x => x === "");
  };

  const handleUpdateArticle = () => {
    if (validate()) {
      const updatedArticle = {
        id: article.id,
        link: article.link,
        title: article.title,
        description: article.description,
        category: selectedCategory,
        source: selectedSource,
        language: i18n.language
      };

      axios.put(`http://localhost/api/articles/${id}`, updatedArticle)
        .then(response => {
          console.log('Article updated successfully:',
response.data);
          navigate('/articles');
        })
        .catch(error => {
          console.error('Error updating article:', error);
        });
    }
  };

  return (
    <div>
      <h2>{t('articleEdit.editArticle')}</h2>
      <TextField
        label={t('articleEdit.link')}
        name="link"
        value={article.link}
        onChange={handleChange}
        fullWidth
        margin="dense"

```

```

        variant="outlined"
        error={!errors.link}
        helperText={errors.link}
      />
    <Select
      value={selectedCategory}
      onChange={handleCategoryChange}
      fullWidth
      margin="dense"
      variant="outlined"
      error={!errors.category}
      displayEmpty
    >
      <MenuItem
value="">{t('articleEdit.selectCategory')}</MenuItem>
      {Object.keys(categories).map(categoryKey => (
        <MenuItem key={categoryKey}
value={categoryKey}>{categories[categoryKey]}</MenuItem>
      ))}
    </Select>
    <p style={{ color: 'red', margin: '4px 0' }}>{errors.category}</p>
    <Select
      value={selectedSource}
      onChange={handleSourceChange}
      fullWidth
      margin="dense"
      variant="outlined"
      error={!errors.source}
      displayEmpty
    >
      <MenuItem value="">{t('articleEdit.selectSource')}</MenuItem>
      {Object.keys(sources).map(sourceKey => (
        <MenuItem key={sourceKey}
value={sourceKey}>{sources[sourceKey]}</MenuItem>
      ))}
    </Select>
    <p style={{ color: 'red', margin: '4px 0' }}>{errors.source}</p>
    <TextField
      label={t(`articleEdit.title.${i18n.language === 'en' ?
'title_en' : 'title_ua'}`)}
      name="title"
      value={article.title}
      onChange={handleChange}
      fullWidth
      margin="dense"
      variant="outlined"
      error={!errors.title}
      helperText={errors.title}
    />
    <TextField
      label={t(`articleEdit.description.${i18n.language === 'en' ?
'description_en' : 'description_ua'}`)}
      name="description"
      value={article.description}
      onChange={handleChange}
      fullWidth
      margin="dense"
      variant="outlined"
      multiline
      rows={4}
      error={!errors.description}
      helperText={errors.description}
    />

```

```

        <Button onClick={handleUpdateArticle} variant="contained"
color="primary">
        {t('articleEdit.update')}
    </Button>
</div>
    );
};

export default ArticleEdit;

```

Файл ArticleList.jsx

Реалізація функціональних вимог: пошук статті по ключовому слову, пошук статті по встановленим параметрам, перегляд статті.

```

const ArticleList = () => {
    const { t, i18n } = useTranslation();
    const [articles, setArticles] = useState([]);
    const [filteredArticles, setFilteredArticles] = useState([]);
    const navigate = useNavigate();

    useEffect(() => {
        axios.get('http://localhost/api/articles')
            .then(response => {
                const sortedArticles = response.data.sort((a, b) => new
Date(b.articleDate) - new Date(a.articleDate));
                setArticles(sortedArticles);
                setFilteredArticles(sortedArticles);
            })
            .catch(error => {
                console.error('Error fetching articles:', error);
            });
    }, []);

    const handleSearch = (searchParams) => {
        const { searchTerm, startDate, endDate, selectedCategory,
selectedSource } = searchParams;
        let filtered = articles.filter(article => {
            if (!article || !article.title_en || !article.description_en ||
!article.articleDate || !article.categories || !article.source) {
                return false;
            }
            const matchesSearchTerm =
article.title_en.toLowerCase().includes(searchTerm.toLowerCase()) ||
article.description_en.toLowerCase().includes(searchTerm.toLowerCase());
            const articleDate = new Date(article.articleDate);
            const isWithinDateRange = (!startDate || articleDate >= new
Date(startDate)) && (!endDate || articleDate <= new Date(endDate));
            const matchesCategory = !selectedCategory || selectedCategory ===
"ALL_CATEGORIES" || article.categories.includes(selectedCategory);
            const matchesSource = !selectedSource || selectedSource ===
"ALL_SOURCES" || article.source === selectedSource;
            return matchesSearchTerm && isWithinDateRange && matchesCategory
&& matchesSource;
        });
        setFilteredArticles(filtered.sort((a, b) => new Date(b.articleDate) -
new Date(a.articleDate)));
    };

    const handleViewMore = (id) => {
        navigate(`/articles/${id}`);
    };

```

```

return (
  <div className="container">
    <h1 className="mb-4">{t('articleList.title')}</h1>
    <SearchBar onSearch={handleSearch} />
    <div className="row">
      {filteredArticles.length > 0 ? (
        filteredArticles.map(article => (
          <div className="col-md-4 mb-4" key={article.id}>
            <div className="card">
              <div className="card-body">
                <h5 className="card-title">{i18n.language
=== 'en' ? article.title_en : article.title_ua}</h5>
                <p className="card-text">{i18n.language
=== 'en' ? article.description_en : article.description_ua}</p>
                <p className="card-
text">{t('articleList.source')}: {article.source}</p>
                <p className="card-
text">{t('articleList.date')}: {new
Date(article.articleDate).toLocaleString()}</p>
                <button className="btn btn-secondary"
onClick={() =>
handleViewMore(article.id)}>{t('articleList.viewMore')}</button>
              </div>
            </div>
          </div>
        ))
      ) : (
        <div className="col-12">
          <p className="text-center mt-
4">{t('articleList.noArticles')}</p>
        </div>
      )}
    </div>
  </div>
);
};

export default ArticleList;

```

Файл ArticleView.jsx

Реалізація функціональних вимог: видалення статті, перегляд статті.

```

import React, {useEffect, useState} from 'react';
import {Link, useNavigate, useParams} from 'react-router-dom';
import axios from 'axios';
import {useCookies} from 'react-cookie';
import {useTranslation} from 'react-i18next';
import {Button, Container, Typography} from '@mui/material';

const ArticleView = () => {
  const {id} = useParams();
  const [article, setArticle] = useState(null);
  const [user, setUser] = useState(null);
  const [cookies] = useCookies(['USER_ID']);
  const navigate = useNavigate();
  const {t, i18n} = useTranslation();

  useEffect(() => {
    axios.get(`http://localhost/api/articles/${id}`)
      .then(response => {
        setArticle(response.data);
      })
  })

```

```

        .catch(error => {
            console.error('Error fetching article details:', error);
        });

        if (cookies.USER_ID) {
            axios.get(`http://localhost/api/users/${cookies.USER_ID}`)
                .then(response => {
                    setUser(response.data);
                })
                .catch(error => {
                    console.error('Error fetching user details:', error);
                });
        }
    }, [id, cookies.USER_ID]);

    const handleDelete = () => {
        axios.delete(`http://localhost/api/articles/${id}`)
            .then(response => {
                console.log('Article deleted successfully:', response.data);
                navigate('/articles');
            })
            .catch(error => {
                console.error('Error deleting article:', error);
            });
    };

    if (!article || !user) {
        return <Typography>Loading...</Typography>;
    }

    const isAuthor = article.user.id === parseInt(cookies.USER_ID);
    const isAdmin = user.roles.includes('ROLE_ADMIN');

    return (
        <Container>
            <Typography variant="h2">{i18n.language === 'en' ?
article.title_en : article.title_ua}</Typography>
            <Typography
                variant="body1">{i18n.language === 'en' ?
article.description_en : article.description_ua}</Typography>
            <Typography variant="body2">{t('articleList.source')}:
{article.source}</Typography>
            <Typography
                variant="body2">{t('articleList.date')}: {new
Date(article.articleDate).toLocaleString()}</Typography>
            <Typography variant="body2">{t('articleList.categories')}:
{article.categories.join(', ')}</Typography>
            <Typography variant="body2">
                {t('articleList.author')}: <Link
to={`/profile/${article.user.id}`}>{article.user.username}</Link>
            </Typography>

            {(isAdmin || isAuthor) && (
                <div>
                    <Button
                        component={Link}
                        to={`/articles/${article.id}/edit`}
                        variant="contained"
                        color="primary"
                        style={{marginRight: '10px'}}
                    >
                        {t('articleList.edit')}
                    </Button>
                    <Button

```

```

        onClick={handleDelete}
        variant="contained"
        color="secondary"
      >
        {t('articleList.delete')}
      </Button>
    </div>
  )}
</Container>
);
};

export default ArticleView;

```

Файл SearchBar.jsx

Реалізація функціональних вимог: пошук статті по ключовому слову, пошук статті по встановленим параметрам.

