2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для створення програмного забезпечення для агрегації матеріалів 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)

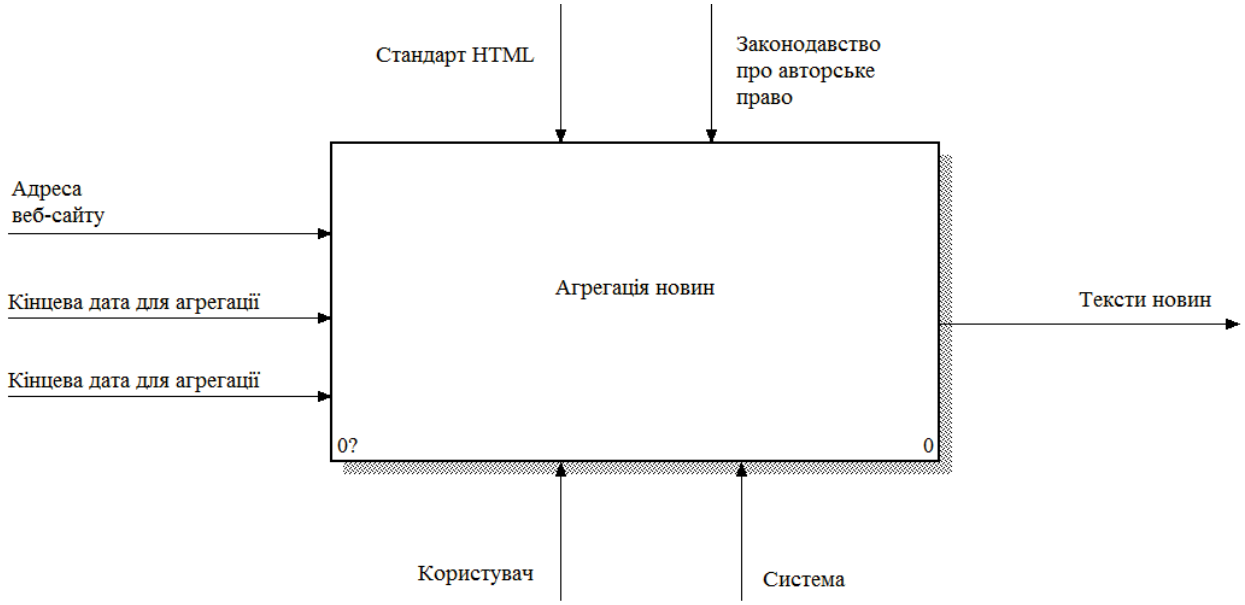

Рисунок 2.1 – Схема структурна бізнес-процесу

У схемі структурній бізнес-процесу (див. рисунок2.1) 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. рисунок2.2) 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.

