ABSTRACT

Explanatory note the project consists of four sections, contains 12 figures, 13 tables, 4 appendices, 28 sources.

Diploma project is dedicated to the development of software for data collection of Internet resources. The purpose of creating software for the aggregation of materials the Internet-editions is providing opportunities to create an updateable archive of materials suitable for further analysis by other software. The thesis project reviewed current approaches to the problem of aggregation of web pages, technical solutions which can solve such a problem.

In the analyze section of the software requirements were defined inputs and outputs to the complex of the tasks, describes the features download papers from online sources. This section describes the use cases, formulated the main functional and non-functional requirements.

In the section simulation and design software was developed by data storage structure, defined the software architecture modeled diagrams of business processes. Identified components and their interaction, constructed class diagram of the software.

In the third section the analysis of the quality and testing of the product, the results obtained are presented

In the fourth section describes the process of software deployment.

ВМЕСТО ЭТОГО ЛИСТА ВСТАВИТЬ ЛИСТ ТИТУЛА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GUI – Graphical User Interface, графічний інтерфейс користувача.

CLI – Command Line Interface, інтерфейс командного рядка.

NoSQL – Not Only SQL, підхід до проектування не реляційних БД.

SQL – Structured Query Language, структурована мова запитів (до БД).

БД – база даних.

СКБД – система керування базами даних.

HTML – HyperText Markup Language, мова розмітки гіпертексту.

XML – eXtensible Markup Language, розширювана мова розмітки.

XSD – XML Schema Definition, мова опису структури XML.

ВСТУП

Інформація завжди була основою для людської роботи. Первісні племена зберігали дані про місця полювання, збору ягід і пасовиська. Торговці давніх цивілізацій вели записники із обліком своїх клієнтів та торгових операцій. Полководці – зберігали знання про свої походи, літописці – зберігали знання про історію, прості люди – знання про свята, традиції й свою працю. Спочатку усе зберігали в усній формі та передавали з покоління в покоління, але такий спосіб збереження інформації не був надійним. Із розвитком цивілізації почала розвиватись писемність, що спростило процеси збереження та передачі інформації.

Логічним розвитком прагення людей спростити процес збереження та передачі інформації стала поява книгодрукування. Дешеві та доступні для мас книги, а отже і освіта, стали основою бурхливого розвитку Європи протягом XVI-XX сторіч.

Із розвитком поліграфії почали з’являтись газети – регулярні видання, де публікували найсвіжішу, найактуальнішу інформацію для читачів. Серед читачів та дописувачів перших газет були королі та кардинали, графи та лорди. Новини, опубліковані у виданнях впливали на світову політику. У наш час цільова аудиторія новин є набагато ширшою – від президентів та міністрів, до простих робочих. На сьогодні через розивиток мережі Інтернет друковані видання заміщуються своїми онлайн-версіями, де публікують тексти, що впливають на сприйняття світу мільйонів людей.

Через великий об’єм публікацій, їх структурну неоднорідність тяжко автоматизувати їх обробку комп’ютером. Машинну обробку текстів необхідно проводити для спрощення дослідження суспільних трендів, світової думки про ту чи іншу тему [1]. Для спрощення машинної обробки необхідно створити інструмент, який дозволить узагальнити, зберегти та гомогенізувати дані з багатьох неоднорідних джерел, таких як сайти новин.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Із самого початку свого існування всесвітня мережа Інтернет росла із приголомшливою швидкістю: щодня з'являлися сотні й тисячі нових веб-сторінок, наповнених вмістом. У результаті бурхливого росту пошук інформації став надзвичайно складним. Для вирішення цієї проблеми з'явились перші пошукові сервіси – WebCrawler та Lycos (що використовували робота для збору інформації з усієї доступної мережі та її подальшої індексації), і служби каталогів, що заповнювались спеціальними редакторами вручну, – такі як Yahoo. Ці сервіси стали дуже популярними серед користувачів, оскільки зробили веб більш дружнім та легким у використанні. Людям подобається мати доступ до усієї необхідної інформації з одного місця. Пізніше також з'явились сервіси AltaVista та HotBot, що намагались сформувати найбільший, найповніший індекс мережі Інтернет. [2]

Проте ці сервіси мали численні недоліки. Вони були повільними, – повільно відповідали на запити користувачів, повільно будували та оновлювали свої індекси. Дуже часто сторінки, що досі були у каталогах, насправді змінювали свою адресу або й зовсім зникали, що приводило до поламаних посилань і невдоволення користувачів. Починають з'являтись технології пошукової оптимізації сайтів, через що у перші місця пошукової видачі потрапляли сторінки, що формально підходили під запит, але насправді не містили бажаного користувачем вмісту. З подальшим експоненційним ростом числа веб-користувачів навіть ці пошукові системи, що робили повну індексацію Інтернет, вже не могли задовольнити потреби своїх відвідувачів. [3]

Так само, як наприкінці 20-го сторіччя з'явився попит на пошукові системи, у першому десятилітті 21-го сторіччя з'являється попит на аналітичні системи для роботи з Інтернет. Величезні маси інформації, що

накопичились та кожного дня з’являються у Всесвітній павутині, являють чималий інтерес для великого числа дослідників – лінгвістів, маркетологів, політологів, соціологів, економістів.

Для вирішення проблеми агрегації даних у мережі Інтернет необхідно створити сервіс із відповідними функціями, який на відміну від наявних пошукових систем міг би гарантувати повноту та цілісність отриманих текстів.

Для найкращої роботи такий сервіс має працювати із розподіленою СКБД. Розподілена база даних – це сукупність логічно зв’язаних баз даних або частин однієї бази, які розпаралелені між декількома територіально-розподіленими ПЕОМ і забезпечені відповідними можливостями для управління цими базами або їх частинами. Тобто, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення. В дійсності розподілена база даних є віртуальною базою даних, компоненти якої фізично зберігаються на декількох різних реальних базах даних на декількох різних вузлах.

Розподілена система керування базою даних (СКБД) – це програмний комплекс, призначений для управління розподіленими базами даних і забезпечує прозорий доступ користувачів до розподіленої інформації. Прозорий доступ означає непомітний для користувача, тобто у користувача має складатися враження ніби він працює з єдиною базою даних, яка розміщена на його власному комп’ютері. [4]

1.2 Змістовний опис і аналіз предметної області

Предметною областю проекту є створення рішення для агрегації текстів Інтернет-видань, що дозволить з мінімальними зусиллями додавати нові джерела інформації, завантажувати з них дані та надасть єдиний інтерфейс для отримання агрегованих даних сторонніми сервісами.

У сфері штучного інтелекту та машинної обробки великих масивів даних однією з важливих задач є збір даних для обробки. Задача збору текстових даних, що розміщені у мережі Інтернет (таких як тексти новин, дописи у онлайн-журналах або щоденниках), не є тривіальною – кожен веб-сайт має свою розмітку, структуру та обмеження на кількість запитів. В свою чергу, аналітичні системи, що обробляють такі тексти, для своєї коректної роботи повинні мати доступ до свіжих та актуальних даних.

На Інтернет-ресурсах, що спеціалізуються на новинах, матеріали зазвичай відсортовано за датою, і за датою ж до них можна отримати доступ. Різна структура кожного Інтернет-видання обумовлює необхідність створення окремого обробника для кожного з них. Зібрані дані повинні також бути збережені у структурованому вигляді для подальшого легкого до них доступу.

На діаграмі діяльності на рисунку 1.1 наведено ті види діяльності, що можуть бути здійснені у рамках предметної області.

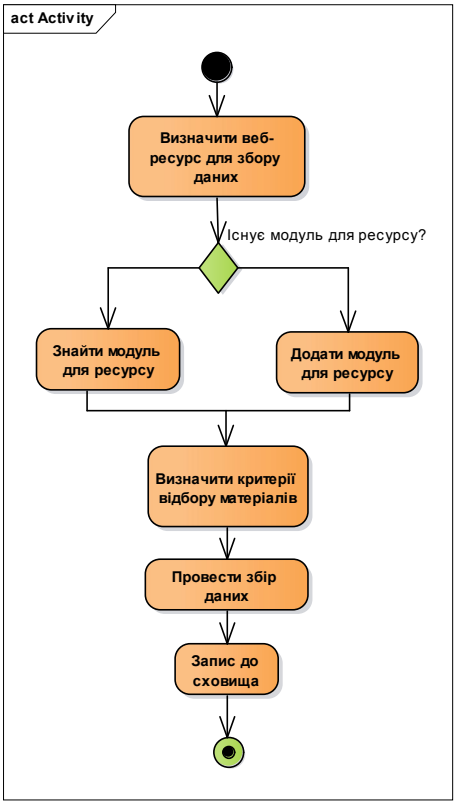

Рисунок 1.1 – Схема структурна діяльності процесу «Агрегація матеріалів»

1.3 Аналіз успішних ІТ-проектів

Пошук рішень для даної предметної області не показав результатів, які б повністю задовольняли поставленим вимогам, проте є декілька рішень, що виконують схожі функції.

1.3.1 Аналіз відомих технічних рішень

Для агрегації даних з Інтернет-видань та стрічок новин був розроблений формат RSS (Rich Site Summary) – універсальний формат XML файлів, у яких сайти передають всім зацікавленим клієнтам дані про новини, – у тому числі, їх заголовки, тексти, дати публікації та посилання на повний текст новини. Практично усі сайти новин надають RSS-стрічку для своїх користувачів. Спецаільні додатки – читачі RSS-стрічок та агрегатори новин дають можливість читати ці стрічки з багатьох веб-сайтів одночасно.