```

const SearchBar = ({ onSearch }) => {
  const { t } = useTranslation();
  const [searchTerm, setSearchTerm] = useState('');
  const [startDate, setStartDate] = useState('');
  const [endDate, setEndDate] = useState('');
  const [selectedCategory, setSelectedCategory] = useState('');
  const [selectedSource, setSelectedSource] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    onSearch({
      searchTerm,
      startDate,
      endDate,
      selectedCategory,
      selectedSource
    });
  };

  const allCategories = t('articleList.allCategories', { returnObjects: true });
  const allSources = t('articleList.allSources', { returnObjects: true });

  return (
    <form onSubmit={handleSubmit} className="form-inline mb-4">
      <input type="text" className="form-control mr-2"
        placeholder={t('articleList.searchPlaceholder')} value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)} />
      <input type="date" className="form-control mr-2"
        value={startDate} onChange={(e) => setStartDate(e.target.value)} />
      <input type="date" className="form-control mr-2" value={endDate}
        onChange={(e) => setEndDate(e.target.value)} />
      <select className="form-control mr-2" value={selectedCategory}
        onChange={(e) => setSelectedCategory(e.target.value)}>
        {Object.keys(allCategories).map(categoryKey => (
          <option key={categoryKey}
            value={categoryKey}>{allCategories[categoryKey]}</option>
        ))}
      </select>
      <select className="form-control mr-2" value={selectedSource}
        onChange={(e) => setSelectedSource(e.target.value)}>
        {Object.keys(allSources).map(sourceKey => (
          <option key={sourceKey}
            value={sourceKey}>{allSources[sourceKey]}</option>

```

```

    ))}
  </select>
  <button type="submit" className="btn btn-
primary">{t('articleList.search')}</button>
</form>
);
};

export default SearchBar;

```

Файл LanguageSwitcher.jsx

Реалізація функціональних вимог: зміна мови.

```

const LanguageSwitcher = () => {
  const { i18n } = useTranslation();

  const changeLanguage = (lng) => {
    i18n.changeLanguage(lng);
  };

  return (
    <div className="btn-group" role="group" aria-label="Language
Switcher">
      <button type="button" className={ `btn ${i18n.language === 'en' ?
'btn-primary' : 'btn-secondary'} ` } onClick={() =>
changeLanguage('en')}>English</button>
      <button type="button" className={ `btn ${i18n.language === 'ua' ?
'btn-primary' : 'btn-secondary'} ` } onClick={() =>
changeLanguage('ua')}>Українська</button>
    </div>
  );
};

export default LanguageSwitcher;

```

Файл ArticleRssParser.java

Реалізація функціональних вимог: автоматичне завантаження статей з RSS-feed сторонніх ресурсів, автоматичний переклад статті, автоматичне визначення категорій статей зі сторонніх ресурсів.

```

@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ArticleRssParser implements Parser<ArrayList<Article>> {
  private final URLConnector urlConnector;

  private final TranslateAPIParser translateAPIParser;

  @Override
  public ArrayList<Article> doParse(String resource) throws IOException,
FeedException, ParseException {
    HttpURLConnection connection = urlConnector.setConnection(resource);
    ArrayList<Article> list = new ArrayList<>();
    try {
      SyndFeedInput input = new SyndFeedInput();
      SyndFeed feed = input.build(new XmlReader(connection));
      List entries = feed.getEntries();
    }
  }
}

```

```

        Iterator itEntries = entries.iterator();

        while (itEntries.hasNext()) {
            SyndEntry entry = (SyndEntry) itEntries.next();
            Article article = new Article();
            String language =
translateAPIParser.detectLanguage(entry.getTitle());
            switch (language) {
                case "uk" -> setUkArticle(article, entry);
                case "en" -> setEnArticle(article, entry);
            }
            list.add(article);
        }
    } finally {
        urlConnector.endConnection(connection);
    }
    return list;
}

private void setUkArticle(Article article, SyndEntry entry) {
    try {
        article.setTitle_ua(entry.getTitle());
    } catch (NullPointerException e) {
        article.setTitle_ua("Помилка заголовку");
    }
    article.setLink(entry.getLink());
    if (entry.getAuthor() != null) article.setSource(entry.getAuthor());
    try {
        if (!entry.getDescription().getValue().isEmpty())
            article.setDescription_ua(entry.getDescription().getValue());
        else
            article.setDescription_ua("Немає опису, перегляньте
оригінальну статтю для отримання додаткової інформації");
    } catch (NullPointerException e) {
        article.setDescription_ua("Немає опису, перегляньте оригінальну
статтю для отримання додаткової інформації");
    }
    if (entry.getPublishedDate() != null)
        article.setArticleDate(entry.getPublishedDate());
    else article.setArticleDate(Date.from(Instant.now()));
}

private void setEnArticle(Article article, SyndEntry entry) {
    try {
        article.setTitle_en(entry.getTitle());
    } catch (NullPointerException e) {
        article.setTitle_en("Tittle Error");
    }
    article.setLink(entry.getLink());
    if (entry.getAuthor() != null) {
        article.setSource(entry.getAuthor());
    }
    try {
        if (!entry.getDescription().getValue().isEmpty()) {
            article.setDescription_en(entry.getDescription().getValue());
        } else {
            article.setDescription_en("No description, view original
article for additional information");
        }
    } catch (NullPointerException e) {
        article.setDescription_en("No description, view original article
for additional information");
    }
}

```

```
        if (entry.getPublishedDate() != null)
article.setArticleDate(entry.getPublishedDate());
        else article.setArticleDate(Date.from(Instant.now()));
    }
}
```

Файл ArticleParser.java

Реалізація функціональних вимог: автоматичне завантаження статей з RSS-feed сторонніх ресурсів, автоматичний переклад статті, автоматичне визначення категорій статей зі сторонніх ресурсів.

```
@Service
@RequiredArgsConstructor
public class ArticleParser {
    private final ArticleRssParser articleRssParser;
    private final TranslateAPIParser translateAPIParser;

    // Constants

    @Value("${id.pravda}")
    private Long PRAVDA_ID;
    @Value("${id.cnn}")
    private Long CNN_ID;
    @Value("${id.fox}")
    private Long FOX_ID;
    @Value("${id.unian}")
    private Long UNIAN_ID;
    private final APIConnector apiConnector;

    @Value("${category.url}")
    public String categoryUrl;

    private final RabbitMQArticleProducer rabbitMQArticleProducer;

    public void parseArticle(String link) throws FeedException, IOException,
    ParseException {
        ArrayList<Article> listFromRss = articleRssParser.doParse(link);
        for (Article articleRss : listFromRss) {
            final String PRAVDA_LINK = "https://www.pravda.com.ua/rss/";
            final String CNN_LINK =
"http://rss.cnn.com/rss/cnn_topstories.rss";
            final String FOX_LINK = "https://moxie.foxnews.com/google-
publisher/world.xml";
            final String UNIAN_LINK =
"https://rss.unian.net/site/news_ukr.rss";
            switch (link) {
                case PRAVDA_LINK -> parseAssist(articleRss, "PRAVDA",
PRAVDA_ID);
                case CNN_LINK -> parseAssist(articleRss, "CNN", CNN_ID);
                case FOX_LINK -> parseAssist(articleRss, "FOX NEWS", FOX_ID);
                case UNIAN_LINK -> parseAssist(articleRss, "VHIAH",
UNIAN_ID);
            }
            if (isArticleValid(articleRss)) {
                articleRss.getCategories()
                    .add(consumeCategory(articleRss.getTitle_en() +
articleRss.getDescription_en()));
                rabbitMQArticleProducer.sendArticleEntity(articleRss);
                System.out.println(articleRss);
            } else {
                System.out.println("Skipping article due to null/empty
fields: " + articleRss);
            }
        }
    }

    private boolean isArticleValid(Article article) {
```

```

        return article.getArticleDate() != null &&
            article.getTitle_en() != null &&
!article.getTitle_en().isEmpty() &&
            article.getDescription_en() != null &&
!article.getDescription_en().isEmpty();
    }

    public Article.Category consumeCategory(String text) throws IOException {
        HttpURLConnection conn = apiConnector.setConnection(categoryUrl +
"predict");
        try {
            String jsonString = "{\"text\":\"" + "\"" + text.replace("\"",
"\\\"")
                + "\"}";
            return getCategory(conn, jsonString);
        } finally {
            apiConnector.endConnection(conn);
        }
    }

    private Article.Category getCategory(HttpURLConnection connection, String
jsonInputString) throws IOException {
        try (OutputStream os = connection.getOutputStream()) {
            byte[] input = jsonInputString.getBytes("utf-8");
            os.write(input, 0, input.length);
        }

