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

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

"___"_____2022 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-застосунок "Блог"

Виконав студент IV курсу, групи ІТ-83
(шифр групи)

Вікторов Нікіта Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Крамар Ю.М _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Ткач М. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ –2022

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

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютеризованих систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2022 р.

ЗАВДАННЯ
на дипломний проект студенту

Вікторову Нікіті Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту Веб-застосунок "Блог"
керівник проекту Крамар Юлія Михайлівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»квітня 2022 р. №_____

2. Термін подання студентом проекту « 19 » червня 2022 року

3. Вихідні дані до проекту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна класів програмного забезпечення

6. Консультанти розділів проекту

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

7. Дата видачі завдання «10» березня 2022 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення рекомендованої літератури	10.03.2022	
2	Аналіз існуючих методів розв'язання задачі	06.03.2022	
3	Постановка та формалізація задачі	30.03.2022	
4	Розробка інформаційного забезпечення	01.04.2022	
5	Алгоритмізація задачі	13.04.2022	
6	Обґрунтування вибору використаних технічних засобів	25.04.2022	
7	Розробка програмного забезпечення	05.05.2022	
8	Налагодження програми	27.05.2022	
9	Виконання графічних документів	30.05.2022	
10	Оформлення пояснювальної записки	01.06.2022	
11	Подання ДП на попередній захист	06.06.2022	
12	Подання ДП рецензенту	12.06.2022	
13	Подання ДП на основний захист	19.06.2022	

Студент _____ Нікіта ВІКТОРОВ
(підпис) (ініціали, прізвище)

Керівник _____ Юлія КРАМАР
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 39 таблиць, 14 рисунків та 9 джерел – загалом 69 сторінки.

Дипломний проект присвячений написання веб-застосунку “Блог”

Мета надати можливість користувачам дізнаватися про щось нове за короткій проміжок часу та меншу кількість дій та розширення можливостей блогу

Об'єкт дослідження: різні блоги

Предмет дослідження: :методи формування блогу …

У розділі «аналіз вимог до програмного забезпечення» зроблено опис та аналіз предметної області, аналіз успішних ІТ-проектів, аналіз вимог до програмного забезпечення та постановка задачі.

Розділ «моделювання та конструювання програмного забезпечення» присвячений моделюванню та аналізу програмного забезпечення, архітектурі програмного забезпечення, структурі даних та ресурсів програми та аналізу безпеки даних.

Наступний розділ «аналіз якості та тестування програмного забезпечення» описує процес тестування та аналіз якості програмного забезпечення.

Останній розділ «впровадження та супроводження програмного забезпечення» розкриває такіпитання, як розгортання програмного забезпечення, робота з програмним забезпеченням.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, ВЕБ-ЗАСТОСУНОК

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 39 tables, 14 figures and 9 sources – in total 69 pages.

The purpose of the diploma project is providing an opportunity to regularly increase the amount of useful content on the site, which means - to increase the credibility of the resource and strengthen search engine promotion. Each blog article is an additional keyword entry.

Object of research: various blogs Subject of research:: methods of blog formation… In the section "analysis of software requirements" a description and analysis of the subject area, analysis of successful IT projects, analysis of software requirements and problem statement. The section "Software modeling and design" is devoted to software modeling and analysis, software architecture, data structure and program resources and data security analysis.

The next section, Software Quality Analysis and Testing, describes the software testing and analysis process. The last section "software implementation and maintenance" reveals such issues as software deployment, work with software.

KEYWORDS: DATABASE, WEB APPLICATION

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІТ-8305.045440.02.83	Технічне завдання	10	
3	А4	КПІ.ІТ-8305.045440.02.83	Пояснювальна записка	69	
4	А4	КПІ.ІТ-8305.045440.02.83	Опис програми	7	
5	А4	КПІ.ІТ-8305.045440.04.83	Програма та методика тестування	5	
6	А4	КПІ.ІТ-8305.045440.04.83	Керівництво користувача	5	
7	А4	КПІ.ІТ-8305.045440.04.83	Схема структурна варіантів використань	1	
8	А4	КПІ.ІТ-8305.045440.01.83 л	Схема база даних	1	
9	А4	КПІ.ІТ-8305.045440.01.83	Схема структурна класів програмного забезпечення	1	
10	А4		Звіт Unicheck	1	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-8305.045440.02.83		
Розроб.		Вікторов Н.С.			Відомість дипломного проекту	Літ.	Аркуш Аркушів
Перевір.		Крамар Ю.М.					1
Н.контр.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-83	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ВЕБ-ЗАСТОСУНОК “БЛОГ”

Технічне завдання

КПІ.ІТ-8305.045440.02.83

“ПОГОДЖЕНО”
Керівник проекту:
_____ Юлія Крамар

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Нікіта ВІКТОРОВ

Київ – 2022

ЗМІСТ

– НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосунок “Блог”

Галузь застосування:

Наведене технічне завдання поширюється на розробку Веб-застосунку “Blog”, котра використовується для регулярного нарощування обсягу корисного контенту сайту, а значить – підвищити авторитет ресурсу та посилити пошукове просування. Кожна стаття блогу – це додаткові входження ключових слів. та призначена для рекламування та для створення інфоповодів у соц.мережах

– ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунок “Блог” є завдання на дипломне проектування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

– ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації та полегшенню використання блогу на сьогоднішній день

Метою розробки є зручного програмного забезпечення для поширення інформації та спілкування

– ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

– Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

Для користувача:

– Редагування статті;
– Створення статті;
– Пошук за тегом
– Редагування коментаря
– Створення коментаря
– Вхід в систему

Для адміністратора системи (якщо він передбачений):

– Редагуання усіх статей;
– Створення статей;
– Пошук за тегом
– Редагування усіх коментарів
– Створення коментарів
– Вхід в систему
– Надання ролей

Вимоги до надійності

– Передбачити контроль введення інформації.
– Передбачити захист від некоректних дій користувача.

Умови експлуатації

Спілкування та дізнаватись нову інформацію в мережі

– Вимоги до складу і параметрів технічних засобів

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

Мінімальна конфігурація технічних засобів:

Тип процесору ..... Pentium.

Об‘єм ОЗП..... 32 Мб.

– Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних си-стем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

– Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

– Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

– Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

– ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

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

– Програмне забезпечення повинно мати довідникову систему.

– У склад супроводжувальної документації повинні входити наступні документи:

Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

Технічне завдання.

Керівництво користувача.

Опис програми

Програма та методика тестування

– Графічна частина повинна бути виконана не менше ніж на 3 аркушах формату А3, котрі включаються у якості додатків до пояснювальної записки:

Схема структурна варіантів використання

Схема структурна класів програмного забезпечення

– СТАДІЇ І ЕТАПИ РОЗРОБКИ

<Брати з листа завдання>

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.04	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04	Технічна документація

– ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

Пояснювальна записка
до дипломного проекту

на тему: Веб-застосунок “Блог”

КПІ.ІТ-8305.045440.02.83

Київ – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

JWT	JSON Web Token
API	Application programming interface, прикладний програмний Інтерфейс
IT	Інформаційні технології
БД	База даних
JSON	JavaScript Object Notation

Вступ

Статус блогів в екосистемі контенту, бездумно, низько, переглянутою в центрі, описані потоки відео, подкасти та короткий контент у Instagram та TikTok.

Першою можливістю створювати контент на зорі Інтернету було ведення блогу, тому що це було легко – це письмове слово.

На той час створення відео контенту було ще складним завданням. Вам потрібне обладнання, навички редагування та програмне забезпечення. Потім виникла проблема з повільним інтернет-з'єднанням, через яке завантаження було практично неможливим. З появою смартфонів створення відео контенту стало доступним кожному. Ця розробка призвела до вибухового розвитку відео, яке задовольнило вкрай незадоволені потреби споживачів.

Сьогодні з тієї ж причини спостерігається різке зростання під кастингу. З'являються більш доступні технології та кращі канали розповсюдження, щоб задовольнити потреби споживачів, яким подобається аудіо контент.

Коли всі працюють над відео та подкастами, ведення блогу може здатися старомодним.

Насправді ведення блогу ніколи не було таким яскравим. Створено рекордну кількість постів. А для передачі деяких складних повідомлень, особливо у сфері високих технологій, часто потрібне письмове спілкування.

Для багатьох компаній досі немає кращого способу наростити короткострокові SEO-м'язи, ніж сильний та корисний блог. LinkedIn показало, що письмовий контент, безумовно, є переважною формою контенту на цій платформі.

Відео та подкасти ростуть набагато швидше, тому що довгий час вони були недостатньо затребуваною нішою, але блоги залишаються надійним основним продуктом контенту.

Мета блогу – створити привабливіший веб-сайт для вашого бренду. Поряд із використанням вашого веб-сайту для навчання користувачів вашим продуктам та послугам, ви також можете включити розділ блогу, щоб знайомити споживачів з темами, що стосуються вашої області. Поряд з цим блог може допомогти вам отримати більше трафіку на сайт з пошукових систем. Включивши релевантні ключові слова, більше людей зможуть знайти ваші повідомлення у блозі через пошук і, можливо, знайти решту вашого веб-сайту.

Є багато причин, де блог допоможе и требу його використовувати на сьогоднішній день.

Написання статей на конкретні теми, про які хоче дізнатись ваша цільова аудиторія, може показати їм, що ви є лідером у вашій галузі. Наприклад, якщо ви продаєте кухонне начиння, ви можете створити блог, в якому будуть представлені різні кулінарні прийоми та рецепти. Маючи на своїй веб-сторінці добре написаний та опрацьований контент, ви показуєте користувачам, що знаєте, про що кажете. Окрім звернення до вас за рекомендаціями, користувачі можуть відчути, що ваш досвід впливає на якість ваших товарів та послуг.

Якщо ви зацікавлені у співпраці з іншими брендом, блог – чудовий спосіб розпочати роботу. Ви можете попросити когось із іншої компанії написати гостьовий пост у блозі. Інша компанія також може захотіти уявити вас у своєму блозі. Посилання на авторитетні зовнішні веб-сайти і те, що вони роблять те саме, можуть збільшити кількість людей, які відвідують обидва ваші веб-сайти. Ви навіть можете попросити галузевого експерта або відому людину написати гостьовий пост у блозі, щоб зацікавити ваших читачів.

Після того, як ви опублікуєте повідомлення в блогу, воно може залишитися в Інтернеті назавжди. Створюючи вічнозелений контент, користувачі можуть постійно перенаправлятися на ваш сайт за пошуковими

запитами. Реалізуючи цю довгострокову маркетингову стратегію, ви зможете зберегти потік людей, які відвідують ваш сайт на довгі роки.

1.ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Людині хочеться дізнаватися про щось цікаве та витрачати на це мало часу. Інтернет дає нам ці можливості, щоб бути в курсі усіх новин сьогодення. Але також людям хочеться ділитися новинами та спілкуватися без скачування програм, так як може бути немає можливості завантажити програму для спілкування(відсутність пам’яті, слабкий комп’ютер). Для цього були придумані соціальні мережі, такі як блоги.

1.1.1 Опис процесу діяльності

Блог — це веб-сторінка, яку ви регулярно оновлюєте. Часто бренди ведуть блоги прямо на своїх веб-сайтах, що спрощує користувачам пошук та взаємодію з вашими повідомленнями. У своєму блозі ви можете публікувати довгі статті на теми, які ваша цільова аудиторія може захотіти прочитати або дізнатися. Ви також можете включити посилання на внутрішні або зовнішні веб-сторінки, щоб підвищити зручність читання ваших користувачів. Додавання зображень або відео також може зробити читання вашого блогу більш цікавим.

1.1.2 Опис функціональної моделі

На рисунку 1.1 наведена модель роботи системи побудови маршруту подорожі:

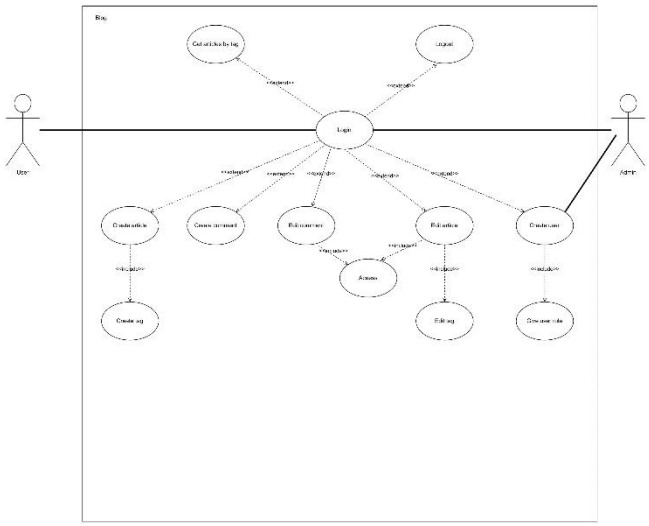


Рис 1.1 – Схема варіантів використання

У функціональній моделі присутні два актори:

Користувач – може авторизуватись, вигадувати теги до статей, редагувати статі та коментарі, також створити коментар та статтю. Користувач може редагувати тільки свої коментарі та статті та виходити с блогу.

Адміністратор – може робити теж саме, що і користувач та створювати користувача та надавати роль користувачеві. Це було створено для того, щоб у адміністратора була можливість, хто може управляти блогом повноцінно, а хто не може.

1.1.3. Розроблення функціональних вимог

Таблиця 3.1 - Варіант використання UC001

Назва	Реєстрація користувача
Опис	Користувач може отримати акаунт завдяки надання логіну та паролю
Учасники	Користувач
Передумови	Неавторизований користувач
Післяумови	У користувача з’являється аккаунт
Основний сценарій	1) Користувач відкриває застосунок. 2) У Користувача відчиняється сторінка для надання даних,щоб пройшла реєстрації. 3) Користувач авторизується та автоматично переходить на головну сторінку.

Таблиця 3.2 - Варіант використання UC002

Назва	Отримання статей за тегом
Опис	Користувач може отримати статті завдяки обраному тегу
Учасники	Користувач
Передумови	Обраний тег
Післяумови	Статті в яких є обраний тег

Продовження таблиці 3.2

Основний сценарій	1) Користувач обирає тег. 2) У Користувача відчиняється сторінка, яка показує статті, в яких існує обраний тег пройшла.

Таблиця 3.3 - Варіант використання UC003

Назва	Створення статті
Опис	Користувач може створити статтю
Учасники	Користувач
Передумови	Авторизований користувач
Післяумови	Користувач створив статтю
Основний сценарій	1) Користувач відкриває застосунок. 2) У Користувача відчиняється сторінка з усіма статтями. 3) Користувач створює статтю, потім відкривається сторінка для створення статті.

Таблиця 3.4 - Варіант використання UC004

Назва	Редагування статті
Опис	Користувач може відредагувати статтю
Учасники	Користувач
Передумови	Авторизований користувач
Післяумови	Користувач відредагував статтю
Основний сценарій	1) Користувач відкриває застосунок. 2) У Користувача відчиняється сторінка з усіма статтями. 3) Користувач обирає статтю, до яких він має доступ, потім відкривається сторінка для редагування статті.

Таблиця 3.5 - Варіант використання UC005

Назва	Вихід з системи
Опис	Вихід з системи до наступного логування
Учасники	Користувач
Передумови	Авторизований користувач
Післяумови	Вихід з системи
Основний сценарій	1) Користувач відкриває застосунок. 2) У Користувача відчиняється сторінка з усіма статтями – головна сторінка. 3) Користувач обирає вихід з системи, та виходить до наступної спроби увійти в систему

Таблиця 3.6 - Варіант використання UC004

Назва	Створення користувача та надання ролі
Опис	Додавання нового адміністратора
Учасники	Користувач
Передумови	Адміністратор
Післяумови	Новий адміністратор
Основний сценарій	1) Користувач відкриває застосунок. 2) Користувач повинен бути адміністратором. 3) Користувач створює адміністратора, в якого є права, як в нашого користувача

1.2 Огляд наявних аналогів

1. ProBlogger - Даррен Роуз, також відомий як ProBlogger, надихнув усіх блогерів з 2004 року, коли він запустив ProBlogger. Його поради щодо опису блогу дуже практичні і завжди сповнені реального досвіду життя.

2. Smart Passive Income - Історія Пета Флінна про те, як він був викинутий з роботи після економічної депресії 2008 року і став одним із

найвідоміших блогерів, — це те, з чого складаються мрії. Сьогодні Smart Passive Income надає не лише поради щодо ведення блогу, а й кілька книг, курсів та надихаючий подкаст, на який повинен налаштуватися кожен блогер.

3. Ryan Robinson - Райан Робінсон стверджує, що не дає порад BS щодо ведення блогів, і я не можу погодитися з цим. Хоча я нещодавно відкрив його блог, мене дуже вразили докладні поради щодо блогів, які Райан регулярно публікує у своєму блозі. Він також дуже затребуваний консультант з маркетингу, який працював з деякими великими брендами, такими як LinkedIn, Google, Adobe тощо.

4. Backlinko - Браян Дін з Backlinko в даний час вважається провідним експертом із SEO. Його посібники з SEO дуже докладні і сповнені оригінальних тематичних досліджень та фактів. Брайан ділиться деякими з найкращих стратегій SEO, які допоможуть вам зайняти високі позиції та залучити трафік на ваш блог. Про його популярність можна судити за тим фактом, що кожен пост, який Браян публікує на Backlinko, отримує сотні коментарів та репостів лише за кілька годин після публікації.

5. Ahrefs - є одним з найкращих інструментів SEO, доступних прямо зараз. Крім цього, він також публікує видатні поради та тематичні дослідження у своєму блозі з SEO, які не можна пропустити, якщо ви хочете покращити свої знання в галузі SEO. Я також раджу вам перевірити їх канал на YouTube та безкоштовні курси SEO, які безкоштовно надають масу інформації з усіх аспектів SEO.

6. HubSpot - великий постачальник послуг з управління відносинами з клієнтами (CRM), який має понад 76 000 клієнтів. Його блог охоплює широкий спектр тем, таких як утримання клієнтів, маркетинг електронною поштою, маркетинг у соціальних мережах, маркетинг у пошукових системах, пошук продажів і т. д. Якщо ви шукаєте будь-який термін, пов'язаний з маркетингом, ви незмінно знайдете сторінку Hubspot в топ-10 результатів. Не дивно, що кожен місяць блог HubSpot приваблює понад 7 мільйонів читачів!

7. HackerNoon - це популярна платформа для технічних блогерів, які хочуть публікувати що-небудь пов'язане з технологіями. Щомісячна читацька аудиторія становить 4 мільйони людей, що дуже суттєво. Він має унікальний дизайн у стилі Minecraft. Якщо вам подобаються технології, вам варто подумати про те, щоб проводити дні за хакерством в цьому блозі.

8. Gizmodo - це універсальний технічний блог, який висвітлює не лише хардкорні технічні історії, але й ігри, гаджети та технології, що використовуються у нашому повсякденному житті. Він також включає космічні технології та випадкові історії про зомбі. Якщо ви любите отримувати задоволення від технологічних оновлень, слід відразу ж відправитися в блог Gizmodo.

9. Nerd Fitness - це блог про фітнес і здоровий спосіб життя, розпочатий у 2009 році Стівом Камбом і користується величезною популярністю серед ботаніків, які прагнуть позбутися зайвих кілограмів. Це цікавий спосіб прищепити своїм читачам та платним передплатникам звички здорового способу життя, кількість яких обчислюється тисячами. Навряд чи знайдеться людина, яка шукає поради з фітнесу, яка не натрапила на цей популярний блог.

10. Live strong - це блог про здоровий спосіб життя, що пропонує ряд порад про здоровий спосіб життя, йогу, поради з фітнесу, вправи, яких слід уникати, та багато іншого. Він також ділиться 20-хвилинними порадами з тренувань для чоловіків та жінок, які допоможуть їм прийти у форму та вести здоровий спосіб життя.