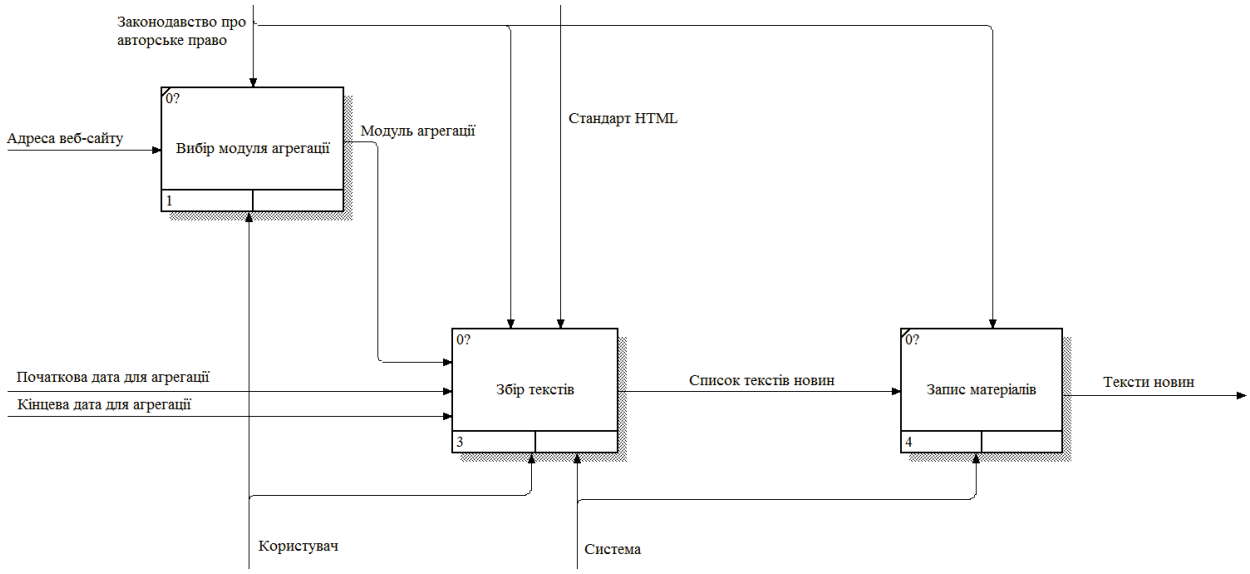

Рисунок 2.2 – Схема структурна бізнес-процесів, рівень 1

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.

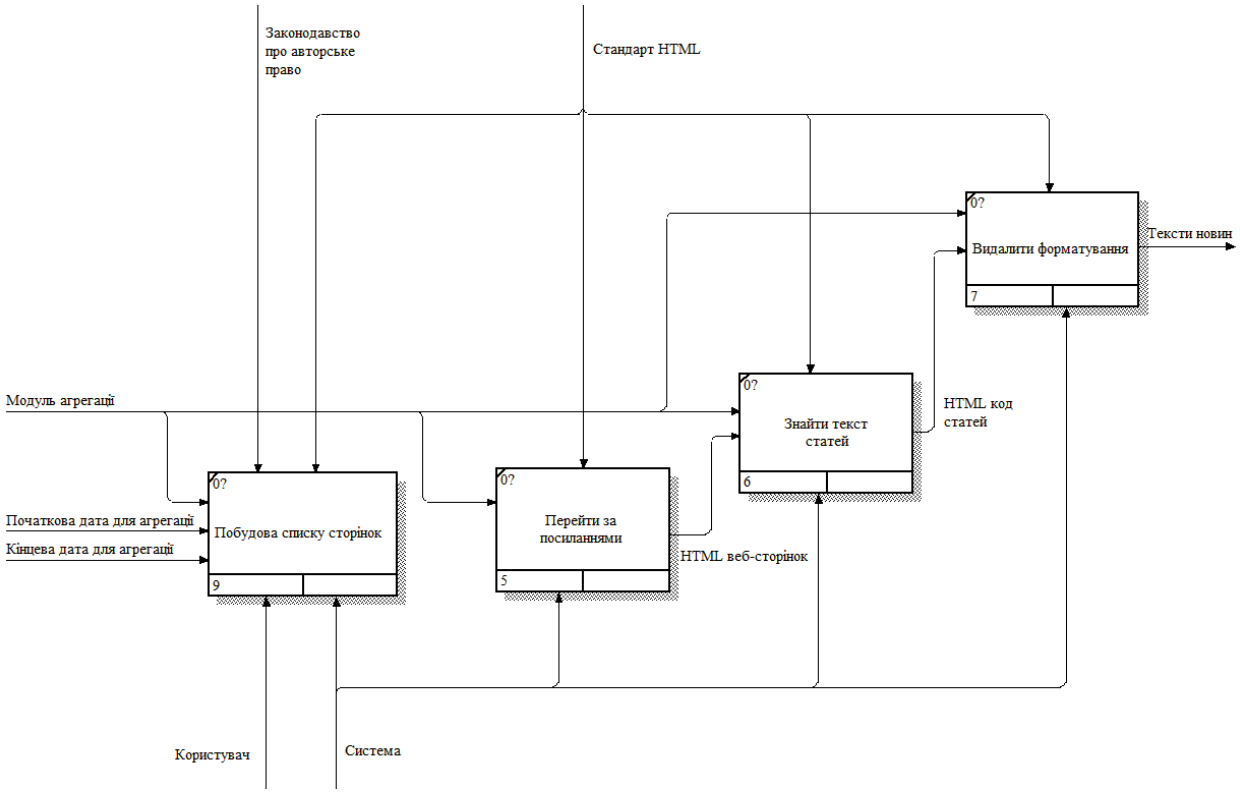

Рисунок 2.3 – Схема структурна бізнес-процесу «Збір текстів»

2.2 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. рисунок2.4). Підкреслені атрибути входять до ключа сутності.