        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
                StringBuilder response = new StringBuilder();
                String inputLine;
                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }
                JSONObject jsonResponse = new
JSONObject(response.toString());
                String categoryString = jsonResponse.getString("category");
                return
Article.Category.valueOf(categoryString.toUpperCase());
            }
        } else {
            System.out.println("HTTP POST request failed with response code:
" + responseCode);
            return Article.Category.UNKNOWN;
        }
    }

    private void parseAssist(Article articleRss, String source, Long id)
throws IOException, ParseException {
        try {
            articleRss.setUserId(id);
            if (articleRss.getSource().isEmpty())
articleRss.setSource(source);
            try {
                if (articleRss.getTitle_en().isEmpty()) {
                    addTranslation(articleRss, "uk", "en");
                } else addTranslation(articleRss, "en", "uk");
            } catch (NullPointerException e) {
                addTranslation(articleRss, "uk", "en");
            }
        } catch (NullPointerException e) {
            System.err.println("Error in parseAssist");
        }
    }

```

```

    }
}

private void addTranslation(Article article, String sourceLang, String
targetLang) throws IOException, ParseException {
    switch (sourceLang) {
        case "uk" -> {

article.setTitle_en(translateAPIParser.doParse(article.getTitle_ua(),
sourceLang, targetLang));

article.setDescription_en(translateAPIParser.doParse(article.getDescription_u
a(), sourceLang, targetLang));
        }
        case "en" -> {

article.setTitle_ua(translateAPIParser.doParse(article.getTitle_en(),
sourceLang, targetLang));

article.setDescription_ua(translateAPIParser.doParse(article.getDescription_e
n(), sourceLang, targetLang));
        }
    }
}
}
}

```

Файл URLConnector.java

Реалізація функціональних вимог: автоматичне завантаження статей з RSS-feed сторонніх ресурсів, автоматичний переклад статті, автоматичне визначення категорій статей зі сторонніх ресурсів.

```

@Service
public class URLConnector implements Connector {
    @Override
    public HttpURLConnection setConnection(String url) throws IOException {
        URL toConnect = new URL(url);
        return (HttpURLConnection) toConnect.openConnection();
    }

    @Override
    public void endConnection(HttpURLConnection httpURLConnection) {
        httpURLConnection.disconnect();
    }
}

```

Файл APIConnector.java

Реалізація функціональних вимог: автоматичний переклад статті, автоматичне визначення категорій статей зі сторонніх ресурсів.

```

@Service
public class APIConnector implements Connector {

    @Override
    public HttpURLConnection setConnection(String pageUrl) throws IOException

```

```

{
    URL url = new URL(pageUrl);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Content-Type", "application/json");
    conn.setRequestProperty("Accept", "application/json");
    conn.setDoOutput(true);
    return conn;
}

@Override
public void endConnection(HttpURLConnection httpURLConnection) {
    httpURLConnection.disconnect();
}
}

```

Файл TranslateAPIParser.java

Реалізація функціональних вимог: автоматичний переклад статті.

```

@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class TranslateAPIParser implements Parser<String> {

    private final APIConnector apiConnector;

    @Value("${translate.url}")
    public String translateUrl;

    @Override
    public String doParse(String textToTranslate) throws IOException,
    ParseException {
        HttpURLConnection conn = apiConnector.setConnection(translateUrl +
        "translate");
        try {
            String jsonString = "{\"q\":\"" + "\"" +
            textToTranslate.replace("\"", "\\\"") + "\"" +
            + "\", \"source\": \"uk\", \"" +
            + "\"target\": \"en\", \"" +
            + "\"format\": \"text\"}";
            return getString(conn, jsonString, "translatedText");
        } finally {
            apiConnector.endConnection(conn);
        }
    }

    public String doParse(String textToTranslate, String source, String
    target) throws IOException, ParseException {
        HttpURLConnection conn = apiConnector.setConnection(translateUrl +
        "translate");
        try {
            String jsonString = "{\"q\":\"" + "\"" +
            textToTranslate.replace("\"", "\\\"")
            .replace("\r\n", ",")
            .replace("\n", ",") + "\"" +
            + "\", \"source\": \"" + source.replace("\"", "\\\"") +
            "\"\", \"" +
            + "\"target\": \"" + target.replace("\"", "\\\"") + "\", \"" +
            + "\"format\": \"text\"}";
            return getString(conn, jsonString, "translatedText");
        } finally {
            apiConnector.endConnection(conn);
        }
    }
}

```

```

    }
}

private String getString(URLConnection conn, String jsonString,
String objToSearch) throws IOException, ParseException {
    try (OutputStream os = conn.getOutputStream()) {
        byte[] input = jsonString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
    }
    StringBuilder inline = new StringBuilder();
    Scanner scanner = new Scanner(conn.getInputStream());
    while (scanner.hasNext()) {
        inline.append(scanner.nextLine());
    }
    scanner.close();
    JSONParser parser = new JSONParser();
    if (parser.parse(inline.toString()).getClass() == JSONObject.class) {
        JSONObject data_obj = (JSONObject)
parser.parse(inline.toString());
        return (String) data_obj.get(objToSearch);
    } else {
        JSONArray jsonArray = (JSONArray)
parser.parse(inline.toString());
        String lang = "";
        for (Object o : jsonArray) {
            JSONObject jsonObject = (JSONObject) o;
            lang = (String) jsonObject.get("language");
        }
        return lang;
    }
}

public String detectLanguage(String text) throws IOException,
ParseException {
    HttpURLConnection conn = apiConnector.setConnection(translateUrl +
"detect");
    try {
        String jsonString = "{\"q\":\"" + text.replace("\"",
"\\\"") + "\"}";
        return getString(conn, jsonString, "language");
    } finally {
        apiConnector.endConnection(conn);
    }
}
}

```

Файл app.py

Реалізація функціональних вимог: автоматичне визначення категорій статей зі сторонніх ресурсів.

```

from flask import Flask, request, jsonify
import pickle

app = Flask(__name__)
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

label_to_category = {
    0: "U_S_NEWS",

```

```

1: "COMEDY",
2: "PARENTING",
3: "WORLD_NEWS",
4: "CULTURE_AND_ARTS",
5: "TECH",
6: "SPORTS",
7: "ENTERTAINMENT",
8: "POLITICS",
9: "WEIRD_NEWS",
10: "ENVIRONMENT",
11: "EDUCATION",
12: "CRIME",
13: "SCIENCE",
14: "WELLNESS",
15: "BUSINESS",
16: "STYLE_AND_BEAUTY",
17: "FOOD_AND_DRINK",
18: "MEDIA",
19: "QUEER_VOICES",
20: "HOME_AND_LIVING",
21: "WOMEN",
22: "BLACK_VOICES",
23: "GEOPOLITICS",
24: "MONEY",
25: "RELIGION",
26: "LATINO_VOICES",
27: "IMPACT",
28: "WEDDINGS",
29: "COLLEGE",
30: "PARENTS",
31: "ARTS_AND_CULTURE",
32: "STYLE",
33: "GREEN",
34: "TASTE",
35: "HEALTHY_LIVING",
36: "THE_WORLDPOST",
37: "GOOD_NEWS",
38: "WORLDPOST",
39: "FIFTY",
40: "ARTS",
41: "DIVORCE"
}

@app.route('/api/category-ai/predict', methods=['POST'])
def predict_category():
    data = request.get_json()
    text = data['text']
    category_label = model.predict([text])[0] - 1
    category_name = label_to_category.get(category_label, "Unknown")
    return jsonify({'category': category_name})

if __name__ == '__main__':
    app.run(debug=True)

```

Файл ArticleScheduler.java

Реалізація функціональних вимог: автоматичне завантаження статей з RSS-feed сторонніх ресурсів, автоматичне оновлення списку новин зі сторонніх ресурсів кожні 30 хвилин.

```
@Service
@RequiredArgsConstructor
public class ArticleScheduler {
    private static final String[] RESOURCES_LIST = {
        "https://www.pravda.com.ua/rss/",
        "http://rss.cnn.com/rss/cnn_topstories.rss",
        "https://moxie.foxnews.com/google-publisher/world.xml",
        "https://rss.unian.net/site/news_ukr.rss";
        // https://nv.ua/ukr/rss/all.xml - alternative
    }

    private final ArticleParser articleParser;

    @Scheduled(initialDelay = 10000, fixedDelayString = "PT30M") // on start
    and every 30 minutes
    public void updateArticlesPeriodically() {
        try {
            updateArticlesFromResources();
        } catch (FeedException | IOException | ParseException e) {
            System.err.println("Failed to update articles: " +
                e.getMessage());
        }
    }

    private void updateArticlesFromResources() throws FeedException,
    IOException, ParseException {
        for (String resource : RESOURCES_LIST) {
            articleParser.parseArticle(resource);
            System.out.println("Update articles from: " + resource);
        }
    }
}
```

Файл RabbitMQArticleConsumer.java

Реалізація функціональних вимог: автоматичне завантаження статей з RSS-feed сторонніх ресурсів, створення статті, редагування статті, автоматичний переклад статті.

