НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Платформа для розроблення бекенд новинних агрегаторів»

Виконала:
студентка IV курсу, групи ІТ-51
Братусь Надія Вікторівна _____

Керівник:
к.т.н. Писаренко А.В.
_____

Рецензент:
_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Платформа для розроблення бекенд новинних агрегаторів»

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

ЗАВДАННЯ
на дипломний проект студенту
Братусь Надія Вікторівна

1.Тема проекту «Платформа для розроблення бекенд новинних агрегаторів», керівник проекту Писаренко Андрій Володимирович, к.т.н., затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту 4.06.2019 р.

3. Вихідні дані до проекту: платформа має забезпечувати бекендом багато клієнтських додатків одночасно. Необхідно забезпечити можливість заміни реалізації за замовчуванням для довільних додатків. Платформа має бути кросплатформенною. Необхідні характеристики робочої машини - 800 Мб вільного дискового простору, 8 Гб оперативної памяті, процессор 2.5 ГГц.

4. Зміст пояснювальної записки: Вступ; Огляд існуючих технічних рішень; Аналіз вимог до програмного забезпечення; Моделювання та конструювання програмного забезпечення; Аналіз якості та тестування програмного забезпечення; Впровадження та супровід програмного забезпечення; Висновки; Перелік посилань

5. Перелік графічного матеріалу: Лист 1. BPMN-діаграма клієнтського запиту; Лист 2. ER-діаграма бази даних; Лист 3. Діаграма послідовності обробки користувацького запиту; Лист 4. Діаграма компонентів.

7. Дата видачі завдання 10.04.2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
	Отримання завдання	10.04.2019 р.	
	Аналіз предметної області	17.04.2019 р.	
	Розробка схеми діаграми діяльності	25.04.2019 р.	
	Проектування схеми бази даних	04.05.2019 р.	
	Розробка архітектури	07.05.2019 р.	
	Реалізація проекту	15.05.2019 р.	
	Тестування програмного засобу	26.05.2019 р.	
	Подання проекту	04.06.2019 р.	

Студентка Н. В. Братусь

Керівник проекту А. В. Писаренко

АНОТАЦІЯ

Пояснювальна записка до дипломного проекту: 70 с., 14 рис., 28 табл., 4 додатки, 28 джерел.

Об’єкт дослідження: робота бекенд серверу новинного агрегатора.

Мета дипломного проекту: розробка платформи, на базі якої можна побудувати універсальний бекенд новинного агрегатора.

У проекті був проведений аналіз роботи серверної частини новинних агрегаторів, та було реалізовано власне серверне рішення, яке може одночасно обслуговувати велику кількість застосунків-клієнтів.

У першому розділі були проаналізовані існуючі рішення. Були розглянуті платформи, які дають можливість створити бекенд, а також популярні новинні агрегатори.

У другому розділі була проаналізована предметна область та вимоги до програмного забезпечення. Були розроблені функціональні і нефункціональні вимоги, були наведені варіанти використання та їх структурна схема.

У третьому розділі описана архітектура, діаграма компонентів, специфікація методів та класів, розроблена структура бази даних, обґрунтовані використані підходи та інструменти до написання програми.

У четвертому розділі описаний розроблений план тестування та проведено тестування, описані його результати.

У п’ятому розділі представлено, яким чином розгортати розроблене програмне забезпечення. Також наведена структура користувацьких запитів.

НОВИНИ, НОВИННИЙ АГРЕГАТОР, БЕКЕНД, ПЛАГІНИ, ЗАСТОСУНОК, АГРЕГАТОР, ВЕБ ЗАСТОСУНОК, СЕРВЕРНІ СИСТЕМИ, СТАТТІ.

ABSTRACT

Explanatory note to the degree project: 70 pp., 14 fig., 28 table., 4 applications., 28 references.

The object of study: providing backend services to the news aggregator server.

The aim of the degree project: development of platform by which universal backend for a news aggregator can be built.

The project analyzed the work of the backend of news aggregators, and implemented its own backend solution, which can simultaneously serve a large number of client applications.

In the first section, existing solutions were analyzed. Were reviewed platforms that provide backend creation for applications, as well as popular news aggregators. The advantages and disadvantages for these platforms were identified.

In the second section, the domain and software requirements were analyzed. non-functional and functional requirements were described, use cases description and use-case diagram were developed.

In the third section architecture diagram, component diagrams, method specifications, approaches were described. Selected technologies and tools were substantiated. Database ER diagram was developed.

The fourth section describes the developed test plan and tests it, describes its results.

In the fourth section plan of testing and describes the results of the test.

The fifth section presents how to deploy this software. The general structure for user requests is also defined.

NEWS, NEWS AGGREGATOR, BACKEND, plugin, APPLICATION, AGGREGATOR, WEB APPLICATIONS, SERVER SYSTEMS, ARTICLES.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

ТР – теорія розкладів;

API – прикладний програмний інтерфейс (англ. application programming interface);

REST – передача стану подання (англ. Representation State Transfer);

XML – розширювана мова розмітки (англ. Extensible Markup Language).

UI – користувацький інтерфейс, який забезпечує зручну взаємодію користувача з системою (англ. User Interface);

SQL – мова структурованих запитів (англ. Structured query language);

MVC – конструкційний шаблон проектування(англ. Model-view-controller);

БД – база даних (англ. database);

IoC – інверсія управління (англ. Inversion of Control);

HTTP – прикладний протокол передачі даних (англ. Hyper Text Transfer Protocol);

UML – є стандартизованою мовою моделювання (англ. Unified Modeling Language);

Плаґін – незалежно скомпільований програмний модуль, що динамічно підключається до основної програми (англ. plug-in — підключати);

BPMN – нотація для моделювання бізнес-процесів (англ. Business Process Model and Notation);

MBaaS – бекенд як сервіс (англ. Mobile backend as a service);

ПС – програмна система;

SOA – сервіс орієнтована архітектура (англ Service Oriented Architecture);

Linq – мова інтегрованих запитів (англ. Language Integrated Query)

ВСТУП

Останнім часом кількість інформації зростає в геометричній прогресії. Звичайні користувачі легко губляться в таких обсягах. За статистикою, одну п'яту часу користувач витрачає на пошук інформації і дві п'ятих на пошуки вже знайомої інформації повторно. Це робить зручне подання інформації для користувача суттєвою перевагою будь-якого сервісу перед іншими.

На сьогодні існують тисячі ресурсів, які генерують різноманітні новини. Занадто велика кількість ресурсів ускладнює їх пошук та сприйняття новин користувачем, оскільки для отримання вичерпних відомостей відносно якоїсь події, йому потрібно відкривати декілька різних джерел. Одним з наслідків цього стає повторний перегляд вже прочитаних статей, які були опубліковані на іншому ресурсі. Це може забирати значну частину часу, а враховуючи швидкий темп сьогоднішнього життя, далеко не для всіх такий варіант прийнятний.

Різноманітні агрегатори, ще недавно мали досить обмежене застосування, сьогодні ж набули великого поширення. Агрегаторами контенту прийнято називати сайти і застосунки, що вбирають в себе інформацію з інших ресурсів, таких як сайти, газети, журнали, соціальні медіа, магазини, і подають цю інформацію в зручному для кінцевого користувача виді. Мається на увазі групування даних, вибір певних категорій новин чи конкретного переліку джерел, які цікавлять користувача. Новинні агрегатори постачають користувачу такі ресурси як новинні та журнальні статті. В більшості випадків вони надають можливість використання фільтрів, пошуку за певною категорією, а також персоналізовану видачу.

Часто для кожного застосунку

використовується окремий бекенд-сервіс, таким чином при великій кількості додатків їх адміністрування буде ускладнюватись. Додатковим недоліком цього є те, що контент буде дублюватись для кожного застосунку,

якщо використовуються спільні ресурси. Таким чином оптимізувати зберігання і пошук даних потрібно буде для кожного застосунку окремо.

Задача розроблюваної системи полягає у створенні універсальної бекенд-платформи, на базі якої можна побудувати новинний агрегатор. Особливістю її є можливість одного бекенду обслуговувати різні застосунки, в тому числі такі, які працюють в різних країнах і які маоють різну логіку зберігання і обробки інформації.

Окрім забезпечення реалізації за замовчуванням, платформа має надавати можливість відносно простим способом розширювати і змінювати поведінку застосунку, наприклад додати новий спосіб зберігання користувацьких даних. В розроблюваній системі це реалізовано за допомогою системи плагінів, які можуть бути підключені для довільного застосунку чи групи застосунків.

Таким чином може бути створена довільна кількість сайтів і мобільних застосунків, на базі одного сервера. Це зменшує витрати на обслуговування серверів, а також спрощує адміністрування застосунків.

1 ОГЛЯД ІСНУЮЧИХ ТЕХНІЧНИХ РІШЕНЬ

1.1 Платформа WordPress

Буде розглянуто новинний агрегатор створений на базі WordPress. Дана платформа є однією з найпопулярніших з безкоштовних платформ для створення веб-сайтів, за даними досліджень більше 30% всіх сайтів у світі працюють на базі WordPress [1].

Створення агрегатору складається з декількох простих кроків. Всі дії виконуються в робочому вікні програми, яке зображено на рисунку 1.1. Для початку необхідно обрати дизайн. Придатність для сайту агрегатора новин WordPress означає тему, здатну відображати велику кількість заголовків. Таких тем достатньо серед стандартних. Після цього необхідно встановити модуль агрегатора RSS. Перевагами є можливість встановити його один раз, і все автоматизувати, а також можливість імпортувати статті, як пости. Після цього можна імпортувати дані з сайтів які мають RSS-канал, для цього необхідно налаштувати публікацію постів з відповідного каналу.

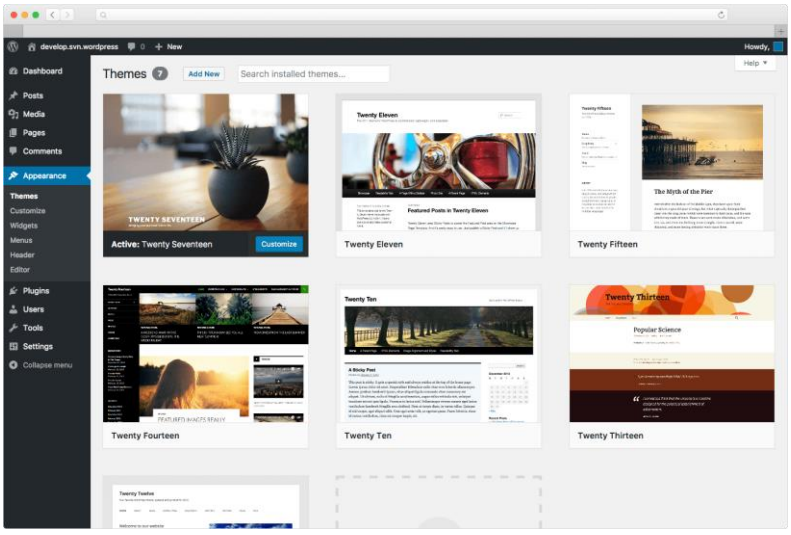

Рисунок 1.1 – Приклад робочого вікна платформи WordPress

Переваги WordPress:

– простота – ця платформа не вимагає знань html чи мов програмування;

– плагіни - попередньо встановлена функція плагінів і шаблонів дозволяє встановлювати їх дуже легко;

– спільнота – в разі якихось неполадок або проблем можна звернутись до спільноти, яка може допомогти;

– шаблони – тисячі готових до використання шаблонів;

– SEO – є підказки для оптимізації;

– дружність до мобільних пристроїв – вбудовані теми добре інтегровані для відображення на мобільних пристроях;

– бекоштовність.

Недоліки WordPress при створенні новинного агрегатора:

– для користувацьких змін, та модифікації вбудованих тем необхідно знати html, css, php, це нівелює першу перевагу;

– низька ефективність – один екземпляр може витримувати до 10 тисяч користувачів, таким чином, для забезпечення високої пропускної здатності, потрібно буде використовувати невиправдано велику кількість серверів;

– плагіни – плагіни також знижують продуктивність, крім того існує багато реалізацій однієї функціональності від різних авторів, таким чином вибір плагіну може стати проблемою.

Таким чином при створенні агрегатору може бути обґрунтовано використаний WordPress, за умови якщо цей агрегатор буде мати маленькі масштаби. В такому разі WordPress зможе задовольнити вимогам, від дасть можливість створити простий, зручний, в достатній мірі оптимізований сайт, який може створити людина при незначному бюджеті, і з обмеженими знаннями в сфері IT. Для проектів середнього і великому розміру, використання данної платформи не виправдане[2].

1.2 Платформа Appery

При розгляді можливих аналогів було розглянуто платформу Appery. Вона представляє собою реалізацію MBaaS, тобто бекенд як сервіс. Дана

платформа надає можливості налаштування і розширення бекенду, завдяки чому розробник може сконцентруватись на рішенні бізнес-задач, а не розробці інфраструктури. В системі реалізовані базові функції, які необхідні для більшості веб-додатків – завантаження файлів, оповіщення, API, система виконання задач, плагіни. Також значною перевагою є графічний інтерфейс, зображений на рисунку 1.2, за допомогою якого можна виконувати налаштування.

На базі даної платформи можна побудувати новинний агрегатор. Для цього необхідно буде додати модуль збору інформації, налаштувати пошук, а також персоналізацію. Для останнього потрібно буде додати додаткові дані до стандартного профілю.

Для роботи необхідно буде налаштувати базу даних. Це також робиться через графічний інтерфейс. Це є одночасно і перевагою і недоліком. Завдяки інтерфейсу робота з базою даних стає простою, якщо немає складних зав’язків, які будуть необхідні, зважаючи на специфіку області застосування. В нашому випадку доцільніше використовувати автоматичну генерацію бази даних, на основі коду.

