

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

керівник

_____ Максим ГОЛОВЧЕНКО

“___” _____ 2024 р.

**Веб-сервіс формування новин на основі публічних сторінок у соціальних
мережах**

Технічне завдання

КПІ.ІП-3324.045440.02.81

“ПОГОДЖЕНО”

Керівник роботи:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олександр СОКОЛОВ

Київ – 2025

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	7
4.1	Вимоги до функціональних характеристик	7
4.1.1	Користувацького інтерфейсу	7
4.1.2	Для користувача:	12
4.1.3	Для адміністратора системи (якщо він передбачений):	12
4.1.4	Додаткові вимоги:	13
4.2	Вимоги до надійності	13
4.3	Умови експлуатації	14
4.3.1	Вид обслуговування.....	14
4.3.2	Обслуговуючий персонал.....	14
4.4	Вимоги до складу і параметрів технічних засобів	15
4.5	Вимоги до інформаційної та програмної сумісності	15
4.5.1	Вимоги до вхідних даних	16
4.5.2	Вимоги до вихідних даних	16
4.5.3	Вимоги до мови розробки	16
4.5.4	Вимоги до середовища розробки.....	17
4.5.5	Вимоги до представленню вихідних кодів	17
4.6	Вимоги до маркування та пакування.....	17
4.7	Вимоги до транспортування та зберігання	17
4.8	Спеціальні вимоги	18
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	19
5.1	Попередній склад програмної документації	19
5.2	Спеціальні вимоги до програмної документації	19
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	20
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	21

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сервіс формування новин на основі публічних сторінок у соціальних мережах

Наведене технічне завдання поширюється на розробку веб-сервісу формування новин на основі публічних сторінок у соціальних мережах **<Найменування>** [КП.ІП-3324.045440.02.81], котра використовується для створення новин на основі публічної інформації, доступної в соціальних мережах та інших онлайн-ресурсах та призначена для використання в галузі інформаційних технологій, зокрема в медіа, новинних агрегаторах і платформах для контент-менеджменту. Основними користувачами є медіа-агенції, журналісти, контент-менеджери, які потребують швидкого доступу до актуальних новин, а також адміністратори платформ, що забезпечують модерацію контенту. Система також може бути корисною для індивідуальних користувачів, зацікавлених у створенні персоналізованих новинних стрічок, та організацій, що займаються моніторингом соціальних мереж.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки <Найменування> є індивідуальний навчальний план студента та РНП затверджена кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Веб-сервіс для формування новин на основі публічних сторінок у соціальних мережах призначений для автоматизації збору, обробки та публікації новинного контенту з різноманітних джерел, таких як соціальні мережі (Telegram, Instagram) та RSS-канали. Функціонально система забезпечує користувачам можливість переглядати сторінку привітання, реєструватися, авторизуватися, додавати та перевіряти джерела, фільтрувати й редагувати контент, створювати та публікувати новини, а також залишати скарги на некоректний контент. Адміністратори отримують інструменти для управління джерелами та розгляду скарг, що сприяє підтримці якості інформації. Експлуатаційно веб-сервіс розроблено для роботи на настільних і мобільних пристроях, забезпечуючи зручний доступ через адаптивний інтерфейс і оптимізовану взаємодію з HTTP API соціальних мереж і RSS-каналів. Це полегшує роботу контент-менеджерів, журналістів і медіа-агенцій, дозволяючи ефективно обробляти великі обсяги даних у реальному часі.

Метою розробки є створення високопродуктивного та безпечно програмного продукту, який оптимізує процеси збору й обробки новин, забезпечуючи високу швидкість відповіді (до 2 секунд для 95% запитів), безпеку даних користувачів і контенту, а також доступність сервісу на рівні 99.9%. Якість досягається завдяки монолітній архітектурі з використанням Domain-Driven Design для структуризації коду, інтеграції з REST API через бібліотеку axios, застосуванню PostgreSQL для надійного зберігання даних і контейнеризації для масштабованості. Система підтримує адаптивний інтерфейс і локалізацію, що підвищує зручність використання для широкої аудиторії.

Мета дослідження полягає у створенні суспільно корисного веб-сервісу, який покращує якість і ефективність формування новинного контенту порівняно з традиційними методами ручної обробки. Це досягається шляхом автоматизації збору даних через HTTP API, забезпечення

інтуїтивного інтерфейсу для користувачів із різним рівнем технічної підготовки та реалізації надійних механізмів модерації контенту. Результатом є продукт, який пропонує швидший доступ до актуальних новин, підвищену безпеку та можливість масштабування для обробки зростаючих обсягів даних, що має бути чітко підтверджено у висновках курсової роботи.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

- Перегляд сторінки привітання (Рисунок 4.1);
- Реєстрація користувача (Рисунок 4.2);
- Авторизація користувача (Рисунок 4.3);
- Додавання джерела новин (Рисунок 4.4);
- Перегляд і фільтрація контенту (Рисунок 4.5);
- Створення та редагування новин (Рисунок 4.6);
- Публікація новин (Рисунок 4.7);
- Залишення скарги (Рисунок 4.8);
- Управління джерелами адміністратором (Рисунок 4.9);
- Обробка скарг адміністратором (Рисунок 4.10).

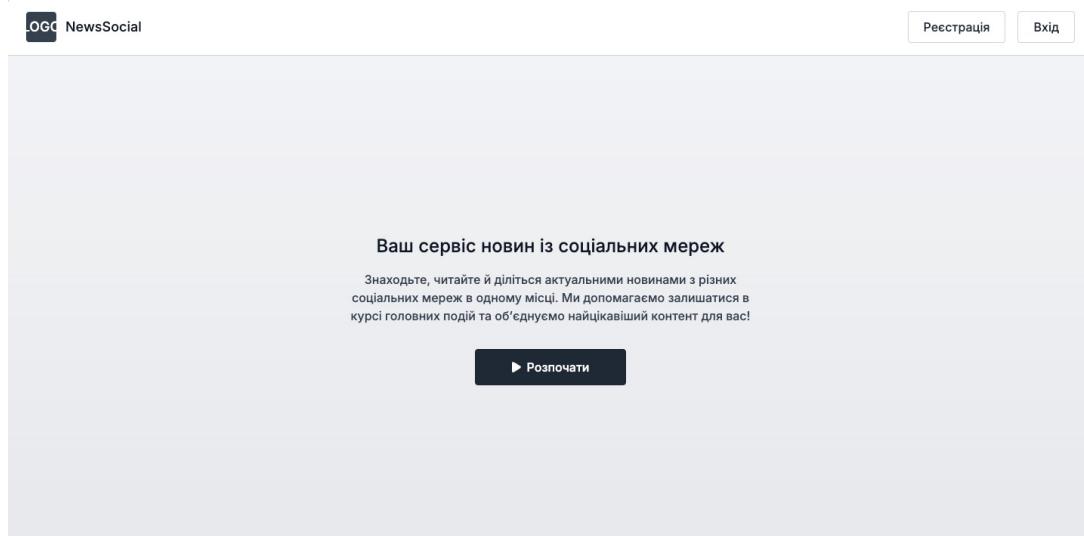


Рисунок 4.1 – Перегляд сторінки привітання

The screenshot shows the registration form for the NewsSocial platform. At the top left is the logo 'LOGO NewsSocial'. At the top right are two buttons: 'Реєстрація' (Registration) and 'Вхід' (Login). The main form is titled 'Створити обліковий запис' (Create account). It contains four input fields: 'Email' (with placeholder 'Введіть ваш email'), 'Пароль' (Password) (with placeholder 'Введіть пароль'), 'Підтвердження пароля' (Confirmation) (with placeholder 'Повторіть пароль'), and a 'Зареєструватися' (Register) button. Below the form is a link 'Уже маєте акаунт? Увійдіть' (Already have an account? Log in).

Рисунок 4.2 – Реєстрація користувача

The screenshot shows the login form for the NewsSocial platform. At the top left is the title 'Авторизація / Authorization'. At the top right is a button 'Гость' (Guest). The main form is titled 'Вхід до системи / Login'. It contains two input fields: 'Email' (with placeholder 'your@email.com') and 'Пароль / Password' (with placeholder '*****'). Below these is a 'Увійти / Sign In' (Log in) button. At the bottom is a link 'Забули пароль? / Forgot password?'.

Рисунок 4.3 – Авторизація користувача

NewsSocial

Реєстрація Вхід

Додати джерело / Add Source

Назва джерела / Source Name
Введіть назву / Enter name

Тип / Type
Оберіть тип / Select type

RSS Telegram Instagram

URL або API-ключ / URL or API key
Введіть URL або ключ / Enter URL or key

Додати / Add

© 2025 NewsSocial

Рисунок 4.4 – Додавання джерела новин

NewsSocial

Реєстрація Вхід

Фільтри / Filters

Джерело / Source
Усі / All

Дата / Date
dd.mm.yyyy

Ключові слова / Keywords
Введіть слова / Enter

▼ Фільтрувати / Filter

Зображення / Image	Зображення / Image	Зображення / Image
Заголовок новини / News Title Короткий опис новини, що складається до 100 символів. / Short news summary up to 100 chars. RSS 22.05.2025	Заголовок / Title Опис новини з Instagram, коротко. / Instagram news summary, short. Instagram 20.05.2025	Заголовок / Title Новина з Telegram, короткий зміст. / Telegram news, short summary. Telegram 19.05.2025
Зображення / Image Заголовок новини / News Title Короткий опис новини, до 100 символів. / Short news summary. RSS 18.05.2025	Зображення / Image Instagram новина / Instagram News Це короткий опис Instagram новини. / This is a short Instagram news description. Instagram 17.05.2025	Зображення / Image Telegram новина / Telegram News Опис Telegram новини, коротко. / Telegram news short description. Telegram 16.05.2025

© 2025 NewsSocial

Рисунок 4.5 – Перегляд і фільтрація контенту

Створення/Редагування новини / Create/Edit News

Заголовок / Title
Введіть заголовок / Enter title

Текст / Text
Введіть текст новини / Enter news text

Зображення / Image
Choose file No file chosen

Джерело / Source
Обрати джерело / Select source

Зберегти чернетку / Save Draft Попередній перегляд / Preview

Попередній перегляд / Preview

Зображення / Image Preview

Заголовок новини / News Title
Текст новини буде тут. / News text will appear here.

RSS 22.05.2025

Рисунок 4.6 – Створення та редагування новин

Підтвердити публікацію / Confirm Publication

Зображення / Image Preview

Заголовок новини / News Title
Текст новини буде тут. / News text will appear here.

RSS 22.05.2025

Скасувати / Cancel Опублікувати / Publish

© 2025 NewsSocial

Рисунок 4.7 – Публікація новин

The screenshot shows a web page titled 'Залишити скаргу / Submit a Complaint'. At the top right are 'Реєстрація' (Registration) and 'Вхід' (Login) buttons. The main form has fields for 'Причина / Reason' (with a dropdown menu 'Оберіть причину / Select reason') and 'Опис / Description' (with placeholder text 'Опишіть проблему... / Describe the issue...'). A large '↗ Надіслати / Submit' button is at the bottom.

Рисунок 4.8 – Залишення скарги

The screenshot shows a table titled 'Джерела / Sources' with three entries. The columns are 'Назва / Name', 'Тип / Type', 'URL', 'Статус / Status', and 'Дії / Actions'. The entries are:

Назва / Name	Тип / Type	URL	Статус / Status	Дії / Actions
Новини UA	RSS	news-ua.com/rss	Активний / Active	
TechBlog	API	api.techblog.com/v1	Неактивний / Inactive	
World Updates	Manual	worldupdates.org	Активний / Active	

Рисунок 4.9 – Управління джерелами адміністратором

Скарги / Complaints			
Причина / Reason	Опис / Description	Статус / Status	Дії / Actions
Неправдива інформація / False information	Пост містить неправдиві факти / The post contains false facts	Нова / New	<input checked="" type="checkbox"/> <input type="checkbox"/>
Порушення правил / Rule violation	Використано ненормативну лексику / Offensive language used	Нова / New	<input checked="" type="checkbox"/> <input type="checkbox"/>
Інше / Other	Зображення не відповідає темі / Image does not match topic	Оброблена / Processed	<input type="checkbox"/> <input checked="" type="checkbox"/>

Рисунок 4.10 – Обробка скарг адміністратором

4.1.2 Для користувача:

- Реєстрація облікового запису;
- Авторизація в системі;
- Вихід із системи;
- Додавання нового джерела новин;
- Перевірка валідності джерела;
- Пошук джерел за назвою або типом;
- Збір контенту з джерел;
- Фільтрація контенту за ключовими словами, датою чи джерелом;
- Редагування контенту перед створенням новини;
- Створення новин на основі зібраного контенту;
- Публікація новин;
- Залишення скарги на контент;
- Перегляд власного профілю.

4.1.3 Для адміністратора системи:

- Перегляд списку джерел;

- Редагування або видалення джерел;
- Перегляд списку скарг;
- Обробка скарг (прийняття або відхилення);
- Управління правами доступу користувачів.

4.1.4 Додаткові вимоги:

- Автоматична перевірка валідності джерел перед додаванням до системи
- Забезпечення доступу до системи через сучасні браузери

4.2 Вимоги до надійності

Веб-сервіс для формування новин на основі публічних сторінок у соціальних мережах розроблено з акцентом на забезпечення надійності роботи системи, що є критично важливим для збереження даних і безперебійного функціонування. Для захисту від некоректних дій користувача реалізовано контроль введення інформації: усі поля форм (реєстрація, додавання джерел, створення новин, скарги) проходять валідацію на клієнтській (Vue.js) і серверній (NestJS) сторонах. Наприклад, поля для email перевіряються на відповідність формату, URL джерел – на валідність, а текстові поля – на захист від XSS-атак через екранування. У разі введення некоректних даних користувач отримує чіткі повідомлення про помилку, що мінімізує ризик некоректної поведінки. Цілісність інформації в базі даних PostgreSQL гарантується використанням транзакцій для всіх операцій запису, таких як додавання джерел, створення новин чи обробка скарг. Індекси та обмеження цілісності (foreign keys) забезпечують консистентність даних. Для відновлення після збоїв система створює автоматичні резервні копії бази даних кожні 24 години, які зберігаються в хмарному сховищі. Час відновлення системи після критичного збою не перевищує 10 хвилин завдяки контейнеризації (Docker) і автоматичному переключенню на резервні сервери. Контрольні точки для транзакцій

дозволяють відновлювати проміжні результати, наприклад, збережені чернетки новин, у разі переривання роботи. Логування всіх дій користувачів і системи через NestJS Logger забезпечує можливість аудиту та швидкого виявлення причин збоїв.

4.3 Умови експлуатації

Програмне забезпечення призначене для експлуатації на стаціонарних комп’ютерах, ноутбуках, планшетах і смартфонах із доступом до мережі Інтернет через сучасні веб-браузери (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge).

4.3.1 Вид обслуговування

Веб-сервіс для формування новин на основі публічних сторінок у соціальних мережах не потребує спеціального обслуговування, оскільки розроблений як хмарний додаток із автоматизованим управлінням. Система розгортається на платформах, таких як AWS або Google Cloud, де оновлення, моніторинг і масштабування виконуються автоматично через контейнеризацію (Docker) і оркестрацію (Kubernetes). Регулярне оновлення програмного забезпечення, включаючи патчі безпеки та функціональні покращення, здійснюється через CI/CD-пайплайн без залучення спеціалізованого персоналу. Резервне копіювання бази даних PostgreSQL і логування подій налаштовано автоматично, що виключає потребу в ручному обслуговуванні. Таким чином, вимоги до виду обслуговування не висуваються, оскільки система спроектована для автономної роботи з мінімальним втручанням.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

- Тип пристрою: ПК, ноутбук, планшет або смартфон із сучасним веб-браузером актуальних версій (Google Chrome, Firefox, Safari, Edge);
- Центральний процесор: 2-ядерний, з тактовою частотою від 1.4 ГГц;
- Оперативна пам'ять: не менше 2 ГБ;
- Інтернет-з'єднання: стабільне підключення зі швидкістю від 5 Мбіт/с;

Рекомендована конфігурація технічних засобів:

- Тип пристрою: ПК, ноутбук, планшет або смартфон із сучасним веб-браузером (останні версії Chrome, Firefox, Safari, Edge);
- Центральний процесор: 4-ядерний, з тактовою частотою від 2.0 ГГц.
- Оперативна пам'ять: не менше 4 ГБ;
- Інтернет-з'єднання: стабільне підключення зі швидкістю від 20 Мбіт/с.

4.5 Вимоги до інформаційної та програмної сумісності

Веб-сервіс для формування новин розроблено для забезпечення широкої сумісності з сучасними платформами та технологіями, що дозволяє використовувати його на різноманітних пристроях і в різних середовищах. Клієнтська частина (веб-додаток на Vue.js із TypeScript) сумісна з сучасними браузерами, включаючи Chrome (версія 120+), Firefox (115+), Safari (16+), Edge (120+), що забезпечує коректну роботу на операційних системах Windows 10+, macOS 11+, Android 10+, iOS 15+. Серверна частина (NestJS, Node.js, PostgreSQL) розгортається на хмарних платформах із Linux-based середовищем (наприклад, Ubuntu 20.04+), що замінює застарілі вимоги до WIN32 чи Unix. Система інтегрується із зовнішніми джерелами (Telegram Bot

API, Instagram Graph API, RSS-канали) через стандартні REST API та XML/JSON формати, гарантуючи інформаційну сумісність.

4.5.1 Вимоги до вхідних даних

Вхідні дані для системи представлені у структурованому форматі, залежно від джерела. Для соціальних мереж (Telegram, Instagram) дані надходять у форматі JSON через REST API, де кожен об'єкт містить поля, такі як текст повідомлення, URL зображення, дата публікації та метадані автора. RSS-канали надають дані у форматі XML або JSON, із структурою, що включає заголовок, опис, посилання та дату. Користувачькі дані, такі як email, пароль чи URL джерел, вводяться через веб-форми та валіduються на відповідність формату (наприклад, RFC 5322 для email, валідний URL для джерел). Некоректні дані відхиляються з повідомленням про помилку.