11. Fit men cook – це блог, який надихає людей, які хочуть їсти хорошу їжу, залишаючись у формі. У цьому блозі Кевін розповідає про свій шлях від надмірної ваги до кухаря-мачо завдяки правильній дієті та заняттям у тренажерному залі.

12. Ape to Gentleman - це блог про чоловічий спосіб життя із забавною назвою. Він дає поради щодо догляду, щоб перетворити мавпу на вас у ідеального джентльмена. Що і як носити, секрети зачісок, дизайнерський

годинник, туфлі, які притягують погляди. Якщо це те, що вас чарує, то не пропустить цей блог про стиль життя.

При дослідженні наявних аналогів, можна дійти висновку, що більшість базується на ідеї довести до користувачів свої думки. Це допомагає ділитися досвідом та навіть допомагає людям направлятися у вірному напрямку.

1.3 Постанова задачі

1.3.1 Призначення розробки

Призначення розробки є блог , який приближений до кращих блогу світу з зрозумілим функціоналом та зручним інтерфейсом.

1.3.2 Цілі та задачі розробки

Ціль розробки – створення блогу, якого немає в українських аналогів, щоб це був перший блог нового формату.

Для реалізації поставленої цілі, необхідно:

1. Побудувати базу даних, яка міститиме інформацію, необхідну для отримання інформації щодо цільової інформації користувача.
2. Створити веб-інтерфейс, для зручного користування для користувача.

1.4. Аналіз вимог до програмного забезпечення та специфікацію вимог.

Таблиця 1.1 – Функціональні вимоги та їх пріоритети

Варіант використання	**Функціональна вимога**	**Пріоритет**
Реєстрування	1 Програма має надавати можливість зареєструватися користувачу в системі.	Високий
Авторизуватись	2 Програма має надавати Можливість авторизуватися користувачу: 2.1 Програмно повинно бути реалізовано можливість авторизуватись для звичайного користувача(без прав адміністратора) 2.2 Програма має надавати можливість авторизуватися адміністратору	Високий

Таблиця 1.1 – Функціональні вимоги та їх пріоритети

Додати статтю	3 Програма має надавати Можливість авторизованому користувачу додавати статтю	Високий
Переглянути статті	4 Програма має надавати можливість переглянутивсі статті незалежно від авторизації	Високий
Додати нову статтю	5 Програма має надавати можливість додавати новустаттю 5.1 Програма дозволяє користувачу увійти в систему. Застосунок надає можливість створити нову статтю.	Високий
Авторизований користувач	6 Програма повинна надавати можливість редагувати статтю: 6.1 Програма дозволяє користувачу увійти в систему. 6.2 Програма має надавати можливість вибрати статтю з переліку; 6.3 Програма має надавати можливість користувачу,що створював статтю,змінювати її змінювати коментар адміністратором.	Середній

Таблиця 1.1 – Функціональні вимоги та їх пріоритети

Додати коментар до статті	7 Програма повинна надавати можливість додавати коментар: 7.1 Програма дозволяєкористувачу увійти в систему; 7.2 Програма має надавати можливість вибрати статтю; 7.3 Програма має надавати можливість користувачамдодавати коментарі до,вже створеної будь якимкористувачем, статті.	Високий
Провести пошук за тегом	8 Програма має надавати можливість проводити пошук за тегом; 8.1 Користувач вибирає тегі йому видаються тільки ті статті, що підписані вибраним тегом.	Середній

Продовження таблиці 1.1 – Функціональні вимоги та їх пріоритети

Змінити коментар до статті	9. Додаток надає змогузмінити вже написаний коментар; 9.1 Програма дозволяє користувачу увійти в систему; 9.2 Програма має надавати можливість вибрати зпереліку статтю; 9.3 Програма повиннанадавати можливість вибрати коментар, що написав користувач; 9.4 Також програмно повинно реалізовано можливість змінити вже створений цим користувачем вибраний коментар/адміністратором будь-який коментар.	Середній
Додати тег	10. Додаток надаєможливість створити тег: 10.1 Програма дозволяє користувачу увійти в систему; 10.2.Програма має надавати можливість при створенні статті додавати тег.	Середній

1.4 Висновок до розділу

У цьому розділі було сформульовано опис предметного середовища, наведений приклад процесу діяльності, побудовано функціональну модель та визначено основних дійових осіб та функції, які будуть ними виконуватися. Було розглянуто наявні аналоги продукту, виділення їх переваг та функцій, що вони надають, згідно їх спеціалізації. Описано призначення та цілі розробки, сформульовано задачі, вирішення яких є ключовим для реалізації дипломного проекту.

2.МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Загальні процеси які проходить користувач включають процес реєстрації, входу в систему, пошуку статті з перерахованих чи за тегом, процес вибраної статті та можливість редагування коментаря також процес створення власної статті та її редагування.

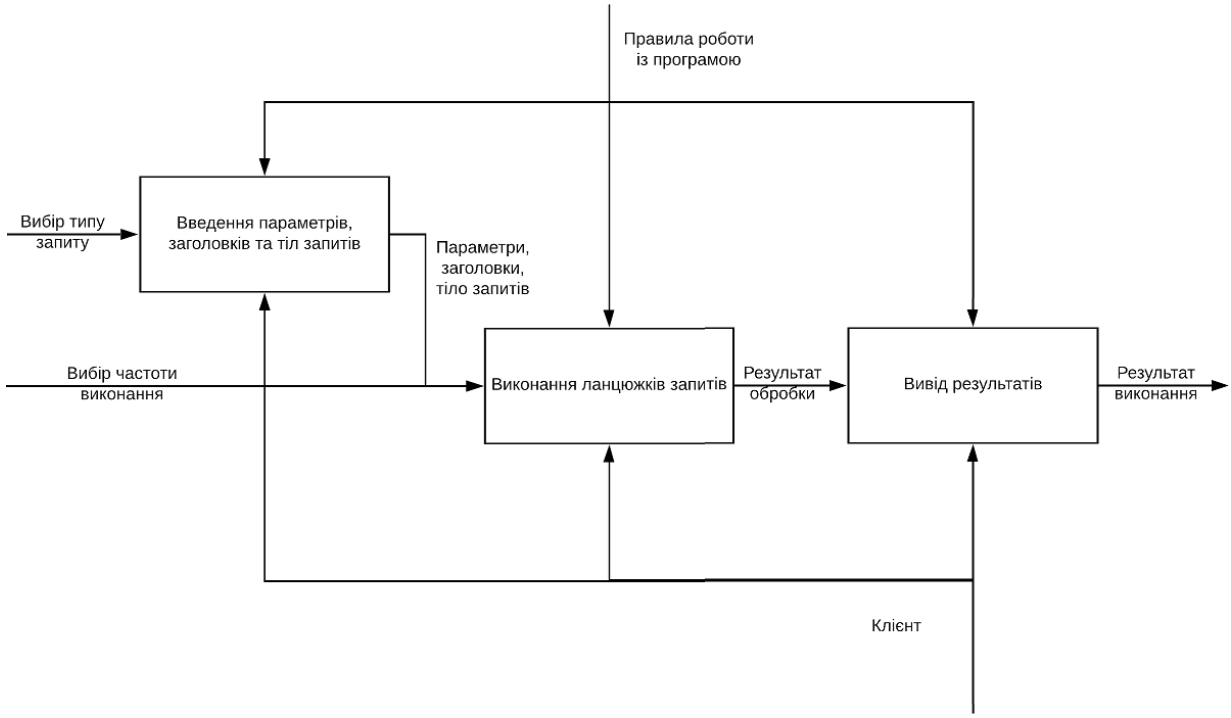


Рисунок 2.1 – Схема бізнес-процесів

Послідовний опис реєстрації нового користувача:

- Користувач заходить на сайт;
- Користувач натискає кнопку “Registration”;
- Вводить данні в поле логін та пароль;
- Натискає кнопку “Submit” і якщо всі поля заповнені коректно то процес реєстрації вважається завершеним.

Послідовний опис входу в систему зареєстрованого користувача:

- Користувач заходить на сайт;
- Користувач натискає кнопку “Login”;

- Вводить данні що він вказував в поле логін та пароль при реєстрації;
- Натискає кнопку “Submit” і якщо всі поля заповнені коректно – входить в системі під своїм логіном;

Послідовний опис створення статті:

- Після входу в систему користувач натискає кнопку “Create”;
- Користувач заповнює поле “Titile”, “Text” та при бажані поле “Tags”;
- Після натискання кнопки “Submit” створюється нова стаття. Послідовний опис додавання коментаря;
- Після входу в систему користувач вибирає статтю , яку хоче прокоментувати;
- Натискає на елемент що має вигляд хмарки діалогу та перед ним відкривається інтерфейс додавання коментарів;
- В поле “Comment” користувач вводить бажаний коментар;
- Після всіх перелічених дій користувач натискає кнопку “Comment” що відправляє коментар в базу даних таким чином створюючи новий коментар.

Послідовний опис редагування статті:

- Користувач заходить на сайт;
- Користувач знаходить статтю що хоче редагувати та натискає на неї;
- Перед користувачем відкривається інтерфейс додавання коментаря дейому потрібно заповнити поле “Comment” даними якими він хоче замінити текст статті;
- Після користувачу потрібно натиснути на елемент що має вигляд нотатника з олівцем – що призведе до заміни тексту статті на те що буловведено користувачем в поле “Comment”.

2.2 Архітектура програмного забезпечення

Розробка програмного забезпечення велась за допомогою мови програмування C# та бази даних MS SQL Server 2019.

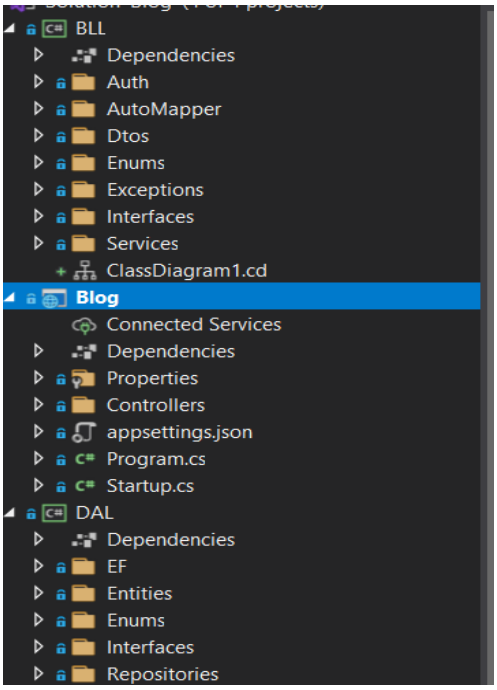


Рисунок 2.2 – проекти програмного забезпечення

Програмне забезпечення було створено за трирівневою архітектурою.

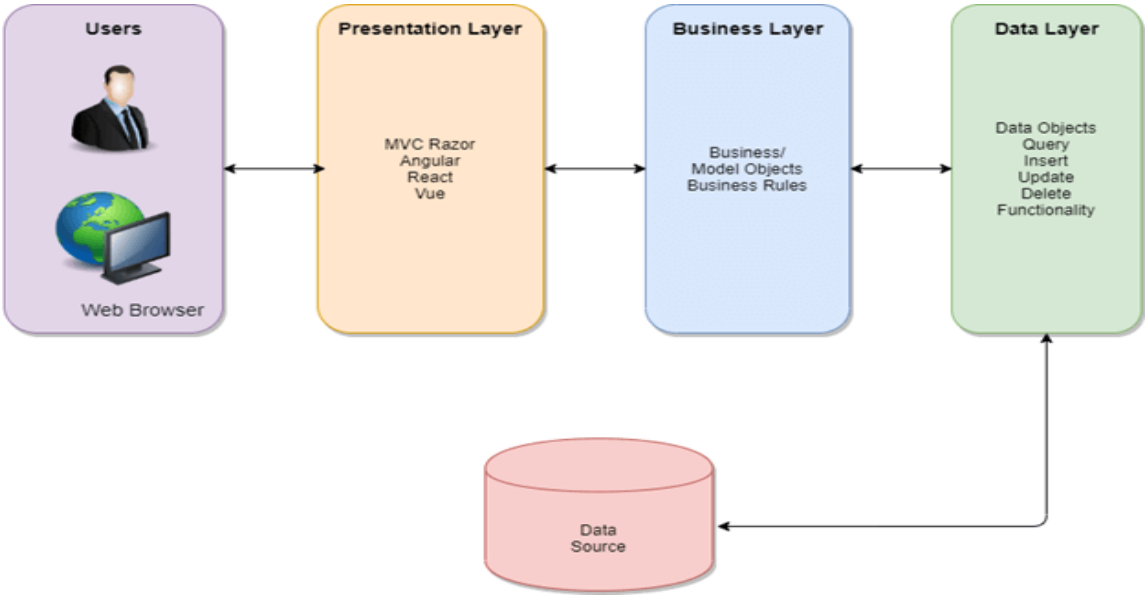


Рисунок 2.3 – Трирівнева архітектура

Складається з трьох частин: Presentation Layer, Business Layer, Data Layer.

У звичайному сенсі трирівнева архітектура ділить все бізнес-додаток на: рівень інтерфейсу користувача, рівень бізнес-логіки і рівень доступу до даних. При проектуванні архітектури програмного забезпечення багаторівнева структура є найбільш поширеною і найважливішою структурою.) та презентаційний шар.

Трирівнева архітектура — це добре встановлена архітектура програмного додатка, яка організовує програми на три логічні та фізичні обчислювальні рівні: рівень презентації або інтерфейс користувача; рівень програми, на якому обробляються дані; і рівень даних, де зберігаються й керуються дані, пов’язані з програмою. Головна перевага трирівневої архітектури полягає в тому, що, оскільки кожен рівень працює на власній інфраструктурі, кожен рівень може одночасно розроблятися окремою командою розробників і за потреби оновлюватися або масштабуватися, не впливаючи на інші рівні.

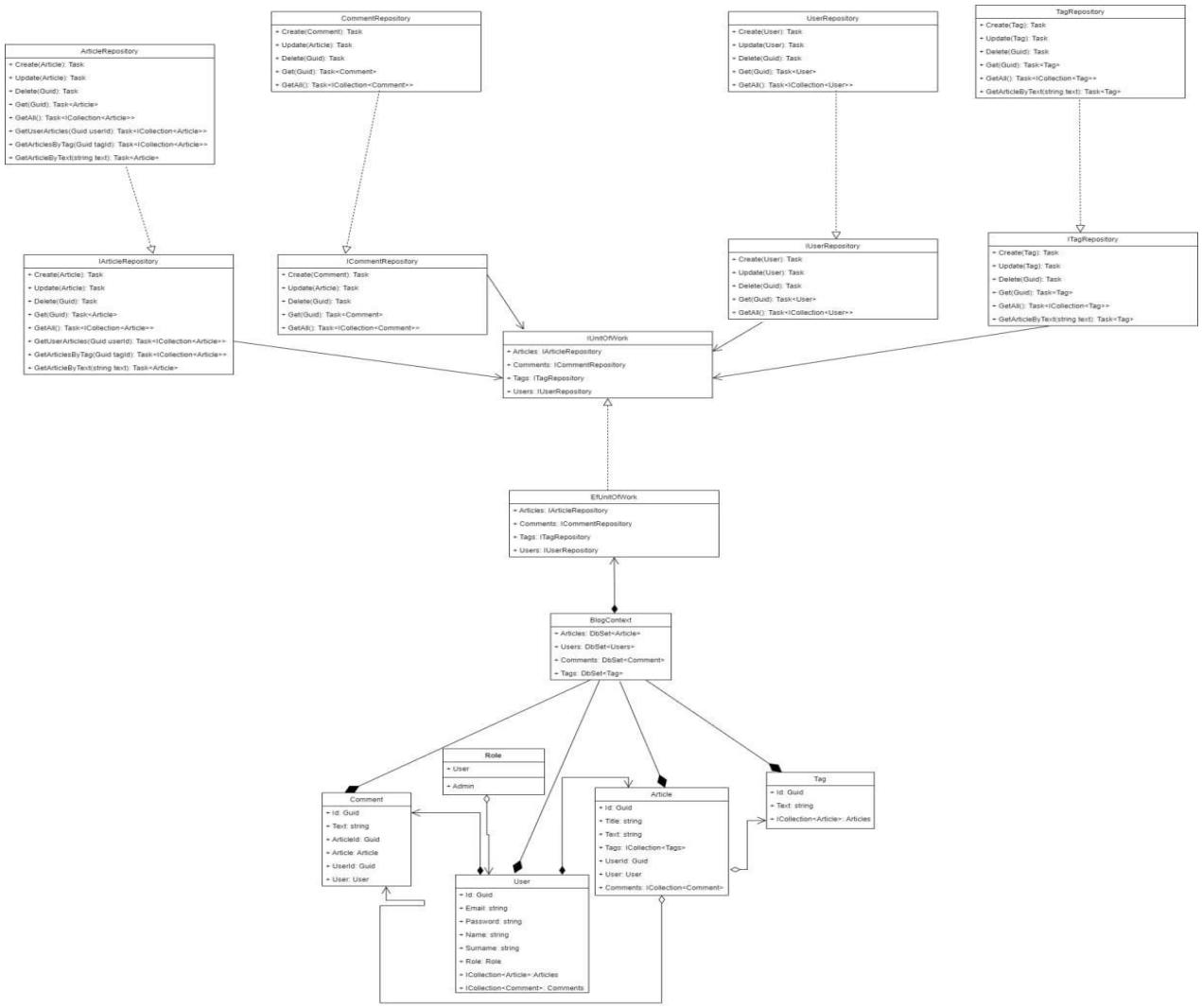


Рисунок 2.4 – діаграма класів шару DAL (Data Access Layer)

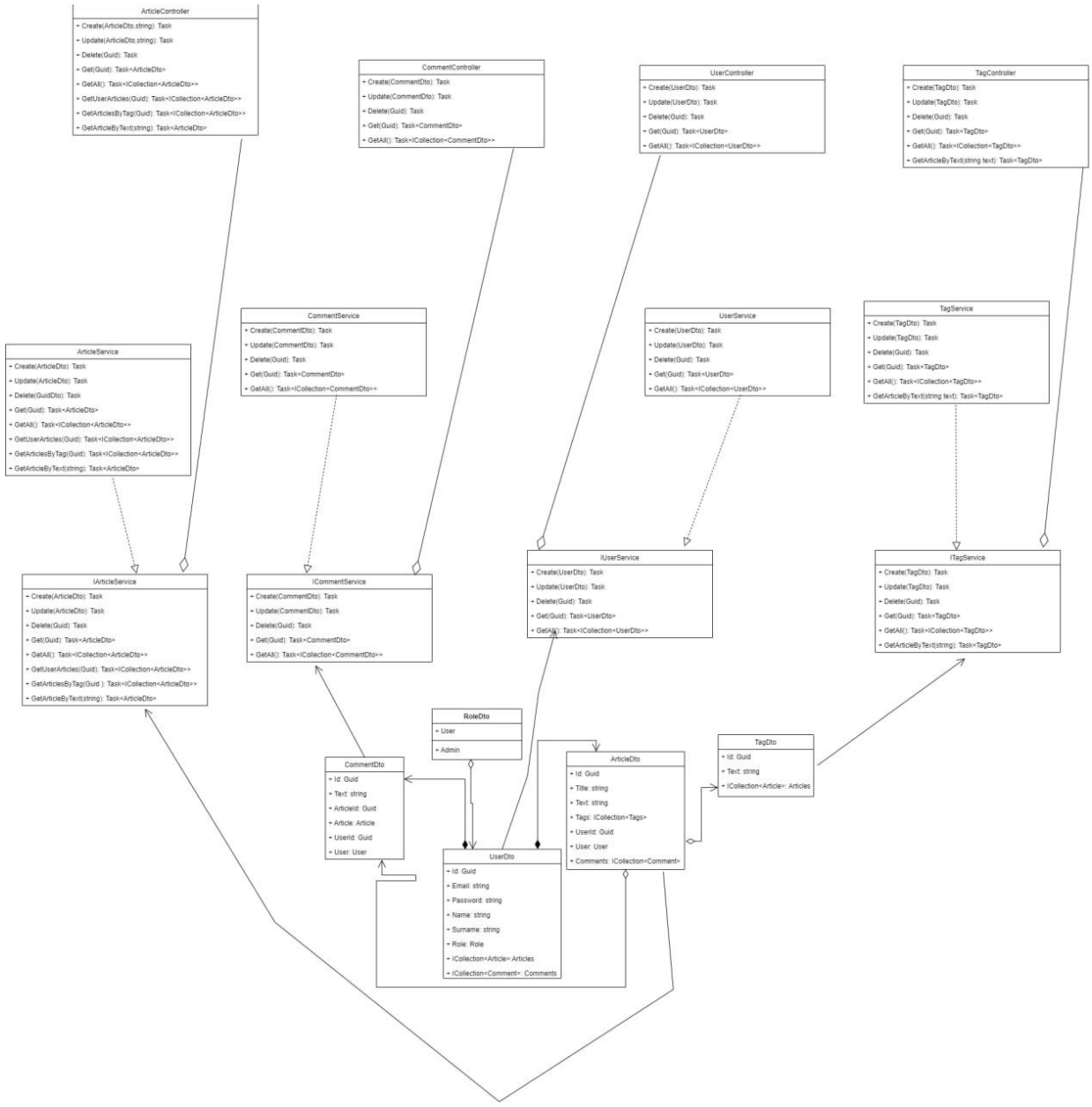


Рисунок 2.5 – діаграма класів для шару BLL(Business Logic Layer)

DAL – це проект, який зберігає моделі бази даних, та опираючись на ці моделі формуються таблиці та зв’язки між таблиці

BLL – це проект, який відповідає за логіку і зберігає сервіси та інтерфейси

Blog – проект, який надає інтерфейс для взаємодії с системою, для виконання запитів

Таким чином, рішення представляє MVC-архітектуру, де відповідно є моделі, які зберігаються як дані і використовуються для обробки різними частинами коду. Є види, які в даному контексті генеруються у вигляді JSON-відповіді. І остання складова – контролери, що сполучають два попередніх компоненти, і відповідають за перетворення даних. Було визначено кожен шар робити окремою бібліотекою класів, оскільки у кожного проекті важлива прозорість структури і це допомагає уникненню дублювання коду.

