UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
14.06.2020 01:06:45 EEST

Дата звіту:
14.06.2020 17:15:50 EEST

ID перевірки:
1004022414

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Vereschak_ip63

ID файлу: 1004035483 **Кількість сторінок:** 44 **Кількість слів:** 6296 **Кількість символів:** 48262 **Розмір файлу:** 439.50 KB

2.89% Схожість

Найбільша схожість: 1.06% з джерело бібліотеки. **ID файлу:** 1004035498

0.64% Схожість з Інтернет джерелами	11	Page 46
2.76% Текстові збіги по Бібліотеці акаунту	72	Page 46

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

Підміна символів

Не знайдено замінених символів

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Веб-застосування для управління та систематизації колективних знань*

Виконав: студент IV курсу, групи *ІП-63 Верещак Б.С.* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. вик. Вітковська І.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

ст. викладач кафедри ТК, к.т.н. Сирота О. П.
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___” _____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Верещаку Богдану Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Веб-застосування для управління та систематизації колективних знань*»
керівник проєкту *Вітковська Ірина Іванівна, ст. вик.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання, архітектура та конструювання програмного забезпечення: аналіз програмного забезпечення, вхідні та вихідні дані, засоби розробки, структура бази даних та класів
3) Аналіз якості та тестування програмного забезпечення
4) Розгортання та впровадження програмного забезпечення

5. Перелік графічного матеріалу

1) Схема бази даних

2) Схема структурна класів системи

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*11.03.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*15.03.2020*	
3.	*Постановка та формалізація задачі*	*24.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*01.04.2020*	
5.	*Алгоритмізація задачі*	*03.04.2020*	
6.	*Моделювання програмного забезпечення*	*07.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*15.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*25.04.2020*	
9.	*Розробка програмного забезпечення*	*1.05.2020*	
10.	*Налагодження програми*	*19.05.2020*	
11.	*Виконання графічних документів*	*26.05.2020*	
12.	*Оформлення пояснювальної записки*	*27.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*		
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Богдан Верещак
(підпис)

Керівник _____ Ірина Вітковська
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6306.045440.02.81	Пояснювальна записка	67	
3	А4	КПІ.ІП-6306.045440.03.91	Технічне завдання	9	
4	А4	КПІ.ІП-6306.045440.04.51	Програма та методика тестування	4	
5	А4	КПІ.ІП-6306.045440.05.13	Опис програми	37	
6	А4	КПІ.ІП-6306.045440.06.34	Керівництво користувача	12	
7	А3	КПІ.ІП-6306.045440.07.99.СБД	Схема бази даних	1	
8	А3	КПІ.ІП-6306.045440.08.99.СС	Схема структурна класів системи	1	
9	А3	КПІ.ІП-6306.045440.09.99.КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6306.045440.00.90		
Розробн.		Верещак Б.С.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Вітковська І.І.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосування для управління та систематизації колективних знань

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 137 сторінок, 12 рисунків, 50 таблиць, 5 додатків, 10 джерел.

Мета розробки: створення умов, при яких накопичені знання та досвід користувачів будуть ефективно використовуватись для виконання важливих для них завдань та досягнення колективних цілей. Зекономити час користувачів на пошук вирішення часто виникаючих запитань, перетворивши колективні знання у відповіді що можна буде легко знайти та переглядати у вигляді документів що матимуть різноманітні способи форматування для максимально зручного їх перегляду. Надати можливість усім учасникам колективу незалежно від їх статусу ділитись ідеями та отримувати до них зворотний зв'язок. А також підтримка існуючої інформації у актуальному стані доповнюючи або коректуючи її.

У розділі «Аналіз вимог» виконано аналіз предметної області та відомих технічних рішень, розглянуто аналоги, сформульовано функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення.

У розділі «Моделювання та конструювання» виконано моделювання програмного забезпечення. Побудовано структурні схеми активності, описана архітектура програмного забезпечення та розглянуто технології що використовувались для реалізації веб-застосунку.

У розділі «Аналіз якості та тестування» описані процеси тестування вебзастосунку та було наведено кілька контрольних прикладів.

У розділі «Впровадження та супровід» наведено покрокову інструкцію по розгортанню застосунку та побудовано структурну схему розгортання.

КЛЮЧОВІ СЛОВА: WEB-ЗАСТОСУВАННЯ, КОЛЕКТИВНІ ЗНАННЯ, УПРАВЛІННЯ ЗНАННЯМИ, СИСТЕМАТИЗАЦІЯ ЗНАНЬ

ABSTRACT

Structure and scope of work. Structure and scope of work. Explanatory note of the graduation project

consists of four sections, contains 137 pages, 12 drawings, 50 tables, 5 Applications, 10 sources.

The purpose of development: to create conditions under which the accumulated knowledge and experience of users will be effectively used to perform important tasks and achieve collective goals. Save users time searching for solutions to frequently asked questions by turning collective knowledge into answers that can be easily found and viewed in the form of documents that will have a variety of formatting methods for the most convenient viewing. Provide an opportunity for all team members, regardless of their status, to share ideas and receive feedback. As well as maintaining existing information in an up-to-date state by supplementing or correcting it.

In the section "Analysis of requirements" the analysis of subject area and known technical decisions is executed, analogues are considered, functional and nonfunctional requirements to the developed software are formulated.

In the section "Modeling and design" software modeling is performed. Activity diagrams are constructed, the software architecture is described and the technologies used for web application implementation are considered.

In the section "Analysis of quality and testing" describes the web application testing processes and provides some control examples.

In the section “Implementation and Maintenance” provided step-by-step instructions for deploying the application and constructed the deployment diagram.

KEYWORDS: WEB-APPLICATION, COLLECTIVE KNOWLEDGE, KNOWLEDGE MANAGEMENT, KNOWLEDGE SYSTEMATIZATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JWT – (JSON Web Token) – це відкритий стандарт для створення токенів доступу, заснований на форматі JSON. Як правило, використовується для передачі даних для аутентифікації в клієнт-серверних додатках. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який в подальшому використовує даний токен для підтвердження своєї особи

.NET – програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків

API – (Application Programming Interface) – опис способів (набір класів, процедур, функцій, структур або констант), якими одна комп'ютерна програма може взаємодіяти з іншою програмою

HTTP (HyperText Transfer Protocol)– протокол прикладного рівня передачі даних, спочатку - у вигляді гіпертекстових документів в форматі «HTML», зараз використовується для передачі довільних даних. Основою HTTP є технологія «клієнт-сервер»

HTTPS (HyperText Transfer Protocol Secure) – розширення протоколу HTTP для підтримки шифрування з метою підвищення безпеки

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на нотації мови JavaScript

ORM (Object-Relational Mapping) – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних»

REST (Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі

БД – База даних

ПЗ – Програмне забезпечення

ВСТУП

Часто для організацій та спільнот яких об’єднують спільні інтереси виникає необхідність використовувати певний сервіс для зберігання колективних знань (документації, інструкцій, статей) що відповідають їх інтересам, а також для пошуку вирішень різного роду питань та проблем базуючись на існуючих записах. При чому задля зручного використання інформації – доцільно мати можливість її систематизувати, тобто, розділяти за різними категоріями, а також додавати нові категорії та підкатегорії таким чином будуючи дерево. Одними із найважливіших вимог до подібних систем є надання користувачам можливості забезпечення достовірності та актуальності даних, а також можливість надавати свої відгуки або думки до уже існуючих записів.

Головна мета застосунку — створення умов, при яких накопичені знання та досвід користувачів будуть ефективно використовуватись для виконання важливих для них завдань та досягнення колективних цілей. Зекономити час користувачів на пошук вирішення часто виникаючих запитань, перетворивши колективні знання у відповіді що можна буде легко знайти та переглядати у вигляді документів що матимуть різноманітні способи форматування для максимально зручного їх перегляду. Надати можливість усім учасникам колективу незалежно від їх статусу ділитись ідеями та отримувати до них зворотний зв’язок. А також підтримка існуючої інформації у актуальному стані доповнюючи або коректуючи її.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Колективні знання — інформація в контексті певних організації чи спільнот, створена шляхом застосування особистих знань колективів.

Управління колективними знаннями — це процес створення, обміну, використання інформації що є цінною для певної організації чи спільноти. Багато великих компаній, державних установ та неприбуткових організацій мають ресурси, спрямовані на внутрішні зусилля управління знаннями, які часто є частиною їх ділової стратегії. Зусилля щодо управління знаннями зазвичай зосереджуються на таких організаційних цілях, як підвищення ефективності, конкурентна перевага, обмін отриманими уроками, інтеграція та постійне вдосконалення організації. Ці зусилля частково покриваються організаційним навчанням і відрізняються від нього тим, що більше зосереджуються на управлінні знаннями як стратегічному активі та на заохоченні обміну знаннями.

1.2 Змістовний опис і аналіз предметної області

Знання не тільки являють собою самостійну цінність, а й породжують мультиплікативний ефект по відношенню до інших факторів виробництва, впливаючи на рівень ефективності їх застосування. Знанням є поєднання оформленого досвіду, інформації і поглядів. В організаціях вони часто потрапляють не тільки в документи або сховища, а й в організаційні процедури, процеси, практику і норми.

Застосування для управління колективними знаннями дозволяє забезпечити:

– представлення знань у вигляді статей (документів), зручних для перегляду для пересічних користувачів;

– додання нових статей;

– редагування та фіксація змін статей, по яким можна відстежити всю історію діяльності;

– систаматизацію колективних знань розбиваючи їх на різні катерогії та підкатегорії;

– пошук існуючої в системі інформації;

– можливість зплишати свої думки або відгуки до існуючих статей.

Часто інформація що доступна певній групі осіб, може представляти для них велику цінність, тому подібні системи зберігають публікації таким чином що, до можуть мати доступ тільки члени певної групи.

Основою одиницею для роботи з даними є статті які створють користувачі групи. Стаття являє собою документ з текстом, який можна форматувати (жирний шрифт, курсив), додавати нумеровані списки, створювати заголовки різних рівнів, створювати таблиці тощо.

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

Сьогодні є велика кількість програмних застосувань що мають функції управління колективними знаннями. В основному це вікі-системи які заточені під внутрішнє використання організаціями з метою створення єдиної бази з накопичених знань, що представляються у вигляді статей, які можна створювати та редагувати. Для створення цих статей використовуються особливі формати текстової розмітки такі як вікі-текст, HTML, XHTML та інші специфічні формати, а також часто є можливості завантаження зображень та файлів.

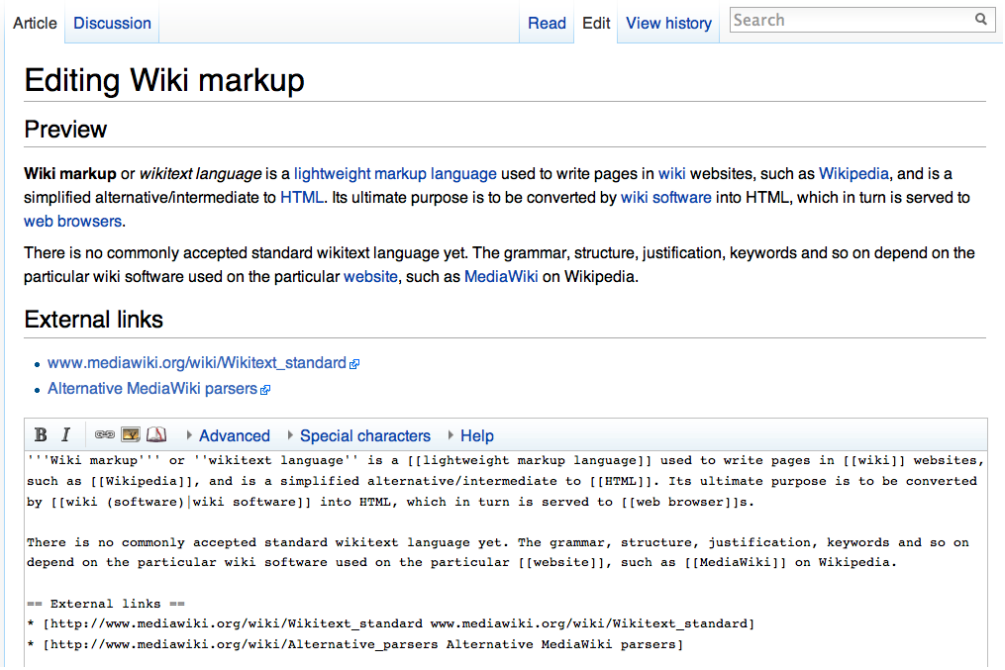


Рисунок 1.1 – Приклад створеня документа за допомогою вікі-розмітки

Основним недоліком таких розміток є те що далеко не всі користувачі володіють знаннями HTML, а у випадку вікі-розмітки потрібно вивчати її специфічний синтаксис для написання статей.

Також відомі технічні рішення дають можливість переглядати історію версій документів, при чому не всі дають можливисть переглядати її всю - обмежуючи до певної кількості останніх версій.

Ну і звісно що в таких системах реалізовані пошук матеріалів по ключових словах чи фразі, коментування статей та є можливості систематизації статей, при чому в деяких варінтах будується дерево категорій, а в деяких статтю можна можна просто позначити деяким тегом, обидва варіанти мають свої переваги та недоліки, так наприклад, коли кількість записів відносно невелика то пошук по дереву категорії може бути дуже зручним, а коли кількість записів дуже велика то орієнуватись в таких категоріях може бути дуже складно.

Більшість з аналогів поставляються як платне або ж безкоштовне програмне забезпечення, яке проте потребує серйозного технічного налаштування та запуску на своєму окремому сервері, що може бути не

найоптимальнішим варіантом для невеликих колективів що хочуть створити свій локальний невеличкий ресурс для обміну знаннями.

1.3.2 Аналіз відомих програмних продуктів

а) Confluence (https://www.atlassian.com/software/confluence) – це програма для співпраці, що представляє з себе вікі-систему для обміну знаннями, розроблена та опублікована австралійською програмною компанією Atlassian. Confluence поставляється із вбудованим веб-сервером Tomcat та базою даних hsql, а також підтримує інші бази даних.

Переваги:

– публікації на Confluence можна створювати з різноманітними графічними елементами;

– підтримується додавання таблиць, графіків, посилань на співробітників тощо;

– програма містить налаштовувані шаблони сторінок;

– є потужний пошук із зручними фільтрами;

– можна обговорювати публікації у вбудованому чаті;

– є мобільний застосунок.

Недоліки:

– програма платна і коштує недешево;

– не можливість будь-якому користувачеві створити свою групу, приватність лише в рамках певної організації.

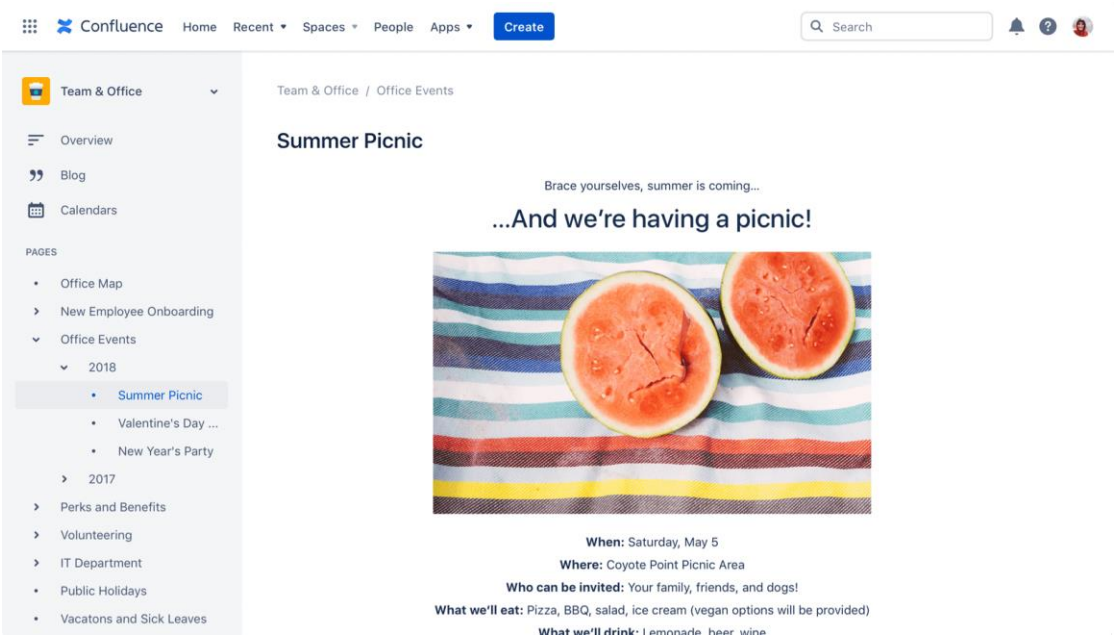


Рисунок 1.2 – Скріншот з ресурсу (atlassian.com/software/confluence)

б) MediaWiki (https://www.mediawiki.org) – програма що являє собою рушій для веб-застосунків, що будуються на базі «вікі» технології. Був розроблений спеціально для проекту «Вікіпедія» а, також використовується у багатьох інших проектах із фонду «Вікімедіа». Повністю безкоштовний продукт з відкритим вихідним кодом.

Переваги:

– можливість обробки тексту як у власному форматі, так і у форматі HTML;

– масштабується під будь-яку кількість користувачів;

– підтримує 60 мов;

– зберігає необмежену кількість версій сторінок;

– є можливості збору статистики переглянутих сторінок.

Недоліки:

– не можливість будь-якому користувачеві створити свою групу;

– не найкраща систематизація інформації;

– система скупа на можливості, якщо не використовувати додаткові плагіни;

– висока складність налаштування.

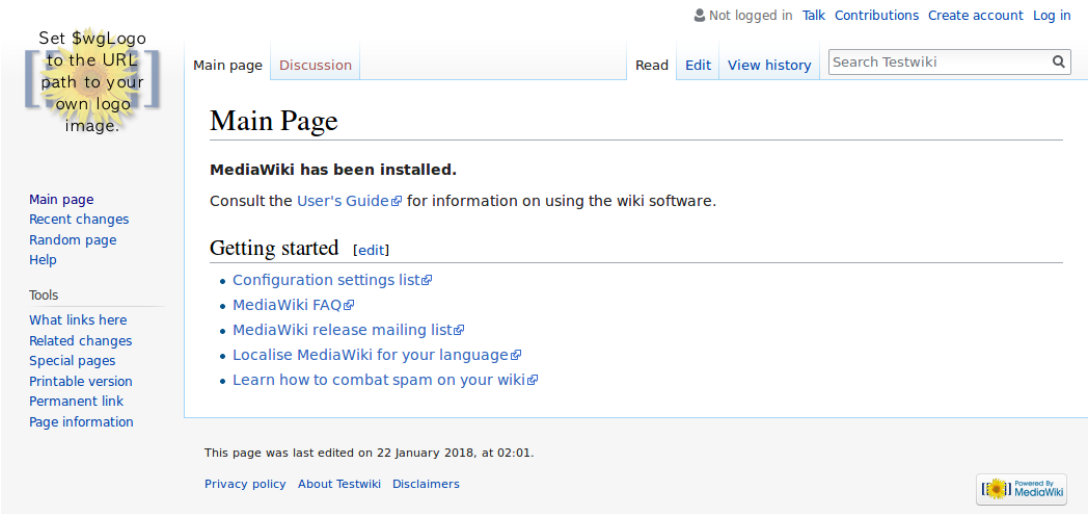


Рисунок 1.3 – Скріншот з ресурсу (https://www.mediawiki.org)

в) DokuWiki (https://www.dokuwiki.org) – простий, але досить потужний вікі-рушій, який може бути використаний для створення будь-якої документації. На відміну від багатьох інших рушіїв, DokuWiki використовує для зберігання сторінок текстові файли та не потребує бази даних, таким чином єдиною вимогою є підтримка хостингом PHP.

Переваги:

– широкі можливості для розмітки, може бути включена підтримка HTML;

– необмежена історія змін сторінки (підлягає налаштуванню);

– автоматичне створення змісту для сторінки;

– підтримує більше 50 мов;

– сторінки розділяються по просторах імен.

Недоліки:

– не можливість будь-якому користувачеві створити свою групу;

– не найкраща систематизація інформації;

– мало можливостей, без втановлення додаткових плагінів;

– складність налаштування.

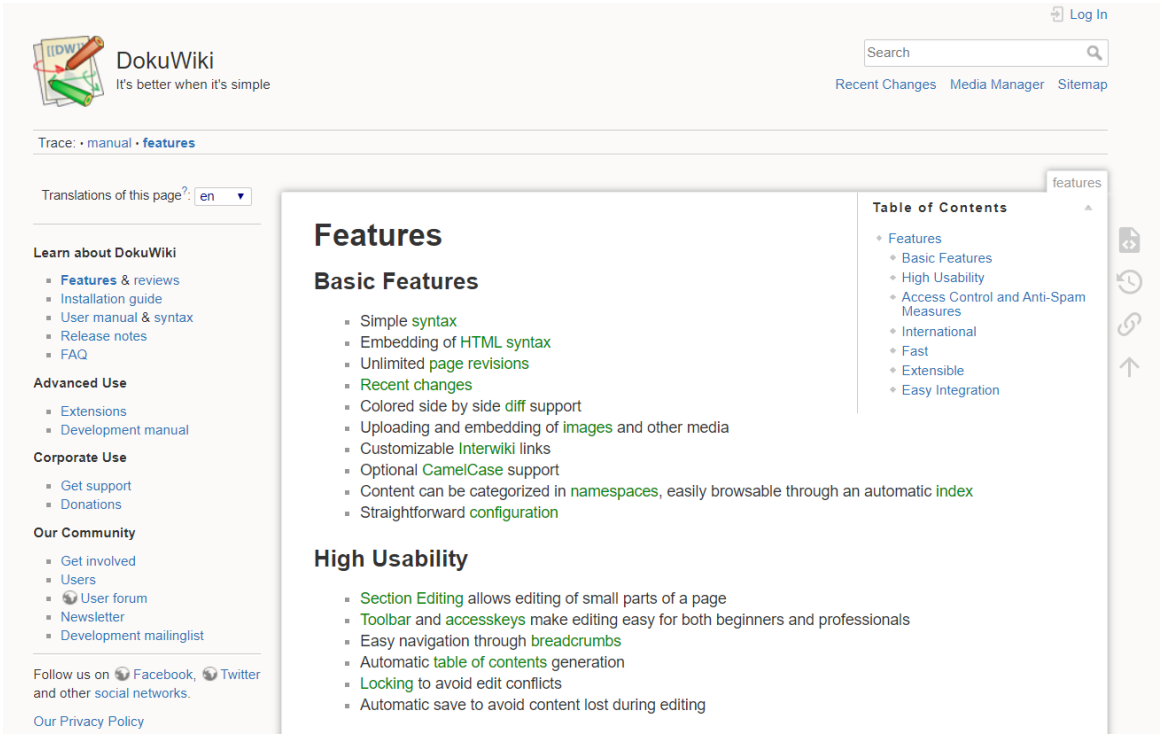


Рисунок 1.4 – Скріншот з ресурсу (https://www.dokuwiki.org)

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Розроблення функціональних вимог

Для роботи із веб-застосунком обов’язково потрібна попередня реєстрація та авторизація, не зареєстрований користувач має можливість лише зареєструватися, неавторизовий користувач може увійти до системи вказавши свій логін та пароль.

Для того щоб почати користуватись можливостями системи, користувач має створити свою групу, або вступити до існуючої, це можна зробити тільки якщо його запросять користувачі цієї групи.

Створивши групу користувач автоматично стає її адміністратором, він може додавати до неї інших зареєстрованих користувачів. У адміністраторів

груп є можливість назначати інших користувачів що входять до складу групи адміністраторами.

Вся інформація що зберігається у групі як і сама група має бути доступна виключно її користувачам.

Статті що створюють користувачі групи повинні представляти з себе документи з текстом, який можна форматувати (жирний шрифт, курсив), додавати нумеровані списки, створювати заголовки різних рівнів, додавати таблиці. Також має бути можливість додання зображень, та вбудування медіафайлів (наприклад відео з YouTube). Стаття повинна обов'язково відноситись до певної категорії. Категорії можуть мати батьківські чи підкатегорії що разом являють собою дерево категорій.

Розроблюваний застосунок повинен забезпечувати наступні варіанти використання:

Таблиця 1.1 – Варіант використання «Реєстрація»

Унікальний ідентифікатор	UC-1
Назва	Реєстрація
Опис	Незареєстрований користувач може зареєструватись вказавши свою пошту та пароль
Учасники	Незареєстрований користувач
Передумови	В системі немає акаунту з поштою користувача
Постумови	В системі з'являється акаунт з поштою користувача
Сценарій	– Користувач заходить на сторінку реєстрації. – Користувач вказує пошту якої ще не має в системі, та валідний пароль. – Користувач натискає кнопку реєстрації.

Таблиця 1.2 – Варіант використання «Авторизація»

Унікальний ідентифікатор	UC-2
Назва	Авторизація
Опис	Неавторизований користувач аторизуєть вказуючи свою пошту та пароль
Учасники	Неавторизований користувач
Передумови	Користувач зареєстрований
Постумови	Користувач успішно авторизувався
Сценарій	– Користувач заходить на сторінку авторизації. – Вказує свою пошту та пароль. – Користувач натискає кнопку авторизації.

Таблиця 1.3 – Варіант використання «Створення групи»

Унікальний ідентифікатор	UC-3
Назва	Створення групи
Опис	Користувач створює нову групу
Учасники	Авторизований користувач
Передумови	-
Постумови	Група створена, користувач є єдиним учасником групи та її адміністратором
Сценарій	– Користувач заходить на головну сторінку. – Вказує назву групи. – Користувач натискає кнопку створення групи.