4.5.2 Вимоги до вихідних даних

Вихідні дані системи формуються у вигляді опублікованих новин, представлених у форматі JSON для внутрішнього використання (API) та HTML для відображення у веб-інтерфейсі. Кожна новина містить заголовок (строка до 255 символів), текст (строка без обмеження довжини), опціональне зображення (URL або base64), дату публікації (ISO 8601) і джерело (посилання). У веб-інтерфейсі новини відображаються як картки з адаптивним дизайном, стилізовані через Tailwind CSS. Скарги та звіти адміністратора експортуються у форматі JSON або CSV для аналізу, із полями: ідентифікатор, причина, опис, статус.

4.5.3 Вимоги до мови розробки

Розробка виконана на мовах програмування TypeScript (для клієнтської та серверної частин) і SQL (для запитів до PostgreSQL). TypeScript обрано через його строгу типізацію, що підвищує надійність коду, а SQL – для ефективної роботи з реляційною базою даних. Додаткові бібліотеки, такі як

axios (для REST API) і xml2js (для RSS), використовуються для обробки вхідних даних.

4.5.4 Вимоги до середовища розробки

Середовище розробки включає платформу Node.js (версія 18+) для серверної частини та Vue.js (версія 3+) для клієнтської. NestJS слугує фреймворком для бекенду, забезпечуючи модульну структуру, а TypeORM – для взаємодії з PostgreSQL. Розробка ведеться в інтегрованому середовищі Visual Studio Code із плагінами для TypeScript, ESLint і Prettier. Для тестування використовується Jest, а для контейнеризації – Docker. CI/CD-пайплайн налаштовані через GitHub Actions для автоматизації розгортання.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми представлений у вигляді структурованих файлів TypeScript, організованих за принципами Domain-Driven Design. Клієнтська частина містить компоненти Vue.js (файли .vue), модулі Pinia для управління станом і конфігурацію Vue Router. Серверна частина включає модулі NestJS (controllers, services, entities), структуровані за доменами (авторизація, джерела, контент). Код супроводжується коментарями англійською мовою, відповідає стандартам ESLint і форматується через Prettier. Усі вихідні файли зберігаються в репозиторії Git із чіткою структурою директорій, включаючи README.md із інструкціями для розгортання.

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

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

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

- технічне завдання;
- пояснівальна записка;
- текст програми.

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

Відповідь

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Розділу тестування програмного забезпечення”.

Пояснювальна записка до курсової роботи

на тему: **Веб-сервіс формування новин на основі публічних сторінок у соціальних мережах**

КПЛП-3324.045440.02.81

Київ – 2025

ЗМІСТ

ВСТУП.....	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	7
1.1 Аналіз предметної області	7
1.2 Аналіз існуючих рішень	9
1.2.1 Аналіз відомих програмних продуктів	9
1.2.2 Аналіз відомих алгоритмічних та технічних рішень.....	19
1.3 Опис бізнес-процесів.....	23
Висновки до розділу	26
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	28
2.1 Варіанти використання програмного забезпечення.....	28
2.2 Аналіз системних вимог	43
2.3 Розроблення функціональних вимог	44
2.4 Розроблення нефункціональних вимог	49
2.4.1 Продуктивність	49
2.4.2 Безпека	49
2.4.3 Доступність.....	50
2.4.4 Масштабованість.....	50
2.4.5 Зручність використання.....	50
2.4.6 Надійність	51
2.4.7 Сумісність	51
2.4.8 Портативність	52
2.5 Постановка задачі	52
Висновки до розділу	53
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	56
3.1 Архітектура програмного забезпечення.....	56
3.2 Важливі архітектурні рішення	62
3.3 Обґрунтування вибору засобів розробки	63
3.4 Конструювання програмного забезпечення.....	66

3.5 Аналіз безпеки даних	69
Висновки до розділу	69
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	
70	
4.1 Аналіз якості ПЗ	70
4.2 Опис процесів тестування	70
4.3 Опис контрольного прикладу	71
Висновки до розділу	71
5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	72
5.1 Розгортання програмного забезпечення	72
5.2 Супровід програмного забезпечення	72
5.3 Повна інструкція користувача	72
Висновки до розділу	73
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75
ДОДАТКИ	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	= Integrated Development Environment – інтегроване середовище розробки.
API	= Application programming interface, прикладний програмний Інтерфейс
SDK	= Software development kit
ІТ	= Інформаційні технології
ER	= Entity-Relation diagram
ОС	= Операційна система.
БД	= База даних.

ВСТУП

У сучасному світі інформаційні потоки відіграють надзвичайно важливу роль, а ефективна обробка та публікація новин стає критичним завданням для багатьох організацій. Однак у контексті великого потоку даних, який надходить із різних джерел, виникає потреба в інструментах, що дозволяють створювати релевантний і якісний контент на основі публічної інформації, доступної в соціальних мережах та інших онлайн-ресурсах.

Актуальність даної роботи зумовлена необхідністю автоматизації процесів збору інформації, її аналізу та формування новин. У зв'язку з активним використанням таких платформ, як Telegram, Instagram, Facebook, та інших, стає важливим забезпечення інтеграції з цими джерелами для збору публічних даних та їхньої обробки відповідно до потреб редакторів і журналістів.

На світовому рівні подібні проблеми вирішуються за допомогою інструментів агрегування новин, але переважна більшість з них фокусується на автоматичному розподілі новин без можливості створення контенту на їхній основі. Пропонований веб-сервіс поєднує функціонал агрегатора новин з редакторським інструментом для створення оригінальних матеріалів на основі зібраної інформації.

Об'єктом розробки є монолітний веб-застосунок, що дозволяє:

- обирати джерела публічної інформації (канали в Telegram, сторінки в Instagram та інші);
- автоматизовано зтягувати дані з джерел за допомогою RSS [2] , API або веб-скрапінгу;
- надавати користувачам можливість редагувати отриману інформацію та створювати власні новини;
- публікувати готовий контент у зручному форматі.

Сфери застосування даного сервісу включають редакції новинних видань, PR-агентства, блогерів та аналітичні центри. Завдяки інтеграції з популярними соціальними мережами та підтримці багаторазового

використання даних, цей веб-застосунок може стати універсальним інструментом для створення якісного контенту.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Веб-додаток [1] – це програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера. Зовнішній інтерфейс веб програми розробляється за допомогою таких мов програмування: HTML, CSS, Javascript, які підтримуються на будь-якому браузері (Opera, Chrome, Mozilla, Yandex). У той час як для написання серверної частини (Back-end) може використовуватися будь-яка інша мова програмування або фреймворк, Python, PhP, Ruby, Java.

RSS [2] – це родина XML [7] -форматів, що використовується для публікації та постачання інформації, що часто змінюється, наприклад, нових записів в блозі, заголовків новин, анонсів статей, зображень, аудіо і відео матеріалів (в стандартизованому форматі). Документ в стандарті RSS (який також інколи називають «стрічкою», «вебстрічкою» або «каналом») складається з повного або часткового тексту і метаданих (дата і авторство).

API [3] – це набір інструкцій, які дозволяють різним програмам спілкуватись між собою та обмінюватись даними.

Веб-скрапінг [4]- це процес автоматичного вилучення даних з веб-сайтів, веб-сервісів і веб-додатків.

Предметна область даного проекту – це створення новин на основі публічної інформації, доступної в соціальних мережах та інших онлайн-ресурсах. У контексті сучасного розвитку інформаційних технологій, соціальні мережі стали основним джерелом новин, які швидко поширяються серед користувачів. Такі платформи, як Telegram, Instagram, Facebook, Twitter, та інші, дозволяють отримувати оперативну інформацію про події у світі, але її обробка та подальше використання для створення якісного контенту вимагає значних зусиль.

На сьогодні існують численні сервіси для агрегування новин, які автоматично збирають дані з різних джерел. Прикладами є Google News,

Feedly та Pocket. Вони надають можливість користувачам стежити за новинами, але не підтримують редагування або створення новин на основі зібраних даних. Це є суттєвим недоліком для редакцій і PR-агентств, які потребують інструментів для адаптації контенту до потреб аудиторії.

У сфері IT реалізація роботи з соціальними мережами переважно базується на використанні таких технологій:

- RSS – для отримання новин із джерел, що підтримують цей формат;
- API – для інтеграції з платформами, які надають відкриті прикладні інтерфейси;
- Веб-скрапінг – для збору даних із джерел, що не мають інших методів інтеграції, за умови дотримання правил використання.

Недоліки поточного стану:

- Брак інструментів, які об'єднують функціонал збору новин та їх редагування;
- Обмеження доступу до даних через зміни політик API платформ;
- Велика кількість інформаційного шуму, що ускладнює аналіз даних.

Шляхи покращення ситуації:

- Розробка універсального інструменту, що поєднує збір, обробку та редагування новин;
- Використання сучасних технологій для автоматизації процесів;
- Забезпечення зручного інтерфейсу для користувачів із різним рівнем підготовки.

У рамках даної курсової роботи обрано шлях розробки монолітного веб-застосунку, що підтримує інтеграцію з основними джерелами інформації через RSS, API та веб-скрапінг. Такий підхід дозволяє досягти поставлених цілей, забезпечуючи ефективну обробку даних і створення якісного контенту.

1.2 Аналіз існуючих рішень

1.2.1 Аналіз відомих програмних продуктів

Для аналізу відомих програмних продуктів у сфері роботи з новинами та їх обробки було обрано кілька рішень, які частково реалізують функціонал, схожий до описаного в технічному завданні: Feedly і NewsBlur. Нижче наведено їхній опис та порівняльний аналіз із пропонованою розробкою.

Опис програмних продуктів:

- Feedly є одним із найпопулярніших RSS-агрегаторів. Він дозволяє користувачам збирати новини з різних джерел, організовувати їх у тематичні стрічки та читати новини через зручний інтерфейс. Основний недолік – відсутність функцій редактування зібраного контенту або створення нових публікацій;
- NewsBlur – це RSS-агрегатор, який дозволяє користувачам не лише читати новини, але й використовувати автоматичні фільтри для персоналізації потоку інформації. Проте NewsBlur, як і Feedly, обмежується лише читанням контенту без можливості його редактування чи створення нових матеріалів.

Перейдемо до аналізу Feedly. Після реєстрації та авторизації, нам доступна сторінка із пошуком стрічок новин, на які користувач хоче підписатися (рисунок 1.1). Для пошуку ми можемо використовувати теги, посилання на сайти та RSS посилання.

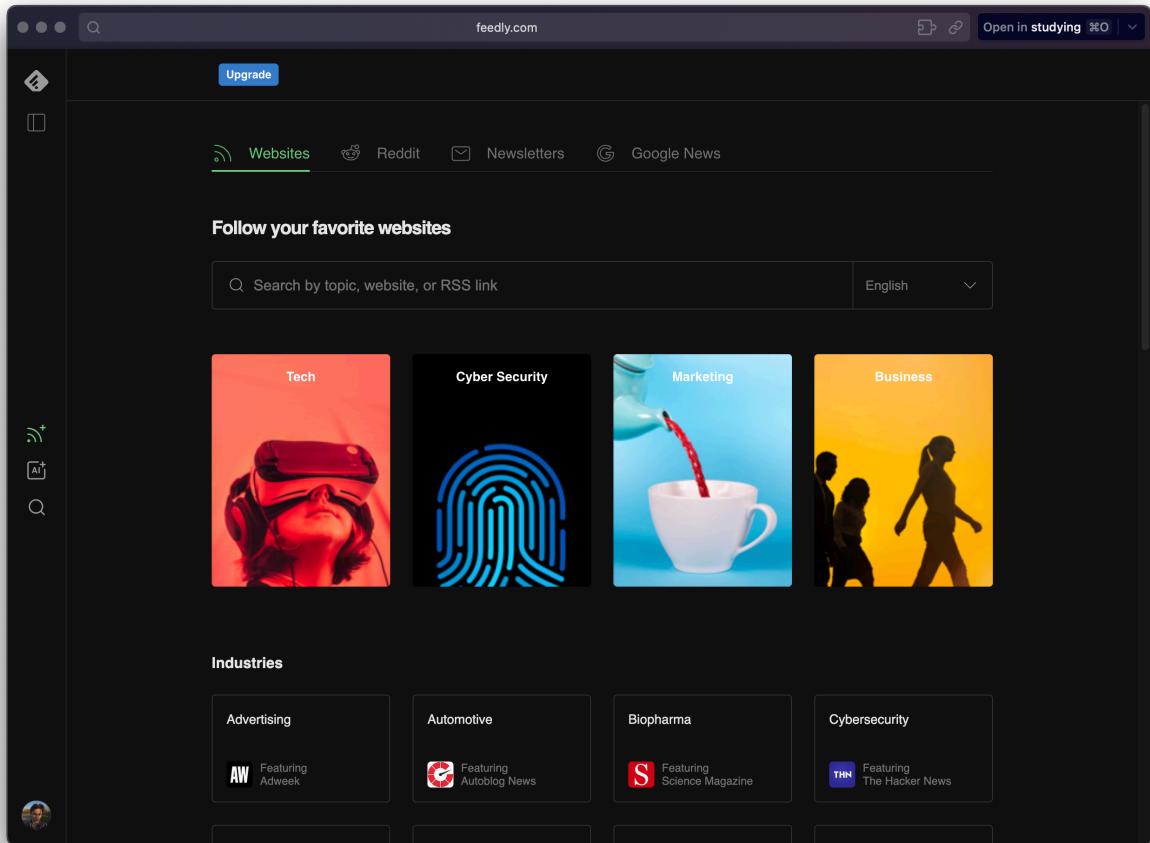


Рисунок 1.1 – Сторінка пошуку

Скористаємося пошуком, знайдемо стрічки новин за тегом “programming”. Нам виводяться відповідні стрічки новин (рисунок 1.2).

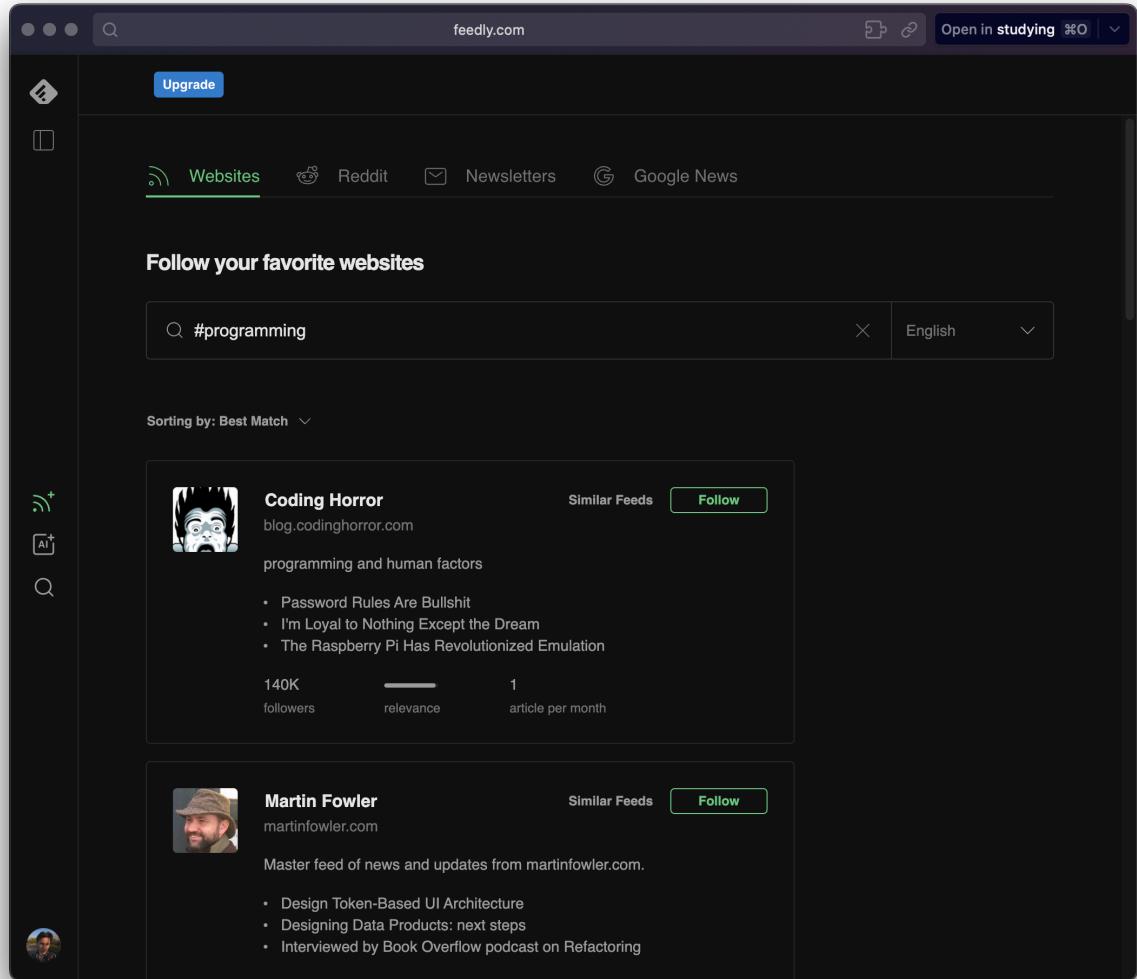


Рисунок 1.2 – Пошук по тегу

Для того, щоб підписатися на стрічку новин, потрібно спочатку мати створену папку для стрічок, куди користувач хоче зберегти (рисунок 1.3).