RSS – спеціальний формат, призначений для опису стрічок новин, анонсів статей, змін у блогах тощо. Інформація з різних джерел, подана у форматі RSS, може бути зібрана, опрацьована і подана користувачеві в зручному для нього вигляді спеціальними програмами.

Зазвичай за допомогою RSS подається короткий опис нової інформації, що з’явилася на сайті, і посилання на її повну версію. Інтернет-ресурс у форматі RSS називається RSS- каналом, RSS-стрічкою або RSS-фідом. [5]

Більшість сучасних браузерів та поштових клієнтів працюють з RSS-стрічками. Крім того, існують спеціалізовані програми (RSS-агрегатори), які збирають і опрацьовують інформацію RSS-каналів. Також дуже популярними є веб-агрегатори, які спеціалізуються на збиранні та відображенні RSS-каналів, такі як Яндекс.Лента[6], Feedly Reader[7], Новотека[8]. [9]

Недоліком RSS є те, що як правило у стрічках розміщують не повні, а лише часткові тексти новин та їх заголовки, тому для досягнення поставленої мети застосування стрічок RSS не є прийнятним варіантом.

Чимало веб-сайтів також надають API – програмні інтерфейси для взаємодії із собою. Через такі інтерфейси зазвичай можна напряму отримувати дані. На жаль, не всі веб-сайти новин надають публічний API. Деякі сайти мають API, але доступ до нього платний. У будь-якому разі, кожен сайт має свій програмний інтерфейс, тому не вийде зробити універсальний засіб взаємоді з API.

Іншим варіантом технічного рішення задачі агрегації даних є розбір (скрейпінг) Інтернет-ресурсу з метою пошуку текстів новин. Необхідно повністю переглянути html-тексти сторінок Інтернет-ресурсу та виявити, де саме розміщуються тексти новин, вилучити їх з коду сторінки та зберегти на зовнішній носій (у файл чи у базу даних). Такий підхід є більш трудомістким порівняно з розбором RSS-стрічок, оскільки необхідно досліджувати дані з нерегулярною структурою (структура різних веб-сайтів може кардинально відрізнятися). Іншим недоліком скрейпінгу є велике навантаження на сервер стороннього веб-ресурсу. Якщо надсилати запитів для завантаження текстів надто часто, сервер може заблокувати клієнта, що його перевантажує, і таким чином можна втратити можливість завантажувати дані. Перевагою ж скрейпінгу є набагато більша гнучкість при виборі даних та незалежність від поставника даних. У кінцеву вибірку завжди потраплять повні тексти новин, якщо не перевантажувати сервер веб-ресурсу. [10]

Існує також технологія розбору веб-сторінок за допомогою «комп’ютерного зору». За допомогою такої технології програма сприймає сайт візуально, як його сприймав би користувач, і зчитує певні елементи спершу у вигляді зображень, а потім оброблює їх та перетворює у текстові або інші дані. [11]

Зважаючи на переваги та недоліки кожної технології, а також на розмаїття даних, що необхідно обробити, тяжко обрати якусь одну підходящу для роботи технологію. Найбільш потужною була б можливість скомбінувати

усі доступні технології та дати можливість користувачу самому обрати, яку використовувати, у залежності від його потреб.

1.3.2 Аналіз відомих програмних продуктів

Відомими програмними продуктами, які реалізують функціонал агрегації новин з RSS-стрічок, є Inoreader.Com[12], Digg.Com[13], Theoldreader.Com[14], численні розширення для браузерів, а також додатки для Windows – QuiteRSS[15], RSS Bandit[16], FeedReader[17], FeedDemon[18]. Поштові клієнти типу Microsoft Outlook[19], Mozilla Thunderbird[20] мають функціонал для завантаження RSS-підписок. [9]

Усі ці продукти дають можливість завантажувати і переглядати RSS-стрічки. На жаль, для економії трафіку в RSS-стрічках немає повних текстів новин, лише короткий опис та посилання.

Також описані продукти не дають можливості завантажувати інформацію у придатному для подальшої автоматичної обробки вигляді. По суті, вони просто є інтерфейсом для читання матеріалів.

Для вирішення задачі скрейпінгу створено чимало різних додатків, сервісів та фреймворків. Усі вони дають можливість у тому чи іншому вигляді запрограмувати спосіб завантаження даних з певної веб-сторінки. Одним з найвідоміших сервісів такого роду є ParseHub[21], а також його найбільші конкуренти – Scrapy, import.io, Portia. Можна використати утиліти, призначені для емуляції роботи браузера, на кшталт Selenuim Web Driver. Також є чимало додатків до веб-браузерів, що виконують схожі функції, але не є достатньо гнучкими, щоб задовольнити задані вимоги.

ParseHub – це онлайн сервіс, що дає можливість у прямо у веб-браузері задати формат даних, що необхідно зібрати, налаштувати агент для збору даних і запустити його. Результати зберігаються на серверах ParseHub, а отримати доступ до них можна, експортувавши дані в Excel, JSON або з власного додатку за допомогою наявного у сервісі API. [21]

Нижче наведено переваги ParseHub.

а) Можливості налаштування роблять інструмент гнучким та дають повний контроль над тим, які дані та як будуть витягнуті.

б) Багатий вбудований інструментарій дасть раду навіть складним сучасним веб-сторінкам з динамічним контентом.

в) Результати роботи можна переглядати наживо та відповідно змінювати модель даних.

г) Можна використовувати API проекту та завантажувати результати;

д) Хороші посібники користувача, оформлені у вигляді додатків до браузера.

е) Відмінна технічна підтримка – команда розробників відкрита до діалогу та співпраці.

Нижче наведено недоліки ParseHub.

а) Крута крива навчання – необхідно затратити деякий час, щоб опанувати інструмент.

б) Динамічні результати сильно сповільнюють роботу на великих обсягах даних.

в) При запуску свого проекту на сервері результати не відображаються у браузері, що сповільнює розробку.

г) Немає API для отримання усіх результатів одинм викликом.

д) Порівняно великий час виконання запитів.

е) Немає кешування результатів, а отже і доступу до них, якщо вихідний сайт недоступний. [10]

ж) Безкоштовна версія є обмеженою та повільною. Платна ж версія стартує від $150 в місяць, що є достатньо дорого. [21]

Scrapy – це безкоштовний проект з відкритим вихідним вихідним кодом, розроблений на мові Python. Scrapy – це фреймворк, що дозволяє створювати свої. Scrapy використовує HTTP-запити до веб сторінок, щоб отримати їх HTML-код, після чого є можливість розбирати HTML за

допомогою XPath-запитів, зберігати результат у файлову систему або в базу даних у вигляді регулярної структури. Перевагою Scrapy є вже готова інфраструктура для забору даних з веб-сторінок, а також реалізація на мові Python, що автоматично означає доступ до широкої та розвиненої інфраструктури бібліотек Python. Для Scrapy реалізований сервер Scrapyd, також із відкритим вихідним кодом. Перевагою, як і недоліком Scrapy є використання ним протоколу HTTP напряму, що робить роробку більш гнучкою, але й трудомісткою. Це також зумовлює ще один великий недолік – Scrapy не виконує клієнтський javascript-код, за допомогою якого багато сайтів генерують свій вміст. [22]

Selenium Web Driver – це безкоштовний фреймворк, призначений для написання автоматизованих тестів для веб-додатків. Через своє основне призначення цей фреймворк має багато корисних функцій для навігації веб-сайтами, у тому числі перехід по сторінках за допомогою емуляції натискань на елементи веб-інтерфейсу, вибір елементів інтерфейсу за її CSS-класом, XPath, повну навігацію DOM-деревом сторінки. Selenium повністю емулює роботу веб-переглядача [23] Така емуляція означає, що коректно виконується весь javascript-код, а веб-сайти в принципі не можуть відрізнити робота, що збирає вміст сайтів, від справжнього користувача.

Отже, усі згадані вище програмні продукти мають свої переваги та недоліки, але загалом, із деякими доопрацюваннями можуть задовольнити заданим вимогам.

1.4 Аналіз вимог до програмного забезпечення

Після аналізу технічного завдання було розроблено модель варіантів використання системи, зображену на рисунку 1.2. Опираючись на варіанти використання було розроблено вимоги до системи.