```
@Service
@RequiredArgsConstructor
public class RabbitMQArticleConsumer {
    private static final Logger LOGGER =
    LoggerFactory.getLogger(RabbitMQArticleConsumer.class);
    private final TranslateAPIParser translateAPIParser;
    private final RabbitMQArticleProducer rabbitMQArticleProducer;

    @RabbitListener(queues = {"${rabbitmq.parsing.queue.name}"})
    public void consume(ArticleSendDTO articleSendDTO) throws IOException,
    ParseException {
        LOGGER.info(String.format("Received article to translate -> %s",
            articleSendDTO.getArticle().toString()));
        switch (articleSendDTO.getLanguage()) {
            case "en":
                LOGGER.info("Article is in English. Translating to
                Ukrainian.");
            default:
                // ...
        }
    }
}
```

```

        var title_ua = translateAPIParser
            .doParse(articleSendDTO.getArticle().getTitle_en(),
"en", "uk");
        LOGGER.info("Translated title: " + title_ua);
        articleSendDTO.getArticle()
            .setTitle_ua(title_ua);
        var description_ua = translateAPIParser
            .doParse(articleSendDTO.getArticle().getDescription_en(), "en", "uk");
        LOGGER.info("Translated description: " + description_ua);

        articleSendDTO.getArticle().setDescription_ua(description_ua);
        LOGGER.info("Translated article: " +
        articleSendDTO.getArticle().toString());

        rabbitMQArticleProducer.sendTranslatedArticle(articleSendDTO.getArticle());
        break;
    case "ua":
        LOGGER.info("Article is in Ukrainian. Translating to
English.");
        var title_en = translateAPIParser
            .doParse(articleSendDTO.getArticle().getTitle_ua(),
"uk", "en");
        LOGGER.info("Translated title: " + title_en);
        articleSendDTO.getArticle()
            .setTitle_en(title_en);
        var description_en = translateAPIParser
            .doParse(articleSendDTO.getArticle().getDescription_ua(), "uk", "en");
        LOGGER.info("Translated description: " + description_en);

        articleSendDTO.getArticle().setDescription_en(description_en);
        LOGGER.info("Translated article: " +
        articleSendDTO.getArticle().toString());