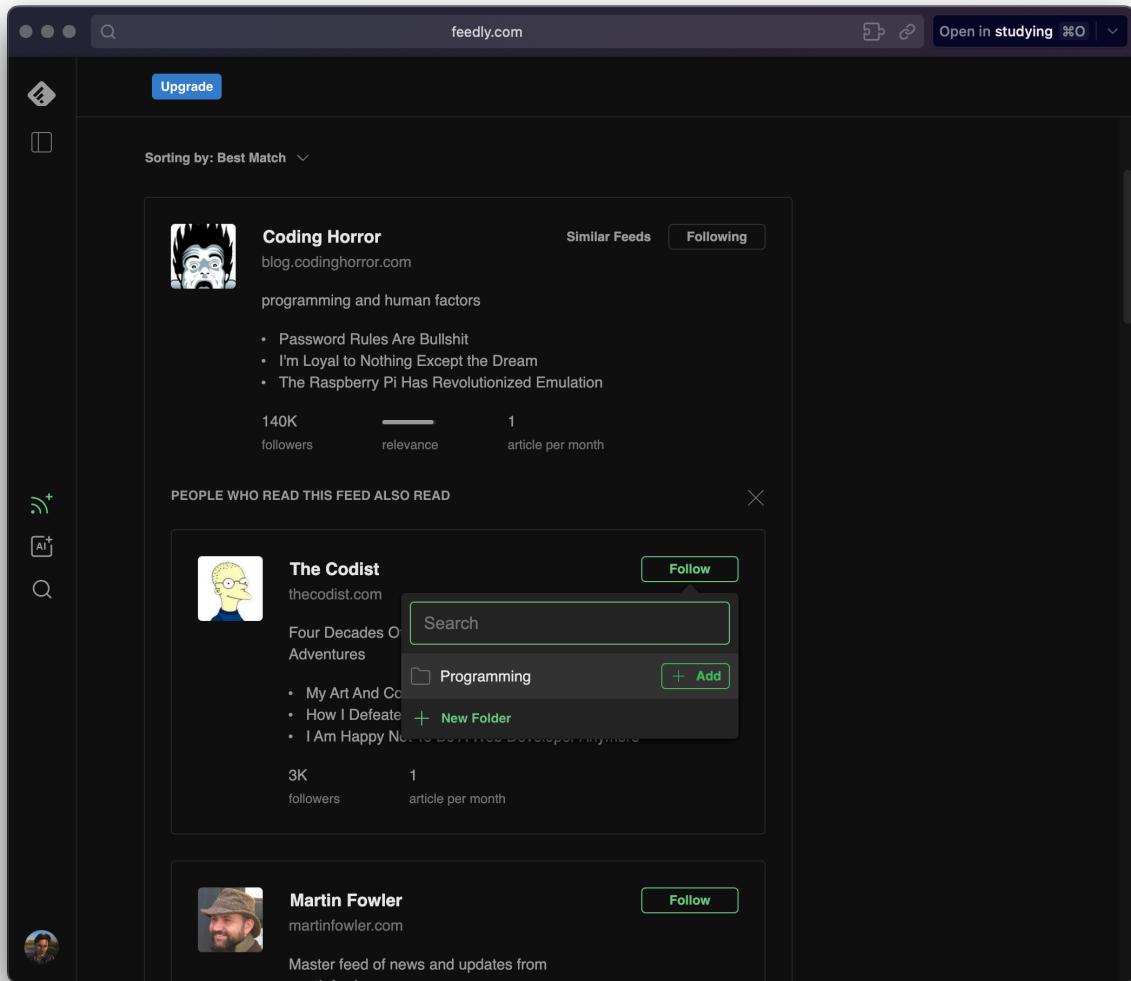


Рисунок 1.3 – Додавання стрічки новин в папку

Після налаштування наших папок із стрічками, ми можемо їх переглянути. Для цього потрібно перейти у відповідний розділ (рисунок 1.4).

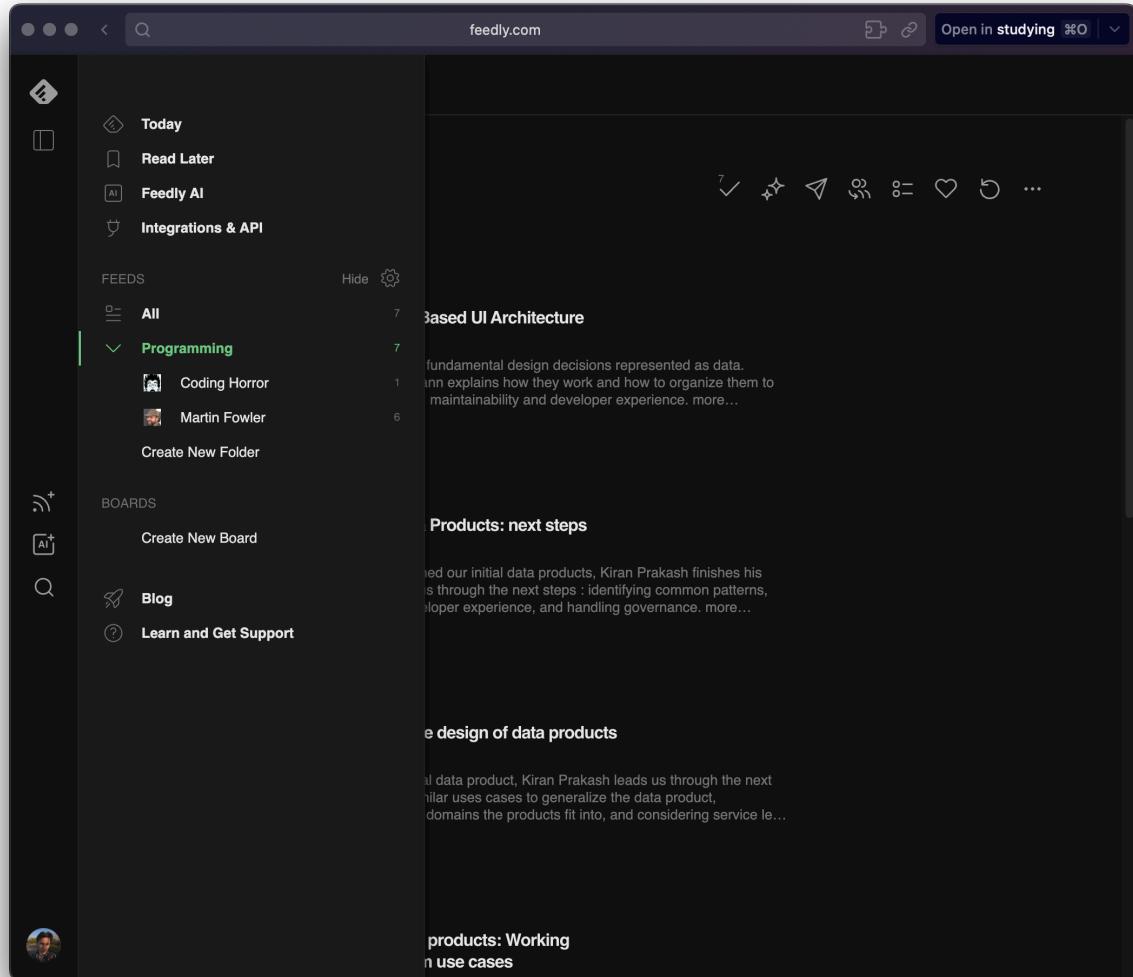


Рисунок 1.4 – Перегляд папок користувача із стрічками новин

Користувач може переглянути новини із стрічок, які він додав попередньо (рисунок 1.5).

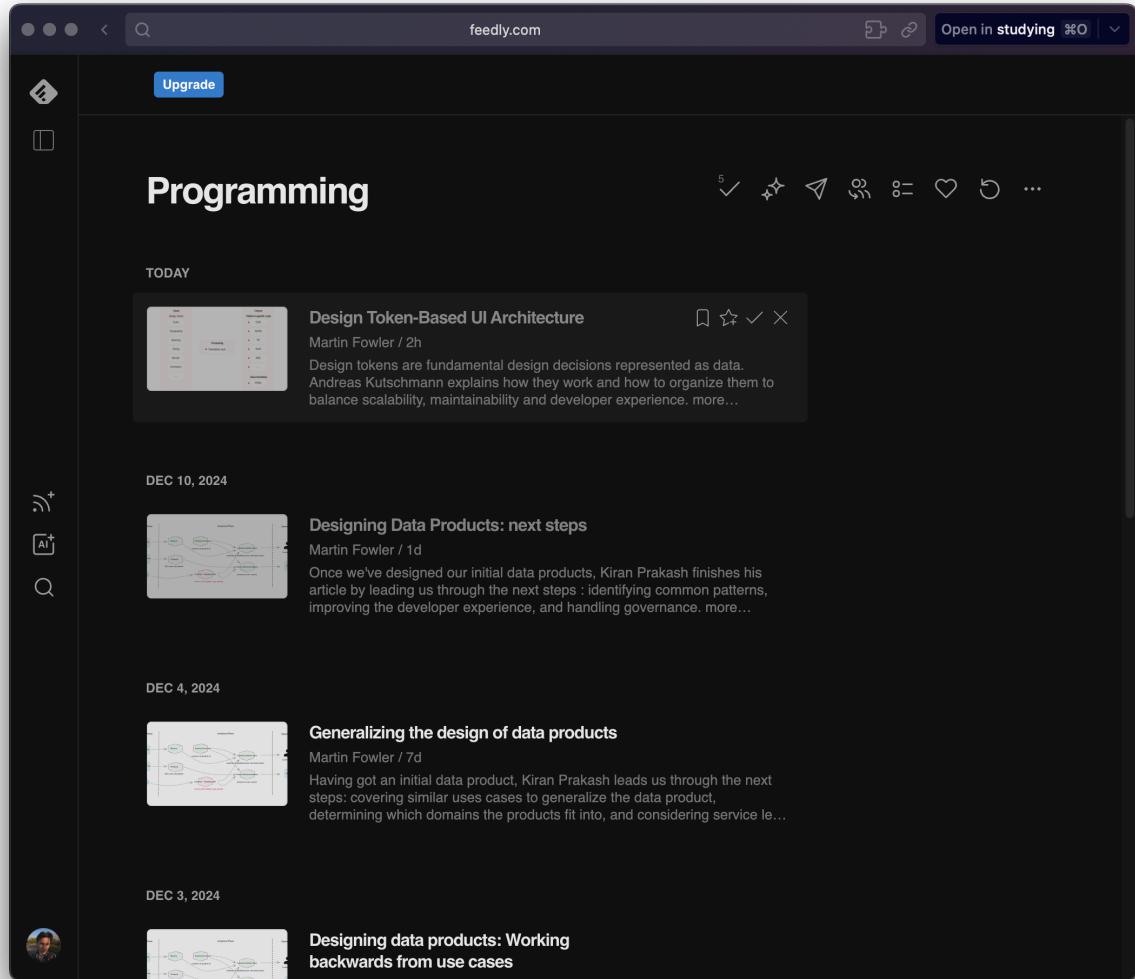


Рисунок 1.5 – Сторінка новин в папці користувача

Для того, щоб переглянути джерело, потрібно натиснути на новину, де буде її короткий огляд та можливість перейти на сайт-джерело (рисунок 1.6).

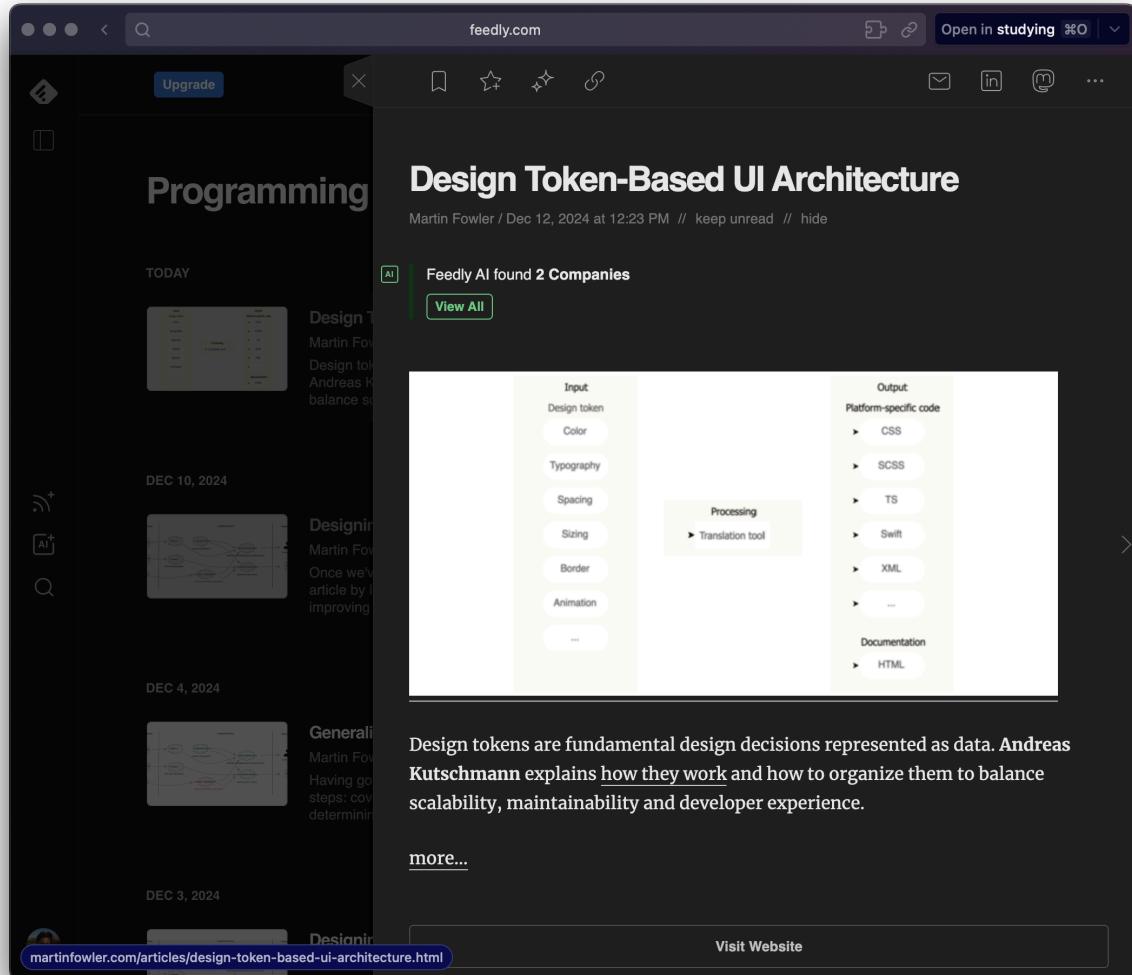


Рисунок 1.6 – Огляд новини із можливістю перейти до сайту-джерела

Переглянемо застосунок BlurNews. Аналогічно до Feedly, після реєстрації та авторизації, користувач бачить початкову сторінку.

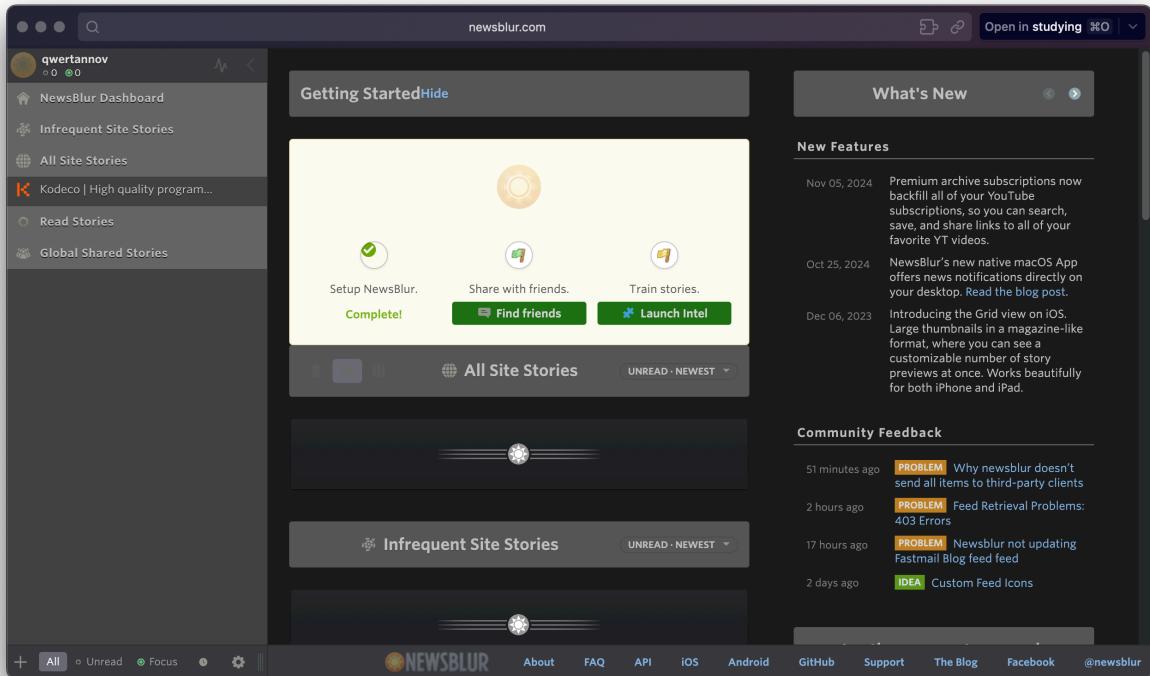


Рисунок 1.7 – Початкова сторінка

У нижній лівій частині сторінки можна додати стрічки новин. Для пошуку користувач може ввести називу RSS стрічки, або напряму RSS посилання (рисунок 1.8).