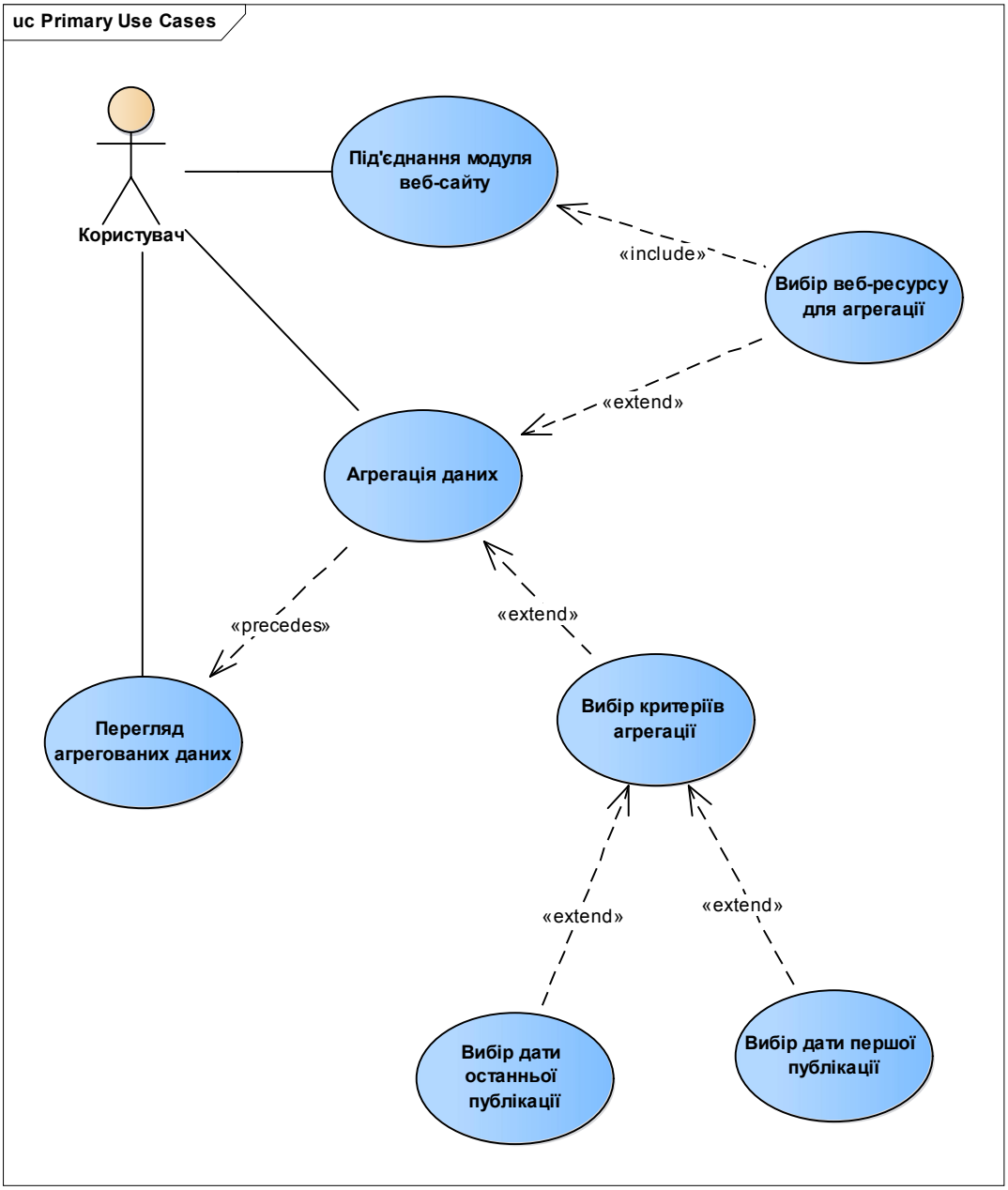

Рисунок 1.2 – Схема структурна варіантів використання

1.4.1 Розроблення функціональних вимог

Програмне забезпечення для агрегації матеріалів інтернет-видань повинна забезпечувати виконання наступних функцій:

– обрання періоду для збереження матеріалів;
– обрання джерела або джерел матеріалів;
– перегляд збережених матеріалів;
– експорт збережених матеріалів до зовнішнього сховища даних;

– створення та підключення додатків (модулів) для завантаження матеріалів з інших джерел.

Функціональні вимоги (див. рисунок 1.3) напряму зв’язані із прецедентами варіантів використання. Їх описано у схемі трасування вимог, яку зображено на рисунку 1.4.

На рисунку 1.5 зображена матриця трасування вимог. Вона допомагає визначити джерело будь-яких вимог та їх залежність від найвищого рівня до найнижчого.

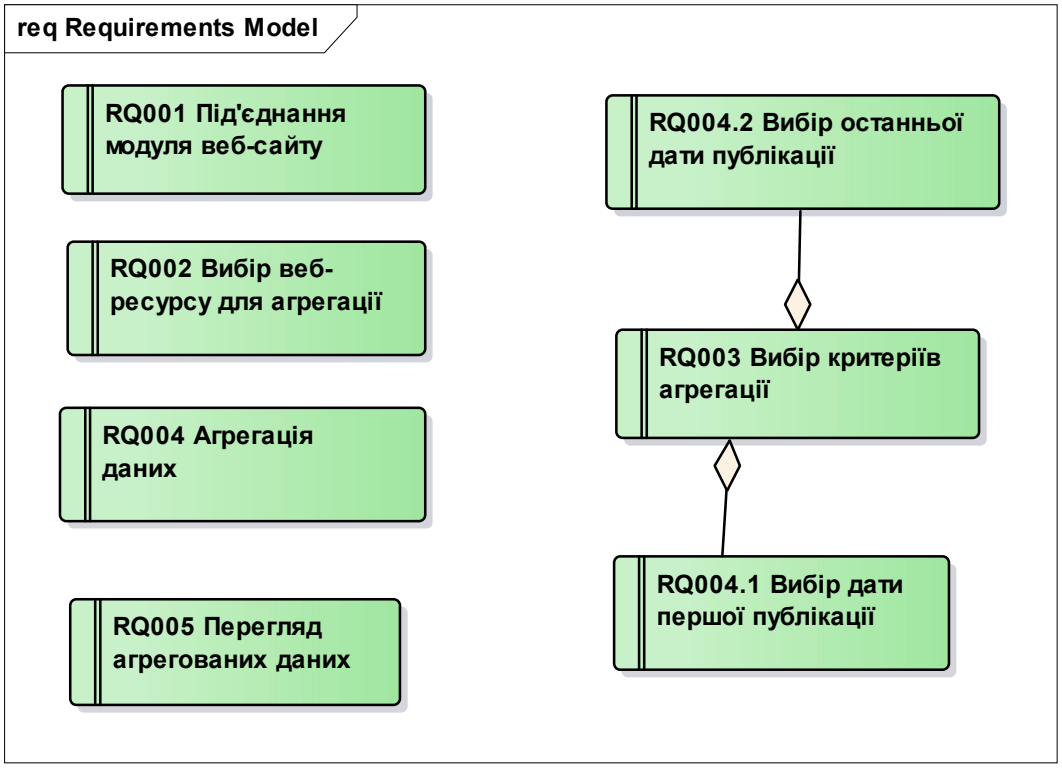

Рисунок 1.3 – Схема структурна вимог

Рисунок 1.4 – Схема структурна трасування вимог

Специфікацію вимог та детальний опис кожного прецеденту наведено нижче у формі таблиць опису прецедентів.

У таблиці 1.1 наведено опис прецеденту «Під'єднання модуля веб-сайту».

Таблиця 1.1 – UC001 «Під'єднання модуля веб-сайту»

Дійові особи	Користувач
Передумова	Модуль веб-сайту не під'єднано
Сценарій	1. Користувач копіює модуль веб-сайту у папку модулів. 2. Система завантажує модуль.
Альтернативний сценарій	3.1. При невірному типі модулю система виводить на екран повідомлення про помилку 3.2. Система продовжує роботу, але модуль не завантажується.
Постумова	Успішне завершення: у списку сайтів з'являється новий сайт. Невдале завершення: модуль не завантажено.

У таблиці 1.2 наведено опис прецеденту «Вибір веб-ресурсу для агрегації. Сценарій цього прецеденту містить лише одну дію – обрання користувачем веб-ресурсу для агрегації.

Таблиця 1.2 – UC002 «Вибір веб-ресурсу для агрегації»

Дійові особи	Користувач
Передумова	Систему запущено. Модулі веб-ресурсів завантажено.
Сценарій	1. Користувач обирає веб-ресурс для агрегації зі списку.
Альтернативний сценарій	3.1. Якщо модуль веб-ресурсу завантажено неправильно, система видає помилку та аварійно завершує роботу.
Постумова	Успішне завершення: є можливість запустити агрегацію даних.

У таблиці 1.3 наведено опис прецедента «Вибір критеріїв агрегації».

Таблиця 1.3 – UC003 «Вибір критеріїв агрегації»

Дійові особи	Користувач
Передумова	Систему запущено.
Сценарій	1. Користувач обирає критерій агрегації. 2. Користувач задає значення критерія агрегації.
Альтернативний сценарій	3.1. Якщо користувач увів неправильне значення критерія, система видає повідомлення про помилку. 3.2 Користувач натискає кнопку «ОК», і продовжує роботу.
Постумова	Успішне завершення: критерії агрегації встановлено.

У таблиці 1.4 наведено опис прецедента «Вибір дати першої публікації».

Таблиця 1.4 – UC004 «Вибір дати першої публікації»

Дійові особи	Користувач
Передумова	Систему запущено. Користувач обрав критерій «Дата першої публікації»
Сценарій	1. Користувач задає значення дати першої публікації.
Альтернативний сценарій	3.1. Якщо користувач увів неправильне значення дати, система видає повідомлення про помилку. 3.2 Користувач натискає кнопку «ОК», і продовжує роботу.
Постумова	Успішне завершення: дату першої публікації встановлено.

У таблиці 1.5 наведено опис прецедента «Вибір дати останньої публікації».

Таблиця 1.5 – UC005 «Вибір дати першої публікації»

Дійові особи	Користувач
Передумова	Систему запущено. Користувач обрав критерій «Дата останньої публікації»
Сценарій	1. Користувач задає значення дати останньої публікації.
Альтернативний сценарій	3.1. Якщо користувач увів неправильне значення дати, система видає повідомлення про помилку. 3.2 Користувач натискає кнопку «ОК», і продовжує роботу.
Постумова	Успішне завершення: дату останньої публікації встановлено.

У таблиці 1.6 наведено опис прецедента «Агрегація даних».