        rabbitMQArticleProducer.sendTranslatedArticle(articleSendDTO.getArticle());
        break;
    }
}
}

```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2024 р.

Вебзастосунок для агрегації та автоматичного перекладу новин

Програма та методика тестування

КПІ.IT-0325.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена СИРОТА

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Антон ЧАБАН

Київ – 2024

ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є веб-застосунок агрегації та автоматичного перекладу новин «MultiNews». Тестуванню підлягають функціональні вимоги та графічний інтерфейс. Проведено тестування клієнтської та серверної частини програмного забезпечення.

Веб-застосунок сумісний із будь-якою операційною системою та із будь-яким веб-браузером. Тестування програмного забезпечення здійснене у веб-браузері Microsoft Edge (версія 118.0), а його розробка на операційній системі Windows 11.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Edge, Chrome, Opera, Firefox);
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об'єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «сірої скриньки» – об'єктом тестування тут є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Також перевірка системи відбувається за рахунок статичного аналізатора коду, для перевірки на дотримання норм програмування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- тестування на відповідність функціональним вимогам;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування вихідного коду на предмет дотримання стандартів програмування;
- тестування інтерфейсу користувача;
- тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2024 р.

Вебзастосунок для агрегації та автоматичного перекладу новин

Керівництво користувача

КПІ.IT-0325.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена СИРОТА

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Антон ЧАБАН

Київ – 2024

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ	5

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«MultiNews» - це веб-застосунок для агрегації та автоматичного перекладу новин, а також з можливістю додавання власних. Застосунок створено з метою підвищення зручності та полегшення роботи з новинами для різномовних користувачів, отримання інформації з останніми змінами та їх перегляду за категоріями.

Функціонал веб-застосунку:

- логін та реєстрація користувачів;
- створення/редагування/видалення статті та її автоматичний переклад;
- статті користувачів;
- пошук статті за ресурсом/датою/ключовим словом/категорією;
- зчитування новин з встановлених ресурсів, їх автоматичний переклад та визначення теми.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність веб-браузеру Edge або Chrome з версіями 90.0 або вище, Firefox з версією 78.0 та вище, Safari з версією 14.0 або вище;