Було вирішено, робити кожен шар робити окремою бібліотекою. Завдяки цьому підходу ми маємо можливість на повторюватись, а просто додавати посилання на проект. Це зручно та ми не витрачаємо час на переписування.

Такий підхід називається DRY (“Don’t Repeat Yourself”

2.3 Конструювання програмного забезпечення

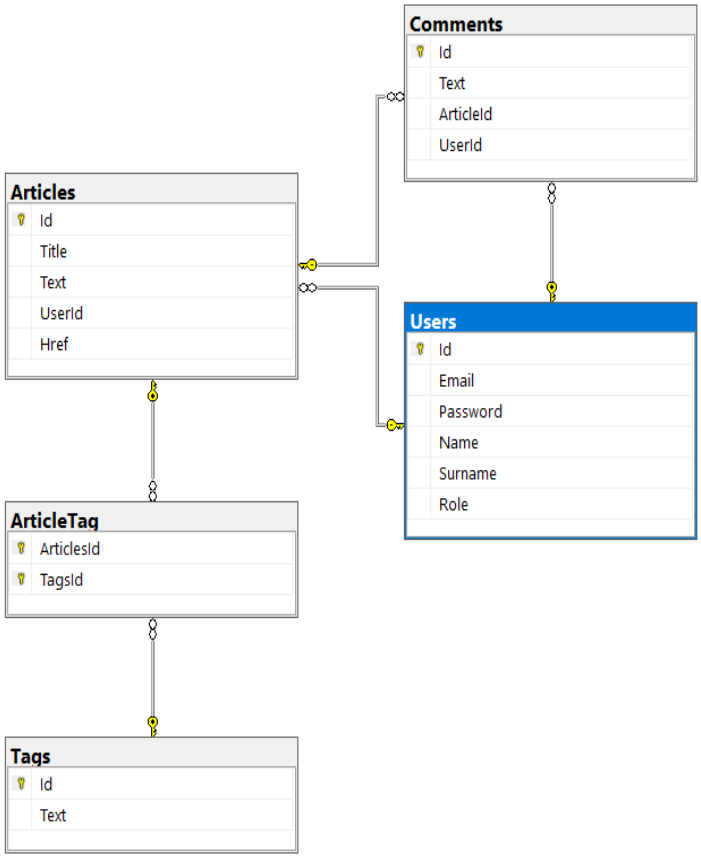


Рисунок 2.6 – діаграма бази даних

Таблиці 2.1 Опис таблиць бази даних

Назва таблці	Опис таблиці	Назва колонки	Тип даних	Опис
Articles	Таблиця статтей	Id	uniqueidentifier	Автоінкремент та первиний ключ
		Text	nvarchar(max)	Текст статті
		Title	nvarchar(max)	Назва статті
		UserId	uniqueidentifier	Зовнішній ключ для таблиці користувачі
		Href	nvarchar(max)	Посилання на картинку
Users	Таблиця користувачів	Id	uniqueidentifier	Автоінкремент та первиний ключ
		Email	nvarchar(max)	Пошта користувача
		Password	nvarchar(max)	Пароль користувача
		Name	nvarchar(max)	Ім’я користувача

Продовження таблиці 2.1

		Surname	nvarchar(max)	Прізвище користувача
		Role	nvarchar(50)	Роль
Comments	Таблиця коментарів	Id	uniqueidentifier	Автоінкремент та первиний ключ
		Text	nvarchar(max)	Текст коментаря
		UserId	uniqueidentifier	Зовнішний ключ користувача
		ArticleId	uniqueidentifier	Зовнішний ключ статті
Tags	Таблця тегів	Id	uniqueidentifier	Автоінкремент та первиний ключ
		Text	nvarchar(max)	Текст тегу
ArticleTag	Таблиця тегів до статтей	ArticleId	uniqueidentifier	Зовнішний ключ статті
		TagId	uniqueidentifier	Зовнішний ключ тега

DAL (Data Access Layer) був створений, щоб код, який ми використовуємо для вилучення даних із нашого сховища даних, був відокремлений від бізнес-логіки, згідно з яким вся логіка, необхідна для вашої бізнес-логіки для взаємодії з вашим рівнем даних, ізольована в одному

наборі класів. Це дозволяє нам легко змінити внутрішню технологію фізичного зберігання даних (перейти від XML-файлів до бази даних або, наприклад, від SQL Server до Oracle або MySQL), не надаючи великого впливу (і, якщо все зроблено правильно, не впливаючи) на нашу бізнес-логіку.

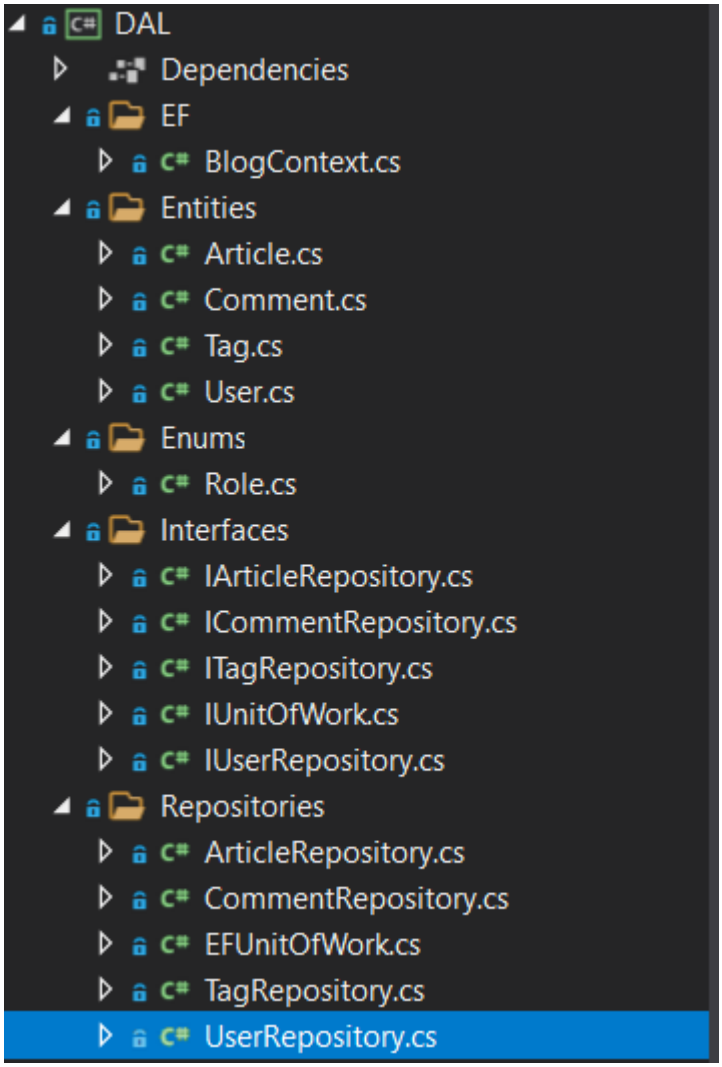


Рисунок 2.6 структура проекту DAL

Для кращої читаємості були придумані рекомендації для написання Data Access Layer. В цих говориться про використання патернів Repository та Unit of Work.

Repository - це не що інше, як клас, визначений для об'єкта, з усіма операціями, можливими цього конкретного об'єкта.

Unit of Work - називається одна транзакція, яка включає кілька операцій вставки/оновлення/видалення і т. д. Простіше кажучи, це означає,

що для конкретної дії користувача (скажімо, реєстрації на веб-сайті) всі транзакції, такі як вставка/оновлення/видалення і т. д. виконуються в одній транзакції, а не в декількох транзакціях бази даних. Це означає, що одна одиниця роботи тут включає операції вставки/оновлення/видалення, все в одній транзакції.

Таблиця 2.1 – Опис спроектованих модулів та відповідних класів в Data Access Layer

DAL (Data Access Layer)	BlogContext	Примірник класу контексту представляє шаблони Unit of Work та Repository, у яких може об'єднувати кілька змін у межах однієї транзакції бази даних. Класконтексту використовується для запиту або збереження даних у базі даних. Він також використовується для налаштуваннякласів домену, зіставлень, пов'язаних із базою даних, параметрів відстеження змін, кешування, транзакцій тощо.

Продовження таблиці 2.1

	EFUnitOfWork	Одиниця роботи в реалізації C# управляє операціями CRUD бази даних у пам'яті над сутностями як однією транзакцією. Отже, якщо одназ операцій завершиться невдало, то будуть відкочуватися всі операції з базою даних. Одиниця роботив C# - це концепція, пов'язана з ефективною реалізацією репозиторій шаблон оформлення.
	ArticleRepository	Реалізує CRUD-інтерфейс для Article
	CommentRepository	Реалізує CRUD-інтерфейс для Comment
	TagRepository	Реалізує CRUD-інтерфейс для Tag
	UserRepository	Реалізує CRUD-інтерфейс для User
	IArticleRepository	Оголошує CRUD-операції для Article

Продовження таблиці 2.1

	ICommentRepository	Оголошує CRUD-операції для Comment
	ITagRepository	Оголошує CRUD-операції для Tag
	IUserRepository	Оголошує CRUD-операції для User
	IUnitOfWork	Оголошує репозиторії
	Role	Перерахування ролей
	Article	Використовується для групування пов'язаних властивостей Article
	Comment	Використовується для групування пов'язаних властивостей Comment
	User	Використовується для групування пов'язаних властивостей User
	Tag	Використовується для групування пов'язаних властивостей Tag

Таблиця 2.2 – Опис викорвстаних методів в Data Access Layer

Назва методу	Параметри	Результат	Опис
Create	Article	Task	Створення статі
Delete	Guid id	Task	Видалення статі
Get	Guid id	Task<Article>	Пошук статі

Продовження таблиці 2.2

GetAll		Task<ICollection <Article>>	Отримати усі статті
Update	Article item	Task	Оновити статтю
GetArticles ByTag	Guid tagId	Task<ICollection <Article>>	Отримати статті за тегом
Create	Comment item	Task	Створити коментар
Delete	Guid id	Task	Отримати коментар
GetAll		Task<ICollection <Comment>>	Отримати усі коментарі
Update	Comment item	Task	Оновити коментар
GetComments ByArticles	Guid articleId	Task<ICollection <Comment>>	Отримати усі коментарі за статею
Create	Tag item	Task	Створити тег
Delete	Guid id	Task	Видалити тег
Get	Guid id	Task<Tag>	Отримати тег
Update	Tag item	Task	Оновити тег
Create	User item	Task	Створення користувача

Основний процес – обробка запитів, їх виконання, а допоміжний – збереження, запис, діставання, оновлення, видалення. Для таких речей були зроблені репозиторії Repository і відповідні сервіси, де останній тип – тип, на який мапиться модель бази даних за допомогою Mapper, а для них – загальні інтерфейси (і для сервісів, і для репозиторіїв).

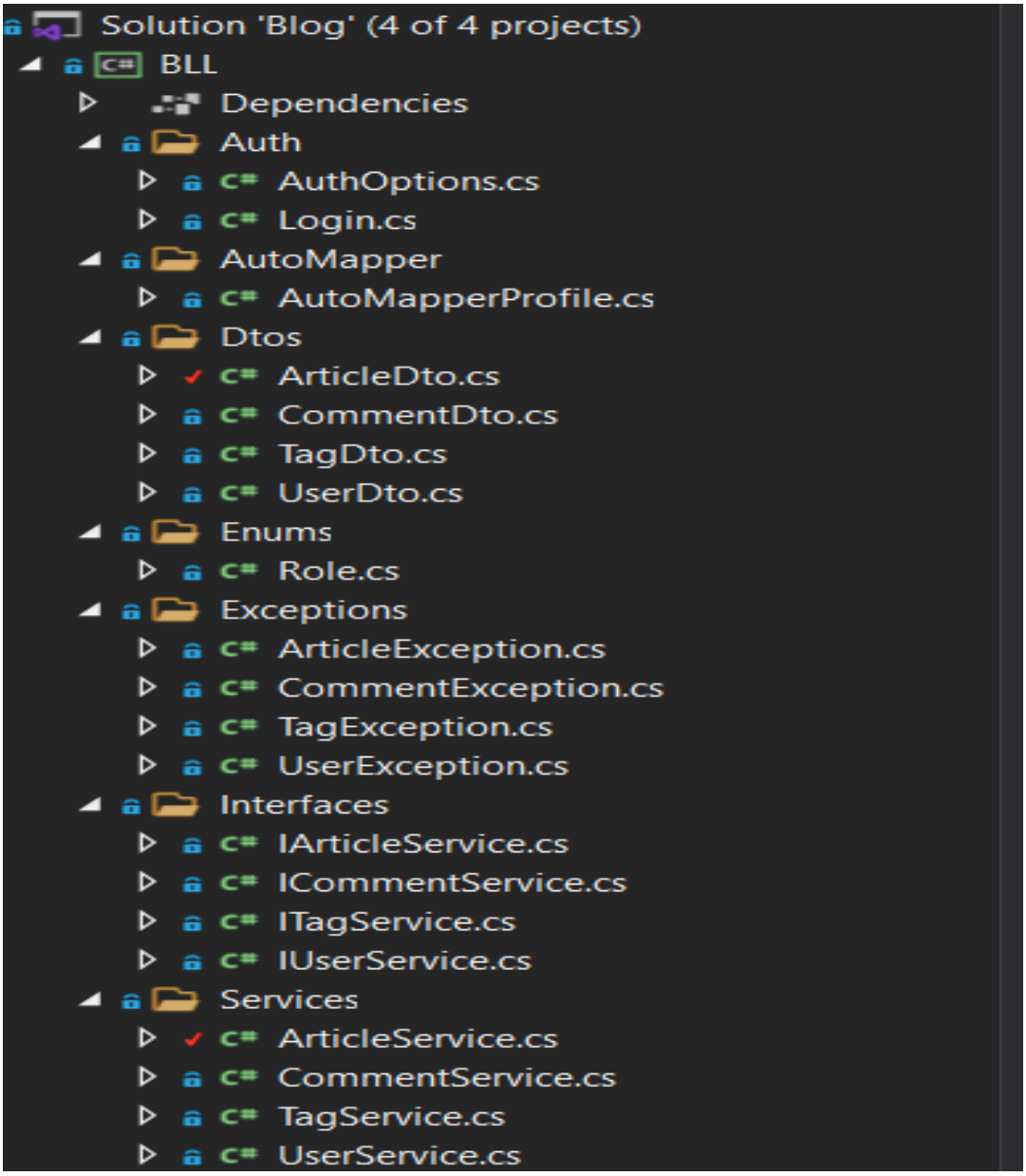


Рисунок 2.7 структура BLL

Рівень бізнес-логіки: бізнес-логіка – це програмування, яке управляє зв'язком між інтерфейсом кінцевого користувача та базою даних.

Основна ідея полягає в тому, щоб ваші контролери були як можна тонше. В основному це означає, що контролер приймає дані із мережі, встановлює змінні, необхідні в представленні та вибирає представлення.

Сервіси та DTO грають величезну роль у Business Logic Layer. За допомогою сервісів, ми можемо ми обробляємо запит, та логіка усього застосунка зберігається в сервісах. DTO – це клас, який містить дані без будь-якої логіки для роботи з ними. DTO зазвичай використовуються для передачі даних між різними

програмами, або між шарами всередині однієї програми. Їх можна як сховище інформації, єдина мета якого — передати цю інформацію одержувачу.

Таблиця 2.3 – Опис спроектованих модулів та відповідних класів в Business Logic Layer

BLL(Business Logic Layer)	ArticleService	Відповідає за обробку отриманих від рівня уявлень даних для Article, реалізує всюнеобхідну логіку додатка, всі обчислення, взаємодіє з базоюданих та передає рівню уявлення результат обробки.
	CommentService	Відповідає за обробку отриманих від рівня уявлень даних для Comment, реалізує всю необхідну логіку додатка,всі обчислення, взаємодіє з базою даних та передає рівню уявлення результат обробки.
	TagService	Відповідає за обробку отриманих від рівня уявлень даних для Tag, реалізує всю необхідну логіку додатка, всі обчислення, взаємодіє з базою даних.

Продовження таблиці 2.3

	UserService	Відповідає за обробку отриманих від рівня уявлень даних для User, реалізує всю необхідну логіку додатка, всі обчислення, взаємодіє з базою даних та передає рівню уявлення результат обробки.
	IArticleService	Оголошує методи обробку отриманих від рівня уявлень даних для Article, реалізує всюнеобхідну логіку додатка, всі обчислення, взаємодіє з базою даних та передає рівню уявлення результат обробки.
	IUserService	Оголошує методи обробку отриманих від рівня уявлень даних для User, реалізує всю необхідну логіку додатка, взаємодіє з базою даних та передає рівню уявлення результат обробки.

Продовження таблиці 2.3

	ICommentService	Оголошує методи обробку отриманих від рівня уявлень даних для Comment, реалізуєвсю необхідну логіку додатка, всі обчислення, взаємодіє збазою даних та передає рівню уявлення результат обробки.
	ITagService	Оголошує методи обробку отриманих від рівня уявлень даних для Tag, реалізує всю необхідну логіку додатка, всі обчислення, взаємодіє з базою даних та передає рівню уявлення результат обробки.
	ArticleException	Article помилка, що виникають під час виконання програми.
	UserException	User помилка, що виникають під час виконання програми.
	CommentException	Comment помилка, що виникають під час виконання програми.