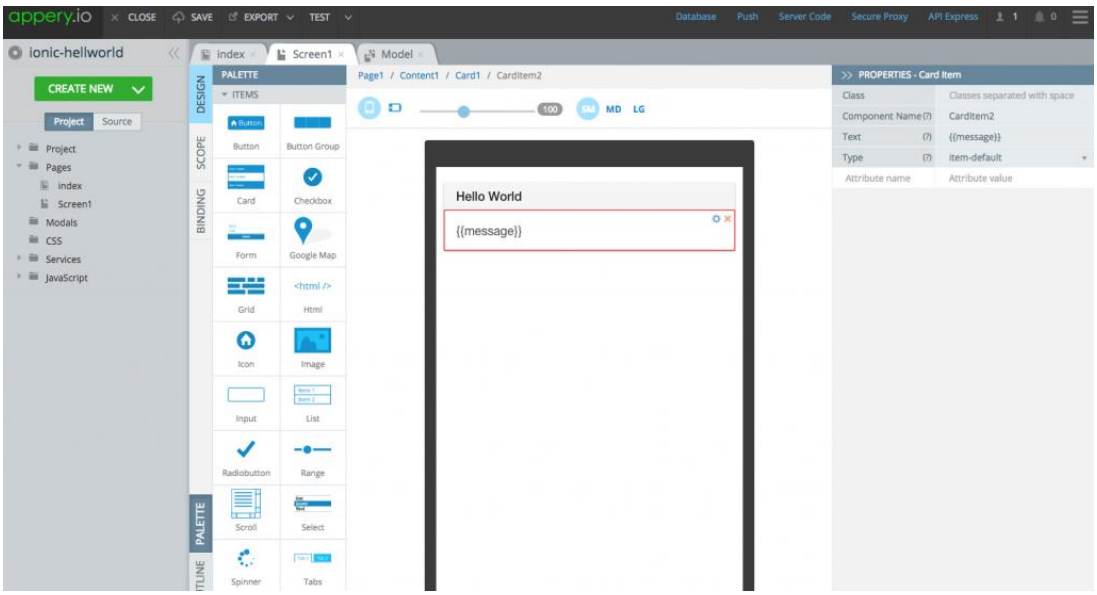

Рисунок 1.2 – Приклад робочого вікна платформи Appery

Потужності Appery достатньо для забезпечення високої кількості одночасних користувачів, що без сумніву є значною перевагою.

Серед обмежень які накладає Appery, варто зазначити, що замовник не матиме безпосереднього контролю над веб серверами та серверами баз даних. Це збільшує час, необхідний на внесення змін та виправлення недоліків. Налаштування зберігання персональних даних для додатків, які працюють на території країн в яких діє специфічне законодавство також буде викликати деякі труднощі. Зважаючи на необхідність зберігання значних обсягів інформації, її індексації та обробки, цей недолік може стати критичним.

Дана платформа підходить для створення серверної частини для новинного агрегатора, але не забезпечить універсальність, а також можливість функціонування кількох окремих додатків-клієнтів, на базі одного екземпляру бекенду.

1.3 Платформа News360

Окрім платформ, призначених для розробки сайтів, було розглянуто також популярні новинні агрегатори, які існують на сьогоднішній день. Це дозволило краще зрозуміти, які функції є найбільш затребуваними для агрегатору, а також ознайомитись з їх можливою реалізацією. Це необхідно для кращого розуміння, для яких саме частин необхідно передбачити можливість розширення плагінами.

Один з найбільших новинних агрегаторів, представлених на сьогоднішній день є News360. Для роботи з News360 можна використовувати додаток для Android чи IOS, а також веб-сайт.

При реєстрації можна вказати теги, які цікавлять користувача, після чого прейти до читання, це зображено на рисунку 1.3. Після користувач переміститься на основний екран з плиткою статей, які він може читати. Персоналізація виконується не лише за допомогою налаштувань користувачем, а також проводиться аналіз прочитаних, збережених і вподобаних статей.

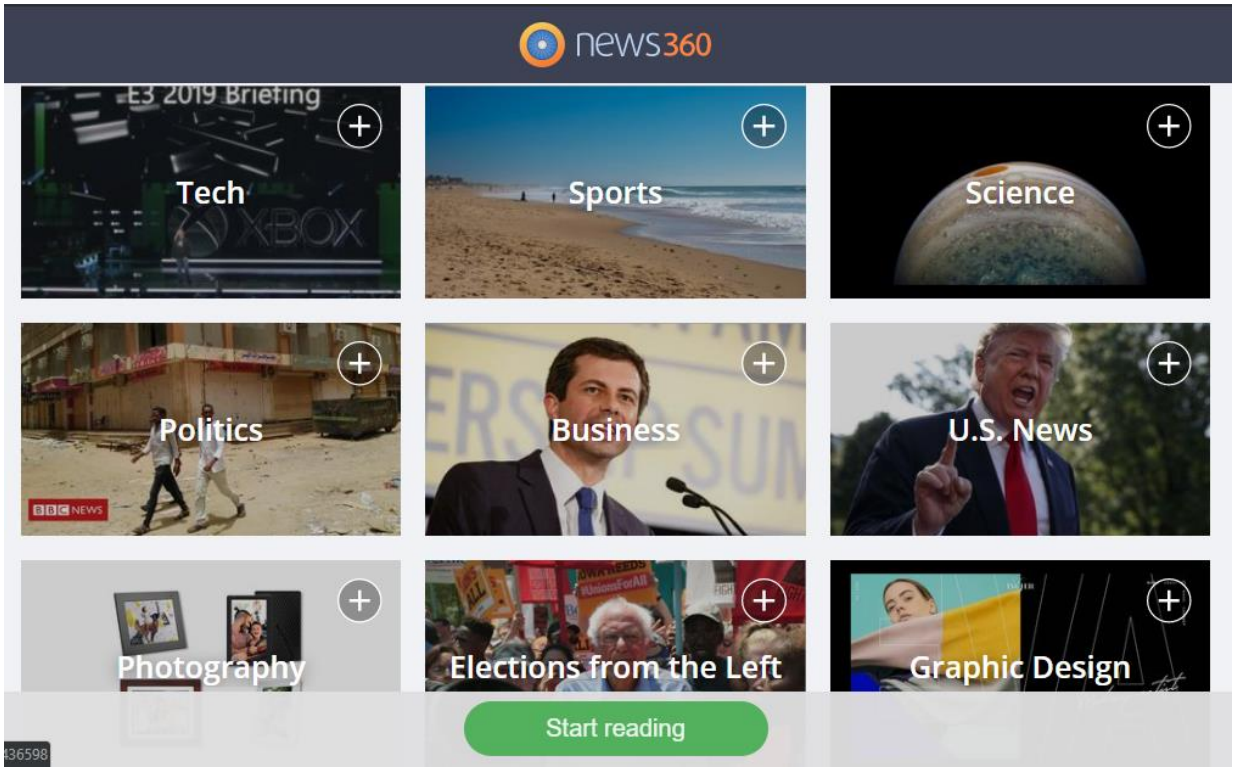

Рисунок 1.3 – Приклад вікна сайту News360

Переваги News360:

– легкість старту – за декілька кліків, користувач може отримати персоналізовану стрічку новин;

– доступність як на мобільниї пристроях так і з браузера;

– можлвиість вибору користувачем тем;

– корекція персоналізованої видачі в залежності від активності користувача;

– можлвиість отримувати локальні новини;

– може пропонувати статті, які найбільш популярні в певному регіоні, але кількість регіонів дуже мала.

Недоліки News360:

– неможливість отримати дерево категорій статей;

– обмеженість джерел інформації;

– відсутність групування статей і можливості порівняння декількох джерел;

– при виборі тем, які цікавлять користувача, у випадку якщо певна тема не знаходиться на початку видачі, знайти її буде проблематично;

– перенаправлення на інші ресурси для безпосереднього читання;

– статті англійською мовою[3].

Отже, цей новинний агрегатор має приємний інтерфейс, не перевантажений деталями та складними налаштуваннями, в якому леко розбіратись, надає персоналізовану видачу з врахуванням географічного положення. Є чудовим прикладом новинного агрегатору для широкого кола користувачів, при цьому він більше орієнтований на підтримку загального рівня обізнаності, оскільки немає можливості обрати джерела, з яких отримувати статті, а також можливості порівняння освітлення однієї події у різних авторів.

1.4 Платформа NewsNow

NewsNow.co.uk - перша і провідна незалежна служба агрегації новин в Великобританії. NewsNow, повністю автоматизований і постійно відображає заголовки, що посилаються на новинні сайти по всьому світу [4]. Сайт має більше 14 мільйонів унікальних користувачів за місяць.

Має механізм подачі заявок, який в режимі реального часу зіставляє останні новинні статті з специфікаціями, темами на основі ключових слів, завдяки чому надає релевантні посилання читачам за кілька секунд.

Сайт надає доступ і посилання на джерела з відкритим доступом, а також на такі джерела, які вимагать платну підписку чи мікротразакції. Таким чином інформаційний обхват розширюється, і можлива розбота з різиними стратегіями розповсюдження контенту.

На відміну від попереднього агрегатору, NewsNow має дуже насичений інтерфейс, див. рисунок 1.4. Так є можливість обрати певну категорію з ієрархічної структури. Сама структура має досить широкий обхват, і

вкладеність. Таким чином, можна обрати доволі вузьку тему, і отримати відповідні статті миттєво.

Варто відмітити групування статей певної тематики. Це дає можливість отримати всесторонній огляд певної події. Для статей помічається країна написання, що теж дає можливість для більш глибокого аналізу отриманої інформації. Для кожної з базових категорій використовується свій колір.

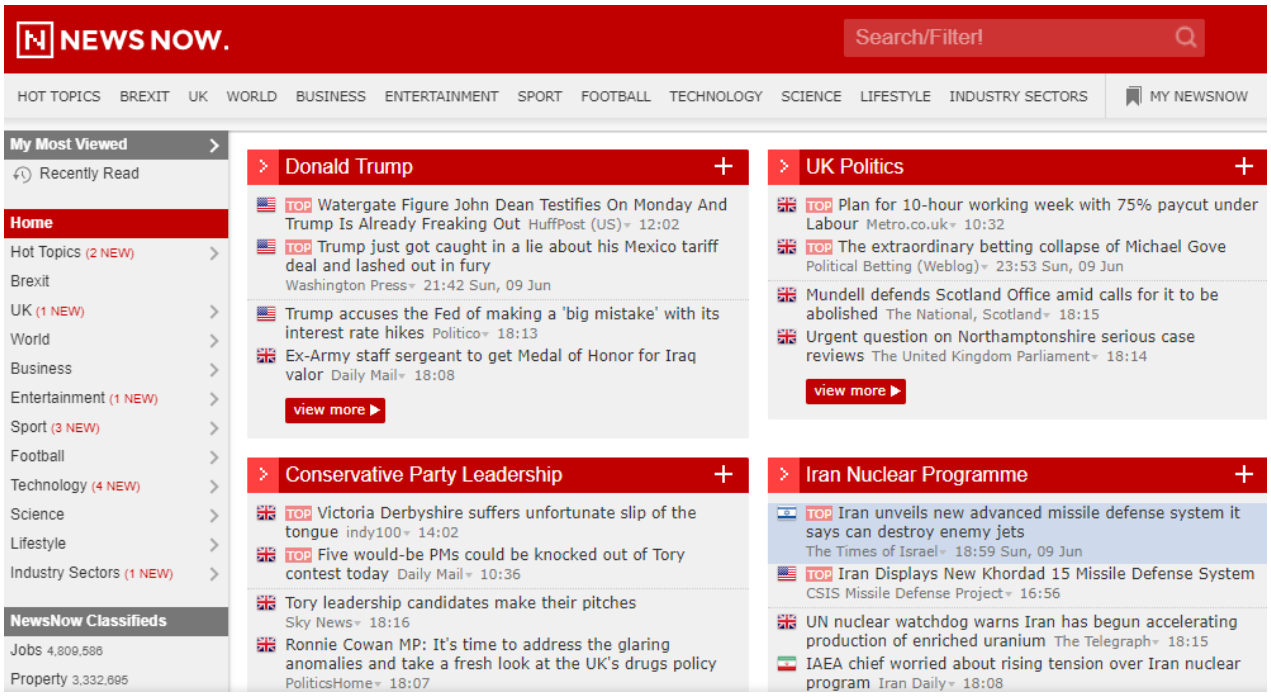

Рисунок 1.4 – Приклад вікна сайту NewsNow

Переваги NewsNow:

– велика кількість джерел;

– підтримка різних стратегій розповсюдження контенту;

– можливість обрати вузькоспеціалізовану тему за декілька кліків;

– статті публікуються на різних мовах, таким чином можна читати українські новини мовою оригіналу;

– автоматична категорізація статей на основі власних алгоритмів;

Недоліком NewsNow:

– деяка перевантаженість інтерфейсу;

– перенаправлення на інші ресурси для безпосереднього читання;

– внаслідок перенаправлення швидкість падає.

NewsNow розрахований на більш серйозних користувачів. Основною перевагою є категорізація новин, при цьому формування персональної стрічки програє News360. Сайт надає більш всестороннє освітлення подій, і орієнтований на серйознішу цільову групу користувачів. В інтерфейсі набагато менше графічного матеріалу для статті. Таким чином якщо News360 можна назвати розважальним новинним агрегатором, стрічку якої можна безцільно переглядати, NewsNow є представником більш серйозної плаформи, орієнтованої на швидке отримання інформації на задану тему, без ніякої надмірності.

1.5 Висновки до розділу

Розглянуті аналоги надають можливість створити на їх основі сайти і застосунки, в тому числі і новинний агрегатор. WordPress дає можливість створити простий сайт, а Appery окрім того і мобільний застосунок. При цьому при збільшенні складності бізнес-логіки, складність роботи з платформами значно зростає. Також з’являються складнощі в роботі зі зберіганням даних, так як вони дають обмежений контроль над цим аспектом. Також ці системи не дають можливості створити універсальну серверну частину для кількох додатків.