- наявність доступу до Інтернету;

2.2 Завантаження застосунку

Програмне забезпечення є веб-застосунком та не потребує завантаження з боку користувача.

2.3 Перевірка коректної роботи

Користувачу необхідно ввести адресу застосунку в браузері, після чого йому стане доступний базовий функціонал гостя, з яким можна переглядати новини. Користувач може авторизуватися в системі, для отримання інших прав.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску веб-застосунку користувачу буде відображено головну сторінку з останніми новинами, тут можуть бути як статті отримані з новинних ресурсів, так і написані редакторами системи (рисунки 3.1).

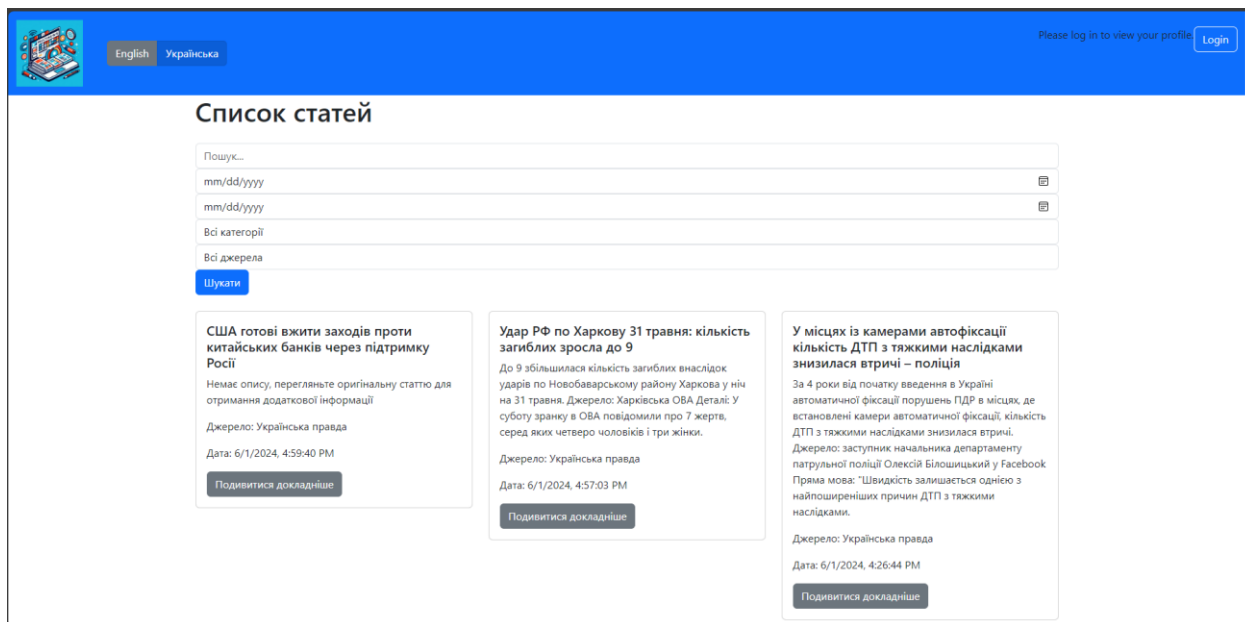


Рисунок 3.1 – Головна сторінка

Користувач може увійти в систему, натиснувши кнопку Login. Йому буде відображено два поля для вводу логіну та паролю (рисунки 3.2). Якщо користувач зареєстрований – він заповнює ці поля та входить до системи.

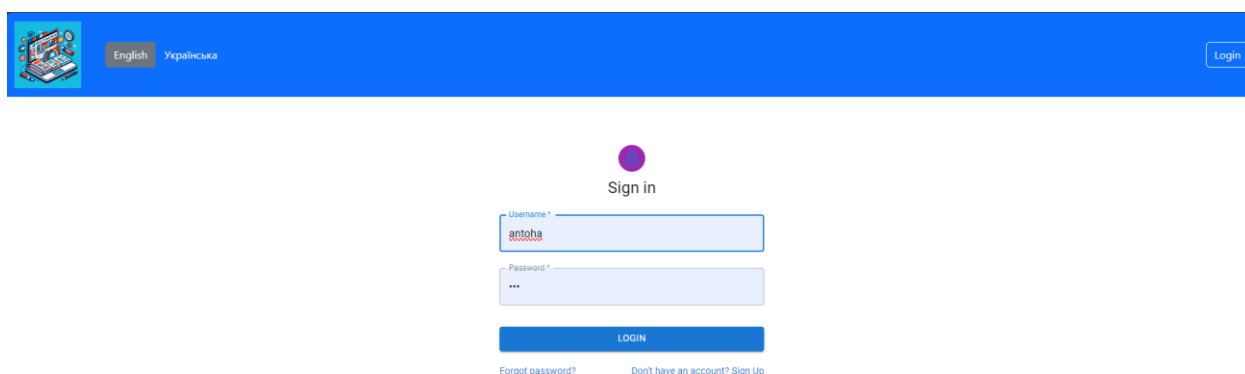


Рисунок 3.2 – Вхід в систему

У випадку, коли користувач не зареєстрований, йому треба натиснути кнопку Sign Up/Зареєструватися. На екрані реєстрації (рисунок 3.3) користувач вводить логін та пароль.

Рисунок 3.3 – Реєстрація у систему

Щоб додати статтю користувач натискає кнопку Create Article, після чого відкриваються сторінка для введення відповідних даних (рисунок 3.4).

Рисунок 3.4 – Додавання статті

Далі користувач заповнює всі поля форми (рисунок 3.5) та натискає «Create». Після цього користувач перенаправляється на головну сторінку, де можна побачити щойно створену статтю (рисунок 3.6).

English Українська

Welcome, antohai [Create Article](#) [Profile](#) [LOGOUT](#)

Create Article

Link
chess.com

Tech

Other users

Title (EN)
Simple article title

Description (EN)
Simple article description

[CREATE](#)

Рисунок 3.5 – Заповнення полів

Article List

Search...

mm/dd/yyyy

mm/dd/yyyy

All Categories

All Sources

[Search](#)

Simple article title
Simple article description

Source: Other users

Date: 6/3/2024, 10:58:49 PM

[View More](#)

The United States is ready to take action against Chinese banks through support of Russia

No description, revisit the original article to obtain additional information

Source: Українська правда

Date: 6/1/2024, 4:59:40 PM

[View More](#)

The attack of the Russian Federation on 31 May: the number of deaths increased to 9

By 9, the number of killed were increased due to strikes in the Novobavard district of Kharkiv on the night of 31 May. , Source: Kharkiv OVA, Details: On Saturday morning, OVA reported 7 casualties, including four men and three women.

Source: Українська правда

Date: 6/1/2024, 4:57:03 PM

[View More](#)

In places with chambers, the number of DTPs with severe consequences decreased three times – police