Продовження таблиці 2.3

	TagException	Tag помилка, що виникають під час виконання програми.
	Role	Перерахування ролей
	ArticleDto	Використовується для передачі даних для Article
	UserDto	Використовується для передачі даних для User
	TagDto	Використовується для передачі даних для Tag
	CommentDto	Використовується для передачі даних для Comment

Таблиця 2.4 – Опис викорвстаних методів в Business Logic Layer

Назва методу	Параметри	Результат	Опис
Create	ArticleDto	Task	Створення статі
Delete	Guid id	Task	Видалення статі
Get	Guid id	Task<ArticleDto>	Пошук статі
GetAll	-	Task<ICollection<ArticleDto>>	Отримати статті за тегом
Update	Comment item	Task	Оновити коментар

Продовження таблиці 2.4

GetCommentsByArticles	Guid articleId	Task<ICollection<CommentDto>>	Отримати коментарі за статтею
Create	Tag item	Task	Створити тег
Delete	Guid id	Task	Видалити тег
Get	Guid id	Task<TagDto>	Отримати тег
Update	TagDto item	Task	Оновити тег
Create	UserDto item	Task	Створення користувача

Таблиця 2.5 – Опис спроектованих модулів та відповідних класів в Blog

Blog	ArticleController	Клас, який містить логіку, потрібну для обробки запиту для Article.
	UserController	Клас, який містить логіку, потрібну для обробки запиту для User.
	TagController	Клас, який містить логіку, потрібну для обробки запитудля Tag.

Продовження таблиці 2.5

	CommentController	Клас, який містить логіку, потрібну для обробки запиту для Comment.
	AuthController	Клас, який містить логіку, потрібну для обробки запиту для авторизації.
	Startup	Клас робить конфігурацію програми, налаштовує сервіси,які програма буде використовувати, встановлює компоненти для обробки запиту або middleware.
	Program	запуск застосунка

Frontend був зроблений за допомогою Angular ,TypeScript та JavaScript

Frontend-частина нашого додатку поділяється на три категорії: моделі, сервіси та компоненти. Компоненти поділяється на файл сторінки(html-файл), файл стилю(css-файл) та файл поведінки(ts-файл). Frontend отримує дані з Backend у вигляді JSON, і далі береться за діло компоненти. Angular з’ясовує, який компонент використовується, потім далі їде отримання даних з JSON. Сервіс бере дані з JSON обробляє їх, та зберігає, поки ми не використовуємо дані. Під кожну дію в Angular, для кращого написання коду, та кращого розуміння ми виносили код в окремі файли. Компоненти – це клас, з якимсь випадком. Компоненти - це найголовніший будівельний блок інтерфейсу програми Angular. Angular містить дерево

компонентів Angular. Компоненти Angular - це підмножина директив, завжди пов'язаних із шаблоном. На відміну від інших директив, лише один компонент може бути створений для даного елемента шаблону. Приклад: потрібно було у програмному забезпечені, щоб користувач мав можливість створити статтю. Для цього ми створюємо компонент для цієї дії.

```
@Component({
  selector: 'app-article-create',
  templateUrl: './article-create.component.html',
  styleUrls: ['./article-create.component.css']
})
export class ArticleCreateComponent implements OnInit {

  constructor(public service:ArticleService) { }
  article: Article = new Article();
  token: string | any = localStorage.getItem("jwt");
  form: FormGroup;
  tagsText: string;
  helper = new JwtHelperService();
  receivedArticle: Article | undefined; // полученная статья
  done: boolean = false;
  userId: Guid;



  decodedToken = this.helper.decodeToken(this.token);
```

Рисунок 2.7 приклад використання сервісу в компоненті

```
<div class="container">
    <form novalidate #form="ngForm" >
        <div class="form-group">
            <label for="exampleInputTitle">Title</label>
            <input  required type="email" class="form-control" id="exampleInputTitle"
                placeholder="Input title" #title="ngModel" name="title" #email="ngModel" [(ngModel)]="article.title">
            <small class="text-danger" [class.d-none]="title.valid || title.untouched">Title is required</small>
        </div>
        <div class="form-group">
            <label for="exampleInputText">Text</label>
                <div>
                    <textarea required name="" id="exampleInputText" cols="168" rows="5" placeholder="Content" name="text" #text="ngModel
                    [(ngModel)]="article.text" class="form-control" style="width: 100%; box-sizing: border-box; font: 400 13.3333px Arial
                </div>
                <small class="text-danger" [class.d-none]="text.valid || text.untouched">Text is required</small>
        </div>
        <div class="form-group">
            <label for="exampleInputTags">Tags</label>
            <input type="text" required class="form-control" id="exampleInputTags" name="tags" #tags="ngModel" placeholder="Input tags" n
                [(ngModel)]="tagsText">
            <small style="color: red;" class="tags-danger" [class.d-none]="tags.valid || tags.untouched">Tag or tags is
                required</small>
        </div>
```

Рисунок 2.8 html – код компонента

Таблиця 2.6 Файли Angular

Файл	Опис
article-create.component.ts	Компонент для створення статі. Компонент створює статі на стороні Frontend створюється JSON, потім усе відправляється на Backend, щоб створити статтю
article-create.component.html	Веб-сторінка для створення статті
article-create.component.css	Файл стилів, що визначає позиціонування та відображення контенту на веб-сторінці для створення статей.
article-edit.component.ts	Компонент для редагування статі. Компонент редагування статі на стороні Frontend створюється JSON, потім усе відправляється на Backend
article-edit.component.html	Веб-сторінка для редагування статті
article-edit.component.css	Файл стилів, що визначає позиціонування та відображення контенту на веб-сторінці. Для редагування статті
article-show.component.ts	Компонент для отримання статті, Backend надсилає на Frontend статтю, яка нас цікавить

Продовження таблиці 2.6

article-show.component.html	Веб-сторінка для показу статті
article-show.component.css	Файл стилів, що визначає позиціонування та відображення контенту на веб-сторінці. Для показу статті
articles-show.component.ts	Компонент для отримання статей, Backend надсилає на Frontend статті.
articles-show.component.html	Веб-сторінка для показу статей
articles-show.component.css	Файл стилів, що визначає позиціонування та відображення контенту на веб-сторінці для показу статей
articles-tag.component.ts	Компонент для отримання тегів для статті, Backend вертає JSON з статтями та тегами до статей
articles-tag.component.html	Веб-сторінка для показу тегів
article-tag.component.css	Файл стилів для виведення тегів для статей
login.component.ts	Компонент для логування користувача в систему
login.component.html	Веб-сторінка для логування
nav.component.ts	Компонент для навігаційної панелі
nav.component.html	Веб-сторінка для навігаційної панелі
user-form.component.ts	Компонент для заповнення даних користувача

Продовження таблиці 2.6

user-form.component.html	Веб-сторінка для заповнення даних користувача
article.model.ts	Файл, який містить усі характеристики статті
article.service.ts	Файл, який впроваджує логіку для статті та дає можливість відсилає запити на Backend та приймати запити з Backend
comment.model.ts	Файл, який містить усі характеристики коментаря
comment.service.ts	Файл, який впроваджує логіку для коментаря та дає можливість відсилає запити на Backend та приймати запити Backend
tag.model.ts	Файл, який містить усі характеристики тегу
tag.service.ts	Файл, який впроваджує логіку для тегу та дає можливість відсилати запити на Backend та приймати запити з Backend
user.model.ts	Файл, який містить усі характеристики користувача
user.service.ts	Файл, який впроваджує логіку для користувача та дає можливість відсилати запити на Backend та приймати запити з Backend

Продовження таблиці 2.6

auth-guard.service.ts	Відповідає за авторизацію та аутентифікацію користувачів
app.module.ts	Файл, який запускаю проект

Таблиця 2.7 Опис методів в Angular

Файл	Метод	Результат	Параметри	Опис
article-create-component.ts	addArticle	void	Article	Додає статтю
	ngOnInit	void	-	Викликається для встановлення властивостей
article-edit.component.ts	isUserRoleAdmin	void	-	З`ясовує роль користувача
	editArticle	void	Article	Редагує статтю
	ngOnInit	void	-	Викликається для встановлення властивостей
article-show.component	addComment	void	Comment	Добавляє коментар до статті
	isUserRoleAdmin	void	Article	З’ясовуємо роль користувача

Продовження таблиці 2.7

	isAccess	boolean	Article	З’ясовуємо чи має доступ користувач до статті
	isUserComment	boolean	Comment	З’ясовуємо чи має доступ користувач до коментаря
	isUserAuthenticated	boolean	-	З’ясовуємо чи існує користувач
	onEdit	void	Guid	Редагування коментаря
articles-shows.component.ts	isUserRoleAdmin	void	-	З’ясовуємо роль користувача
	isAccess	boolean	-	З’ясовуємо чи має доступ користувач до статті
	isUserAuthenticated	boolean	-	З’ясовуємо чи існує користувач
	sortByTag	void	Article[]	Сортує по тегам статті

Продовження таблиці 2.7

	initialize EventListeners	void	-	Обробник кнопки
	ngOnInit	void	-	Викликається для встановлення властивостей
	ngDoCheck	void	-	Викликається для перевірки змін у властивостей
articles-tag-component.ts	ngOnInit	void	-	Викликається для встановлення властивостей
	Reload	void	-	Збереження змін
auth-guard.service.ts	canActivate	boolean	-	Визначає термін придатності
login.component.ts	login	any	-	Виконує вхід у систему
	reboot	void	-	Перезавантаження веб-застосунку
	l	void	NgForm	Заповнення форми входу

Продовження таблиці 2.7

nav.component.ts	isUserAuthenticated	boolean	-	З’ясовуємо чи існує користувач
	isAdmin	boolean	-	З’ясовуємо роль користувача
	logOut	void	-	Вихід з веб-застосунка
article-service.ts	getArticles	Article[]	-	Отримання усіх статей
	getArticle	Article	Guid	Отримання статті
	getArticlesByTag	Article[]	Guid	Отримання усіх статей за тегом
	addArticle	Object	Article, Guid	Додає статтю
	editArticle	Object	Article	Редагування статті
comment.service.ts	createComment	Object	Guid, Comment	Створити коментар
	updateComment	Object	Guid, Comment	Редагування коментаря
tag.service.ts	getTags	Tag[]	-	Отримати теги

Продовження таблиці 2.7

user.service.ts	getUsers	User[]	-	Отримування усіх користувачів
	postUser	Object	User	Створити користувача
user-form.component.ts	addUser	void	User	Створити користувача
	isUserAuthenticated	boolean	-	З’ясовуємо чи існує користувач
	isAdmin	void	-	Перевірка чи є роль користувача адміністратор
user-show.component.ts	fetchUsers	void	-	Отримати усіх користувачів
	ngOnInit	void	-	Викликає метод fetch при завантаженні веб-сторінки та ініціалізує усі властивості

2.4 Аналіз безпеки даних

Питання забезпечення безпеки та працездатності системи є невід'ємною частиною етапу її проектування. Розуміння того, що саме має бути захищене від зовнішніх факторів, допомагає з найефективнішим вибором та застосуванням захисних заходів. Подальше ускладнення у тому, що у одному комп'ютері чи сервері може розміщуватися як загальнодоступна, і конфіденційна інформація. В результаті кілька політик безпеки можуть застосовуватись до однієї машини або в межах однієї системи. Тому при розробці інформаційної системи межі безпеки повинні враховуватися та описуватися у відповідній документації та політиках безпеки системи. Ми повинні вміти забезпечити безпеку системи при проектуванні, розробці, управлінні та конфігуруванні, інтеграції, правильно провести тестування.

У програмному забезпечені відсутні секретні дані , проте захист є у веб-застосунку. Є JWT, яка може ідентифікувати кожен запит за користувачем, також, якщо змінюємо дані чи інше, йде перевірка чи потрібні дані належать даному користувачу.

JWT відрізняються від інших веб-токенів тим, що містять набір тверджень. Затвердження використовуються передачі інформації між двома сторонами. Що є ці твердження, залежить від розглянутого варіанта використання. Наприклад, у твердженні може бути зазначено, хто видав токен, як довго він дійсний або які дозволи були надані клієнту.

JWT - це рядок, що складається з трьох частин, розділених точками (.), і серіалізований з використанням base64. У найбільш

поширеному форматі серіалізації, компактної серіалізації, JWT виглядає приблизно так: xxxxx.yyyyy.zzzzz.

2.5 Висновки до розділу

Реалізували архітектура для нашого програмного забезпечення, реалізували методи, інтерфейси та залежності для роботи нашого програмного забезпечення також був створений Frontend завдяки Angular та створили компоненти. Також імплементували JWT для безпеки даних.

3. АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування Програмного Забезпечення — це широкий процес, який складається з декількох взаємопов'язаних процесів. Іншими словами сукупності процесів[1].

Тестування та аналіз програмного забезпечення грає величезну роль у побудові програмного забезпечення. Бажано не пропускати цей пункт, а відноситись с усією серйозністю, тому що дає нам розуміння на якій фазі розробка програмного забезпечення і надання актуальної інформації. Тестувати можна різні рівні (операційне, модульне, інтеграційне, системне, прийняття) та типів (функціональне, нефункціональне, пов`язане зі змінами).Тестування може проводитись мануальне та автоматизоване. При розробці програмного забезпечення було використано такі фреймворки, як NUnit. На сьогоднішній день існує багато бібліотек для написання тестів, а саме створення мок-класів, створення стандартних задач, вхідних параметрів, налаштувань. Для .NET Core є бібліотеки такі як FluentAssertions, AutoFixture, Moq, AutoFixture, AutoMoq, AutoFixture.Idioms та багато інших.

При написанні тестів, варто ще перевіряти відповідність вимог до інтерфейсу користувача, від цього залежить коректність роботи та сприйняття.

3.2 Опис процесів тестування

Тестування відбувається за принципом white-box та block-box.

White-box – метод тестування програмного забезпечення, коли тестувальнику потрібно знати про архітектуру, внутрішню структуру та реалізацію програмного забезпечення. Обираються дані, базуючись на розумінні коду. Також ми знаємо про результат, який повинен бути, при виконанні тесту. White-box – коли були створенні тести за допомогою фреймворків з знанням коду

Black-box – метод тестування програмного забезпечення, коли тестувальнику не потрібно про архітектуру, внутрішню структуру та реалізацію програмного забезпечення. Базується на роботі с інтерфейсом користувача. Black-box – при тестуванні інтерфейсу користувача, діючи як звичайний користувач.

Під тестування було виявлено певні недоліки, що привело видаленню методів, які не використовуються. В результаті були класи більш лаконічні, ніж раніше.

Метою було переконатись у стабільності роботи програмного забезпечення, виявлення недоліків, чи виконуються поставленні задачі та чи виконуються, як потрібно.

У наступному розділі розглядаються тести, протоколюючи мету, вхідні дані, початковий стан, опис проведення тесту, очікуваний результат та фактичний результат для кожного.

3.3 Опис контрольно прикладу

При тестування програмного забезпечення були перевірені усі вимоги, що наводилися та результати буду показані нижче у таблицях

- Додавання користувача

- Додавання статті
- Редагування статті
- Перегляд статті
- Вхід в систему
- Доступ до редагування статті
- Створити коментар
- Редагувати коментар
- Доступ до редагування коментаря
- Перегляд статті
- Перегляд усіх статей за тегом
- Перегляд користувачів
- Видалення статті
- Доступ до редагування статті іншого користувача
- Доступ до редагування коментаря іншого користувача
- Вихід із системи

Далі можна переглянути, як і що тестувалось

Таблиця 3.1 – Тест 1

Мета тесту	Додавання користувача веб-застосунок
Початковий стан	Відкритий веб-застосунок з усіма статтями
Вхідні дані	Прізвище, ім’я, електронна пошта, пароль

Продовження таблиці 3.1

Опис проведення тесту	Були введені дані та створений користувач та натиснута кнопка <<Зареєструватись>>
Очікуваний результа	Переглядаючи користувачів ми бачимо нового користувача, якого ми створили
Отриманий результат	Переглядаючи користувачів ми бачимо нового користувача, якого ми створили

Таблиця 3.2 – Тест 2

Мета тесту	Додавання статті у веб-застосунок
Початковий стан	Відкритий веб-застосунок з усіма статтями
Вхідні дані	Теги, текст статті
Опис проведення тесту	Були введені дані та створена стаття та натиснута кнопка <<Створити>>
Очікуваний результат	Переглядаючи статті ми бачимо нову статтю, яку ми створили
Отриманий результат	Переглядаючи статті ми бачимо нову статтю, яку ми створили

Таблиця 3.3 – Тест 3

Мета тесту	Редагування статті у веб-застосунок
Початковий стан	Відкритий веб-застосунок з усіма статтями
Вхідні дані	Теги, текст статті
Опис проведення тесту	Була обрана стаття та введені дані, потім натиснута кнопка <<Редагувати>>
Очікуваний результат	Бачимо оновлену статтю.
Отриманий результат	Бачимо оновлену статтю.

Таблиця 3.4 – Тест 4

Мета тесту	Перегляд статей у веб-застосунку
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Переходимо на головну сторінку та бачимо усі статті
Очікуваний результат	Переглядаючи статті ми бачимо оновлену статтю.
Отриманий результат	Переглядаючи статті ми бачимо оновлену статтю.

Таблиця 3.5 – Тест 5

Мета тесту	Вхід у систему
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	Пошта та пароль
Опис проведення тесту	Натискаємо кнопку <<Ввійти у систему>> та вводимо дані та потім натискаємо кнопку <<Зареструватись>>
Очікуваний результат	Увійшли як користувач
Отриманий результат	Увійшли як користувач.

Таблиця 3.6 – Тест 6

Мета тесту	Доступ до редагування статті
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	Стаття, текст, теги та користувач
Опис проведення тесту	Обираємо статтю та натискаємо кнопку <<Редагувати>> та вводимо дані та потім натискаємо кнопку <<Опублікувати>>
Очікуваний результат	Переглядаючи статті ми бачимо, що ми увійшли як користувач та редагували статтю.
Отриманий результат	Переглядаючи статті ми бачимо, що ми увійшли як користувач та редагували статтю.

Таблиця 3.7 – Тест 7

Мета тесту	Додавання коментаря у веб-застосунок
Початковий стан	Відкритий веб-застосунок з усіма статтями
Вхідні дані	Стаття та текст коментаря
Опис проведення тесту	Була обрана стаття, потім натиснута кнопка <<Коментарі>>, де створюємо
Очікуваний результат	Переглядаючи статті ми бачимо новий коментар, який ми створили
Отриманий результат	Переглядаючи статті ми бачимо новий коментар, який ми створили

Таблиця 3.8 – Тест 8

Мета тесту	Редагування коментаря
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	Стаття,текст та теги та користувач
Опис проведення тесту	Обираємо статтю та натискаємо кнопку <<Коментарі>> та вводимо текст коментаря та потім натискаємо кнопку <<Редагувати>>
Очікуваний результат	Переглядаючи статті ми бачимо оновлені коментарі
Отриманий результат	Переглядаючи статті ми бачимо оновлені коментарі

Таблиця 3.9 – Тест 9

Мета тесту	Доступ до редагування коментаря
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	Стаття, текст та користувач
Опис проведення тесту	Обираємо статтю та натискаємо кнопку <<Коментарі>>, потім обираємо коментар, вводимо дані та натискаємо кнопку <<Редагувати>>
Очікуваний результат	Переглядаючи статті ми бачимо, що коментарі змінюються
Отриманий результат	Переглядаючи статті ми бачимо, що коментарі змінюються

Таблиця 3.10 – Тест 10

Мета тесту	Перегляд статей за тегом
Початковий стан	Веб-сторінка с усіма статтями
Вхідні дані	Тег
Опис проведення тесту	Обираємо тег
Очікуваний результат	Переглядаючи статі бачимо, усі статі з обраним тегом
Отриманий результат	Переглядаючи статі бачимо, усі статі з обраним тегом

Таблиця 3.11 – Тест 11

Мета тесту	Доступ до редагування коментаря
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	Стаття, текст та користувач
Опис проведення тесту	Обираємо статтю та натискаємо кнопку <<Коментарі>>, потім обираємо коментар, вводимо дані та натискаємо кнопку <<Редагувати>>
Очікуваний результат	Переглядаючи статті ми бачимо, що коментарі змінюються
Отриманий результат	Переглядаючи статті ми бачимо, що коментарі змінюються

Таблиця 3.12 – Тест 12

Мета тесту	Огляд користувачів
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Заходимо, як адміністратор та дивимося усіх зареєстрованих користувачів
Очікуваний результат	Бачимо усіх користувачів
Отриманий результат	Бачимо усіх користувачів

Таблиця 3.13 – Тест 13

Мета тесту	Видалення статі
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Обираємо статтю, Натискаємо кнопку <<Видалити>>
Очікуваний результат	Видалена стаття
Отриманий результат	Видалена стаття

Таблиця 3.14 – Тест 14

Мета тесту	Видалення статі
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Обираємо статтю, Натискаємо кнопку <<Видалити>>
Очікуваний результат	Видалена стаття
Отриманий результат	Видалена стаття

Таблиця 3.15 – Тест 15

Мета тесту	Доступ до редагування статті
Початковий стан	Доступ до редугавання статті
Вхідні дані	-

Продовження таблиці 3.15

Опис проведення тесту	Заходимо, як користувач та обираємо статтю іншого користувача.
Очікуваний результат	Немає можливості
Отриманий результат	Немає можливості

Таблиця 3.16 – Тест 16

Мета тесту	Доступ до редагування коментаря іншого користувача
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Заходимо, як користувач та обираємо коментар іншого користувача.
Очікуваний результат	Немає можливості
Отриманий результат	Немає можливості

Таблиця 3.17 – Тест 17

Мета тесту	Вихід із системи
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Натискаємо кнопку <<Вийти>>
Очікуваний результат	Вихід
Отриманий результат	Вихід

Таблиця 3.18– Тест 18

Мета тесту	Вихід із системи
Початковий стан	Перегляд статей веб-застосунку з усіма статтями
Вхідні дані	-
Опис проведення тесту	Натискаємо кнопку <<Вийти>>
Очікуваний результат	Вихід
Отриманий результат	Вихід

3.4 Висновки по розділу

Проводили тестування, порівняли принципи white-box та black-box

Написали тести для перевірки роботоздатності програмного забезпечення.