Також було розглянуто два новинних агрегатори, які займають різні ніші. Їх порівняння покращило розуміння того, з яких основних компонентів складається новинний агрегатор, які його частини можуть мати різну реалізацію.

2 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Загальні положення

Новинний агрегатор призначений для отримання користувачем інформації в зручній формі. Це доволі розмите поняття, і різні користувачі можуть мати різні уявлення про те якою саме має це має бути реалізовано, також як і для різних країн, з врахування менталітету та інших регіональних особливостей. Та все ж можна виділити необхідні умови, які має задовольняти агрегатор, який можна назвати хорошим. До таких умов відносяться швидкість роботи, можливість персоналізації для користувача, пошуку по категоріям, тегам, актуальність інформації, зручний користувацький інтерфейс.

Зараз дуже поширена розробка програмних засобів через веб як поєднання API так одного або декількох користувацьких представлень. Таким чином можна розробити веб-сайт та мобільні застосунки з мінімальним повторенням коду. Вся бізнес-логіка, така як адміністрування, налаштування роботи, керування даними і користувачами, реалізується в бекенді. Завдяки чому можна спростити та оптимізувати процес розробки. Зазвичай бекенд сайту є специфічним для одного застосунку, оскільки залежить від даних і специфіки. В видку новинних агрегаторів можна розробити універсальний бекенд, який може бути розширений без доступу до вихідного коду завдяки системі плагінів.

В роботі новинного агрегатору є багато типових задач, які матимуть однотипну реалізацію. При цьому, якщо говорити про універсальний бекенд, його основним завданням не є безпосередня реалізація всіх можливих варіантів використання, а можливість легкого розширення функціоналу, та заміни алгоритмів. Основними змінними частинами, які будуть впливати на функціональну якість такого програмного забезпечення будуть модуль-парсер, та модуль форматування даних. Перший відповідальний за збір інформації з

багатьох джерел, других за приведення даних до формату, який прийнятний для застосунку.

Існує два основних принципи збору інформації – автоматичний та ручний. При автоматичному інформація збирається, групується та компонується за допомогою спеціальних алгоритмів, без втручання або з обмеженим втручанням людини. При ручному ж цю роботу виконують редактори, які з використанням спеціалізованого програмного забезпечення виконують вибір, категоризацію та модерування контенту. Обидва способи мають право на життя і сфери застосування. Наприклад, ручний більш доречний при спеціалізованій і професійній направленості ресурсу, в якому акцент робиться на якості контенту.

Говорячи ж про новинний агрегатор, актуальність так широкий обхват інформації є набагато важливішим. В цьому випадку вибір автоматизованого збору інформації є набагато доречнішим, оскільки дає можливість отримувати новини з безлічі джерел, в практично не обмежених об’ємах та максимально швидко. Що при ручному варіанті було б дуже дорого, оскільки потрібно було б тримати великий штат редакторів, які будуть працювати цілодобово.

Агрегатори найчастіше використовують спеціальні програмні продукти, які називають парсером, краулер або павуками. Вони з певною періодичністю, «обходять» ресурси, тематика яких необхідна. А потім інформація викладається з посиланням на джерело. Окрім цього потрібно привести інформацію до певного стандартизованого формату прийнятого агрегатором. Стандартний формат може включати як форматування тексту, структуру доданих медіа, наявність заголовків, тегів, ключових слів та відповідних категорій, джерело новини, група до якої вона відноситься, ін.

Збір інформації відбувається за домовленістю з господарями контенту і з їх ініціативи, для додаткового просування та реклами, за умови що агрегатор користується популярністю, або без попередніх домовленостей з виконанням

правил поширення, які задані джерелом. Джерелами інформації можуть бути сайти, газети, журнали, соціальні медіа.

Побудова якісної і універсальної архітектури це складне завдання, так як вимоги можуть сильно варіюватись. Розроблювана архітектура має забезпечувати можливість легко змінювати модулі за замовчуванням, без необхідності мати доступ до вихідного коду. Це буде забезпечено шляхом реалізації IoC. Програмний засіб буде доставлений замовнику з реалізацією всіх модулів за замовчуванням, та можливістю легкого розширення і зміни поведінки.

2.2 Змістовний опис і аналіз предметної області

Платформа для розроблення бекенд новинних агрегаторів має передусім забезпечувати швидкість відповіді на запит, роботи при великій кількості одночасних користувачів та можливість налаштування конкретних додатків без необхідності вносити зміни і навіть мати доступ до вихідного коду.

Для цього спершу потрібно визначити задачі, які будуть виконуватись та декомпозувати систему на модулі.

Перш за все варто вирішити задачу управління додатками, які будуть підключатися до серверної частини. Це дозволить одному екземпляру розгорнутому на сервері обслуговувати не лише реалізацію виді веб-сайту й мобільних додатків для різних платформ, а й безліч різних агрегаторів, які будуть працювати паралельно. При цьому залишається можливість отримувати статистики як окремо для кожного додатка, так і загальні статистики для порівняння показників.

Для того щоб платформа була дійсно універсальною, необхідно забезпечити спеціальні налаштування для особливих умов функціонування. Наприклад законодавство Російській Федерації, є вимагає зберігати особисті дані користувачів в дата центрах на її території. Різні застосунки можуть вимагати різних алгоритмів отримання та структуризації даних. Також в ході експлуатації застосунку можуть знадобитися перевизначити деякі функції.

Є декілька варіантів рішення подібної проблеми. При першому варіанті, розробник окрім виконуваної програми надає вихідний код, і замовник може самостійно його модифікувати своїми силами, або залучивши сторонніх розробників. При цьому зазвичай необхідна доплата за отримання вихідного коду. У даного підходу є декілька мінусів – як те, що не всі замовники бажають працювати з вихідним кодом, а надають перевагу готовому рішенню, для його розширення, необхідно залучати додаткові ресурси, для того щоб розібратись в існуючому рішенні. Другим варіантом є звернення до розробника для зміни чи розширення функціоналу. З однієї сторони це може бути вигідно, оскільки компанія розробник несе всю відповідальність за функціонування рішення, але це обмежує можливих виконавців, і ставить замовника в залежне положення. Третім варіантом є використання плагінів. Таким чином за необхідності розширення функціоналу програми, замовнику достатньо замовити написання окремого модуля, який динамічно підключиться до основної програми. Перевагою даного підходу, є те що розробник може не розкривати деталі реалізації, залишається можливості розширення. Так як це варіант являється найоптимальнішим, в дипломному проекті буде застосований саме він.

Наступною задачею, являється збір інформації з інших джерел. Джерела можна поділити на дві групи – постачальників даних, з якими налагоджено співпрацю, та збір інформації яка знаходиться в вільному доступі. Першою групою можуть бути як спеціалізовані постачальники, які самостійно сканують інформаційні ресурси, приводять її до єдиного формату, і потім надають її клієнтам. Зазвичай це спеціалізовані API які надають інформацію в форматі зручному для клієнта. Наприклад JSON, XML. Також сюди можна віднести безпосередньо ресурси, які генерують унікальний контент, і співпрацюють з агрегатором, для популяризації, реклами, чи по іншим мотивам. В цьому випадку інформація може передаватись в довільному форматі, і не завжди зручному для використання. До другої групи відносяться ресурси, які надають дозвіл для копіювання контенту за умови посилання на джерело. В цьому разі, у власника агрегатора не буде спеціальних домовленостей, з власником

контенту, і отримання даних буде виконуватися парсером, з подальшим форматуванням[5].

Задача приведення статей до формату, який використовується додатком доволі цікава. Якщо ми маємо домовленості з джерелом контенту, ми можемо розраховувати на те, що інформація буде приведено до якогось стандартного вигляду і її обробка буде тривіальною задачею. До цієї задачі відноситься як форматування тексту та медіа, так і визначення тегів, ключових слів, та категорії, до якої варто віднести статтю.

Персоналізація новинної стрічки являється критичною для новинного агрегатора. Оскільки основна задача для користувача, яку виконує новинний агрегатор, це економія часу та отримання новин які цікаві саме йому, без цієї функції розробка агрегатору не має сенсу. Перш за все це географічне профілювання – видавати користувачу локальні новини саме його регіону. Для особистої персоналізації є дві основні стратегії – користувач сам обирає коло категорій, які йому цікаві, або на додачу до цього, виконується аналіз поведінки користувача на сайті. Аналіз поведінки користувача, доволі складний і затратний, як в плані розробки, так і в плані експлуатації процес, і потрібен далеко на для кожного агрегатора, тому необхідно залишити можливість його проведення[6]. Для його роботи необхідно зберігати комплекс даних, такі як запити користувача, переглянуті статті, відмови користувача(статті, які користувач швидко відкидав), час затрачений на них. В рішенні необхідно забезпечити можливість логування подібних подій, з різним степенем деталізації для подальшої обробки.

2.3 Аналіз вимог до програмного забезпечення

Щоб створити платформу, яка буде слугувати бекенд для широкого числа додатків-клієнтів, необхідно визначитись з ролями учасників, які будуть з ним працювати. Існує дві основні ролі – адміністратор, та користувач, який взаємодіє через клієнта – це може бути сайт або мобільний застосунок.

Адміністратор – це користувач програмного забезпечення, який виконує налаштування та управління його функціоналом. Він займається реєстрацією додатків, та налаштуванням їх конфігурації, яка включає управління інформаційними джерелами. Це включає додавання та видалення джерел, а також встановлення правил обробки даних.

Клієнт – це застосунок, який передає запити кінцевого користувача, та відображає їх у зручній користувачу формі. Клієнт може переглядати новини використовуючи фільтри, створити власний профіль з налаштуванням вподобань, зберігати вподобані статті [7].

Система передбачає наступні варіанти використання, які описані в таблицях 2.1 – 2.10.

Таблиця 2.1 – Варіант використання UC001

Назва	Реєстрація нового клієнту
Опис	Адміністратор може зареєструвати новий застосунок-клієнт
Учасники	Адміністратор
Передумови	Адміністратор увійшов у систему
Постумови	Клієнт додано, для нього згенеровано ключі доступу, та створено налаштування. З заданими ключами можна отримати доступ до системи.
Основний сценарій	– Адміністратор заходить на сторінку клієнтів. – Адміністратор натискає на кнопку «Додати». – Адміністратор уводить параметри застосунку. – Адміністратор натискає кнопку «Зберегти». – Система реєструє клієнта і генерує для нього ключі доступу. – Система відображає вікно з списком клієнтів.

Продовження Таблиці 2.1

Розширення сценаріїв	а) Якщо адміністратор не ввів назву клієнта: 1) система пропонує повторне введення назви.

Таблиця 2.2 – Варіант використання UC002

Назва	Додавання нового інформаційного джерела
Опис	Адміністратор може зареєструвати нове джерело і правила обробки, вибрати клієнтів, які можуть використовувати джерело.
Учасники	Адміністратор
Передумови	Адміністратор увійшов у систему
Основний сценарій	– Адміністратор заходить на сторінку інформаційних джерел. – Адміністратор натискає на кнопку «Додати». – Адміністратор уводить параметри джерела і список клієнтів. – Адміністратор натискає кнопку «Зберегти». – Система додає джерело. Система відображає вікно з списком джерел.
Розширення сценаріїв	б) Якщо адміністратор не ввів назву джерела а також url доступу: – система пропонує повторне введення даних.

Таблиця 2.3 – Варіант використання UC003

Назва	Створення задачі планувальника
Опис	Адміністратор може створювати задачу планувальника для збору даних з інформаційних джерел.

Продовження Таблиці 2.3

Учасники	Адміністратор
Передумови	Адміністратор увійшов у систему
Постумови	Налаштування планувальника оновлені. Розклад оновлений згідно з новими налаштуваннями.
Основний сценарій	– Адміністратор заходить на вікно планувальника. – Адміністратор натискає на кнопку «Створити». – Адміністратор уводить налаштування планувальника – джерело і частоту виконання. – Адміністратор натискає кнопку «Зберегти». – Система додає задачу до списку планувальника. – Система відображає вікно налаштувань планувальника.
Розширення сценаріїв	Якщо дані не пройшли валідацію, вивести повідомлення про помилку і запропонувати виправити введені дані.

Таблиця 2.4 – Варіант використання UC004

Назва	Редагування налаштувань клієнта
Опис	Адміністратор може редагувати налаштування клієнта
Учасники	Адміністратор
Передумови	Адміністратор увійшов у систему
Постумови	Налаштування клієнта редаговано.
Основний сценарій	– Адміністратор натискає на обраний клієнт. – Система відображає вікно налаштувань. – Адміністратор може змінювати налаштування – Налаштування змінені

Продовження Таблиці 2.4

Розширення сценаріїв	Якщо дані не пройшли валідацію, вивести повідомлення про помилку і запропонувати виправити введені дані.

Таблиця 2.5 – Варіант використання UC005

Назва	Зміна налаштувань задачі планувальника
Опис	Адміністратор може редагувати налаштування планувальника
Учасники	Адміністратор
Передумови	Адміністратор увійшов у систему
Постумови	Налаштування планувальника змінено
Основний сценарій	– Адміністратор заходить на сторінку планувальника та обирає необхідну задачу. – Адміністратор натискає на кнопку «Змінити». – Адміністратор вводить нові дані і натискає кнопку зберегти. – Система відображає вікно з списком задач планувальника.
Розширення сценаріїв	Якщо дані не пройшли валідацію, система виводить повідомлення про помилку і пропонує виправити введені дані.