Таблиця 1.6 – UC006 «Агрегація даних»

Дійові особи	Користувач, система
Передумова	Систему запущено.
Сценарій	1. Користувач обирає пункт «Агрегація даних». 2. Система завантажує встановлені значення критеріїв агрегації. Якщо критерії не встановлені, вони ігноруються.
Альтернативний сценарій	3.1. У випадку будь-якої помилки система видає повідомлення та аварійно завершує роботу.
Постумова	Успішне завершення: виведено повідомлення про успішну агрегацію даних веб-ресурсів.

У таблиці 1.7 наведено опис прецедента «Перегляд агрегованих даних»

Таблиця 1.7 – UC007 «Перегляд агрегованих даних»

Дійові особи	Користувач
Передумова	Систему запущено. Було проведено агрегацію даних.
Сценарій	1. Користувач обирає пункт «Перегляд агрегованих даних». 2. Система відображає деталі агрегованих даних.
Альтернативний сценарій	3.1. У випадку будь-якої помилки система видає повідомлення та аварійно завершує роботу.
Постумова	Успішне завершення: показано тексти агрегованих статей.

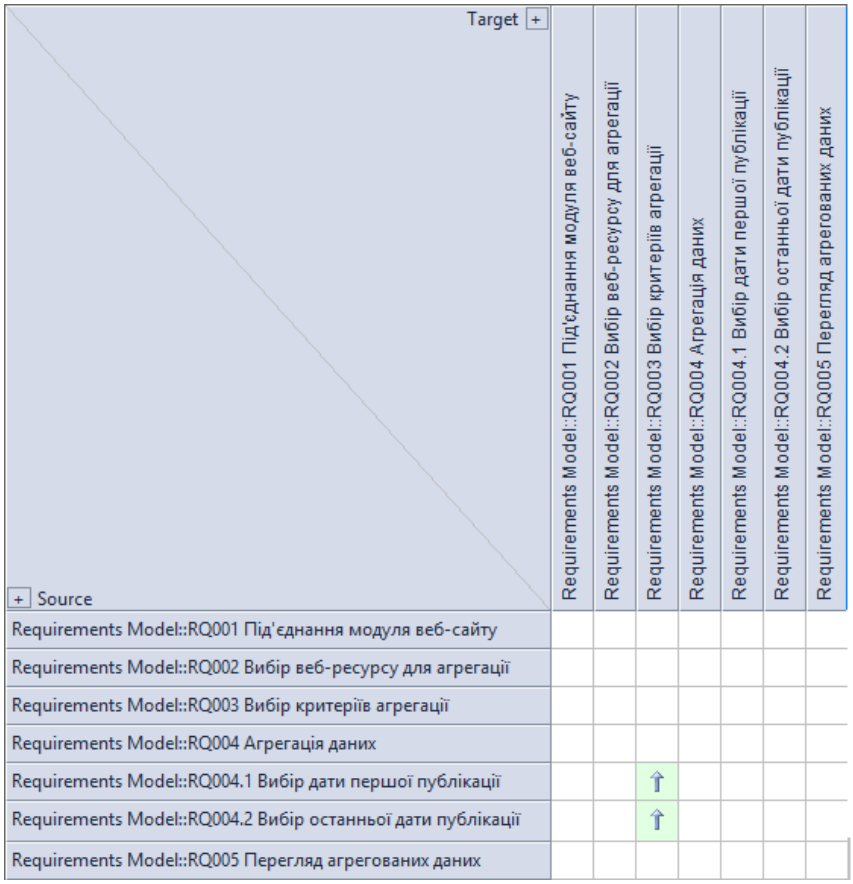

Рисунок 1.5 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Для зручного використання, програмне забезпечення для агрегації матеріалів Інтернет-видань повинне задовольняти наступним нефункціональним вимогам:

- система повинна забезпечувати повноту отриманих даних;
- система повинна бути відмовостійкою;
- сховище даних системи повинне надавати можливість одночасного доступу до нього багатьох корстувачів;
- сховище даних системи повинне бути доступним незалежно від самої системи;
- наявність графічного інтерфейсу;
- наявність інтерфейсу командного рядка;
- наявність документованого інтерфейсу програмного забезпечення для розробки модулів агрегації.

1.4.3 Постановка комплексу задач модулю

На основі наведених вимог, опишемо комплекс задач, що повинен виконувати програмний продукт для агрегації матеріалів Інтернет-видань.

а) Приймати модулі для агрегації різних веб-ресурсів.

б) Завантажувати тексти статей заданих Інтернет-видань.

в) Давати можливість перегляду завантажених даних.

Висновки по розділу

У першому розділі обґрунтовано потребу у розробці програмного забезпечення для агрегації матеріалів Інтернет-видань, розглянуто технічні та програмні рішення, що існують на сьогодні. Проведено детальний опис та аналіз предметної області. Наведено діаграму варіантів використання із їх описом. Показано зв’язок вимог із прецедентами. Визначено функціональні

та нефункціональні вимоги. Моделювання та конструювання програмного забезпечення

1.5 Моделювання та аналіз програмного забезпечення

Для створення програмного забезпечення для агрегації матералів Інтернет-видань необхідно спроектувати схеми бізнес-процесів. Схеми бізнес-процесів спроектовано у відповідності із методологією IDEF0. Розроблено 2 рівні структурної схеми бізнес-процесів:

а) схема структурна рівня системи в цілому (Рисунок 2.1);

б) схема структурна рівня окремих модулів системи (Рисунок 2.2);

в) схема структурна бізнес-процесу «Збір текстів» (Рисунок 2.3)

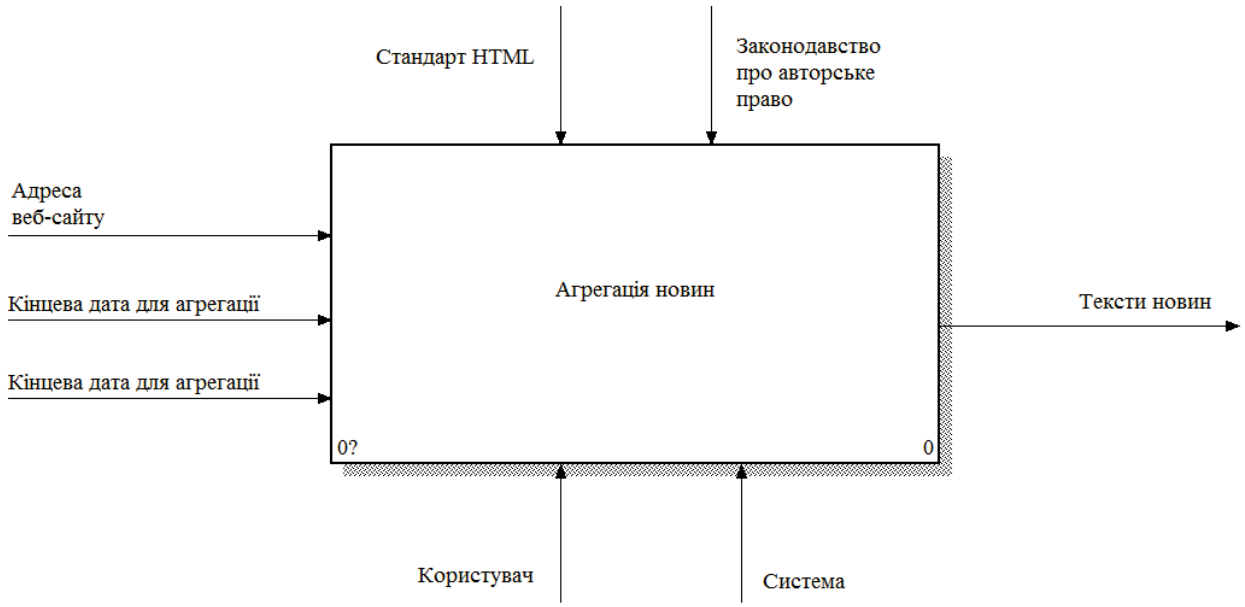

Рисунок 2.1 – Схема структурна бізнес-процесу

У схемі структурній бізнес-процесу (див. рисунок 2.1) систему представлено як «чорний ящик». Як видно із схеми, вхідними даними до системи є адреса веб-сайту з новинами, стартова та кінцева дати для збору матеріалів. Вихідними даними системи є тексти новин, опублікованих на веб-сайті у вказаному діапазоні дат.

У схемі структурній бізнес-процесів (див. рисунок 2.2) зображено декомпозицію бізнес-процесу агегації новин. Процес починається із уведення

користувачем адреси веб-сайту, на якому буде відбуватися агрегація. На основі уведеної адреси, система обирає модуль агрегації для конкретного веб-сайту. Цей модуль, а також дата початку і кінця публікації матеріалів, що необхідно агрегувати, служать вхідними даними для процесу збору текстів. Завантажені тексти новин передаються у процес запису матеріалів, що зберігає їх до зовнішнього сховища, формуючи таким чином вихідний результат роботи системи.