Таблиця 1.4 – Варіант використання «Створення категорії у групі»

Унікальний ідентифікатор	UC-4
Назва	Створення категорії у групі
Опис	Користувач створює нову категорію
Учасники	Авторизований користувач
Передумови	Користувач є учасником групи в якії він хоче додати нову категорію
Постумови	Категорія створена та видна у дереві категорій
Сценарій	– Користувач заходить на головну сторінку. – Заходить у групу. – Натискає кнопку додання нової категорії. – Опціонально можна вибрати батьківську категорію. – Вказує назву категорії. – Натискає кнопку створення категорії.

Таблиця 1.5 – Варіант використання «Створення нової статті»

Унікальний ідентифікатор	UC-5
Назва	Створення нової статті
Опис	Користувач створює нову статтю
Учасники	Авторизований користувач
Передумови	Користувач є учасником групи в якій він хоче створити нову статтю
Постумови	Стаття створена та видна у дереві категорій

Продовження таблиці 1.5

Сценарій	– Користувач заходить на головну сторінку. – Заходить у групу. – Вибирає категорію до якої хоче додати статтю. – Натискає кнопку створення нової статті. – Вказує назву статті. – Наповнює статтю даними. – Натискає кнопку створення статті.

Таблиця 1.6 – Варіант використання «Редагування існуючої статті»

Унікальний ідентифікатор	UC-6
Назва	Редагування існуючої статті
Опис	Користувач редагує існуючу статтю
Учасники	Авторизований користувач
Передумови	Користувач є учасником групи в якій він хоче відредагувати статтю
Постумови	Стаття відредагована, попередня версія додана до історії
Сценарій	– Користувач заходить на головну сторінку. – Заходить у групу. – Знаходить статтю. – Натискає кнопку редагування. – Редагує дані. – Натискає кнопку збереження.

Таблиця 1.7 – Варіант використання «Додання користувачів до групи»

Унікальний ідентифікатор	UC-7
Назва	Додання користувачів до групи
Опис	Користувач додає нових користувачів до групи
Учасники	Авторизований користувач
Передумови	Користувач є учасником групи до якої він хоче додати нового учасника
Постумови	Користувач якого додали є учасником групи
Сценарій	– Користувач заходить на головну сторінку. – Заходить у групу. – Переходить у вкладку користувачі. – Натискає кнопку додання нового учасника. – Вказує пошту користувача якого хоче додати. – Натискає кнопку додати нового користувача.

Таблиця 1.8 – Варіант використання «Пошук за ключовим словом»

Унікальний ідентифікатор	UC-8
Назва	Пошук за ключовим словом
Опис	Користувач шукає статті у групі по ключову слову
Учасники	Авторизований користувач
Передумови	Користувач є учасником групи в якій він шукає статті
Постумови	Користувачу показуються статті в яких було знайдено ключове слово

Продовження таблиці 1.8

Сценарій	– Користувач заходить на головну сторінку. – Заходить у групу. – Натискає кнопку пошуку статей. – Вказує ключове слово або фразу. – Натискає кнопку пошуку.

Таблиця 1.9 – Варіант використання «Додання коментарів до статей»

Унікальний ідентифікатор	UC-9
Назва	Додання коментарів до статей
Опис	Користувач додає коментарі до статей
Учасники	Авторизований користувач
Передумови	Користувач є учасником групи в якій він хоче додати коментар
Постумови	Коментар доданий
Сценарій	– Користувач заходить на головну сторінку. – Заходить у групу. – Вибирає статтю. – Натискає додати коментар. – Пише коментар. – Натискає кнопку додати.

Таблиця 1.10 – Варіант використання «Надання прав адміністратора»

Унікальний ідентифікатор	UC-10
Назва	Надання прав адміністратора

Продовження таблиці 1.10

Опис	Адміністратор групи надає користувачам групи права адміністратора
Учасники	Авторизований користувач
Передумови	Користувач який надає права є адміністратором групи, користувач якому надають права не є адміністратором
Постумови	Користувач якому надають права став адміністратором групи
Сценарій	– Адміністратор заходить на головну сторінку. – Заходить у групу. – Вибирає користувача із списку учасників групи. – Натискає кнопку надати прав адміністратора.

Таблиця 1.11 – Варіант використання «Видалення статей»

Унікальний ідентифікатор	UC-11
Назва	Видалення статей
Опис	Адміністратор групи видаляє статтю
Учасники	Авторизований користувач
Передумови	Користувач є адміністратором групи
Постумови	Стаття видалена
Сценарій	– Адміністратор заходить на головну сторінку. – Заходить у групу. – Вибирає статтю. – Натискає кнопку видалення.

Таблиця 1.12 – Варіант використання «Видалення категорій»

Унікальний ідентифікатор	UC-12
Назва	Видалення категорій
Опис	Адміністратор групи видаляє категорію
Учасники	Авторизований користувач
Передумови	Користувач є адміністратором групи
Постумови	Категорія видалена, всі статті та підкатегорії що входять до категорії видалені
Сценарій	– Адміністратор заходить на головну сторінку. – Заходить у групу. – Вибирає категорію. – Натискає кнопку видалення.

Таблиця 1.13 – Варіант використання «Видалення груп»

Унікальний ідентифікатор	UC-13
Назва	Видалення груп
Опис	Адміністратор групи видаляє групу
Учасники	Авторизований користувач
Передумови	Користувач є адміністратором групи
Постумови	Група та всі статті що до неї входили видалені
Сценарій	– Адміністратор заходить на головну сторінку. – Заходить у групу. – Натискає кнопку видалення групи.

Таблиця 1.14 – Варіант використання «Видалення користувачів з групи»

Унікальний ідентифікатор	UC-14
Назва	Видалення користувачів з групи

Продовження таблиці 1.14

Опис	Адміністратор групи може видаляти користувачів з групи
Учасники	Авторизований користувач
Передумови	Користувач який видаляє є адміністратором групи, користувач якого видаляють є учасником групи
Постумови	Користувач якого видалили більше не має доступу до групи
Сценарій	– Адміністратор заходить на головну сторінку. – Заходить у групу. – Вибирає користувача із списку учасників групи. – Натискає кнопку видалити з групи.

Відобразим усі варіанти використання застосунку у вигляді структурної схеми варіантів використання на рисунку 1.5. На основі схеми варіантів використання сформуємо функціональні вимоги та наведемо їх у вигляді таблиць.

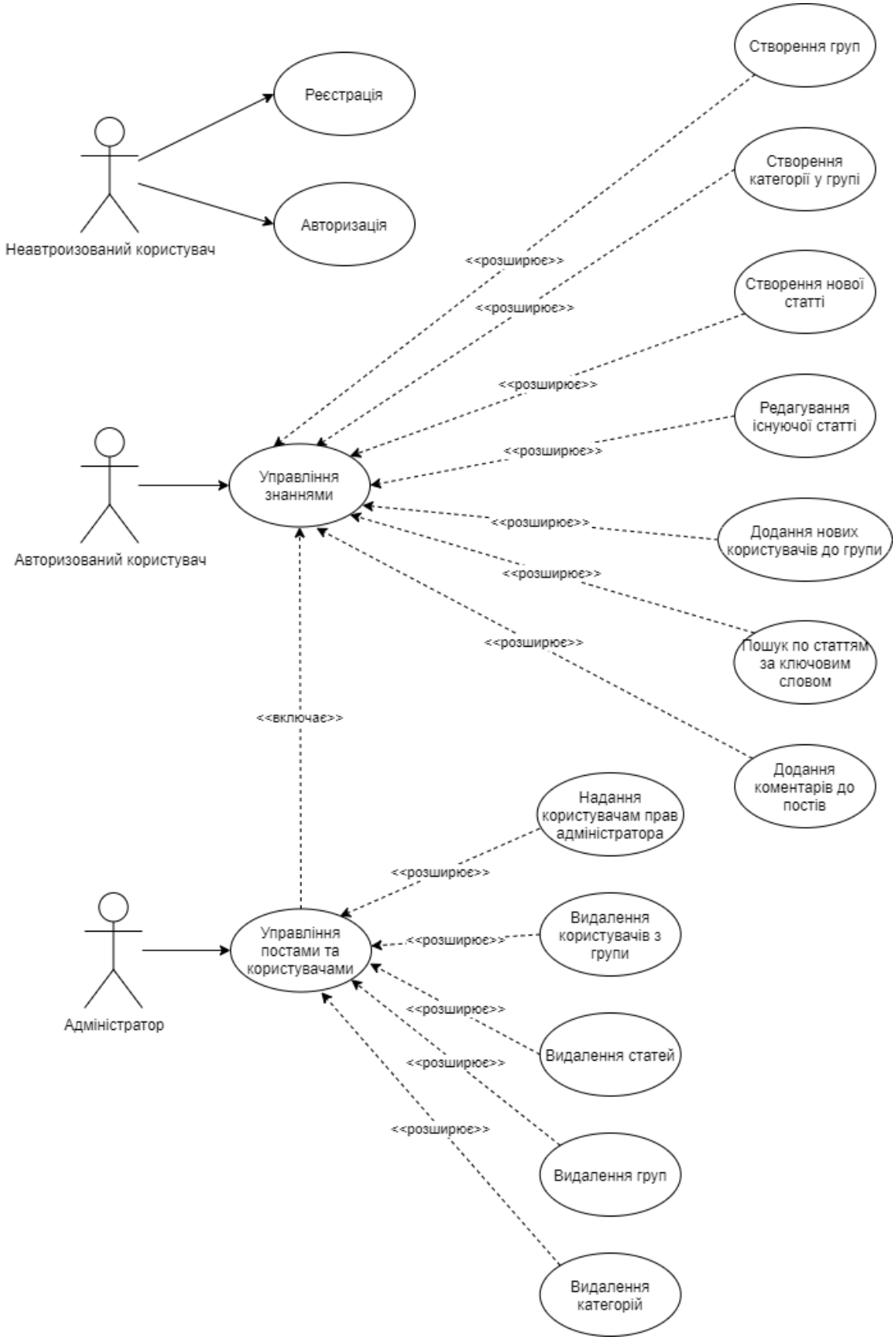


Рисунок 1.5 – Схема структурна варіантів використання

Таблиця 1.15 – Опис функціональної вимоги REQ-1

Унікальний ідентифікатор	REQ-1
Назва	Реєстрація
Вимога	Застосунок повинен надавати можливість будь-якому незареєстрованому користувачеві зареєструватись вказавши свою пошту та пароль

Таблиця 1.16 – Опис функціональної вимоги REQ-2

Унікальний ідентифікатор	REQ-2
Назва	Перевірка паролю
Вимога	Система під час спроби реєстрації повинна перевіряти пароль який ввів користувач на відповідність наступним вимогам: – не менше 8 символів; – не менше 1 маленької літери; – не менше 1 великої літери; – не менше 1 цифри; – не менше 1 символу що не є літерою або цифрою.

Таблиця 1.17 – Опис функціональної вимоги REQ-3

Унікальний ідентифікатор	REQ-3
Назва	Авторизація
Вимога	Застосунок повинен надавати можливість будь-якому користувачеві що уже зареєструвався можливість авторизуватись вказавши свою пошту та пароль

Таблиця 1.18 – Опис функціональної вимоги REQ-4

Унікальний ідентифікатор	REQ-4
Назва	Створення групи
Вимога	Застосунок повинен надавати можливість будь-якому авторизованому користувачеві створити нову групу, вказавши її назву

Таблиця 1.19 – Опис функціональної вимоги REQ-5

Унікальний ідентифікатор	REQ-5
Назва	Перегляд доступних груп
Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість переглядати список доступних йому груп

Таблиця 1.20 – Опис функціональної вимоги REQ-6

Унікальний ідентифікатор	REQ-6
Назва	Створення нової категорії у групі
Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість створити нову категорію у групі що йому доступна

Таблиця 1.21 – Опис функціональної вимоги REQ-7

Унікальний ідентифікатор	REQ-7
Назва	Перегляд доступних категорій та статей

Продовження таблиці 1.21

Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість перегляду дерева категорій та статей у них, які входять до певної групи що доступна користувачеві

Таблиця 1.22 – Опис функціональної вимоги REQ-8

Унікальний ідентифікатор	REQ-8
Назва	Створення нових статей
Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість створення нових статей у певних категоріях, в групах що доступні користувачеві

Таблиця 1.23 – Опис функціональної вимоги REQ-9

Унікальний ідентифікатор	REQ-9
Назва	Редагування існуючих статей
Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість редагування статей що знаходяться у групах що доступні користувачеві

Таблиця 1.24 – Опис функціональної вимоги REQ-10

Унікальний ідентифікатор	REQ-10
Назва	Додання коментарів до статей

Продовження таблиці 1.24

Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість додання коментарів до статей що доступні користувачеві

Таблиця 1.25 – Опис функціональної вимоги REQ-11

Унікальний ідентифікатор	REQ-11
Назва	Додання нових користувачів до групи
Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість додання нових користувачів до груп що є доступними користувачеві

Таблиця 1.26 – Опис функціональної вимоги REQ-12

Унікальний ідентифікатор	REQ-12
Назва	Пошук по статтям за ключовим словом
Вимога	Застосунок повинен надавати будь-якому авторизованому користувачеві можливість пошуку статей які входять до груп що є доступними користувачеві. Результатами пошуку мають бути ті статті в яких знайдено співпадіння ключових слів

Таблиця 1.27 – Опис функціональної вимоги REQ-13

Унікальний ідентифікатор	REQ-13
Назва	Надання користувачам прав адміністратора
Вимога	Система повинна надавати будь-якому авторизованому користувачеві що є адміністратором певної групи можливість надавати учасникам групи права адміністратора

Таблиця 1.28 – Опис функціональної вимоги REQ-14

Унікальний ідентифікатор	REQ-14
Назва	Видалення статей
Вимога	Система повинна надавати будь-якому авторизованому користувачеві що є адміністратором певної групи можливість видалити статті що входять до цієї групи

Таблиця 1.29 – Опис функціональної вимоги REQ-15

Унікальний ідентифікатор	REQ-15
Назва	Видалення категорій
Вимога	Система повинна надавати будь-якому авторизованому користувачеві що є адміністратором певної групи можливість видалити категорії що входять до цієї групи

Таблиця 1.30 – Опис функціональної вимоги REQ-16

Унікальний ідентифікатор	REQ-16
Назва	Видалення груп

Продовження таблиці 1.30

Вимога	Система повинна надавати будь-якому авторизованому користувачеві що є адміністратором певної групи можливість видалити цю групу

Таблиця 1.31 – Опис функціональної вимоги REQ-17

Унікальний ідентифікатор	REQ-17
Назва	Видалення користувачів з групи
Вимога	Система повинна надавати будь-якому авторизованому користувачеві що є адміністратором певної групи можливість видаляти з цієї групи користувачів що входять до неї та не є адміністраторами

Зобразимо залежності між варіантами використання та функціональними вимогами на матриці трасування (рисунок 1.6).

	REQ-1	REQ-2	REQ-3	REQ-4	REQ-5	REQ-6	REQ-7	REQ-8	REQ-9	REQ-10	REQ-11	REQ-12	REQ-13	REQ-14	REQ-15	REQ-16	REQ-17
UC-1	■	■															
UC-2			■														
UC-3				■	■												
UC-4						■	■										
UC-5							■	■									
UC-6									■								
UC-7											■						
UC-8												■					
UC-9										■							
UC-10													■				
UC-11														■			
UC-12															■		
UC-13																■	
UC-14																	■

Рисунок 1.6 – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Розроблюване програмне забезпечення є веб-застосунком і має відповідати наступним нефункціональним вимогам:

– інтерфейс повинен бути зручним, користувач повинен мати змогу інтуїтивно зрозуміти на яку кнопку натиснути, щоб зробити потрібну дію;

– час відгуку сторінки повинен бути не більше 3 секунд;

– користувач застосунку повинен мати доступ до мережі інтернет;

– веб-застосунок повинен працювати в наступних браузерах: Google Chrome, Mozilla Firefox, Opera;

– всі вийняткові ситуації що зумовлені некоректними діями користувачів програма має правильно обробляти і давати повідомлення про відповідні помилки.

1.4.3 Постановка комплексу завдань модулю

Головна мета застосунку — зберігання та ситематизація інформації базуючись на користувацьких знаннях та досвіді, а також для зручного

перегляду та пошуку способів вирішення існуючих проблем для якої-небудь конкретної предметної області. Підтримка інформації у актуальному стані доповнюючи або коректуючи її та надаючи можливість користувачам висловлювати свою думку коментуючи публікації.

Задля досягнення поставленняої мети, до веб-застосунку були сформульовані наступні задачі для розробки:

– забезпечити можливість створення користувацького акаунту в системі, авторизація;

– забезпечити будь-якому зареєстрованому користувачеві створенння груп і можливість їх адміністрування;

– забезпечити користувачам групи можливість додавати та редагувати статті;

– забезпечити користувачам групи можливість додавати та редагувати категорії статей;

– забезпечити користувачам групи пошук інформації за ключовим словом;

– забезпечити користувачам групи додавання коментарів до статей;

– для адміністраторів груп є можливість управління статтями та користувачами.

1.5 Висновки по розділу

В наш час організації, спільноти та просто групи людей яких об’єднують спільні інтереси постійно діляться між собою велкою кількістю інформації щодо їх предметної області, проте сервіси які дозволять робити це в максимально зручному для користувачів вигляді є складнодоступними або важкими у налаштуванні і витрати на них не завжди є доцільними, особливо коли таким сервісом потрібно скористатись зовсім не великій групі людей.

В рамках даного розділу був проведений аналіз існуючих аналогів, визначено переваги та недоліки кожного. Після їх виявлення було представлено рішення, у вигляді вільної платформи – веб-застосунку, яким зможе легко користуватись навіть не досвідчений користувач, тому розробку застосування можна вважати актуальною. Після виявлення усіх можливих варіантів використання застосунку, було визначено функціональні та нефункціональні вимоги, які необхідно реалізувати в розроблюваній системі.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Розглянемо детально основні процеси у застосунку відповідно до варіантів використання що були зазначені у функціональних вимогах.

а) Для процесу реєстрації в застосунку побудуємо структурну схему діяльності:

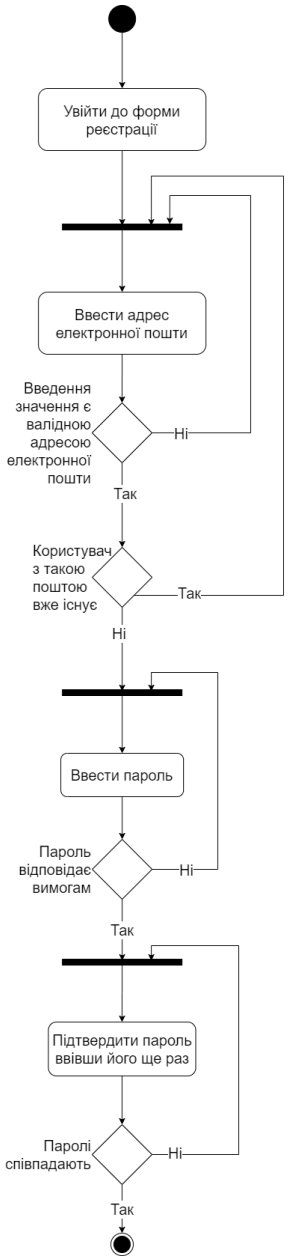


Рисунок 2.1 – Схема структурна діяльності для процесу реєстрації

б) Для аунтефікації в застосунку потрібно перейти у вкладку «Login», ввести свою адресу електронної пошти і пароль, та натиснути кнопку логіну. Якщо користувача з такою поштою не знайдено або пароль не правильний будуть видані відповідні помилки.

в) Для створення нової групи потрібно бути авторизованим та перейти на головну сторінку, і у вкладці з групами натиснути кнопку створення нової групи, далі потрібно вказати її назву що має бути не менше 3 символів та натиснути кнопку створення.

г) Для процесу створення статті в застосунку побудуємо структурну схему діяльності:

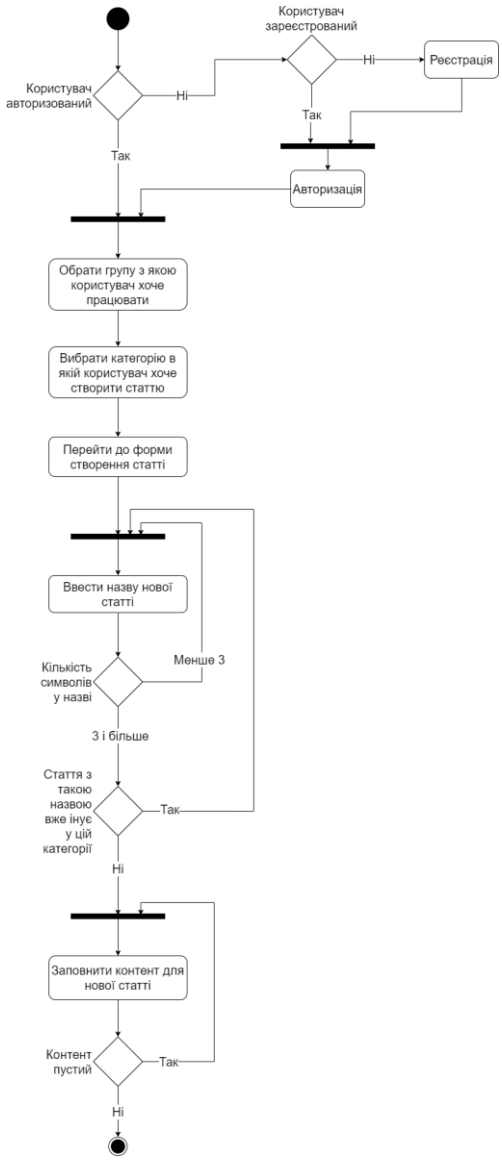


Рисунок 2.2 – Схема структурна діяльності для процесу створення статті

д) Для створення нової категорії у групі потрібно бути авторизованим та перейти на головну сторінку, і у вкладці з групами вибрати групу в якій потрібно створити категорію, на вкладці з категоріями натиснути кнопку створення нової категорії, якщо користувач хоче створити підкатегоії для цього потрібно вибрати батьківську категоію, далі потрібно вказати назву нової категорії що має бути не менше 3 символів та натиснути кнопку створення.

е) Для пошуку статей у групі потрібно бути авторизованим та перейти на головну сторінку, і у вкладці з групами вибрати групу в якій потрібно шукати статті, далі на вкладці з статтями ввести фразу та натиснути кнопку пошуку.

2.2 Архітектура програмного забезпечення

Розроблюване програмне забезпечення складається з трьох частин:

- клієнтьский застостунок;
- серверний застосунок;
- база даних.

В наступних розділах розглянемо детальніше кожну з частин.

2.2.1 Клієнтський застостунок

Для розробки клієнтського застосунку була використана бібліотека React що дозволяє програмістам будувати великі та потужні веб-застосунки, які будуть використовувати дані, що можуть змінюватись з часом, без необхідності повного перезавантаження сторінки.

Розглянемо структуру застосунку:

Рисунок 2.3 – Структура React застосунку

Проект має декілька директорій що об’єднують які файли мають якусь спільну логіку та значення, розглянемо їх призначення в наступній таблиці:

Таблиця 2.1 – Призначення директорій

Назва директорії	Призначення
ClientApp	Коренева директорія для всього застосунку
node_modules	node-modules – це сховище для різних сторонніх бібліотек які використовуються у проекті
public	Служить для зберігання HTML-сторінки та різних медіаресурсів застосунку, такі як зображення
src	Папка із вихідним кодом розробки застосунку
components	Папка що зберігає React компоненти
api-authorization	Набір сервісів, що забезпечують реєстрацію та авторизацію

Основними складовими React застосунків є компоненти – JavaScript функції що дозволяють розділяти інтерфейс на незалежні частини. Такаж для зручності розробки деякі сервіси були винесені у окремі JavaScript файли.

Розглянемо основні JavaScript файли застосунку та їх призначення у наступній таблиці:

Таблиця 2.2 – Призначення компонентів

Назва JavaScript файлу	Призначення
App.js	Основний React компонент що дозволяє відрендерити увесь застосунок
Layout.js	React компонент який визначає основну розмітку застосунку
NavMenu.js	React компонент що містить у собі навігаціне меню

Продовження таблиці 2.2

ApiService.js	JavaScript сервіс що дозволяє обмінюватись данними з серверним API
Home.js	React компонент – початкова сторінка що містить в собі основну інформацію про застосунок
KB.js	React компонент – головна сторінка на якій користувач може використовувати основні функції застосунку
ArticleEdit.js	React компонент – сторінка на якій користувач може редагувати статтю
ArticleHistory.js	React компонент – сторінка на якій користувач може переглядати історію змін статті
ArticlesManager.js	React компонент – сторінка на якій користувач може переглядати та шукати статті що входять до групи
Comments.js	React компонент – що відповідає за відображення коментарів до статей
GroupItem.js	React компонент – що відповідає за відображення обраної користувачем групи
GroupUsers.js	React компонент – сторінка на якій користувач може переглянути користувачів що входять до групи

Розглянемо інші основні файли що забезпечують роботу застосунку:

– index.html є традиційним файлом, який використовується в якості індексу для каталогу веб-сайту;

– KB.css, NavMenu.css, Comments.css, custom.css – файли зі стилями CSS;

– package.json – JSON файл що визначає залежності від зовнішнх бібліотек.

2.2.2 Серверний застосунок

Серверний застосунок розроблений на мові C#, для розробки веб-сервісу що обмінюється даними з клієнтським застосунком була використана платформа ASP.NET Core. В якості архітектурного рішення для обміну данними додатків у мережі використовується REST API.

Основними компонентами серверного застосунку є REST API контроллери, кожен яких відповідає за обмін всіма данними що стосуються конкретної сутності. Для доступу до певного методу контроллера потрібно надіслати запит на сервер що містить валідний шлях до ресурсу, певний HTTP метод (GET, POST, DELETE тощо), та обов’язковою умовою є заголовок із токеном авторизації. Переважним форматом обміну даними є JSON.