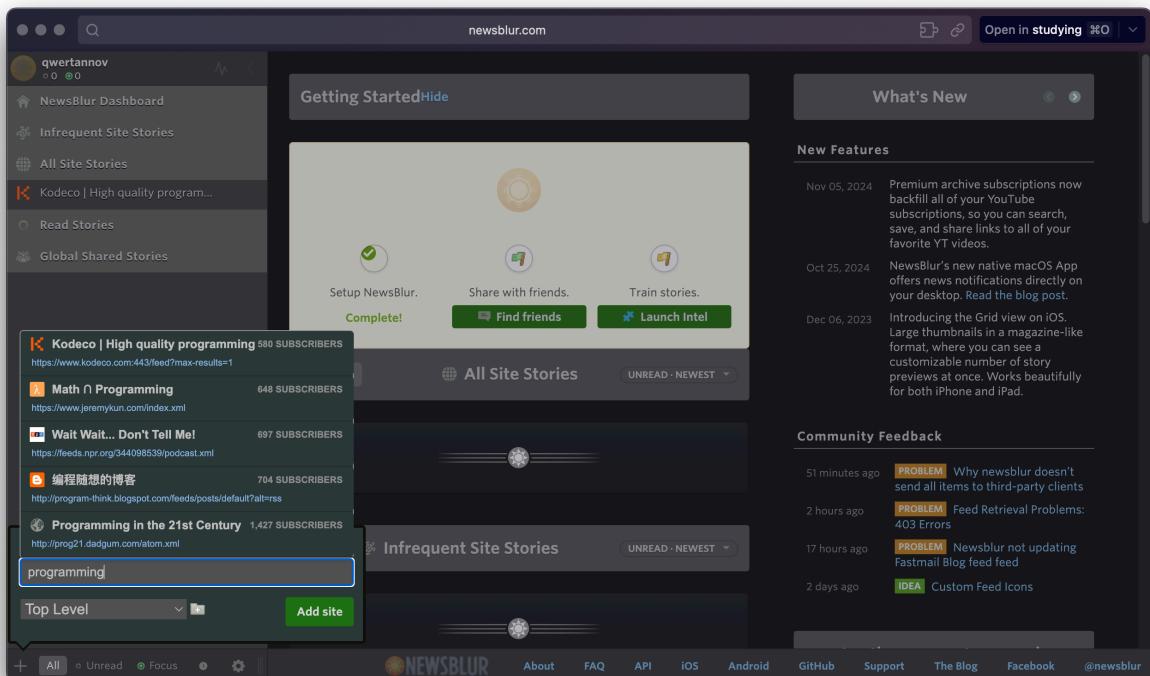


Рисунок 1.8 – Добавання стрічок новин

Після того, як користувач налаштував (додав) бажані стрічки новин, вони показуються у лівій частині сторінки. На сторінці, для перегляду новин, можна побачити список та саму новину (рисунок 1.9).

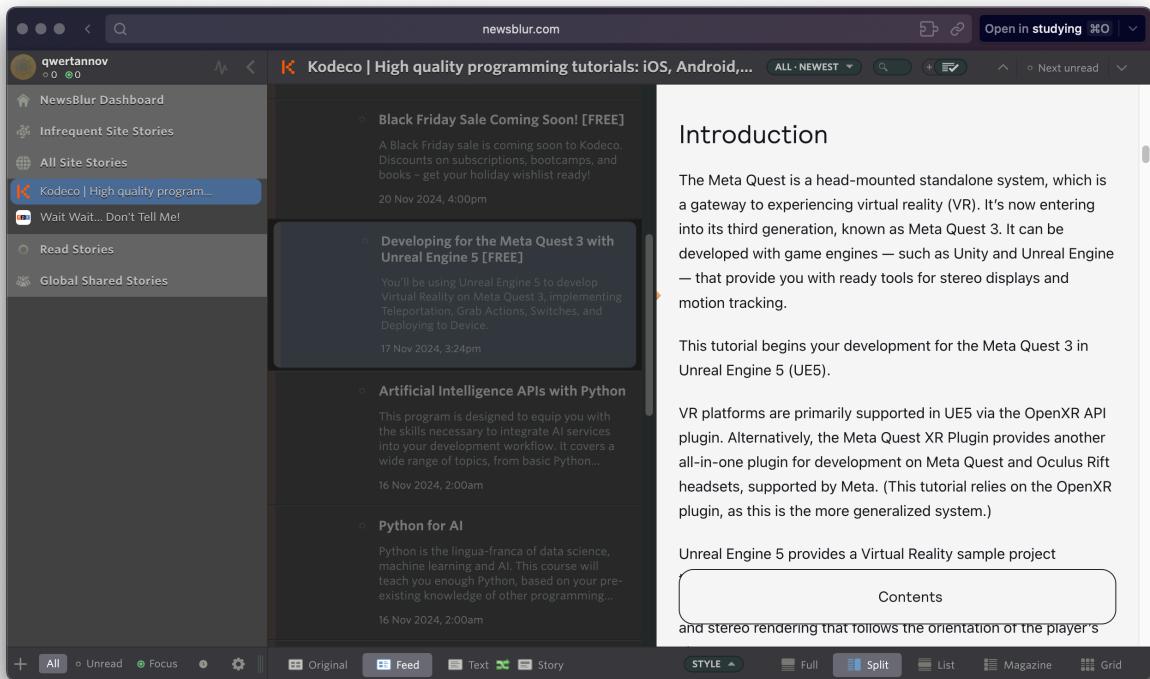


Рисунок 1.9 – Сторінка перегляду новин із відкритою новиною

Отже, Feedly – це популярний RSS-агрегатор, який дозволяє користувачам підписуватися на новини та оновлення з різних джерел (блогів, новинних сайтів, наукових журналів тощо) в одному зручному інтерфейсі. Веб-сервіс надає користувачам можливість організовувати інформацію за категоріями, використовувати теги для сортування та зберігати важливі статті для подальшого перегляду.

NewsBlur – це інший популярний RSS-агрегатор, який дозволяє користувачам підписуватися на новини з різних джерел, організовувати їх в одному місці та читати оновлення в реальному часі. NewsBlur має кілька унікальних особливостей, які роблять його привабливим для користувачів, які шукають більше контролю над своїм інформаційним потоком.

Основні переваги Feedly:

- інтуїтивно зрозумілий інтерфейс і можливість доступу до вмісту на різних пристроях;
- підтримка інтеграції з іншими платформами, такими як Evernote, Pocket, Twitter, LinkedIn;
- користувачі можуть налаштовувати потоки новин, вибираючи тільки ті теми або джерела, які їх цікавлять.

Недоліки:

- обмежена функціональність у безкоштовній версії. Для доступу до розширених можливостей (наприклад, більш детальної аналітики або підтримки більше ніж 100 джерел) необхідно придбати преміум-підписку;
- через велику кількість підписок користувачі можуть відчувати труднощі з організацією та фільтрацією важливих новин.

Основні переваги NewsBlur:

- система дозволяє "навчити" агрегатор, які новини або пости важливі для користувача, а які можна ігнорувати, що покращує досвід читання;
- NewsBlur підтримує функцію попереднього перегляду або повного завантаження статей з деяких джерел;
- можливість створення анонімних акаунтів і читання новин без реєстрації або надання особистих даних;
- NewsBlur здатен обробляти мультимедійний контент, зокрема зображення та відео, без необхідності переходити за посиланнями.

Недоліки:

- безкоштовна версія дозволяє підписатися лише на обмежену кількість джерел (до 64), а також має обмежену функціональність для просунутих користувачів;
- хоча інтерфейс досить гнучкий, деякі новачки можуть знайти його трохи складним для орієнтації, особливо з урахуванням великої кількості налаштувань на застарілим дизайном.

Для порівняння розробки з аналогом можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняння з аналогом

Функціонал	Курсова робота	Feedly	Newsblur	Пояснення
Збір даних через RSS	Так	Так	Так	Використання технології RSS для збору даних
Збір даних через API	Так	Hi	Hi	Використання API для збору даних
Збір даних через веб- скрапінг	Так (у разі необхідності)	Hi	Hi	Використання веб-скрапінгу для збору даних
Редагування зібраного контенту	Так	Hi	Hi	Можливість збору даних і створення новин на їхній основі
Публікація контенту	Так	Hi	Hi	Можливість публікації власного контенту

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Існуючі підходи до збору даних для формування новин можна розділити на три основні категорії: використання RSS-каналів, інтеграція через API платформ, а також веб-скрапінг. Кожен із цих підходів має свої переваги та

недоліки, що впливають на вибір інструментів для реалізації програмного продукту.

а) використання RSS-каналів. RSS є одним із найпоширеніших способів отримання новин із джерел, які підтримують цей формат;

Переваги:

- 1) простота інтеграції;
- 2) широка підтримка форматів багатьма новинними ресурсами;
- 3) відсутність необхідності авторизації чи отримання спеціальних дозволів.

Недоліки:

- 1) обмеження в обсязі доступної інформації (зазвичай доступні лише заголовки та короткий опис);
- 2) не всі сучасні джерела підтримують RSS;
- 3) відсутність інтерактивного оновлення даних у реальному часі.
- б) інтеграція через API. Багато сучасних платформ, зокрема соціальні мережі, надають API для доступу до своїх даних.

Переваги:

- 1) доступ до структурованих даних із можливістю вибору полів і параметрів;
- 2) інтеграція в реальному часі з автоматичним оновленням;
- 3) підтримка розширених функцій, наприклад, пошуку чи фільтрації даних.

Недоліки:

- 1) політика використання API часто обмежує обсяг доступних даних (наприклад, через квоти);
- 2) необхідність отримання ключів доступу та авторизації;
- 3) часті зміни в політиці платформ, які можуть вимагати адаптації до нових умов.
- в) веб-скрапінг. Цей підхід передбачає вилучення інформації безпосередньо з веб-сторінок за допомогою спеціалізованих інструментів;

Переваги:

- 1) можливість отримання даних із джерел, які не підтримують RSS чи API;
- 2) повний контроль над тим, які саме дані зберігаються.

Недоліки:

- 1) ризик порушення політики конфіденційності або правил користування платформами;
- 2) складність підтримки коду через можливі зміни у верстці веб-сторінок;
- 3) затримка через необхідність аналізу та парсингу HTML-коду.

У рамках пропонованого рішення буде використовуватися комбінований підхід. У першу чергу застосовуватимуться RSS-канали та API, а веб-скрапінг – лише у разі відсутності інших можливостей доступу до інформації. Це дозволить досягти балансу між простотою реалізації, відповідністю політикам платформ та гнучкістю отримання даних.

У процесі створення веб-застосунку для роботи з новинами було розглянуто різні алгоритмічні та технічні рішення, які застосовуються для розробки подібних систем. Особливу увагу приділено вибору архітектурних підходів, здатних забезпечити високу ефективність, продуктивність і зручність розробки.

а) архітектурні підходи. Розробка веб-додатків передбачає використання різних архітектур залежно від вимог до масштабу, продуктивності та функціональності. Основні підходи включають:

- 1) монолітна архітектура: Цей підхід полягає у створенні застосунку як єдиного блоку, в якому всі компоненти безпосередньо взаємодіють між собою. Моноліт підходить для невеликих і середніх проектів завдяки простоті реалізації;
- 2) мікросервісна архітектура: Мікросервіси розділяють систему на незалежні компоненти, кожен з яких виконує конкретну

функцію. Такий підхід дозволяє легко масштабувати систему та інтегрувати нові модулі;

3) клієнт-серверна архітектура: Застосовується поділ на клієнську частину (фронтенд) і серверну (бекенд), які взаємодіють через мережу. Це забезпечує гнучкість у розробці та підтримці;

4) архітектура, керована подіями (Event-driven architecture): Цей підхід орієнтований на обробку подій у реальному часі, що дозволяє створювати реактивні та динамічні системи;

5) сервіс-орієнтована архітектура (SOA): SOA будується на незалежних сервісах, які взаємодіють через стандартизовані інтерфейси, що забезпечує гнучкість і модульність системи.

б) технічні платформи для розгортання. Етап розгортання є важливим для забезпечення доступності застосунку для кінцевих користувачів. Розглянуто кілька сучасних рішень:

1) хмарні сервіси: Платформи на зразок Heroku, Vercel і Netlify пропонують простоту використання та автоматизоване розгортання, що робить їх ідеальними для невеликих проектів;

2) великі хмарні платформи: Amazon Web Services (AWS), Google Cloud Platform (GCP) і Microsoft Azure забезпечують високу гнучкість, масштабованість і широкий набір інструментів, таких як бази даних і хмарне зберігання. AWS популярний завдяки широким можливостям, а GCP зручний для команд, які працюють із машинним навчанням чи аналізом даних;

3) віртуальні приватні сервери (VPS): DigitalOcean, Linode і Hetzner дозволяють самостійно налаштовувати сервери, що забезпечує контроль, але вимагає базових знань адміністрування;

4) контейнеризація та оркестрація: Docker забезпечує ізоляцію середовища для застосунків, а Kubernetes дозволяє ефективно масштабувати системи. Ці рішення ідеальні для складних проектів, але вимагають досвіду в DevOps;

5) власні сервери: Найгнучкіше рішення, яке дає повний контроль над конфігурацією, але потребує значних технічних знань.

Кожне з описаних рішень має свої переваги та недоліки, що слід враховувати залежно від вимог до проекту. Для створення веб-застосунку з оптимальним поєднанням функціональності, масштабованості та простоти реалізації може бути обрано монолітну архітектуру з подальшим використанням хмарних платформ або VPS для розгортання.

1.3 Опис бізнес-процесів

Для опису бізнес процесу використовується BPMN [7] модель (рисунок 1.10).

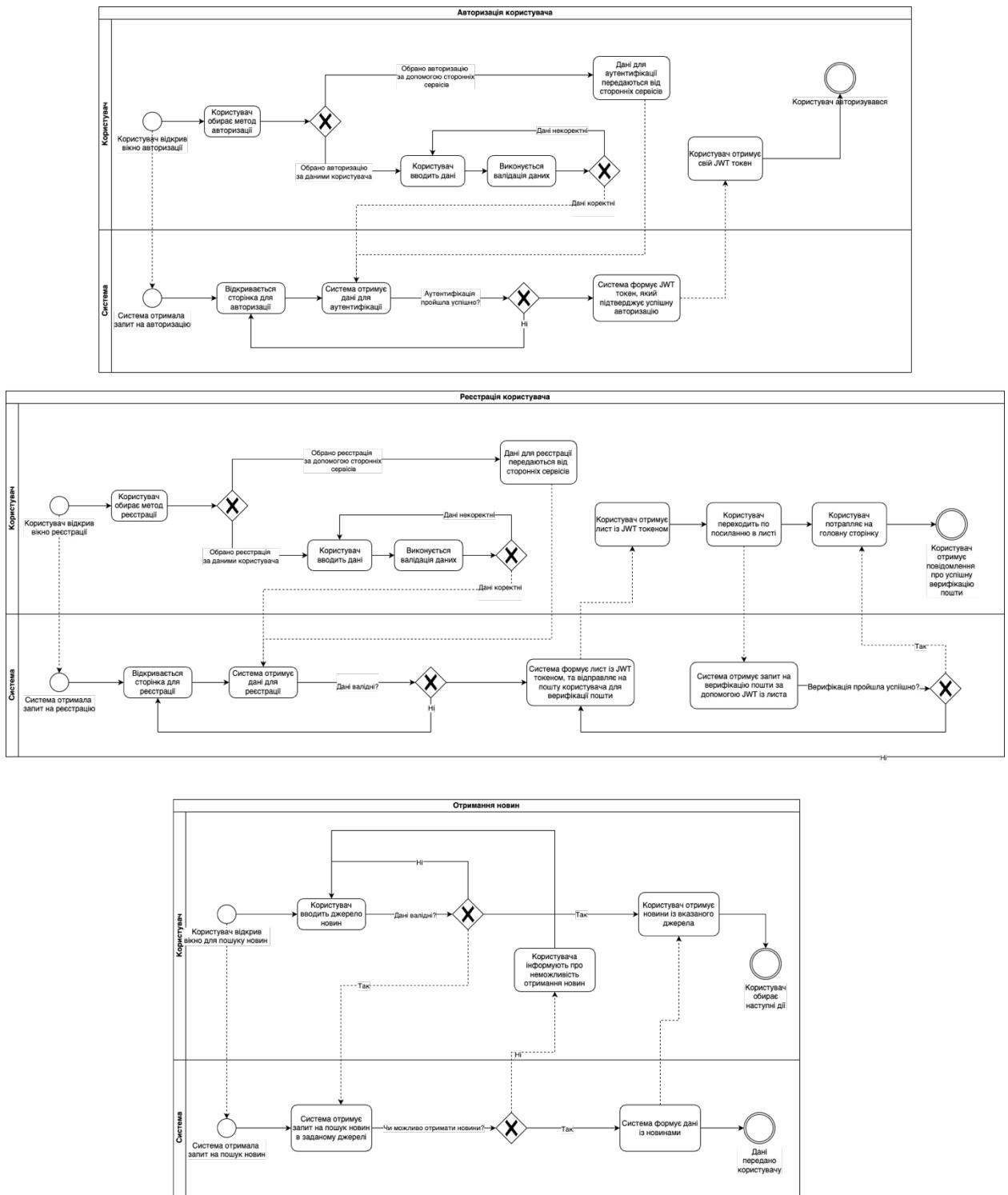


Рисунок 1.10 – Схема бізнес-процесу

Опис послідовності авторизації користувача:

- Користувач відкриває вікно авторизації
- Користувач обирає метод авторизації
- Якщо обрано посторонні сервіси для авторизації
 - а) Дані для аутентифікації передаються сторонніми сервісами

- Якщо обрано авторизацію за даними користувача
 - a) Користувач вводить дані
 - b) Виконується валідація даних
- Якщо дані коректні
- Система отримує дані для аутентифікації
- Якщо аутентифікація не відбулася
 - a) Відкривається сторінка для авторизації
- Система формує JWT [9] токен, який підтверджує успішну авторизацію
 - Користувач отримує JWT токен
 - Користувач авторизувався

Опис послідовності реєстрації користувача:

- Користувач відкриває вікно реєстрації
- Користувач обирає метод реєстрації
- Якщо обрано посторонні сервіси для реєстрації
 - a) Дані для реєстрації передаються сторонніми сервісами
- Якщо обрано реєстрацію за даними користувача
 - a) Користувач вводить дані
 - b) Виконується валідація даних
- Якщо дані коректні
- Система отримує дані для реєстрації
- Якщо дані не валідні
 - a) Відкривається сторінка для реєстрації
- Система формує лист із JWT токеном, який підтверджує успішну авторизацію
 - Користувач отримує лист із JWT токен
 - Користувач переходить по посиланню в листі

- Система отримує запит на верифікацію пошти за допогою JWT із листа
 - Якщо верифікація пройшла невдало
 - а) Система формує лист із JWT токеном, який підтверджує успішну авторизацію
 - Користувач потрапляє на головну сторінку
 - Користувач отримує повідомлення про успішну верифікацію пошти

Опис послідовності отримання новин:

- Користувач відкрив вікно для пошуку новин
- Користувач вводить джерело новин
- Якщо введені дані не валідні
 - а) Користувач вводить джерело новин
- Система отримує запит на пошук новин в заданому джерелі
- Якщо новини із джерела отримати не можливо
 - а) Користувача інформують про неможливість отримання новин
 - б) Користувач вводить джерело новин
- Користувач отримує новини із вказаного джерела
- Користувач обирає наступні дії

Висновки до розділу

У цьому розділі було проведено передпроектне обстеження предметної області, що включало аналіз сучасного стану об'єкта розробки та існуючих рішень. Було визначено основні недоліки поточних рішень, серед яких відсутність інструментів для редактування новинного контенту та велика кількість інформаційного шуму.