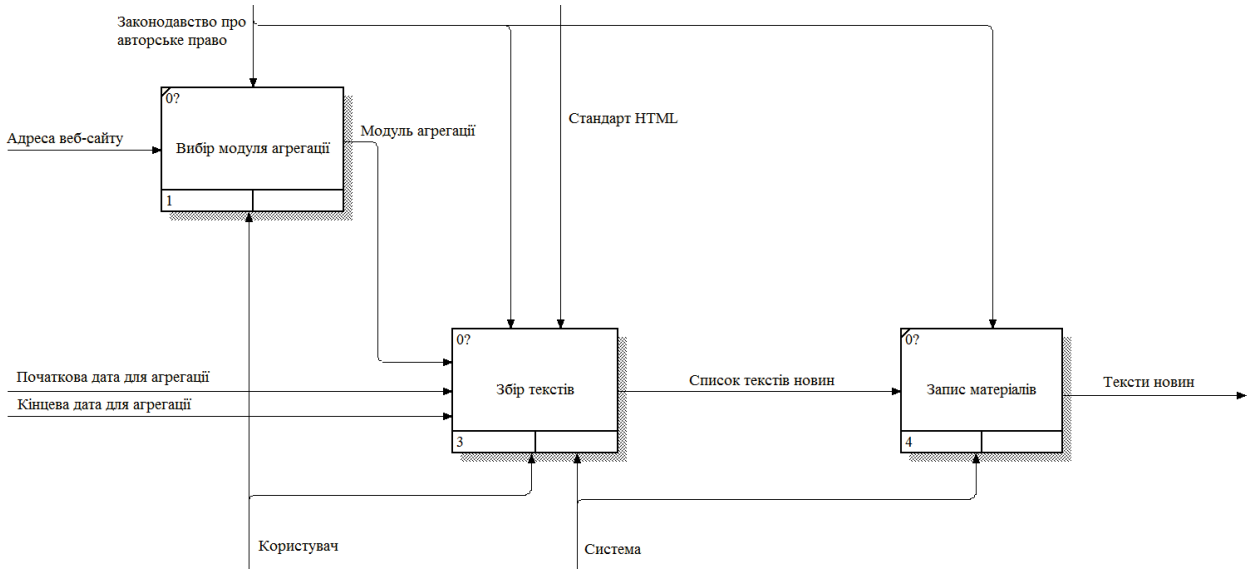

Рисунок 2.2 – Схема структурна бізнес-процесів, рівень 1

Бізнес-процес збору текстів, що зображено на рисунку 2.3, починається з процесу побудови списку сторінок, що отримує на вхід модуль агрегації та діапазон дат для агрегації. Цей процес формує список посилань на веб-сторінки статей, що входять у заданий користувачем діапазон дат. Процес переходу на посилання приймає на вхід модуль агрегації та список посилань, а результатом його є HTML-код веб-сторінок статей, з якого процес «Знайти текст статей» виділяє HTML-код, що стосується лише тексту статей. Останній процес видаляє форматування із прийнятого HTML-коду статей. Результатом бізнес-процесу «Збір текстів» є тексти новин у заданому діапазоні дат.

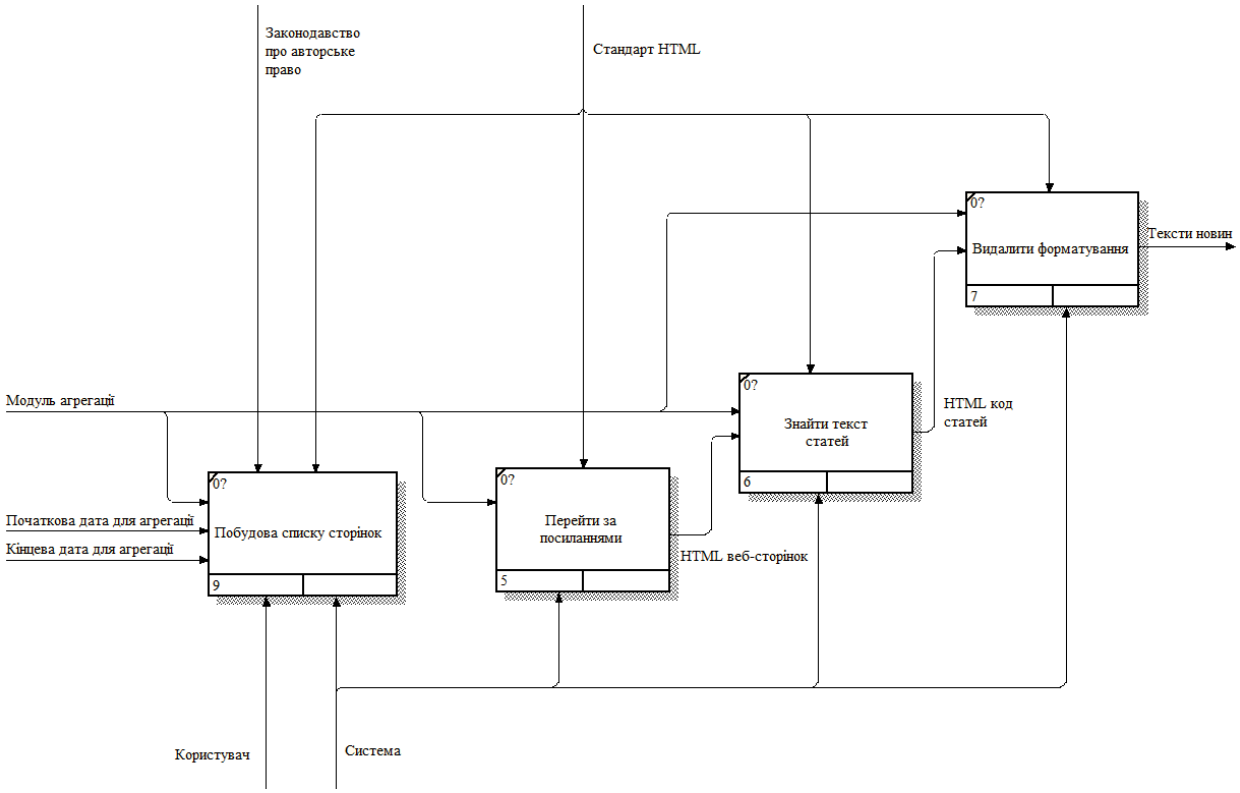

Рисунок 2.3 – Схема структурна бізнес-процесу «Збір текстів»

1.6 Архітектура програмного забезпечення

Програмне забезпечення для агрегації матеріалів Інтернет-видань складається з трьох модулів та ядра програмного комплексу. Ядро програмного комплексу надає інтерфейси для взаємодії модулів між собою.

Модуль взаємодії з користувачем надає інтерфейс користувача для взаємодії із програмним комплексом. Модуль взаємодії призначений для заданння вхідних даних системи, його реалізація може мати графічний (GUI) або консольний (CLI) вигляд. Модуль приймає від користувача назву сайту для агрегації та діапазон дат публікації для агрегації, та передає ці відомості до ядра системи.

Модуль агрегації даних виконує всю роботу із завантаження текстів з Інтернет-сторінок. Він використовує Selenium WebDriver, RSS або чисті HTTP-запити для навігації інтернет-сторінками і завантаження з них текстів. Завантажені тексти у синхронному (після завантаження усіх) або

асинхронному (по завантаженні кожного окремого) режимі передаються до ядра системи.

Модуль збереження даних виконує запис завантажених даних до постійного сховища. Це може бути як просто файлова система, XML-файли або реляційна СКБД, так і NoSQL (наприклад, MongoDB) СКБД або розподілене сховище даних (таке як Hive).

У сховищі даних ми зберігатимемо єдину сутність «Стаття», що має такі атрибути: текст, джерело, дату і час завантаження, дату і час публікації (див. рисунок 2.4). Підкреслені атрибути входять до ключа сутності.

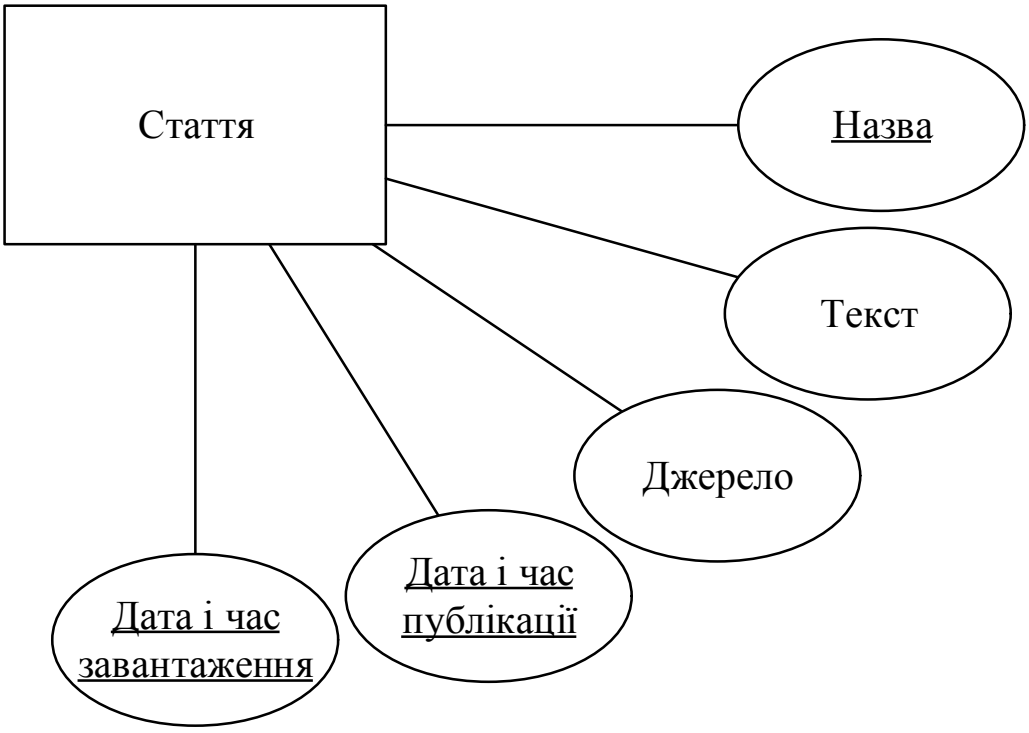

Рисунок 2.4 – Схема структурна сутностей системи

У випадку використання файлової системи та XML-файлів як сховища даних, файл опису схеми XML-структури XSD має такий вигляд:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element
    name="Articles"
    minOccurs="1"
```