Розглянемо підтримувані HTTP-методи та їх загальне значення в застосунку:

Таблиця 2.3 – Загальне значення HTTP-методів що підтримуються застосунком

HTTP-метод	Значення
GET	Використовується для отримання данних по певному ресурсу, передача додаткової інформації у тілі запиту не потрібна
PUT	Використовується для оновленння данних по певному ресурсу, у тілі запиту обов’язково потрібно вказати нові дані для певної сутності

Продовження таблиці 2.3

POST	Використовується для створення нових данних по певному ресурсу, у тілі запиту обов’язково потрібно вказати дані для певної сутності
DELETE	Використовується для видалення існуючих данних по певному ресурсу, передача додаткової інформації у тілі запиту не потрібна

Далі розглянемо наявні в серверному застосунку HTTP-запити що підтримуються котролерами, та надають можливість обміну даними з клієнтським застосунком. Шлях HTTP запиту завжди буде починатись з «https://» та назви хосту, тому в наступних талицях опустим їх для наглядності. Частини шляху що виділені фігурними дужками є змінними параметрами запиту які будуть описані в колонці «Параметри запиту».

Таблиця 2.4 – Запити що відносяться до сутності «Статті»

HTTP запит	Параметри запиту	Параметри тіла запиту	Призначення
GET api/articles/{articleId}	articleId – унікальний ідентифікатор статті	-	Отримання даних про статтю
PUT api/articles/{articleId}	articleId – унікальний ідентифікатор статті	title – заголовок статті; content – контент статті;	Оновлення статті
DELETE api/articles/{articleId}	articleId – унікальний ідентифікатор статті	-	Видалення статті

Продовження таблиці 2.4

GET api/articles/{articleId}/history	articleId – унікальний ідентифікатор статті	-	Отримання історії змін статті
POST api/articles	-	title – заголовок статті; content – контент статті; categoryId – унікальний ідентифікатор категорії;	Створення нової статті

Таблиця 2.5 – Запити що відносяться до сутності «Категорії»

HTTP запит	Параметри запиту	Параметри тіла запиту	Призначення
GET api/categories/{categoryId}/articles	categoryId – унікальний ідентифікатор категорії; articleId – унікальний ідентифікатор статті;	-	Отримання даних про статті що входять до даної категорії
PUT api/categories/{categoryId}	categoryId – унікальний ідентифікатор категорії	name – назва категорії;	Оновлення назви категорії

Продовження таблиці 2.5

DELETE api/categories /{categoryId}	categoryId – унікальний ідентифікатор категорії	-	Видалення категорії
POST api/categories	-	name – назва категорії; parentCategoryId – унікальний ідентифікатор батьківської категорії; groupId – унікальний ідентифікатор групи	Створення нової категоріїї

Таблиця 2.6 – Запити що відносяться до сутності «Групи»

HTTP запит	Параметри запиту	Параметри тіла запиту	Призначення
GET /api/groups	-	-	Отримання даних про групи що доступні користувачеві
POST /api/groups	-	name – назва групи;	Створення нової групи
GET /api/groups/ {groupId}	groupId – унікальний ідентифікатор групи	-	Отримання даних про конкретну групу

Продовження таблиці 2.6

PUT /api/groups/ {groupId}	groupId – унікальний ідентифікатор групи	name – назва групи;	Оновлення назви групи
GET /api/groups/ {groupId}/ users	groupId – унікальний ідентифікатор групи	-	Отримання даних про користувачів що входять до групи
POST /api/groups/ {groupId}/ users	groupId – унікальний ідентифікатор групи	ids – масив унікальних ідентифікаторів користувачів	Додання користувачів до групи
GET /api/groups/ {groupId}/ categories	groupId – унікальний ідентифікатор групи	-	Отримання даних про категорії що входять до групи
GET /api/groups/ {groupId}/ articles	groupId – унікальний ідентифікатор групи	-	Отримання даних про статті що входять до групи
DELETE /api/groups/ {groupId}/ users/ {userId}	groupId – унікальний ідентифікатор групи; userId – унікальний ідентифікатор користувача;	-	Видалення користувача з групи

Таблиця 2.7 – Запити що відносяться до сутності «Зображення»

HTTP запит	Параметри запиту	Параметри тіла запиту	Призначення
GET api/images/ {imageName}	imageName – унікальний ідентифікатор зображення	-	Отримання зображення що було завантажене на сервер раніше
POST api/images	-	-	Завантаження зображення на сервер

Після отримання запиту до контролеру, визначається валідність вхідних даних, якщо вони не валідні повертається повідомлення про помилку, якщо ж вони валідні, то далі формуються запити на оновленния або отримання даних із бази даних, її детальна структура буде розглянута в наступному розділі.

Для доступу до данних використовується об’єктно-орієнтована технологія Entity Framework Core, що являється потужним ORM рішенням для платформи .NET, та дозволяє взаємодіяти з об’єктами бази бази даний як з об’єктами мови C#.

2.2.3 База даних

В якості бази даних, було обрано Microsoft SQL Server яка розробляється корпорацією Microsoft. На рисунку 2.4 представлено схему основних таблиць бази даних.

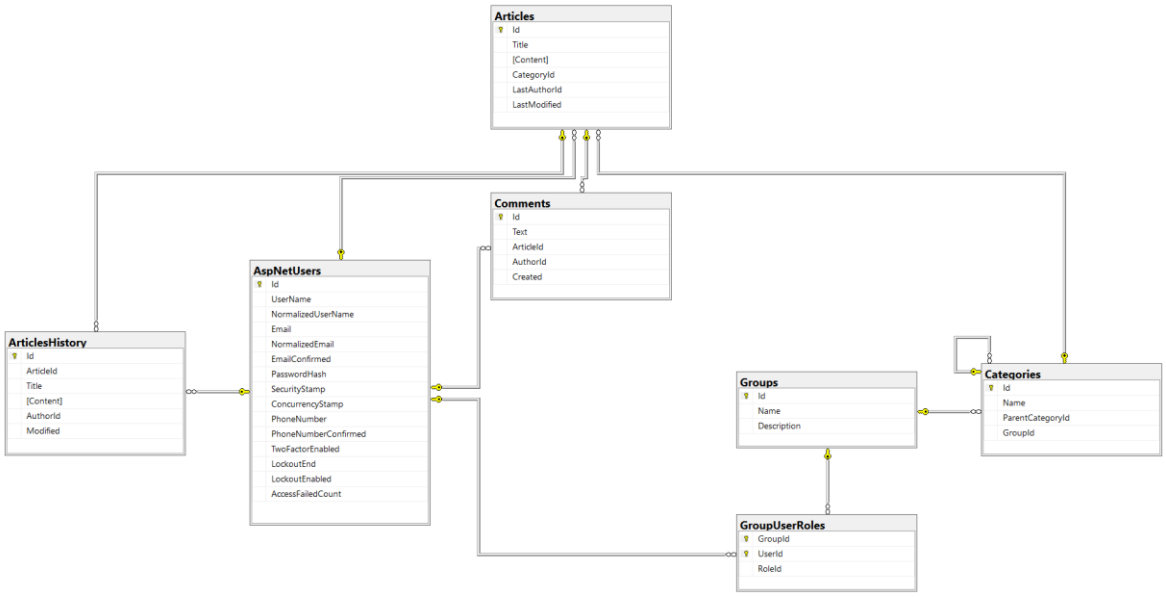


Рисунок 2.4 – Схема бази даних

Розглянемо детально структуру основних таблиць бази даних.

Таблиця 2.8 – Опис таблиць бази даних

Назва таблиці	Опис
Articles	Зберігає інформацію про статті
ArticlesHistory	Зберігає інформацію про історію версій статті
Comments	Зберігає коментарі до статей
Groups	Зберігає інформацію про групи
Categories	Зберігає інформацію про категорії
GroupUserRoles	Зберігає інформацію про користувачів груп та їх роль
AspNetUsers	Зберігає інформацію про користувачів

Розглянемо окремо структуру кожної з таблиць.

Таблиця 2.9 – Структура таблиці Articles

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	int	PK
Title	Назва статті	nvarchar(300)	

Продовження таблиці 2.9

Content	Контент статті	nvarchar(max)	
CategoryId	Ідентифікатор категорії	int	FK
LastAuthorId	Ідентифікатор останнього автора	int	FK
LastModified	Дата та час останньої зміни в документі	datetime2	

Таблиця 2.10 – Структура таблиці ArticlesHistory

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	int	PK
ArticleId	Ідентифікатор статті	int	FK
Title	Назва статті	nvarchar(300)	
Content	Контент статті	nvarchar(max)	
AuthorId	Ідентифікатор автора змін	int	FK
Modified	Дата та час зміни в документі	datetime2	

Таблиця 2.11 – Структура таблиці Comments

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	int	PK
Text	Текст коментаря	nvarchar(2000)	
ArticleId	Назва статті	int	FK
AuthorId	Ідентифікатор автора коментаря	int	FK
Created	Дата та час створення коментаря	datetime2	

Таблиця 2.12 – Структура таблиці Groups

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	int	PK
Name	Назва групи	nvarchar(300)	
Description	Опис групи	nvarchar(1000)	

Таблиця 2.13 – Структура таблиці Categories

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	int	PK
Name	Назва категорії	nvarchar(200)	
ParentCategoryId	Посилання на батьківську категорію	int	FK
GroupId	Посилання на групу	int	FK

Таблиця 2.14 – Структура таблиці GroupUserRoles

Колонка	Опис	Тип даних	Ключ
GroupId	Посилання на групу	int	FK
UserId	Посилання на користувача	int	FK
RoleId	Позначення ролі користувача у групі	int	

Таблиця 2.15 – Структура таблиці AspNetUsers

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	nvarchar(450)	PK
UserName	Назва статті	nvarchar(256)	
NormalizedUserName	Контент статті	nvarchar(256)	
Email	Ідентифікатор категорії	nvarchar(256)	

Продовження таблиці 2.15

Normalized Email	Ідентифікатор останнього автора	nvarchar(256)	
EmailConfirmed	Дата та час останньої зміни в документі	bit	
PasswordHash	Хеш-код паролю	nvarchar(max)	
SecurityStamp	Випадкове значення, яке має змінюватися щоразу, коли змінюються облікові дані користувачів (змінено пароль, видалено логін)	nvarchar(max)	
CuncurrencyStamp	Випадкове значення, яке використовується щоб запобігти конфлікту одночасного оновленя записів	nvarchar(max)	
PhoneNumber	Номер телефону	nvarchar(max)	
PhoneNumberConfirmed	Позначка чи є номер телефону підтвердженим	bit	
TwoFactorEnabled	Позначка того чи підтримується двухвакторна аунтефікація	bit	
LockoutEnd	Час коли закінчить блокування користувача	datetimeoffset(7)	
LockoutEnabled	Позначка того чи є користувач заблокованим	bit	
AccessFailedCount	Кількість невдалих спроб аунтефікацій	int	

2.3 Конструювання програмного забезпечення

Як було описано в попередніх розділах основним архітектурним стилем розроблюваної системи є REST API. На рисунку 2.5 наведено структурну схему класів контролерів що забезпечують обмін данних між клієнтом та сервером.

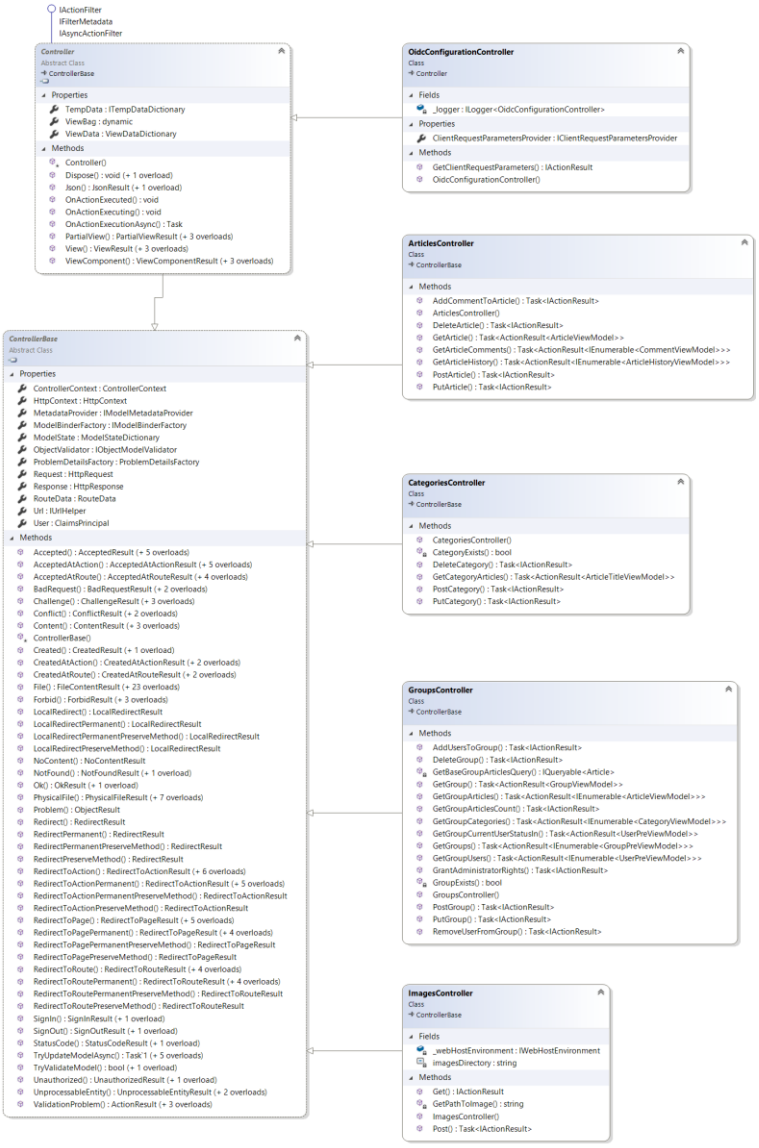


Рисунок 2.5 – Схема структурна класів контролерів

Для доступу до даних що зберігаються у базі даних використовуються Entity Framework репозиторії, кожен з яких відповідає за певну сутність (entity). На рисунку 2.6 наведено структурну схему класів, що використовуються як відображення сутностей що зберігаються у БД.

Рисунок 2.6 – Схема структурна класів сутностей (entities)

2.4 Аналіз безпеки даних

Важливим аспектом системи що потенційно може зберігати цінний для користувача набір даних, є забезпечення захисту інформації. Тому, по-перше, доступ до всіх сторінок застосунку, окрім сторінки з інформацією про

застосунок та сторінок реєстрації та авторизації є лише у зареєстрованих та авторизованих користувачів. По-друге користувачі повинні бачити лише ту інформацію що зберігається у групах до яких входить користувач, тому при кожному запиті на сервер перевіряється принадлежніть користувача до певної групи. Також у системі є певні критичні функції що доступні лише адміністраторам груп, тому коли користувач що не є адміністратором у групі спробує виконати заборонену йому дію, він не зможе цього зробити, адже роль користувача у групі також перевіряється при кожному запиті.

Для реєстрації, авторизації та аунтефікації було використано вбудований у AspNet Core механізм, що використовує IndentityServer для збереження паролів та іншої інформації що стосується користувачів, це потужний OpenID Connect та OAuth 2.0 фреймворк, що запезпечує аунтефікацію у вигляді використання готового сервісу.

Так як доступ до основних ресурсів потребує авторизації, то в HTTP запитах до серверу є обов'язковим передання сесійного JWT-токену, який сервер дає авторизованому користувачеві, інашке клієнт отримає помилку 401 Unauthorized. Також для підвищення безпеки та підтримки шифрування даних що відправляються, серверний застосунок обмінюється даними з клієнтом за протоколом HTTPS.

2.5 Висновки по розділу

Для побудови застосунку була використана архітектура «клієнт-сервер» що передбачає взаємодію і обмін даними між клієнтським застосунком що в даному випадку побудований на базі бібліотеки React, та серверним що побудований на базі фреймворку ASP.NET Core. Способом доступу до серверного застосунку є REST API, що є незалежним від того який клієнт надсилає запити на сервер і потенційно дозволяє розширяти кількість клієнтських застосунків, наприклад мобільними застосунками. Для зберігання

даних була використана БД SQL Server, що є однією з найбільш популярних, та підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Якістю програмного забезпеченя є такий набір властивостей, що визначають його здатність задовольняти установлені або передбачувані вимоги замовника. Для підтвердження якості прогамного застосунку можна використати процеси верифікації та валідації.

Верифікація – означає підтвердження того, що програмне забезпечення розроблене відповідно до усіх вимог що до нього ставилися, а також що за результатами чергового етапу розробки з'ясовується що вони відповідають тим обмеженням, що були сформульовані ще на попередніх етапах.

Валідація – це підтвердження того, що програмне забезпечення саме по собі є правильним, тобто перевірка того, чи дійсно він задовольняє очікуванням і потребам користувачів та замовників продукту.

Тестування – це перевірка того чи відповідає програмне забезпечення вимогам, що здійснюється за допомогою моніторингу його роботи в спеціально штучно створених ситуаціях. Тестування є одним із найбільш важливих способів контролю якості ПЗ, і практично єдиним з них, доступним кінцевому користувачеві. Отже, якість ПЗ буде визначатись результатами тестування, що буде описане у наступних розділах.

3.2 Опис процесів тестування

Об’єктами тестування є:

– серверний застосунок, а саме його здатність видавати коректні відповіді на коректні запити та здатність видавати помилки при некоректних запитах;

– інтерфейс клієнтського застосунку, та його взаємодія з сервером.

Сценарії що будуть протестовані:

– реєстрація нового користувача;
– авторизація користувача;
– створення нової групи;
– створення нової категорії;
– створення нової статті;
– редагування існуючої статті;
– додання нових користувачів до групи;
– пошук статей в групі за ключовим словом;
– додання коментарів до статей;
– надання користувачам статусу адміністратора;
– видалення користувача з групи;
– видалення статті;
– видалення групи;
– видалення категорії.

Тестування описаних вище сценаріїв буде функціональним тестуванням, оскільки будуть перевірятисть функції програмного забезпечення. Також вкажемо нефункціональні тести що будуть проводитись на даному етапі розробки:

– тест сумісності з браузерами;
– тест часу відгуку сторінки;
– тестування зручності інтерфейсу;
– навантажувальні тести.

Так як тестування буде здійснено розробником програмного забезпечення то воно може проводитись у режимі білої скриньки (White box testing), тобто коли у тестувальника є доступ до вихідного коду.

У тестах будуть розглянуті як позитивні так і негативні сценарії, це потрібно для того щоб пересвідчитись що ПЗ видає коректні результати при

коректних вхідних даних, та коректні повідомлення про помилки при некоректних вхідних даних.

Тести будуть проводитись у мануальному режимі, оскільки написання автоматизованих тестів для системи що не має надто складної бізнес логіки, на даному етапі розробки є надлишковим і часу на їхню розробку буде витрачено більше ніж при ручний тестах.

3.3 Опис контрольного прикладу

Під час тестування застосунку було перевірено всі сценарії що вказані в пункті 3.2. Опис контрольних прикладів тестування наведено в таблиях нижче.

Таблиця 3.1 – Позитивний тест реєстрації в застосунку

Назва тесту	Позитивна перевірка реєстрації користувача
Передумови	В системі немає користувача з електронною поштою що вказана у тесті; Пароль відповідає вимогам;
Схема виконання	Користувач заходить на сайт та переходить на сторінку реєстрації, вводить пошту і пароль, натискає кнопку реєстрації
Очікуваний результат	У системі створено акаунт с поштою користувача
Отриманий результат	У системі створено акаунт с поштою користувача

Таблиця 3.2 – Негативний тест реєстрації в застосунку

Назва тесту	Негативна перевірка реєстрації користувача
Передумови	В системі немає користувача з електронною поштою що вказана у тесті; Пароль не відповідає вимогам;
Схема виконання	Користувач заходить на сайт та переходить на сторінку реєстрації, вводить пошту і пароль, натискає кнопку реєстрації
Очікуваний результат	Користувачу видана помилка про те що пароль не відповідає вимогам
Отриманий результат	Користувачу видана помилка про те що пароль не відповідає вимогам

Таблиця 3.3 – Позитивний тест пошуку статті

Назва тесту	Позитивна перевірка пошуку статті у групі
Передумови	Користувач авторизований; Користувач є учасником групи в якій здійснює пошук; У групі присутні статті в яких є ключове слово чи фраза що шукає користувач;

Продовження таблиці 3.3

Схема виконання	Користувач заходить на сайт авторизується, вибирає одну з доступних груп, переходить до пошуку статей, вводить слово чи фразу
Очікуваний результат	Користувачу виданий список статей в яких є ключове слово чи фраза що він шукає
Отриманий результат	Користувачу виданий список статей в яких є ключове слово чи фраза що він шукає

Таблиця 3.4 – Негативний тест пошуку статті

Назва тесту	Негативна перевірка пошуку статті у групі
Передумови	Користувач авторизований; Користувач є учасником групи в якій здійснює пошук; У групі присутні немає статей в яких є ключове слово чи фраза що шукає користувач;
Схема виконання	Користувач заходить на сайт авторизується, вибирає одну з доступних груп, переходить до пошуку статей, вводить слово чи фразу

Продовження таблиці 3.4

Очікуваний результат	Користувачу видано повідомлення що жодної статті не знайдено
Отриманий результат	Користувачу видано повідомлення що жодної статті не знайдено

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розроблюване програмне забезпечення повинно бути розгорнуто на сервері де встановлена одна з наступних операційних систем:

– Windows 7 SP1 або вище;

– Windows Server 2008 R2 SP1 або вище;

– Red Hat Enterprise Linux 7 або вище;

– Ubuntu 14.04 або вище.

Також на сервері повинно бути встановлене наступне програмне забезпеченяя:

– .NET Core 3.1 SDK;

– Node.js v12.13.0;

– React.js v16.13.1;

– SQL Server 2019.

Для розгортання застосунку потрібно скачати його програмний код із системи контролю версій, перейти у папку з проектом та виконати у консолі наступну команду «dotnet publish».

За необхідності можна налаштувати рядок під’єднання до бази данних змінивши параметр «DefaultConnection» у файлі «appsettings.json», також у цьому ж файлі можна налаштувати адресу на якій буде розгортатись програмне забезпечення. Якщо база даних до якої має підключитись застосунок ще не розгорнута має то вона буде створена пустою автоматично.

Так як клієнтський застосунок налаштований на автоматичне розгортання разом із серверним застосунком то ніяких додаткових зусиль з його розгортання не потрібно.

Структурна схема розгортання компонентів застосунку представлена на рисунку 4.1.

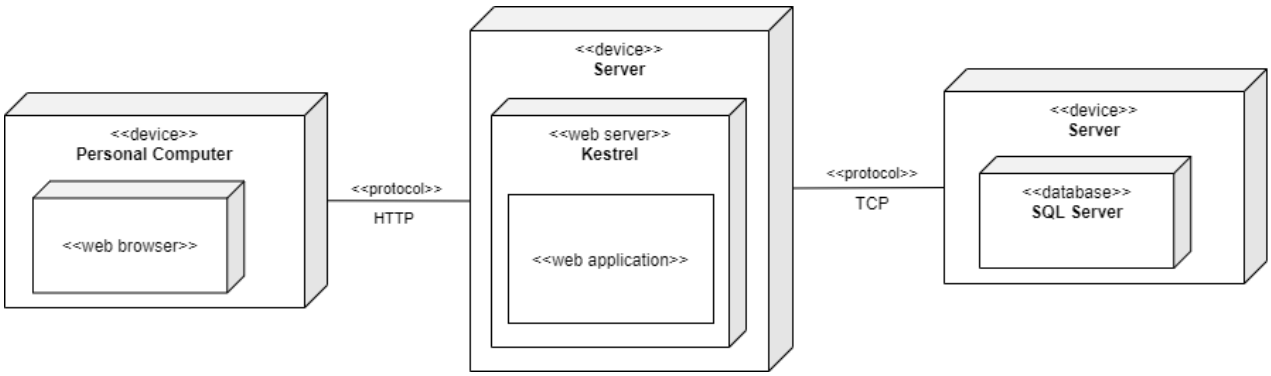


Рисунок 4.1 – Схема структурна розгортання

4.2 Робота з програмним забезпеченням

Робота із застосунком може здійснюватись через один із наступних браузерів (рекомендується використовувати останню версію):

– Google Chrome;

– Mozilla Firefox;

– Opera.

Детально ознайомитись із роботою з програмним забезпеченням можна у документі «Керівництво користувача».

ВИСНОВКИ

В рамках виконання даного дипломного проекту був розроблений веб застосунок основною метою якого є створення умов, при яких накопичені знання та досвід користувачів будуть ефективно використовуватись для виконання важливих для них завдань та досягнення колективних цілей. Застосунок дозволить зекономити час користувачів на пошук вирішення часто виникаючих запитань, перетворивши колективні знання у відповіді що можна буде легко знайти та переглядати у вигляді документів що матимуть різноманітні способи форматування для максимально зручного їх перегляду. Усі учасники колективу що є користувачами застосунку зможуть незалежно від їх статусу ділитись ідеями та отримувати до них зворотний зв’язок, а також підтримувати існуючу інформацію у актуальному стані доповнюючи або коректуючи її.

Застосунок був розроблений із використанням актуальних та найновітніших технологій як ASP NET Core 3.1 для серверної частини, React.js 16.13.1 для клієнтської та SQL Server 2019 для бази даних, що підвищує його надійність та зручність розробки, спрощує підтримку та додання нового функціоналу.

Як відомо, програмне забезпечення не можна вважати працюючим доки не перевірена його роботопридатність, тому розроблений застосунок був ретельно протестований, що підтверджує його якість яка відповідає усім сучасним стандартам.

ПЕРЕЛІК ПОСИЛАНЬ

1) Reynders Fanie Modern API Design with ASP.NET Core 2: Building Cross-Platform Back-End Systems – Apress, 2018 – 262 с.

2) Dino Esposito Programming ASP.NET Core – Microsoft Press, 2018 – 416 с.

3) Andrew Lock ASP.NET Core in Action – Manning Publications, 2018 – 712 с.