Було надано методичку для тестування програмного забезпечення, де описується поведінки та кінцеві результати.

4. ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Щоб працювало програмне забезпечення потрібно .Net Core, ASP.NET, NPM, TypeScript, C#, Node.Js, Angular, T-SQL та MSSQLServer

.Net Core - це платформа загального призначення, яку можна використовувати для створення програмних програм для Windows, Linux та MacOS.[2]

ASP.NET – це платформа веб-розробки, яка надає модель програмування, комплексну програмну інфраструктуру та різноманітні служби, необхідні для створення надійних веб-додатків для ПК та мобільних пристроїв.[3]

NPM - є менеджером пакетів за замовчуванням для середовища виконання JavaScript Node.js.[4]

TypeScript - це наднабір JavaScript, що означає, що він містить усі функції JavaScript, а потім деякі.[5]

Angular - це інтерфейсний фреймворк на основі JavaScript з відкритим вихідним кодом, який використовується для створення користувацьких програм у HTML, CSS і Typescript.[6]

C# - це сучасна, об’єктно-орієнтована та безпечна для типів мова програмування. C# дозволяє розробникам створювати багато типів безпечних і надійних програм, які працюють на .NET.[7]

Node.js - це кросплатформне середовище виконання з відкритим вихідним кодом для розробки серверних і мережевих додатків. Програми Node.js написані на JavaScript і можуть запускатися в середовищі виконання Node.js в OS X, Microsoft Windows і Linux.[8]

T-SQL — це процедурна мова, яка використовується Microsoft у SQL Server. Він додає до SQL оголошені змінні, контроль

транзакцій, обробку помилок і винятків, а також обробку рядків.[9]

MSSQLServer - це система управління реляційною базою даних, або RDBMS, розроблена та продана Microsoft.[10]

Програма ASP.NET Core виконується із внутрішньопроцесною реалізацією HTTP-сервера. Реалізація сервера прослуховує HTTP-запити і передає їх додатку як набору функцій запиту, об'єднаних в HttpContext.

Основний модуль ASP.NET – це власний модуль IIS, який обробляє власні запити IIS між IIS та внутрішньо процесним HTTP-сервером IIS.

При зборі внутрішньо процесного розкриття програми ASP.NET Core застарів у тому процесі, як і робочий процес IIS. Розміщення внутрішнього процесу виявило високу продуктивність порівняно з розміщенням поза процесом, оскільки запити не передаються через адаптер зворотного зв'язку, мережевий інтерфейс, який виявляє вихідний мережевий трафік назад на той самий комп'ютер. IIS отримує управління процесами за допомогою служби активації процесів Windows (WAS).

При розміщенні поза процесом програми ASP.NET Core виконуються в процесі, окремому від робочого процесу IIS, а модуль обробляє управління процесом. Модуль запускає процес для програми ASP.NET Core при надходженні першого запиту та перезапускає програму, якщо вона завершує роботу або дає збій. По суті, це те саме поведінка, що й з програмами, які запускаються в процесі і керуються службою активації процесів Windows (WAS). Використання окремого процесу також дозволяє розміщувати більше однієї програми з одного пулу додатків.

У якості сервера ми використовуємо нашу робочу машину, яку ми обрали за такими причинами:

1. Простіше у використанні

2. Більш безпечніше ніж на хмарі чи на серверу
3. Економія часу та ресурсів

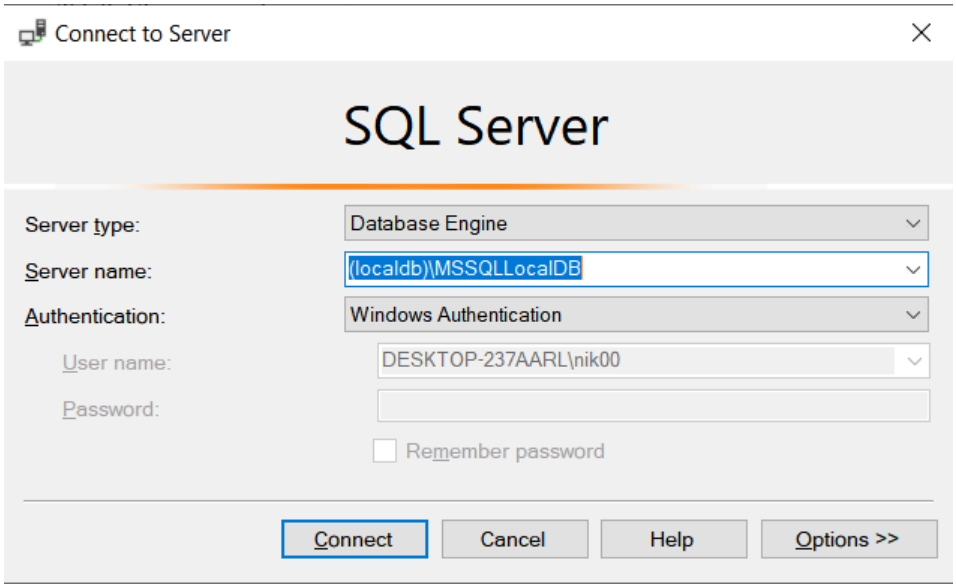


Рисунок 4.1 Ім’я сервера

За замовчуванням спілкування відбуваються за 1433 портом за протоколом TCP/IP

TCP/IP використовує модель зв'язку клієнт-сервер, в якій користувачеві або машині (клієнту) надається послуга, наприклад, відправлення веб-сторінки, іншим комп'ютером (сервером) у мережі.

У сукупності набір протоколів TCP/IP класифікується як стан, що не зберігає , що означає, що кожен запит клієнта вважається новим, оскільки він не пов'язаний з попередніми запитами. Відсутність стану звільняє мережеві шляхи, щоб їх можна використовувати постійно.

Проте сам транспортний рівень має статки. Він передає одне повідомлення, і його з'єднання залишається чинним до тих пір, поки всі пакети повідомлення не будуть отримані та зібрані в пункті призначення.

TCP/IP можна використовувати для забезпечення віддаленого входу в систему через мережу для інтерактивної передачі файлів для доставки електронної пошти, для доставки веб-сторінок по мережі та для віддаленого доступу до файлової системи хост-сервера. У більш

широкому сенсі він використовується для уявлення того, як змінюється форма інформації в міру її переміщення по мережі від конкретного рівня до рівня абстрактних додатків.

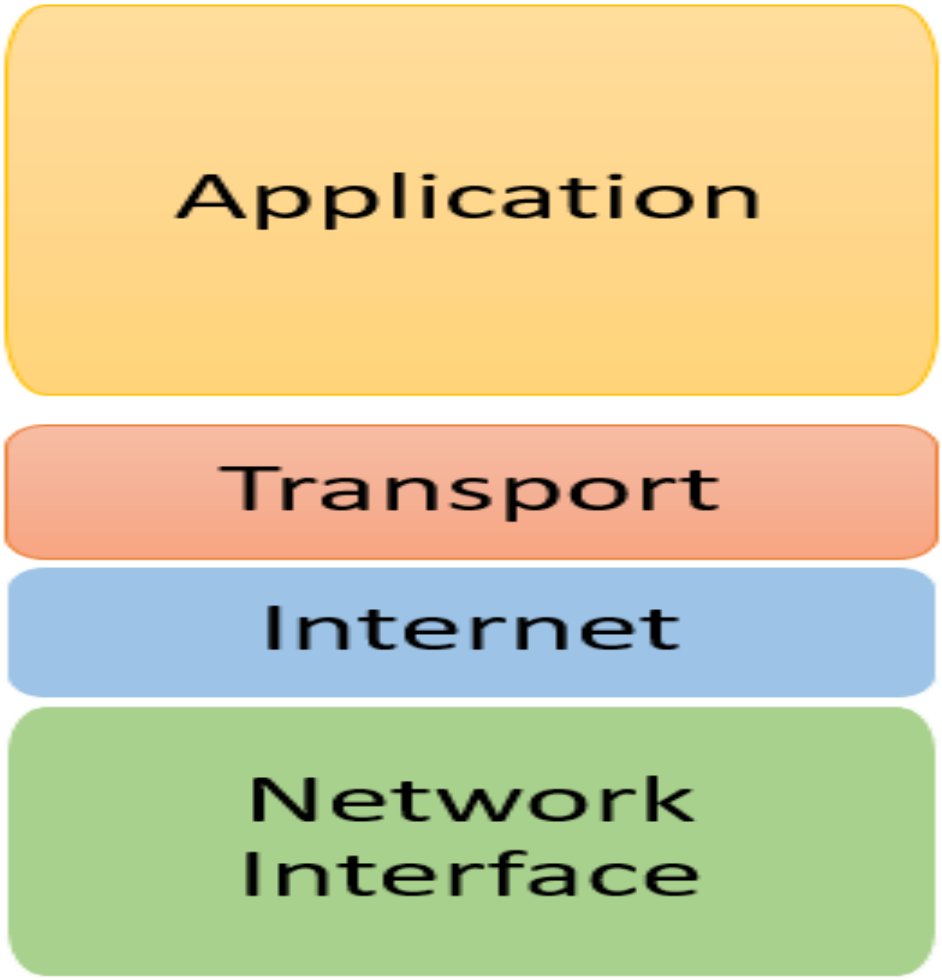


Рисунок 4.2 Модель TCP/IP

Шаблон проекту ASP.NET Core з Angular надає зручну відправну точку для програм ASP.NET Core, що використовують Angular і Angular CLI, для реалізації розширеного клієнтського інтерфейсу (UI).

Шаблон проекту еквівалентний варіанту проекту ASP.NET Core для роботи як веб-API і проекту Angular CLI для роботи як інтерфейс користувача. Таке поєднання проектів вимагало вирішення кількох проектів в одному проекті ASP.NET Core, який можна створити та опублікувати як єдине ціле. Шаблон проекту не призначений для рендерингу на сервері серверів (SSR).

Під час розробки програма працює в режимі, оптимізованому для зручності розробників. Наприклад, пакети JavaScript включають вихідні карти (щоб під час налагодження ви могли побачити свій оригінальний код TypeScript). Програма спостерігає за змінами файлів TypeScript, HTML і CSS на диску та автоматично перекомпілює та перезавантажує, коли бачить зміни цих файлів.

У робочій версії показуйте версію свого додатка, оптимізовану для продуктивності. Це налаштовано на автоматичне виконання. Коли ви публікуєте, конфігурація збірки випускає мінімізовану завчасну (AoT) компільовану збірку вашого клієнтського коду. На відміну від збірки для розробки, виробнича збірка не вимагає встановлення Node.js на сервері (якщо ви не ввімкнули рендеринг на стороні сервера (SSR)).

Проект налаштовано на запуск власного екземпляра сервера Angular CLI у фоновому режимі, коли програма ASP.NET Core запускається в режимі розробки. Це зручно, оскільки вам не потрібно запускати окремий сервер вручну.

У цього налаштування за замовчуванням є недолік. Кожного разу, коли ви змінюєте свій код C# і програму ASP.NET Core потрібно перезапускати, сервер Angular CLI перезапускається. Щоб почати резервне копіювання, потрібно приблизно 10 секунд. Якщо ви часто редагуєте код C# і не хочете чекати перезапуску Angular CLI, запустіть сервер Angular CLI ззовні, незалежно від процесу ASP.NET Core.

Проект налаштовано на запуск власного екземпляра сервера Angular CLI у фоновому режимі, коли програма ASP.NET Core запускається в режимі розробки. Це зручно, оскільки вам не потрібно запускати окремий сервер вручну. У цього налаштування за замовчуванням є недолік. Кожного разу, коли ви змінюєте свій код C# і програму ASP.NET Core потрібно перезапускати, сервер Angular CLI перезапускається. Щоб почати резервне копіювання, потрібно

приблизно 10 секунд. Якщо ви часто редагуєте код C# і не хочете чекати перезапуску Angular CLI, запустіть сервер Angular CLI ззовні, незалежно від процесу ASP.NET Core.

Backend розташовується на веб-сервері IIS Express за портом 26561

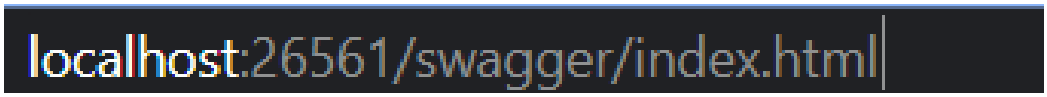


Рисунок 4.3 – url-code Backend-частина

Frontend розташовується за портом 4200. Роль сервера виконує локальна машина

Рисунок 4.4 – url-code Frontend-частина

Між цими портами відбувається робота програмного забезпечення

4.2. Робота з програмним забезпеченням

Як користувачу використовувати програму, який функціонал вона містить, вказано в керівництві користувача.

4.3 Висновки до розділу

У цьому розділі було зроблено розгортання програмного забезпечення та продемонстровано як користувач може користуватися застосунком. Також було надано керівництво роботи з програмним забезпеченням.

ВИСНОВКИ

У процесі виконання програмного забезпечення для дипломного проекту, буда створена та спроектована архітектура для програмного забезпечення. Після проектування та розробки був процес тестування програмного засобу та створений до нього план тестування.

Була створена інструкція для підготовки середовища для запуску, а також розгортанню для запуску застосунка.

Потім тестування на пристроях з різними дисплеями. Це було зроблено, щоб переконатися, що усе відображається правильно на різних пристроях.

Створено програмне забезпечення “блог”, у якого багато сфер використання такі як: маркетинг, співробництво з відомими брендами,

веб-сайт, який маю можливість ділитися інформацією з іншими користувачами

В якості технології було обрано: ASP.NET Core Web Api та Angular. Базу Даних ми використовували MSSQL Server 2019.

За час написання програмного забезпечення для дипломної роботи були засвоєні нові знання з ASP.NET та Angular та для архітектури будь-якого програмного забезпечення. Був вивчений матеріал для застосування тестування в нашому застосунку

Результатом є базова версія програмного забезпечення, яке у майбутньому може перетворитись в щось глобальніше. За допомогою нашого додатку. Люди не витрачаючи час мають можливість дізнаватись інформацію за мінімальну кількість послідовних дій. Наш застосунок також може бути рекламною площадкою для звернення ували іншими людьми та великими компаніями. На сьогодення маркетинг грає величезну роль для бізнесу. Правильна збудована рекламна компанія дає гарантію, що люди та майбутні бізнес-партнери звернуть увагу саме на вас.

Також блог можна використовуватись, як книжка. Наприклад: Окрема стаття – це окрема глава у книзі. В статті описуєш коротко про суть та виділяєш найголовніше – це допомагає усім зберегти час та витрати свій час на щось інше.

ПЕРЕЛІК ПОСИЛАНЬ

1. Тестування програмного забезпечення [Електронний ресурс]. - https://www.quality-assurance-group.com/shho-take-testuvannya-programnogo-zabezpechennya-ta-yake-jogo-znachennya/
2. C# [Електронний ресурс]. - https://www.c-sharpcorner.com/article/what-is-dot-net-core/
3. Asp.Net [Електронний ресурс]. - https://www.tutorialspoint.com/asp.net/asp.net_introduction.htm[
4. Node.JS [Електронний ресурс]. - https://www.freecodecamp.org/news/what-is-npm-a-node-package-manager-tutorial-for-beginners/
5. Typescript [Електронний ресурс]. - https://medium.com/front-end-weekly/typescript-what-is-it-when-is-it-useful-c4c41b5c4ae7
6. Angular [Електронний ресурс]. – https://flatlogic.com/blog/what-is-angular
7. .Net [Електронний ресурс]. - https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/
8. Node.JS [Електронний ресурс]. - https://www.tutorialspoint.com/nodejs/nodejs_introduction.html
9. T-SQL [Електронний ресурс]. - https://www.simplilearn.com/tutorials/sql-tutorial/transact-sql

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ВЕБ-ЗАСТОСУНОК “БЛОГ”

Опис програми

КПІ.ІП-8305.045440.02.83

“ПОГОДЖЕНО”
Керівник проекту:
_____ Нікіта Вікторов

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Нікіта ВІКТОРОВ

Київ – 2022

ТЕКСТ ПРОГРАМНОГО КОДУ

Тексти програмного коду

ВЕБ-ЗАСТОСУНОК “БЛОГ”

(Найменування програми (документа))

CD-R

(Вид носія даних)

18 арк, 23400 Кб

(Обсяг програми, арк., Кб)