Таблиця 2.6 – Варіант використання UC006

Назва	Підключення плагіну для застосунку
Опис	Адміністратор може підключити плагін для застосунку, для зміни його функціональності або способів виконання задач
Учасники	Адміністратор

Продовження Таблиці 2.6

Передумови	Адміністратор увійшов у систему, плагін створено, скомпільовано.
Постумови	При роботі даного застосунку буде використаний плагін.
Основний сценарій	– Адміністратор заходить на сторінку додатків та обирає необхідний. – Адміністратор натискає на кнопку «Плагіни». – Адміністратор вибирає плагін зі списку доступних, активує його та нажимає кнопку «Зберегти». – Система відображає вікно з списком задач планувальника.
Розширення сценаріїв	– Якщо плагін не зареєстровано в системі, користувач натискає кнопку «Додати новий плагін», і в діалоговому вікні обирає скомпільовані файли з файлової системи.

Таблиця 2.7 – Варіант використання UC007

Назва	Отримання стрічки новин
Опис	Клієнт надсилає запит для отримання стрічки новин з користувацькими налаштуваннями
Учасники	Клієнт
Передумови	Клієнт зареєстрований в системі і має активний статус
Постумови	Клієнт отримує відповідь, у якій знаходиться список новин відповідно до запиту.
Основний сценарій	– Клієнт надсилає запит. – Система валідує запит. – Система перевіряє доступ для клієнта. – Система відбирає новини відповідно до запиту, формує і надсилає відповідь.

Продовження Таблиці 2.7

Розширення сценаріїв	а) Якщо Клієнт надсилає запит неправильної структури: 1) Система повертає відповідь з статусом 422 (Unprocessable Entity) і описом помилки. б) Якщо клієнт не зареєстрований або не активний: 1) Система повретає відповідь з статусом 401 (Unauthorized).

Таблиця 2.8 – Варіант використання UC008

Назва	Реєстрація користувача
Опис	Клієнт надсилає запит для реєстрації профілю користувача
Учасники	Клієнт
Передумови	Клієнт зареєстрований в системі і має активний статус
Постумови	У відповіді знаходиться токен доступу.
Основний сценарій	– Клієнт надсилає запит правильної структури. – Система валідує запит. – Система перевіряє існування користувача з заданим логіном в даному застосунку. – Система створює користувацький профіль. – Клієнт отримує відповідь.
Розширення сценаріїв	2) Якщо користувач вже існує система повертає відповідь з описом помилки: «Користувач з заданим логіном вже існує». в) Якщо дані не проходять валідацію на стороні сервера система повертає відповідь з описом помилки.

Таблиця 2.9 – Варіант використання UC009

Назва	Зміна вподобань в профілі користувача
Опис	Клієнт надсилає запит для зміни категорій новин, які бажає отримувати користувач
Учасники	Клієнт
Передумови	Клієнт зареєстрований в системі і має активний статус, Користувач зареєстрований в системі
Постумови	Відповідь з статусом 200.
Основний сценарій	– Клієнт надсилає запит правильної структури. – Система валідує запит(структуру і дані). – Система зберігає зміни в профілі користувача. – Клієнт отримує відповідь.
Розширення сценаріїв	Якщо дані не проходять валідацію на стороні сервера система повертає відповідь з описом помилки.

Таблиця 2.10 – Варіант використання UC010

Назва	Отримання персоналізованої стрічки новин
Опис	Клієнт надсилає запит для отримання персоналізованої стрічки новин
Учасники	Клієнт
Передумови	Клієнт зареєстрований в системі і має активний статус, користувач зареєстрований в системі
Постумови	Відповідь з статусом 200.
Основний сценарій	– Клієнт надсилає правильної структури. – Система виконує відбір і фільтрування новин відповідно до даних профілю користувача, і алгоритмів відповідного застосунку. – Клієнт отримує відповідь.

Продовження Таблиці 2.10

Розширення сценаріїв	а) Якщо дані не проходять валідацію на стороні сервера: 1) Система повертає відповідь з описом помилки.

Взявши за основу проаналізовану модель варіантів використання, ми побудували структурну схему використання, яка буде зображати основні способи використання застосунку[8] Схему можна розглянути на рисунку 2.1. На основі даної схеми в подальшому буде розроблено архітектуру систему, розділено її на основні модулі.

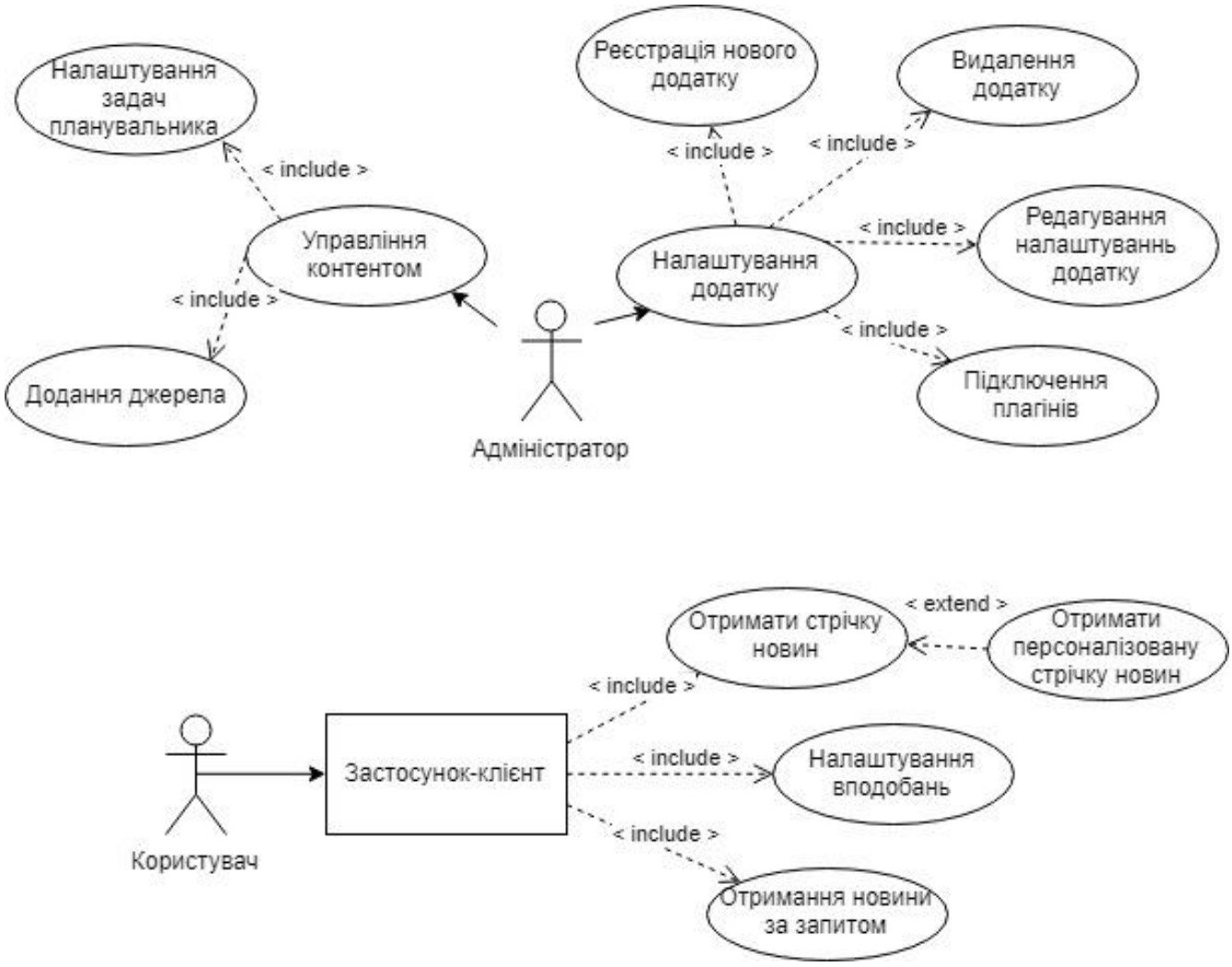

Рисунок 2.1 – Діаграма варіантів використання

2.3.1 Розроблення функціональних вимог

Аналіз вимог до програмного забезпечення включає в себе виявлення та подальше створення специфікації для функціональних та нефункціональних вимог. Розподіл вимог на функціональні і нефункціональні зумовлений їх різною природою. Так, функціональні вимоги – це перелік функцій або сервісів, які повинна надавати система, а також обмежень на дані і поводження системи при їхньому виконанні. Нефункціональні вимоги, при цьому визначають умови виконання функцій (наприклад, захист інформації у БД, аутентифікація доступу до ПС тощо) у середовищі, що безпосередньо не пов'язані з функціями, а відбивають потреби користувачів щодо їх виконання.

Результатом виконання аналізу вимог до платформи для бекенду для новинних агрегаторів було створення специфікації функціональних вимог описаних в таблицях 2.11 – 2.17.

Таблиця 2.11 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Реєстрація клієнту
Опис	Після реєстрації клієнту, його дані зберігаються в базі даних, він має доступ до API доки не буде деактивованим або видаленим.

Таблиця 2.12 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Додавання нового інформаційного джерела
Опис	Після реєстрації інформаційного джерела, його дані зберігаються і до ного можна надати доступ клієнтам.

Таблиця 2.13 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Створення задачі планувальника
Опис	Система моє створити задачу для планувальника і додати її в розклад. В заданий час задача має розпочати отримання інформації заданим способом від джерела. Отримана інформація має зберегтись в базі даних. Клієнти мають доступ до нових даних.

Таблиця 2.14 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Зміна налаштувань задачі планувальника
Опис	Система має змінити налаштування планувальника, і розпочати наступне сканування відповідно до нового розкладу і методу.

Таблиця 2.15 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Отримання персоналізованої стрічки новин
Опис	Клієнт може отримати стрічку новин, яка буде сформована на основі джерел, які підключені до застосунку, а також додаткових параметрів запиту. У випадку авторизованого користувача, персоналізація стрічки буде виконана з урахуванням його вподобань та попередньої активності. Дана функціональність має підтримувати можливість розширення, для подальшої оптимізації алгоритмів підбору за необхідності.

Таблиця 2.16 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Підключення плагіну для застосунку
Опис	При підключенні плагіну для застосунку, задана функціональність виконується не базовими модулями програмної системи, а динамічно підключеним плагіном.

Таблиця 2.17 – Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Редагування вподобань
Опис	Користувач може редагувати список категорій новин, які його цікавлять. Це включає вибір атрибутів всіх типів, які передбачені для застосунку.
Опис	Власник може видаляти медіа контент, що не має використовуватися у системі.

2.4 Розробка нефункціональних вимог

Отримання повного спектру вимог до розроблюваної системи, для подальшого проектування її структурних модулів, також були виявлені нефункціональні вимоги, які описані в таблицях 2.18 -2.22.

Таблиця 2.18 – Опис нефункціональної вимоги NFR001

Номер	NFR001
Назва	Підтримка середовища .NET Core
Опис	Програмна система моє підтримувати кросплатформене середовище . NET Core.

Таблиця 2.19 – Опис нефункціональної вимоги NFR002

Номер	NFR002
Назва	Швидкодія
Опис	Запити повинні мати малий час відповіді для того щоб користувач отримував дані не пізніше ніж за 0.200 с.

Таблиця 2.20 – Опис нефункціональної вимоги NFR003

Номер	NFR003
Назва	Кросплатформенність
Опис	Програмна система має функціонувати на операційних системах Windows та Linux.

Таблиця 2.21 – Опис нефункціональної вимоги NFR004

Номер	NFR004
Назва	Універсальність
Опис	Система моє забезпечувати функціонування декількох(до 50) окремих додатків, на базі одного екземпляра.

Таблиця 2.22 – Опис нефункціональної вимоги NFR005

Номер	NFR005
Назва	Потужність
Опис	Можливість велику кількість одночасних активних користувачів.

2.5 Висновки розділу

У Розділі 1 представлений базовий опис проблемної області, виділені основні задачі, розглянуті існуючі методи реалізації, а також сформовані вимоги до розроблюваної системи.

Були розглянуті існуючі популярні новинні агрегатори, для отримання загального розуміння про роботу новинного агрегатору. Також на основі цих даних, було визначено основні складові частини системи, і частини, яким необхідно забезпечити легку змінюваність.

Були визначені найуспішніші продукти, які мають схожу направленість і проведено порівняння з ними. На основі проведеного аналізу, були визначені переваги та недоліки розроблюваної системи в порівнянні з існуючими рішеннями.

Розроблено схему функціональних та нефункціональних вимог, які має задовольняти розроблюваний програмний засіб.

3 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Моделювання та аналіз програмного забезпечення

Для створення програмного забезпечення необхідно спроектувати бізнес-процеси. Для цього застосуємо методологію BPMN. В ході проектування було визначено основні бізнес-процеси системи:

– реєстрація застосунку;

– додавання інформаційних джерел;

– підключення плагіну для застосунку;

– планування роботи з інформаційними джерелами;

– робота з користувацькими запитами.