Multi-promising step: scientists have created a " cheap and fast" test for prostate cancer

Zelensky: Aid from the United States now had little assistance

Рисунок 3.6 – Сторінка останніх новин із доданою статтею

Розглянемо ситуацію, коли користувачу треба відредагувати текст статті (рисунок 3.7). Натискаємо «Edit», переходимо на сторінку редагування і бачимо, наші поля, в які користувач може внести правки (рисунок 3.8).

Simple article title that was updated

Simple article description that was updated

Source: Other users

Date: 5/29/2024, 10:11:49 PM

Categories: SPORTS

Author: antohai

[EDIT](#) [DELETE](#)

Рисунок 3.7 – Сторінка новини з можливістю редагування

Рисунок 3.8 – Сторінка редагування новини

Користувач знаходиться на сторінці з новинами (рисунок 3.9), йому треба виконати пошук за певними параметрами. Заповнюємо поля та виконуємо пошук натиснувши кнопку Search (рисунок 3.10).

Рисунок 3.9 – Сторінка з параметрами пошуку

Рисунок 3.10 – Результати пошуку за параметрами

Користувач може змінити мову сайту натиснувши на потрібну мову (рисунок 3.11).

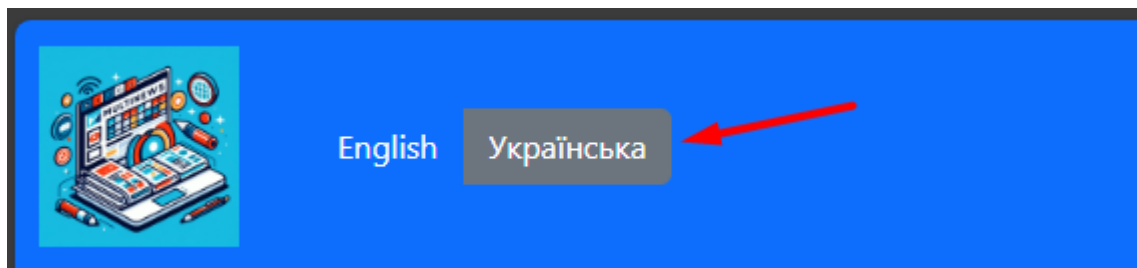


Рисунок 3.11 – Вибір мови сайту

Змінивши мову, можемо побачити останні новини обраною мовою (рисунок 3.12).

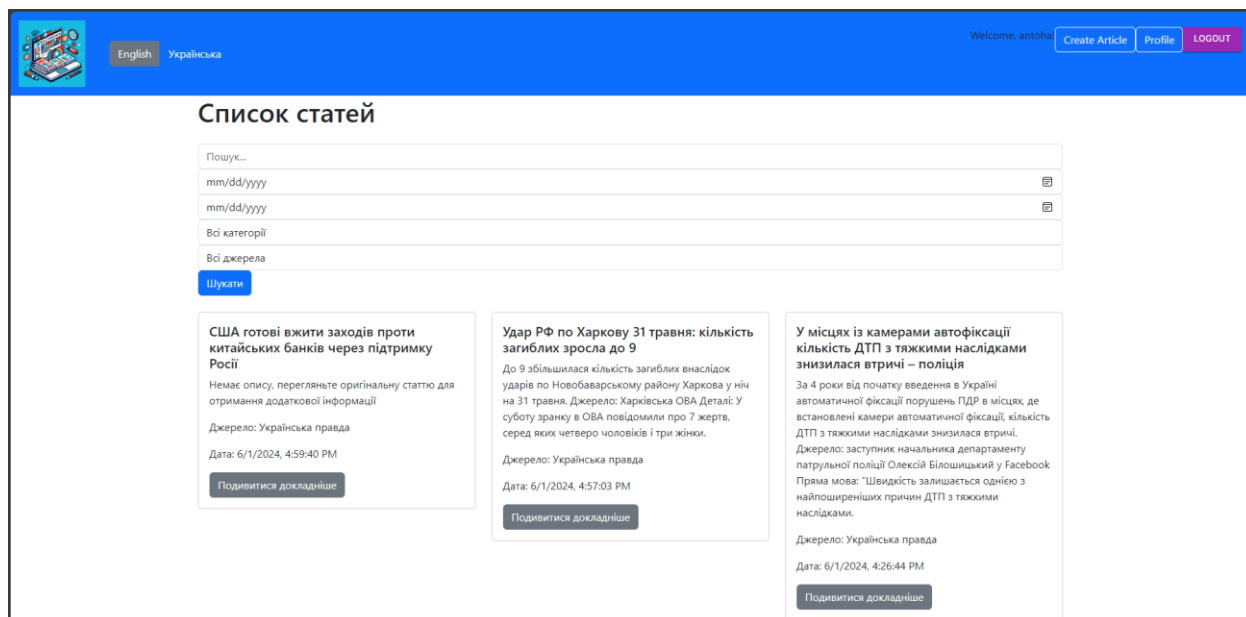


Рисунок 3.12 – Відображення сайту обраною мовою

Якщо користувач є адміністратором, на сторінці користувача стає видимою кнопка панелі адміністратора (рисунок 3.13).

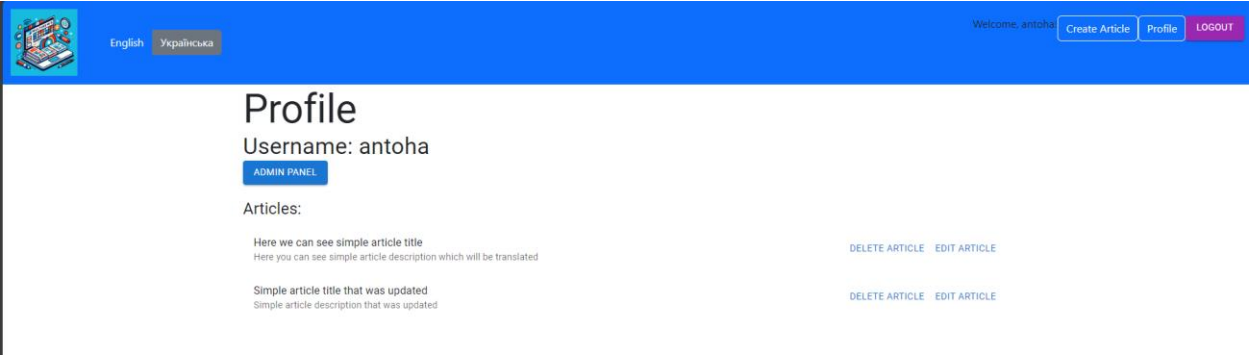


Рисунок 3.13 – Профіль адміністратора

Розглянемо панель адміністратора, тут користувач може видаляти/редагувати ролі інших користувачів, а також перейти в їх профіль (рисунок 3.14).

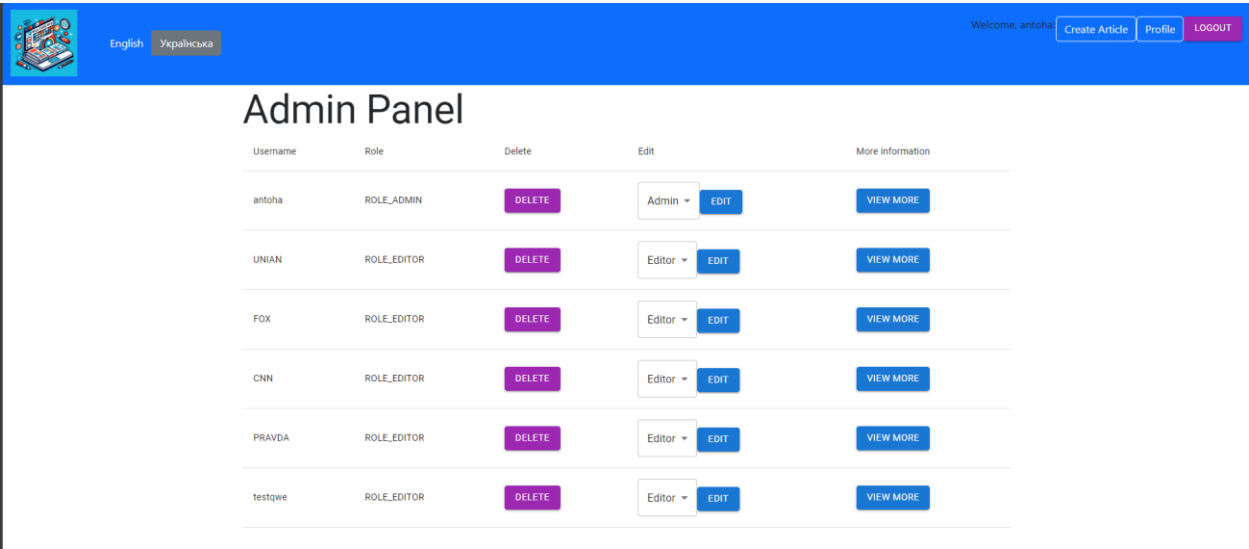


Рисунок 3.14 – Панель адміністратора

Якщо користувач хоче змінити ролі, необхідно обрати нову роль користувача (рисунок 3.15) та натиснути кнопку «Edit» (див. рисунок 3.14).

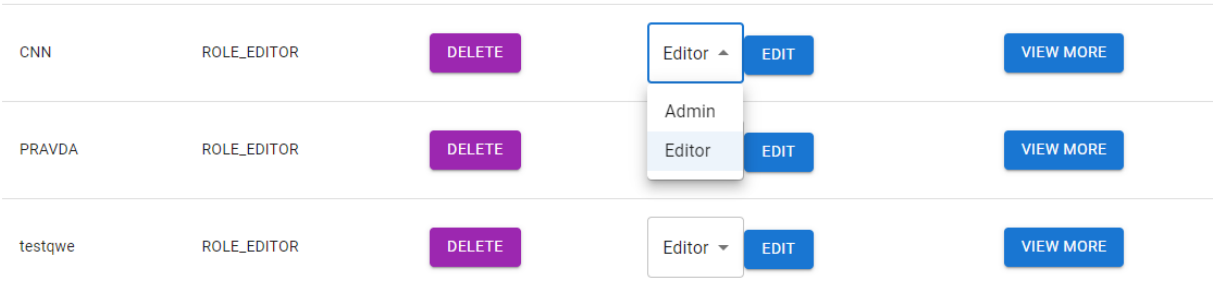


Рисунок 3.15 – Редагування ролей користувача

Для виходу із акаунту користувача треба натиснути кнопку «Logout» (рисунок 3.16) в верхній частині сторінки, після чого користувача буде перенаправлено на головну сторінку.



Рисунок 3.16 – Вихід із акаунту користувача

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2024 р.

Вебзастосунок для агрегації та автоматичного перекладу новин

Графічний матеріал

КПІ.IT-0325.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена СИРОТА

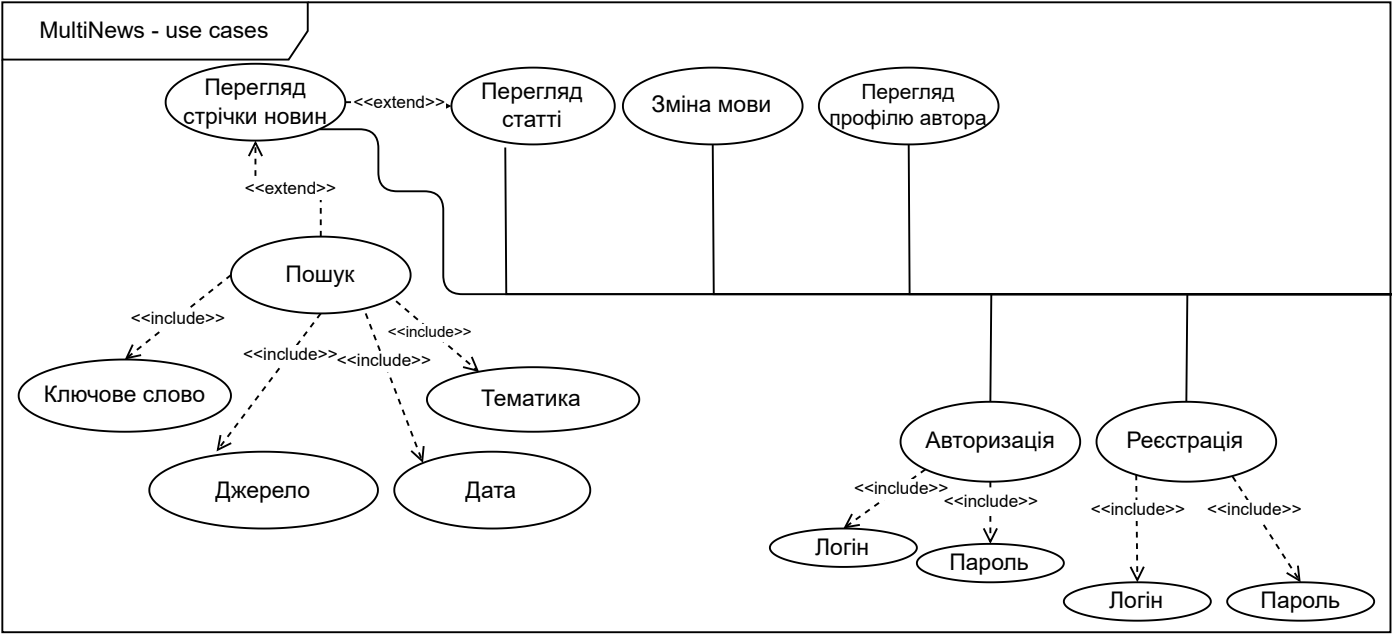
Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

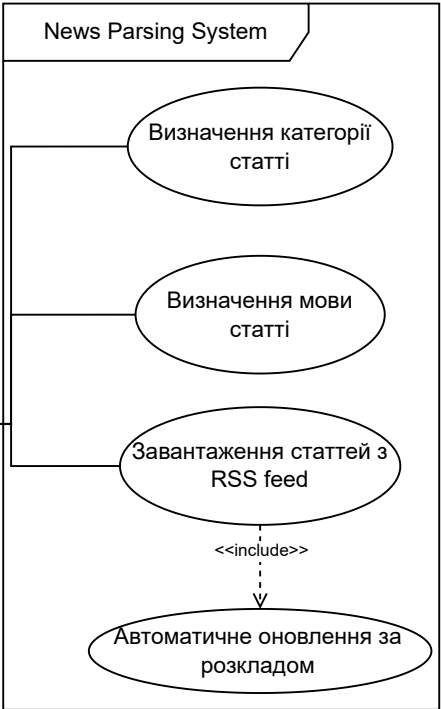
Виконавець:

_____ Антон ЧАБАН

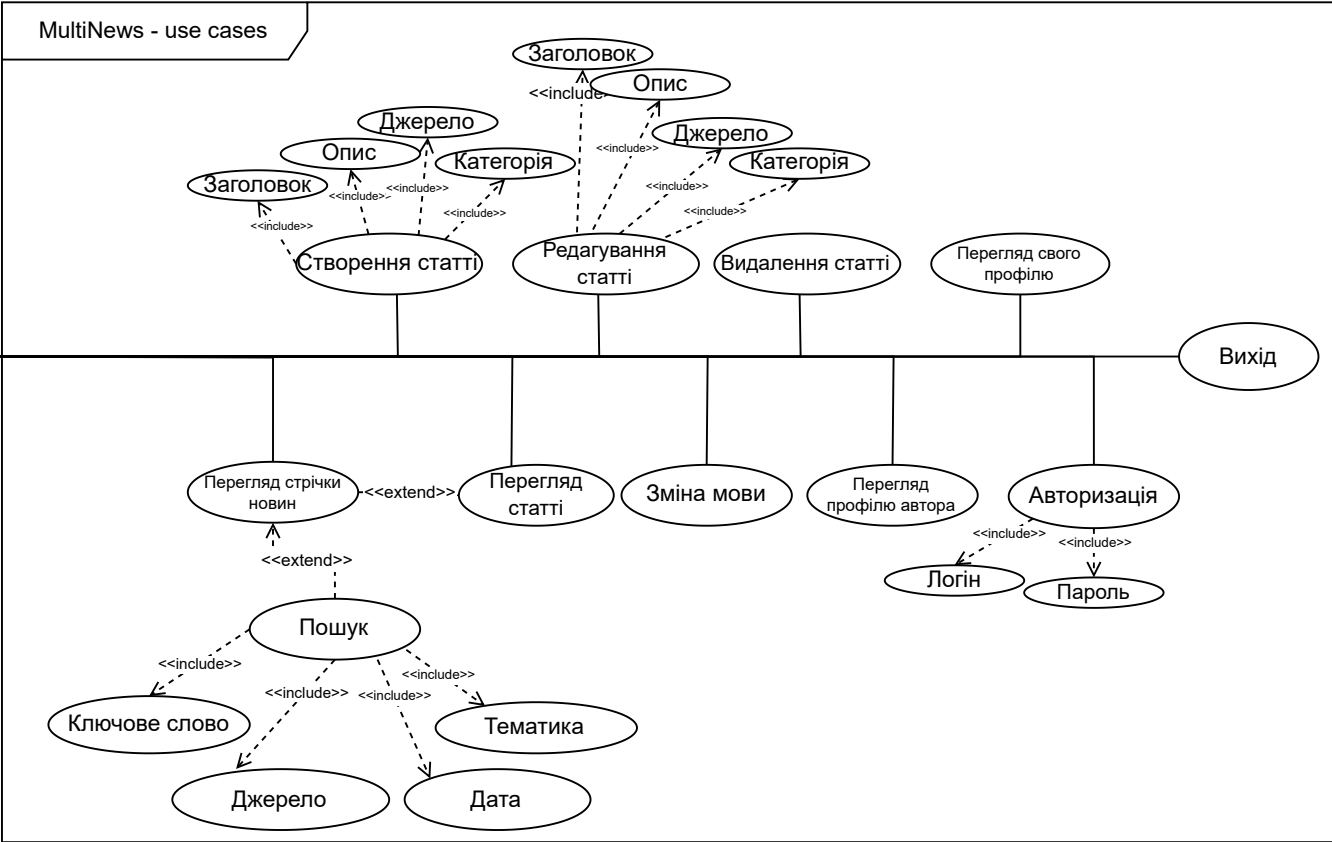
Київ – 2024



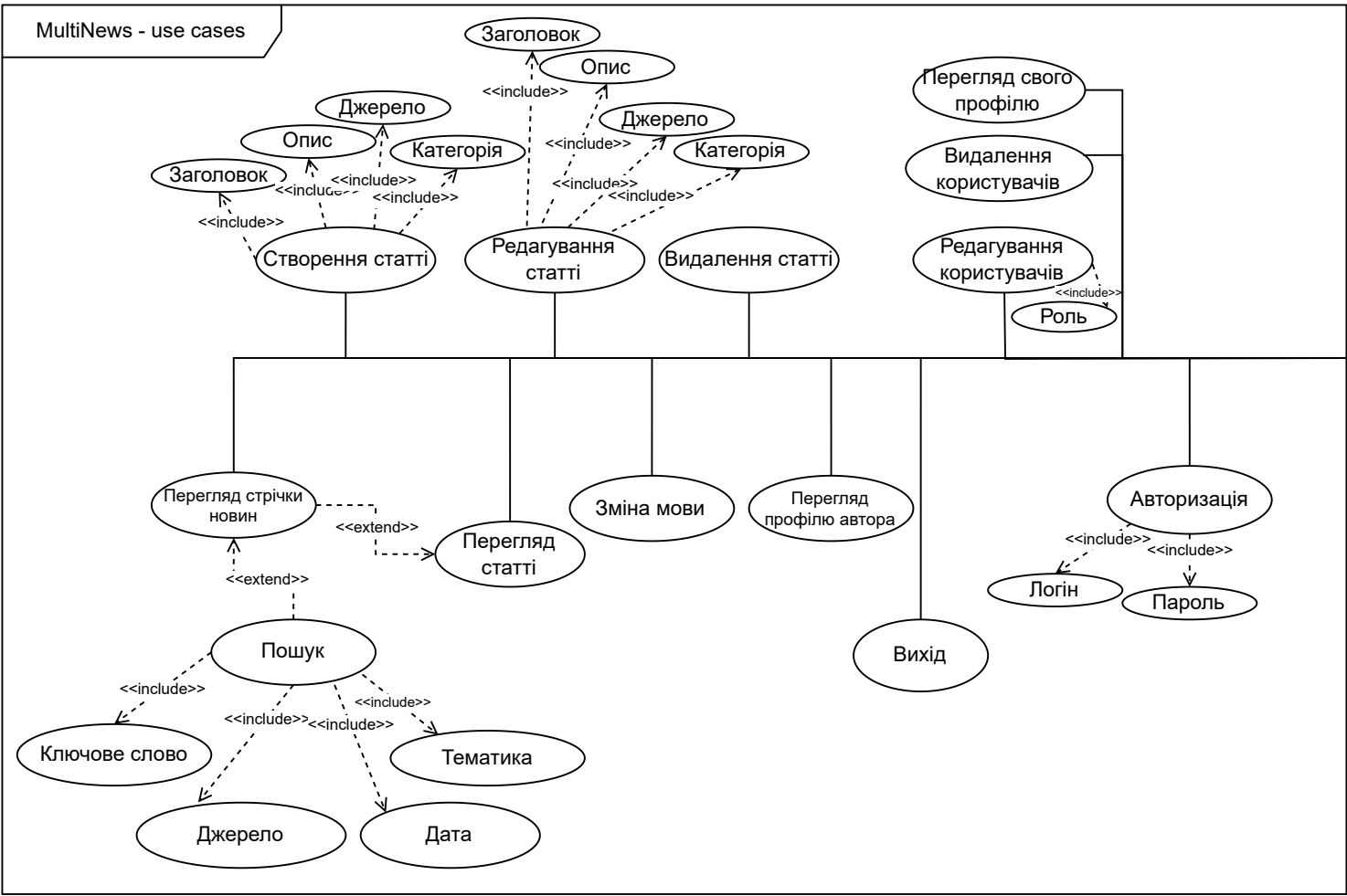
Гість



Article Parser



Редактор

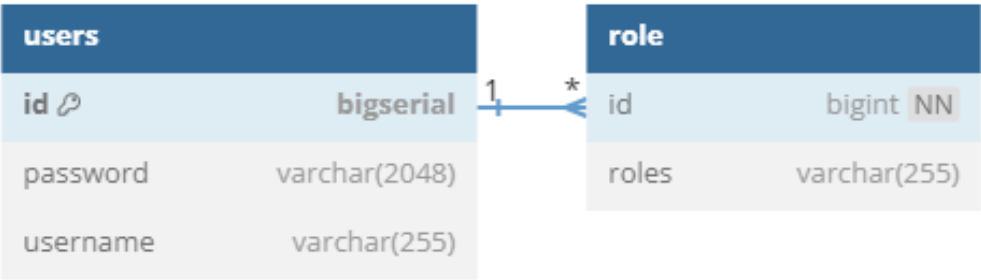


Адміністратор

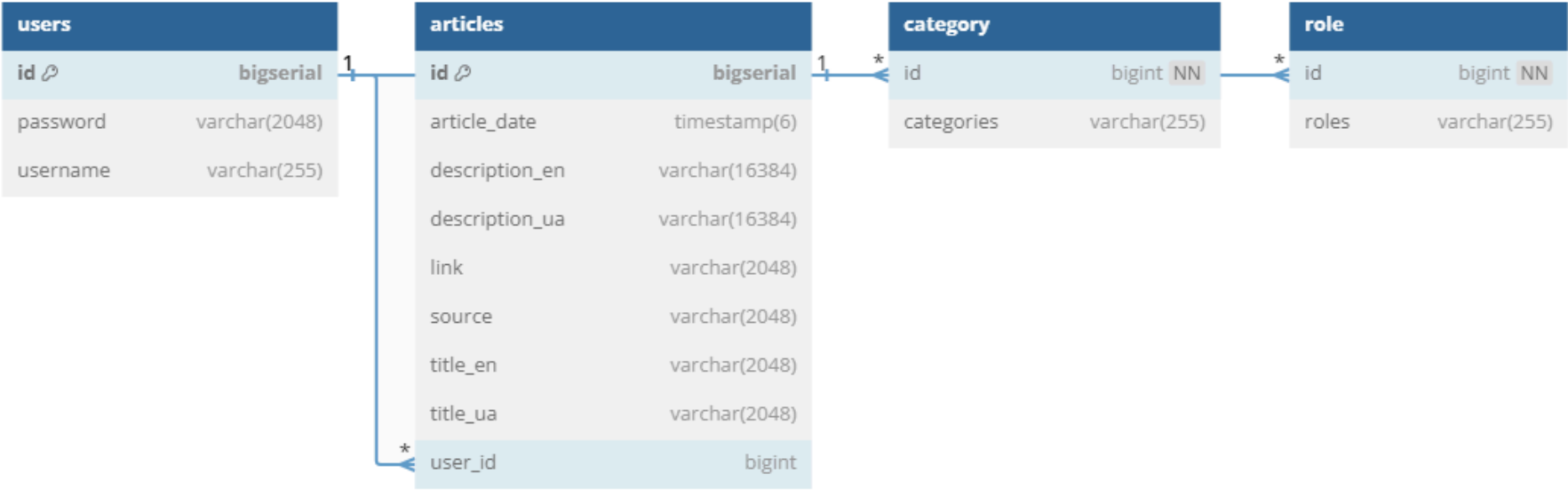
Зм.	Арк.	№ докум.	Підп.	Дата