```
maxOccurs="1">
<xs:complexType>
  <xs:element
    name="Article"
    minOccurs="1"
    maxOccurs="unbounded">
    <xs:element
      name="PublishedDateTime"
      type="xs:dateTime"
      minOccurs="1"
      maxOccurs="1">
    </xs:element>
    <xs:element
      name="RetrievedDateTime"
      type="xs:dateTime"
      minOccurs="1"
      maxOccurs="1">
    </xs:element>
    <xs:element
      name="Source"
      type="xs:anyURI"
      minOccurs="1"
      maxOccurs="1">
    </xs:element>
    <xs:element
      name="Text"
      type="xs:string"
      minOccurs="1"
      maxOccurs="1">
```

```
            </xs:element>
            <xs:element
               name="Title"
               type="xs:string"
               minOccurs="1"
               maxOccurs="1">
            </xs:element>
         </xs:element>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Структура файлової система має такий вигляд:

./Articles/<назва сайту>/<дата публікації>/<Назва>.xml

У зображеній на рисунку 2.5 схемі структурній класів програмного забезпечення інтерфейси IConfigurationService, IFetchService, IPersistanceService відповідають модулям взаємодії із користувачем, агрегації та збереження даних відповідно. Класові Core відповідає модуль ядра програми. Клас Config містить конфігурацію, задану користувачем. Клас Article зберігає дані про статтю.

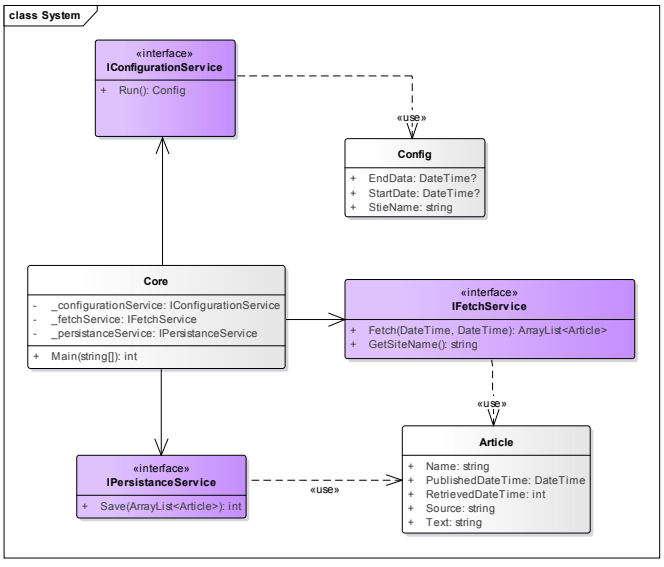

Рисунок 2.5 – Схема структурна класів програмного забезпечення

При проектуванні програмного забезпечення було використано принцип впровадження залежностей (Dependency Injection, DI) для реалізації підтримки модульності застосування. Впровадження залежностей дозволяє зменшити сильне зв'язування між програмними компонентами. Замість жорсткого кодування залежностей (наприклад, драйвера якої-небудь бази даних), впроваджується список сервісів, яких може потребувати компонент. Після цього сервіси підключаються третьою стороною. Такий підхід забезпечує краще управління майбутніми змінами і рішення проблем в програмному забезпеченні, що розробляється. [24] Залежності ядра від модулів, замість того, щоб бути жорстко заданими, описано за допомогою залежності від специфікації (інтерфейсу), а не реалізації.

На рисунку 2.6 наведено діаграму компонентів системи, повний опис якої наведено у На ній зображено чотири компоненти: ядро системи (Core), компонент для завантаження даних (Data), компонент збереження даних (Persistence) та компонент взаємодії з користувачем (Interaction).

Таблиця 2.1 – Опис компонентів програмного забезпечення

Компонент	**Опис**
Core	Ядро додатку. Виконує функцію комунікації між модулями.
ConfigurationService	Надає можливості конфігурації додатку.
CLI	Конфігурація через командний рядок.
WebAPI	Конфігурація за допомогою WebAPI.
WinForms	Конфігурація у вікні графічного інтерфейсу.
ConfigurationFile	Завантаження конфігурації з файлу.
FetchService	Надає можливості агрегації текстів статей.
SeleniumWebDriver	Агрегація через SeleniumWebDriver.
HTTPProvider	Агрегація за допомогою HTTP-запитів.

Продовження таблиці 2.1

Компонент	Опис
PersistenceService	Надає можливості збереження текстів.
FileSystem	Збереження текстів у файловій системі.
XMLWriter	Збереження текстів у вигляді XML-файлів.
SQLProvider	Збереження текстів у реляційній базі даних.
HiveProvider	Збереження текстів у Hive.

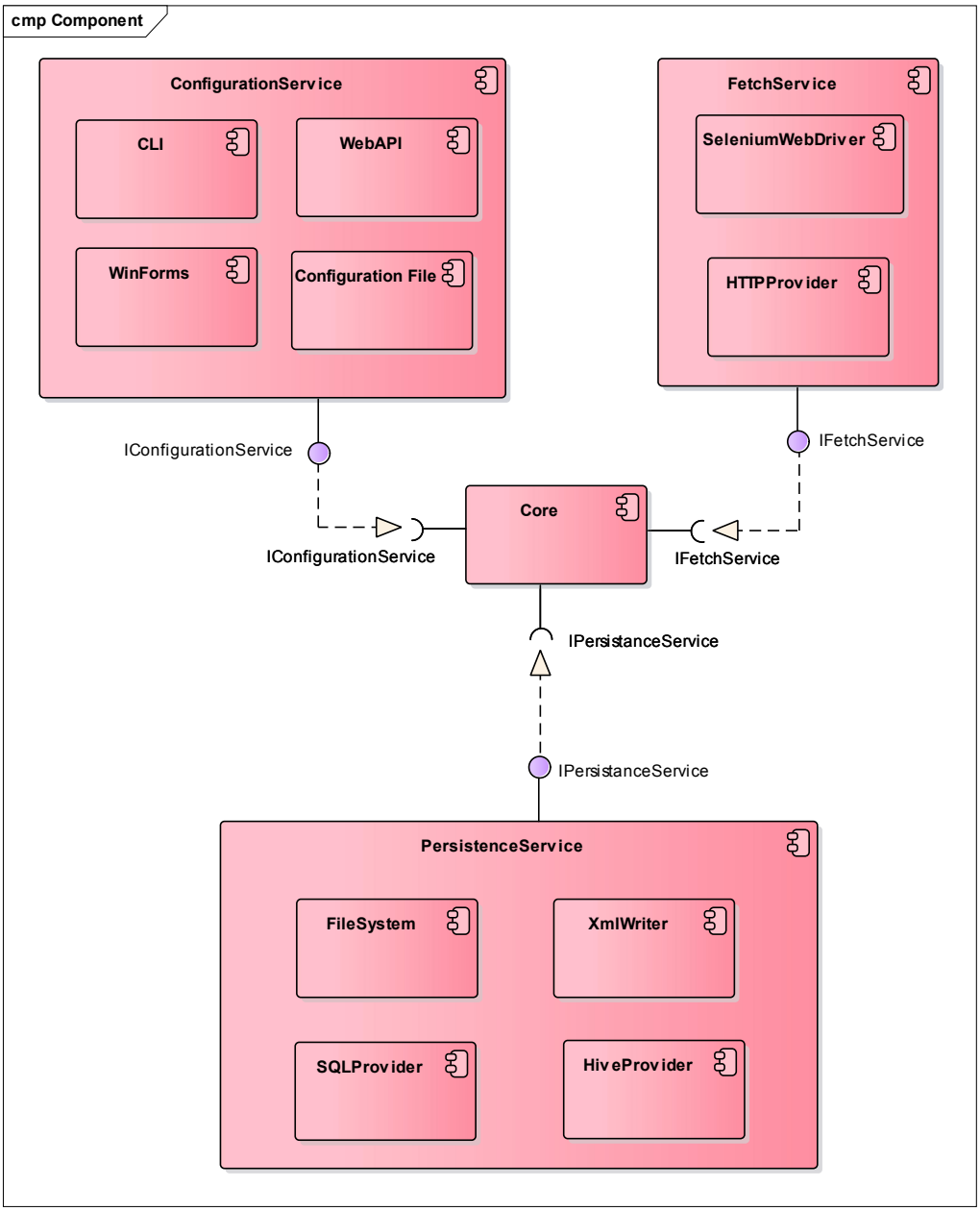

Рисунок 2.6 – Схема структурна компонентів системи.

На структурній схемі станів системі (Рисунок 2.7) зображено послідовність взаємодії системи із користувачем та модулів системи між собою. Користувач викликає конфігуратор та задає у ньому параметри агрегації. Конфігурація передається в ядро, що передає завантажувачу діапазон дат, і отримує статті. Далі статті з ядра відправліються у записувач, що зберігає дані у сховище та повертає статус операції. Ядро передає статус назад до користувача. Послідовність роботи системи завершено.