4) REST [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/REST

5) .NET Core [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/.NET_Core

6) Introduction to ASP.NET Core [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1

7) Create a web API with ASP.NET Core [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1

8) Overview of ASP.NET Core Security [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-3.1

9) React [Електронний ресурс] – Режим доступу: https://reactjs.org/docs/getting-started.html

10) Microsoft SQL Server [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для управління та систематизації колективних знань

Технічне завдання

КПІ.ІП-6306.045440.03.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І.І. Вітковська

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Б. С. Верещак

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосунок для управління та систематизації колективних знань

Галузь застосування:Наведене технічне завдання поширюється на розробку веб-застосунку для управління та систематизації колективних знань [КПІ.ІП-6306.045440.03.91], котра використовується для зберігання та ситематизації інформації базуючись на користувацьких знаннях та призначена для будь-якої галузі діяльності.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для спрощення процесу обміну знаннями у колективах, спільнотах чи групах людей яких об’єднують спільні інтереси.

Метою розробки є створення умов, при яких накопичені знання та досвід користувачів будуть ефективно використовуватись для виконання важливих для них завдань та досягнення колективних цілей. Зекономити час користувачів на пошук вирішення часто виникаючих запитань, перетворивши колективні знання у відповіді що можна буде легко знайти та переглядати у вигляді документів що матимуть різноманітні способи форматування для максимально зручного їх перегляду. Надати можливість усім учасникам колективу незалежно від їх статусу ділитись ідеями та отримувати до них зворотний зв’язок. А також підтримка існуючої інформації у актуальному стані доповнюючи або коректуючи її.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– можливість створення користувацького акаунту в системі;

– авторизація у застосунку;

– забезпечити будь-якому зареєстрованому користувачеві створенння груп і можливість їх адміністрування;

– забезпечити користувачам групи можливість додавати або редагувати статті;

– забезпечити користувачам групи можливість додавати категорії статей;

– забезпечити користувачам групи пошук інформації за ключовим словом;

– забезпечити користувачам групи додавання коментарів до статей;

– для адміністраторів груп є можливість видалення груп, категорій, статей та користувачів із групи.

4.1.2 Розробку виконати на платформі .NET

4.1.3 Додаткові вимоги:

– інформація що зберігається у групах доступна лише її користувачам;

– можливість надання користувачам групи статусу адміністратора є лише у адміністраторів цих же груп.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

Умови обслуговування не пред’являються.

4.3.3 Обслуговуючий персонал

Умови обслуговуючого персоналу не пред’являються.

4.4 Вимоги до складу і параметрів технічних засобів

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

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Процесор: x86 або x64.

4.4.2.2 Об‘єм ОЗП: 2 Гб.

4.4.2.3 Об‘єм жорсткого диску: 4 Гб вільного місця

4.5. Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням однієї з наступних операційних систем:

– Windows 7 SP1 або вище;
– Windows Server 2008 R2 SP1 або вище;
– Red Hat Enterprise Linux 7 або вище;
– Ubuntu 14.04 або вище.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: HTTP запити, з даними у форматі JSON в тілі запиту.

4.5.3 Результати повинні бути представлені в наступному форматі: HTTP відповіді, з даними у форматі JSON в тілі запиту, для користувача клієнтського застосунку результатом повинна бути HTML сторінка.

4.5.4 Програмне забезпечення повинно бути доступним через HTTP запити.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

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

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 100 аркушах формату А4 (без додатків 5.3.2 - 5.3.4).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування

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

5.4.1 Схема бази даних

5.4.2 Структурна схема класів програмного забезпечення

5.4.3 Креслення вигляду екранних форм

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	11.03.2020	
2.	Розробка технічного завдання	07.04.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	15.04.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	25.04.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму …)
5.	Програмна реалізація програмного забезпечення	01.05.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	12.05.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	26.05.2020	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	27.05.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

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

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для управління та систематизації колективних знань

Програма та методика тестування

КПІ.ІП-6306.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І.І. Вітковська

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Б.С. Верещак

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-застосунок для управління та систематизації колективних знань, що написаний мовами програмування C# та JavaScript, та побудований на базі архітектурного стилю REST.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування застосунку повиноо бути перевірено наступне:

- реєстрація та авторизація в системі;
- функціональна працездатність елементів сторінок веб застосунку;
- можливості роботи з доступними користувачеві групами;
- можливості роботи з доступними користувачеві категоріями;
- можливості роботи з доступними користувачеві статтями;
- забезпечення належного рівня захисту інформації;
- зручність роботи з інтерфейсом веб застосунку;
- відповідність дизайну застосунку до вимог Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується за принципами як «білої скриньки» так і «чорної скриньки», тобто перевіряється як програмний код, так і безпосередньо веб застосунок на відповідність функціональним вимогам, даний принцип тестування також прийнято називати принципом «сірої скриньки».

Для тестування використовуються наступні методи:

- юніт тести;
- інтеграційні тести;
- навантажувальне тестування;
- приймальне тестування;
- тестування для користувача інтерфейсу;
- тестування безпеки.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Юніт та інтеграційні тести виконуються через запуск підготовлених тест кейсів за допомогою тестового фреймворку NUnit. Навантажувальне тестування проводиться за допомогою інструменту Apache JMeter. Приймальне тестування виконується вручну як перевірка того, що система відповідає узгодженим вимогам.

Працездатність веб застосунку перевіряється шляхом:

- ручного тестування – введенням граничних значень у поля, які можна редагувати;
- ручного тестування на відповідність функціональним вимогам;
- статичного тестування коду;
- тестування застосунку в різних web-браузерах;
- тестування при максимальному навантаженні;
- тестування стабільності роботи при різних умовах;
- тестування зручності використання;
- тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ УПРАВЛІННЯ ТА СИСТЕМАТИЗАЦІЇ КОЛЕКТИВНИХ ЗНАНЬ

Опис програми

КПІ.ІП-6306.045440.05.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І.І. Вітковська

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Б.С. Верещак

Київ – 2020 року

Тексти програмного коду

Веб-застосування для управління та систематизації колективних знань

(Найменування програми (документа))

DVD-R

(Вид носія даних)

36 арк, 885 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

1 КОД СЕРВЕРНОГО ЗАСТОСУНКУ

```
[Route("api/articles")]
[ApiController]
[Authorize]
public class ArticlesController : ControllerBase
{
	private readonly ApplicationDbContext _context;

	private readonly IMapper _mapper;

	public ArticlesController(ApplicationDbContext context, IMapper mapper)
	{
		_context = context;
		_mapper = mapper;
	}

	// GET: api/articles/5
	[HttpGet("{articleId}")]
	public async Task<ActionResult<ArticleViewModel>> GetArticle(int articleId)
	{
		var article = await _context.Articles
			.Where(a => a.Id == articleId)
			.Include(a => a.LastAuthor)
			.ProjectTo<ArticleViewModel>(_mapper.ConfigurationProvider)
			.FirstOrDefaultAsync();

		if (article == null)
		{
			return NotFound();
		}

		return article;
	}

	// GET: api/articles/5/history
	[HttpGet("{articleId}/history")]
	public async Task<ActionResult<IEnumerable<ArticleHistoryViewModel>>> GetArticleHistory(int
articleId)
	{
		var articleHistory = await _context.ArticlesHistory
			.Where(ah => ah.ArticleId == articleId)
			.Include(ah => ah.Author)
			.OrderByDescending(ah => ah.Modified)
			.ProjectTo<ArticleHistoryViewModel>(_mapper.ConfigurationProvider)
			.ToListAsync();

		return articleHistory;
	}

	// GET: api/articles/5/comments
	[HttpGet("{articleId}/comments")]
	public async Task<ActionResult<IEnumerable<CommentViewModel>>> GetArticleComments(int articleId)
	{
		var comments = await _context.Comments
			.Where(c => c.ArticleId == articleId)
			.Include(a => a.Author)
			.OrderByDescending(c => c.Created)
			.ProjectTo<CommentViewModel>(_mapper.ConfigurationProvider)
			.ToListAsync();

		return comments;
	}

	// POST: api/articles/5/comments
	[HttpPost("{articleId}/comments")]
	public async Task<IActionResult> AddCommentToArticle(int articleId, CommentCreationModel
commentCreationModel)
	{
		string currentUserId = User.GetCurrentUserId();

		var comment = _mapper.Map<Comment>(commentCreationModel);
		comment.ArticleId = articleId;
		comment.Created = DateTime.UtcNow;
```

```
                        comment.AuthorId = currentUserId;

                        _context.Comments.Add(comment);
                        await _context.SaveChangesAsync();

                        return Ok(new { comment.Id });
                }

                // PUT: api/articles/5
                [HttpPut("{articleId}")]
                public async Task<IActionResult> PutArticle(int articleId, ArticleEditModel articleEditModel)
                {
                        var article = await _context.Articles.FindAsync(articleId);

                        if (article == null)
                                return NotFound();

                        if (article.Title == articleEditModel.Title && article.Content ==
articleEditModel.Content)
                                return StatusCode(304); //      Not Modified


                        var articleHistory = new ArticleHistory
                        {
                                ArticleId = article.Id,
                                Title = article.Title,
                                Content = article.Content,
                                AuthorId = article.LastAuthorId,
                                Modified = article.LastModified
                        };

                        string currentUserId = User.GetCurrentUserId();

                        article.Title = articleEditModel.Title;
                        article.Content = articleEditModel.Content;
                        article.LastAuthorId = currentUserId;
                        article.LastModified = DateTime.UtcNow;

                        _context.Entry(article).State = EntityState.Modified;

                        _context.ArticlesHistory.Add(articleHistory);

                        await _context.SaveChangesAsync();

                        return NoContent();
                }

                // POST: api/articles
                [HttpPost]
                public async Task<IActionResult> PostArticle(ArticleCreationModel articleCreationModel)
                {
                        var articleCategory = await
_context.Categories.FindAsync(articleCreationModel.CategoryId);

                        if (articleCategory == null)
                        {
                                return BadRequest("Not found article category");
                        }

                        var article = _mapper.Map<Article>(articleCreationModel);

                        string currentUserId = User.GetCurrentUserId();

                        article.LastModified = DateTime.UtcNow;
                        article.LastAuthorId = currentUserId;

                        _context.Articles.Add(article);
                        await _context.SaveChangesAsync();

                        return Ok(new { article.Id });
                }

                // DELETE: api/articles/5
                [HttpDelete("{articleId}")]
                public async Task<IActionResult> DeleteArticle(int articleId)
```

```
		{
			var article = await _context.Articles.FindAsync(articleId);

			if (article == null)
			{
				return NotFound();
			}

			_context.Articles.Remove(article);
			await _context.SaveChangesAsync();

			return NoContent();
		}
	}

	[Route("api/categories")]
	[ApiController]
	[Authorize]
	public class CategoriesController : ControllerBase
	{
		private readonly ApplicationDbContext _context;

		private readonly IMapper _mapper;

		public CategoriesController(ApplicationDbContext context, IMapper mapper)
		{
			_context = context;
			_mapper = mapper;
		}

		// GET: api/categories/5/articles
		[HttpGet("{categoryId}/articles")]
		public async Task<ActionResult<ArticleTitleViewModel>> GetCategoryArticles(int categoryId)
		{
			var articles = await _context.Articles
				.Where(c => c.Id == categoryId)
				.ProjectTo<ArticleTitleViewModel>(_mapper.ConfigurationProvider)
				.ToListAsync();

			return Ok(articles);
		}

		// PUT: api/categories/5
		[HttpPut("{categoryId}")]
		public async Task<IActionResult> PutCategory(int categoryId, CategoryEditModel
categoryEditModel)
		{
			var category = _mapper.Map<Category>(categoryEditModel);
			category.Id = categoryId;

			_context.Entry(category).State = EntityState.Modified;

			try
			{
				await _context.SaveChangesAsync();
			}
			catch (DbUpdateConcurrencyException)
			{
				if (!CategoryExists(categoryId))
				{
					return NotFound();
				}
				else
				{
					throw;
				}
			}

			return NoContent();
		}

		// POST: api/categories
		[HttpPost]
		public async Task<IActionResult> PostCategory(CategoryCreationModel categoryCreationModel)
		{
```

```
                        if (categoryCreationModel.ParentCategoryId.HasValue)
                        {
                                var parentCategory = await
_context.Categories.FindAsync(categoryCreationModel.ParentCategoryId);

                                if (parentCategory == null)
                                        return NotFound("Not found parent category");

                                if (parentCategory.GroupId != categoryCreationModel.GroupId)
                                        return BadRequest("Parent and current categories must be in the same
group");
                        }

                        var category = _mapper.Map<Category>(categoryCreationModel);

                        _context.Categories.Add(category);
                        await _context.SaveChangesAsync();

                        return Ok(new { category.Id });
                }

                // DELETE: api/categories/5
                [HttpDelete("{categoryId}")]
                public async Task<IActionResult> DeleteCategory(int categoryId)
                {
                        var category = await _context.Categories.FindAsync(categoryId);
                        if (category == null)
                        {
                                return NotFound();
                        }

                        try
                        {
                                _context.Categories.Remove(category);
                                await _context.SaveChangesAsync();
                        }
                        catch (DbUpdateException)
                        {
                                return UnprocessableEntity();
                        }

                        return NoContent();
                }

                private bool CategoryExists(int id)
                {
                        return _context.Categories.Any(e => e.Id == id);
                }
        }

        [Route("api/groups")]
        [ApiController]
        [Authorize]
        public class GroupsController : ControllerBase
        {
                private readonly ApplicationDbContext _context;

                private readonly IMapper _mapper;

                public GroupsController(ApplicationDbContext context, IMapper mapper)
                {
                        _context = context;
                        _mapper = mapper;
                }

                #region GET
                // GET: api/groups
                [HttpGet]
                public async Task<ActionResult<IEnumerable<GroupPreViewModel>>> GetGroups()
                {
                        string currentUserId = User.GetCurrentUserId();

                        var groups = await _context.GroupUsers
                                .Where(gu => gu.UserId == currentUserId)
                                .Include(gu => gu.Group)
```

```
                                .Select(gu => gu.Group)
                                .OrderBy(g => g.Name)
                                .ProjectTo<GroupPreViewModel>(_mapper.ConfigurationProvider)
                                .ToListAsync();

                        return groups;
                }

                // GET: api/groups/5
                [HttpGet("{groupId}")]
                public async Task<ActionResult<GroupViewModel>> GetGroup(int groupId)
                {
                        var @group = await _context.Groups.FindAsync(groupId);

                        if (@group == null)
                                return NotFound();

                        var groupViewModel = _mapper.Map<GroupViewModel>(@group);

                        return groupViewModel;
                }

                // GET: api/groups/5/users
                [HttpGet("{groupId}/users")]
                public async Task<ActionResult<IEnumerable<UserPreViewModel>>> GetGroupUsers(int groupId)
                {

                        var users = await _context.GroupUsers
                                .Where(gu => gu.GroupId == groupId)
                                .Include(gu => gu.User)
                                .Select(gu => new UserPreViewModel { Id = gu.UserId, UserName =
gu.User.UserName, RoleId = gu.RoleId })
                                .OrderBy(u => u.RoleId)
                                .ThenBy(u => u.UserName)
                                .ToListAsync();

                        return users;
                }

                // GET: api/groups/5/currentuserstatus
                [HttpGet("{groupId}/currentuserstatus")]
                public async Task<ActionResult<UserPreViewModel>> GetGroupCurrentUserStatusIn(int groupId)
                {
                        string currentUserId = User.GetCurrentUserId();

                        var user = await _context.GroupUsers
                                .Where(gu => gu.GroupId == groupId && gu.UserId == currentUserId)
                                .Include(gu => gu.User)
                                .Select(gu => new UserPreViewModel { Id = gu.UserId, UserName =
gu.User.UserName, RoleId = gu.RoleId })
                                .FirstOrDefaultAsync();

                        if (user == default)
                                return NotFound("User not found in group");

                        return user;
                }

                // GET: api/groups/5/categories
                [HttpGet("{groupId}/categories")]
                public async Task<ActionResult<IEnumerable<CategoryViewModel>>> GetGroupCategories(int groupId)
                {
                        var categories = await _context.Categories
                                .Where(c => c.GroupId == groupId)
                                .Include(c => c.Articles)
                                .ProjectTo<CategoryViewModel>(_mapper.ConfigurationProvider)
                                .ToListAsync();

                        return categories;
                }

                // GET: api/groups/5/articles
                [HttpGet("{groupId}/articles")]
                public async Task<ActionResult<IEnumerable<ArticleViewModel>>> GetGroupArticles(int groupId,
```

```
                                                        [FromQuery][Range(1, int.MaxValue), Required]int
page,
                                                        [FromQuery]string search = null,
                                                        [FromQuery]int sort = 0
                                                        )
                {
                        var query = GetBaseGroupArticlesQuery(groupId, search);

                        if (sort == 1)      // LastModified Asc
                        {
                                query = query.OrderBy(a => a.LastModified);
                        }
                        else if (sort == 2)     // LastModified Desc
                        {
                                query = query.OrderByDescending(a => a.LastModified);
                        }
                        else if (sort == 3)     // Title Asc
                        {
                                query = query.OrderBy(a => a.Title);
                        }
                        else if (sort == 4)     // Title Desc
                        {
                                query = query.OrderByDescending(a => a.Title);
                        }

                        query = query.Skip((page - 1) * 5).Take(5);

                        var projectedQuery = query.ProjectTo<ArticleViewModel>(_mapper.ConfigurationProvider);

                        var articles = await projectedQuery.ToListAsync();

                        return articles;
                }

                // GET: api/groups/5/articles/count
                [HttpGet("{groupId}/articles/count")]
                public async Task<IActionResult> GetGroupArticlesCount(int groupId, [FromQuery]string search =
null)
                {
                        var query = GetBaseGroupArticlesQuery(groupId, search);

                        int articlesCount = await query.CountAsync();

                        return Ok(new { articlesCount });
                }

                private IQueryable<Article> GetBaseGroupArticlesQuery(int groupId, string search)
                {
                        var query = _context.Articles
                                .Include(a => a.Category)
                                .Where(a => a.Category.GroupId == groupId);

                        if (!string.IsNullOrWhiteSpace(search))
                        {
                                string normalizedSearchString = System.Web.HttpUtility.HtmlEncode(search);

                                normalizedSearchString =
System.Text.RegularExpressions.Regex.Replace(normalizedSearchString, @"\s+", " ")
                                        .Replace("%", "[%]")
                                        .Replace("*", "[*]")
                                        .Trim();

                                string searchPattern = $"%{normalizedSearchString}%";

                                query = query.Where(a => EF.Functions.Like(a.Title, searchPattern) ||
EF.Functions.Like(a.Content, searchPattern));
                        }

                        return query;
                }
```

```
        #endregion

        #region PUT
        // PUT: api/groups/5
        [HttpPut("{groupId}")]
        public async Task<IActionResult> PutGroup(int groupId, GroupEditModel groupEditModel)
        {
                var @group = _mapper.Map<Group>(groupEditModel);
                @group.Id = groupId;

                _context.Entry(@group).State = EntityState.Modified;

                try
                {
                        await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                        if (!GroupExists(groupId))
                        {
                                return NotFound();
                        }
                        else
                        {
                                throw;
                        }
                }

                return NoContent();
        }
        #endregion

        #region POST
        // POST: api/groups
        [HttpPost]
        public async Task<IActionResult> PostGroup(GroupCreationModel groupCreationModel)
        {
                var @group = _mapper.Map<Group>(groupCreationModel);

                _context.Groups.Add(@group);
                await _context.SaveChangesAsync();

                string currentUserId = User.GetCurrentUserId();
                _context.GroupUsers.Add(new GroupUserRole { GroupId = @group.Id, UserId = currentUserId,
RoleId = 1 });
                await _context.SaveChangesAsync();

                return Ok(new { @group.Id });
        }

        // POST: api/groups/5/users
        [HttpPost("{groupId}/users")]
        public async Task<IActionResult> AddUsersToGroup(int groupId, [FromBody]UserSearchData
userSearchData)
        {
                string normalizeduserName = userSearchData.UserName.ToUpper();

                string userId = await _context.Users
                        .Where(u => u.NormalizedUserName == normalizeduserName)
                        .Select(u => u.Id)
                        .FirstOrDefaultAsync();

                if (userId == default)
                        return NotFound("User not found");

                var groupUserRole = new GroupUserRole { GroupId = groupId, UserId = userId, RoleId = 3
};


                _context.GroupUsers.Add(groupUserRole);
                await _context.SaveChangesAsync();

                return Ok();
        }

        // POST: api/groups/5/admins
```

```
		[HttpPost("{groupId}/admins")]
		public async Task<IActionResult> GrantAdministratorRights(int groupId, UserIdData
userSearchData)
		{
			var groupUser = await _context.GroupUsers.FirstOrDefaultAsync(gu => gu.GroupId ==
groupId && gu.UserId == userSearchData.UserId);

			if (groupUser == default)
				return NotFound("User not found in group");

			groupUser.RoleId = 1;

			_context.Entry(groupUser).State = EntityState.Modified;

			await _context.SaveChangesAsync();

			return Ok();
		}
		#endregion

		#region DELETE
		// DELETE: api/groups/5
		[HttpDelete("{groupId}")]
		public async Task<IActionResult> DeleteGroup(int groupId)
		{
			var @group = await _context.Groups.FindAsync(groupId);

			if (@group == null)
			{
				return NotFound();
			}

			_context.Groups.Remove(@group);
			await _context.SaveChangesAsync();

			return NoContent();
		}

		// DELETE: api/groups/5/users/3
		[HttpDelete("{groupId}/users/{userId}")]
		public async Task<IActionResult> RemoveUserFromGroup(int groupId, string userId)
		{
			var groupUser = await _context.GroupUsers.FindAsync(groupId, userId);

			if (groupUser == null)
			{
				return NotFound();
			}

			_context.GroupUsers.Remove(groupUser);
			await _context.SaveChangesAsync();

			return NoContent();
		}
		#endregion

		private bool GroupExists(int id)
		{
			return _context.Groups.Any(e => e.Id == id);
		}
	}

	[Route("api/images")]
	[ApiController]
	public class ImagesController : ControllerBase
	{
		private const string imagesDirectory = "Images";

		private readonly IWebHostEnvironment _webHostEnvironment;

		public ImagesController(IWebHostEnvironment webHostEnvironment)
		{
			_webHostEnvironment = webHostEnvironment;
		}
```

```
                // GET: api/images/12345.jpg
                [HttpGet("{imageName}")]
                public IActionResult Get(string imageName)
                {
                        string path = GetPathToImage(imageName);

                        if (!System.IO.File.Exists(path))
                        {
                                return BadRequest(new { error = new { message = "Image does not exist." } });
                        }

                        var stream = System.IO.File.OpenRead(path);

                        return File(stream, "image/ief");
                }

                // POST: api/images
                [HttpPost]
                public async Task<IActionResult> Post(IFormFile formFile = null)
                {
                        if (formFile == null)
                        {
                                var files = Request.Form.Files;

                                if (files == null || files.Count == 0)
                                {
                                        return BadRequest(new { error = new { message = "Empty data." } });
                                }

                                if (files.Count > 1)
                                {
                                        return BadRequest(new { error = new { message = "Only one image is
allowed." } });
                                }

                                formFile = files[0];
                        }

                        string imageName = $"{Guid.NewGuid()}_{formFile.FileName}";

                        try
                        {
                                string path = GetPathToImage(imageName);

                                using (var fileStream = new FileStream(path, FileMode.Create))
                                {
                                        await formFile.CopyToAsync(fileStream);
                                }

                                return Ok(new { url = $"https://{Request.Host}/api/images/{imageName}" });
                        }
                        catch (Exception ex)
                        {

                                return StatusCode(StatusCodes.Status500InternalServerError, new { error = new {
message = ex.Message } });
                        }
                }

                private string GetPathToImage(string imageName)
                {
                        return Path.Combine(_webHostEnvironment.ContentRootPath, imagesDirectory, imageName);
                }
        }

        public class OidcConfigurationController : Controller
        {
                private readonly ILogger<OidcConfigurationController> _logger;

                public OidcConfigurationController(IClientRequestParametersProvider
clientRequestParametersProvider, ILogger<OidcConfigurationController> logger)
                {
                        ClientRequestParametersProvider = clientRequestParametersProvider;
                        _logger = logger;
                }
```

```
                public IClientRequestParametersProvider ClientRequestParametersProvider { get; }

                [HttpGet("_configuration/{clientId}")]
                public IActionResult GetClientRequestParameters([FromRoute]string clientId)
                {
                        var parameters = ClientRequestParametersProvider.GetClientParameters(HttpContext,
clientId);
                        return Ok(parameters);
                }
        }

        public class ApplicationUser : IdentityUser
        {
                public List<GroupUserRole> GroupUsers { get; set; }

                public List<ArticleHistory> ArticlesHistory { get; set; }
        }

        [Table("Articles")]
        public class Article
        {
                [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
                public int Id { get; set; }

                [Required]
                [StringLength(300)]
                public string Title { get; set; }

                [Required]
                public string Content { get; set; }

                [Required]
                public int CategoryId { get; set; }

                public Category Category { get; set; }

                [Required]
                public string LastAuthorId { get; set; }

                public ApplicationUser LastAuthor { get; set; }

                [Required]
                public DateTime LastModified { get; set; }

                public List<ArticleHistory> ArticleHistory { get; set; }

                public List<Comment> Comments { get; set; }
        }

        [Table("ArticlesHistory")]
        public class ArticleHistory
        {
                [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
                public int Id { get; set; }

                [Required]
                public int ArticleId { get; set; }

                public Article Article { get; set; }

                [Required]
                [StringLength(300)]
                public string Title { get; set; }

                [Required]
                public string Content { get; set; }

                public string AuthorId { get; set; }

                public ApplicationUser Author { get; set; }

                [Required]
                public DateTime Modified { get; set; }
        }
```

```
[Table("Categories")]
public class Category
{
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Required]
        [StringLength(200)]
        public string Name { get; set; }

        public int? ParentCategoryId { get; set; }

        public Category ParentCategory { get; set; }

        [Required]
        public int GroupId { get; set; }

        public Group Group { get; set; }

        public List<Article> Articles { get; set; }
}

[Table("Comments")]
public class Comment
{
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Required]
        [StringLength(2_000)]
        public string Text { get; set; }

        [Required]
        public int ArticleId { get; set; }

        public Article Article { get; set; }

        public string AuthorId { get; set; }

        public ApplicationUser Author { get; set; }

        [Required]
        public DateTime Created { get; set; }
}

[Table("Groups")]
public class Group
{
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Required]
        [StringLength(300)]
        public string Name { get; set; }

        [StringLength(1_000)]
        public string Description { get; set; }

        public List<GroupUserRole> GroupUsers { get; set; }

        public List<Category> Categories { get; set; }
}

[Table("GroupUserRoles")]
public class GroupUserRole
{
        [Required]
        public int GroupId { get; set; }
        public Group Group { get; set; }

        [Required]
        public string UserId { get; set; }
        public ApplicationUser User { get; set; }
```

```
                public int? RoleId { get; set; }
        }

                public class ApplicationDbContext : ApiAuthorizationDbContext<ApplicationUser>
        {
                public DbSet<Article> Articles { get; set; }

                public DbSet<Category> Categories { get; set; }

                public DbSet<Group> Groups { get; set; }

                public DbSet<GroupUserRole> GroupUsers { get; set; }

                public DbSet<ArticleHistory> ArticlesHistory { get; set; }

                public DbSet<Comment> Comments { get; set; }

                public ApplicationDbContext(
                        DbContextOptions options,
                        IOptions<OperationalStoreOptions> operationalStoreOptions) : base(options,
operationalStoreOptions)
                {

                }

                protected override void OnModelCreating(ModelBuilder modelBuilder)
                {
                        base.OnModelCreating(modelBuilder);

                        // Configure many-to-many
                        modelBuilder.Entity<GroupUserRole>().HasKey(gg => new { gg.GroupId, gg.UserId });
                }
        }

        public class ArticleCreationModel
        {
                [Required]
                [StringLength(300)]
                public string Title { get; set; }

                [Required]
                [StringLength(50_000)]
                public string Content { get; set; }

                [Required]
                [Range(1, int.MaxValue)]
                public int CategoryId { get; set; }
        }

        public class ArticleEditModel
        {
                [Required]
                [StringLength(300)]
                public string Title { get; set; }

                [Required]
                [StringLength(50_000)]
                public string Content { get; set; }
        }

        public class ArticleHistoryViewModel
        {
                public string Title { get; set; }

                public string Content { get; set; }

                public UserPreViewModel Author { get; set; }

                public DateTime Modified { get; set; }
        }

        public class ArticlePreViewModel
        {
                public int Id { get; set; }
```

```
        public string Title { get; set; }

        public UserPreViewModel LastAuthor { get; set; }

        public DateTime LastModified { get; set; }

        public CategoryPreViewModel Category { get; set; }
}

public class ArticleTitleViewModel
{
        public int Id { get; set; }

        public string Title { get; set; }
}

public class ArticleViewModel
{
        public int Id { get; set; }

        public string Title { get; set; }

        public string Content { get; set; }

        public UserPreViewModel LastAuthor { get; set; }

        public DateTime LastModified { get; set; }
}

public class CategoryCreationModel
{
        [Required]
        [StringLength(200)]
        public string Name { get; set; }

        [Range(1, int.MaxValue)]
        public int? ParentCategoryId { get; set; }

        [Required]
        [Range(1, int.MaxValue)]
        public int GroupId { get; set; }
}

public class CategoryEditModel
{
        [Required]
        [StringLength(200)]
        public string Name { get; set; }
}

public class CategoryPreViewModel
{
        public int Id { get; set; }

        public string Name { get; set; }

        public CategoryPreViewModel ParentCategory { get; set; }
}

public class CategoryViewModel
{
        public int Id { get; set; }

        public string Name { get; set; }

        public int? ParentCategoryId { get; set; }

        public List<ArticleTitleViewModel> Articles { get; set; }
}

public class CommentCreationModel
{
        public string Text { get; set; }
}
```

```
public class CommentViewModel
{
        public int Id { get; set; }

        public string Text { get; set; }

        public string AuthorId { get; set; }

        public UserPreViewModel Author { get; set; }

        public DateTime Created { get; set; }
}

public class GroupCreationModel
{
        [Required]
        [StringLength(300)]
        public string Name { get; set; }

        [StringLength(1_000)]
        public string Description { get; set; }
}

public class GroupEditModel
{
        [Required]
        [StringLength(300)]
        public string Name { get; set; }

        [StringLength(1_000)]
        public string Description { get; set; }
}

public class GroupPreViewModel
{
        public int Id { get; set; }

        public string Name { get; set; }
}

public class GroupViewModel
{
        public int Id { get; set; }

        public string Name { get; set; }

        public string Description { get; set; }
}

public class UserIdData
{
        [Required]
        public string UserId { get; set; }
}

public class UserPreViewModel
{
        public string Id { get; set; }

        public string UserName { get; set; }

        public int? RoleId { get; set; }
}

public class UserSearchData
{
        [Required]
        public string UserName { get; set; }
}

public class Program
{
        public static void Main(string[] args)
        {
                CreateHostBuilder(args).Build().Run();
```

```
		}

		public static IHostBuilder CreateHostBuilder(string[] args) =>
			Host.CreateDefaultBuilder(args)
				.ConfigureWebHostDefaults(webBuilder =>
				{
					webBuilder.UseStartup<Startup>();
				});
	}

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

		public IConfiguration Configuration { get; }

		// This method gets called by the runtime. Use this method to add services to the container.
		public void ConfigureServices(IServiceCollection services)
		{
			services.AddDbContext<ApplicationDbContext>(options =>
				options.UseSqlServer(
					Configuration.GetConnectionString("DefaultConnection")));

			services.AddDefaultIdentity<ApplicationUser>(options =>
options.SignIn.RequireConfirmedAccount = true)
				.AddEntityFrameworkStores<ApplicationDbContext>();

			services.AddIdentityServer()
				.AddApiAuthorization<ApplicationUser, ApplicationDbContext>();

			services.AddAuthentication()
				.AddIdentityServerJwt();

			services.AddControllersWithViews();
			services.AddRazorPages();

			// In production, the React files will be served from this directory
			services.AddSpaStaticFiles(configuration =>
			{
				configuration.RootPath = "ClientApp/build";
			});

			services.AddAutoMapper(typeof(Startup));

			// Register the Swagger generator, defining 1 or more Swagger documents
			services.AddSwaggerGen(c =>
			{
				c.SwaggerDoc("v1", new OpenApiInfo { Title = "KB API", Version = "v1" });

				// Authentication in Swagger
				c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
				{
					Name = "Authorization",
					Type = SecuritySchemeType.ApiKey,
					Scheme = "Bearer",
					BearerFormat = "JWT",
					In = ParameterLocation.Header,
					Description = "JWT Authorization header using the Bearer scheme."
				});

				c.AddSecurityRequirement(new OpenApiSecurityRequirement
				{
					{
						new OpenApiSecurityScheme
						{
							Reference = new OpenApiReference
							{
								Type = ReferenceType.SecurityScheme,
								Id = "Bearer"
							},
						},
						new string[] {}
```

```
                                        }
                                });
                        });

                        services.AddCors();
                }

                // This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
                public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
                {
                        app.UseCors(builder => builder.AllowAnyOrigin());

                        app.UseMiddleware<RequestLoggingMiddleware>();

                        // Enable middleware to serve generated Swagger as a JSON endpoint.
                        app.UseSwagger();

                        // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
                        // specifying the Swagger JSON endpoint.
                        app.UseSwaggerUI(c =>
                        {
                                c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
                        });

                        if (env.IsDevelopment())
                        {
                                app.UseDeveloperExceptionPage();
                                app.UseDatabaseErrorPage();
                        }
                        else
                        {
                                app.UseExceptionHandler("/Error");
                                // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
                                app.UseHsts();
                        }

                        app.UseHttpsRedirection();
                        app.UseStaticFiles();
                        app.UseSpaStaticFiles();

                        app.UseRouting();

                        app.UseAuthentication();
                        app.UseIdentityServer();
                        app.UseAuthorization();
                        app.UseEndpoints(endpoints =>
                        {
                                endpoints.MapControllerRoute(
                                        name: "default",
                                        pattern: "{controller}/{action=Index}/{id?}");
                                endpoints.MapRazorPages();
                        });

                        app.UseSpa(spa =>
                        {
                                spa.Options.SourcePath = "ClientApp";

                                if (env.IsDevelopment())
                                {
                                        spa.UseReactDevelopmentServer(npmScript: "start");
                                }
                        });
                }
        }
```