Розробив	Чабан А. Є.			
Перевірив	Сирота О. П.			
Т. контр.				
Н. контр.	Вітковська І. І.			
Затвердив	Жаріков Е. В.			

КПІ.ІТ-0325.045440.06.99.CCB						
Схема структурна варіантів використань	Лит.		Маса		Масштаб	
	Аркуш 1		Аркушів 4			
Вебзастосунок для агрегації та автоматичного перекладу новин			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01			

Users Database



Articles Database



					КПІ.ІТ-0325.045440.06.99.СБД				
					Схема бази даних				
Зм.	Арк.	№ докум.	Підп.	Дата					
Розробив	Чабан А. Є.								
Перевірів	Сирота О. П.								
Т. контр.									
Н. контр.		Вітковська І. І.							
Затвердив		Жаріков Е.В.			Вебзастосунок для агрегації та автоматичного перекладу новин			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01	

Sign in

Username *

antoha

Password *

...

LOGIN

Forgot password?

Don't have an account? Sign Up

"Ukrenergo" added an hour of possible shutdowns in the evening

No description, revisit the original article to obtain additional information
Source: Українська правда
Date: 6/1/2024, 2:50:37 PM
Categories: GOOD_NEWS
Author: PRAVDA

EDITDELETE

Edit Article

Link

https://www.epravda.com.ua/news/2024/06/1/714556/

Environment

UP

Title (EN)

"Ukrenergo" added an hour of possible shutdowns in the evening

Description (EN)

No description, revisit the original article to obtain additional information

EDIT

Profile

Username: antoha

ADMIN PANEL

Articles:

- Here we can see simple article title
Here you can see simple article description which will be translated

DELETE ARTICLEEDIT ARTICLE
- Simple article title that was updated
Simple article description that was updated

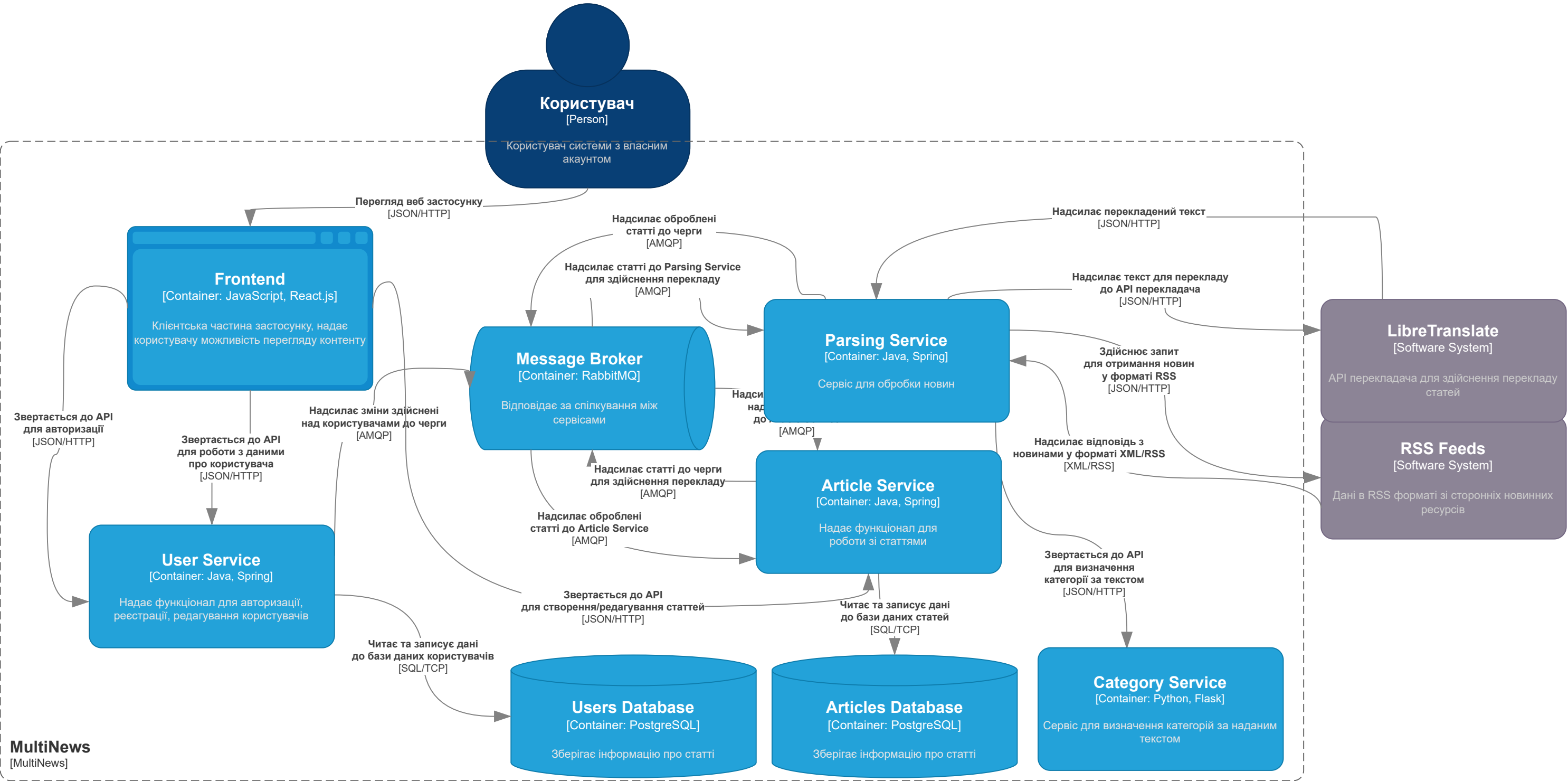
DELETE ARTICLEEDIT ARTICLE

Admin Panel

Username	Role	Delete	Edit	More information
antoha	ROLE_ADMIN	DELETE	Admin EDIT	VIEW MORE
UNIAN	ROLE_EDITOR	DELETE	Editor EDIT	VIEW MORE
FOX	ROLE_EDITOR	DELETE	Editor EDIT	VIEW MORE
CNN	ROLE_EDITOR	DELETE	Editor EDIT	VIEW MORE
PRAVDA	ROLE_EDITOR	DELETE	Editor EDIT	VIEW MORE
testqwe	ROLE_EDITOR	DELETE	Editor EDIT	VIEW MORE

					КПІ.ІТ-0325.045440.06.99.КЕ								
					Креслення вигляду екранних форм				Літера		Маса	Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата									
Розробив	Чабан А. Є.												
Перевірів	Сирота О. П.												
Т. контр.													
					Вебзастосунок для агрегації та автоматичного перекладу новин				Аркуш 3		Аркушів 4		
Н. контр.	Вітковська І. І.												
Затвердив		Жаріков Е.В.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01							

Діаграма контейнерів



MultiNews
[MultiNews]

**[Containers] MultiNews Container
Diagram**

MultiNews system container diagram

Демонстраційний плакат до дипломного проекту

Вебзастосунок для агрегації та автоматичного перекладу новин

Виконав студент гр. ІТ-01 Чабан А. Є.

Керівник Сирота О. П.