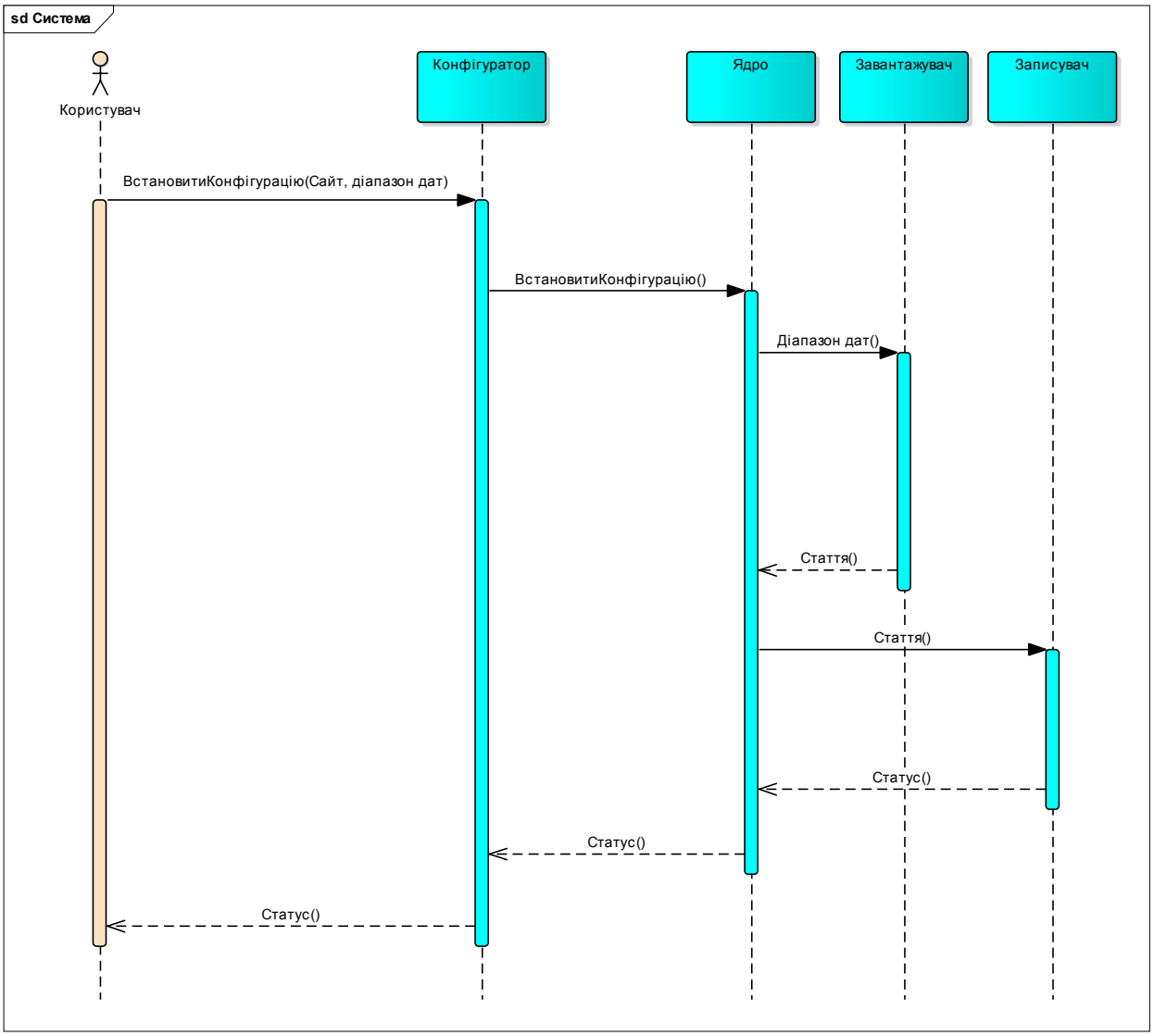

Рисунок 2.7 – Схема структурна послідовностей системи

1.7 Конструювання програмного забезпечення

Для розробки було обрано платформу .NET Core. Ядро додатку та основні модулі будуть розроблені на мові C# 7.0 із використанням модуля DI

StructureMap. Для модуля агрегації даних із веб-сторінок буде використано Selenium WebDriver. Модуль збереження використовуватиме запис до XML-файлів. Середовищем розробки обрано Visual Studio 2017 Community Edition. Незважаючи на те, що проект виконується одноосібно, в майбутньому до нього можуть долучитися інші розробники, тому для збереження вихідних кодів та ведення контролю версій використано Git.

C# (вимовляється Сі-шарп) — багатопарадигменна мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).[25]

Платформа .NET Core – це новий, переосмислений випуск платформи .NET Framework від Microsoft. .NET Core від самого початку був задуманий як гнучкий, кросплатформний та відкритий проект. .NET Core повністю відкидає стару кодову базу .NET Framework, – майже усі компоненти платформи були переписані заново, щоб забезпечити максимальну швидкодію програм, що працюють під нею. Завдяки своїй модульній системі (навіть системні компоненти поставляються у вигляді окремих пакетів) .NET Core легко розширюється та переноситься на інші платформи, а отже додатки, розроблені для .NET Core можна запускати на Windows, Linux, MacOS. [26]

Visual Studio 2017 – це інтегроване середовище розробки, що підтримує розробку на C# 7.0 із використанням платформи .NET Core та технології ASP.NET. Зручний, перевірений часом інтерфейс цього середовища було покращено для досягнення ліпшої продуктивності у розробці. Вбудовані шаблони генерації коду роблять Visual Studio незамінним помічником при розробці типових додатків. Visual Studio 2017 Community Edition є безкоштовною для розробки проектів, що розробляються із некомерційною метою.

Git — розподілена система керування версіями файлів та спільної роботи. Проект створив Лінус Торвальдс для управління розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. Серед принципів роботи Git: збереження файлів, локальні операції, цілісність даних, легке галуження, зливання та перебазування даних.[27]

Для створення модульної структури програмного забезпечення було використано принцип впровадження залежностей (Dependency Injection, DI) для реалізації підтримки модульності застосування. Впровадження залежностей дозволяє зменшити сильне зв'язування між програмними компонентами. Замість жорсткого кодування залежностей (наприклад, драйвера якої-небудь бази даних), впроваджується список сервісів, яких може потребувати компонент. Після цього сервіси підключаються третьою стороною. Такий підхід забезпечує краще управління майбутніми змінами і рішення проблем в програмному забезпеченні, що розробляється. [24] Залежності ядра від модулів, замість того, щоб бути жорстко заданими,

описано за допомогою залежності від специфікації (інтерфейсу), а не реалізації.

Висновки по розділу

У другому розділі було проведено моделювання та аналіз програмного забезпечення. Для опису бізнес-процесів, що автоматизуються програмним забезпеченям, наведено структурну схему IDEF0. Детально описано архітектуру програмного забезпечення. Наведено спосіб збереження даних (схема Entity-Relationship, файл XSD). Наведено архітектуру побудови програми, зображену у повністю описаних схемах класів, компонентів та послідовностей.

Наведено підходи до розробки ПЗ, використані засоби розробки, управління версіями програмного коду, принципи розробки.

2 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Мета випробувань та тестування

Метою випробувань є перевірка відповідності функцій програмного забезпечення для агрегації матеріалів Інтернет-видань вимогам, поставленим у технічному завданні.

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

2.2 Опис методів випробувань

Для проведення тестування основних функцій програмного забезпечення для агрегації матеріалів Інтернет-видань було складено план випробувань. Цим планом передбачено ручну перевірку основного функціоналу системи (Smoke Testing) та перевірку інтеграції між модулями системи (Integration Testing).

Коректну роботу кожного модулю необхідно забезпечити покриттям коду за допомогою модульних тестів. Правильно розроблена модульна структура додатку дозволяє легко розробляти автоматизовані тести для кожного із модулів. Наразі розробка модульних тестів не є актуальною, оскільки під час дослідної експлуатації додатку можуть змінитись вимоги до нього, що призведе до необхідності переробки і функціоналу, і тестів.

2.3 План випробувань

План випробувань програмного забезпечення для агрегації матеріалів Інтернет-видань наведеон у таблицях 3.1-3.4

Таблиця 3.1 – Перевірка під’єднання модуля веб-сайту.

Мета тесту:	**Перевірка функції «Під’єднання модуля веб-сайту»**
Початковий стан КЗ:	Додаток не запущено.
Вхідні дані:	Модуль веб-сайту www.pravda.com.ua.
Схема проведення тесту:	1. Скопіювати модуль у папку Plugins/Downloaders/ 2. Запустити додаток.
Очікуваний результат:	Сайт www.pravda.com.ua є у списку доступних сайтів.
Стан КЗ після проведення випробування:	Система коректно запущена.

Таблиця 3.2 – Перевірка вибору веб-ресурсу для агрегації.

Мета тесту:	**Перевірка функції «Вибір веб-ресурсу для агрегації»**
Початковий стан КЗ:	1. Додано модуль веб-сайту www.pravda.com.ua 2. Запущено додаток.
Вхідні дані:	Адреса веб-сайту www.pravda.com.ua
Схема проведення тесту:	1. Вікрити випадаючий список доступних сайтів.
Очікуваний результат:	Сайт www.pravda.com.ua є у списку доступних сайтів.
Стан КЗ після проведення випробування:	Система коректно запущена.

Таблиця 3.3 – Перевірка вибору критеріїв агрегації

Мета тесту:	**Перевірка функції «Вибір критеріїв агрегації»**
Початковий стан КЗ:	1. Додано модуль веб-сайту www.pravda.com.ua 2. Запущено додаток. 3. Обрано веб-сайт www.pravda.com.ua
Вхідні дані:	Адреса веб-сайту www.pravda.com.ua Дати першої і останньої публікації для агрегації.
Схема проведення тесту:	1. Увести як дату першої публікації 01.01.2017 2. Увести як дату останньої публікації 01.06.2017.
Очікуваний результат:	Обрано діапазон дат.