Київ – 2022

Startup.cs

```
using BLL.Auth;
using BLL.Interfaces;
using BLL.Sevices;
using DAL.EF;
using DAL.Interfaces;
using DAL.Repositories;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.OpenApi.Models;

namespace Blog
{
   public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      public void ConfigureServices(IServiceCollection services)
      {
         services.AddControllers();
         services.AddDbContext<BlogContext>(options =>
               options.UseSqlServer(
                  Configuration.GetConnectionString("DefaultConnection")));
         services.AddControllers().AddNewtonsoftJson(options =>
               options.SerializerSettings.ReferenceLoopHandling =
Newtonsoft.Json.ReferenceLoopHandling.Ignore);

         services.AddSwaggerGen(swagger =>
         {
            swagger.SwaggerDoc("v1", new OpenApiInfo
            {
               Version = "v1",
               Title = "Blog API",
               Description = "ASP.NET Core 5.0 Web API"
```

```
            });
            swagger.AddSecurityDefinition("Bearer", new
OpenApiSecurityScheme()
            {
               Name = "Authorization",
               Type = SecuritySchemeType.ApiKey,
               Scheme = "Bearer",
               BearerFormat = "JWT",
               In = ParameterLocation.Header,
               Description = "JWT Authorization header using the Bearer scheme.
\r\n\r\n Enter 'Bearer' [space] and then your token in the text input
below.\r\n\r\nExample: \"Bearer 12345abcdef\"",
            });
            swagger.AddSecurityRequirement(new OpenApiSecurityRequirement
            {
               {
                  new OpenApiSecurityScheme
                  {
                     Reference = new OpenApiReference
                     {
                        Type = ReferenceType.SecurityScheme,
                        Id = "Bearer"
                     }
                  },
                  new string[] {}
               }
            });
         });
         services.AddScoped<IUnitOfWork, EFUnitOfWork>();
         services.AddTransient<IArticleRepository, ArticleRepository>();
         services.AddTransient<ICommentRepository, CommentRepository>();
         services.AddTransient<ITagRepository, TagRepository>();
         services.AddTransient<IUserRepository, UserRepository>();
         services.AddTransient<IArticleService, ArticleService>();
         services.AddTransient<ICommentService, CommentService>();
         services.AddTransient<ITagService, TagService>();
         services.AddTransient<IUserService, UserService>();
         services.AddTransient<IPasswordHasher, PasswordHasher>();
         var authOptionsConfiguration = Configuration.GetSection("Auth");
         var authOptions = Configuration.GetSection("Auth").Get<AuthOptions>();
         services.Configure<AuthOptions>(authOptionsConfiguration);
         services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
            .AddJwtBearer(options =>
            {
               options.RequireHttpsMetadata = false;
```

```
                options.TokenValidationParameters = new
Microsoft.IdentityModel.Tokens.TokenValidationParameters
                {
                    ValidateIssuer = true,
                    ValidIssuer = authOptions.Issuer,

                    ValidateAudience = true,
                    ValidAudience = authOptions.Audience,

                    ValidateLifetime = true,

                    IssuerSigningKey = authOptions.GetSymmetricSecurityKey(),
                    ValidateIssuerSigningKey = true,
                };
            });
            services.AddCors(options =>
            {
                options.AddDefaultPolicy(
                    builder =>
                    {
                        builder.AllowAnyOrigin()
                            .AllowAnyMethod()
                            .AllowAnyHeader();
                    });
            });
            services.AddControllersWithViews();
            services.AddAuthorization();
        }
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseCors(options =>
options.WithOrigins("http://localhost:4200").AllowAnyMethod().AllowAnyHeade
r());

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseSwagger();
                app.UseSwaggerUI(c =>
c.SwaggerEndpoint("/swagger/v1/swagger.json", "Blog v1"));
            }

            app.UseRouting();

            app.UseAuthentication();
```

```
            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

AuthController.cs

```
using BLL.Auth;
using BLL.DTOs;
using BLL.Interfaces;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace Blog.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private readonly IOptions<AuthOptions> _authOptions;
        private readonly IPasswordHasher _passwordHasher;
        private readonly IUserService _userService;
        public AuthController(IUserService userService, IOptions<AuthOptions>
authOptions, IPasswordHasher passwordHasher)
        {
            _userService = userService;
            _authOptions = authOptions;
            _passwordHasher = passwordHasher;
        }

        [Route("login")]
        [HttpPost]
        public async Task<IActionResult> Login([FromBody] Login request)
```

```
        {

            var user = await AuthenticateUser(request.Email, request.Password);
            if (user != null)
            {
                var token = GenerateJWT(user);
                return Ok(new
                {
                    access_token = token
                });
            }

            return Unauthorized();
        }

        private async Task<UserDTO> AuthenticateUser(string email, string
password)
        {
            IEnumerable<UserDTO> users = await _userService.GetAll();
            var allUsers = users.ToList();
            //return allUsers.SingleOrDefault(u => u.Email == email && u.Password
== password);
            var user = allUsers.SingleOrDefault(u => u.Email == email);
            if(user != null &&
_passwordHasher.VerifyHashedPassword(user.Password, password))
            {
                return user;
            }

            return null;
        }

        private string GenerateJWT(UserDTO userDTO)
        {
            var authParams = _authOptions.Value;

            var securityKey = authParams.GetSymmetricSecurityKey();
            var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

            var claims = new List<Claim>()
            {
                new Claim(JwtRegisteredClaimNames.Email,userDTO.Email),
                new Claim(JwtRegisteredClaimNames.Sub,userDTO.Id.ToString())
            };
```

```
            claims.Add(new Claim("Password", userDTO.Password));
            claims.Add(new Claim("Name", userDTO.Name));
            claims.Add(new Claim("Surname", userDTO.Surname));
            claims.Add(new Claim(ClaimsIdentity.DefaultRoleClaimType,
userDTO.Role.ToString()));

            var token = new JwtSecurityToken(authParams.Issuer,
                authParams.Audience,
                claims,
                expires: DateTime.Now.AddSeconds(authParams.TokenLifeTime),
                signingCredentials: credentials);

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

ArticleController.cs

```
using BLL;
using BLL.DTOs;
using BLL.Exceptions;
using BLL.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace Blog.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ArticleController : ControllerBase
    {
        private readonly IArticleService _articleService;
        private Guid UserId => Guid.Parse(User.Claims.Single(c => c.Type ==
ClaimTypes.NameIdentifier).Value);
        public ArticleController(IArticleService articleService)
        {
            _articleService = articleService;
        }
```

```
[HttpGet]
public async Task<IActionResult> GetArticles()
{
    try
    {
        return Ok(await _articleService.GetAll());
    }
    catch (ArticleException ex)
    {
        return BadRequest(ex.Message);
    }
}

[Route("{id}")]
[HttpGet]
public async Task<IActionResult> GetArticle([FromRoute] Guid id)
{
    try
    {
        return Ok(await _articleService.Get(id));
    }
    catch (ArticleException ex)
    {
        return NotFound(ex.Message);
    }
}

[Route("GetArticlesByTag/{id}")]
[HttpGet]
public async Task<IActionResult> GetArticlesByTag([FromRoute] Guid id)
{
    try
    {
        return Ok(await _articleService.GetArticlesByTag(id));
    }
    catch(ArticleException ex)
    {
        return NotFound(ex.Message);
    }
}

[Route("GetUserArticles")]
[Authorize(Roles = "Admin")]
[HttpGet]
public async Task<IActionResult> GetUserArticles()
```

```
        {
            try
            {
                return Ok(await _articleService.GetUserArticles(UserId));
            }
            catch(ArticleException ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [Route("CreateArticle/{userIdArt}")]
        [HttpPost]
        public async Task<IActionResult> CreateArticle([FromBody] ArticleDTO
articleDTO, [FromRoute]string userIdArt)
        {
            try
            {
                articleDTO.UserId = Guid.Parse(userIdArt);

                await _articleService.Create(articleDTO);

                return Ok();
            }
            catch (ArticleException ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [Route("AddTag/{ArticleId}")]
        [HttpPost]
        public async Task<IActionResult> AddTag([FromRoute] Guid
ArticleId,[FromBody] TagDTO tagDTO)
        {
            try
            {
                await _articleService.AddTag(ArticleId, tagDTO);

                return Ok();
            }
            catch(ArticleException ex)
            {
                return BadRequest(ex.Message);
            }
```

```
        }

        [HttpPut("EditArticle/{id}/{userIdArt}")]
        public async Task<IActionResult> UpdateArticle([FromRoute] string id,
    [FromRoute] string userIdArt,[FromBody] ArticleDTO articleDTO)
        {
            try
            {
                articleDTO.UserId = Guid.Parse(userIdArt);

                await _articleService.Update(Guid.Parse(id), articleDTO);

                return Ok();
            }
            catch (ArticleException ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [Route("DeleteArticle/{id}")]
        [HttpDelete]
        public async Task<IActionResult> DeleteArticle([FromRoute] Guid id)
        {
            try
            {
                await _articleService.Delete(id);

                return Ok();
            }
            catch (Exception ex)
            {
                return Problem(ex.Message);
            }
        }

    }
}
```

CommentController.cs

```
using BLL.DTOs;
using BLL.Exceptions;
using BLL.Interfaces;
using Microsoft.AspNetCore.Mvc;
```

```
using System;
using System.Threading.Tasks;

namespace Blog.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CommentController : ControllerBase
    {
        private readonly ICommentService _commentService;
        public CommentController(ICommentService commentService)
        {
            _commentService = commentService;
        }

        [HttpGet]
        public async Task<IActionResult> GetComments()
        {
            try
            {
                return Ok(await _commentService.GetAll());
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }

        [HttpGet("{id}")]
        public async Task<IActionResult> GetComment([FromRoute] Guid id)
        {
            try
            {
                return Ok(await _commentService.Get(id));
            }
            catch (CommentException ex)
            {
                return NotFound(ex.Message);
            }
        }

        [Route("CreateComment/{id}")]
        [HttpPost]
        public async Task<IActionResult> CreateComment([FromRoute]Guid id,
[FromBody] CommentDTO commentDTO)
```

```
        {
            try
            {
                await _commentService.Create(id, commentDTO);
                return Ok();
            }
            catch (CommentException ex)
            {
                return BadRequest(ex.Message);
            }
        }
        [Route("UpdateComment/{id}")]
        [HttpPut]
        public async Task<IActionResult> UpdateComment([FromRoute] Guid id,
[FromBody] CommentDTO commentDTO)
        {
            try
            {
                await _commentService.Update(id, commentDTO);
                return Ok();
            }
            catch (CommentException ex)
            {
                return BadRequest(ex.Message);
            }
        }
        [Route("DeleteComment/{id}")]
        [HttpDelete]
        public async Task<IActionResult> DeleteComment([FromRoute] Guid id)
        {
            try
            {
                await _commentService.Delete(id);
                return Ok();
            }
            catch (CommentException ex)
            {
                return BadRequest(ex.Message);
            }
        }

    }
}
```

TagController.ca

```
using BLL.DTOs;
using BLL.Exceptions;
using BLL.Interfaces;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Threading.Tasks;

namespace Blog.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TagController : ControllerBase
    {
        private readonly ITagService _tagService;
        public TagController(ITagService tagService)
        {
            _tagService = tagService;
        }

        [Route("GetTag/{id}")]
        [HttpGet]
        public async Task<IActionResult> GetTag([FromRoute] Guid id)
        {
            try
            {
                return Ok(await _tagService.Get(id));
            }
            catch (TagException ex)
            {
                return NotFound(ex.Message);
            }
        }

        [HttpGet]
        public async Task<IActionResult> GetTags()
        {
            try
            {
                return Ok(await _tagService.GetAll());
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
```

```
        }
    }

    [Route("CreateTag")]
    [HttpPost]
    public async Task<IActionResult> CreateTag([FromBody] TagDTO
tagDTO)
    {
        try
        {
            await _tagService.Create(tagDTO);
            return Ok();
        }
        catch (TagException ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [Route("UpdateTag/{id}")]
    [HttpPut]
    public async Task<IActionResult> UpdateTag([FromRoute] Guid id,
[FromBody] TagDTO tagDTO)
    {
        try
        {
            await _tagService.Update(id, tagDTO);

            return Ok();
        }
        catch (TagException ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [Route("DeleteTag/{id}")]
    [HttpDelete]
    public async Task<IActionResult> DeleteTag([FromRoute]Guid id)
    {
        try
        {
            await _tagService.Delete(id);

            return Ok();
```

```
            }
            catch (TagException ex)
            {
                return BadRequest(ex.Message);
            }
        }
    }
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule ,ReactiveFormsModule}   from '@angular/forms';
import { AppComponent } from './app.component';
import { UserComponent } from './user/user.component';
import { UserFormComponent } from './user/user-form/user-form.component';
import { UserShowComponent } from './user/user-show/user-show.component';
import { Routes, RouterModule } from '@angular/router';
import { HttpClientModule, HttpClient } from '@angular/common/http';
import { UserService } from './shared/user.service';
import { JwtModule } from '@auth0/angular-jwt';
import { JwtHelperService } from '@auth0/angular-jwt';

import { Guid } from "guid-typescript";
import { LoginComponent } from './login/login.component';
import { HomeComponent } from './home/home.component';
import { from } from 'rxjs';
import { AuthGuard } from './guards/auth-guard.service';
import { NavComponent } from './nav/nav.component';
import { ArticleComponent } from './article/article.component';
import { TagComponent } from './tag/tag.component';
import { TagService } from './shared/tag.service';
import { CommentService } from './shared/comment.service';
import { ArticleService } from './shared/article.service';
import { ArticlesShowComponent } from './article/articles-show/articles-
show.component';
import { ArticleShowComponent } from './article/article-show/article-
show.component';
import { ArticlesTagComponent } from './article/articles-tag/articles-
tag.component';
import { ArticleCreateComponent } from './article/article-create/article-
create.component';
import { ErrorComponent } from './error/error.component';
import { ArticleEditComponent } from './article/article-edit/article-
edit.component';

export function tokenGetter()
{
  return localStorage.getItem("jwt");
```

```
}

const itemRoutes: Routes = [
    { path: 'login', component: LoginComponent},
    { path: 'form', component: UserFormComponent},
    { path: 'show', component: UserShowComponent,canActivate: [AuthGuard]},
    { path: '', component: ArticlesShowComponent },
    { path: 'article/:id', component: ArticleShowComponent },
   { path:  'tag/:id', component: ArticlesTagComponent },
  { path: 'create', component: ArticleCreateComponent },
  { path: 'edit/:id', component: ArticleEditComponent },
  {path:'nav',component: NavComponent},
  {path:'**',component:ErrorComponent}
];
@NgModule({
  imports: [
    BrowserModule, FormsModule, HttpClientModule, ReactiveFormsModule,
ReactiveFormsModule, RouterModule.forRoot(itemRoutes),


JwtModule.forRoot({config:{tokenGetter:tokenGetter,allowedDomains:["localhost:265
61"]}})
  ],
  declarations: [
    AppComponent,
    UserComponent,
    UserFormComponent,
    UserShowComponent,
    LoginComponent,
    HomeComponent,
    NavComponent,
    ArticleComponent,
    TagComponent,
    ArticlesShowComponent,
    ArticleShowComponent,
    ArticlesTagComponent,
    ArticleCreateComponent,
    ErrorComponent,
    ArticleEditComponent
  ],
  providers:
[UserService,TagService,CommentService,ArticleService,JwtHelperService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

user-form.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Roles, User } from 'src/app/shared/user.model';
import { UserService } from 'src/app/shared/user.service'
import{FormBuilder,FormGroup} from '@angular/forms'
import { JwtHelperService } from '@auth0/angular-jwt';


@Component({
  selector: 'app-user-form',
  templateUrl: './user-form.component.html',
  styleUrls: [ './user-form.component.css'
  ]
})
export class UserFormComponent implements OnInit {
  user: User = new User(); // данные вводимого пользователя
  form: FormGroup;
  receivedUser: User | undefined; // полученный пользователь
  done: boolean = false;
  token: string | any = localStorage.getItem("jwt");
  helper = new JwtHelperService();
  decodedToken = this.helper.decodeToken(this.token);
  isAdministrator: boolean;
  constructor(public service:UserService) { }

  ngOnInit(): void {
  }

  public AddUser(user : User)
  {
    if (user.role == null)
      user.role = Roles.User;
    this.service.postUser(user).subscribe((data: any) => { this.receivedUser =
data; this.done = true; }, err => { console.log(err); });
  }
  isUserAuthenticated()
  {
    if (this.token) {
      return true;
    }
    else {
      return false;
    }
  }
  isAdmin()
  {
    let decodedRole =
this.decodedToken['http://schemas.microsoft.com/ws/2008/06/identity/claims/role']
    if (decodedRole == 'Admin')
```

```
      this.isAdministrator = true;
    else
      this.isAdministrator = false;

  }
}
```

nav.component.ts

```
import { Component, OnInit } from '@angular/core';
import { JwtHelperService } from '@auth0/angular-jwt';

@Component({
  selector: 'app-nav',
  templateUrl: './nav.component.html',
  styles: [
  ]
})
export class NavComponent implements OnInit {

  token: string | any = localStorage.getItem("jwt");
  helper = new JwtHelperService();
  decodedToken = this.helper.decodeToken(this.token);
  isAdministrator: boolean;
  name: string;
  surname: string;
  constructor() { }

  ngOnInit(): void {

  }
  isUserAuthenticated()
  {
    const token: string | any = localStorage.getItem("jwt");
    if (token) {
      const token: string | any = localStorage.getItem("jwt");
    const helper = new JwtHelperService();
    const decodedToken = helper.decodeToken(token);
      this.name = decodedToken['Name'];
      this.surname = decodedToken['Surname']
      return true;
    }
    else {
      return false;
    }
  }

  isAdmin()
  {
    const token: string | any = localStorage.getItem("jwt");
    const helper = new JwtHelperService();
```

```
    const decodedToken = helper.decodeToken(token);
    let decodedRole =
decodedToken['http://schemas.microsoft.com/ws/2008/06/identity/claims/role']
    if (decodedRole == null)
      return false;
    if (decodedRole == 'Admin')
      return true;
    else
      return false;
  }

  logOut()
  {
    localStorage.removeItem("jwt");
  }
}
```

login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient,HttpHeaders } from '@angular/common/http'
import { Router } from "@angular/router"
import {NgForm} from '@angular/forms'
import { Observable } from 'rxjs';


@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styles: [
  ]
})
export class LoginComponent implements OnInit {

  invalidLogin: boolean;
  constructor(private router:Router,private http:HttpClient) { }

  login(form: NgForm): Observable<any>{
    const credntials = {
      email: form.value.email,
      password:form.value.password
    }
    return this.http.post('http://localhost:26561/api/Auth/login',credntials)
  }
  reboot()
  {
    window.location.reload();
  }
  l(form: NgForm)
  {
```

```
    this.login(form).subscribe(responce => {
      const token = (<any>responce).access_token;
      localStorage.setItem("jwt", token);
      this.invalidLogin = false;
      this.router.navigate(["/"]);
    },err => {this.invalidLogin = true})
  }
  ngOnInit(): void {
  }
}
```

articles-tag-component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Guid } from 'guid-typescript';
import { Article } from 'src/app/shared/article.model';
import { ArticleService } from 'src/app/shared/article.service';

@Component({
  selector: 'app-articles-tag',
  templateUrl: './articles-tag.component.html',
  styleUrls: ['./articles-tag.component.css']
})
export class ArticlesTagComponent implements OnInit {

  articles: Article[] = [];
  id: Guid;
  constructor(private route: ActivatedRoute,public service: ArticleService) { }

  ngOnInit(): void {
    this.id = this.route.snapshot.params['id'];
    this.service.getArticlesByTag(this.id).subscribe((data: Article[]) =>
this.articles = data);
  }
  Reload()
  {
    this.id = this.route.snapshot.params['id'];
    this.service.getArticlesByTag(this.id).subscribe((data: Article[]) =>
this.articles = data);
  }
}
```

articles-show.component.ts

```
import { Comment } from '@angular/compiler';
import { Component, OnInit } from '@angular/core';
import { JwtHelperService } from '@auth0/angular-jwt';
import { Guid } from 'guid-typescript';
```

```
import { Article } from 'src/app/shared/article.model';
import { ArticleService } from 'src/app/shared/article.service';
import { Tag } from 'src/app/shared/tag.model';
import { TagService } from 'src/app/shared/tag.service';

@Component({
  selector: 'app-articles-show',
  templateUrl: './articles-show.component.html',
  styleUrls: ['./articles-show.component.css']
})
export class ArticlesShowComponent implements OnInit {

  articles: Article[] = [];
  tags: Tag[] = [];
  userId: Guid;
  roleUser: string;
  token: string | any = localStorage.getItem("jwt");
  helper = new JwtHelperService();
  decodedToken = this.helper.decodeToken(this.token);
  isAdmin: boolean;
  sortArticles: Article[] = [];
  constructor(public service: ArticleService, public tagService: TagService) {
  }



  ngOnInit(): void {
    this.service.getArticles().subscribe((data: Article[]) => this.articles =
data);
    this.tagService.getTags().subscribe((t: Tag[]) => this.tags = t);
    this.isUserRoleAdmin();
    // this.initializeEventListeners();

  }

  ngDoCheck(): void{
    this.sortByTag(this.articles)
  }

  initializeEventListeners() {
    document.addEventListener('click', event => {
      if((event.target as HTMLElement).closest('.three-dots')) {
      alert(123);

      }
    })
  }

  sortByTag(articles: Article[])
  {
    this.sortArticles = articles
```

```
    for(var i = 0, endI = this.sortArticles.length - 1; i < endI; i++)
    {
      for (var j = 0, endJ = endI - i; j < endJ; j++)
      {
        if (this.sortArticles[j].tags.length > this.sortArticles[j +
1].tags.length)
        {
          var swap = this.sortArticles[j];
          this.sortArticles[j] = this.sortArticles[j + 1];
          this.sortArticles[j + 1] = swap;
        }
      }
    }

    this.sortArticles.reverse();

    // for(let i = 0; i < this.articles.length; i++)
    // {

    // }

    // for(let article of this.articles)
    // {

    // }
    // this.sortArticles = this.articles.sort(function (a,b) {
    //   if(a.tags.length < b.tags.length)
    //   {
    //     return 1;
    //   }
    //   if(a.tags.length > b.tags.length)
    //   {
    //     return -1;
    //   }
    //   return 0;
    // });
  }
  isUserRoleAdmin()
  {
    let decodedRole =
this.decodedToken['http://schemas.microsoft.com/ws/2008/06/identity/claims/role']
    if (decodedRole == 'Admin')
      this.isAdmin = true;
    else
      this.isAdmin = false;
  }
  isAccess(article:Article)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    if (article.userId == this.userId)
```

```
      return true;
    else
      return false;
  }
  isUserAuthenticated()
  {
    const token: string | any = localStorage.getItem("jwt");
    if (token) {
      return true;
    }
    else {
      return false;
    }
  }
}
```

article-show.component.ts

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { FormGroup } from '@angular/forms';
import { ActivatedRoute } from '@angular/router';
import { JwtHelperService } from '@auth0/angular-jwt';
import { Guid } from 'guid-typescript';
import { Article } from 'src/app/shared/article.model';
import { ArticleService } from 'src/app/shared/article.service';
import { Comment } from 'src/app/shared/comment.model';
import { CommentService } from 'src/app/shared/comment.service'