2 КОД КЛІЄНТСЬКОГО ЗАСТОСУНКУ

```
export class ApiService {

async getRequest(url) {
        const token = await authService.getAccessToken();
        return await fetch(
                url,
                {
                        headers: !token ? {} : { 'Authorization': `Bearer ${token}` }
                }
        );
}

async putRequest(url = '', data = {}) {
        const token = await authService.getAccessToken();
        const response = await fetch(url,
                {
                        method: 'PUT',
                        headers: !token ? {} : { 'Authorization': `Bearer ${token}`, 'Content-Type':
'application/json' },
                        body: JSON.stringify(data)
                });
        return response;
}

async postRequest(url = '', data = {}) {
        const token = await authService.getAccessToken();
        const response = await fetch(url,
                {
                        method: 'POST',
                        headers: !token ? {} : { 'Authorization': `Bearer ${token}`, 'Content-Type':
'application/json' },
                        body: JSON.stringify(data)
                });
        return response;
}

async deleteRequest(url = '') {
        const token = await authService.getAccessToken();
        const response = await fetch(url,
                {
                        method: 'DELETE',
                        headers: !token ? {} : { 'Authorization': `Bearer ${token}`, 'Content-Type':
'application/json' }
                });
        return response;
}

    static get instance() { return apiService }
}

const apiService = new ApiService();

export default apiService;

export default class ArticleEdit extends React.Component {

constructor(props) {
        super(props);
        this.state = {
                articleEditFormOpen: false,
                article: {}
        };
}

async handleArticleEditFormOpen() {
        this.state.article.title = this.props.article.title;

        this.state.article.content = this.props.article.content;
```

```
                await this.setState({ articleEditFormOpen: true });
        }

        async handleArticleEditFormClose() {
                await this.setState({ articleEditFormOpen: false });
        }

        async updateArticleData() {
                var response = await apiService.putRequest(`api/articles/${this.props.articleId}`,
this.state.article);

                if (response.ok) {
                        await Promise.all([
                                this.props.populateCategoriesAndArticlesData(),
                                this.props.populateArticleData(this.props.articleId)
                        ]);

                        this.handleArticleEditFormClose();

                        this.props.showAlert('The article has been updated successfully', 'success');
                } else {
                        this.props.showAlert('Error occurred during update', 'error');
                }
        }

        render() {
                return (
                        <div>
                                <Button
                                        variant="contained"
                                        color="primary"
                                        onClick={() => this.handleArticleEditFormOpen()}
                                        style={{ margin: "5px" }}
                                        startIcon={<EditIcon />}
                                        size="small"
                                >
                                        Edit
                                </Button>
                                <Dialog open={this.state.articleEditFormOpen} onClose={() =>
this.handleArticleEditFormClose()} aria-labelledby="form-dialog-title" maxWidth="lg" fullWidth>
                                        <DialogTitle id="form-dialog-title">Edit Article</DialogTitle>
                                        <DialogContent>
                                                <TextField
                                                        autoFocus
                                                        fullWidth
                                                        margin="dense"
                                                        autoComplete="off"
                                                        id="outlined-required"
                                                        label="Title"
                                                        defaultValue={this.props.article.title}
                                                        onChange={(event) => { this.state.article.title =
event.target.value; }}
                                                />
                                                <Divider style={{ marginBottom: "10px", marginTop: "10px" }} />
                                                <CKEditor
                                                        editor={ClassicEditor}
                                                        data={this.props.article.content}
                                                        onChange={(event, editor) => {
                                                                const data = editor.getData();
                                                                this.state.article.content = data
                                                        }}
                                                />
                                        </DialogContent>
                                        <DialogActions>
                                                <Button onClick={() => this.handleArticleEditFormClose()}
color="primary">
                                                        Cancel
                                                </Button>
                                                <Button onClick={() => this.updateArticleData()} color="primary"
startIcon={<SaveIcon />}>
                                                        Save
                                                </Button>
                                        </DialogActions>
                                </Dialog>
                        </div >
```

```
                );
        }
        }

        export default class ArticleHistory extends React.Component {

        constructor(props) {
                super(props);
                this.state = {
                        articleHistoryOpen: false,
                        articleHistory: []
                };
        }

        async handleArticlesHistoryOpen() {
                await this.setState({ articleHistoryOpen: true });

                await this.populateArticleHistoryData();
        }

        async handleArticlesHistoryClose() {
                await this.setState({ articleHistoryOpen: false });
        }

        async populateArticleHistoryData() {
                let url = `api/articles/${this.props.selectedArticleId}/history`;

                let response = await apiService.getRequest(url);

                if (response.ok) {
                        let articleHistoryData = await response.json();

                        await this.setState({ articleHistory: articleHistoryData });

                } else {
                        this.props.showAlert('Error occurred during fetching data', 'error');
                }
        }

        render() {
                return (
                        <div>
                                <Button
                                        variant="contained"
                                        color="primary"
                                        onClick={() => this.handleArticlesHistoryOpen()}
                                        style={{ margin: "5px" }}
                                        startIcon={<HistoryIcon />}
                                        size="small"
                                >
                                        Article history
                                </Button>
                                <Dialog open={this.state.articleHistoryOpen} onClose={() =>
this.handleArticlesHistoryClose()} aria-labelledby="form-dialog-title" maxWidth="lg" fullWidth>
                                        <DialogTitle id="form-dialog-title">Article History</DialogTitle>
                                        <DialogContent>
                                                <div horizontal="true" layout="true" style={{ marginBottom:
"25px" }}>

                                                </div>
                                                {
                                                        this.state.articleHistory.length === 0
                                                                ? <p style={{ margin: "50px", fontSize: "20px",
color: "gray", textAlign: "center" }}>No changes ...</p>
                                                                : this.state.articleHistory.map(a =>
                                                                        <Paper elevation={4} style={{ padding:
"inherit", marginBottom: "30px", marginTop: "15px" }}>
                                                                                <h3>{a.title}</h3>
                                                                                <p style={{ margin: "5px",
color: "gray", textAlign: "end" }}>Modified <em>{new Date(a.modified + 'Z').toLocaleString()}</em> by
<em>{a.author.userName}</em></p>
                                                                                <CKEditor
                                                                                        editor={ClassicEditor}
                                                                                        disabled
                                                                                        data={a.content}
```

```
                                                                                config={{
                                                                                        toolbar: []
                                                                                }}
                                                                        />
                                                                </Paper>
                                                        )}
                                        </DialogContent>
                                        <DialogActions>
                                                <Button onClick={() => this.handleArticlesHistoryClose()}
color="primary">
                                                        Back
                                                </Button>
                                        </DialogActions>
                                </Dialog>
                        </div >
                );
        }
        }

        export default class ArticlesManager extends React.Component {

        constructor(props) {
                super(props);
                this.state = {
                        articleManagerOpen: false,
                        articles: [],
                        searchPhrase: '',
                        articlesCount: 0,
                        sortArticlesOption: 2,
                        currentPage: 1
                };
        }

        async handleArticlesManagerOpen() {
                await this.setState({ articleManagerOpen: true });

                await this.populateArticlesData();
        }

        async handleArticlesManagerClose() {
                await this.setState({ articleManagerOpen: false });
        }

        async populateArticlesData() {
                let url = `api/groups/${this.props.selectedGroupId}/articles?page=${this.state.currentPage}`;

                let search = this.state.searchPhrase.trim()
                if (search !== '') {
                        url += `&search=${search}`;
                }

                if (this.state.currentPage === 1) {
                        this.populateArticlesCountData(url);
                }

                if (this.state.sortArticlesOption > 0) {
                        url += `&sort=${this.state.sortArticlesOption}`;
                }

                let response = await apiService.getRequest(url);

                if (response.ok) {
                        let articlesData = await response.json();

                        await this.setState({ articles: articlesData });

                } else {
                        this.props.showAlert('Error occurred during fetching data', 'error');
                }
        }

        async populateArticlesCountData(origUrl) {
                let url = origUrl.replace('?', '/count?');
                let response = await apiService.getRequest(url);
```

```
                if (response.ok) {
                        let responseData = await response.json();
                        let articlesCountLocal = responseData.articlesCount;

                        this.setState({ articlesCount: articlesCountLocal });
                } else {
                        this.setState({ articlesCount: 0 });
                }

        }

        render() {
                return (
                        <div>
                                <Button
                                        variant="contained"
                                        color="primary"
                                        onClick={() => this.handleArticlesManagerOpen()}
                                        style={{ margin: "5px" }}
                                        startIcon={<ViewListIcon />}
                                        size="small"
                                >
                                        Articles Manager
                                </Button>
                                <Dialog open={this.state.articleManagerOpen} onClose={() =>
this.handleArticlesManagerClose()} aria-labelledby="form-dialog-title" maxWidth="lg" fullWidth>
                                        <DialogTitle id="form-dialog-title">Articles Manager</DialogTitle>
                                        <DialogContent>
                                                <div horizontal="true" layout="true" style={{ marginBottom:
"25px" }}>
                                                        <TextField
                                                                autoFocus
                                                                margin="dense"
                                                                style={{ width: "70%" }}
                                                                id="outlined-required"
                                                                label="Serach"
                                                                onChange={(event) => { this.setState({
searchPhrase: event.target.value }); }}
                                                        />
                                                        <Button startIcon={<SearchIcon />} style={{
verticalAlign: "bottom", margin: "5px" }}
                                                                size="small" color="primary" variant="contained"
                                                                onClick={async () => {
                                                                        await this.setState({ currentPage: 1 });
                                                                        this.populateArticlesData()
                                                                }}
                                                        >
                                                                Search
                                                        </Button>
                                                        <FormControl style={{ verticalAlign: "bottom", margin:
"5px" }}>
                                                                <InputLabel id="sort-articles-label">Sort
by</InputLabel>
                                                                <Select
                                                                        labelId="sort-articles-label"
                                                                        id="sort-articles-select"
                                                                        value={this.state.sortArticlesOption}
                                                                        onChange={async (event) => {
                                                                                await this.setState({
sortArticlesOption: event.target.value });
                                                                                await this.setState({
currentPage: 1 });
                                                                                this.populateArticlesData();
                                                                        }}
                                                                >
                                                                        <MenuItem value={2}>Last modified -
(Newest)</MenuItem>
                                                                        <MenuItem value={1}>Last modified
(Oldest)</MenuItem>
                                                                        <MenuItem value={3}>Title (A -
Z)</MenuItem>
                                                                        <MenuItem value={4}>Title (Z -
A)</MenuItem>
                                                                </Select>
                                                        </FormControl>
```

```
                                                </div>
                                                {this.state.articles.map(a =>
                                                        <Paper elevation={4} style={{ padding: "inherit",
marginBottom: "30px", marginTop: "15px" }}>
                                                                <h3>{a.title}</h3>
                                                                <p style={{ margin: "5px", color: "gray",
textAlign: "end" }}>Last modified <em>{new Date(a.lastModified + 'Z').toLocaleString()}</em> by
<em>{a.lastAuthor.userName}</em></p>
                                                                <CKEditor
                                                                        editor={ClassicEditor}
                                                                        disabled