Також було розглянуто можливі алгоритмічні та технічні рішення, які допоможуть реалізувати функціонал веб-сервісу, включаючи використання RSS, API та веб-скрапінгу для збору даних, а також сучасних фреймворків

(NestJS [6], Vue.js) і архітектурного підходу RESTful API для побудови системи.

На основі проведеного аналізу було поставлено завдання, сформульовано мету, цілі та перелік функціональних задач для розв'язання в рамках курсової роботи.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є автоматизоване формування новин на основі даних із публічних сторінок у соціальних мережах. Веб-сервіс забезпечує збір інформації з різноманітних джерел (RSS-канали, API соціальних мереж, веб-скрапінг), фільтрацію інформаційного шуму, редагування контенту та створення структурованих новин для подальшої публікації. Користувачі можуть додавати нові джерела, обробляти отримані дані, створювати та публікувати новини через зручний інтерфейс. Адміністратори мають додаткові можливості для управління джерелами та обробки скарг на контент, забезпечуючи якість і відповідність опублікованих матеріалів. Більше функцій можна побачити на рисунку 2.1.

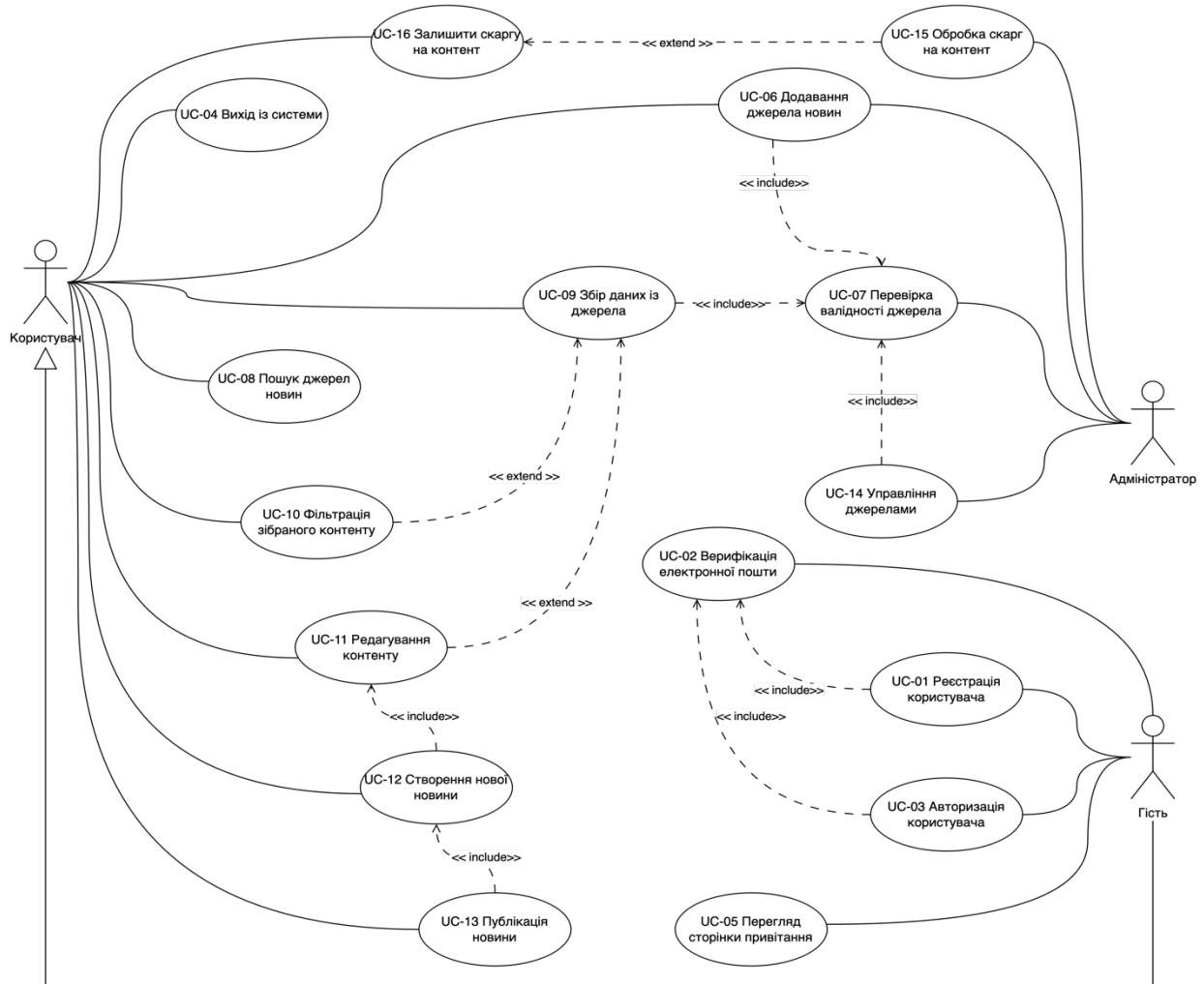


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1 - 2.16 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-01: Реєстрація користувача

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Зареєструвати нового користувача в системі для доступу до функціоналу сервісу
Actors	Гість

Продовження таблиці 2.1

Trigger	Користувач бажає створити обліковий запис для роботи з сервісом
Pre-conditions	-
Flow of Events	Гість відкриває сторінку реєстрації. Гість вводить електронну пошту, пароль, підтвердження пароля та погоджується з умовами сервісу. Гість натискає кнопку "Зареєструватися". Система перевіряє валідність введених даних (формат email, відповідність паролів, мінімальна довжина пароля). Система надсилає лист із посиланням для верифікації електронної пошти (UC-02). Гість перенаправляється на сторінку авторизації.
Extension	Якщо введено некоректні дані (наприклад, email уже зареєстрований або паролі не збігаються), відображається повідомлення про помилку. Некоректне поле підсвічується червоним. Гість повертається до форми реєстрації для виправлення даних.
Post-Condition	Обліковий запис користувача створено, надіслано лист для верифікації, гість перенаправлений на сторінку авторизації.

Таблиця 2.2 - Варіант використання UC-02: Верифікація електронної пошти

Use case name	Реєстрація користувача
Use case ID	UC-02
Goals	Підтвердити електронну пошту користувача для завершення реєстрації
Actors	Гість
Trigger	Користувач отримав лист із посиланням для верифікації після реєстрації

Продовження таблиці 2.2

Pre-conditions	Гість подав заявку на реєстрацію (UC-01)
Flow of Events	Гість отримує електронний лист із JWT-токеном. Гість переходить за посиланням у листі. Система перевіряє валідність токена. Система підтверджує електронну пошту. Гість перенаправляється на головну сторінку або сторінку авторизації.
Extension	Якщо токен недійсний або прострочений, відображається повідомлення про помилку. Гість повертається до сторінки реєстрації для повторного запиту верифікаційного листа.
Post-Condition	Електронна пошта верифікована, користувач може авторизуватися в системі.

Таблиця 2.3 - Варіант використання UC-03: Авторизація користувача

Use case name	Авторизація користувача
Use case ID	UC-03
Goals	Надати користувачу доступ до системи шляхом авторизації
Actors	Гість
Trigger	Користувач бажає увійти в систему для роботи з сервісом
Pre-conditions	Користувач зареєстрований і верифікував електронну пошту (UC-02)
Flow of Events	Гість відкриває сторінку авторизації. Гість вводить електронну пошту та пароль. Гість натискає кнопку "Увійти". Система перевіряє коректність введених даних. Система видає JWT-токен для авторизованого сеансу. Користувач перенаправляється на головну сторінку сервісу.

Продовження таблиці 2.3

Extension	Якщо введено некоректні дані (наприклад, неправильний пароль), відображається повідомлення про помилку. Поле з помилкою підсвічується червоним. Гість повертається до форми авторизації.
Post-Condition	Користувач авторизований, отримує доступ до функціоналу сервісу, перенаправлений на головну сторінку.

Таблиця 2.4 - Варіант використання UC-04: Вихід із системи

Use case name	Вихід із системи
Use case ID	UC-04
Goals	Завершити сесію користувача та анулювати доступ до системи
Actors	Гість
Trigger	Користувач бажає завершити роботу з сервісом
Pre-conditions	Користувач авторизований (UC-03)
Flow of Events	Користувач відкриває меню профілю. Користувач натискає кнопку "Вийти". Система анулює JWT-токен. Користувач перенаправляється на сторінку привітання.
Extension	-
Post-Condition	Сесія користувача завершена, користувач не авторизований, перенаправлений на сторінку привітання.

Таблиця 2.5 - Варіант використання UC-05: Перегляд сторінки привітання

Use case name	Перегляд сторінки привітання
Use case ID	UC-05
Goals	Ознайомити користувача з можливостями сервісу
Actors	Гість
Trigger	Користувач відкриває веб-сервіс для ознайомлення
Pre-conditions	-
Flow of Events	Гість відкриває головну сторінку сервісу. Гість переглядає інформацію про функціонал сервісу, включаючи опис можливостей збору, редагування та публікації новин. Гість може натиснути кнопки для переходу до сторінки реєстрації або авторизації.
Extension	-
Post-Condition	Гість ознайомлений із можливостями сервісу, може продовжити до реєстрації або авторизації.

Таблиця 2.6 - Варіант використання UC-06: Додавання джерела даних

Use case name	Додавання джерела новин
Use case ID	UC-06
Goals	Додати нове джерело для збору даних (RSS, API, URL)
Actors	Користувач, Адміністратор
Trigger	Користувач або адміністратор бажає додати нове джерело для збору новин
Pre-conditions	Користувач або адміністратор авторизований (UC-03)

Продовження таблиці 2.6

Flow of Events	Користувач або адміністратор відкриває розділ "Джерала". Користувач або адміністратор вводить дані джерела (URL, RSS-лінк або API-ключ). Користувач або адміністратор натискає кнопку "Додати". Система виконує перевірку валідності джерела (UC-07). Система додає джерело до списку доступних джерел. Користувач або адміністратор отримує підтвердження про успішне додавання.
Extension	Якщо джерело невалідне (наприклад, RSS-канал не працює), відображається повідомлення про помилку. Користувач або адміністратор повертається до форми введення даних.-
Post-Condition	Джерело додано до системи, готове до використання для збору даних.

Таблиця 2.7 - Варіант використання UC-07: Перевірка валідності джерела

Use case name	Перевірка валідності джерела
Use case ID	UC-07
Goals	Перевірити, чи джерело доступне для збору даних
Actors	Користувач, Адміністратор
Trigger	Користувач відкриває веб-сервіс для ознайомлення
Pre-conditions	Користувач або адміністратор авторизований, введено дані джерела (UC-06 або UC-14)

Продовження таблиці 2.7

Flow of Events	Користувач або адміністратор вводить дані джерела (URL, RSS-лінк, API-ключ). Система перевіряє доступність джерела (наявність RSS-каналу, коректність API-ключа або доступність URL). Система підтверджує валідність джерела. Система зберігає джерело для подальшого використання.
Extension	Якщо джерело невалідне (наприклад, URL недоступний), відображається повідомлення про помилку з описом проблеми. Користувач або адміністратор повертається до введення даних.
Post-Condition	Валідне джерело збережено в системі, готове до використання.

Таблиця 2.8 - Варіант використання UC-08: Пошук джерел новин

Use case name	Пошук джерел новин
Use case ID	UC-08
Goals	Знайти джерела новин за ключовими словами або тегами
Actors	Користувач
Trigger	Користувач бажає знайти нові джерела для додавання
Pre-conditions	Користувач авторизований (UC-03)
Flow of Events	Користувач відкриває розділ "Пошук джерел". Користувач вводить ключові слова або теги для пошуку. Система обробляє запит. Система відображає список відповідних джерел (RSS-канали, сторінки соціальних мереж). Користувач обирає джерело для додавання (UC-06).

Продовження таблиці 2.8

Extension	Якщо джерела не знайдено, відображається повідомлення про відсутність результатів. Користувач може змінити пошуковий запит.
Post-Condition	Користувач отримав список джерел, може додати їх до системи.

Таблиця 2.9 - Варіант використання UC-09: Збір даних із джерела

Use case name	Збір даних із джерела
Use case ID	UC-09
Goals	Автоматично зібрати дані з обраного джерела для подальшої обробки
Actors	Користувач
Trigger	Користувач бажає зібрати дані з доданого джерела
Pre-conditions	Користувач авторизований, джерело додано до системи (UC-06)
Flow of Events	Користувач відкриває розділ "Джерела". Користувач обирає джерело зі списку. Користувач натискає кнопку "Зібрати дані". Система перевіряє валідність джерела (UC-07). Система виконує збір даних через RSS, API або веб-скрапінг. Система відображає зібрані дані (текст, зображення, метадані).

Продовження таблиці 2.9

Extension	Якщо дані не вдалося зібрати (наприклад, через технічну помилку), відображається повідомлення про помилку. Користувач може обрати інше джерело або повторити спробу.
Post-Condition	Дані зібрано, доступні для перегляду, фільтрації або редагування.

Таблиця 2.10 - Варіант використання UC-10: Фільтрація зібраного контенту

Use case name	Фільтрація зібраного контенту
Use case ID	UC-10
Goals	Відфільтрувати інформаційний шум для отримання релевантного контенту
Actors	Користувач
Trigger	Користувач бажає відфільтрувати зібраний контент
Pre-conditions	Дані зібрано з джерела (UC-09)
Flow of Events	Користувач відкриває зібраний контент у розділі "Контент". Користувач обирає параметри фільтрації (ключові слова, дати, джерела, тип контенту). Користувач натискає кнопку "Застосувати фільтр". Система обробляє фільтри. Система відображає відфільтрований контент.
Extension	Якщо відфільтрований контент порожній, відображається повідомлення про відсутність результатів. Користувач може змінити параметри фільтрації.
Post-Condition	Контент відфільтровано, готовий до редагування або створення новин.

Таблиця 2.11 - Варіант використання UC-11: Редагування контенту

Use case name	Редагування контенту
Use case ID	UC-11
Goals	Користувач бажає змінити зібраний контент
Actors	Користувач
Trigger	Користувач бажає відфільтрувати зібраний контент
Pre-conditions	Дані зібрано з джерела (UC-09)
Flow of Events	Користувач відкриває зібраний контент у розділі "Контент". Користувач обирає елемент контенту для редагування. Користувач відкриває редактор контенту. Користувач змінює текст, додає або видаляє зображення, змінює метадані (наприклад, теги, категорії). Користувач натискає кнопку "Зберегти". Система зберігає відредагований контент.
Extension	Якщо контент не відповідає вимогам (наприклад, текст занадто короткий або містить заборонені слова), відображається повідомлення про помилку. Користувач повертається до редагування.
Post-Condition	Контент відредаговано, готовий до створення новин.

Таблиця 2.12 - Варіант використання UC-12: Створення нової новини

Use case name	Створення нової новини
Use case ID	UC-12
Goals	Створити новину на основі відредагованого контенту
Actors	Користувач
Trigger	Користувач бажає сформувати новину для публікації
Pre-conditions	Контент відредаговано (UC-11)
Flow of Events	Користувач відкриває розділ "Створити новину". Користувач додає заголовок, основний текст, зображення та метадані (автор, категорія, теги). Користувач натискає кнопку "Створити". Система перевіряє валідність даних (наявність заголовка, мінімальна довжина тексту). Система зберігає новину в базі даних.
Extension	Якщо дані новини не відповідають вимогам (наприклад, відсутній заголовок), відображається повідомлення про помилку. Користувач повертається до форми створення.
Post-Condition	Новину створено, вона готова до публікації.

Таблиця 2.13 - Варіант використання UC-13: Публікація новини

Use case name	Публікація новини
Use case ID	UC-13
Goals	Користувач бажає опублікувати створену новину
Actors	Користувач
Trigger	Користувач бажає сформувати новину для публікації
Pre-conditions	Новину створено (UC-12)

Продовження таблиці 2.13

Flow of Events	Користувач відкриває створену новину в розділі "Новини". Користувач перевіряє вміст новини. Користувач натискає кнопку "Опублікувати". Система перевіряє новину на відповідність вимогам (відсутність забороненого контенту, відповідність формату). Система публікує новину на веб-сторінці. Користувач отримує підтвердження про успішну публікацію.
Extension	Якщо новина не відповідає вимогам (наприклад, містить заборонений контент), відображається повідомлення про помилку. Користувач повертається до редагування новини (UC-11).
Post-Condition	Новину опубліковано, вона доступна на веб-сторінці сервісу.