@Component({
  selector: 'app-article-show',
  templateUrl: './article-show.component.html',
  styleUrls: ['./article-show.component.css']
})
export class ArticleShowComponent implements OnInit {

  readonly baseURL = 'http://localhost:26561/api/Article';
  article: Article;
  id: Guid;
  commentId: Guid;
  token: string | any = localStorage.getItem("jwt");
  helper = new JwtHelperService();
  decodedToken = this.helper.decodeToken(this.token);
  comment: Comment = new Comment();
  userId: Guid;
  form: FormGroup;
  receivedComment: Comment | undefined;
  done: boolean = false;
```

```
  d: boolean = false;
  isAdmin: boolean;
  constructor(private route: ActivatedRoute, private service:
ArticleService,private http:HttpClient,public commentService : CommentService)
  {
  }

  ngOnInit(): void {

    this.id = this.route.snapshot.params['id']

    this.service.getArticle(this.id).subscribe((data: Article) => this.article =
data)
    this.isUserRoleAdmin();
  }

  public AddComment(comment: Comment)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    comment.userId = this.userId;
    this.commentService.createComment(this.id, comment).subscribe((data: any) =>
{ this.receivedComment = data; this.done = true; this.ngOnInit();
this.comment.text = ""}, err => { console.log(err); })
  }


isUserRoleAdmin()
  {
    let decodedRole =
this.decodedToken['http://schemas.microsoft.com/ws/2008/06/identity/claims/role']
    if (decodedRole == 'Admin')
      this.isAdmin = true;
    else
      this.isAdmin = false;
  }

  isAccess(article:Article)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    if (article.userId == this.userId)
      return true;
    else
      return false;
  }

  isUserArticle(article:Article)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    if (article.userId == this.userId)
      return true;
```

```
    else
      return false;
  }

  isUserComment(comment:Comment)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    if (comment.userId == this.userId)
      return true;
    else
      return false;
  }
  isUserAuthenticated()
  {
    const token: string | any = localStorage.getItem("jwt");
    if (token) {
      return true;
    }
    else {
      return false;
    }
  }
  onEdit(id:Guid)
  {
    this.comment.id = id;
    this.userId = Guid.parse(this.decodedToken.sub);
    this.comment.userId = this.userId;
    this.commentService.updateComment(this.comment.id,
this.comment).subscribe((data: any) => { this.receivedComment = data; this.done =
true; this.service.getArticle(this.id).subscribe((data: Article) => this.article
= data); this.comment.text = "" }, err => { console.log(err); })

  }

  onArticle()
  {
  }

}
```

article-edit.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute } from '@angular/router';
import { JwtHelperService } from '@auth0/angular-jwt';
import { Guid } from 'guid-typescript';
import { Article } from 'src/app/shared/article.model';
import { ArticleService } from 'src/app/shared/article.service';
```

```
import { Tag } from 'src/app/shared/tag.model';

@Component({
  selector: 'app-article-edit',
  templateUrl: './article-edit.component.html',
  styleUrls: ['./article-edit.component.css']
})
export class ArticleEditComponent implements OnInit {

  constructor(private route: ActivatedRoute, public service: ArticleService) { }

  article: Article = new Article();
  token: string | any = localStorage.getItem("jwt");
  form: FormGroup;
  tagsText: string;
  helper = new JwtHelperService();
  receivedArticle: Article | undefined; // полученный пользователь
  done: boolean = false;
  userId: Guid;
  id: Guid;
  isAdmin: boolean;


isUserRoleAdmin()
  {
    let decodedRole =
this.decodedToken['http://schemas.microsoft.com/ws/2008/06/identity/claims/role']
    if (decodedRole == 'Admin')
      this.isAdmin = true;
    else
      this.isAdmin = false;

  }

  decodedToken = this.helper.decodeToken(this.token);


  ngOnInit(): void {

    this.id = this.route.snapshot.params['id']
    this.tagsText = "";
    console.log(this.userId);
  }

  editArticle(article: Article)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    article.userId = this.userId;
    article.id = this.id;
    var str = this.tagsText;
    var splitted = str.split(new RegExp('\#', 'g'));
```

```
    let listTmp: Tag[] = [];

    splitted.forEach( function(item, index) {
      if (item != undefined && item != null && item != "") {
        let tagTmp: Tag = new Tag();

        tagTmp.text = "#" + item;
        listTmp.push(tagTmp);

      }
    });

    article.tags = listTmp;

    this.service.EditArticle(article).subscribe((data: any) => {
      this.receivedArticle = data; this.done = true; window.location.reload();
    }, err => { console.log(err); });
  }
}
```