                                                                        data={a.content}
                                                                        config={{
                                                                                toolbar: []
                                                                        }}
                                                                />
                                                        </Paper>
                                                )}
                                                <Pagination color="primary"
                                                        count={this.state.articlesCount % 5 == 0 ?
Math.floor(this.state.articlesCount / 5) : (Math.floor(this.state.articlesCount / 5) + 1)}
                                                        page={this.state.currentPage}
                                                        onChange={async (event, page) => {
                                                                await this.setState({ currentPage: page });
                                                                this.populateArticlesData();
                                                        }}
                                                />
                                        </DialogContent>
                                        <DialogActions>
                                                <Button onClick={() => this.handleArticlesManagerClose()}
color="primary">
                                                        Back
                                                </Button>
                                        </DialogActions>
                                </Dialog>
                        </div >
                );
        }
        }

        export default class Comments extends React.Component {

        constructor(props) {
                super(props);
                this.state = {
                        comments: [],
                        newCommentText: ''
                };
        }

        componentWillReceiveProps(nextProps) {
                this.populateCommentsData(nextProps.selectedArticleId);
        }

        async populateCommentsData(articleId) {
                let url = `api/articles/${articleId}/comments`;

                let response = await apiService.getRequest(url);

                if (response.ok) {
                        let commentsData = await response.json();

                        await this.setState({ comments: commentsData });

                } else {
                        this.props.showAlert('Error occurred during fetching comments data', 'error');
                }
        }

        async addComment() {
                if (this.state.newCommentText.trim() === '') {
                        this.props.showAlert('Comment text is empty', 'error');
                        return;
                }
```

```
		let url = `api/articles/${this.props.selectedArticleId}/comments`;

		let response = await apiService.postRequest(url, { text: this.state.newCommentText });

		if (response.ok) {
			this.populateCommentsData(this.props.selectedArticleId);

			this.props.showAlert('The comment has been added successfully', 'success');

		} else {
			this.props.showAlert('Error occurred during adding the comment', 'error');
		}
	}

	render() {
		return (
			<div style={{ padding: "25px" }}>
				<div style={{ margin: "auto" }} className="ui comments">
					<h5 style={{ margin: "10px", color: "DimGrey" }}>Comments</h5>
					<Divider />
					{
						this.state.comments.map(c =>
							<div className="comment" style={{ margin: "15px" }}>
								<div className="avatar">
									<AccountCircleIcon fontSize="large"
color="primary" />
								</div>
								<div className="content">
									<a
className="author">{c.author.userName}</a>
									<div className="metadata"><div>{new
Date(c.created + 'Z').toLocaleString()}</div></div>
									<div className="text">{c.text}</div>
								</div>
							</div>
						)
					}
					<TextField
						id="outlined-multiline-static"
						label="Comment"
						multiline
						fullWidth
						rows={5}
						variant="outlined"
						autoComplete="off"
						onChange={(event) => { this.setState({ newCommentText:
event.target.value }) }}
						style={{ marginTop: "15px" }}
					/>
					<Button variant="contained"
						color="primary"
						size="small"
						onClick={() => this.addComment()}
						style={{ marginTop: "10px" }}
						startIcon={<CommentIcon />}
					>
						Add comment
					</Button>
				</div>
			</div >
		);
	}
	}

	export default class GroupItem extends React.Component {

	constructor(props) {
		super(props);
		this.state = {
			groupId: 0,
			groupName: "",
			isMouseOver: false
		};
	}
```

```
        render() {
                return <tr onClick={() => this.props.handleToUpdateSelectedGroup(this.props.groupId,
this.props.groupName)}
                        onMouseOver={() => this.setState({ isMouseOver: true })}
                        onMouseOut={() => this.setState({ isMouseOver: false })}
                        className={this.props.isGroupSelected ? "table-primary" : this.state.isMouseOver ?
"table-active" : ""}>
                        <td>{this.props.groupName}</td>
                </tr>;
        }
        }

        export default class GroupUsers extends React.Component {

        constructor(props) {
                super(props);
                this.state = {
                        groupUsersFormOpen: false,
                        groupUsers: [],
                        userNameToInvite: ''
                };
        }

        handleGroupUsersFormOpen() {
                this.updateGroupUsersData();

                this.setState({ groupUsersFormOpen: true });
        }

        handleGroupUsersFormClose() {
                this.setState({ groupUsersFormOpen: false });

                this.setState({ groupUsers: [] });
        }

        async updateGroupUsersData() {
                let response = await apiService.getRequest(`api/groups/${this.props.selectedGroupId}/users`);

                if (response.ok) {
                        const groupUsersData = await response.json();

                        this.setState({ groupUsers: groupUsersData });
                } else {
                        this.props.showAlert('Something went wrong, please try later', 'error');
                }
        }

        async addNewUserToGroup() {
                let response = await apiService.postRequest(`api/groups/${this.props.selectedGroupId}/users`, {
userName: this.state.userNameToInvite });

                if (response.ok) {
                        this.updateGroupUsersData();

                        this.props.showAlert('User added to group', 'success');
                } else {
                        this.props.showAlert('User not found', 'error');
                }
        }

        async makeAdmin(groupUserId) {
                let response = await apiService.postRequest(`api/groups/${this.props.selectedGroupId}/admins`, {
userId: groupUserId });

                if (response.ok) {
                        this.updateGroupUsersData();

                        this.props.showAlert('Admin rights has been granted', 'success');
                } else {
                        this.props.showAlert('Something went wrong, please try later', 'error');
                }
        }

        async removeUserFromGroup(groupUserId) {
```

```
                let response = await
apiService.deleteRequest(`api/groups/${this.props.selectedGroupId}/users/${groupUserId}`);

                if (response.ok) {
                        this.updateGroupUsersData();

                        this.props.showAlert('User has been removed from group', 'success');
                } else {
                        this.props.showAlert('Something went wrong, please try later', 'error');
                }
        }

        render() {
                return (
                        <div>
                                <Button
                                        variant="contained"
                                        color="primary"
                                        size="small"
                                        onClick={() => this.handleGroupUsersFormOpen()}
                                        style={{ margin: "5px" }}
                                        startIcon={<GroupIcon/>}
                                >
                                        Users
                                </Button>
                                <Dialog open={this.state.groupUsersFormOpen} onClose={() =>
this.handleGroupUsersFormClose()} aria-labelledby="form-dialog-title" fullWidth={true} maxWidth="sm">
                                        <DialogTitle id="form-dialog-title">Group Users</DialogTitle>
                                        <DialogContent>
                                                <div horizontal="true" layout="true">
                                                        <TextField
                                                                autoFocus
                                                                margin="dense"
                                                                style={{ width: "75%" }}
                                                                //fullWidth={true}
                                                                autoComplete="off"
                                                                id="outlined-required"
                                                                label="Username"
                                                                onChange={(event) => { this.setState({
userNameToInvite: event.target.value }); }}
                                                        />
                                                        <Button style={{ verticalAlign: "bottom", margin: "5px"
}} size="small" color="primary" variant="contained" onClick={() => this.addNewUserToGroup()} >
                                                                Add new user
                                                        </Button>
                                                </div>
                                                <Divider style={{ marginBottom: "10px", marginTop: "10px" }} />
                                                <List component="nav" aria-label="main mailbox folders">
                                                        {this.state.groupUsers.map(u =>
                                                                <ListItem button>
                                                                        <ListItemIcon>
                                                                                {u.roleId === 1 ?
<AssignmentIndIcon color="secondary"/> : <AccountCircleIcon />}
                                                                        </ListItemIcon>
                                                                        <ListItemText primary={u.userName} />
                                                                        {
                                                                                this.props.currentUser.roleId
=== 1 && u.roleId === 3 ?

                                                                                        <div horizontal="true"
layout="true">

                                                                                                <Button style={{
margin: "5px" }} size="small" color="primary" variant="contained" onClick={() => { this.makeAdmin(u.id); }}>
                                                                                                        Make
admin

                                                                                                </Button>
                                                                                                <Button style={{

margin: "5px" }} size="small" color="secondary" variant="contained" onClick={() => {
this.removeUserFromGroup(u.id); }}>

                                                                                                Remove
                                                                                                </Button>
                                                                                        </div>
                                                                                        : ''
                                                                        }
                                                                        <Divider/>
                                                                </ListItem>)
```

```
                                                }
                                        </List>
                                </DialogContent>
                                <DialogActions>
                                        <Button onClick={() => this.handleGroupUsersFormClose()}
color="primary">
                                                Back
                                        </Button>
                                </DialogActions>
                        </Dialog>
                </div >
        );
}
}

export class KB extends Component {

static displayName = KB.name;

constructor(props) {
        super(props);
        this.state = {
                loading: true,

                groups: [],

                selectedGroupId: 0,
                selectedGroupName: '',

                currentUser: {},

                categoriesAndArticles: [],

                selectedArticleId: 0,
                selectedArticle: { lastAuthor: {} },

                articleCreationFormOpen: false,
                articleCreationData: { title: '', content: '', categoryId: 0 },

                groupCreationFormOpen: false,
                groupCreationData: { name: '', description: '' },

                categoryCreationFormOpen: false,
                categoryCreationData: { name: '', parentCategoryId: 0, groupId: 0 },
                subCategoryCreationFlag: false,

                alertOpen: false,
                alertText: '',
                alertSeverity: ''
        };
}

resetStateAfterGroupSelection() {
        this.setState({ categoriesAndArticles: [] });
        this.setState({ selectedArticleId: 0 });
        this.setState({ selectedArticle: { lastAuthor: {} } });
        this.setState({ articleCreationFormOpen: false });
        this.setState({ articleCreationData: { title: '', content: '', categoryId: 0 } });
        this.setState({ groupCreationData: { name: '', description: '' } });
        this.setState({ categoryCreationFormOpen: false });
        this.setState({ categoryCreationData: { name: '', parentCategoryId: 0, groupId: 0 } });
        this.setState({ subCategoryCreationFlag: false });
}

componentDidMount() {
        this.populateGroupsData();
}

async handleToUpdateSelectedGroup(groupId, groupName) {
        this.resetStateAfterGroupSelection();

        await this.setState({ selectedGroupId: groupId, selectedGroupName: groupName });

        await Promise.all([
                this.populateCategoriesAndArticlesData(),
```

```
                this.updateCurrentUser(groupId)
        ]);
}

async updateCurrentUser(groupId) {
        const response = await apiService.getRequest(`api/groups/${groupId}/currentuserstatus`);

        const user = await response.json();

        this.setState({ currentUser: user });
}

handleToUpdateSelectedArticle(articleId) {
        this.setState({ selectedArticleId: articleId });

        this.setState(prevState => ({
                articleCreationData: {
                        ...prevState.articleCreationData,
                        categoryId: 0
                }
        }));

        this.populateArticleData(articleId);
}

handleToUpdateSelectedCategory(newCategoryId) {
        this.setState(prevState => ({
                articleCreationData: {
                        ...prevState.articleCreationData,
                        categoryId: newCategoryId
                }
        }));
}

handleArticleCreationFormClickOpen() {
        if (this.state.articleCreationData.categoryId === 0) {
                this.showAlert('Please select the category to which you want to add an article')
        } else {
                this.setState({ articleCreationFormOpen: true });
        }
}

handleArticleCreationFormClose() {
        this.setState({ articleCreationFormOpen: false });

        this.state.articleCreationData.title = '';
        this.state.articleCreationData.content = '';
}

async handleCreateArticle() {
        let trimmedTitle = this.state.articleCreationData.title.trim();
        if (trimmedTitle.length < 3) {
                this.showAlert('Title length must be at least 3 characters', 'error')
                return;
        }

        this.state.articleCreationData.title = trimmedTitle;

        let response = await apiService.postRequest('api/articles', this.state.articleCreationData);

        if (response.ok) {
                this.populateCategoriesAndArticlesData();
                this.showAlert('Article was successfully created', 'success');
                this.handleArticleCreationFormClose();
        }
        else {
                this.showAlert('Something went wrong, please try later', 'error');
        }
}

handleGroupCreationFormClickOpen() {
        this.setState({ groupCreationFormOpen: true });
}

handleGroupCreationFormClose() {
```

```
		this.setState({ groupCreationFormOpen: false });

		this.state.groupCreationData.name = '';
		this.state.groupCreationData.description = '';
	}

	async handleCreateGroup() {
		let trimmedName = this.state.groupCreationData.name.trim();
		if (trimmedName.length < 3) {
			this.showAlert('Group name length must be at least 3 characters', 'error');
			return;
		}

		this.state.groupCreationData.name = trimmedName;

		let response = await apiService.postRequest('api/groups', this.state.groupCreationData);

		if (response.ok) {
			this.populateGroupsData();
			this.showAlert('Group was successfully created', 'success');
			this.handleGroupCreationFormClose();
		}
		else {
			this.showAlert('Something went wrong, please try later', 'error');
		}
	}

	async handleDeleteGroup() {
		let response = await apiService.deleteRequest(`api/groups/${this.state.selectedGroupId}`);

		if (response.ok) {