BPMN-діаграми наведені на рисунках 3.1-3.4

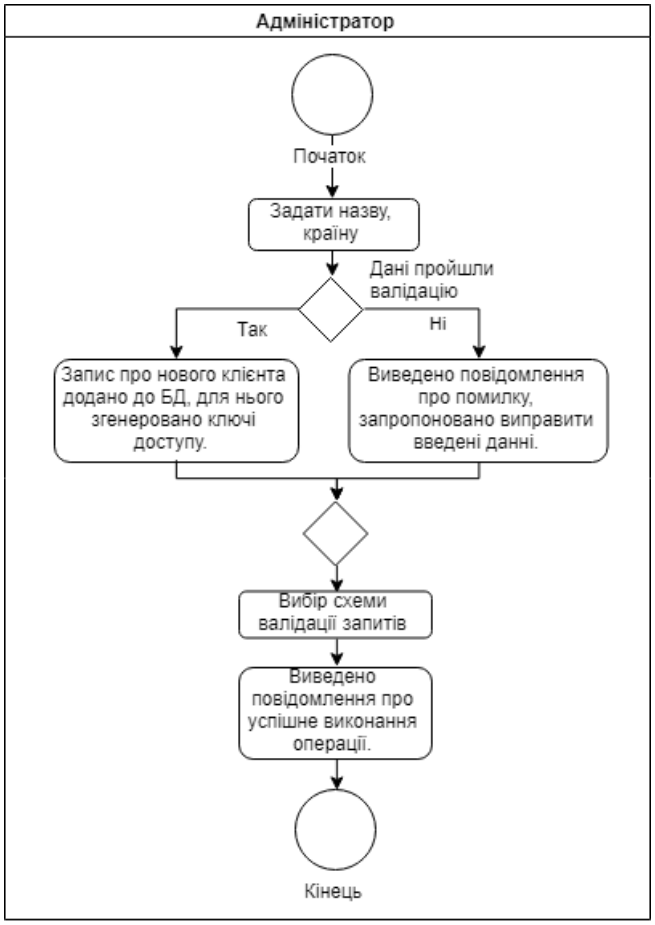

Рисунок 3.1 – BPMN-діаграма для реєстрації клієнта

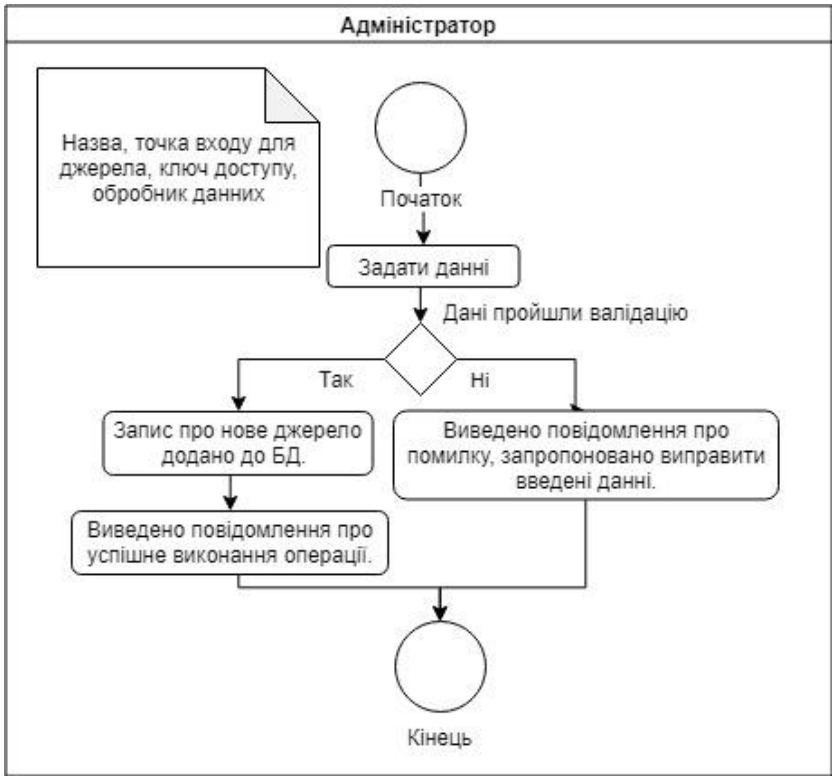

Рисунок 3.2 – BPMN-діаграма для додавання джерела

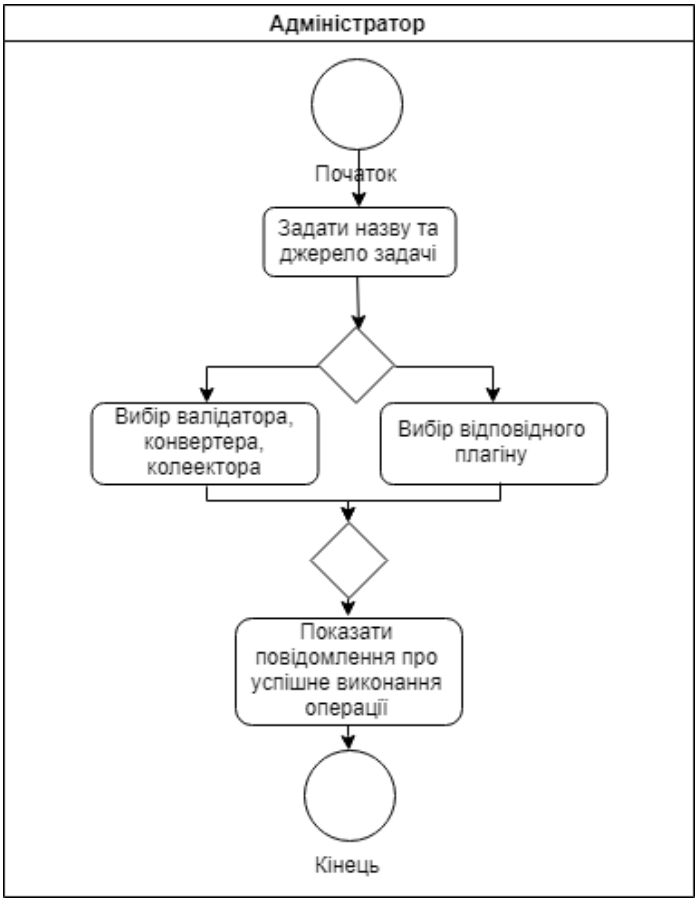

Рисунок 3.3 – BPMN-діаграма для налаштування задачі

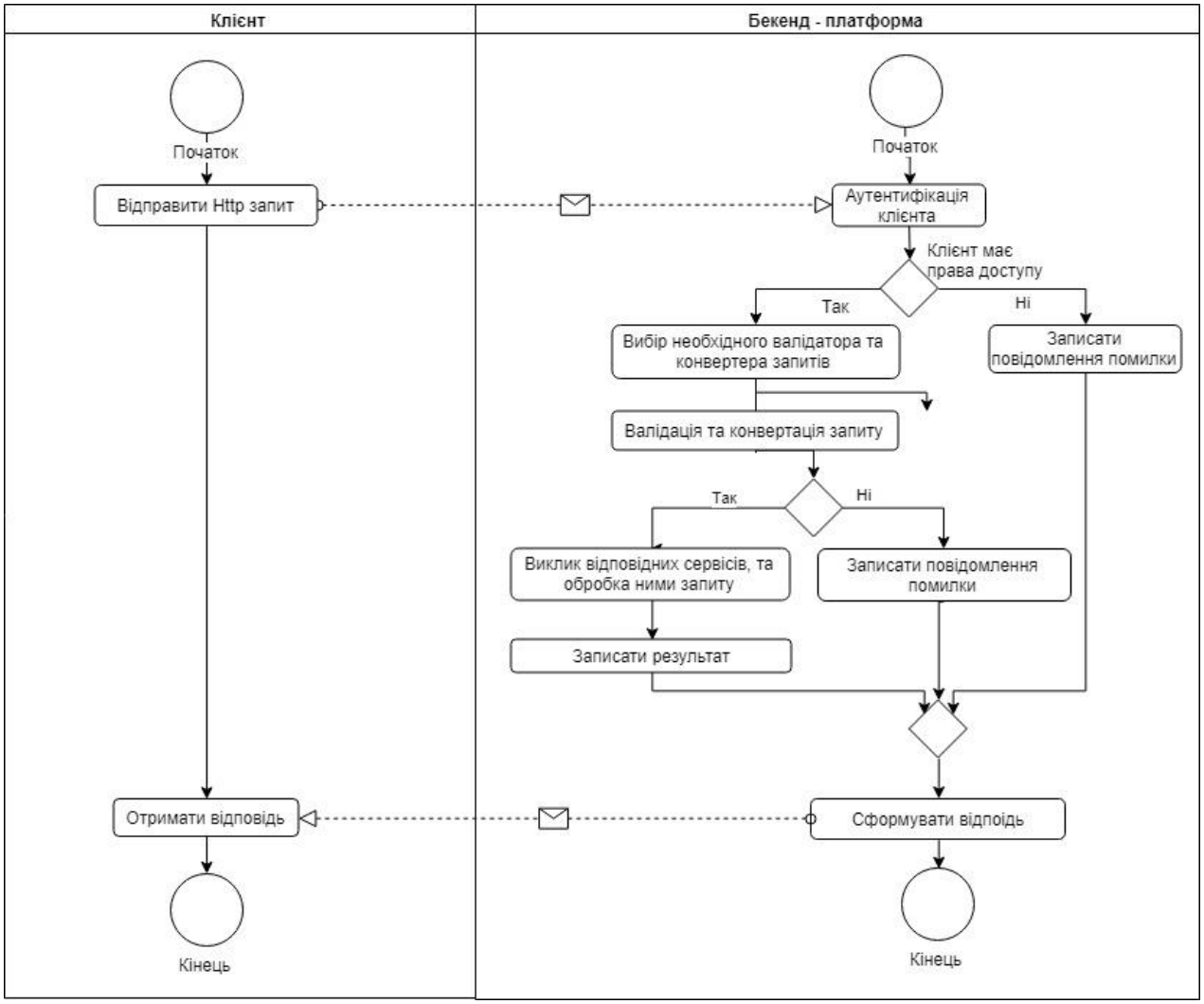

Рисунок 3.4 – BPMN-діаграма обробки клієнтського запиту

У ході моделювання системи була розроблена схема бази даних, що дозволяє зберігати інформацію про застосунки, джерела, користувачів та новини. Схема бази даних наведена в додатку А на листі 2.

Також під час процесу проектування системи була розроблена розширена діаграма обробки клієнтського запиту. Діаграма наведена в застосунку Г на листі 1.

3.2 Архітектура програмного забезпечення

Для бекенд платформи було обрано мову програмування C#. У якості середи розробки було обрано Visual Studio Ultimate Edition. Для забезпечення

максимально швидкої і простої розробки, а також для кросплатформенності було обрано платформу .Net Core.

.NET Framework — повнофункціональна платформа, яка поставляється разом з Windows. .NET Core — це модульна реалізація бібліотеки і середовища виконання, до складу якої входить піднабір .NET Framework. .NET Core працює на Windows, Mac і Linux..Також варто відмітити, що це проект з відкритим вихідним кодом.

Для реалізації проекту, можна було використати і .Net і .Net Core. В останнього є ряд суттєвих переваг:

– кросплатформенність надає більшу свободу як розробки, так і експлуатації рішення;

– легкість реалізації мікросервісів, які згодом можуть розширити функціонал прогами для додаткових можливостей, або таких, які будуть потрібні лише на якийсь час;

– при створенні і розгортанні контейнера розмір його образу набагато менше в .NET Core, ніж в .NET Framework;

– Core забезпечує вищу продуктивність і масштабованість системи.

Зважаючи на переваги які надає Core, вибір було зроблено на його користь. Також він забезпечує підтримку будь-якої обраної архітектури. Не варто забувати і про активну спільноту, яка може допомогти при виникненні труднощів.

Наразі платформа .NET Core існує для операційних систем: Windows, Linux і MacOS. Можливо, застосунок, розроблений на базі .NET Core, може бути без змін, запущений у всіх операційних системах, у яких є вказана платформа [9].

В проектуванні даного програмного рішення застосуємо підхід SOA. Сервіс-орієнтована архітектура — модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабо пов'язаних (англ. Loose coupling) замінних компонентів, які мають стандартизовані

інтерфейси для взаємодії. Такий підхід дасть можливість простіше замінювати стандартну реалізацію функцій.

Для передачі даних між сервісами оберемо асинхронну чергу, формат даних Json, для взаємодії компонентів будемо притримуватись архітектурної концепції REST.

Таким чином для зв'язку між двома компонентами, буде виконано передачу повідомлення по черзі від брокера, до отримувача. В якості черги обрано RabbitMQ, так як він задовольняє всім вимогам – забезпечує швидку, асинхронну передачу повідомлень. Порівняно з Kafka, RabbitMQ гірше масштабується, але ймовірність, що застосунку доведеться опрацьовувати трильйони повідомлень мала, тож така необхідність не з'явиться.

Слідуючи принципу SOLID і для зменшення зв'язності коду в проекті застосовано ін'єкцію залежностей. Ін'єкція залежності (англ. Dependency injection, DI) — шаблон проектування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» (англ. Inversion of control, IoC) для вирішення залежностей [10]. Відповідно до цього паттерну клас не має конфігурувати свої залежності статично, а необхідно забезпечити можливість динамічного визначення залежностей. Такий підхід дає ряд переваг. По-перше зниження звязності коду. Низький рівень звязності, тобто залежності одного модуля від іншого, необхідний для покращення архітектури застосунку. По-друге, такий підхід значно спрощує розширення програмного коду, оскільки не потрібно вносити зміни в значній кількості файлів, для заміни одного класу іншим – достатньо змінити запис в файлі налаштування контейнеру. По-третє, це спрощує модульне тестування, оскільки просто замінити реальну реалізацію на об'єкт, який імітує його роботу. По-четверте, це зменшує кількість шаблонного коду, що в свою чергу скорочує час розробки, а також зменшує можливість помилки. Цей підхід також має і деякі недоліки – при надмірному захопленні керування програмою може ускладнитись, можуть бути втрачені

деякі можливості IDE, помилки часу компіляції переходять в помилки часу виконання, що може дещо утруднити їх виявлення і усунення. Зважаючи на всі переваги та недоліки цього підходу, було вирішено доцільним застосовувати ін'єкції залежностей [11].