Таблиця 2.14 - Варіант використання UC-14: Управління джерелами

Use case name	Управління джерелами
Use case ID	UC-14
Goals	Керувати списком джерел (додавання, редагування, видалення)
Actors	Адміністратор
Trigger	Адміністратор бажає змінити список джерел у системі
Pre-conditions	Адміністратор авторизований (UC-03)

Продовження таблиці 2.6

Flow of Events	Адміністратор відкриває розділ "Управління джерелами". Адміністратор обирає дію (додати, редагувати або видалити джерело). Адміністратор вводить або змінює дані джерела (URL, RSS-лінк, API-ключ). Система виконує перевірку валідності джерела (UC-07). Система зберігає зміни. Адміністратор отримує підтвердження про успішне виконання дії.
Extension	Якщо джерело невалідне або дія неможлива (наприклад, джерело не існує), відображається повідомлення про помилку. Адміністратор повертається до списку джерел.
Post-Condition	Список джерел оновлено (додано, відредаговано або видалено джерело).

Таблиця 2.15 - Варіант використання UC-15: Обробка скарг на контент

Use case name	Обробка скарг на контент
Use case ID	UC-15
Goals	Переглянути та обробити скарги на опублікований контент
Actors	Адміністратор
Trigger	Адміністратор бажає обробити скарги на контент
Pre-conditions	Надіслано скаргу на контент (UC-16)
Flow of Events	Адміністратор відкриває розділ "Скарги". Адміністратор переглядає список скарг. Адміністратор обирає скаргу для обробки. Система відображає деталі скарги та пов'язаний контент. Адміністратор виконує дію (видалити контент, редагувати його або відхилити скаргу). Система зберігає зміни. Адміністратор отримує підтвердження про обробку скарги.

Продовження таблиці 2.15

Extension	Якщо скарг немає, відображається повідомлення про їх відсутність. Адміністратор повертається до головного меню.
Post-Condition	Скаргу оброблено, контент оновлено, видалено або залишено без змін.

Таблиця 2.16 - Варіант використання UC-16: Залишення скарги на контент

Use case name	Залишення скарги на контент
Use case ID	UC-16
Goals	Дозволити користувачу повідомити про проблемний контент
Actors	Користувач
Trigger	Користувач виявляє контент, який вважає невідповідним
Pre-conditions	Користувач авторизований (UC-03), контент опубліковано (UC-13)
Flow of Events	Користувач відкриває опубліковану новину. Користувач натискає кнопку "Поскаржитися". Користувач заповнює форму скарги, вказуючи причину (наприклад, некоректний контент, порушення авторських прав). Користувач натискає кнопку "Надіслати". Система зберігає скаргу для обробки адміністратором (UC-15). Користувач отримує підтвердження про надсилання скарги.
Extension	Якщо форма скарги заповнена некоректно (наприклад, не вказано причину), відображається повідомлення про помилку. Користувач повертається до заповнення форми.
Post-Condition	Скаргу збережено в системі, адміністратор отримує сповіщення для обробки.

2.2 Аналіз системних вимог

Мінімальна конфігурація технічних засобів:

- Тип пристрою: ПК, ноутбук, планшет або смартфон із сучасним веб-браузером [5] актуальних версій (Google Chrome, Firefox, Safari, Edge);
- Центральний процесор: 2-ядерний, з тактовою частотою від 1.4 ГГц;
- Оперативна пам'ять: не менше 2 ГБ;
- Інтернет-з'єднання: стабільне підключення зі швидкістю від 5 Мбіт/с;

Рекомендована конфігурація технічних засобів:

- Тип пристрою: ПК, ноутбук, планшет або смартфон із сучасним веб-браузером [5] (останні версії Chrome, Firefox, Safari, Edge);
- Центральний процесор: 4-ядерний, з тактовою частотою від 2.0 ГГц.
- Оперативна пам'ять: не менше 4 ГБ;
- Інтернет-з'єднання: стабільне підключення зі швидкістю від 20 Мбіт/с.

2.3 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі, кожен із яких відповідає за певний набір функцій. Веб-сервіс для формування новин включає модулі для авторизації, роботи з джерелами, обробки контенту, публікації новин і адміністрування. У таблиці 2.17 наведено загальну модель вимог, у таблиці 2.18 описано функціональні вимоги до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.2.

Таблиця 2.17 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Перегляд сторінки привітання	FR-1	Низький	Низький
2	Система авторизації користувача	FR-2	Високий	Високий
2.1	Реєстрація користувача	FR-3	Високий	Високий
2.2	Верифікація електронної пошти	FR-4	Високий	Високий
2.3	Авторизація користувача	FR-5	Високий	Високий
2.4	Вихід із акаунту	FR-6	Середній	Низький
3	Управління джерелами новин	FR-7	Високий	Середній
3.1	Додавання джерела новин	FR-8	Високий	Середній
3.2	Перевірка валідності джерела	FR-9	Високий	Середній
3.3	Пошук джерел новин	FR-10	Середній	Низький
4	Обробка контенту	FR-11	Високий	Високий
4.1	Збір даних із джерела	FR-12	Високий	Високий

Продовження таблиці 2.17

4.2	Фільтрація зібраного контенту	FR-13	Високий	Середній
4.3	Редагування контенту	FR-14	Високий	Середній
4.4	Створення нової новини	FR-15	Високий	Високий
4.5	Публікація новини	FR-16	Високий	Високий
5	Робота зі скаргами	FR-17	Високий	Високий
5.1	Залишення скарги на контент	FR-18	Високий	Високий
5.2	Обробка скарг на контент	FR-19	Високий	Високий
6	Адміністрування системи	FR-20	Високий	Високий
6.2	Управління джерелами (адмін)	FR-21	Високий	Середній

Таблиця 2.18 – Перелік функціональних вимог

Назва	Опис
FR-1: Перегляд сторінки привітання	Неавторизований користувач бачить веб-сторінку привітання. На ній же можна обрати можливість зареєструватися або авторизуватися. Сторінка містить інформацію про можливості сервісу, включаючи збір, редагування та публікацію новин.
FR-2: Система авторизації користувача	Система забезпечує авторизацію користувачів для доступу до функціоналу сервісу. Включає реєстрацію, верифікацію електронної пошти, авторизацію та вихід із системи. Забезпечує безпечний доступ через JWT-токени.
FR-3: Реєстрація користувача	Гість може створити обліковий запис, ввівши унікальний email, пароль і погодившись із умовами сервісу. Система надсилає верифікаційний лист для підтвердження email.

Продовження таблиці 2.18

FR-4: Верифікація електронної пошти	Гість підтверджує email, переходячи за посиланням із верифікаційним JWT-токеном, отриманим у листі. Після підтвердження користувач може авторизуватися.
FR-5: Авторизація користувача	Користувач входить у систему, ввівши email і пароль. Система видає JWT-токен і перенаправляє на головну сторінку сервісу.
FR-6: Вихід із акаунту	Авторизований користувач може завершити сесію, натиснувши кнопку "Вийти". Система анулює JWT-токен і перенаправляє на сторінку привітання.
FR-7: Управління джерелами новин	Система дозволяє користувачам і адміністраторам додавати, перевіряти та шукати джерела новин (RSS-канали, API соціальних мереж, URL) для збору даних.
FR-8: Додавання джерела новин	Користувач або адміністратор додає нове джерело, вказавши URL, RSS-лінк або API-ключ. Система перевіряє валідність джерела перед збереженням.
FR-9: Перевірка валідності джерела	Система перевіряє доступність джерела (наявність RSS-каналу, коректність API-ключа, доступність URL) перед його додаванням або збором даних.
FR-10: Пошук джерел новин	Користувач може шукати джерела новин за ключовими словами або тегами. Система повертає список відповідних джерел для додавання.
FR-11: Обробка контенту	Система забезпечує повний цикл обробки контенту: збір даних із джерел, фільтрацію інформаційного шуму, редагування, створення та публікацію новин.
FR-12: Збір даних із джерела	Система автоматично збирає дані з обраного джерела через RSS, API або веб-скрапінг, надаючи текст, зображення та метадані для обробки.

Продовження таблиці 2.18

FR-13: Фільтрація зібраного контенту	Користувач може фільтрувати зібраний контент за ключовими словами, датами або джерелами для видалення інформаційного шуму.
FR-14: Редагування контенту	Користувач редагує зібраний контент, змінюючи текст, додаючи або видаляючи зображення, а також додаючи метадані (теги, категорії).
FR-15: Створення нової новини	Користувач створює новину, додаючи заголовок, текст, зображення та метадані на основі відредагованого контенту.
FR-16: Публікація новини	Користувач публікує створену новину на веб-сторінці сервісу після перевірки на відповідність вимогам.
FR-17: Робота зі скаргами	Система дозволяє користувачам залишати скарги на проблемний контент і адміністраторам обробляти їх, видаляючи або редагуючи контент.
FR-18: Залишення скарги на контент	Користувач може залишити скаргу на опубліковану новину, вказавши причину (наприклад, некоректний контент). Скарга зберігається для обробки.
FR-19: Обробка скарг на контент	Адміністратор переглядає скарги, аналізує пов'язаний контент і виконує дії (видалення, редагування або відхилення скарги).
FR-20: Адмініструванн я системи	Система надає адміністраторам інструменти для управління джерелами та контентом, включаючи додавання, редагування та видалення джерел.

Продовження таблиці 2.18

FR-21: Управління джерелами (адмін)	Адміністратор може додавати, редагувати або видаляти джерела новин, перевіряючи їх валідність через систему.
--	--

ID Вимоги	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
FR-1					X											
FR-2	X	X	X	X												
FR-3	X															
FR-4			X													
FR-5				X												
FR-6					X											
FR-7						X	X	X								
FR-8						X										
FR-9							X									
FR-10								X								
FR-11									X	X	X	X	X			
FR-12									X							
FR-13										X						
FR-14										X						
FR-15											X					
FR-16												X				
FR-17													X	X		
FR-18														X		
FR-19														X		
FR-20														X		
FR-21														X		

Рисунок 2.2 – Матриця трасування вимог

2.4 Розроблення нефункціональних вимог

Нефункціональні вимоги визначають якісні характеристики веб-сервісу для формування новин на основі публічних сторінок у соціальних мережах. Вони забезпечують стабільну, безпечну та зручну роботу системи для гостей, користувачів і адміністраторів, а також підтримують обробку великих обсягів даних із соціальних мереж. Вимоги поділено на категорії, кожна з яких описує специфічні аспекти функціонування системи.

2.4.1 Продуктивність

Система повинна забезпечувати швидку обробку запитів користувачів і ефективну роботу з даними, включаючи збір, фільтрацію, редагування та публікацію контенту. Час відповіді на запити має бути мінімальним навіть при високому навантаженні, щоб забезпечити комфортну роботу для всіх користувачів.

- Час відповіді на запит (авторизація, пошук джерел, публікація новин) не повинен перевищувати 2 секунди для 95% запитів при навантаженні до 1000 одночасних користувачів.
- Збір даних із джерел (RSS, API, веб-скрапінг) має виконуватися не довше 5 секунд на одне джерело при стандартному обсязі даних (до 10 МБ).

2.4.2 Безпека

Система повинна гарантувати захист даних користувачів, контенту та процесів авторизації від несанкціонованого доступу, атак і витоків інформації.

- Усі дані користувачів (email, паролі) шифруються за допомогою протоколу HTTPS і алгоритму хешування паролів (наприклад, bcrypt).
- JWT-токени [9] для авторизації мають обмежений час дії (не більше 24 годин) і захищені від модифікації.

- Система запобігає SQL-ін'єкціям [10] і XSS-атакам шляхом валідації та екранування всіх вхідних даних.

2.4.3 Доступність

Система повинна бути доступною для користувачів у будь-який час із мінімальними перебоями в роботі, забезпечуючи стабільний доступ до всіх функцій сервісу.

- Рівень доступності системи становить не менше 99.9% часу на місяць (uptime).
- У разі збою система автоматично перенаправляє запити на резервні сервери протягом 30 секунд.
- Сервіс підтримує безперебійну роботу при піковому навантаженні до 5000 одночасних користувачів.

2.4.4 Масштабованість

Система повинна підтримувати зростання кількості користувачів, джерел і обсягу оброблюваних даних без значного зниження продуктивності.

- Система здатна масштабуватися горизонтально шляхом додавання нових серверів для обробки запитів.
- База даних підтримує розподілене зберігання даних для обробки до 1 ТБ контенту.
- Система забезпечує стабільну роботу при додаванні до 100 нових джерел на день.

2.4.5 Зручність використання

Інтерфейс системи має бути інтуїтивно зрозумілим і зручним для користувачів із різним рівнем технічної підготовки, включаючи гостей, користувачів і адміністраторів.

– Час, необхідний для виконання основних дій (реєстрація, додавання джерела, створення новини), не перевищує 2 хвилин для нового користувача.

– Інтерфейс адаптивний і коректно відображається на пристроях із роздільною здатністю від 320x480 до 1920x1080 пікселів. –

Система підтримує локалізацію інтерфейсу українською та англійською мовами.

2.4.6 Надійність

Система повинна забезпечувати стабільну роботу без критичних збоїв і втрати даних, включаючи збереження контенту, джерел і скарг.

– У разі збою система автоматично створює резервні копії бази даних кожні 24 години.

– Система відновлює дані після збою протягом 10 хвилин без втрати інформації.

– Помилки обробки даних (наприклад, невалідне джерело) не призводять до припинення роботи сервісу.

2.4.7 Сумісність

Система повинна бути сумісною з різними платформами, браузерами та операційними системами, щоб забезпечити доступність для широкої аудиторії.

– Сервіс підтримує сучасні браузери актуальних версій (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge).

– Система сумісна з операційними системами Windows 10+, macOS 11+, Linux (Ubuntu 20.04+), Android 10+, iOS 15+.

- API системи відповідає стандартам REST і підтримує інтеграцію з зовнішніми сервісами (наприклад, Telegram, Instagram).

2.4.8 Портативність

Система повинна бути легко переносимою на нові апаратні або програмні платформи без значних змін у коді.

- Код системи контейнеризовано (наприклад, за допомогою Docker) для швидкого розгортання на різних серверах.
- Система підтримує розгортання в хмарних середовищах (AWS, Google Cloud, Azure) із мінімальними налаштуваннями.

2.5 Постановка задачі

Мета роботи: Розробити веб-сервіс, який дозволяє автоматизовано збирати дані з публічних джерел, таких як Telegram-канали, сторінки в Instagram та інші соціальні мережі, аналізувати їх, редагувати контент і створювати на його основі новини для подальшої публікації.

Цілі:

- Забезпечити зручний механізм інтеграції з джерелами інформації за допомогою RSS, API або веб-скрапінгу;
- Надати функціональність для редагування зібраного контенту та створення нових матеріалів;
- Забезпечити можливість публікації створеного контенту на веб-сторінці новин;

Задачі для розв'язання:

- Реалізувати механізм збору даних із джерел, використовуючи відповідні технології;

- a) RSS для структурованих джерел;
 - б) API для платформ, що надають програмні інтерфейси;
 - в) веб-скрапінг для платформ без RSS або API (з урахуванням політики використання).
- Забезпечити інструменти для аналізу та фільтрації зібраного контенту:
- а) видалення інформаційного шуму;
 - б) виділення релевантних даних.
- Розробити редактор контенту, який дозволяє:
- а) Змінювати текстові й графічні матеріали;
 - б) Додавати нові елементи (зображення, заголовки, метадані).
- Реалізувати веб-інтерфейс для взаємодії з користувачами:
- а) зручний UI для перегляду, редагування та публікації матеріалів;
 - б) функціонал пошуку та категоризації новин.
- Забезпечити надійну архітектуру для зберігання даних:
- а) розробити структуру бази даних для зберігання контенту;
 - б) реалізувати механізми безпечного доступу до даних.
- Тестування і оптимізація роботи системи:
- а) перевірка роботи механізмів збору даних із різних джерел;
 - б) забезпечення швидкої та стабільної роботи веб-застосунку.

Очікуваним результатом є функціональний веб-сервіс, який дозволяє редакторам створювати якісні новини на основі інформації, зібраної з різних джерел, і публікувати їх на новинній сторінці.

Висновки до розділу

У розділі 2 виконано комплексний аналіз і розробку вимог до веб-сервісу для формування новин на основі публічних сторінок у соціальних мережах, що дозволило сформувати чітке бачення функціоналу та характеристик

системи. Робота охопила чотири ключові підпункти, кожен із яких вніс вагомий внесок у визначення специфікацій проекту.

У підпункті 2.1 описано головний функціонал сервісу, який полягає в автоматизованому зборі, обробці, редагуванні та публікації новин із соціальних мереж. Розроблено діаграму варіантів використання (Рисунок 2.1), що відображає взаємодію акторів (Гість, Користувач, Адміністратор) із системою через 16 варіантів використання (UC-01–UC-16). Детальний опис кожного варіанту використання представлено в таблицях 2.1–2.16, що включають цілі, акторів, передумови, потік подій, розширення та постумови, забезпечуючи повне розуміння функціональних сценаріїв.

Підпункт 2.2 присвячено аналізу системних вимог, де визначено мінімальну та рекомендовану конфігурацію технічних засобів для клієнтських пристройів.

У підпункті 2.3 розроблено функціональні вимоги, структуровані за модулями системи: авторизація, управління джерелами, обробка контенту, скарги та адміністрування. Загальна модель вимог (Таблиця 2.17) включає 21 вимогу (FR-1–FR-21) із пріоритетами та ризиками. Перелік вимог представлено в таблиці 2.16, а детальний опис — у таблиці 2.18. Матриця трасування (Рисунок 2.3) пов’язує вимоги з UC, забезпечуючи відстеження їх реалізації.