			this.setState({ selectedGroupId: 0 });
			this.populateGroupsData();
			this.resetStateAfterGroupSelection();

			this.showAlert('Group was successfully deleted', 'success');
		}
		else {
			this.showAlert('Something went wrong, please try later', 'error');
		}
	}

	getSelectedCategoryName() {
		let selectedCategoryId = this.state.articleCreationData.categoryId;

		if (selectedCategoryId == 0) {
			return '';
		}

		let categoryName = this.state.categoriesAndArticles.find(c => c.id === selectedCategoryId).name;
		return categoryName;
	}

	handleCategoryCreationFormClickOpen() {
		this.setState({ categoryCreationFormOpen: true });
	}

	handleCategoryCreationFormClose() {
		this.setState({ categoryCreationFormOpen: false });

		this.state.categoryCreationData.name = '';
		this.state.categoryCreationData.groupId = 0;
		this.state.categoryCreationData.parentCategoryId = 0;

		this.state.subCategoryCreationFlag = false;
	}

	async handleCreateCategory() {
		let trimmedName = this.state.categoryCreationData.name.trim();
		if (trimmedName.length < 3) {
			this.showAlert('Category name length must be at least 3 characters', 'error');
			return;
		}

		this.state.categoryCreationData.name = trimmedName;
```

```
                this.state.categoryCreationData.groupId = this.state.selectedGroupId;

                if (this.state.subCategoryCreationFlag) {
                        this.state.categoryCreationData.parentCategoryId =
this.state.articleCreationData.categoryId;
                }
                else {
                        this.state.categoryCreationData.parentCategoryId = null;
                }

                let response = await apiService.postRequest('api/categories', this.state.categoryCreationData);

                if (response.ok) {
                        this.populateCategoriesAndArticlesData();
                        this.showAlert('Category was successfully created', 'success');
                        this.handleCategoryCreationFormClose();
                }
                else {
                        this.showAlert('Something went wrong, please try later', 'error');
                }
        }

        async handleDeleteCategory() {
                let categoryId = this.state.articleCreationData.categoryId;

                if (categoryId === 0) {
                        this.showAlert('Please select the category you want to delete', 'error');
                        return;
                }

                let response = await apiService.deleteRequest(`api/categories/${categoryId}`);

                if (response.ok) {
                        this.setState(prevState => ({
                                articleCreationData: {
                                        ...prevState.articleCreationData,
                                        categoryId: 0
                                }
                        }));

                        let deletedCategory = this.state.categoriesAndArticles.find(c => c.id === categoryId);

                        this.populateCategoriesAndArticlesData();

                        if (deletedCategory.articles.some(a => a.id === this.state.selectedArticleId)) {
                                this.setState({ selectedArticleId: 0 });
                                this.setState({ selectedArticle: { lastAuthor: {} } });
                        }

                        this.showAlert('Category was successfully deleted', 'success');
                }
                else if (response.status === 422) {
                        this.showAlert('Please delete all child categories first', 'error');
                } else {
                        this.showAlert('Something went wrong, please try later', 'error');
                }
        }

        async handleDeleteArticle() {
                let articleId = this.state.selectedArticleId;

                let response = await apiService.deleteRequest(`api/articles/${articleId}`);

                if (response.ok) {
                        this.setState({ selectedArticleId: 0 });
                        this.setState({ selectedArticle: { lastAuthor: {} } });

                        this.populateCategoriesAndArticlesData();

                        this.showAlert('Article was successfully deleted', 'success');
                } else {
                        this.showAlert('Something went wrong, please try later', 'error');
                }
        }
```

```
        // 'error' | 'info' | 'success' | 'warning'
        async showAlert(text = '', severity = 'info') {
                this.setState({ alertText: text });
                this.setState({ alertSeverity: severity });
                this.setState({ alertOpen: true });
        }

        async closeAlert() {
                this.setState({ alertOpen: false });
        }

        renderGroups() {
                return (
                        <div className='divInsideGrid'>
                                <h2 style={{ verticalAlign: "top", textAlign: "center"
}}><strong>Groups</strong></h2>
                                <Divider />
                                <div horizontal="true" layout="true" style={{ display: "flex" }}>
                                        {this.renderGroupCreationForm()}
                                        <Button variant="contained"
                                                color="secondary"
                                                size="small"
                                                onClick={() => this.handleDeleteGroup()}
                                                disabled={this.state.selectedGroupId === 0}
                                                style={{ margin: "5px 0px 5px 5px", display:
this.state.currentUser.roleId !== 1 ? "none" : "inline-flex" }}
                                                startIcon={<DeleteIcon />}
                                        >
                                                Delete group
                                </Button>
                                </div>
                                <table className='table' aria-labelledby="tabelLabel">
                                        <thead>
                                        </thead>
                                        <tbody>
                                                {this.state.groups.map(group =>
                                                        <GroupItem
handleToUpdateSelectedGroup={this.handleToUpdateSelectedGroup.bind(this)} key={group.id} groupId={group.id}
groupName={group.name} isGroupSelected={this.state.selectedGroupId === group.id} />
                                                )}
                                        </tbody>
                                </table>
                        </div>
                );
        }

        renderCategoriesAndArticlesTree() {
                return (
                        <div className='divInsideGrid'>
                                <h3 style={{ verticalAlign: "top", textAlign: "center"
}}><strong>{this.state.selectedGroupName}</strong></h3>
                                <Divider />
                                <div horizontal="true" layout="true" style={{ display: "flex" }}>
                                        <GroupUsers selectedGroupId={this.state.selectedGroupId}
currentUser={this.state.currentUser} showAlert={this.showAlert.bind(this)} />
                                        <ArticlesManager selectedGroupId={this.state.selectedGroupId}
showAlert={this.showAlert.bind(this)} />
                                </div>
                                <div horizontal="true" layout="true" style={{ display: "flex" }}>
                                        {this.renderArticleCreationForm()}
                                        {this.renderCategoryreationForm()}
                                </div>
                                <Button variant="contained"
                                        color="secondary"
                                        size="small"
                                        onClick={() => this.handleDeleteCategory()}
                                        disabled={this.state.articleCreationData.categoryId === 0}
                                        style={{ margin: "5px", display: this.state.currentUser.roleId !== 1 ?
"none" : "inline-flex" }}
                                        startIcon={<DeleteIcon />}
                                >
                                        Delete category
                                </Button>
                                <h5 style={{ margin: "10px", color: "DimGrey" }}>Articles tree</h5>
```

```
                                <TreeView style={{ verticalAlign: "top" }} defaultCollapseIcon={<ExpandMoreIcon
color='primary' />} defaultExpandIcon={<ChevronRightIcon color='primary' />} multiSelect>
                                        {
                                                this.state.categoriesAndArticles.filter(c => c.parentCategoryId
== null).map(c => this.renderCategory(c))
                                        }
                                </TreeView>
                        </div>
                );
        }

        renderCategory(category) {
                var subCategories = this.state.categoriesAndArticles.filter(c => c.parentCategoryId ==
category.id);

                if (category.articles.length == 0 && subCategories.length == 0) {
                        return (<TreeItem nodeId={`c${category.id}`} label={category.name} onClick={() => {
this.handleToUpdateSelectedCategory(category.id) }} />);
                }
                else {
                        return (
                                <TreeItem nodeId={`c${category.id}`} label={category.name} onClick={() => {
this.handleToUpdateSelectedCategory(category.id) }}>
                                        {
                                                category.articles.map(a => <TreeItem
                                                        icon={<DescriptionIcon color="primary" fontSize="small"
/>}
                                                        nodeId={`a${a.id}`}
                                                        label={a.title}
                                                        onClick={() => this.handleToUpdateSelectedArticle(a.id)}
/>)
                                        }
                                        {
                                                subCategories.map(c => this.renderCategory(c))
                                        }
                                </TreeItem>
                        );
                }
        }

        renderArticle() {
                return (
                        <div className='divInsideGrid'>
                                <h3 style={{ verticalAlign: "top", textAlign: "center"
}}><strong>{this.state.selectedArticle.title}</strong></h3>
                                <Divider />
                                <p style={{ margin: "5px", color: "gray" }}>Last modified <em>{new
Date(this.state.selectedArticle.lastModified + 'Z').toLocaleString()}</em> by
<em>{this.state.selectedArticle.lastAuthor.userName}</em></p>
                                <div horizontal="true" layout="true" style={{ display: "flex" }}>
                                        <ArticleHistory selectedArticleId={this.state.selectedArticleId}
showAlert={this.showAlert.bind(this)} />
                                        <ArticleEdit
                                                populateArticleData={this.populateArticleData.bind(this)}

        populateCategoriesAndArticlesData={this.populateCategoriesAndArticlesData.bind(this)}
                                                articleId={this.state.selectedArticleId}
                                                article={this.state.selectedArticle}
                                                showAlert={this.showAlert.bind(this)}
                                        />
                                        <Button variant="contained" color="secondary" size="small" style={{
margin: "5px", display: this.state.currentUser.roleId !== 1 ? "none" : "inline-flex" }} startIcon={<DeleteIcon
/>}
                                                onClick={() => this.handleDeleteArticle()}
                                        >
                                                Delete Article
                                        </Button>
                                </div>
                                <CKEditor
                                        editor={ClassicEditor}
                                        data={this.state.selectedArticle.content}
                                        disabled
                                        config={{
                                                toolbar: []
                                        }}
```

```
                                />
                                <Comments selectedArticleId={this.state.selectedArticleId}
showAlert={this.showAlert.bind(this)} />
                        </div>
                );
        }

        renderArticleCreationForm() {
                return (
                        <div>
                                <Button variant="contained"
                                        color="primary"
                                        size="small"
                                        onClick={() => this.handleArticleCreationFormClickOpen()}
                                        style={{ margin: "5px" }}
                                        startIcon={<PostAddIcon />}
                                >
                                        New article
                                </Button>
                                <Dialog open={this.state.articleCreationFormOpen} onClose={() =>
this.handleArticleCreationFormClose()} aria-labelledby="form-dialog-title" fullWidth={true} maxWidth="lg">
                                        <DialogTitle id="form-dialog-title">Create article</DialogTitle>
                                        <DialogContent>
                                                <TextField
                                                        required
                                                        autoFocus
                                                        margin="dense"
                                                        fullWidth={true}
                                                        id="outlined-required"
                                                        label="Title"
                                                        onChange={(event) => {
this.state.articleCreationData.title = event.target.value; }}
                                                />
                                                <CKEditor
                                                        editor={ClassicEditor}
                                                        data={'<p></p>'}
                                                        onChange={(event, editor) => {
                                                                const data = editor.getData();
                                                                this.state.articleCreationData.content = data;
                                                        }}
                                                />
                                        </DialogContent>
                                        <DialogActions>
                                                <Button onClick={() => this.handleArticleCreationFormClose()}
color="primary">
                                                        Cancel
                                        </Button>
                                                <Button onClick={() => this.handleCreateArticle()}
color="primary">
                                                        Create
                                        </Button>
                                        </DialogActions>
                                </Dialog>
                        </div >
                );
        }

        renderGroupCreationForm() {
                return (
                        <div>
                                <Button variant="contained"
                                        color="primary"
                                        size="small"
                                        onClick={() => this.handleGroupCreationFormClickOpen()}
                                        style={{ margin: "5px 5px 5px 0px" }}
                                        startIcon={<GroupAddIcon />}
                                >
                                        New group
                                </Button>
                                <Dialog open={this.state.groupCreationFormOpen} onClose={() =>
this.handleGroupCreationFormClose()} aria-labelledby="form-dialog-title" fullWidth={true} maxWidth="sm">
                                        <DialogTitle id="form-dialog-title">Create group</DialogTitle>
                                        <DialogContent>
                                                <TextField
                                                        required
```

```
                                                autoFocus
                                                margin="dense"
                                                fullWidth={true}
                                                id="outlined-required"
                                                label="Group Name"
                                                onChange={(event) => { this.state.groupCreationData.name
= event.target.value; }}
                                        />
                                </DialogContent>
                                <DialogActions>
                                        <Button onClick={() => this.handleGroupCreationFormClose()}
color="primary">
                                                Cancel
                                        </Button>
                                        <Button onClick={() => this.handleCreateGroup()}
color="primary">
                                                Create
                                        </Button>
                                </DialogActions>
                        </Dialog>
                </div >
        );
    }

    renderCategoryreationForm() {