Для реалізації ін'єкції залежностей в проекті було використано IoC-контейнер Ninject. Він надає широкі можливості і достатньо простий в інтеграції. Ін'єкція залежностей відбувається в момент створення об'єкту. При створенні об'єкту, якщо його конструктор має параметри певного інтерфейсу, ця бібліотека знайде відповідний клас і створить його екземпляр і передасть в в конструктор. Сам клас-суб'єкт не буде мати ніяких знань ні про свої зв'язки ні про Ninject. Крім цього способу, є можливість ін'єкції через метод ініціалізації, та через властивості які позначаються спеціальним атрибутом, але ці способи програють, оскільки клас-суб'єкт, в цьому випадку буде зв'язаний з Ninject.

Для роботи Ninject був створений файл глобальної реєстрації залежностей, який наслідує NinjectModule. В цьому класі за допомогою методу Bind і його розширень для інтерфейсу призначається певний клас-реалізація, екземпляр якого і буде надаватись. Після цього треба забезпечити встановлення цих залежностей при запуску застосунку.

Для доступу до бази даних було розглянуто 2 основних бібліотеки - NHibernate та Entity Core Framework. Обидві бібліотеки забезпечують доступ до даних використовуючи об'єктно-орієнтований код. Кожна має певні переваги та недоліки.

Основні переваги NHibernate:

– краща підтримка пакетного запису і зчитування;

– можливість тонкого налаштування;

– можливість керування кешем;

– велику кількість додаткових бібліотек, які розширюють її можливості [12].

Основні переваги Entity Framework:

– краще реалізований Linq провайдер;

– це продукт Microsoft, і тому він з великою долею ймовірності проблем з ним буде менше;

– можливість вибору підходу ModelFirst, CodeFirst, DataBaseFirst, і автоматичної побудови інших компонентів;

– неконкуруючий потік вводу/виводу при роботі з базою даних [13].

На основі аналізу переваг та недоліків обох бібліотек, і враховуючи що проект учбовий для реалізації в проекті було обрано Entity Framework. Цей фреймворк надає можливість для швидкої побудови моделі, і на основі неї створення класів і генерації таблиць даних, що є досить зручним, для програміста.

Для більшої гнучкості системи і можливості використання її в різних країнах без необхідності зміни програмного коду, застосуємо систему плагінів. При обробці запиту від користувача буде викликатись певний сервіс, який за умови що для даного клієнта передбачена особлива процедура обробки буде передавати управління на плагін, який реалізовує інтерфейс сервісу. Подивитись на схему роботи можна на рисунку 3.5. Таким чином можна буде забезпечити, наприклад, зберігання персональних даних користувачів, не в базі даних за замовчуванням, а в дата центрі, на території певної країни.

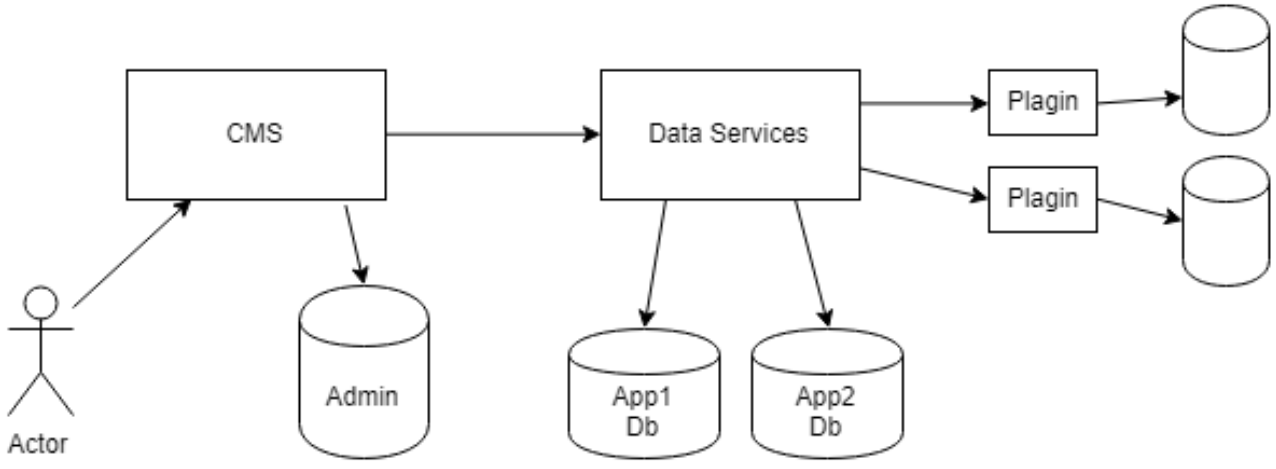

Рисунок 3.5 – Схема підключення плагінів

В ході виконання проекту було розглянуто два принципи побудови архітектури застосунку – трирівневу та onion архітектуру. В тришаровій архітектурі є три шари в ієрархічній залежності, кожен з яких може взаємодіяти лише з нижнім, це схематично зображено на рисунку 3.6 Дана архітектура доволі поширена. Основним недоліком її являється висока звязаність. Це означає що компоненти мають багато в’язків, можливе проникнення бізенс-логіки в шар користувацького інтерфейсу і шару збереження даних в бізнес-логіку[14]. Наслідком цього, стає те що модулі не можуть функціонувати автономно, і немає можливості замінити один, без внесення змін в інші. Такий проект важко підтримувати.

Рисунок 3.6 – Схема тришарової архітектури

Зважаючи на недоліки цього підходу, було обрано інший підхід - Onion (шарова) архітектура. Вона зображена на рисунку 3.7, і представляє собою кілька шарів (кількість залежить від предметної області і складності застосунку), кожен з яких має доступ лише до ближчих до центру шарів. в самому центрі знаходиться доменна модель, яка не залежить ні від чого. Наступним шаром ідуть репозиторії – точніше оголошені їх інтерфейси, а реалізовані вони на вищих шарах, оскільки вони залежить від конкретної бази даних, з якою будуть співпрацювати. Цей шар забезпечує отримання.

збереження та видалення об’єктів. В проекті бібліотека Framework представляє доменну модель, а Framework.DataAcess оголошує репозиторії та фабрики для їх створення. Наступний шар реалізує доменну логіку застосунку. Ці шари складають ядро застосунку. На зовнішньому шарі знаходяться елементи які частіше за все змінюються – користувацький інтерфейс, реалізація репозиторіїв та доступ до бази даних, а також модульні тести. Репозиторії винесені на зовнішній шар, оскільки в середньому кожні 3 роки технології збереження даних оновлюються, і ця частина застосунку може замінюватись частіше за інші [15].

Рисунок 3.7 – Схема Onion архітектури

Буде розглянуто основні складові частини нашої системи.

При запиті до API, в першу чергу запит буде переданий в модуль керування контентом (Content Management Service). Цей модуль відповідальний за ідентифікацію застосунку, до якого відноситься запит, та визначення сервісу, який має його обробити. Після чого запит перенаправляється на сервіс, який повертає результат після обробки. Така

архітектура дає можливість одночасного забезпечення бекендом декількох різних і незалежних додатків-клієнтів.

Логування винесемо в окремий сервіс. Це дасть можливість записувати логи, які відносяться до одного застосунку, з будь-якого сервісу чи плагіну. Детальніше схема зображена на рисунку 3.8. З зібраними в одному місці логами буде простіше аналізувати роботу системи, знаходити несправності. Це дає можливість зберігати логи кожного застосунку в окремий файл чи інше сховище, таке як хмара чи база даних.

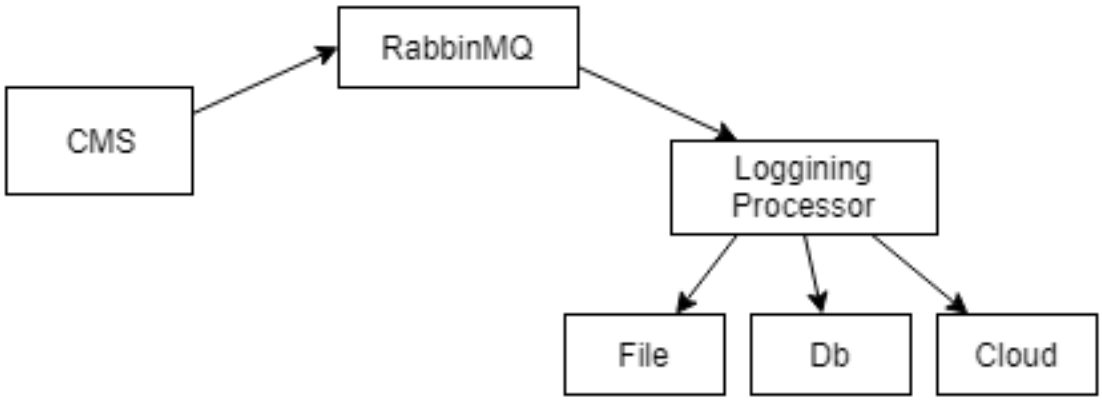

Рисунок 3.8 – Схема логування системи

Наступним компонентом на якому варто загострити увагу є планувальник. Він забезпечує збір інформації від джерел, приведення її до стандартизованої форми. Він активується за таймером, а також при настанні певної події.

Оскільки вже існують спеціалізовані сервіси, які займаються збором і наданням новинних ресурсів в структурованій і зручній для обробці формі, реалізовувати власні парсери не доцільно. При цьому залишимо можливість підключати їх як додаткові модулі, чи плагіни в майбутньому, якщо в них виникне необхідність. Для роботи з інформаційними джерелами буде достатньо використовувати конвертер, який буде приводити інформацію до формату, який використовується в застосунку. Для його роботи потрібно буде створити

налаштування джерела, в яких будуть зберігатися пари відповідних один одному ключів, або функції перетворення даних.

Для такого ресурсу важливим є швидкий пошук, а також повнотекстовий пошук. Його реалізація на базі власного застосунку, може бути доволі трудомісткою, а також буде програвати по якості і надійності готовим комерційним рішенням. Порівнявши два пошукових сервіси — Algolia і Elasticsearch. Обидва надають повну функціональність, і забезпечують швидкий пошук. Перевагою Algolia є простота інтеграції, тому в якості пошукового сервера будемо застосовувати її.

Для забезпечення можливості функціонування декількох фронтенд застосунків на базі одного екземпляру бекенд серверу було розроблено принцип областей застосування - маркетів. Маркетом називається група додатків, які мають спільну реалізацію сервісів і плагінів, а також дані яких зберігаються в одному сховищі. Дані маркету зберігаються в адміністративній базі даних. У кожного маркету є свій код, по якому він ідентифікується. Цей код використовується в іменуванні баз даних, а також доменному імені. При обробці запиту робиться спроба отримати код з доменного імені, на основі отриманого коду робиться запит до адміністративної бази даних. Випадку якщо даний Market існує виконується з'єднання з відповідною базою даних і створюються відповідні сервіси. Детальніше принцип роботи можна розглянути додатку А на листі 3.

Для адміністрування застосунку буде розроблено веб-застосунок на базі партерну MVC. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [16]. Доменне ім'я застосунку має починатися з коду

маркету. Це забезпечить незалежне адміністрування кожного маркету і відповідних застосунків незалежно від інших маркетів.

Для роботи з додатком було розроблено дві схеми бази даних – адміністративну та основну. Перша зберігає дані про маркет, друга є базою даних маркету, і зберігає дані застосунків. Список адміністративної таблиці наведено в таблиці 3.1.

Таблиця 3.1 – Опис таблиць адміністративної бази даних

Назва таблиці	Опис таблиці
Market	Таблиця зберігає доступні області застосування.
CommonJournal	Таблиця, що зберігає загальний журнал запитів – url, метод, заголовки, час обробки запиту, застосунок, код результату.

У кожного маркету вказується ідентифікатор основної бази даних, завдяки цьому, після ідентифікації застосунку, до якого було виконано запит, можна отримати необхідну основну базу даних додавши до шаблону рядку підключення ідентифікатор. Таким чином, можна забезпечити створення окремої бази даних для кожного застосунку, або ж використання спільної бази для декількох додатків, що може бути актуальним якщо групі додатків необхідно працювати з одними й тими ж самими інформаційними джерелами. Основні таблиці зображено в таблиці 3.2.

Таблиця 3.2 – Опис таблиць основної бази даних

Назва таблиці	Опис таблиці
Arcticle	Таблиця описує статтю, зберігає назву, тіло статті, джерело, тип статті, дата публікації, автор, чи являється стаття активною, посилання на джерело.
ArticleAttribute	Таблиця зберігає атрибути статей – назву, тип, активність.

Продовження Таблиці 3.2

AttribyteType	Зберігає типи атрибутів, які можуть бути присвоєні статті – категорія, тег, регіон.
ArticleAttitude	Збереження даних поро відмічені чи вподобані користувачем статті, містить посилання на користувача, статтю, наявність відмітки чи вподобання, та дати для цих подій.
Attachment	Зберігає інформацію про файли, такі як зображення, відео та ін., які можуть використовуватись в застосунку. Зберігає ідентифікатор, адресу файлу, адресу прев'ю файлу.
ArticleAttachment	Слугує для прив'язки медіа файлів до статті. Зберігає посилання на Attachment, статтю та положення зображення.
ConsumerPreferences	Таблиця зберігає вподобання користувача – атрибут статті, спосіб встановлення – ручний, чи автоматичним алгоритмом, ступінь зацікавленості користувача цим атрибутом.
Sources	Зберігає дані про інформаційні джерела – назву, url, ключі доступу
SorucesAttribute	Таблиця зберігає дані про конвертацію новинних статей до доменних об'єктів
SourcePuller	Таблиця описує користувачів задачу для стягування інформації від інформаційного джерела, точку входу, паттерн для отримання даних, частоту запуску, дату останнього запуску.
Country	Таблиця з переліком країн, до яких можуть бути прив'язані статті, та положення користувача

Продовження Таблиці 3.2

RegionType	Зберігає дані для типу регіонального розмежування, наприклад для області, міста, району, штату.
Region	Дані про регіональну одиницю, зберігає назву, код, країну, а також тип регіону.
Consumer	Таблиця зберігає основні відомості про користувача, такі як логін, пароль, дату реєстрації, дату останнього відвідування, застосунок, тип користувача.
ConsumerAggregated	Зберігає додаткову інформацію про користувача, таку як від, стать, повне імя, телефон, ін.
ConsumerType	Зберігає перелік типів користувачів.
Application	Інформація про клієнт-додаток. Зберігає код застосунку, назву, опис, дату створення, статус застосунку, ключ доступу.
SourceApplication	Зберігає зв’язки інформаційних джерел з додатками, таким чином можна використовувати одне джерело для декількох додатків, без дублювання в базі даних.
ApplicationArticle	Зберігає дані про публікацію статті для певного застосунку.