Підпункт 2.4 зосереджено на нефункціональних вимогах, структурованих за категоріями: продуктивність, безпека, доступність, масштабованість, зручність використання, надійність, сумісність і портативність (2.4.1–2.4.8). Кожна категорія включає конкретні вимоги, такі як час відповіді до 2 секунд, шифрування даних через HTTPS, 99.9% доступності та підтримка сучасних браузерів, що забезпечують якісну роботу сервісу.

Таким чином, розділ 2 створив міцну основу для подальшої розробки веб-сервісу, чітко визначивши його функціональні та нефункціональні характеристики, а також системні вимоги. Отримані результати забезпечують

повне розуміння потреб системи та її взаємодії з користувачами, що є ключовим для успішної реалізації проєкту.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Архітектура веб-сервісу для формування новин на основі публічних сторінок у соціальних мережах розроблена як монолітна з використанням принципів Domain-Driven Design [8] (DDD). Монолітна архітектура обрана для спрощення розробки, розгортання та підтримки системи на початкових етапах, враховуючи обмежену складність проєкту та потребу в централізованому управлінні функціоналом. DDD застосовується для структуризації коду навколо бізнес-доменів (авторизація, управління джерелами, обробка контенту, скарги, адміністрування), що забезпечує чітке розділення відповідальностей і полегшує майбутнє масштабування.

Система включає єдиний серверний додаток, який обробляє всі запити користувачів, інтегрується з зовнішніми джерелами даних (RSS, API соціальних мереж, веб-скрапінг) і взаємодіє з базою даних для зберігання контенту, джерел і скарг. Архітектура представлена через діаграми C4 Model на трьох рівнях: System Context (L1), Container (L2) і Component (L3), які деталізують взаємодію системи з зовнішніми акторами, її основні контейнери та внутрішні компоненти.

Діаграма рівня L1 (System Context) зображена на рис. 3.1. На високому рівні веб-сервіс представлено як єдину систему, що взаємодіє з трьома типами акторів та двома зовнішніми системами. Гості переглядають сторінку привітання, реєструються або авторизуються, користувачі додають джерела, обробляють контент, публікують новини та залишають скарги, а адміністратори керують джерелами й обробляють скарги. Система отримує дані з соціальних мереж, таких як Telegram та Instagram, через REST API, а також із RSS-каналів у форматі XML [7] або JSON. Ця структура відображає основні взаємодії через HTTPS для акторів і API для зовнішніх джерел, забезпечуючи просте розуміння контексту системи.

Веб-сервіс формування новин - System Context

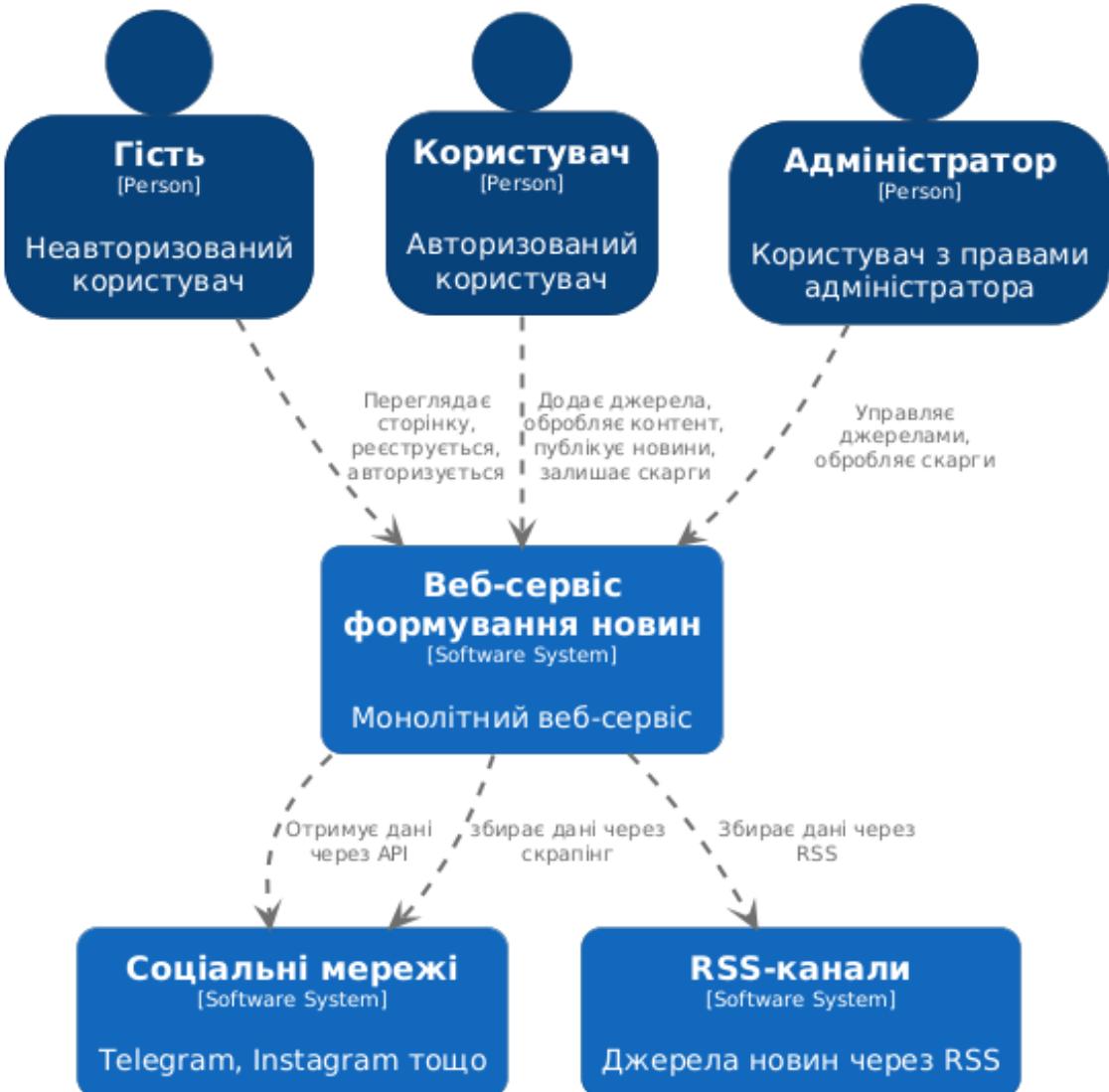


Рисунок 3.1 – C4 Level 1 System Context

Опис діаграми L1:

- Актори:
 - а) Гість – переглядає сторінку привітання, реєструється або авторизується;
 - б) Користувач – виконує основні дії: додає джерела, обробляє контент, створює та публікує новини, залишає скарги;
 - в) Адміністратор – управляє джерелами та обробляє скарги.
- Система: Веб-сервіс, що забезпечує весь функціонал.

- Зовнішні системи:
 - a) Соціальні мережі (наприклад, Telegram, Instagram) – джерела даних через API або скрапінг;
 - б) RSS-канали – джерела текстових даних.
- Зв’язки:
 - a) Актори взаємодіють із системою через HTTPS-запити;
 - б) Система отримує дані від соціальних мереж (REST API) і RSS-каналів (XML [7] /JSON).

Внутрішня структура веб-сервісу (рис. 3.2) розкриває три основні контейнери: веб-додаток, серверний додаток і база даних. Веб-додаток, побудований на Vue.js із TypeScript і стилізований через Tailwind CSS, забезпечує адаптивний інтерфейс для всіх акторів, дозволяючи виконувати дії через HTTPS-запити. Серверний додаток, реалізований на NestJS [6] (Node.js, TypeScript), обробляє бізнес-логіку, інтегрується з зовнішніми джерелами та взаємодіє з базою даних через SQL. База даних на PostgreSQL зберігає інформацію про користувачів, джерела, контент, новини та скарги. Веб-додаток комунікує із серверним через REST API, а серверний додаток отримує дані від соціальних мереж і RSS-каналів, використовуючи відповідні протоколи.

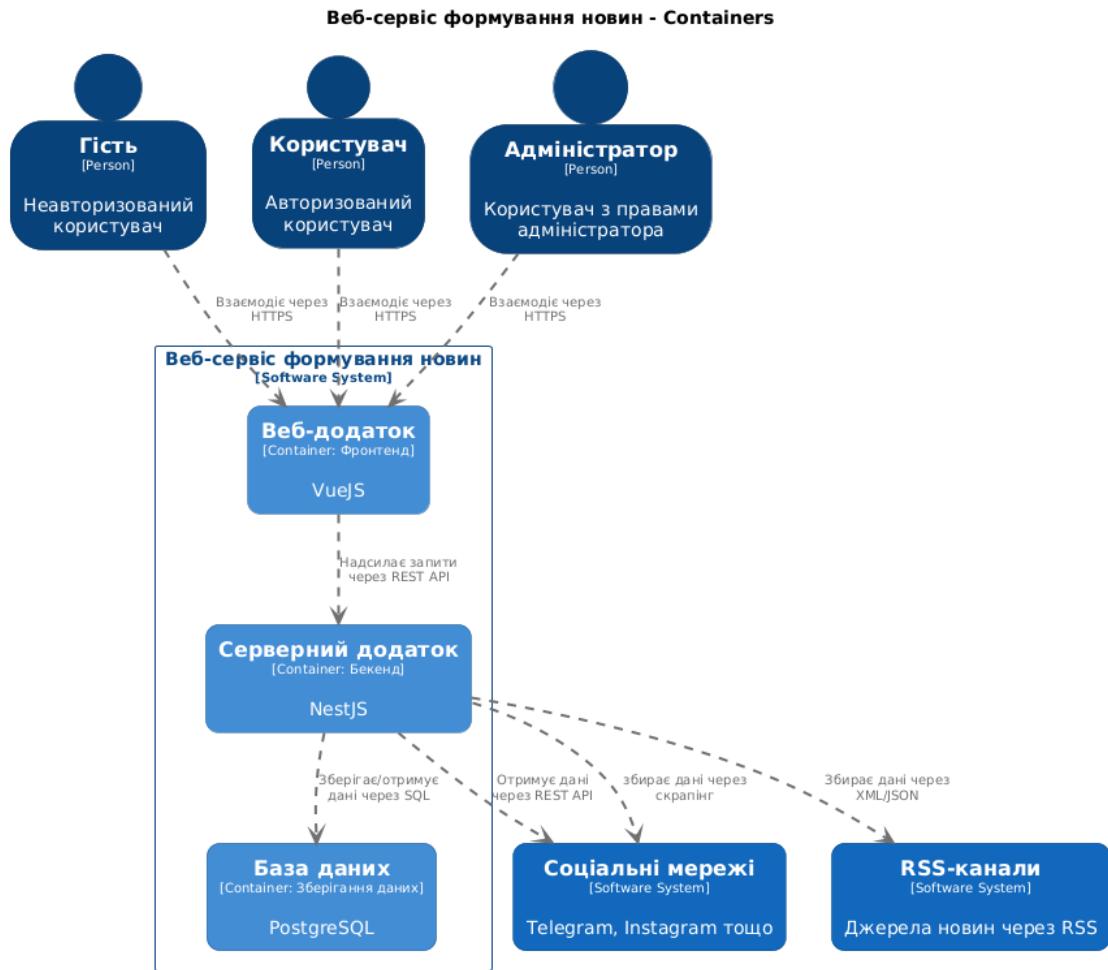


Рисунок 3.2 – C4 Level 2 Containers

Опис діаграми L2:

- Контейнери:
 - a) Веб-додаток: Клієнтський інтерфейс (HTML, CSS, Typescript, VueJS), що працює в браузері користувача. Забезпечує взаємодію через адаптивний UI;
 - b) Серверний додаток: Монолітний бекенд (NestJS [6]), що обробляє бізнес-логіку, інтеграцію з джерелами та запити до бази даних;
 - c) База даних: Реляційна СУБД (PostgreSQL) для зберігання даних користувачів, джерел, контенту та скарг.
- Зв’язки:
 - a) Актори взаємодіють із веб-додатком через HTTPS;
 - b) Веб-додаток надсилає запити до серверного додатка через REST API;

- в) Серверний додаток взаємодіє з базою даних через SQL-запити;
- г) Серверний додаток отримує дані від соціальних мереж (REST API) і RSS-каналів (XML [7] /JSON) або сам скрапить дані.
- Технології:
 - а) Веб-додаток: VueJS, Tailwind CSS;
 - б) Серверний додаток: NestJS [6], REST API;
 - в) База даних: PostgreSQL;

Діаграма рівня L3 (Component) деталізує внутрішню структуру серверного додатка, розбиваючи його на компоненти відповідно до принципів DDD. Серверний додаток деталізовано на компоненти, структуровані за принципами DDD, кожен із яких відповідає окремому домену. Контролер API, побудований на NestJS [6] Controllers, обробляє вхідні REST-запити та делегує їх відповідним сервісам. Сервіс авторизації управляє реєстрацією, верифікацією email, авторизацією та виходом, використовуючи JWT-токени. Сервіс джерел відповідає за додавання, перевірку та пошук джерел новин. Сервіс контенту забезпечує збір, фільтрацію, редагування, створення та публікацію новин. Сервіс скарг обробляє залишення та розгляд скарг, а сервіс адміністрування надає функції управління джерелами для адміністраторів. Інтеграційний модуль, що використовує бібліотеку axios, забезпечує взаємодію із зовнішніми джерелами через REST API та парсинг RSS у форматі XML [7] /JSON. Усі сервіси взаємодіють із базою даних через TypeORM, а сервіси джерел і контенту використовують інтеграційний модуль для збору даних.

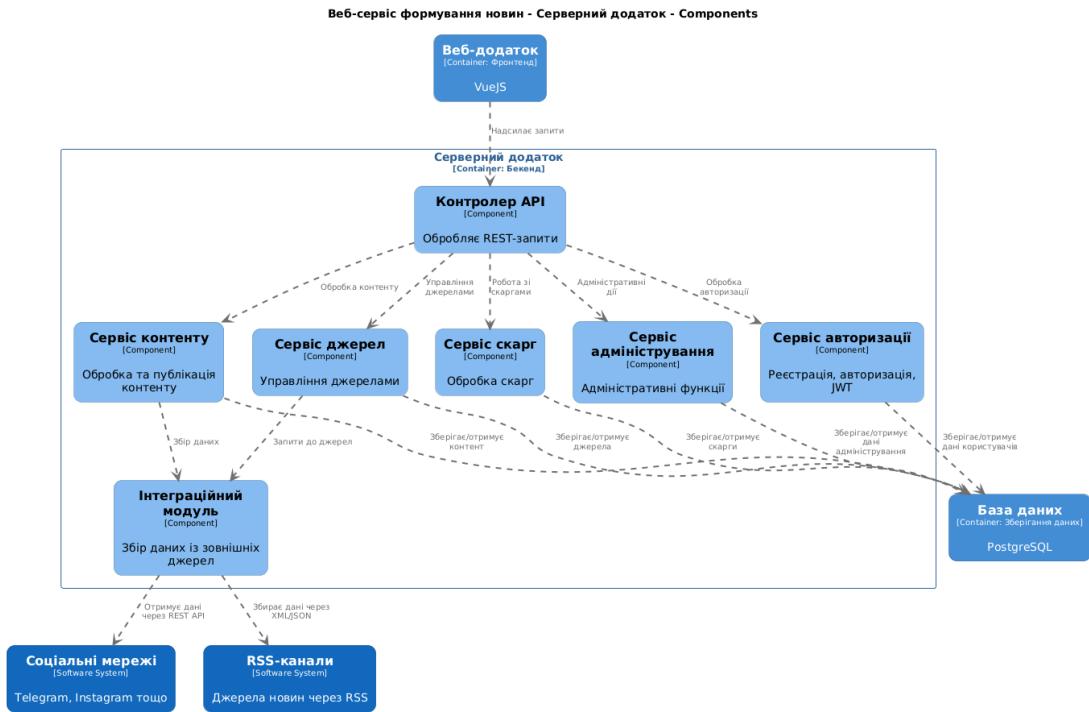


Рисунок 3.3 – C4 Level 3 Components

Опис діаграми L3:

- Компоненти (DDD-домени):
 - a) Контролер API – обробляє вхідні REST-запити від веб-додатка та делегує їх відповідним сервісам;
 - b) Сервіс авторизації – відповідає за реєстрацію, верифікацію email, авторизацію та вихід (FR-2–FR-6, UC-01–UC-04). Використовує JWT-токени;
 - c) Сервіс джерел – керує додаванням, перевіркою та пошуком джерел (FR-7–FR-10, UC-06–UC-08);
 - d) Сервіс контенту – забезпечує збір, фільтрацію, редагування, створення та публікацію контенту (FR-11–FR-16, UC-09–UC-13);
 - e) Сервіс скарг – обробляє залишення та розгляд скарг (FR-17–FR-19, UC-15–UC-16);
 - f) Сервіс адміністрування – надає функції управління джерелами для адміністраторів (FR-20–FR-21, UC-14);
 - g) Інтеграційний модуль – забезпечує взаємодію із зовнішніми джерелами через REST API (соціальні мережі) та XML [7] /JSON (RSS).

- Зв'язки:
 - а) Веб-додаток надсилає запити до контролера API (REST);
 - б) Контролер API викликає відповідні сервіси залежно від запиту;
 - в) Кожен сервіс взаємодіє з базою даних через SQL для зберігання/отримання даних;
 - г) Сервіси джерел і контенту використовують інтеграційний модуль для доступу до зовнішніх джерел.
- DDD-принципи:
 - а) Кожен сервіс є окремим Bounded Context, що відповідає домену (авторизація, джерела, контент, скарги, адміністрування);
 - б) Entities (наприклад, User, Source, News) і Value Objects (наприклад, ContentMetadata) чітко визначені в кожному домені;
 - в) Aggregates (наприклад, News з Content і Metadata) забезпечують консистентність даних;
 - д) Repositories (наприклад, UserRepository, SourceRepository) абстрагують доступ до бази даних;
 - е) Domain Services (наприклад, ContentFilteringService) інкапсулюють складну логіку.