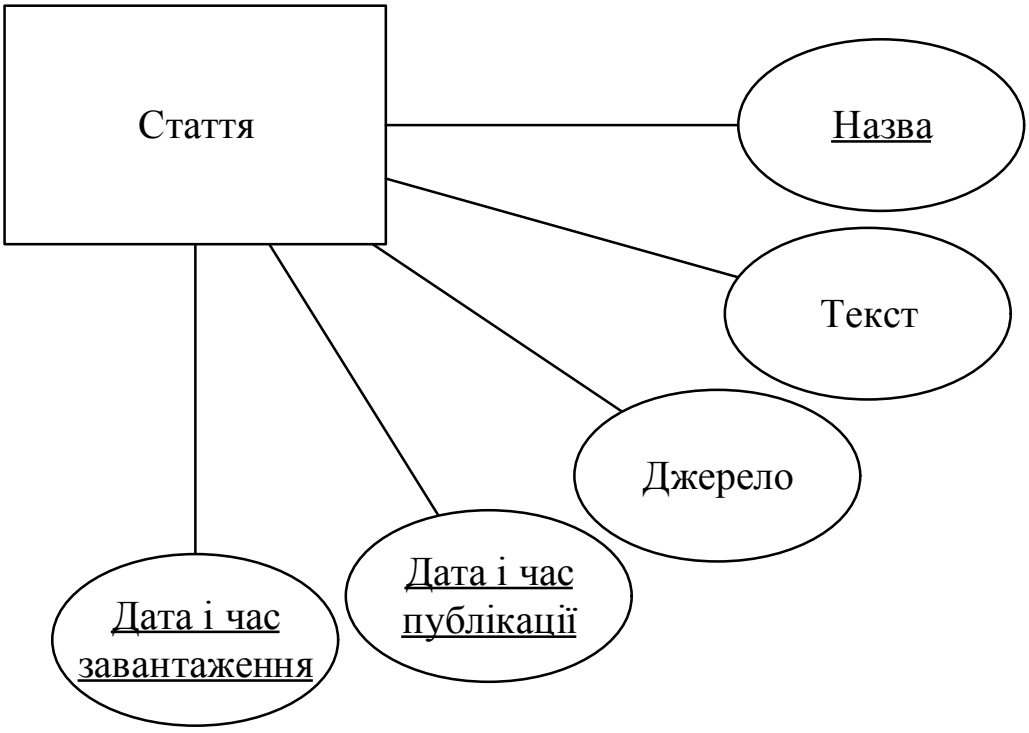

Рисунок 2.4 – Схема структурна сутностей системи

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.

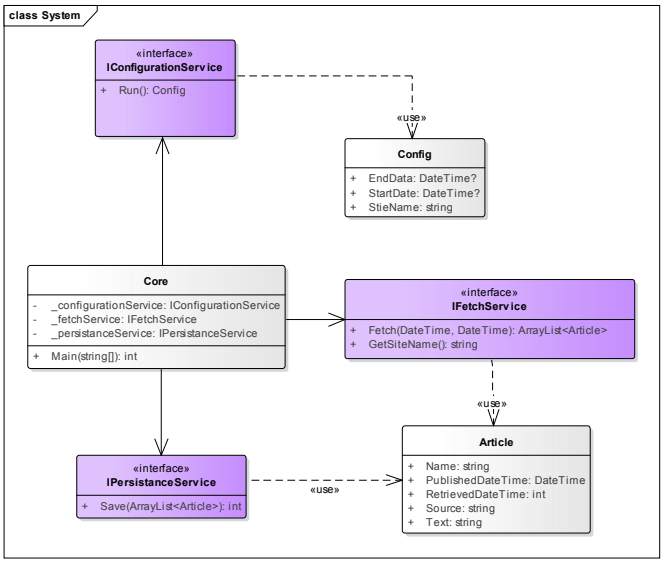

Рисунок 2.5 – Схема структурна класів програмного забезпечення

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.