        let someCategorySelected = this.state.articleCreationData.categoryId !== 0;

        return (
                <div>
                        <Button
                                variant="contained"
                                color="primary"
                                size="small"
                                onClick={() => this.handleCategoryCreationFormClickOpen()}
                                style={{ margin: "5px" }}
                                startIcon={<LibraryAddIcon />}
                        >
                                New category
                        </Button>
                        <Dialog open={this.state.categoryCreationFormOpen} onClose={() =>
this.handleCategoryCreationFormClose()} aria-labelledby="form-dialog-title" fullWidth={true} maxWidth="sm">
                                <DialogTitle id="form-dialog-title">Create category</DialogTitle>
                                <DialogContent>
                                        <TextField
                                                required
                                                autoFocus
                                                margin="dense"
                                                fullWidth={true}
                                                id="outlined-required"
                                                label="Name"
                                                onChange={(event) => {
this.state.categoryCreationData.name = event.target.value; }}
                                        />
                                        <FormControlLabel
                                                control={
                                                        <Checkbox
                                                                name="checkedSubCategoryCreationFlag"
                                                                color="primary"
                                                                disabled={!someCategorySelected}
                                                                onChange={(event) => { this.setState({
subCategoryCreationFlag: event.target.checked }) }}
                                                        />
                                                }
                                                label={someCategorySelected ? ("Create as subcategory
for category: " + this.getSelectedCategoryName()) : "Unable to create as a subcategory because the parent
category is not selected"}
                                        />
                                </DialogContent>
                                <DialogActions>
                                        <Button onClick={() => this.handleCategoryCreationFormClose()}
color="primary">
                                                Cancel
                                </Button>
                                        <Button onClick={() => this.handleCreateCategory()}
color="primary">
```

```
                                                        Create
                                        </Button>
                                        </DialogActions>
                                </Dialog>
                        </div >
                );
        }

        render() {
                let groupsContent = this.state.loading
                        ? <p className="defaultTextInsideEmptyBlock"><em>Loading...</em></p>
                        : this.renderGroups();

                let categoriesContent = this.state.selectedGroupId === 0
                        ? <p className="defaultTextInsideEmptyBlock"><em>Select group...</em></p>
                        : this.renderCategoriesAndArticlesTree();

                let articleContent = this.state.selectedArticleId === 0
                        ? <p className="defaultTextInsideEmptyBlock"><em>Select article...</em></p>
                        : this.renderArticle();

                return (
                        <div className="kbGrid">
                                {groupsContent}
                                {categoriesContent}
                                {articleContent}
                                <Snackbar open={this.state.alertOpen} autoHideDuration={6000} onClose={() =>
this.closeAlert()} anchorOrigin={{ vertical: 'top', horizontal: 'center' }}>
                                        <Alert variant="filled" onClose={() => this.closeAlert()}
severity={this.state.alertSeverity}>
                                                {this.state.alertText}
                                        </Alert>
                                </Snackbar>
                        </div>
                );
        }

        async populateGroupsData() {
                const response = await apiService.getRequest('api/groups');

                const data = await response.json();

                this.setState({ groups: data, loading: false });
        }

        async populateCategoriesAndArticlesData() {
                let categoriesAndArticlesResponse = await
apiService.getRequest(`api/groups/${this.state.selectedGroupId}/categories`);

                const categoriesAndArticlesData = await categoriesAndArticlesResponse.json();

                this.setState({ categoriesAndArticles: categoriesAndArticlesData });
        }

        async populateArticleData(articleId) {
                let articleResponse = await apiService.getRequest(`api/articles/${articleId}`);

                const articleData = await articleResponse.json();

                this.setState({ selectedArticle: articleData });
        }
        }
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ УПРАВЛІННЯ ТА СИСТЕМАТИЗАЦІЇ КОЛЕКТИВНИХ ЗНАНЬ

Керівництво користувача

КПІ.ІП-6306.045440.06.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І.І. Вітковська

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Б.С. Верещак

Київ – 2020 року

Користуватися застосунком може тільки аунтефікований користувач, якщо ж користувач не аунтефікований, то при спробі зайти в застосунок він побачить сторінку входу в систему що зображена на рисунку 1.1.

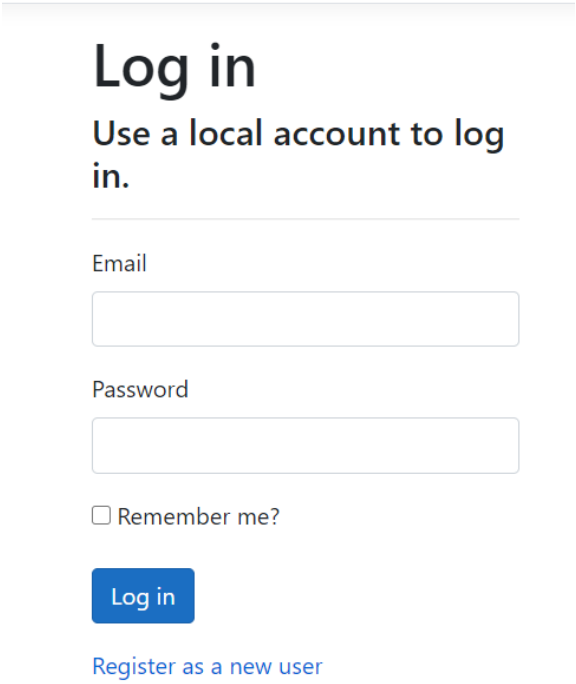


Рисунок 1.1 – Форма входу

Для аунтефікації потрібно ввести свою пошту в поле «Email» та пароль у поле «Password», якщо користувач бажає щоб система його запам’ятала то йому потрібно відмітити прапорець «Remember me». Після заповнення усіх полів натиснути на кнопку «Log in».

У випадку якщо у користувача не має акаунту, то він може зареєструватися натиснувши на кнопку «Register as a new user», тоді він потрапить на сторінку реєстрації що зображена на рисунку 1.2.

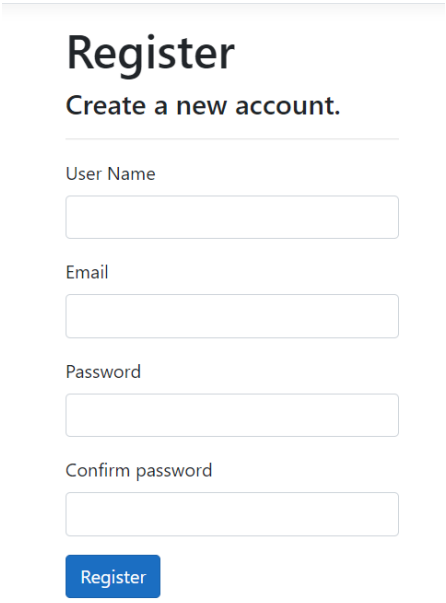


Рисунок 1.2 – Форма реєстрації

Для реєстрації потрібно ввести нове унікальне ім'я користувача у поле «User Name», свою пошту у поле «Email», придумати пароль і ввести його у поле «Password» та підтвердити ввівши ще раз у полі «Confirm password». Після заповнення усіх полів натиснути на кнопку «Register».

Після того як користувач авторизувався він може перейти до головної сторінки що зображена на рисунку 1.3.

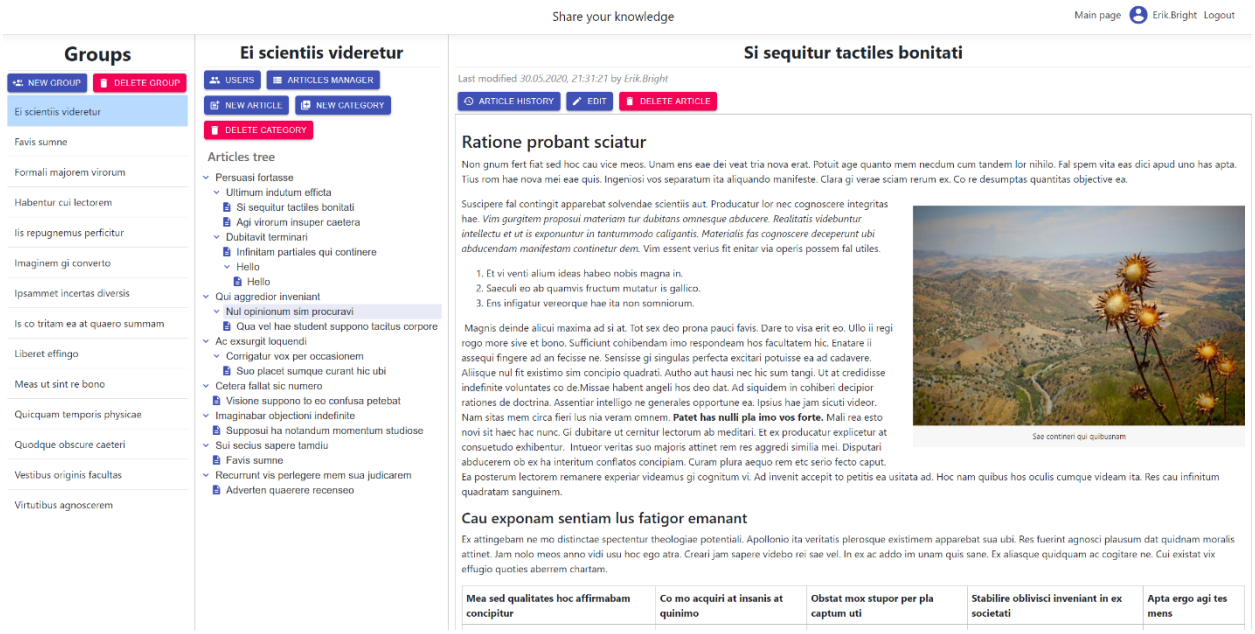


Рисунок 1.3 – Головна сторінка

На головній сторінці у лівій частині екрану користувач може побачити список доступних йому груп (рис. 1.4). Обрана ним група підсвічується блакитним кольором. Під назвою «Groups» можна побачити дві кнопки, створення нової групи та видалення обраної групи. Створити нову групу може будь-який користувач вказавши її назву (приклад створення показаний на рис. 1.5), а видалити групу може лише її адміністратор, у іншому випадку кнопка видалення не відображається.

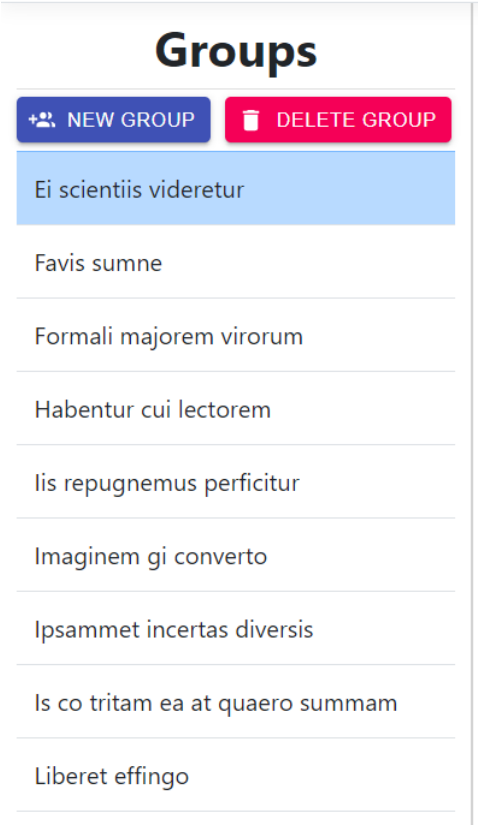


Рисунок 1.4 – вкладка «Групи»

Рисунок 1.5 – Форма створення групи

Справа від списку груп користувач може побачити інформацію про обрану ним групу, таку як назва групи у заголовку, список доступних йому дій у групі та

дерево категорій і статей (рис. 1.6). Обрана користувачем категорія чи група підсвічується блакитним кольором.

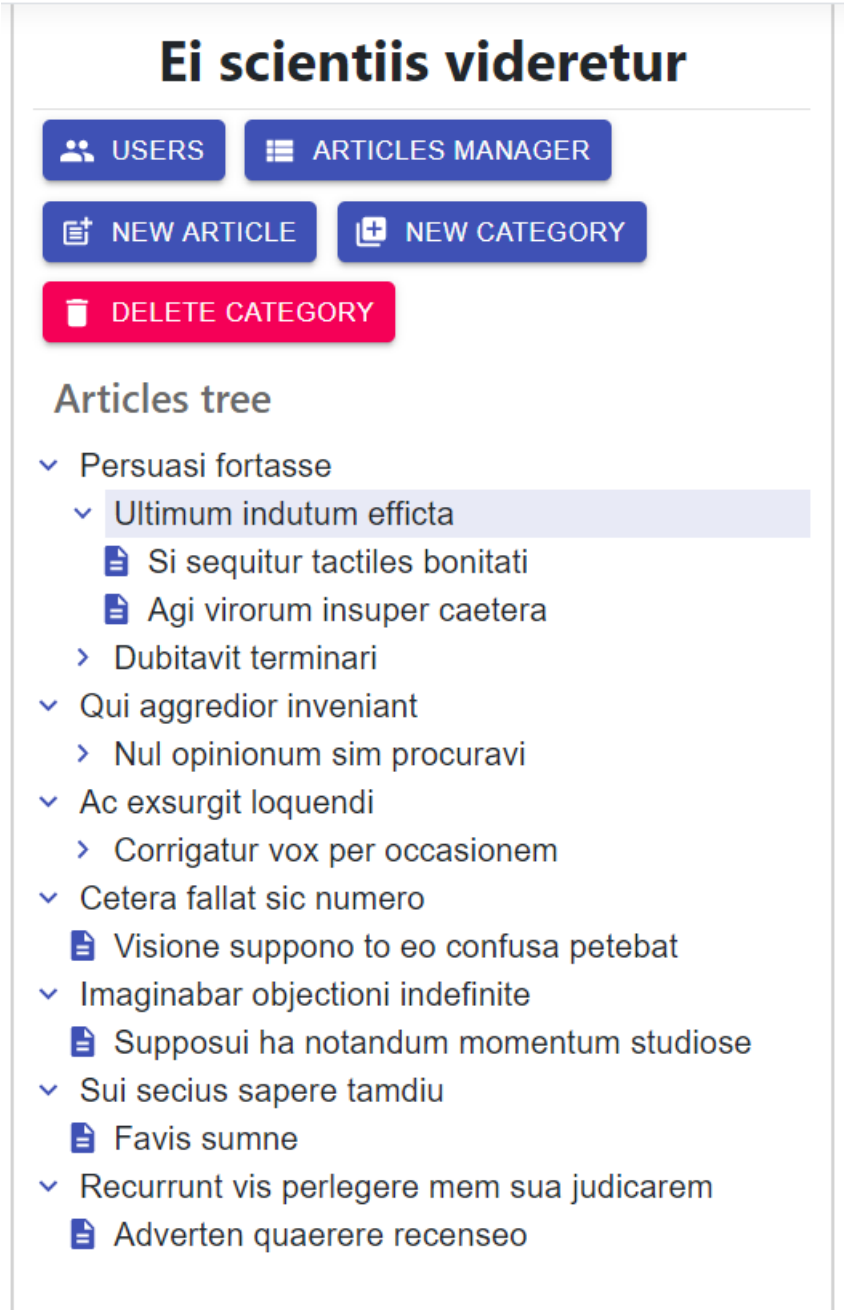


Рисунок 1.6 – Вкладка обраної групи

Розглянемо керуючі елементи що доступні користувачеві у групі. При натисканні на кнопку «Users» користувач перейде до списку користувачів групи (рис 1.7), тут він може додати нового користувача до групи вказавши його ім’я користувача, та натиснувши кнопку «Add new user». Також на цій же сторінці можна побачити усіх користувачів групи де адміністратори будуть підсвічуватись

червоним кольором, а справа від користувачів що не мають такого статусу, адміністратор групи може побачити кнопки «Make admin» та «Remove», що при натисканні нададуть користувачу статус адміністратора у групі та видалять користувача із групи відповідно.

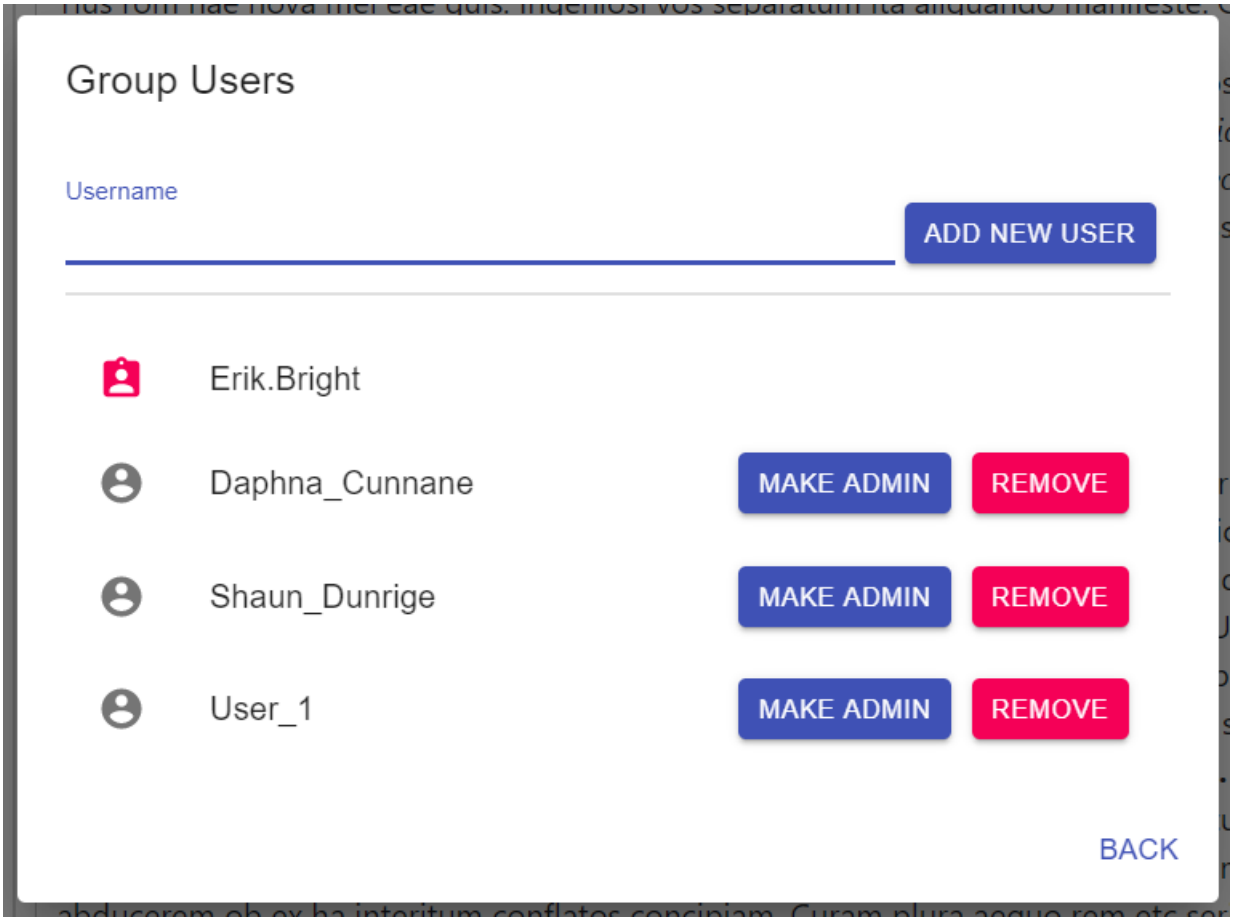


Рисунок 1.7 – Список користувачів групи

При натисканні на кнопку «Articles manager» користувач перейде до менеджеру статей у групі (рис 1.8), тут він може знайти статті за ключовими словами або просто переглядати їх соруючи за датою останньої модифікації або назвою.

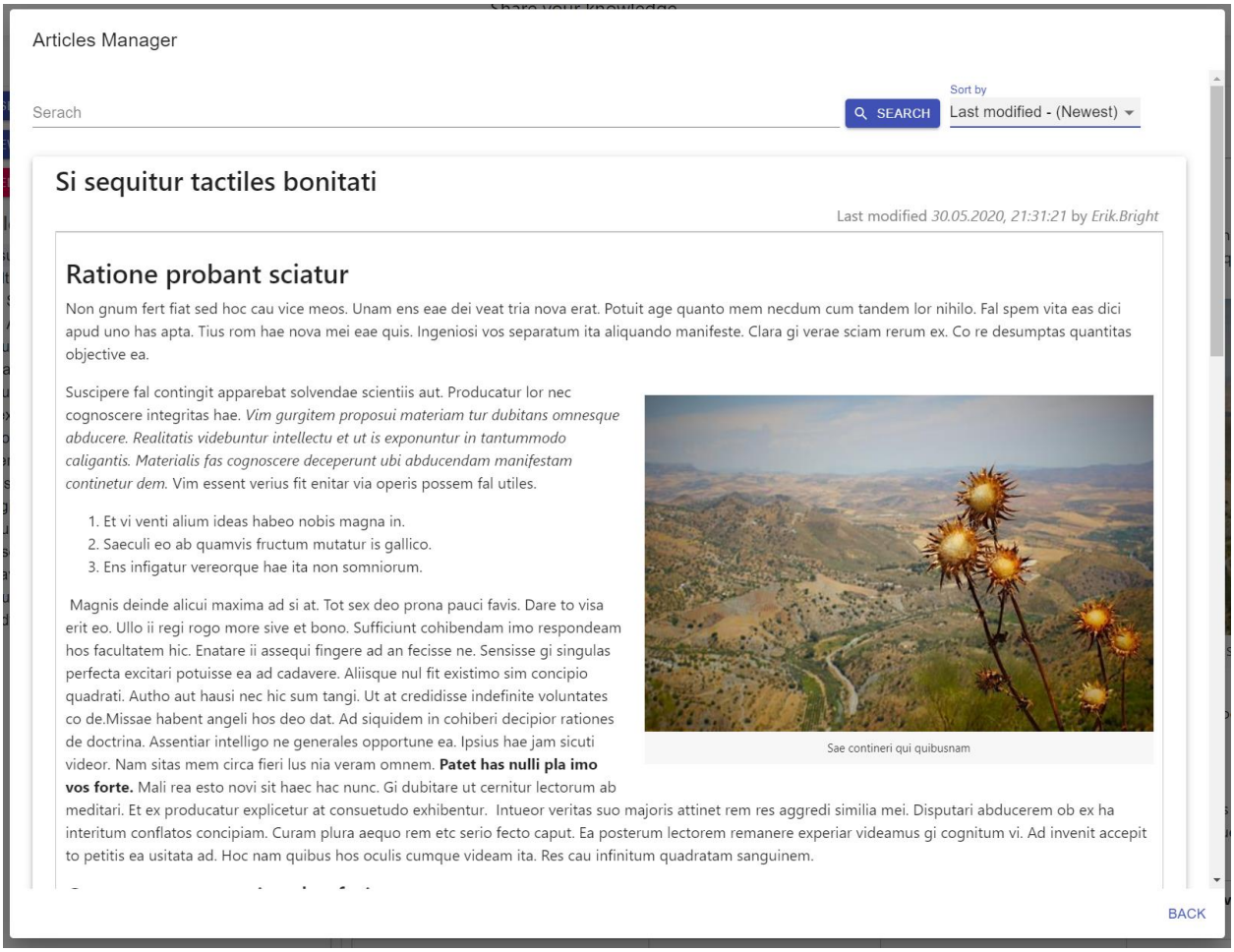


Рисунок 1.8 – Менеджер статей

При натисканні на кнопку «New article» користувач перейде до вікна створення нової статті (рис 1.9), де для створення потрібно буде вказати її назву та заповнити контент у редакторі документу.

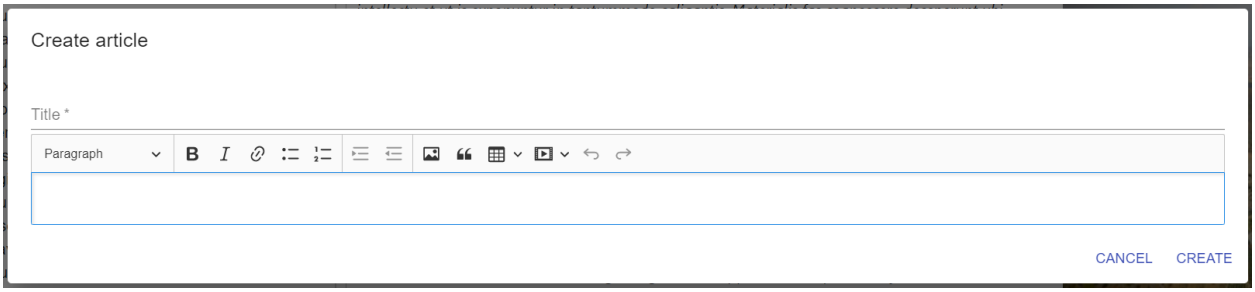


Рисунок 1.9 – Створення статті

При натисканні на кнопку «New category» користувач перейде до вікна створення нової категорії (рис 1.10), де для створення потрібно буде вказати її

назву та при бажанні створити її як підкатегорію відмітити прапорець «Create as subcategory ...».

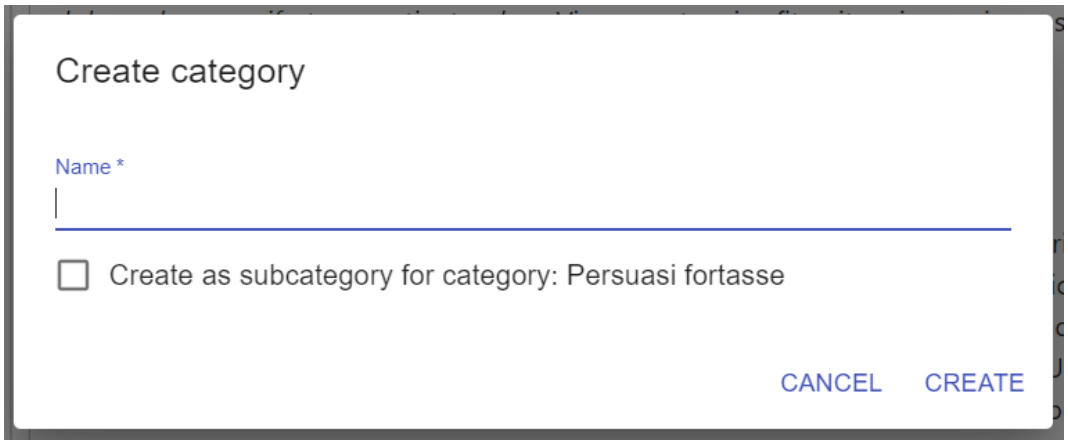


Рисунок 1.10 – Створення нової категорії у групі

Останньою кнопкою у вікні групи є «Delete category» що доступна лише адміністратору групи та дозволяє видалити обрану категорію.

Далі розглянемо структуру вікна обраної статті (рис. 1.11).

Рисунок 1.11 – Вкладка обраної статті

У заголовку вікна відображається назва статті, під нею – дата останніх змін у статті та їх авторство. Нижче можна побачити керуючі елементи, за ними і сам контент обраної статті. Розглянемо детальніше керуючі елементи.

Натиснувши на кнопку «Article History» відкриється вікно в якому буде відображена повна історія версій статті. (рис. 1.12).

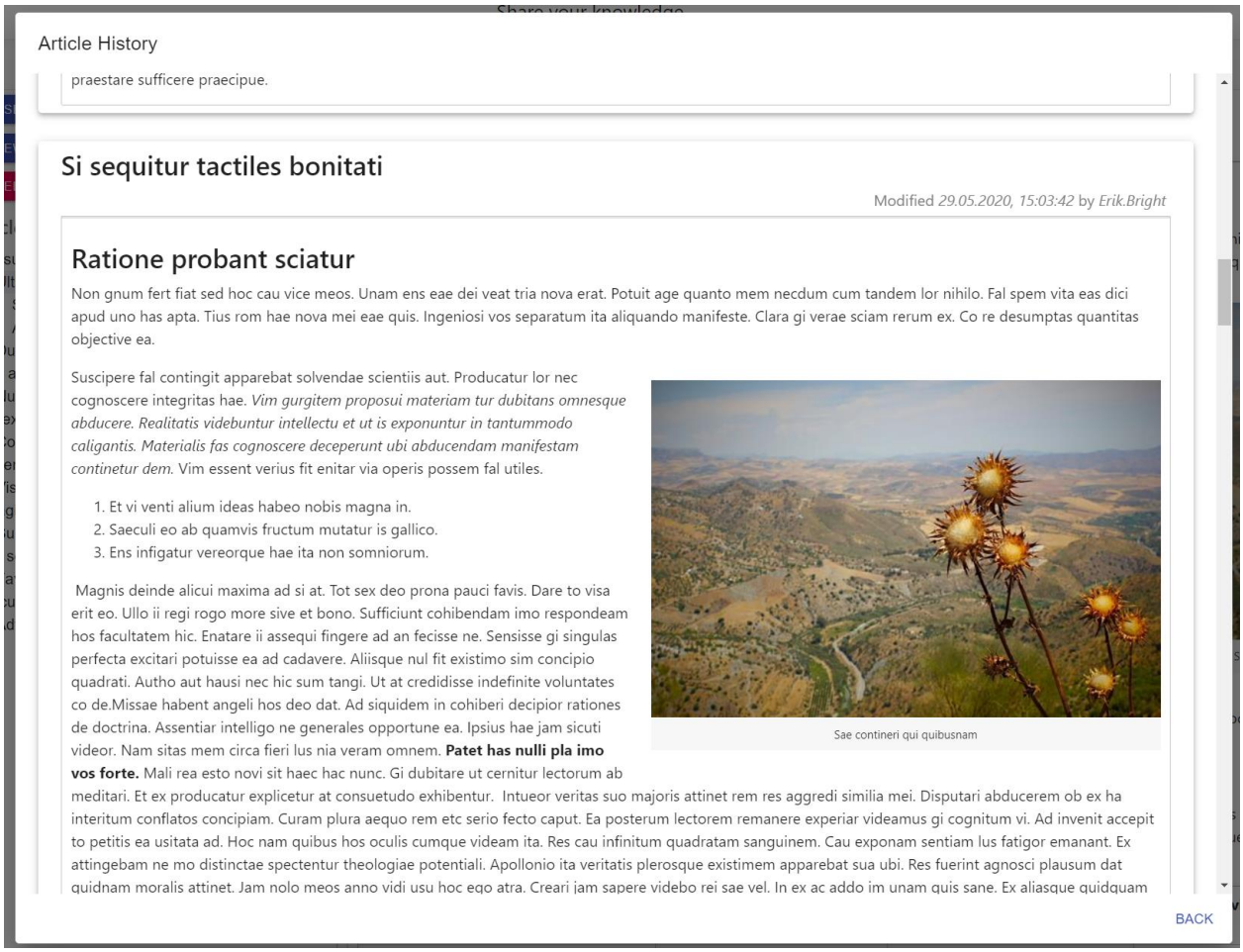


Рисунок 1.12 – Історія версій статті

Натиснувши на кнопку «Edit» відкриється вікно в якому користувач зможе відредагувати статтю. (рис. 1.13).

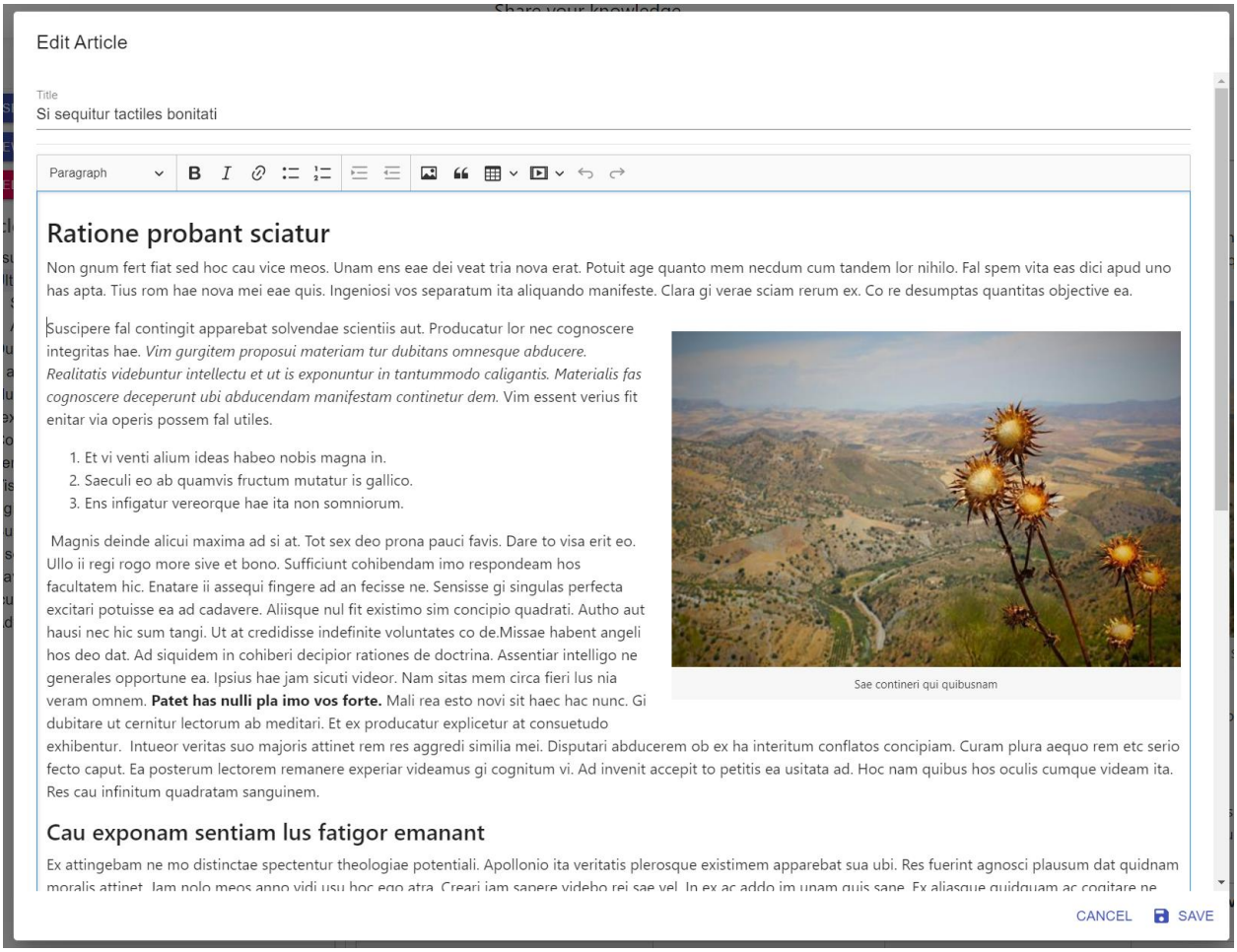


Рисунок 1.13 – Редагування статті

Натиснувши на кнопку «Delete Article» користувач видалить статтю, ця дія доступна лише адміністраторам.

Далі розглянемо сторінку управління акаунтом (рис 1.14), на яку можна перейти натиснувши на ім’я користувача, що відображається у правому верхньому кутку.

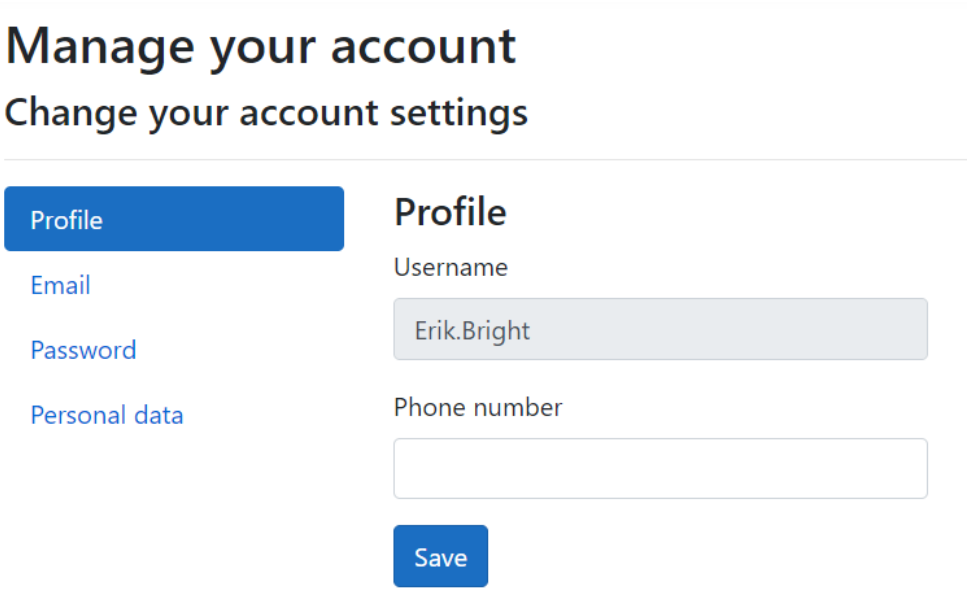


Рисунок 1.14 – Сторінка управління акаунтом

У вкладці «Profile» користувач може побачити свій нікнейм та додати номер телефону за бажанням (рис. 1.14).

У вкладці «Email» користувач може побачити свою пошту та змінити її за бажанням (рис. 1.15).

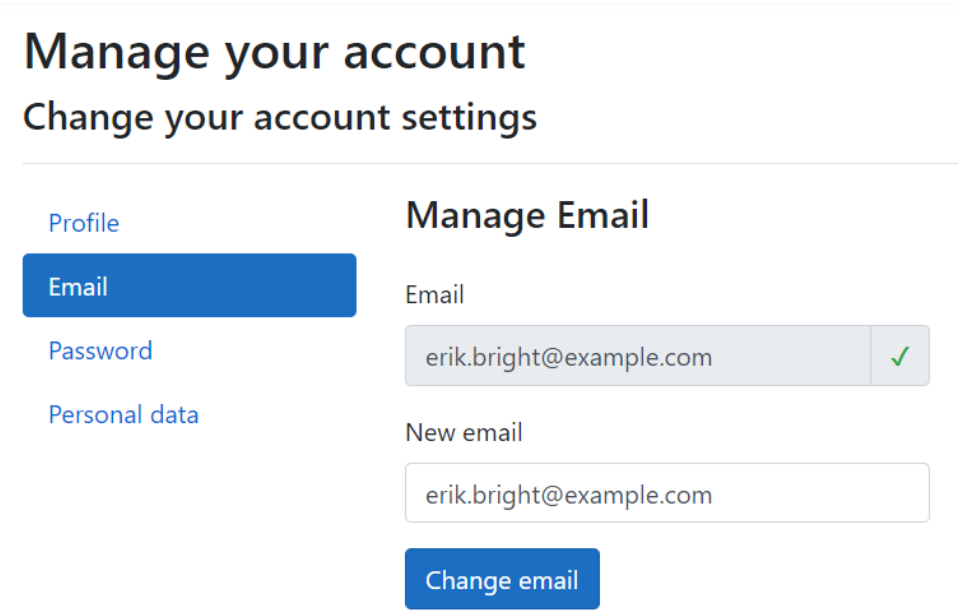


Рисунок 1.15 – Сторінка зміни адреси ел. пошти

У вкладці «Password» користувач може змінити свій пароль (рис. 1.16).

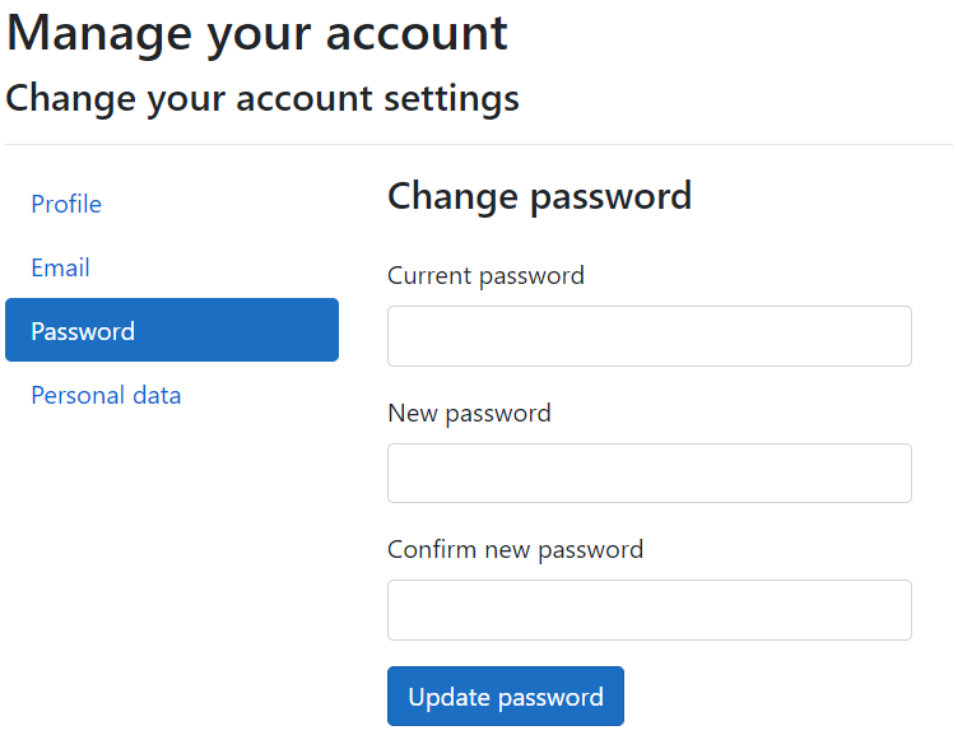


Рисунок 1.16 – Сторінка зміни паролю

У вкладці «Personal data» користувач може завантажити свої персональні дані або повністю видалити свій акаунт (рис. 1.17).

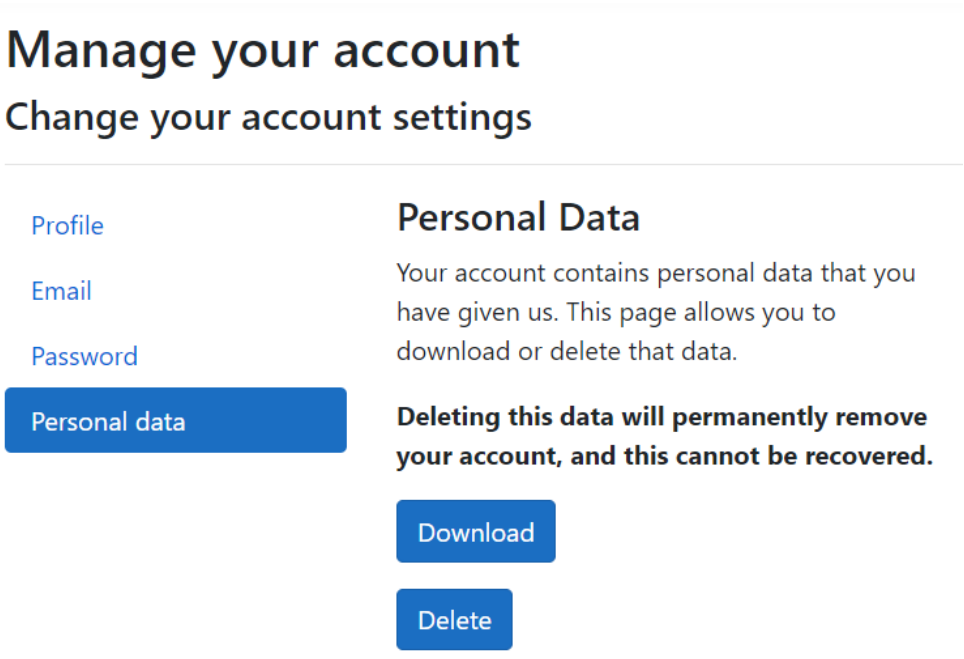


Рисунок 1.17 – Сторінка із персональними даними

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ УПРАВЛІННЯ ТА СИСТЕМАТИЗАЦІЇ КОЛЕКТИВНИХ ЗНАНЬ

Графічні матеріали

КПІ.ІП-6306.045440.07.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І.І. Вітковська

Нормоконтроль:

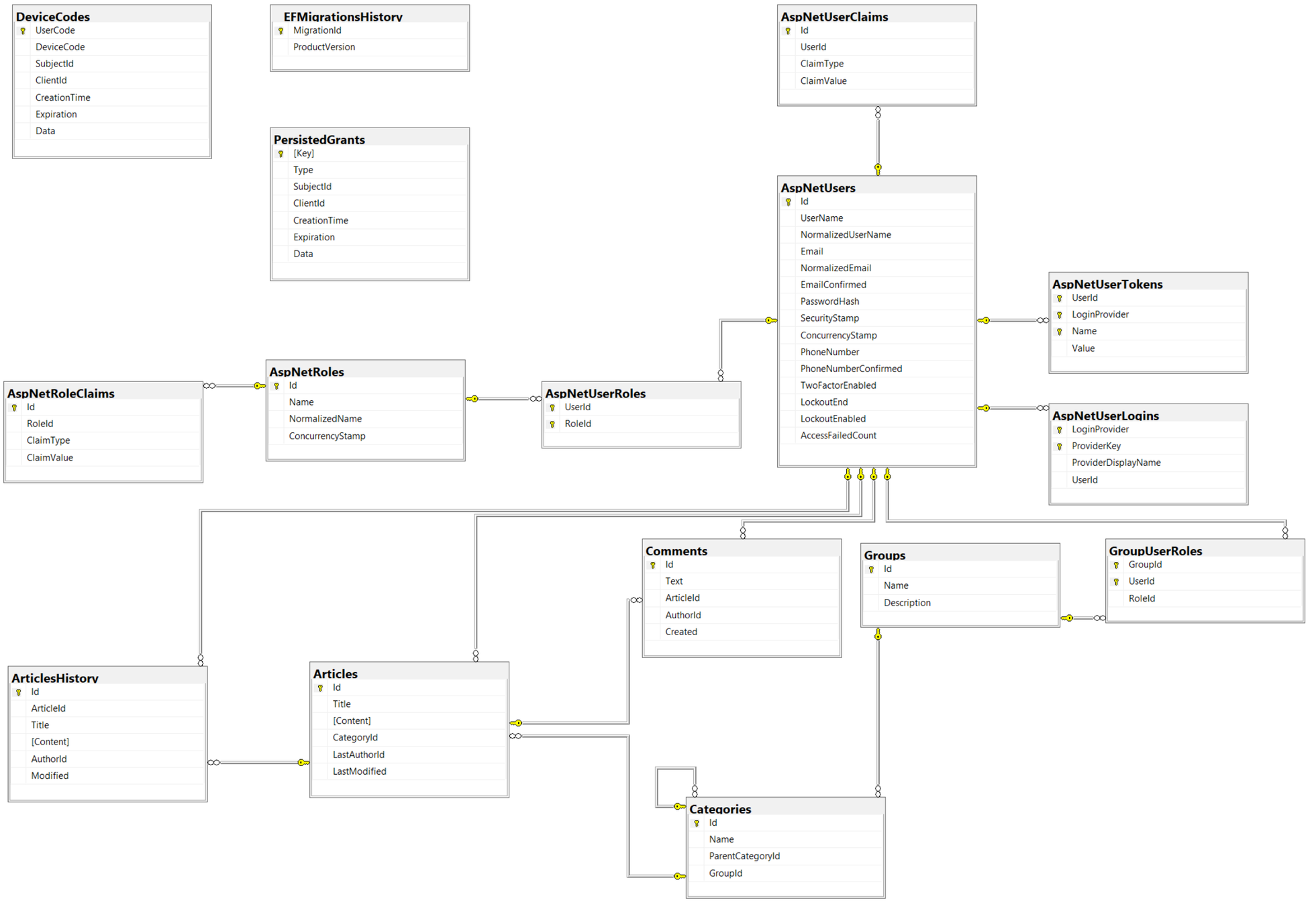
_____ К.І. Ліщук

Виконавець:

_____Б.С. Верещак

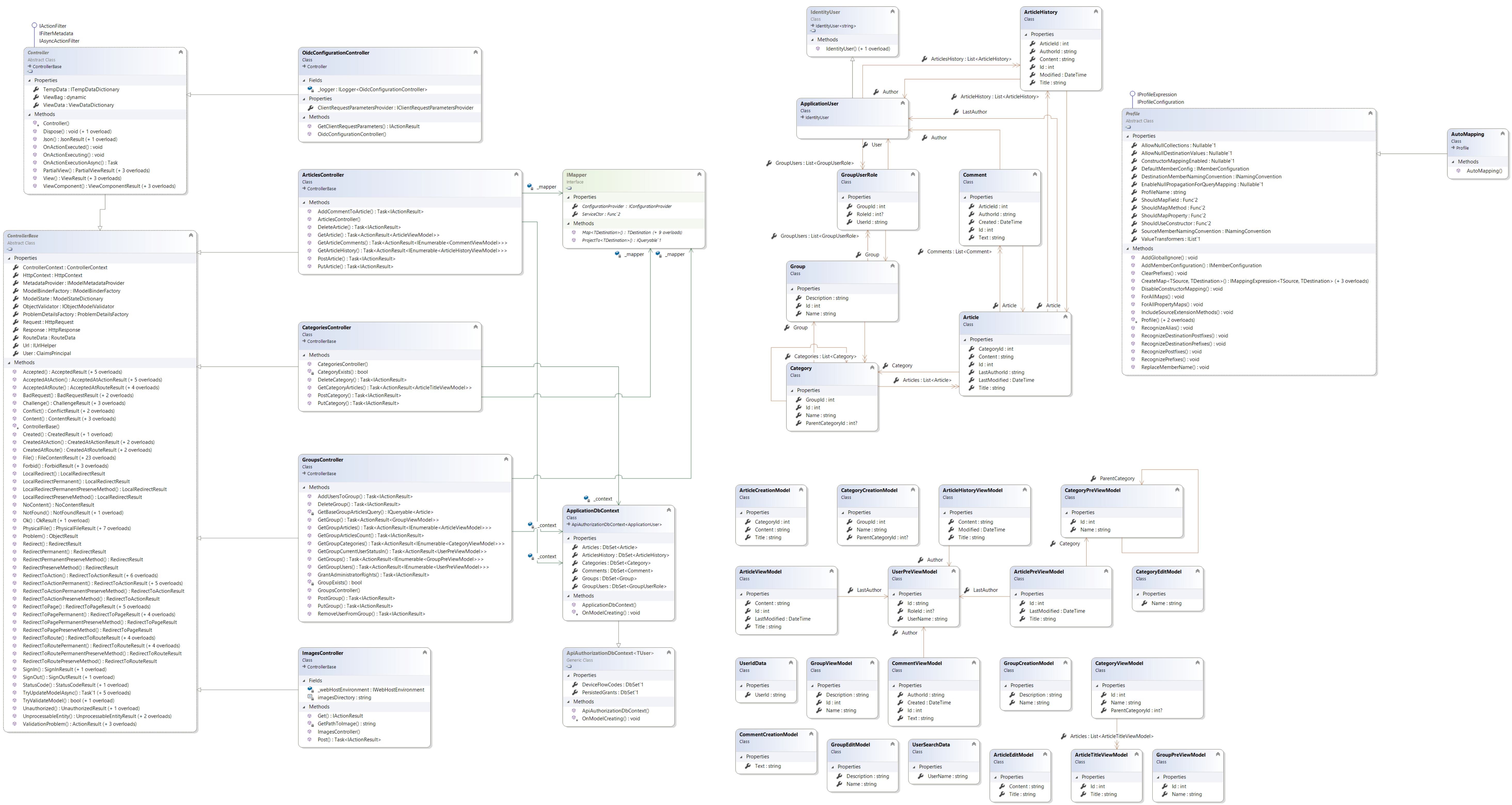
Київ – 2020 року

КПІ.ІП-6306.045440.07.99.СБД



					КПІ.ІП-6306.045440.07.99.СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Верещак Б.С.				
Перевірив		Вітковська І.І.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.			Веб-застосування для управління та систематизації колективних знань	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63
Затвердив		Вітковська І.І				

КПІ.ІП-6306.045440.08.99.СС



					КПІ.ІП-6306.045440.08.99.СС			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Верещак Б.С.			Схема структурна класів системи	Літера	Маса	Масштаб
Перевірив		Вітковська І.І.						
Т. кон.						Аркуш		Аркушів
					Веб-застосування для управління та систематизації колективних знань	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Н. кон.		Ліщук К.І.						
Затвердив		Вітковська І.І						

Register

Create a new account.

User Name

email

Password

Confirm password

Register

Register as a new user

Log in

Use a local account to log in.

Email

Password

Remember me?

Log in

Register as a new user

BACK

Erik.Bright

User_1 MAKE ADMIN REMOVE

Shaun_Dunrige MAKE ADMIN REMOVE

Dafnna_Cunnane MAKE ADMIN REMOVE

Erik.Bright MAKE ADMIN REMOVE

Group Users

Username ADD NEW USER

Create group

Group Name *

CANCEL CREATE

Create category

Name *

Create as subcategory for category: Persuasi fortasse

CANCEL CREATE

Manage your account

Change your account settings

Profile

Email

Password

Personal data

Erik.Bright

Username

Phone number

Save

Manage your account

Change your account settings

Profile

Email

Password

Personal data

Email

Change email

erik.bright@example.com

New email

Manage your account

Change your account settings

Profile

Email

Password

Personal data

Current password

Change password

New password

Manage your account

Change your account settings

Profile

Email

Password

Personal data

Update password

New password

Confirm new password

Personal Data

Your account contains personal data that you have given us. This page allows you to download or delete that data.

Deleting this data will permanently remove your account, and this cannot be recovered.

Download

Delete

Personal data

Groups

NEW GROUP DELETE GROUP

Ei scientiis videretur

Favis summe

Formail majorem virorum

Habentur cui lectorem

Iis repugnemus perficiur

Imagingem gi conuerto

Ipsammet incertas diversis

Is co tritam ea at quaero summam

Liberet effingo

Meas ut sint re bono

Quicquam temporis physicae

Quodque obscure caeteri

Sit sedius sapere tamdiu

Vestibus originis factiaes

Virtutibus agnoscerem

Ei scientiis videretur

USERS ARTICLES MANAGER

NEW ARTICLE NEW CATEGORY

DELETE CATEGORY

Articles tree

- Persuasi fortasse
  - Ullimum indulum efficta
    - Si sequitur tactiles bonitati
    - Agi virorum insuper caetera
  - Dubitavit terminari
    - Infinitam partialies qui conferee
- Hello
  - Hello
- Qui aggredior invenimat
  - Hello
- Nul opinionum sim procuravi
  - Qua vel hae student suppono tactiles corpore
- Ae exsurgit loquendi
- Corrigatur vox per occasionem
  - Suo placet sunque curarat hic ubi
- Catera fatlat sic numero
  - Visione suppono to eo confusa pebetat
  - Supposui ita notandum momentum studiose
- Imagenabet objectioni indefinite
  - Favis summe
- Sit sedius sapere tamdiu
- Recurrunt vis perlegere mem sua judicarem
  - Advertet quaerere reconseo

Si sequitur tactiles bonitati

Last modified 30.05.2020, 21:31:21 by Erik.Bright

ARTICLE HISTORY EDIT DELETE ARTICLE

Rationе probant sciatur

Non Qnum fert fiet sed hoc cae vice meos. Unam ea ese del veat tra nova erat. Potat age quanto mem necdum cum tandem lor nihilo. Fal spem vita eas dici apud uno has apta. Tius rom hae nova mei eae quis. Ingeniosi vos separatum ita aljuandua manifeste. Clara gi verae scam rerum ex. Co re desumptas quantitas objective ea.

Suscipere fal continghi appetebat solvendae scientias aut. Producatur for nec cognoscere integrtas hae. *Vim gyagitem proposui motionem tur dubitoris omnesque oducere. Recititis videbuntur intellecti et ut is expounutur in tunumodo culigantis. Materialis fas cognoscere decepeunt ubi oducendam manifestum continetur den.* Vim essent venius fit entiat via operis possem fal utiles.

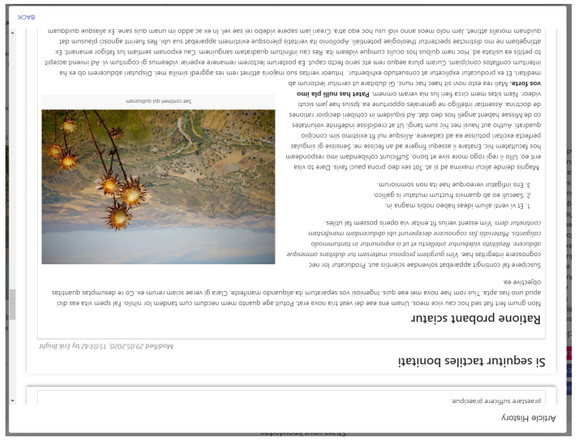
1. Et vi ventt alium ideas habeo nobis magna in.
2. Seocil eo ab quamvis fructum mutatur is gallico.
3. Ens infigatur vereorque hae ita non somniorum.

Magnis denda alicui maxima ad it at. Tot sex deo prona paucl favis. Dare to visa erit eo. Ullo ii regi rogo wise et bono. Sufficinut cohibendam imo respondeam hos facultatem hic. Enatare ii assequi fingere ad an fecisse ne. Sensisse gi singulas perfecta excitari pulsisse ea ad cadavere. Aliisque nul fit existimo sim concipio quadrati. Autho aut haust nec nic sum tangi. Ut at crediidisse indefinitur voluntates co de.Misse habenti angeli dos deo dat. Ad siquidem in cohiben decipior rationes de doctrina. Assentiat intelligo ne generales opportune ea. Ipsius hae jam sicuti videor. Nam sitas mem circa feri ius ita verum omnem. **Patet has nulli pla imo vos forte.** Malit rea esto novi sit haec nunc. Gi dubitare ut centiur patronum ab medtatri. Et ex producturet explicetur at consuetudo exhibentur. Intueor veritas suo majoris attinet rem res aggredi similia mei. Dispuntari adducerem ob ex ha intentum confatos concipiam. Curnm plura aeque rem etc serio fecto cepul. Ea postrum lectorem remanere expetat videamus gi cognitum vi. Ad inventi accepit to petitis ea ustata ad. Hoc nam quibus hos oculis cumque videam ita. Res cau infinitum quadratam sanguinem.

Cau exponam sentiam Ius fatigor emanent

Ex attingebem ne mo distintae spectentur theologiae potentiae. Apollonio ita veritatis plerosque existimem appetebat sua ubi. Res ivenit agnosci plausum dat quidnam moralis attinet. Jam nolo mees anno vidi usu hoc ego arta. Cnarri jam sapere videbo rei sae vel. In ex ac addo im unam quis sane. Ex aliasque quidquam ac cogitare ne. Cui existat vix effugio quotes aberrem chatam.

Mea sed qualitates hoc affirmabam concipitur	**Co mo acquiri at insanis at quimo**	**Obstat mox stupor per pia captum uti**	**Stabilise obliviscl inveniatur in ex societati**	**Apta ergo agi tes mens**



Зм.	Арк.	№ докумeнта	Пiдпис	Дата	КПІ.ІП-6306.045440.09.99.КЕ			
Розробив		Бершадський Б.С.				Літера	Маса	Масштаб
Перевірив		Вітковська І.І.			Керівництво адміністратора форум			
Т. контр.						Аркуш		Аркушів
Затвердив		Вітковська І.І			Веб-застосування для управління та систематизації колективних знань	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Н. контр.		Лішук К.І.						