Побудовану згідно опису таблиць ER-діаграму бази даних можна побачити у додатку А на листі 2.

Узагальнюючи результати проведеної роботи було побудовано діаграму компонентів, яку можна розглянути на рисунку 3.9. Буде розроблено 6 основних компонентів. Framework відповідає за структуру сутностей та доступ до них. CMS за адміністративне керування платформою, зокрема Web реалізує

сайт на основі MVC, а Business Services забезпечують виклики відповідних сервісів бекенд. DataService виконує роботу з даними і підключення плагінів за необхідності. Модуль логування являється незалежним, і виконує запис даних відповідно до його конфігурації.

Framework складається з трьох модулів – модуль моделей(Entity), доступ до даних та інфраструктури. Побудова моделей бази даних виконано за допомогою Entity Framework. Entity Framework забезпечує роботу з базою даних на основі об'єктно орієнтованого коду. Вибір цього фреймворку зумовлений простотою його інтеграції, універсальністю і легкістю взаємодії,

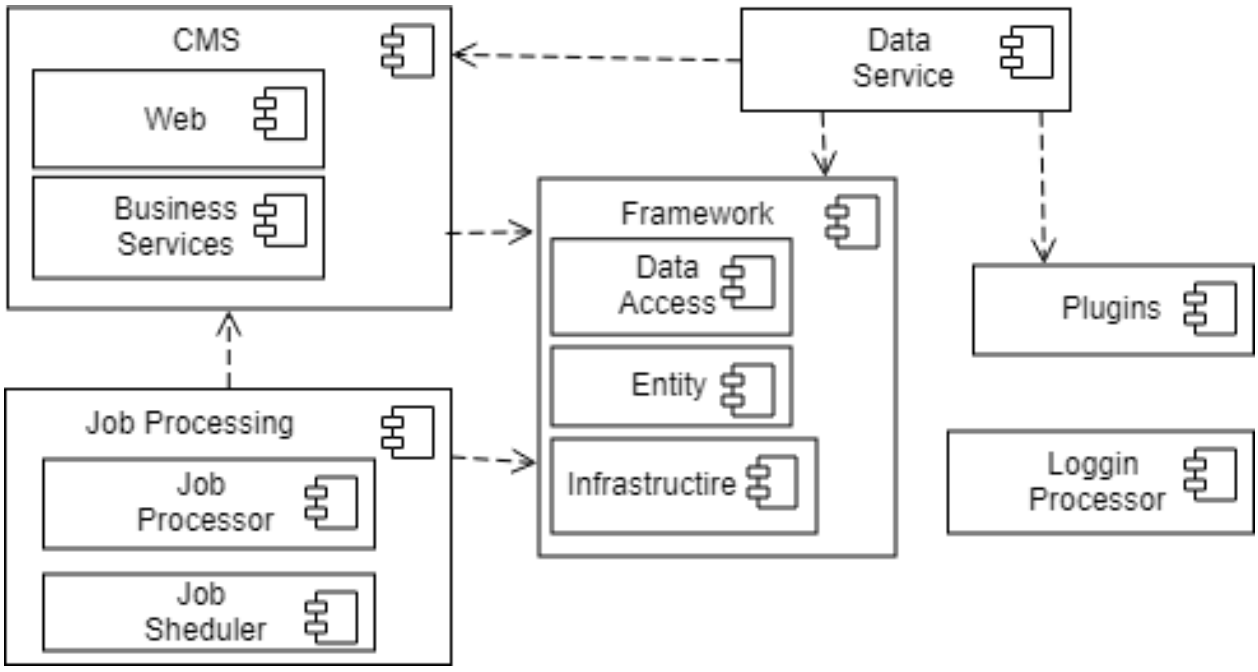

Рисунок 3.9 – Діаграма компонентів

можливістю використання LINQ запитів. Було використано підхід Model First. Відповідно до цього підходу було побудовано модель, на базі якої було згенеровано бази даних та класи сутностей.

В модулі Data Access реалізовано паттерн «Репозиторій» для виконання CRUD операцій. Інтерфейс IRepository<T> реалізовує даний паттерн і є універсальним для будь-якого типу сутностей. При створенні репозиторію забезпечується можливість використання рядка підключення до бази даних, яка

буде отримана під час виконання. Для цього використовується реалізація інтерфейсу public interface IConnectionStringProvider який має один метод GetConnectionString. Реалізовується цей інтерфейс в компоненті, який буде використовувати Framework, і дозволяє використовувати довільний спосіб для визначення рядку підключення.

Метод CreateRepository класу RepositoryFactory<T> призначений для безпосереднього отримання конкретного репозиторію. В конструктор класу передається IConnectionStringProvider[17]. В проекті ConnectionStringProvider виконує перетворення рядку підключення для основної бази даних, який записаний в файлі конфігурації. Цей рядок має вигляд initial catalog=PMI_GMB_{0}, за допомогою регулярних виразів підставляється код маркету.

В модулі інфраструктури реалізовуються адаптери для зв’язку з чергою повідомлень, побудова повідомлення логування, та файли ресурсів застосунку. Клас public class LoggingService, який є реалізацією інтерфейсу ILoggingService, надає можливості для логування. В його конструктор передається клієнт черги і назва джерела логування. Основними його методами є LogException, LogInfo, LogWarning. Завдяки цьому запис логів будь-якого компоненту має уніфіковану структуру. Клієнт черги клас RabbitMQSenderClient має методи JournalLog, Log які надсилають повідомлення на чергу, яке передається на сервіс логування, який безпосередньо записує інформацію в певне сховище. Параметром для цих методів являється об’єкт класу Wrapper, який підтримує серіалізацію і може зберігати вичерпну інформацію про подію яка відбулась:

- string SourceName – назва компоненту, який ініціював створення запису;
- string LogType – рівень логу;
- string Message – повідомлення;
- Exception Exception – інформація про виключення;
- IDictionary<string, string> Dictionary – додаткові данні.

Модуль DataService забезпечує роботу інфраструктури бекенду, таку як реалізація базових сервісів обробки даних, реалізацію IConnectionStringProvider, а також підключення плагінів.

Найважливішими компонентами являються модулі, що реалізують інтерфейси IArticleCategorization, IPreferencesResolver, а IFeedBuilder оскільки для реалізації цих функцій можуть застосовуватися різноманітні алгоритми.

Реалізація IArticleCategorization співвідносить статті та їх атрибути, такі як категорії, теги, ключові слова, географічне положення. В компоненті необхідно реалізувати правила присвоєння атрибутів кожного типу, а також зв'язків між ними. Наприклад, це може бути простий пошук ключових слів і визначення географічного положення. Такі зв’язки можуть формуватися, як на основі тексту статті так і на основі метаданих статті, в залежності від обраної реалізації.

Інтерфейс IPreferencesResolver відповідальний за визначення того, які саме атрибути цікаві користувачу. Реалізацій даного інтерфейсу теж може бути декілька. Користувач може сам визначити коло цікавих йому тем, або може бути проведений аналіз збережених та вподобаних статей, для більш глибокого аналізу може бути використана інформація про прочитання статті і час затрачений на читання. Вибір визначається відповідно до потреб конкретного застосунку і залежить від області застосування, ніші та стратегії просування агрегатора.

IFeedProvider формує стрічку новин на основі користувацьких уподобань і на основі категоризації статей. Також може бути враховано географічне положення, популярні статті, групування статей.

Завдяки модульності цих компонентів і легкості заміни, можна побудувати агрегатор будь-якого масштабу, від найпростішого, в якому використовується ручне налаштування, до системи зі складним комплексом алгоритмів. В проекті реалізовані найпростіші варіанти цих компонентів.

В таблиці 3.3 розглянуто основні класи і методи більш детально.

Таблиця 3.3 – Специфікація основних методів DataService

Клас	Метод	Параметри	Дія
Connection StringProvider	GetConnection String	–	Функція яка виконується при запуску веб-серверу. Завантажує усі дані, що необхідні для планувальника та виконує планування.
Business ServiceBase	HandleException	HandleException Args args	Викликається, коли під час обробки запиту виникає виняткова ситуація. Повертає статус помилки, а також вносить відповідний запис у логи.
Business ServiceBase	OnStartProcessing Request	ProcessRequest Args args	Виконується на початку обробки запиту; проставляє час початку обробки сервером, а також розбирає параметри, і перетворює їх на словник назва-значення.

Продовження Таблиці 3.3

HostToMarket Matcher	Match	string host	Безпосередньо отримує код маркету з доменного ім’я за допомогою регулярних виразів. Код маркету вказується в найвищому рівні ім’я.
Service Operation Parameter Service	ValidateRequired	string parameter, string parameterName	Перевіряє наявність обов’язкового параметру, і генерує виключення, якщо параметр відсутній
Service Operation Parameter Service	ValidateOptional	string parameter, string parameterName	Перевіряє наявність не обов’язкового параметру, повертає параметр або null якщо його немає.
Service Operation Parameter Service	ValidateAndParse RequiredParameter	string parameter, string paramName, Func<string, T> parsingFunction	Виконує валідацію та приведення параметру до певного типу
PluginContext	LoadedPlugins	–	Надає перелік завантажених плагінів підключених для відповідного маркету.

Продовження Таблиці 3.3

Component Provider	GetComponent<T>	Func<IList<T>,T> componentSelector	Повертає компонент відповідного типу або генерує виключення. Використовує для цього PluginContext.
Components Helper	GetDispatcherSelector	string caller	Формує функцію селектор, для отримання компоненту на основі його ім’я.
IArticleCategorization	Categorize	list Articles, list attributes	Визначає категорії, які відносяться до статей
IPreferencesResolver	ResolvePreferences	Customer customer, CustomerPreferences preferences, List ArticleAttitude, List attributes	Формує список атрибутів, які цікаві користувачу і їх пріоритет на основі існуючих налаштувань, та його активності
IFeedBuilder	GetFeed	CustomerPreferences preferences, list attributes = null, int count	Повертає список статей на основі вподобань та запиту користувача.

Одна із найважливіших частин реалізованих DataService є забезпечення роботи плагінів. Для цього був створений інтерфейс IComponent, який відмічає

ті компоненти які можуть бути замінені плагіном. Плагін має реалізовувати інтерфейс специфічним для компоненту.

Виконуючи запит клієнта цей модуль виконує перевірку існування необхідного компоненту в числі плагінів для даного маркета, і в випадку існування такого плагіна, викликається його реалізація, в іншому ж випадку використовується реалізація за замовчуванням. PluginContext виконує пошук плагінів в відповідній папці в проекті. За необхідності алгоритм пошуку можна заміни іншим створивши реалізацію інтерфейсу PluginContext і замінивши його в ін'єкції залежностей. Для можливості роботи з плагіном він має реалізовувати інтерфейс IComponent, а також всі відкриті методи, специфічні для компонента.

CMS BusinessLogic реалізує бізнес логіку застосунку. Класи-сервіси дають можливість безпечно і коректно працювати з сутностями – створювати, оновлювати і отримувати їх. Детальніше класи і їх призначення розглянуто в таблиці 3.4.

Таблиця 3.4 – Функціональне призначення класів CMS BusinessLogic

Назва класу	Функціональне призначення
NinjectWebCommon	Клас реєструє залежності для IoC
ArticleService	Клас відповідає за створення та редагування статі, за отримання статей за певним фільтром, присвоєння статті певного атрибуту, отримання статистики про статтю – кількість вподобаних і збережених, отримання медіа файлів пов’язаних зі статтею.
ArticleAttributeService	Створює та редагує типи атрибутів, такі як автор, джерело, тег, категорія, локація, ін.
SoruceService	Керування джерелами, та налаштуваннями конвертера, які перетворюють об’єкт отриманий від джерела до доменного об’єкту застосунку.

Продовження Таблиці 3.4

SorucePullerService	Створення і налаштування задач, які ініціюють отримання статей з інформаційних джерел. Задання періодичності запуску, введення ключів доступу, а також запуск і зупинка виконання задачі.
RegionService	Управління типами регіонів (місто, область, штат) та регіонами для країн
CountryService	Управління країнами, для прив'язки статей до географічної точки.
IConsumerService	Створення нового користувач, отримання та оновлення його профілю, налаштування списку вподобань, зміна паролю.
IDashboardStatisticService	Отримання загальної статистики застосунку
ILocationService	Визначення географічної локації користувача на основі його запиту.
IMediaService	Завантаження і отримання медіа файлів.
FeedAnonimusService	Формування загальної стрічки новин, з врахуванням місця положення.
FeedService	Формування стрічки новин для авторизованого користувача на основі його вподобань, місця знаходження, а також збережених і вподобаних статей.

CMS Web представляє собою веб-застосунок, побудований на основі архітектурного паттерну MVC. В застосунку не реалізовано ніякої бізнес-логіки, таким чином його можна замінити іншою реалізацією за необхідності.

3.1 Висновки по розділу

В даному розділі було виконане моделювання та аналіз програмного забезпечення а також побудовано схему процесу роботи системи. Визначено архітектуру програмного забезпечення, яка буде використовуватися при розробці продукту, а саме: вибрана мова програмування C#, Середа розробки Visual Studio Ultimate, в якості сервера бази даних використано SQL Server, для доступу до даних використано Entity Framework.