Продовження таблиці 3.3

Мета тесту:	**Перевірка функції «Вибір критеріїв агрегації»**
Стан КЗ після проведення випробування:	Система коректно запущена. Система готова до проведення агрегації.

Таблиця 3.4 – Перевірка агрегації та перегляду текстів.

Мета тесту:	**Перевірка функції «Агрегація та перегляд текстів»**
Початковий стан КЗ:	Додаток запущено. Уведено адресу веб-сайту www.pravda.com.ua. Обрано діапазон дат від 01.01.2017 до 01.06.2017.
Вхідні дані:	Статті веб-сайту www.pravda.com.ua
Схема проведення тесту:	1. Натиснути кнопку «Провести агрегацію» 2. Зачекати завершення агрегації 3. Запустити додаток перегляду текстів.
Очікуваний результат:	Відображаються усі статті веб-сайту www.pravda.com.ua у період з 01.01.2017 до 01.06.2017
Стан КЗ після проведення випробування:	Систему вимкнено.

Висновки по розділу

У третьому розділі було наведено мету випробувань програмного забезпечення для агрегації матеріалів Інтернет-видань. Було наведено список документів, згідно яких проводяться випробування, обґрунтовано методику та заходи випробувань.

Для перевірки якості програмного продукту був розроблений детальний план тестування основного функціоналу. Виконання цього плану гарантує коректну роботу головних функцій та відповідність програми технічному завданню.

3 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розгортання програмного забезпечення

Для розгортання програмного забезпечення необхідно вручну скопіювати файли програми на комп’ютер, або скористатись інсталятором. Для коректної роботи додатку також необхідно встановити .NET Core Runtime (поставляється разом із додатком).

3.1.1 Мінімальна конфігурація технічних засобів

Запуск програмного забезпечення можливий на комп’ютері, мінімальна конфігурація якого дозволяє запустити операційну систему Windows 10. Мінімальну конфігурацію комп'ютера для запуску програми наведено у таблиці 4.1. [28]

Таблиця 4.1 – Мінімальна конфігурація технічних засобів

Процесор:	Процесор або система на чипі з мінімальною тактовою частотою 1 ГГц.
Оперативна пам’ять:	1 ГБ (для 32-розрядної версії) або 2 ГБ (для 64-розрядної версії).
Дисковий простір:	Потрібні 16 ГБ (для 32-розрядної версії) або 20 ГБ (для 64-розрядної версії).
Відеокарта:	З підтримкою DirectX 9 або новішої версії з драйвером WDDM 1.0.
Екран:	800x600.

Вимоги для роботи додатку із розподіленою СКБД наведено у технічному завданні.

3.2 Робота з програмним забезпеченням

Детальну інструкцію користувача наведено у Додатку В.

Висновки по розділу

У четвертому розділі наведено процес розгортання програмного забезпечення для агрегаці матеріалів Інтернет-видань. Зазначено мінімальну необхідну конфігурацію технічних засобів для запуску програмного забезпечення та описано способи роботи із програмним забезпеченням.

ВИСНОВКИ

У дипломній роботі була аргументована протреба у розробці програмного забезпечення для агрегації матеріалів Інтернет-видань. Проведено змістовний аналіз та опис особливостей предметної області. Розроблено сценарії використання застосунку, функціональні та нефункціональні вимоги до нього.

На основі вимог проведено моделювання та конструювання програмного забезпечення. Внаслідок моделювання застосунок розбито на окремі модулі: ядро програми, конфігуратор, завантажувач, записувач.

Визначено архітектуру системи та описано структуру її класів, компонентів. На основі спроектованих схем було реалізовано програмне забезпечення для агрегації матеріалів Інтернет-видань.

Розробку виконано на платформі .NET Core. Модуль завантажувача працює завдяки Selenium WebDriver, а збереження даних відбувається у файловій системі в XML-файлах, що мають єдину схему. Такий вибір технологій дозволяє коректно обробляти сторінки Інтернет-видань, завантажувати їх чистий текст. XML-файли – це простий для машинної обробки формат.

Спроектована архітектура програмного забезпечення дозволяє за рахунок слабкої зв’язаності компонентів у майбутньому з легкістю перейти на інші платформи збереження та завантаження даних. У цій архітектурі просто змінювати інтерфейс користувача.

Розроблений застосунок було протестовано та проаналізовано на предмет відповідності технічному завданню та характеристикам якості програмного забезпечення.

Наведено спосіб розгортання системи разом із діаграмою розгортання. Програмний комплекс може використовуватися у великих агрегаторах даних, як компонент системи обробки даних, сайту-агрегатора новин тощо.

ПЕРЕЛІК ПОСИЛАНЬ

1) Словарь Академик [Електронний ресурс] – Режим доступу до ресурсу: http://dic.academic.ru/dic.nsf/ruwiki/292201/Анализ_текста
2) Erik S. The MetaCrawler Architecture for Resource Aggregation on the Web [Електронний ресурс] / S. Erik, E. Oren // University of Washington. – 1996. – Режим доступу до ресурсу: https://homes.cs.washington.edu/~etzioni/papers/ieee-metacrawler.pdf.
3) Mei K. Information retrieval on the web / K. Mei, T. Koichi. // ACM Computing Surveys. – 2000. – С. 144–173.
4) Пєтух А. М. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних [Електронний ресурс] / А. М. Пєтух, О. В. Романюк, О. Н. Романюк // ВНТУ. – 2016. – Режим доступу до ресурсу: http://posibnyky.vntu.edu.ua/db/31.htm
5) RSS-стрічки | Українська правда [Електронний ресурс] – Режим доступу до ресурсу: http://www.pravda.com.ua/rss-info/
6) Яндекс.Стрічка [Електронний ресурс] – Режим доступу до ресурсу: https://news.yandex.ua
7) Feedly [Електронний ресурс] – Режим доступу до ресурсу: https://feedly.com
8) Новотека [Електронний ресурс] – Режим доступу до ресурсу: www.novoteka.ru
9) Обзор лучших RSS-ридеров для чтения лент новостей [Електронний ресурс] – Режим доступу до ресурсу: http://livelenta.com/obzor-luchshix-rss-riderov-dlya-chteniya-lent-novostej.html
10) ParseHub vs Kimono: In a right scrape [Електронний ресурс] – Режим доступу до ресурсу: http://www.damiancannon.com/blog/parsehub-scraping-websites-for-goodness/

11) Иванова Н. А. Технологии анализа и разбора контента [Електронний ресурс] / Н. А. Иванова, А. И. Чеботарь // Современные научные исследования и инновации.. – 2015. – Режим доступу до ресурсу: http://web.snauka.ru/issues/2015/12/61247

12) Inoreader [Електронний ресурс] – Режим доступу до ресурсу: https://www.inoreader.com/

13) Digg [Електронний ресурс] – Режим доступу до ресурсу: http://digg.com/

14) The Old Reader [Електронний ресурс] – Режим доступу до ресурсу: https://theoldreader.com/home

15) QuiteRSS [Електронний ресурс] – Режим доступу до ресурсу: https://quiterss.org/ru

16) Rss Bandit [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/RssBandit/RssBandit

17) Feed Reader [Електронний ресурс] – Режим доступу до ресурсу: http://feedreader.com/

18) Feed Demon [Електронний ресурс] – Режим доступу до ресурсу: http://www.feeddemon.com/.

19) Microsoft Outlook [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/outlook-com/

20) Mozilla Thunderbird [Електронний ресурс] – Режим доступу до ресурсу: https://www.mozilla.org/ru/thunderbird/

21) ParseHub [Електронний ресурс] – Режим доступу до ресурсу: http://www.parsehub.com

22) Scrapy 1.4 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://doc.scrapy.org/en/latest/

23) What is Selenium? [Електронний ресурс] – Режим доступу до ресурсу: http://www.seleniumhq.org/

24) Симан М. Внедрение зависимостей в .NET / Марк Симан. – СПб.: Питер, 2014. – 464 с.

25) Standard ECMA-334 C# Language Specification 4th edition [Електронний ресурс] // ECMA International. – 2006. – Режим доступу до ресурсу: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

26) Introduction to .NET Core [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://blogs.msdn.microsoft.com/dotnet/2014/12/04/introducing-net-core/

27) Scott C. Pro Git [Електронний ресурс] / C. Scott, S. Ben // Apress – Режим доступу до ресурсу: https://git-scm.com/book/uk/v2

28) Вимоги до системи [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/windows/windows-10-specifications#system-specifications

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

Все эти листы не печатаются! Они нужны только для того, чтоб сделать правильное содержание

ДОДАТОК Б ОПИС ПРОГРАМИ

ДОДАТОК В КЕРІВНИЦТВО КОРИСТУВАЧА

жлдівоадлфоівлдаодлвіфоадло

ДОДАТОК Г ГРАФІЧНИЙ МАТЕРІАЛ

gfdgsdfgsdfg

ЛИСТ 1. СХЕМА СТРУКТУРНА ВАРІАНТІВ ВИКОРИСТАНЬ

ЛИСТ 2. СХЕМА СТРУКТУРНА КОМПОНЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛИСТ 3. СХЕМА СТРУКТУРНА ПОСЛІДОВНОСТЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛИСТ 4. СХЕМА СТРУКТУРНА КЛАСІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛИСТ 5. СХЕМА СТРУКТУРНА ДІЯЛЬНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