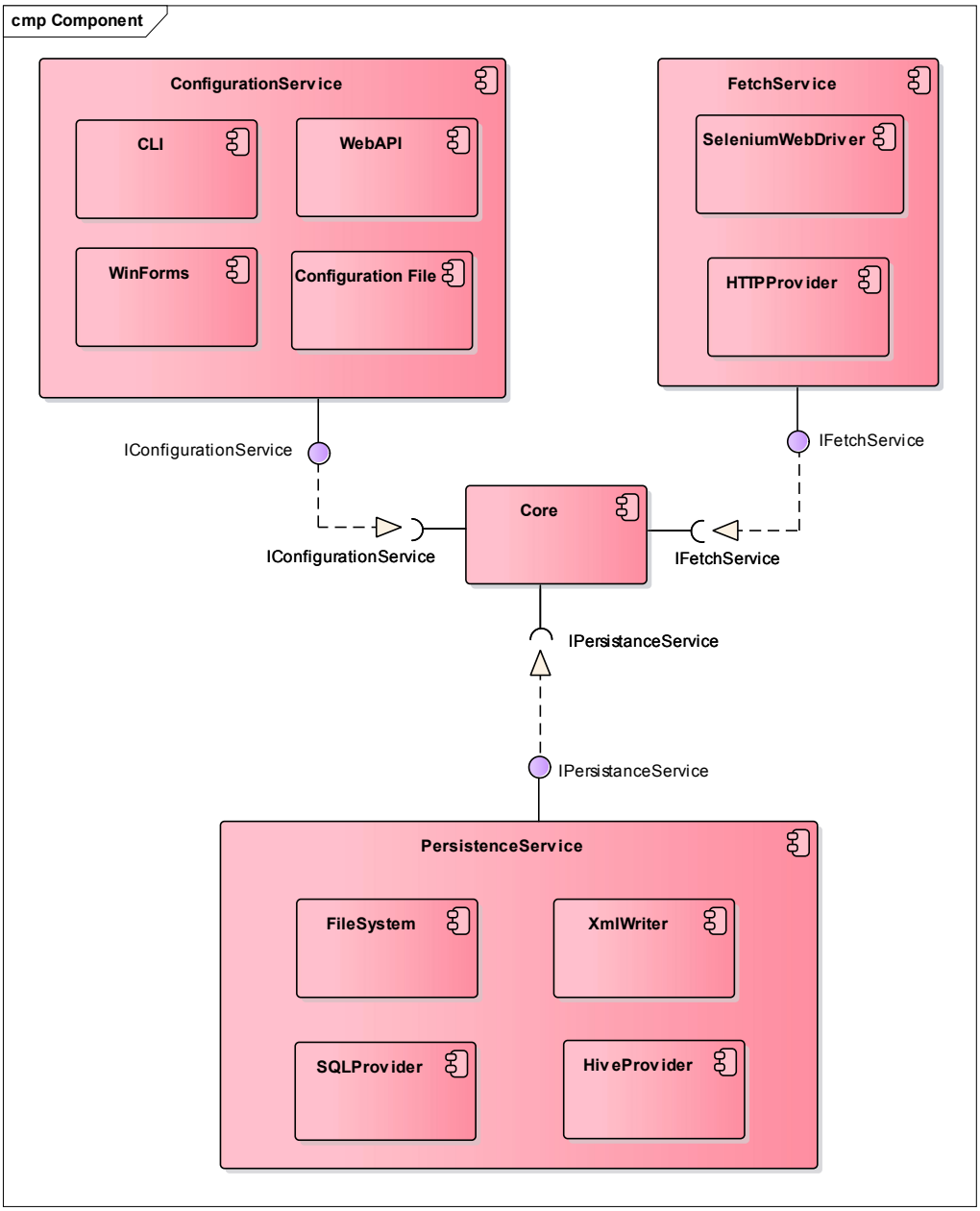

Рисунок 2.6 – Схема структурна компонентів системи.