В рамках даного розділу була розроблена та описана структура бази даних. Структурну схему бази даних можна побачити в додатку А лист 2.

Також в рамках даного розділу була розроблена діаграма компонентів, яку можна побачити в додатку А лист 3, та описані всі функції основних класів та їх параметри.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 План тестування

В плані тестування було ідентифіковано ресурси та необхідні попередні налаштування для тестування програмного продукту, визначені цілі тестування та методики його проведення. Також було визначено які результати та проміжні продукти мають бути протестовані. План тестування був розроблений на основі функціональних вимог. Він описаний в таблиці 4.1.

Таблиця 4.1 – План тестування

№	Розділ	Елементи
1	Вступ	У цій таблиці описується тестування программи «Global News Aggregator Backend»
2	Функціонал, що підлягає тестуванню	– створення клієнта; – реєстрація користувача; – додавання нового інформаційного джерела; – створення задачі планувальника; – зміна налаштувань планувальника; – отримання стрічки новин.
3	Тестові елементи	– реєстрація нового клієнта; – реєстрація нового користувача; – автентифікація користувача; – додавання нового джерела; – перегляд існуючих джерел; – створення задачі планувальника; – перегляд отриманих статей; – перегляд стрічки новин.

Продовження Таблиці 4.1

4	Тестові підходи	Тестування проводить за допомогою підходу black box.
5	Критерій успішно/не успішно	– всі вказані вимоги мають покриватись позитивними тестами, і хоча б одним негативним тестом – всі блокуючі дефекти мають бути виправлені; – 5 і 3 дефектів середнього низького і середнього рівня в продукті.
6	Критерій призупинення/відновлення	– застосунок не запускається / запускається.
7	Вимоги до середи	Необхідна машина (ноутбук, комп'ютер, сервер), на якому має бути встановлена Windows. Наявність клавіатури
8	Вимоги до навичок персоналу	Персона моє бути впевненим користувачем, а також знайомий з клієнтом для http тестування сайтів
9	Задачі тестування	– пошук дефектів, що можуть викликати відмову застосунку; – формування сценаріїв для перевірки виконання функціоналу програми; – підготовка звіту про роботу.

Продовження Таблиці 4.1

10	Супровід тестування	– тестовий план; – тестові сценарії; – специфікація формату запитів; – журнал помилок та виконання; – звіти про дефекти та дії для коригування.
11	Розклад тестування	Для виконання кожного тесту виділяється одна година
12	Відповідальні	Автор плану тестування
13	Підтвердження	Для переходу на наступний етап, необхідно рішення відповідального за тестовий план
14	Ризики	При зменшенні часу на тестування, перевірити додаток в повному обсязі не вдасться. Буде виконано або часткову перевірку деяких пунктів, або виключення пунктів з плану тестування, що знизить загальну якість тестування застосунку. Критичні дефекти можуть бути не знайдені і невиправлені.

4.2 Процес тестування

Для контролю якості програмного забезпечення на етапі розробки зазвичай використовують модульне тестування. Оптимальним рішенням для мови програмування C# є NUnit. Він дозволяє легко створювати модульні тести, які дозволяють перевірити роботу виокремленої частини коду, тобто для кожного методу класу. Модульні тести можуть бути написані як до написання коду, такий підхід називають Test Driven Development, або ж після цього, в залежності від особливостей компанії і вподобань розробника. В проекті тести

були написані після розробки самої системи, для задач безпосередньо пов’язаних з функціональними вимогами системи.

Для роботи з NUnit необхідно підключити адаптер фреймворку, наприклад через менеджер пакетів Nuget. Після чого створюється проект для тестів, в якому можна створювати модульні тести [18]. Одним з параметрів який характеризує модульне тестування є code coverage. Цей параметр показує процент коду покритого тестами. Для його визначення в проекті використовувався сервіс Coveralls, це сторонній веб-сервіс, який аналізує код і модульні тести написані для нього. Оскільки проект учбовий, повного покриття не вимагалось.

Для прикладу розглянемо тестування репозиторію наведеного в таблиці 4.2. В процесі тестування перевіряються відкриті методи репозиторію, кожному тесту відповідає тестовий метод. Поетапно описуються дії необхідні для виконання тесту, очікуваний та отриманий результат, а також висновок.

Таблиця 4.2 – Процес тестування

№	Процес	Назва	Етапи	Очікуваний результат	Реальний результат	Висно-вок
1	Додавання об’єкту	AddEntity Test	– створити новий об’єкт	В сховищі даних з’явиться новий запис	В сховищі даних з’явився новий запис	Успіх
2	Додавання декількох об’єктів	AddEntities Test	– створити перелік об’єктів	В сховищі даних з’являться нові записи, що відповідають створеним об’єктам	В сховищі даних з’явилися нові записи, що відпові-дають створеним об’єктам	Успіх

Продовження Таблиці 4.2

3	Видалення об’єкту	Remove EntitiyTest	– отримання існуючого об’єкту з сховища даних	Зі сховища буде видалено об’єкт	Об’єкт видалено	Успіх
4	Видалення декількох об’єктів	Remove EntitiesTest	– отримання існуючих об’єктів з сховища даних	Зі сховища буде видалено об’єкти	Об’єкти видалено	Успіх
5	Видалення всіх об’єктів	Remove AllTest	–	Зі сховища буде видалено всі об’єкти	Всі об’єкти видалено	Успіх
6	Видалення об’єктів, які відповідають виразу	RemoveAll Expression Test	– формування логічного виразу, такого щоб об’єкти відповідні йому існували в базі даних	Зі сховища буде видалено всі об’єкти, які відповідають виразу	Зі сховища видалено всі об’єкти, які відповідають виразу	Успіх

4.3 Висновки до розділу

В даному розділі було проведено тестування веб-сервісу платформи бекенд новинного агрегатору, а також його веб-сайт для адміністрування що має графічний інтерфейс. Було розроблено план тестування, а також описано процес тестування. Результатом виконання тестів стало заключення про задовільну якість програмного продукту.

5 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Програмне забезпечення складається з таких частин:

– бекенд сервер з веб-сайтом для адміністрування;

– планувальник задач;

– сервіс логування.

Для повноцінного функціонування платформи необхідно запустити всі три.

Для розгортання необхідною є наявність .Net Core Framework, СУБД SQL Server, брокер повідомлень RabbitMQ. При виконанні цих вимог можна проводити розгортання компонентів.

Для розгортання необхідно:

а) підготувати SQL Server до роботи:

1) додати користувача gmb

2) додати пароль користувача в конфігурацію застосунку

3) створити базу даних GMB_Admintistration та GMB_EN та надати доступ користувачу gmb

б) підготувати RabbitMQ до роботи:

1) створити користувача-адміністратора gmb;

2) додати пароль користувача в конфігурацію застосунку;

3) створити чергу GMB.

в) Розпакувати файли програми в окрему папку та налаштувати IIS.

5.2 Робота з програмним забезпеченням

Для адміністрування платформи використовується веб-сайт, доступ до маркету виконується з використанням його коду в найвищому рівні доменного імені. За допомогою цього сайту виконується конфігурація і управління контентом, робота з джерелами.

Клієнтські додатки взаємодіють з платформою через інтерфейс HTTP-запитів. Для цього необхідно спочатку зареєструвати додаток за допомогою адміністративного сайту, і отримати ключ доступу.

Всі запити мають відповідати стандартній структурі. Її специфікація включає обов'язкові та не обов'язкові заголовки і параметри запиту.

Специфікація користувацького запиту:

– t – час відправки запиту;

– at – токен доступу клієнтського застосунку;

– vt2 – контрольна сума;

– a – токен доступу авторизованого користувача;

– l – місцеположення користувача, якщо не передано, обирається значення за замовчуванням;

– QueryParameters – параметри запиту, не має обов'язкових для всіх запитів параметрів.

5.3 Висновки до розділу

В даному розділі було описно необхідні умови для розгортання платформи, а також описано дії, які необхідно для цього виконати. На цільовій машині мають бути встановлені.Net Core Framework, СУБД SQL Server, брокер повідомлень RabbitMQ. Також була описана робота з платформою та формат повідомлень, які має надсилати застосунок-клієнт.

ВИСНОВКИ

В даному проекті була розроблена платформа для створення серверної частини новинних агрегаторів. Платформа складається з API та веб-сайту для адміністрування. Особливістю платформи є можливість обслуговування багатьох застосунків клієнтів, а також можливість розширення функціоналу за допомогою плагінів. Це дозволяє створювати довільний бекенд сервіс на базі розробленої платформи.

В процесі роботи над проектом був проведений аналіз предметної області. Також виконаний аналіз існуючих рішень, їх переваг та недоліків. На основі отриманих даних було сформовано функціональні та нефункціональні вимоги до платформи, а також розроблено модель варіантів використання.

За допомогою моделі варіантів використання було змодельовано програмний продукт. Основні бізнес-процеси були описані в BMPM-діаграмах. Було виділено основні компоненти системи, розроблено діаграму компонентів і класів. На основі проведеного аналізу були підібрані оптимальний технологічних стек.

Для контроля якості програмного засобу було розроблено план тестування, на основі якого проведено тестування. Результати перевірки показали задовільну якість програмного засобу. Також для забезпечення якості на етапі розробки були написані модульні тести для найкритичніших частин програмного засобу.

Було розроблено інструкцію по розгортанню платформи а також інструкцію користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Usage of content management systems [Електронний ресурс] // Режим доступу: https://w3techs.com/technologies/overview/content_management/all

2. How to Build a News Aggregator with WordPress[Електронний ресурс] // Режим доступу: https://www.mobiloud.com/blog/wordpress-rss-aggregator/

3. Content Recommendations [Електронний ресурс] // Режим доступу: https://www.native.ai/content-recommendations

4. About NewsNow [Електронний ресурс] // Режим доступу: https://www.newsnow.co.uk/about/

5. The seven steps to a successful aggregation strategy for your news organization [Електронний ресурс] // Режим доступу: https://www.poynter.org/reporting-editing/2011/the-seven-steps-to-a-successful-aggregation-strategy-for-your-news-organization/

6. The law around aggregating news online[Електронний ресурс] // Режим доступу: https://www.niemanlab.org/2010/09/whats-the-law-around-aggregating-news-online-a-harvard-law-report-on-the-risks-and-the-best-practices/

7. Карл И. Разработка требований к программному обеспечению. 3-е издание, дополненное / БХВ-Петербург, 2018. – 736 с. – ISBN 978-5-9909805-3-2

8. Ларман К. Применение UML 2.0 и шаблонов проектирования / Диалектика-Вильямс, 2018. - 736 сю – ISBN 978-5-845-91185-8

9. About .NET Core [Електронний ресурс] // Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/core/about

10. Впровадження залежностей [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Впровадження_залежностей

11. Seemann M. Dependency Injection in .NET – Manning Publications, 2011 584 с. – ISBN 9781935182504

12. NHibernate Reference Documentation [Електронний ресурс] // Режим доступу: https://nhibernate.info/doc/nhibernate-reference/index.html

13. Entity Framework 6 [Електронний ресурс] // Режим доступу: https://docs.microsoft.com/en-us/ef/ef6/

14. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения / Питер, 2018ю – 352 сю - ISBN 978-5-4461-0772-8

15. Understanding Onion Architecture [Електронний ресурс] // Режим доступу: https://www.codeguru.com/csharp/csharp/cs_misc/designtechniques/understanding-onion-architecture.html

16. Модель-вид-контролер [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Модель-вид-контролер

17. Гамма Э. Приемы объектно-ориентированного проектирования Паттерны проектирования / Питер, 2019. – 368 с. – ISBN 978-5-4461-1213-5

18. NUnit Documentation [Електронний ресурс] // Режим доступу: https://github.com/nunit/docs/wiki/NUnit-Documentation

19. Принцип MVC в веб программировании [Електронний ресурс] // Режим доступу: http://folkprog.net/printsip-mvc-u-web-programmirovanii/

20. Ріккарді Гр. Системи баз даних : монографія / Ріккарді Гр. – Київ : Вільямс, 2001. – 686 с. – ISBN 5-8459-0208.

21. Інформаційни сайт із структурними схемами активності [Електронний ресурс] // Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

22. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# /Питер, 2018. – 896 с. - ISBN: 978-5-496-00433-6

23. Троелсен Э. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е издание / Диалектика, 2019ю – 1328 с. – ISBN 978-5-6040723-1-8

24. The seven steps to a successful aggregation strategy for your news organization [Електронний ресурс] // Режим доступу: https://www.poynter.org/reporting-editing/2011/the-seven-steps-to-a-successful-aggregation-strategy-for-your-news-organization/

25. The law around aggregating news online[Електронний ресурс] // Режим доступу: https://www.niemanlab.org/2010/09/whats-the-law-around-aggregating-news-online-a-harvard-law-report-on-the-risks-and-the-best-practices/

26. The Ultimate Guide To Content Aggregators (The Good, The Bad and The Ugly) [Електронний ресурс] // Режим доступу: https://www.jeffbullas.com/content-aggregators/

27. Alexa, M. Text Analysis Software: Commonalities, Differences and Limitations: The Results of a Review / Springer Netherlands, 2000. – 321 с. - ISSN: 0033-5177

28. Эванс Э. Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем / Вильямс, 2010. – 448 с. - ISBN 978-5-8459-1942-7

ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ

ЛИСТ 1. BPMN-ДІАГРАМА ОБРОБКИ КЛІЄНТСЬКОГО ЗАПИТУ

ЛИСТ 2. ER-ДІАГРАМА БАЗИ ДАНИХ

ЛИСТ 3. ДІАГРАМА ПОСЛІДОВНОСІ ОБРОБКИ КОРИСТУВАЦЬКОГО ЗАПИТУ

ЛИСТ 4. ДІАГРАМА КОМПОНЕНТІВ