3.2 Важливі архітектурні рішення

Інтеграція із зовнішніми системами здійснюється через REST API для соціальних мереж, таких як Telegram (Bot API) та Instagram (Graph API), що використовують HTTPS і JSON для надійної передачі даних. RSS-канали обробляються через HTTP-запити з парсингом XML [7] /JSON, що є стандартом для новинних джерел. Ці протоколи обрано через їхню універсальність і підтримку безпеки. Для виконання нефункціональних вимог продуктивність забезпечується асинхронною обробкою в NestJS [6] і кешуванням запитів через Redis, що гарантує час відповіді до 2 секунд.

Безпека досягається завдяки JWT-токенам, хешуванню паролів через bcrypt і захисту від ін'єкцій через TypeORM. Доступність на рівні 99.9% підтримується хмарним розгортанням із резервними серверами, а масштабованість – через контейнеризацію (Docker) і шардування PostgreSQL. Зручність використання гарантується адаптивним інтерфейсом Vue.js і локалізацією, а надійність – щоденным резервним копіюванням і обробкою помилок у NestJS [6]. Сумісність із сучасними браузерами та ОС забезпечується стандартами Vue.js, а портативність – контейнеризацією та TypeScript. PostgreSQL обрано як основну базу даних через її підтримку реляційних даних, транзакцій і масштабованості, що ідеально відповідає структурі даних системи, відкидаючи альтернативи, такі як MongoDB, через потребу в транзакційній консистентності.

3.3 Обґрунтування вибору засобів розробки

Для розробки веб-сервісу формування новин на основі публічних сторінок у соціальних мережах обрано технологічний стек, що включає TypeScript як основну мову програмування, Vue.js для клієнтської частини, NestJS [6] для серверної частини, PostgreSQL як базу даних, а також допоміжні бібліотеки та інструменти, такі як Tailwind CSS, axios, xml2js, TypeORM, Docker і Visual Studio Code. Вибір цього стеку зумовлений вимогами до продуктивності, безпеки, масштабованості, зручності використання та сумісності, визначеними у нефункціональних вимогах, а також потребою в швидкій розробці та підтримці монолітної архітектури з принципами Domain-Driven Design.

TypeScript обрано як основну мову програмування завдяки строгій типізації, яка зменшує кількість помилок під час розробки та полегшує підтримку коду, що критично для складних бізнес-доменів, таких як авторизація, обробка контенту та адміністрування. Порівняно з JavaScript, TypeScript забезпечує кращу масштабованість проекту та інтеграцію з сучасними фреймворками, такими як Vue.js і NestJS [6]. Альтернативи, такі як

Python або Java, були відхилені: Python, хоча й популярний для веб-розробки (наприклад, Django), менш ефективний для асинхронної обробки великої кількості запитів, а Java потребує більше ресурсів і ускладнює швидке розгортання. TypeScript, завдяки своїй універсальності та підтримці Node.js, ідеально відповідає потребам веб-сервісу.

Для клієнтської частини Vue.js обрано через його легкість, гнучкість і швидке створення адаптивних інтерфейсів, що відповідає вимозі зручності використання. Vue.js у комбінації з TypeScript і Pinia для управління станом забезпечує модульну структуру та інтуїтивний API, що спрощує розробку складних форм, таких як додавання джерел чи редагування новин. Порівняно з React, Vue.js має менший поріг входження та кращу продуктивність для невеликих і середніх проектів, тоді як Angular був відхилений через надмірну складність для монолітної системи. Tailwind CSS використано для стилізації завдяки його утилітарному підходу, який дозволяє швидко створювати адаптивні дизайни без написання великої кількості CSS-коду. Альтернативи, такі як Bootstrap, менш гнучкі для кастомізації, а чисте CSS потребує більше часу на розробку.

Серверна частина реалізована на NestJS [6], фреймворку для Node.js, який підтримує TypeScript і модульну архітектуру, ідеальну для DDD. NestJS [6] забезпечує вбудовану підтримку REST API, асинхронну обробку запитів і захист від ін'єкцій через Pipes, що відповідає вимогам продуктивності та безпеки. У порівнянні з Express.js, NestJS [6] пропонує кращу структуризацію коду та вбудовані інструменти для тестування, тоді як Fastify, хоча й швидший, менш зрілий для складних проектів. Django (Python) був відхилений через меншу ефективність у реальному часі, а Spring (Java) — через складність розгортання. Для взаємодії з базою даних використано TypeORM, який забезпечує зручну роботу з PostgreSQL і підтримує транзакції, необхідні для цілісності даних. Альтернатива, Prisma, хоча й сучасна, менш гнучка для складних реляційних моделей.

PostgreSQL обрано як базу даних через її підтримку реляційних даних, транзакцій, індексів і масштабованості, що ідеально відповідає потребам зберігання користувачів, джерел, контенту, новин і скарг. Порівняно з MongoDB, PostgreSQL краще забезпечує транзакційну консистентність, необхідну для обробки скарг і авторизації, тоді як MySQL був відхищений через меншу підтримку JSONB для метаданих контенту. Redis використано як допоміжну систему для кешування запитів до зовнішніх API, що підвищує продуктивність.

Для інтеграції із зовнішніми джерелами використано бібліотеку axios, яка забезпечує просту та надійну роботу з REST API соціальних мереж (Telegram, Instagram). Порівняно з fetch, axios має кращу підтримку обробки помилок і конфігурації. Для парсингу RSS-каналів обрано xml2js, який ефективно обробляє XML [7] і JSON, тоді як альтернатива, feedparser, менш сумісна з TypeScript. Ці бібліотеки відповідають вимогам універсальності та безпеки.

Розробка ведеться в інтегрованому середовищі Visual Studio Code, яке підтримує TypeScript, ESLint, Prettier і плагіни для роботи з Vue.js і NestJS [6]. VS Code обрано через його легкість, широку підтримку розширень і безкоштовність, порівняно з платними IDE, такими як WebStorm, які надають подібний функціонал, але потребують додаткових витрат. Для тестування використано Jest, що забезпечує швидке написання юніт-тестів для NestJS [6] і Vue.js, а для контейнеризації — Docker, який гарантує портативність і сумісність із хмарними платформами. CI/CD-пайплайн налаштовано через GitHub Actions для автоматизації розгортання.

Таким чином, обраний технологічний стек — TypeScript, Vue.js, NestJS [6], PostgreSQL, Tailwind CSS, axios, xml2js, TypeORM, Docker і VS Code — забезпечує оптимальний баланс між продуктивністю, безпекою, масштабованістю та швидкістю розробки. Порівняльний аналіз альтернатив підтверджив, що ці засоби найкраще відповідають вимогам проекту,

дозволяючи ефективно реалізувати монолітну архітектуру з DDD і забезпечити майбутнє розширення функціоналу.

3.4 Конструювання програмного забезпечення

<У підрозділі викладають:

- опис оригінальних алгоритмів чи модифікацій існуючих;
- опис структур даних, програмних структур та ін.
- опис бази даних з представленням концептуальної, логічної чи фізичної моделі та з описом сущностей чи таблиць;
- опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці. >

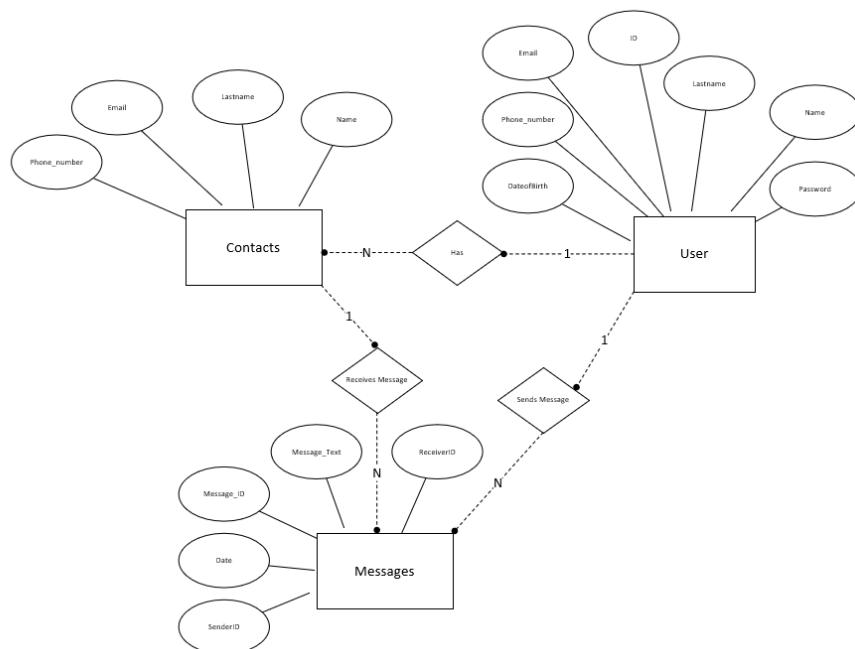


Рисунок 3.2 – ER діаграма сущностей User, Messages, Contacts

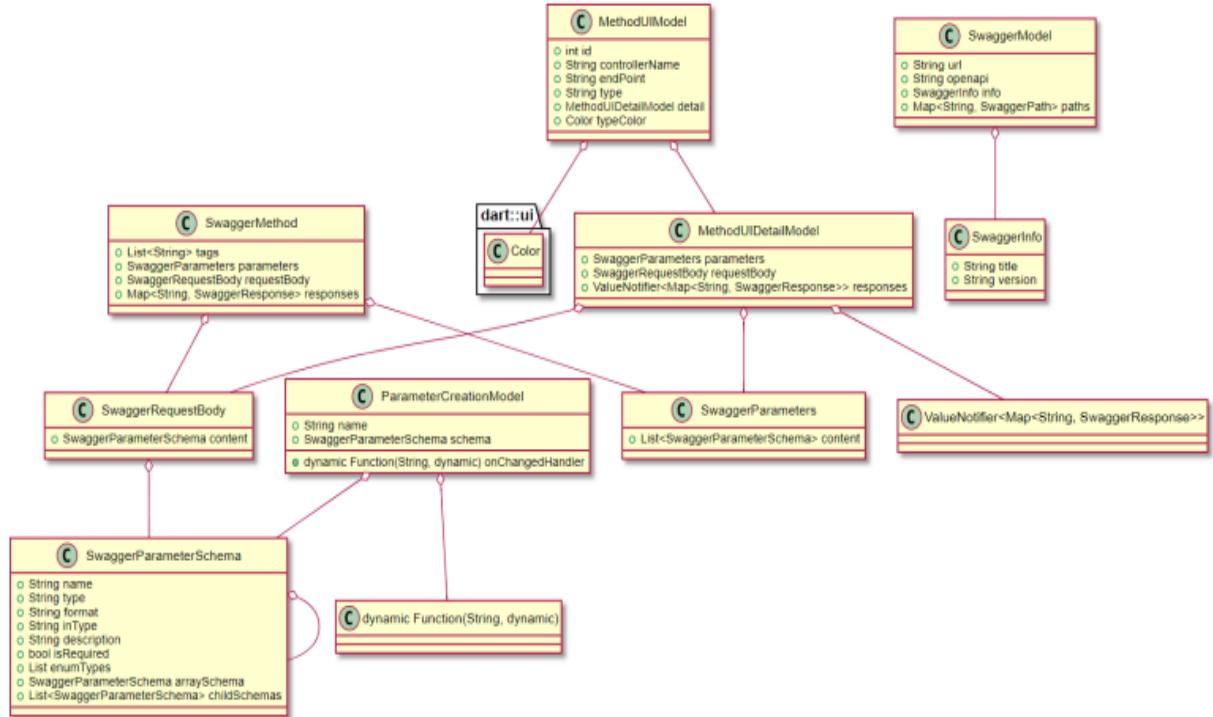


Рисунок 3.3 – Схема структурна класів

Бібліотека xpring, яка виконує запити в мережу блокчейн – однопоточна, тобто не може оброблювати паралельні запити. Через таку специфіку бібліотеки, довелось розробляти рішення яке б не блокувало запити з різних потоків, або інших частин коду. За основу ідеї алгоритму була взята багатопоточність самої мови програмування java. Java для вирішення проблем з потоками представляє ділянки коду які можна синхронізувати, що унеможливлює одночасний доступ до нього з різних потоків. Ця синхронізація проходить за допомогою передачі управління об'єкта-монітора. Алгоритм вирішення проблеми з доступом до однопоточної бібліотеки наведено на рисунку 3.4.

В якості системи управління базами даних використовується Postgres. База даних серверу призначена для зберігання користувачів, а також даних про їх Опис таблиць бази даних наведено у таблицях 3.11 - 3.14. Модель бази даних наведена на рисунку 3.12.

Таблиця 3.11 – Опис таблиці user

Назва поля	Тип даних	Опис
id	serial	ідентифікаційний номер користувача
email	varchar	електронна пошта користувача
password_id	int	посилання на запис у таблиці password, де зберігається пароль користувача

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.22.

Таблиця 3.22 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Головне середовище розробки програмного забезпечення серверної частини курсової роботи.
2	Postman	Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.
3	MySQL Workbench	Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних.

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

3.5 Аналіз безпеки даних

Підрозділ опціональний.

<У підрозділі викладають:

аналіз вразливостей ПЗ та будь-які питання пов'язані з безпекою даних.>

Висновки до розділу

< Необхідно стисло описати все, що було виконано у даному розділі.

Обсяг 0,75-1 сторінка>

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

<У підрозділі викладають:

аналіз метрик якості ПЗ та аналіз якості ПЗ за цими метриками.>

Метриками для оцінки якості ПЗ обрано наступні:

- швидкість ...

4.2 Опис процесів тестування

<У підрозділі викладають:

аналіз типів тестування, опис процесів тестування та приклади тестів, опишіть не більше 10 тестів.>

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.3 – 4.12.

Таблиця 4.3 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 10 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ, і який не входить у топ 10000 найпопулярніших паролей, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.

Фактичний результат	Збігається з очікуваним.
---------------------	--------------------------

4.3 Опис контрольного прикладу

<У підрозділі викладають:

повний опис одного контрольного прикладу з усіма можливими розгалуженнями та особливостями. Кроки доповнюють ілюстраціями.>

Висновки до розділу

< Необхідно стисло описати все, що було виконано у даному розділі.

Обсяг 0,75-1 сторінка>

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

<У підрозділі викладають:

аналіз засобів розгортання (за потреби) і повний опис покрокового розгортання ПЗ. Кроки доповнюють ілюстраціями (діаграма розгортання).>

5.2 Супровід програмного забезпечення

<У підрозділі викладають:

опис того, як буде виконуватись підтримка програмного забезпечення.

Обов'язково мають бути ілюстрації.>

Користувачі повинні мати можливість отримати нову версію консольного застосунку зожною версією. До того ж кожна нова версія консольного застосунку повинна бути опублікована в прт. Для автоматизації цього процесу був використаний сервіс GitHub Actions.

Створення нового випуску починається, коли нова версія консольного застосунку доставляється у репозиторій у гілку main, тобто коли commit має tag формату “v*.*.*.”, де замість “*” знаходиться число. Тоді у середовищі GitHub Actions встановлюється NodeJS. Після цього для проекту встановлюються залежності і проект збирається. Bash скрипт за допомогою бібліотеки pkg генерує виконувані файли (executables) для Linux і для Windows, та пакує файл для Linux у .deb пакет. Після цього .deb пакет і файл для Windows архівуються,

5.3 Повна інструкція користувача

<У підрозділі викладають:

опис того, як користувач має працювати з програмним забезпеченням від установки до завершення роботи. Обов'язково мають бути ілюстрації.>

Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі.
Обсяг 0,75-1 сторінка>

ВИСНОВКИ

Практична значимість одержаних результатів або отримані практичні результати.

Короткі підсумки по розв'язаннюожної функціональної задачі програмного забезпечення курсової роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Що таке веб додаток? | Блог WEBCASE. *Webcase*. URL: <https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/> (дата звернення: 15.12.2024).
- 2) Учасники проектів Вікімедіа. RSS – вікіпедія. *Vikipedija*. URL: <https://uk.wikipedia.org/wiki/RSS> (дата звернення: 15.12.2024).
- 3) Що таке API? - QALight. *QALight*. URL: <https://qalight.ua/baza-znaniy/shho-take-api/> (дата звернення: 15.12.2024).
- 4) Астуріас Д. Повний посібник з веб-скрепінгу [оновлення 2024] - RapidSeedbox. *RapidSeedbox*. URL: <https://www.rapidseedbox.com/uk/blog/web-scraping> (дата звернення: 15.12.2024).
- 5) What is a Web Browser and How does it Work?. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/web-browser/> (дата звернення: 22.05.2025).
- 6) NestJS - A progressive Node.js framework. *NestJS - A progressive Node.js framework*. URL: <https://nestjs.com/> (дата звернення: 22.05.2025).
- 7) BPMN Specification - Business Process Model and Notation. *BPMN Specification - Business Process Model and Notation*. URL: <https://www.bpmn.org/> (date of access: 22.05.2025).
- 8) Contributors to Wikimedia projects. Domain-driven design - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Domain-driven_design (date of access: 22.05.2025).
- 9) SuperTokens. What is JWT? Understand JSON Web Tokens | SuperTokens. *SuperTokens, Open Source User Authentication*. URL: <https://supertokens.com/blog/what-is-jwt> (date of access: 22.05.2025).
- 10) W3Schools.com. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/sql/sql_injection.asp (date of access: 22.05.2025).

ДОДАТКИ