article-create.component.ts

```
import { Component, OnInit } from '@angular/core';
import { JwtHelperService } from '@auth0/angular-jwt';
import { Guid } from "guid-typescript";
import { FormGroup } from "@angular/forms";
import { Article } from "src/app/shared/article.model";
import { ArticleService } from "src/app/shared/article.service";
import { Tag } from "src/app/shared/tag.model";

@Component({
  selector: 'app-article-create',
  templateUrl: './article-create.component.html',
  styleUrls: ['./article-create.component.css']
})
export class ArticleCreateComponent implements OnInit {

  constructor(public service:ArticleService) { }
  article: Article = new Article();
  token: string | any = localStorage.getItem("jwt");
  form: FormGroup;
  tagsText: string;
  helper = new JwtHelperService();
  receivedArticle: Article | undefined; // полученная статья
  done: boolean = false;
  userId: Guid;
```

```
  decodedToken = this.helper.decodeToken(this.token);


  ngOnInit(): void {
    this.tagsText = "";

    console.log(this.userId);
  }
  public AddArticlee(article:Article)
  {
    this.userId = Guid.parse(this.decodedToken.sub);
    article.userId = this.userId;
    var str = this.tagsText;

    var splitted = str.split(new RegExp('\#', 'g'));

    let listTmp: Tag[] = [];

    splitted.forEach( function(item, index) {
      if (item != undefined && item != null && item != "") {
        let tagTmp: Tag = new Tag();

        tagTmp.text = "#" + item;
        listTmp.push(tagTmp);

      }
    });

    article.tags = listTmp;
    this.service.AddArticle(article,article.userId).subscribe((data: any)
=> {
      this.receivedArticle = data; this.done = true;
window.location.reload();
    }, err => { console.log(err); });


  }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ВЕБ-ЗАСТОСУНОК “БЛОГ”

Програма та методика тестування

КПІ.ІТ-8305.045440.02.83

“ПОГОДЖЕНО”
Керівник проекту:
_____ Юлія Крамар

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Нікіта ВІКТОРОВ

Київ – 2022

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробувань є веб-застосунок “Блог”, створене за допомогою мови програмування C#.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення у відповідно до функціональних вимог;

– знаходження проблем та недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється уся документація, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на безлічі тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми в роботі програми;

– тестування «білої скриньки» – об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується вручну, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності в користуванні. Для того, щоб перевірити працездатність та відмовостійкість додатку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на мобільних пристроях з різною роздільною здатністю екрану;

– тестування зміни орієнтації екрану;

– тестування працездатності програми у випадку відсутності з’єднання до мережі;

– тестування інтерфейсу користувача;

– тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ВЕБ-ЗАСТОСУНОК “БЛОГ”

Керівництво користувача

КПІ.ІП-8305.045440.02.83

“ПОГОДЖЕНО”
Керівник проекту:
_____ Юлія Крамар

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Нікіта ВІКТОРОВ

Київ – 2022

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Веб-застосунок “Блог” – веб-застосунок, дозволяє вести блог, щоденник без необхідності самостійно займатися обслуговуванням та програмуванням двигуна.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність доступу до Інтернету;

– для встановлення додатку на мобільному пристрої повинно бути не менше 70 МБ вільної пам’яті.

- Наявність .CLR.

2.2 Завантаження застосунку

На даний момент застосунок доступний тільки на локальному сервері, Для цього треба запустити сервер на локальній машині та ввести url в рядок.

2.3 Перевірка коректної роботи

По завершенню запуска додатка у веб-браузері повинна відобразитись застосунок. У разі, якщо застосунок не з’явився, то запуск відбувся не успішно. Інакше користувач має змогу запустити знову. Після цього повинна відобразитись початкова сторінка застосунка.

3 РОБОТА ІЗ ЗАСТОСУНКОМ

Для того, щоб почати роботу із нашим сервісом, необхідно запустити веб версію у браузері. Посилання буде виведене у консолі після старту серверу

Рисунок 3.1 – Посилання на веб-версію

Сторінка логування потрібна для того, щоб дізнатися чи є користувач у системі, якщо то надавати йому відповідні права для користування, якщо користувач він не може нічого робити окрім переглядання статей

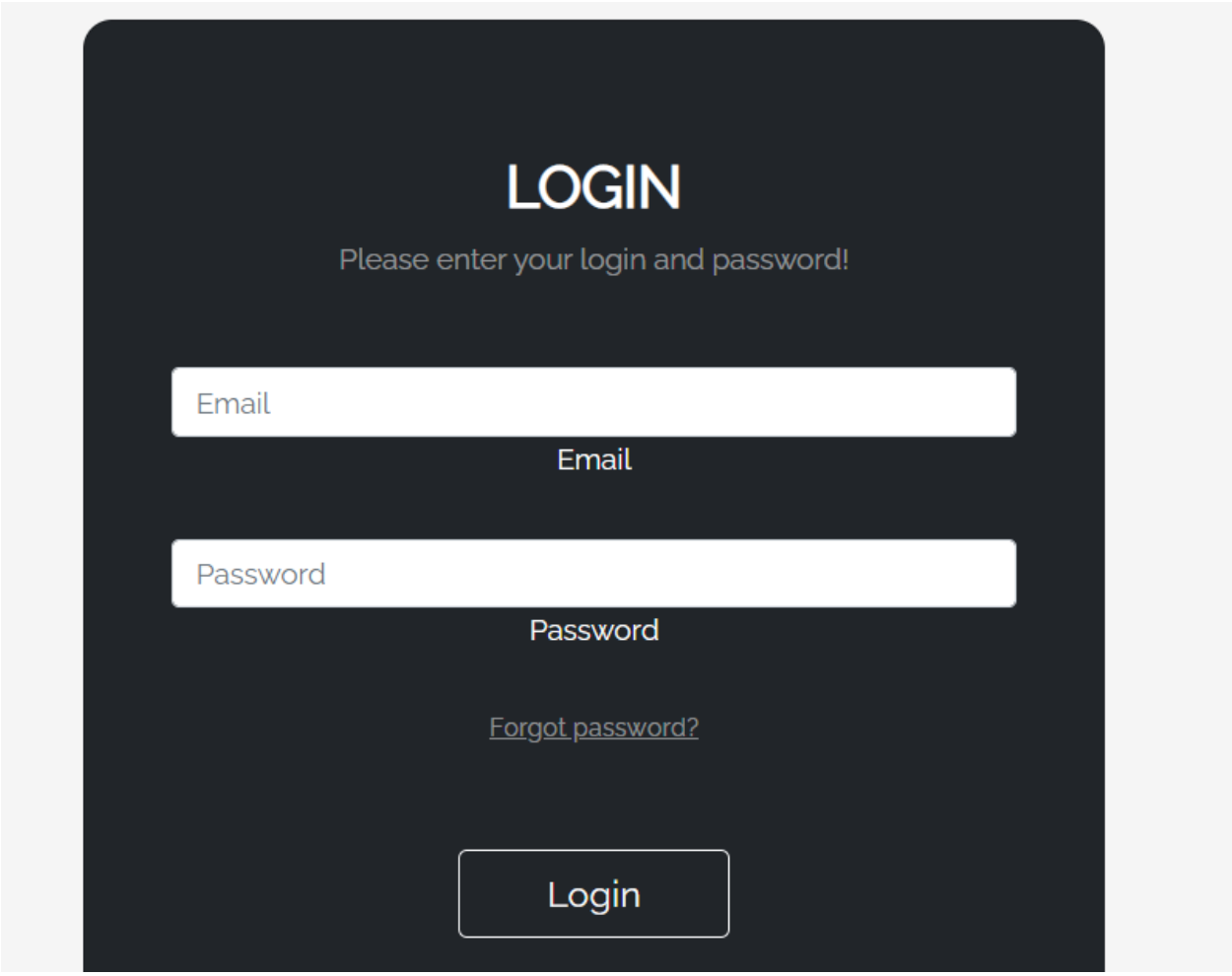


Рисунок 3.2 – Сторінка логування

Після успішного логування користувач має можливість переглядати статті, та користуватись застосункам. Було це зроблено для будь-якого розміру екрана.

Це було для більш зручного читання та користування нашого програмного застосунка. Це надає багато можливостей.

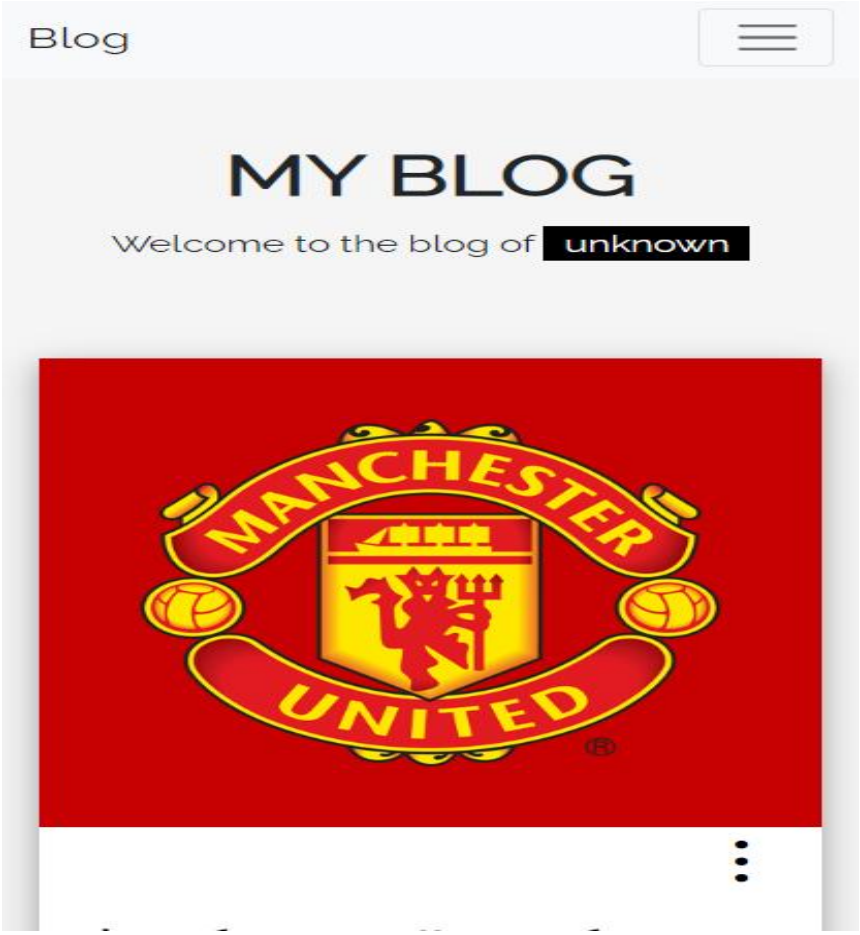


Рисунок 3.3 – Сторінка з усіма статтями для мобільного телефону

У застосунку є можливість переглядати статті за тегами. Це було зроблено, щоб користувач мав можливість переглядати тільки те, що йому цікаве.

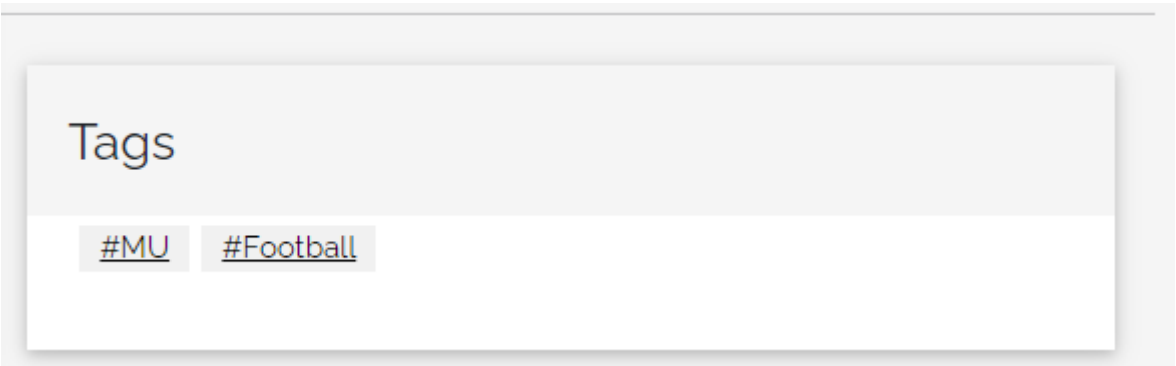


Рисунок 3.4 - Теги

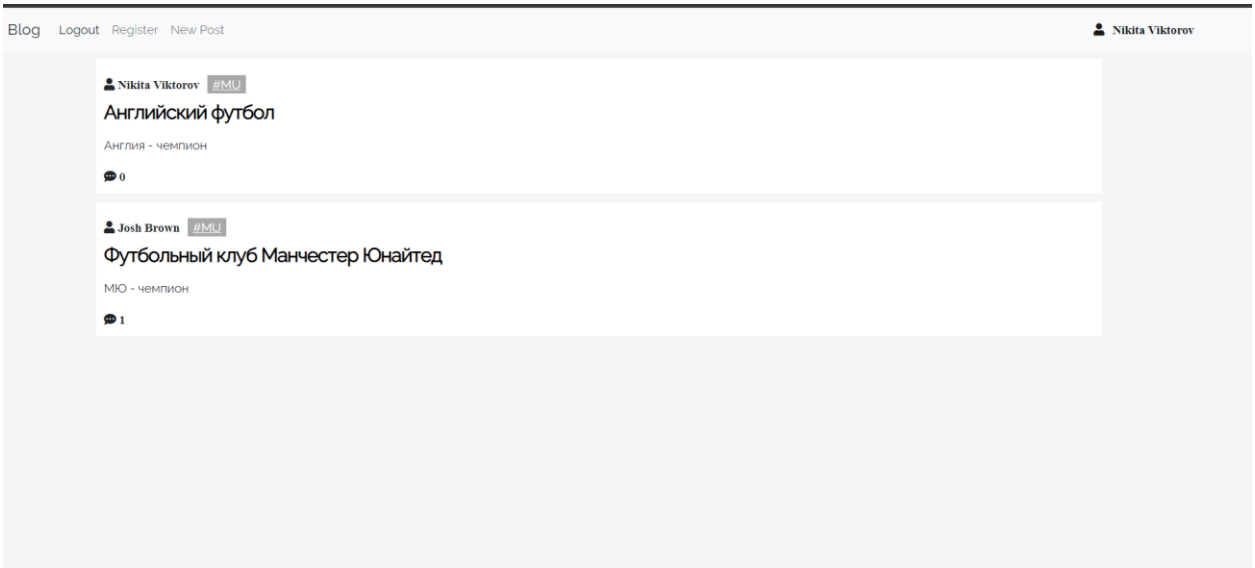


Рисунок 3.5 – статті за тегом MU

Також є можливість реєструватись. Це було зроблено, щоб користувач мав ділитись інформацію,коментувати та створювати статті.

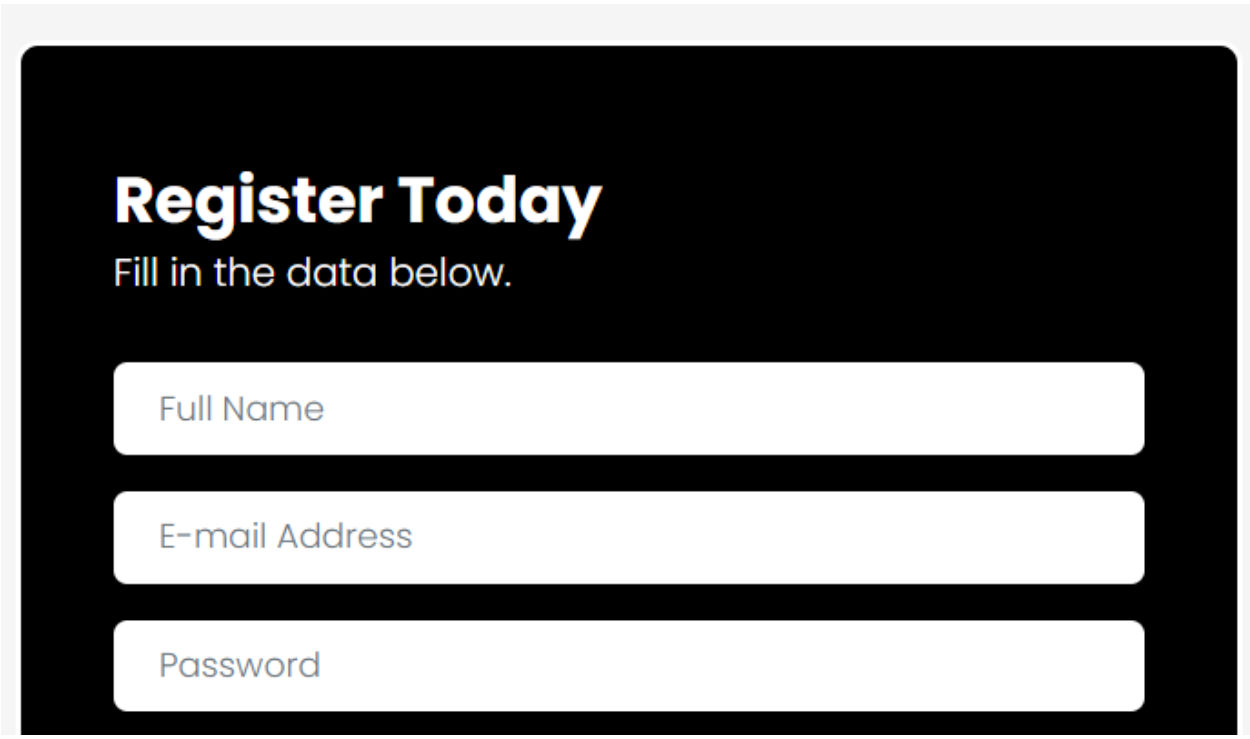


Рисунок 3.6 – сторінка реєстрування

Якщо користувач має намір відредагувати статтю, він зможе це зробити. Він може редагувати тільки свою статтю та коментар.

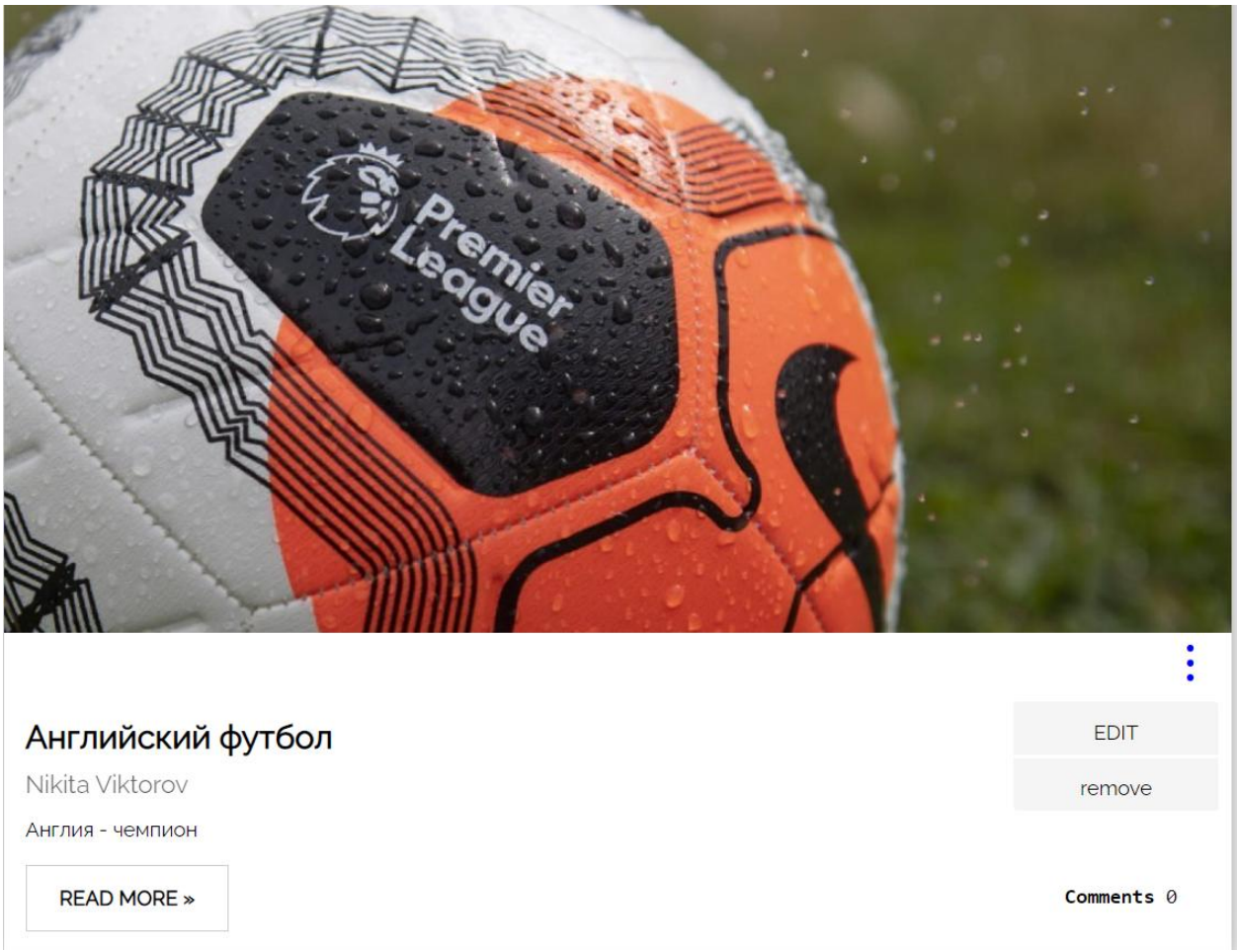


Рисунок 3.7 – Можливість редагування

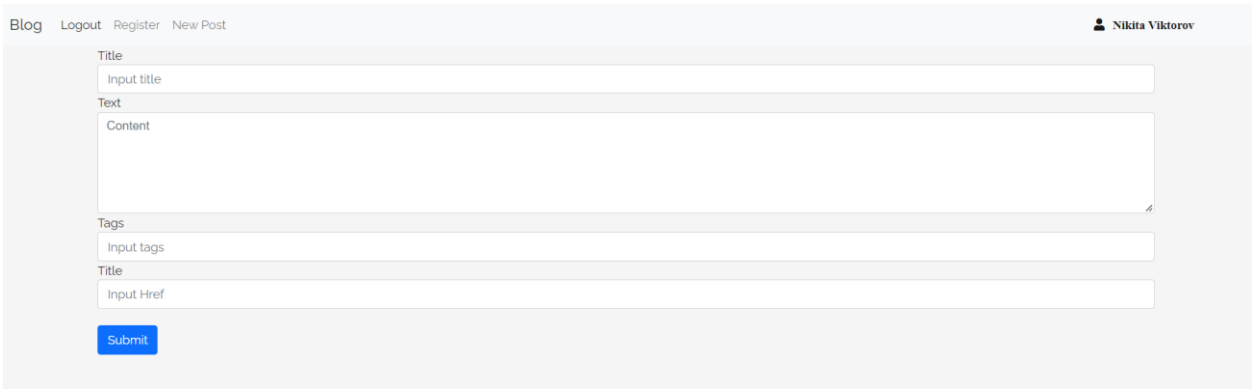


Рисунок 3.8 – Сторінка редагування

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

ВЕБ-ЗАСТОСУНОК “ БЛОГ ”

Графічний матеріал

КПІ.ІТ-8305.045440.02.83

“ПОГОДЖЕНО”
Керівник проекту:
_____ Юлія КРАМАР

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Нікіта ВІКТОРОВ

Київ – 2022

КПІ.ІТ-8305.045440.02.05.СБД

					КПІ.ІТ-8305.045440.02.05.СБД	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	
Розробив		Вікторов Н.С				Аркуш / Аркушів
Перевірив		Крамар Ю.М				
Т. кон.						
Н. кон.		Ліщук К.І.			Веб-застосунок "Блог"	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-83
Затвердив		Крамар Ю.М				

КПІ.ІТ-8305.045440.02.05.СС

КПІ.ІТ-8305.045440.02.05.СС

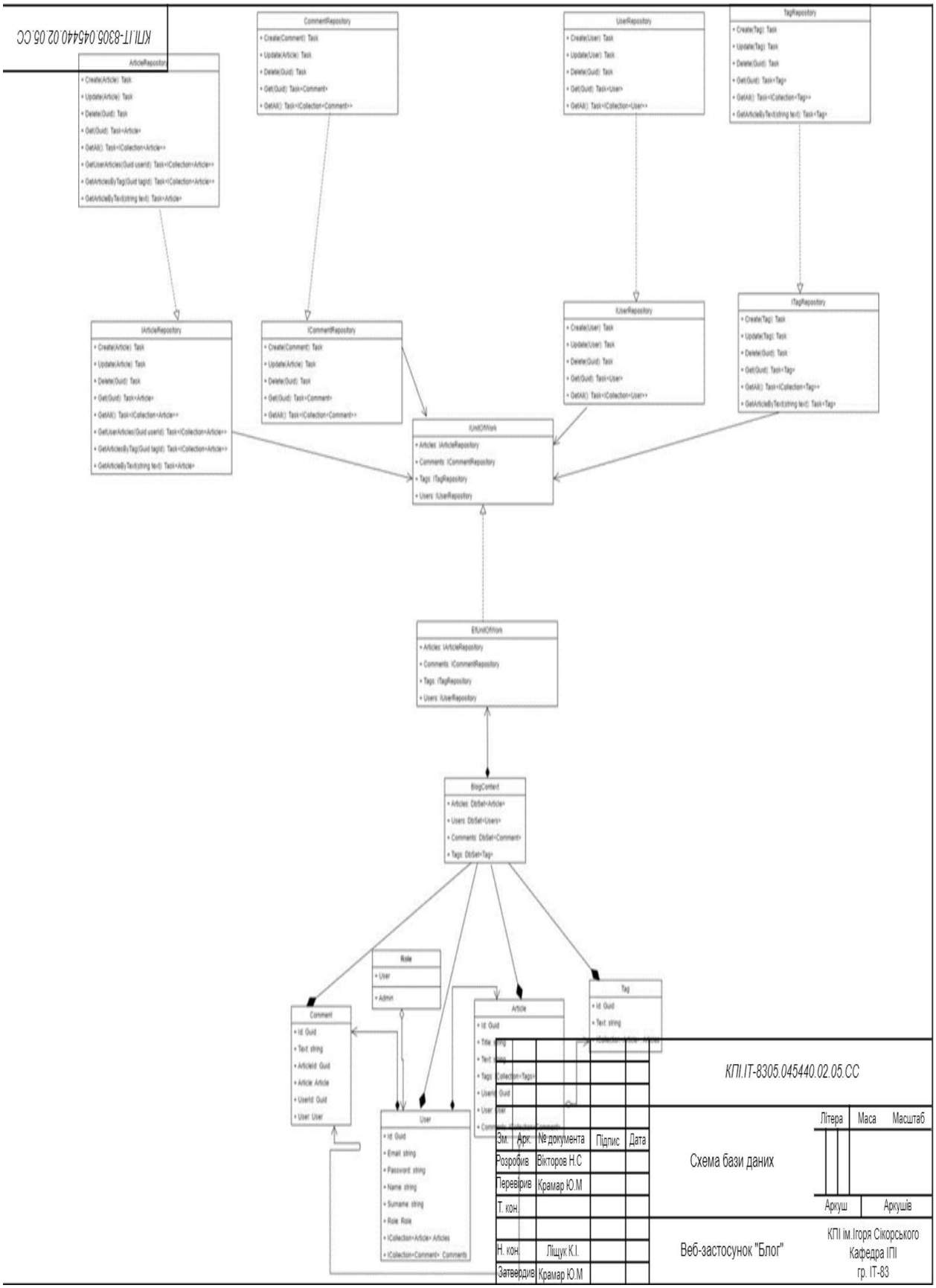
Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Вікторов Н.С		
Перевірив		Крамар Ю.М		
Т. кон.				
Н. кон.		Ліщук К.І.		
Затвердив		Крамар Ю.М		

Схема бази даних

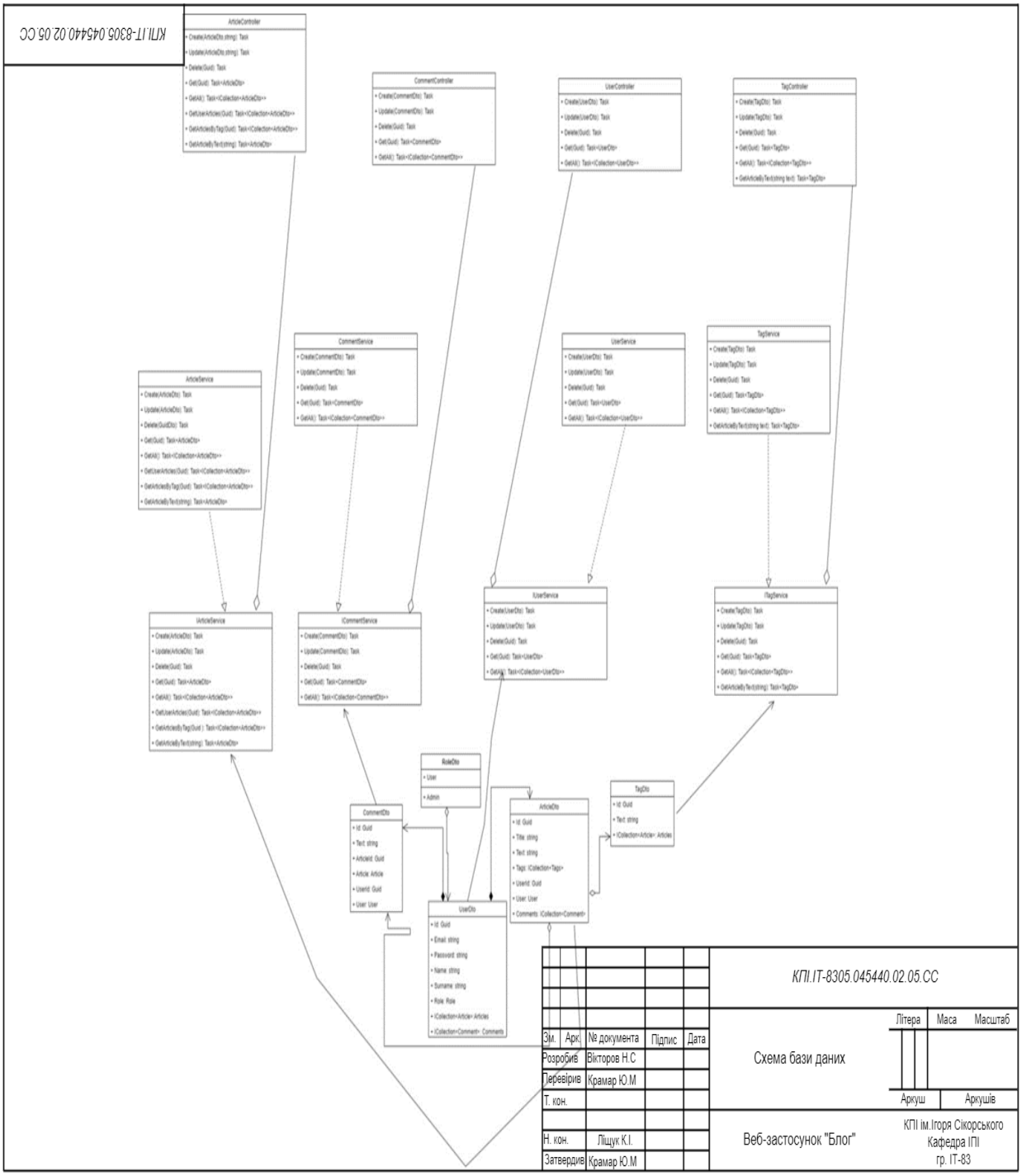
Літера	Маса	Масштаб
Аркуш	Аркушів	

Веб-застосунок "Блог"

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-83



					КПІ.ІТ-8305.045440.02.05.СС	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Вікторов Н.С			Схема бази даних	Літера: — Маса: — Масштаб: —
Перевірив		Крамар Ю.М				Аркуш: — Аркушів: —
Т. кон.						
Н. кон.		Ліщук К.І.			Веб-застосунок "Блог"	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-83
Затвердив		Крамар Ю.М				



					КПІ.ІТ-8305.045440.02.05.СС		
						Літера	Маса Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних		
Розробив		Вікторов Н.С					
Перевірив		Крамар Ю.М				Аркуш	Аркушів
Т. кон.							
Н. кон.		Ліщук К.І.			Веб-застосунок "Блог"	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-83	
Затвердив		Крамар Ю.М					

UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1011600350

Дата проверки:
17.06.2022 07:29:53 EEST

Тип проверки:
Doc vs Internet + Library

Дата отчета:
17.06.2022 07:30:53 EEST

ID пользователя:
76913

Название файла: **IT-83_Вікторов_ПЗ**

Количество страниц: **65** Количество слов: **7976** Количество символов: **58315** Размер файла: **772.03 KB** ID файла: **1011468931**

Обнаружены модификации текста (могут влиять на процент совпадений)

9.06%
Совпадения

Наибольшее совпадение: **1.98%** с источником из Библиотеки (ID файла: **1003935009**)

0.33% Источники из Интернета	2	Страница 67
9.06% Источники из Библиотеки	162	Страница 67

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы	9
Подозрительное форматирование	12 страниц