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.

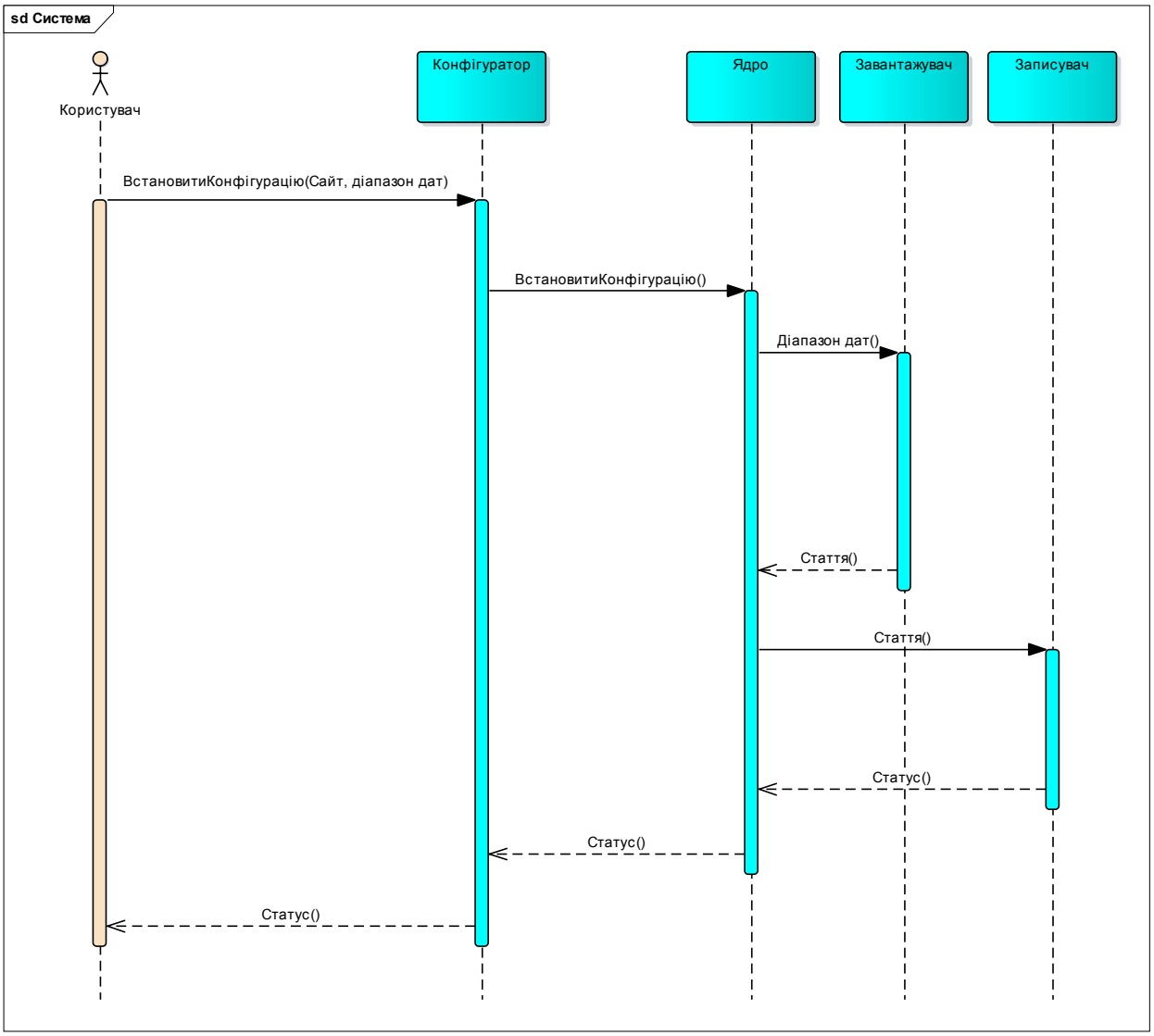

Рисунок 2.7 – Схема структурна послідовностей системи

2.3 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.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 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.

3.2 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.

3.3 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.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 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).

4.1.1 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.

4.2 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